



Guía para desarrolladores

AWS IoT Events



AWS IoT Events: Guía para desarrolladores

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Las marcas registradas y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna forma que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es AWS IoT Events?	1
Beneficios y características	1
Casos de uso	3
Supervise y mantenga los dispositivos remotos	3
Administre robots industriales	3
Rastrea los sistemas de automatización de edificios	3
Configuración	4
Configuración de una Cuenta de AWS	4
Inscríbase en un Cuenta de AWS	4
Creación de un usuario con acceso administrativo	5
Configurar permisos para AWS IoT Events	6
Permisos de acción	7
Protección de los datos de entrada	9
Política de funciones de CloudWatch registro de Amazon	9
Política de funciones SNS de mensajería de Amazon	11
Introducción	13
Requisitos previos	15
Crear una entrada	15
Crear un archivo JSON de entrada	16
Cree y configure una entrada	16
Cree una entrada dentro del modelo de detector	17
Crear un modelo de detector	17
Enviar entradas para probar el modelo de detector	25
Prácticas recomendadas	29
Habilite el CloudWatch registro de Amazon al desarrollar modelos AWS IoT Events de detectores	29
Publique periódicamente para guardar su modelo de detector cuando trabaje en la AWS IoT Events consola	30
Tutoriales	31
Uso AWS IoT Events para monitorear sus dispositivos de IoT	31
¿Cómo puede saber qué estados necesita en un modelo de detector?	33
¿Cómo puede saber si necesita una instancia, o varias, de un detector?	34
step-by-step Ejemplo sencillo	35
Creación de una entrada para capturar datos del dispositivo	37

Creación de un modelo de detector para representar los estados de los dispositivos	38
Envío de mensajes como entradas a un detector	41
Restricciones y limitaciones del modelo de detector	45
Un ejemplo comentado: el control de temperatura HVAC	49
Introducción	49
Definiciones de entrada para modelos de detectores	49
Crear una definición de modelo de detector	53
Se usa BatchUpdateDetector para actualizar	73
BatchPutMessageÚselo para entradas	75
Ingiera mensajes MQTT	77
Generar SNS mensajes de Amazon	79
Configure el DescribeDetector API	80
Usa el motor de AWS IoT Core reglas	82
Acciones admitidas	85
Usa acciones integradas	86
Establecer la acción del temporizador	86
Restablecer la acción de temporizador	87
Eliminar la acción del temporizador	87
Establecer una acción de variable	87
Trabajando con otros AWS servicios	88
AWS IoT Core	89
AWS IoT Events	90
AWS IoT SiteWise	91
Amazon DynamoDB	94
Amazon DynamoDB(v2)	96
Amazon Data Firehose	97
AWS Lambda	98
Amazon Simple Notification Service	99
Amazon Simple Queue Service	100
Expressions	103
Sintaxis para filtrar los datos del dispositivo y definir acciones	103
Literales	103
Operadores	103
Funciones para usar en expresiones	105
Referencia para las entradas y variables de las expresiones	110
Plantillas de sustitución	112

Uso	113
Escribir AWS IoT Events expresiones	113
Ejemplos de modelos de detectores	116
HVACcontrol de temperatura	116
Antecedentes	116
Definiciones de entradas	117
Definición del modelo de detector	119
BatchPutMessage ejemplos	137
BatchUpdateDetector ejemplo	143
AWS IoT Core motor de reglas	145
Grúas	148
Antecedentes	148
Enviar comandos	149
Modelos de detector	150
Entradas	157
Mensajes	157
Detección de eventos con sensores y aplicaciones	159
Dispositivo HeartBeat	161
ISAalarma	163
Alarma sencilla	173
Monitoreo con alarmas	178
Trabajando con AWS IoT SiteWise	178
Flujo de reconocimiento	178
Creación de un modelo de alarma	179
Requisitos	180
Creación de un modelo de alarma (consola)	180
Respuesta a las alarmas	183
Administración de las notificaciones de alarma	184
Creación de una función de Lambda	185
Uso de la función de Lambda proporcionada por AWS IoT Events	193
Gestione los destinatarios de las alarmas	194
Seguridad	196
Administración de identidades y accesos	196
Público	197
Autenticación con identidades	198
Administración de acceso mediante políticas	201

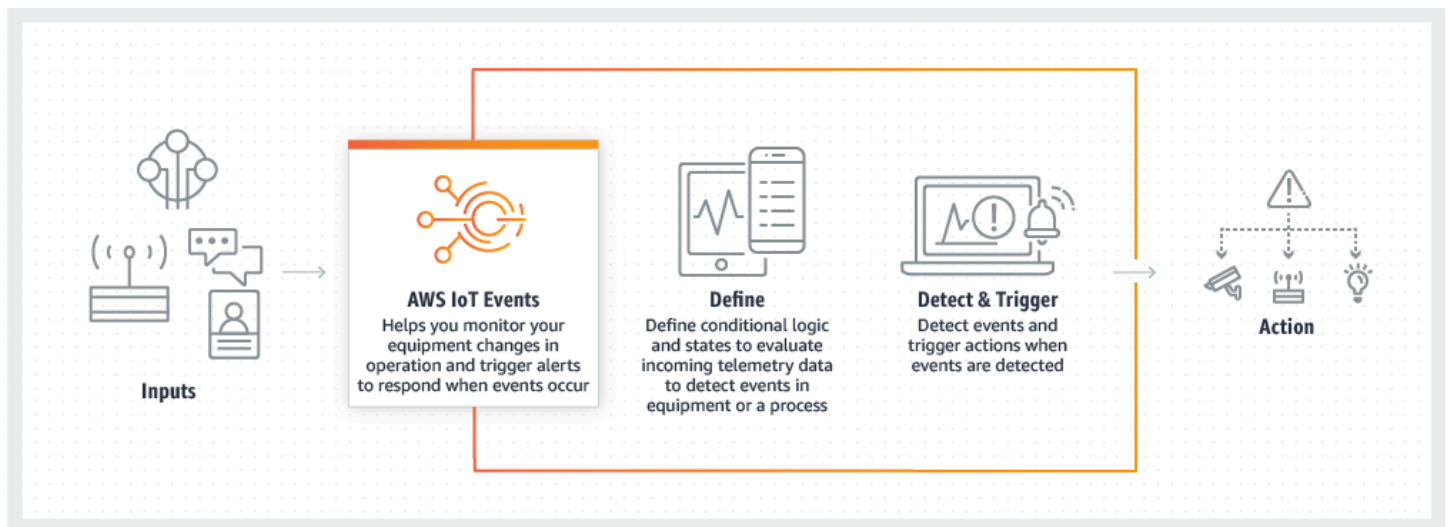
Más información	203
¿Cómo AWS IoT Events funciona con IAM	203
Ejemplos de políticas basadas en identidades	208
Prevención de la sustitución confusa entre servicios	213
Resolución de problemas	217
Supervisión	219
Herramientas de monitoreo	220
Monitorización con Amazon CloudWatch	221
Registrar AWS IoT Events API llamadas con AWS CloudTrail	223
Validación de conformidad	240
Resiliencia	242
Seguridad de la infraestructura	242
Cuotas	244
Etiquetado	245
Conceptos básicos de etiquetas	245
Restricciones y limitaciones en las etiquetas	246
Uso de etiquetas con políticas de IAM	246
Resolución de problemas	250
AWS IoT Events Problemas y soluciones comunes	250
Errores de creación del modelo de detector	251
Actualizaciones de un modelo de detector eliminado	251
Fallo de activación de la acción (al cumplirse una condición)	251
Fallo de activación de la acción (al superar un umbral)	252
Uso incorrecto de los estados	252
Mensaje de conexión	252
InvalidRequestException mensaje	253
action.setTimerErrores de Amazon CloudWatch Logs	253
Errores de CloudWatch carga útil de Amazon	254
Tipos de datos incompatibles	256
No se pudo enviar el mensaje a AWS IoT Events	257
Solución de problemas de un modelo de detector	258
Información de diagnóstico	258
Análisis de un modelo de detector (consola)	272
Análisis de un modelo de detector (AWS CLI)	274
Comandos	279
AWS IoT Events acciones	279

AWS IoT Events datos	279
Historial del documento	280
Actualizaciones anteriores	281
.....	cclxxxiv

¿Qué es AWS IoT Events?

AWS IoT Events le permite monitorizar sus equipos o flotas de dispositivos para detectar fallos o cambios en el funcionamiento y activar acciones cuando se producen dichos eventos. AWS IoT Events monitorea continuamente los datos de los sensores de IoT de los dispositivos, procesos, aplicaciones y otros AWS servicios para identificar eventos importantes y poder tomar medidas.

Úselo AWS IoT Events para crear aplicaciones complejas de monitoreo de eventos en la AWS nube a las que pueda acceder a través de la AWS IoT Events consola o APIs.



Temas

- [Beneficios y características](#)
- [Casos de uso](#)

Beneficios y características

Acepte entradas de múltiples fuentes

AWS IoT Events acepta entradas de muchas fuentes de datos de telemetría de IoT. Estos incluyen dispositivos sensores, aplicaciones de administración y otros AWS IoT servicios, como AWS IoT Core y AWS IoT Analytics. Puede insertar cualquier entrada de datos de telemetría AWS IoT Events mediante una API interfaz estándar (BatchPutMessageAPI) o la AWS IoT Events consola.

Para obtener más información sobre cómo empezar AWS IoT Events, consulte [Primeros pasos con la AWS IoT Events consola](#)

Utilice expresiones lógicas sencillas para reconocer patrones complejos de eventos

AWS IoT Events puede reconocer patrones de eventos que involucran múltiples entradas desde un solo dispositivo o aplicación de IoT, o desde diversos equipos y muchos sensores independientes. Esto es de especial utilidad porque cada sensor y aplicación proporciona información importante. Pero solo combinando diversos datos de sensores y aplicaciones puede obtener una imagen completa del rendimiento y la calidad de las operaciones. Puede configurar AWS IoT Events los detectores para que reconozcan estos eventos mediante expresiones lógicas sencillas en lugar de códigos complejos.

Para obtener más información sobre las expresiones lógicas, consulte [Expresiones para filtrar, transformar y procesar datos de eventos](#).

Activa acciones en función de eventos

AWS IoT Events le permite activar acciones directamente en Amazon Simple Notification Service (AmazonSNS), Lambda AWS IoT Core, Amazon SQS y Amazon Kinesis Firehose. También puede activar una AWS Lambda función mediante el motor de AWS IoT reglas, lo que permite realizar acciones mediante otros servicios, como Amazon Connect, o sus propias aplicaciones de planificación de recursos empresariales (ERP).

AWS IoT Events incluye una biblioteca prediseñada de acciones que puede realizar y también le permite definir las suyas propias.

Para obtener más información sobre cómo activar acciones en función de eventos, consulte [Acciones compatibles para recibir datos y activar acciones](#)

Escale automáticamente para satisfacer las demandas de su flota

AWS IoT Events se escala automáticamente cuando se conectan dispositivos homogéneos. Puede definir un detector una vez para un tipo específico de dispositivo, y el servicio escalará y gestionará automáticamente todas las instancias de ese dispositivo a las que se conecte AWS IoT Events.

Para ver ejemplos de modelos de detectores, consulte [AWS IoT Events ejemplos de modelos de detectores](#).

Casos de uso

AWS IoT Events tiene muchos usos. Estos son algunos ejemplos de casos de uso.

Supervise y mantenga los dispositivos remotos

Supervisar una flota de máquinas desplegadas de forma remota puede resultar difícil, especialmente cuando se produce un mal funcionamiento sin un contexto claro. Si una máquina deja de funcionar, puede ser necesario sustituir toda la unidad de procesamiento o la máquina. Pero esto no es sostenible. Con AWS IoT Events él, puede recibir mensajes de varios sensores en cada máquina para ayudarlo a diagnosticar problemas específicos a lo largo del tiempo. En lugar de sustituir toda la unidad, ahora dispone de la información necesaria para enviar a un técnico la pieza exacta que necesita sustituir. Con millones de máquinas, el ahorro podría ascender a millones de dólares, reduciendo sus costos totales de propiedad o mantenimiento de cada máquina.

Administre robots industriales

La implementación de robots en sus instalaciones para automatizar el movimiento de paquetes puede mejorar considerablemente la eficiencia. Para minimizar los costes, los robots pueden equiparse con sensores sencillos y de bajo coste que envíen los datos a la nube. Sin embargo, con docenas de sensores y cientos de modos de funcionamiento, detectar problemas en tiempo real puede resultar difícil. Con AWS IoT Events él, puede crear un sistema experto que procese los datos de los sensores en la nube y cree alertas para notificar automáticamente al personal técnico en caso de que se produzca un fallo inminente.

Rastrea los sistemas de automatización de edificios

En los centros de datos, el monitoreo de las altas temperaturas y la baja humedad ayuda a prevenir fallas en los equipos. Los sensores suelen adquirirse de muchos fabricantes y cada tipo viene con su propio software de gestión. Sin embargo, el software de administración de diferentes proveedores a veces no es compatible, lo que dificulta la detección de problemas. Con AWS IoT Events él, puede configurar alertas para notificar a sus analistas de operaciones los problemas en sus sistemas de calefacción y refrigeración con bastante antelación a las averías. De este modo, puede evitar un cierre no programado del centro de datos que le costaría miles de dólares en sustitución de equipos y una posible pérdida de ingresos.

Con AWS IoT Events figuración

Esta sección proporciona una guía de configuración AWS IoT Events, que incluye la creación de una AWS cuenta, la configuración de los permisos necesarios y el establecimiento de funciones para administrar el acceso a los recursos.

Temas

- [Configuración de una Cuenta de AWS](#)
- [Configurar permisos para AWS IoT Events](#)

Configuración de una Cuenta de AWS

Inscríbase en un Cuenta de AWS

Si no tiene una Cuenta de AWS, complete los siguientes pasos para crearlo.

Para suscribirte a una Cuenta de AWS

1. Abrir <https://portal.aws.amazon.com/billing/registro>.
2. Siga las instrucciones que se le indiquen.

Parte del procedimiento de registro consiste en recibir una llamada telefónica e indicar un código de verificación en el teclado del teléfono.

Cuando te registras en un Cuenta de AWS, Usuario raíz de la cuenta de AWS se crea un. El usuario raíz tendrá acceso a todos los Servicios de AWS y recursos de esa cuenta. Como práctica recomendada de seguridad, asigne acceso administrativo a un usuario y utilice únicamente el usuario raíz para realizar [tareas que requieren acceso de usuario raíz](#).

AWS te envía un correo electrónico de confirmación una vez finalizado el proceso de registro. Puede ver la actividad de la cuenta y administrar la cuenta en cualquier momento entrando en <https://aws.amazon.com/> y seleccionando Mi cuenta.

Creación de un usuario con acceso administrativo

Después de crear un usuario administrativo Cuenta de AWS, asegúrelo Usuario raíz de la cuenta de AWS AWS IAM Identity Center, habilite y cree un usuario administrativo para no usar el usuario root en las tareas diarias.

Proteja su Usuario raíz de la cuenta de AWS

1. Inicie sesión [AWS Management Console](#) como propietario de la cuenta seleccionando el usuario root e introduciendo su dirección de Cuenta de AWS correo electrónico. En la siguiente página, escriba su contraseña.

Para obtener ayuda para iniciar sesión con el usuario raíz, consulte [Iniciar sesión como usuario raíz](#) en la Guía del usuario de AWS Sign-In .

2. Activa la autenticación multifactorial (MFA) para tu usuario root.

Para obtener instrucciones, consulte [Habilitar un MFA dispositivo virtual para el usuario Cuenta de AWS root \(consola\)](#) en la Guía del IAM usuario.

Creación de un usuario con acceso administrativo

1. Habilite IAM Identity Center.

Consulte las instrucciones en [Activar AWS IAM Identity Center](#) en la Guía del usuario de AWS IAM Identity Center .

2. En IAM Identity Center, conceda acceso administrativo a un usuario.

Para ver un tutorial sobre cómo usar el Directorio de IAM Identity Center como fuente de identidad, consulte [Configurar el acceso de los usuarios con la configuración predeterminada Directorio de IAM Identity Center](#) en la Guía del AWS IAM Identity Center usuario.

Iniciar sesión como usuario con acceso de administrador

- Para iniciar sesión con su usuario de IAM Identity Center, utilice el inicio de sesión URL que se envió a su dirección de correo electrónico cuando creó el usuario de IAM Identity Center.

Para obtener ayuda para iniciar sesión con un usuario de IAM Identity Center, consulte [Iniciar sesión en el portal de AWS acceso](#) en la Guía del AWS Sign-In usuario.

Concesión de acceso a usuarios adicionales

1. En IAM Identity Center, cree un conjunto de permisos que siga la práctica recomendada de aplicar permisos con privilegios mínimos.

Para conocer las instrucciones, consulte [Create a permission set](#) en la Guía del usuario de AWS IAM Identity Center .

2. Asigne usuarios a un grupo y, a continuación, asigne el acceso de inicio de sesión único al grupo.

Para conocer las instrucciones, consulte [Add groups](#) en la Guía del usuario de AWS IAM Identity Center .

Configurar permisos para AWS IoT Events

En esta sección se describen los permisos necesarios para utilizar algunas funciones de AWS IoT Events. Puede usar AWS CLI los comandos o la consola AWS Identity and Access Management (IAM) para crear roles y las políticas de permisos asociadas para acceder a los recursos o realizar determinadas funciones AWS IoT Events.

La [guía del IAM usuario contiene](#) información más detallada sobre cómo controlar de forma segura los permisos de acceso a AWS los recursos. Para obtener información específica sobre AWS IoT Events, consulte [Acciones, recursos y claves de condición para AWS IoT Events](#).

Si quieres usar la IAM consola para crear y administrar roles y permisos, consulta el [IAMtutorial: Delega el acceso entre AWS cuentas mediante IAM roles](#).

Note

Las claves pueden tener entre 1 y 128 caracteres e incluir:

- letras (a-z) en mayúsculas y minúsculas
- números (0-9)
- caracteres especiales (-, _ o :).

Permisos de acción para AWS IoT Events

AWS IoT Events le permite activar acciones que utilizan otros AWS servicios. Para ello, debe conceder AWS IoT Events permiso para realizar estas acciones en su nombre. Esta sección contiene una lista de las acciones y una política de ejemplo que concede permiso para realizar todas estas acciones en sus recursos. Cambia el *region* y *account-id* referencias según sea necesario. De ser posible, también debería cambiar los comodines (*) para hacer referencia a los recursos específicos a los que se accederá. Puedes usar la IAM consola para conceder permiso AWS IoT Events para enviar una SNS alerta de Amazon que hayas definido.

AWS IoT Events admite las siguientes acciones que le permiten usar un temporizador o configurar una variable:

- [setTimer](#) para crear un temporizador.
- [resetTimer](#) para restablecer el temporizador.
- [clearTimer](#) para eliminar el temporizador.
- [setVariable](#) para crear una variable.

AWS IoT Events admite las siguientes acciones que le permiten trabajar con AWS los servicios:

- [iotTopicPublish](#) publicar un mensaje sobre un MQTT tema.
- [iotEvents](#) para enviar datos a AWS IoT Events como valor de entrada.
- [iotSiteWise](#) para enviar datos a una propiedad de recurso en AWS IoT SiteWise.
- [dynamoDB](#) para enviar datos a una tabla de Amazon DynamoDB.
- [dynamoDBv2](#) para enviar datos a una tabla de Amazon DynamoDB.
- [firehose](#) para enviar datos a una transmisión de Amazon Data Firehose.
- [lambda](#) para invocar una función de AWS Lambda .
- [sns](#) para enviar datos como notificación de inserción.
- [sqs](#) para enviar datos a una SQS cola de Amazon.

Example Política

```
{
  "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>:*"
  }
]

```

```
}
```

Asegurar los datos de entrada en AWS IoT Events

Es importante tener en cuenta quién puede conceder acceso a los datos de entrada para su uso en un modelo de detector. Si tiene un usuario o entidad cuyos permisos generales desea restringir, pero que tenga permiso para crear o actualizar un modelo de detector, también debe conceder permiso para que ese usuario o entidad actualice la ruta de entrada. Esto significa que además de conceder permiso para `iotevents:CreateDetectorModel` y `iotevents:UpdateDetectorModel`, también debe conceder permiso para `iotevents:UpdateInputRouting`.

Example

La siguiente política añade permiso para `iotevents:UpdateInputRouting`.

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

Puede especificar una lista de entradas de Amazon Resource Names (ARNs) en lugar del comodín `*` para el "Resource" para limitar este permiso a entradas específicas. Esto le permite restringir el acceso a los datos de entrada que consumen los modelos de detector creados o actualizados por el usuario o la entidad.

Política de funciones de CloudWatch registro de Amazon

Los siguientes documentos de política proporcionan la política de roles y la política de confianza que AWS IoT Events permiten enviar registros CloudWatch en su nombre.

Política de roles:


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

Política de confianza:

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

También necesitas adjuntar al usuario una política de IAM permisos que le permita transferir roles, como se indica a continuación. Para obtener más información, consulte [Otorgar permisos a un usuario para transferir un rol a un AWS servicio](#) en la Guía del IAM usuario.

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

Puede usar el siguiente comando para colocar la política de recursos para CloudWatch los registros. Esto permite AWS IoT Events colocar los eventos de registro en las CloudWatch transmisiones.

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

Utilice el siguiente comando para colocar las opciones de registro. Sustituya el `roleArn` por el rol de registro que haya creado.

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

Política de funciones SNS de mensajería de Amazon

Los siguientes documentos de política proporcionan la política de roles y la política de confianza que AWS IoT Events permiten enviar SNS mensajes.

Política de roles:

```
{
```

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

Política de confianza:

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

Primeros pasos con la AWS IoT Events consola

En esta sección le mostramos cómo crear una entrada y un modelo de detector utilizando la [consola de AWS IoT Events](#). Se modelan dos estados de un motor: un estado normal y un estado de sobrepresión. Cuando la presión medida en el motor supera un determinado umbral, el modelo pasa del estado normal al estado de sobrepresión. A continuación, envía un SNS mensaje de Amazon para alertar a un técnico sobre el estado. Cuando la presión vuelve a caer por debajo del umbral durante tres lecturas de presión consecutivas, el modelo vuelve al estado normal y envía otro SNS mensaje de Amazon como confirmación.

Comprobamos que haya tres lecturas consecutivas por debajo del umbral de presión para eliminar posibles tartamudeos de mensajes de estado de sobrepresión o normal por si hubiera una fase de recuperación no lineal o una lectura de presión anómala.

En la consola también puede encontrar varias plantillas prefabricadas de modelos de detectores que puede personalizar. También puede utilizar la consola para importar modelos de detectores que otros hayan creado o exportar sus modelos de detector y utilizarlos en distintas AWS regiones. Si importa un modelo de detector, asegúrese de crear las entradas necesarias o de volver a crearlas para la nueva región y actualice cualquier función ARNs utilizada.

Usa la AWS IoT Events consola para obtener información sobre lo siguiente.

Definir entradas

Para supervisar sus dispositivos y procesos, deben tener una forma de transferir datos de telemetría a AWS IoT Events. Esto se hace enviando mensajes como entradas a AWS IoT Events. Puede hacer esto de varias formas:

- Usa la [BatchPutMessage](#) operación.
- En AWS IoT Core, escriba una regla de [AWS IoT Events acción](#) para el motor de AWS IoT reglas al que reenvía los datos de sus mensajes. AWS IoT Events Debe identificar la entrada por su nombre.
- En AWS IoT Analytics, utilice la [CreateDataset](#) operación para crear un conjunto de datos con `contentDeliveryRules`. Estas reglas especifican la AWS IoT Events entrada a la que se envía automáticamente el contenido del conjunto de datos.

Para que sus dispositivos puedan enviar datos de este modo, debe definir una o más entradas. Para ello, asigne un nombre a cada entrada y especifique qué campos de los datos del mensaje entrante supervisa la entrada.

Crear un modelo de detector

Defina un modelo de detector (un modelo de su equipo o proceso) utilizando estados. Para cada estado, defina una lógica condicional (booleana) que evalúe las entradas para detectar eventos significativos. Cuando el modelo de detector detecta un evento, puede cambiar el estado o iniciar acciones personalizadas o predefinidas mediante otros AWS servicios. Puede definir eventos adicionales que inicien acciones al entrar o salir de un estado y, opcionalmente, al cumplirse una condición.

En este tutorial, envías un SNS mensaje de Amazon como acción cuando el modelo entra o sale de un estado determinado.

Cómo supervisar un dispositivo o proceso

Si supervisa varios dispositivos o procesos, especifique un campo en cada entrada que identifique el dispositivo o proceso en particular del que procede la entrada. Consulte el campo `key` en `CreateDetectorModel`. Cuando el campo de entrada identificado por `key` reconoce un nuevo valor, se identifica un nuevo dispositivo y se crea un detector. Cada detector es una instancia del modelo de detector. El nuevo detector sigue respondiendo a las entradas procedentes de ese dispositivo hasta que actualice o elimine su modelo de detector.

Si supervisa un único proceso (incluso con varios dispositivos o subprocessos enviando entradas), no especifica un campo `key` de identificación único. En este caso, el modelo crea un único detector (instancia) cuando llega la primera entrada.

Cómo enviar mensajes como entradas a su modelo de detector

Existen varias formas de enviar un mensaje desde un dispositivo o proceso como entrada a un detector de AWS IoT Events que no requieren que realice un formateo adicional del mensaje. En este tutorial, utilizarás la AWS IoT consola para escribir una regla de [AWS IoT Events acción](#) para el motor de AWS IoT reglas al que reenvía los datos de tus mensajes. AWS IoT Events

Para ello, identifique la entrada por su nombre y siga utilizando la AWS IoT consola para generar mensajes que se reenvíen como entradas. AWS IoT Events

Note

En este tutorial utiliza la consola para crear los mismos `input` y `detector model` que se muestran en el ejemplo en [Tutoriales para casos de AWS IoT Events uso](#). Puedes usar este JSON ejemplo como ayuda para seguir el tutorial.

Temas

- [Requisitos previos para empezar AWS IoT Events](#)
- [Cree una entrada para los modelos](#)
- [Crear un modelo de detector](#)
- [Enviar entradas para probar el modelo de detector](#)

Requisitos previos para empezar AWS IoT Events

Si no tienes una AWS cuenta, crea una.

1. Sigue estos pasos [Con AWS IoT Events figuración](#) para asegurarte de que la cuenta y los permisos estén configurados correctamente.
2. Cree dos temas de Amazon Simple Notification Service (AmazonSNS).

En este tutorial (y el ejemplo correspondiente) se supone que has creado dos SNS temas de Amazon. Estos temas se muestran como: `arn:aws:sns:us-east-1:123456789012:underPressureAction` y `arn:aws:sns:us-east-1:123456789012:pressureClearedAction`. ARNs Sustituya estos valores por los SNS temas ARNs de Amazon que haya creado. Para obtener más información, consulte la [Guía para desarrolladores de Amazon Simple Notification Service](#).

Como alternativa a publicar alertas sobre los SNS temas de Amazon, puedes hacer que los detectores envíen MQTT mensajes con un tema que especifiques. Con esta opción, puede comprobar que su modelo de detector crea instancias y que esas instancias envían alertas mediante la consola AWS IoT Core para suscribirse a los mensajes enviados MQTT sobre esos temas y supervisarlos. También puede definir el nombre del MQTT tema de forma dinámica en tiempo de ejecución mediante una entrada o variable creada en el modelo de detector.

3. Elija una Región de AWS que sea compatible AWS IoT Events. Para obtener más información, consulte [AWS IoT Events](#) en la Referencia general de AWS. Para obtener ayuda, consulte [Uso de la AWS Management Console](#) en Introducción a la AWS Management Console.

Cree una entrada para los modelos

Al construir las entradas para sus modelos, le recomendamos que reúna archivos que contengan cargas de mensajes de muestra que sus dispositivos o procesos envíen para informar de su estado. Disponer de estos archivos le ayudará a definir las entradas necesarias.

Puede crear una entrada a través de varios métodos, que se describen en esta sección.

Crear un archivo JSON de entrada

1. Para empezar, cree un archivo llamado `input.json` en su sistema de archivos local con el siguiente contenido:

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


2. Ahora que tiene este archivo `input.json` de inicio, puede crear una entrada. Hay dos maneras de crear una entrada. Puede crear una entrada mediante el panel de navegación de la [AWS IoT Events consola](#). O bien, puede crear una entrada dentro del modelo de detector después de crearla.

Cree y configure una entrada

Aprenda a crear una entrada para un modelo de alarma o un modelo de detector.

1. Inicie sesión en la [AWS IoT Events consola](#) o seleccione la opción Crear una AWS IoT Events cuenta nueva.
2. En la AWS IoT Events consola, en la esquina superior izquierda, selecciona y expande el panel de navegación.
3. En el panel de navegación izquierdo, seleccione Entradas.
4. En la esquina derecha de la consola, seleccione Crear entrada.
5. Proporcione un único InputName.
6. Opcional: introduce una descripción para tu entrada.
7. Para cargar un JSON archivo, selecciona el `input.json` archivo para el que lo creaste en la descripción general [Crear un archivo JSON de entrada](#). La opción Elegir atributos de entrada aparece junto con una lista de los atributos introducidos.
8. En Elegir atributos de entrada, seleccione los atributos que desee utilizar y luego Crear. En este ejemplo, seleccionamos `motorid` y `sensorData.pressure`.

9. Opcional: agrega las etiquetas relevantes a la entrada.

 Note

También puede crear entradas adicionales dentro del modelo de detector de la [AWS IoT Events consola](#). Para obtener más información, consulte [Cree una entrada dentro del modelo de detector](#).

Cree una entrada dentro del modelo de detector

En esta sección se muestra cómo definir una entrada para que un modelo de detector reciba datos o mensajes de telemetría.

1. Abra la [consola de AWS IoT Events](#).
2. En la AWS IoT Events consola, elija Crear modelo de detector.
3. Elija Crear nuevo.
4. Elija Create input (Crear entrada).
5. Para la entrada, introduzca una InputName, una descripción opcional y elija Cargar archivo. En el cuadro de diálogo que aparece, seleccione el `input.json` archivo para el que creó en la vista general [Crear un archivo JSON de entrada](#).
6. En Elegir atributos de entrada, seleccione los atributos que desee utilizar y luego Crear. En este ejemplo, seleccionamos `motorId` y `sensorData.pressure`.

Crear un modelo de detector

En este tema, define un modelo de detector (un modelo de su equipo o proceso) utilizando estados.

Para cada estado, debe definir una lógica condicional (booleana) que evalúe las entradas para detectar un evento significativo. Al detectar un evento, este cambia el estado y puede iniciar acciones adicionales. Estos eventos se conocen como eventos de transición.

En sus estados, también define eventos que pueden ejecutar acciones cada vez que el detector entra o sale de ese estado o cuando se recibe una entrada (se conocen como eventos `OnEnter`, `OnExit` y `OnInput`). Las acciones se ejecutan solo si la lógica condicional del evento da como resultado `true`.

Para crear un modelo de detector

1. Se ha creado el primer estado del detector para usted. Para modificarlo, seleccione el círculo con la etiqueta `State_1` en el espacio de edición principal.
2. En el panel Estado, introduzca el nombre del estado y OnEnter elija Añadir evento.
3. En la página Añadir OnEnter evento, introduzca el nombre del evento y la condición del evento. En este ejemplo, introduzca `true` para indicar que el evento siempre se inicia al introducir el estado.
4. En Acciones del evento, seleccione Añadir acción.
5. En Acciones del evento, realice lo siguiente:
 - a. Seleccione Establecer variable
 - b. En Operación de la variable, seleccione Asignar valor.
 - c. En Nombre de la variable, introduzca el nombre de la variable que va a establecer.
 - d. En Valor de la variable, introduzca el valor `0` (cero).
6. Seleccione Guardar.

Una variable, como la que ha definido, se puede establecer (darle un valor) en cualquier evento del modelo de detector. Solo se puede hacer referencia al valor de la variable (por ejemplo, en la lógica condicional de un evento) después de que el detector haya alcanzado un estado y ejecutado una acción donde esté definido o establecido.

7. En el panel Estado, seleccione la X situada junto a Estado para volver a la Paleta de modelos de detectores.
8. Para crear un segundo estado de detector, en la Paleta de modelos de detectores, seleccione Estado y arrástrelo al espacio de edición principal. Esto crea un estado titulado `untitled_state_1`.
9. Haga una pausa en el primer estado (Normal). Aparece una flecha en la circunferencia de estado.
10. Pulse y arrastre la flecha del primer estado al segundo estado. Aparece una línea dirigida del primer estado al segundo estado (denominada Sin título).
11. Seleccione la línea Sin título. En el panel Evento de transición, introduzca el Nombre del evento y una Lógica de activación del evento.
12. En el panel Evento de transición, seleccione Añadir acción.
13. En el panel Añadir acciones de evento de transición, seleccione Añadir acción.

14. En Elegir una acción, seleccione Establecer variable.
 - a. En Operación de la variable, seleccione Asignar valor.
 - b. En Nombre de la variable, introduzca el nombre de la variable.
 - c. En Asignar valor, escriba el valor, como `$variable.pressureThresholdBreached + 3`:
 - d. Seleccione Guardar.
15. Seleccione el segundo estado `untitled_state_1`.
16. En el panel Estado, introduzca el Nombre del estado y en OnEnter, seleccione Añadir evento.
17. En la página Añadir OnEnter evento, introduzca el nombre del evento y la condición del evento. Seleccione Agregar acción.
18. En Elegir una acción, selecciona Enviar SNS mensaje.
 - a. Para SNSel tema, introduce el objetivo ARN de tu SNS tema de Amazon.
 - b. Seleccione Guardar.
19. Siga añadiendo los eventos indicados en el ejemplo.
 - a. Para OnInput, selecciona Añadir evento e introduce y guarda la siguiente información del evento.

```
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. Para OnInput, elija Agregar evento e introduzca y guarde la siguiente información del evento.

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

- c. Para OnExit, selecciona Añadir evento e introduce y guarda la siguiente información del evento utilizando el SNS tema ARN de Amazon que has creado.

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

20. Haga una pausa en el segundo estado (Peligroso). Aparece una flecha en la circunferencia del estado
21. Pulse y arrastre la flecha del segundo estado al primer estado. Aparece una línea dirigida con la etiqueta Sin título.
22. Seleccione la línea Sin título y en el panel Evento de transición, introduzca el Nombre del evento y la Lógica de activación del evento utilizando la siguiente información.

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

Para obtener más información sobre por qué probamos el valor `$input` y el valor `$variable` en la lógica de activación, consulte en la entrada la disponibilidad de los valores de las variables en [Restricciones y limitaciones del modelo de detector](#).

23. Seleccione el estado Inicio. Por defecto, este estado se generó cuando creó un modelo de detector). En el panel Inicio, seleccione el Estado de destino (por ejemplo, Normal).
24. A continuación, configure su modelo de detector para que escuche las entradas. En la esquina superior derecha, seleccione Publicar.
25. En la página Publicar modelo de detector, haga lo siguiente.
 - a. Introduzca el Nombre del modelo de detector, una Descripción y el nombre de un Rol. Este rol se crea para usted.
 - b. Seleccione Crear un detector para cada valor clave único. Para crear su propio Rol y utilizarlo, siga los pasos que se indican en [Configurar permisos para AWS IoT Events](#) e introdúzcalo como Rol aquí.

26. En Clave de creación del detector, elija el nombre de uno de los atributos de la entrada que definió anteriormente. El atributo que elija como clave de creación del detector debe estar presente en cada entrada de mensaje y debe ser único para cada dispositivo que envíe mensajes. En este ejemplo se utiliza el atributo `motorid`.
27. Seleccione Guardar y publicar.

Note

El número de detectores únicos creados para un modelo de detector determinado se basa en los mensajes de entrada enviados. Cuando se crea un modelo de detector, se selecciona una clave a partir de los atributos de entrada. Esta clave determina qué instancia de detector se utilizará. Si la clave no se ha visto antes (para este modelo de detector), se crea una nueva instancia de detector. Si se ha visto antes, se utiliza la instancia de detector existente correspondiente a este valor clave.

Puede hacer una copia de seguridad de la definición de su modelo de detector (inJSON) recrear o actualizar el modelo de detector o utilizarla como plantilla para crear otro modelo de detector.

Puede hacerlo desde la consola o mediante el siguiente CLI comando. De ser necesario, cambie el nombre del modelo de detector para que coincida con el que utilizó al publicarlo en el paso anterior.

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

Esto crea un archivo (`motorDetectorModel.json`) que tiene un contenido similar al siguiente.

```
{
  "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":
"pressureThresholdBreach",
                      "value":
"$variable.pressureThresholdBreach + 3"
                    }
                  }
                ],
                "condition": "$input.PressureInput.sensorData.pressure
> 70",
                "nextState": "Dangerous"
              }
            ],
            "events": []
          },
          "stateName": "Normal",
          "onEnter": {
            "events": [
              {
                "eventName": "init",
                "actions": [
                  {
                    "setVariable": {
                      "variableName":
"pressureThresholdBreach",
                      "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"
}
}
```

Enviar entradas para probar el modelo de detector


Hay varias formas de recibir datos de telemetría en AWS IoT Events (consulte [Acciones compatibles para recibir datos y activar acciones](#)). En este tema se muestra cómo crear una AWS IoT regla en la AWS IoT consola que reenvíe los mensajes como entradas al detector. AWS IoT Events Puede usar el MQTT cliente de la AWS IoT consola para enviar mensajes de prueba. Puedes usar este método para obtener datos de telemetría para saber AWS IoT Events cuándo tus dispositivos pueden enviar MQTT mensajes mediante el AWS IoT intermediario de mensajes.

Para enviar entradas para probar el modelo de detector

1. Abra la [consola de AWS IoT Core](#). En el panel de navegación izquierdo, en Administrar, seleccione Enrutamiento de mensajes y luego Reglas.
2. Seleccione Crear regla en la esquina superior derecha.
3. En la página Crear una regla, siga los pasos que se indican a continuación:

1. Paso 1. Especificar las propiedades de la regla. Complete los siguientes campos:

- Nombre de la regla. Escriba un nombre para su regla, como `MyIoTEventsRule`.

 Note

No utilice espacios.

- Descripción de la regla. Es opcional.
- Elija Next (Siguiendo).

2. Paso 2. Instrucción de configuración SQL. Complete los siguientes campos:

- SQLversión. Seleccione la opción apropiada en la lista.
- SQLdeclaración. Escriba **`SELECT *, topic(2) as motorid FROM 'motors/+/' status'`**.

Elija Next (Siguiendo).

3. Paso 3. Añadir acciones de la regla. En la sección Acciones de la regla, complete lo siguiente:

- Acción 1. Seleccione IoT Events. Se muestran los siguientes campos:
 - a. Nombre de la entrada. Seleccione la opción apropiada en la lista. Si la entrada no aparece, seleccione Actualizar.

Para crear una entrada nueva, seleccione Crear entrada de IoT Events. Complete los siguientes campos:

- Nombre de la entrada. Escriba PressureInput.
- Descripción. Es opcional.
- Sube un JSON archivo. Sube una copia de tu JSON archivo. Si no tuviera un archivo, hay un enlace a un archivo de muestra en esta pantalla. El código incluye:

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

- Elegir atributos de entrada. Seleccione las opciones apropiadas.
- Etiquetas. Es opcional.

Seleccione Crear.

Vuelva a la pantalla Crear regla y actualice el campo Nombre de la entrada. Seleccione la entrada que acaba de crear.

- Modo Lote. Es opcional. Si la carga es una matriz de mensajes, seleccione esta opción.
- ID de mensaje. Esto es opcional, pero recomendable.
- IAMrol. Seleccione el rol apropiado en la lista. Si el rol no figura en la lista, seleccione Crear nuevo rol.

Escriba el Nombre del rol y seleccione Crear.

Para añadir otra regla, seleccione Añadir regla.

- Acción de error. Esta sección es opcional. Para añadir una acción, seleccione Añadir acción de error y seleccione la acción apropiada en la lista.

Complete los campos que aparezcan.

- Elija Next (Siguiendo).


4. Paso 4. Revisar y crear. Revise la información en pantalla y seleccione Crear.

~~4. En el panel de navegación izquierdo, en Probar, elija cliente MQTT de prueba.~~

5. Elija Publicar en un tema. Complete los siguientes campos:

- Nombre del tema. Introduzca un nombre para identificar el mensaje, por ejemplo, `motors/Fulton-A32/status`.
- Carga útil del mensaje. Introduzca lo siguiente:

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

 Note

Cambie el `messageId` cada vez que publique un nuevo mensaje.

6. Para Publicar, mantenga el tema igual, pero cambie la `pressure` en la carga por un valor superior al valor umbral que especificó en el modelo de detector (por ejemplo **85**).
7. Seleccione Publicar.

La instancia de detección que has creado genera y te envía un SNS mensaje de Amazon. Continúe enviando mensajes con lecturas de presión por encima o por debajo del umbral de presión (70 en este ejemplo) para ver el detector en funcionamiento.

En este ejemplo, debes enviar tres mensajes con lecturas de presión por debajo del umbral para volver al estado Normal y recibir un SNS mensaje de Amazon que indique que se ha superado la condición de sobrepresión. Una vez que vuelve al estado Normal, un mensaje con una lectura de presión superior al límite hace que el detector entre en estado Peligroso y envíe un SNS mensaje de Amazon indicándolo.

Ahora que ha creado un modelo de detector y entrada sencillos, pruebe lo siguiente.

- Consulte más ejemplos de modelos de detectores (plantillas) en la consola.
- Siga los pasos que se indican [step-by-step Ejemplo sencillo](#) a continuación para crear un modelo de entrada y detector utilizando el AWS CLI
- Conozca los detalles de las [Expresiones para filtrar, transformar y procesar datos de eventos](#) utilizadas en los eventos.

- Información sobre [Acciones compatibles para recibir datos y activar acciones](#).
- Si algo no funciona, consulte [Solución de problemas AWS IoT Events](#).

Mejores prácticas para AWS IoT Events

Siga estas prácticas recomendadas para obtener el máximo beneficio de AWS IoT Events.

Temas

- [Habilite el CloudWatch registro de Amazon al desarrollar modelos AWS IoT Events de detectores](#)
- [Publique periódicamente para guardar su modelo de detector cuando trabaje en la AWS IoT Events consola](#)

Habilite el CloudWatch registro de Amazon al desarrollar modelos AWS IoT Events de detectores

Amazon CloudWatch monitorea tus AWS recursos y las aplicaciones en las que AWS ejecutas en tiempo real. De este CloudWatch modo, obtiene visibilidad en todo el sistema sobre el uso de los recursos, el rendimiento de las aplicaciones y el estado operativo. Cuando desarrolla o depura un modelo de AWS IoT Events detector, le CloudWatch ayuda a saber qué AWS IoT Events es lo que está haciendo y cualquier error que detecte.

Para habilitar CloudWatch

1. Si aún no lo ha hecho, siga los pasos que se indican [Configurar permisos para AWS IoT Events](#) para crear un rol con una política adjunta que otorgue permiso para crear y administrar CloudWatch registros AWS IoT Events.
2. Vaya a la [consola de AWS IoT Events](#).
3. En el panel de navegación, seleccione Configuración.
4. En la página Configuración, seleccione Editar.
5. En la página Editar opciones de registro, en la sección Opciones de registro, haga lo siguiente:
 - a. En Nivel de verbosidad, seleccione una opción.
 - b. En Seleccionar rol, seleccione un rol con permisos suficientes para realizar las acciones de registro que haya elegido.
 - c. (Opcional) Si has elegido Depurar como nivel de verbosidad, puedes añadir objetivos de depuración de la siguiente manera:
 - i. En Objetivos de depuración, selecciona Añadir opción de modelo.

- ii. Introduzca un nombre de modelo de detector y (opcional) especifique KeyValues modelos de detector y los detectores específicos (instancias) que desee registrar.
6. Elija Actualizar.

Sus opciones de registro se han actualizado correctamente.

Publique periódicamente para guardar su modelo de detector cuando trabaje en la AWS IoT Events consola

Al utilizar la AWS IoT Events consola, el trabajo en curso se guarda localmente en el navegador. Sin embargo, debe seleccionar Publicar para guardar su modelo de detector en AWS IoT Events. Después de publicar un modelo de detector, su trabajo publicado estará disponible en cualquier navegador que utilice para acceder a su cuenta.

Note

Si no publica su trabajo, no se guardará. Después de publicar un modelo de detector, ya no puede cambiar su nombre. Sin embargo, puede seguir modificando su definición.

Tutoriales para casos de AWS IoT Events uso

En este capítulo le mostramos cómo:

- Obtener ayuda para decidir qué estados incluir en su modelo de detector y determinar si necesita una instancia de detector o varias.
- Seguir un ejemplo que utilice la AWS CLI.
- Crear una entrada para recibir datos de telemetría de un dispositivo y un modelo de detector para supervisar el dispositivo que envía esos datos e informar de su estado.
- Revisar las restricciones y limitaciones de las entradas, los modelos de detector y el servicio de AWS IoT Events .
- Consultar un ejemplo más complejo de modelo de detector, con comentarios incluidos.

Temas

- [Uso AWS IoT Events para monitorear sus dispositivos de IoT](#)
- [step-by-step Ejemplo sencillo](#)
- [Restricciones y limitaciones del modelo de detector](#)
- [Un ejemplo comentado: el control de temperatura HVAC](#)

Uso AWS IoT Events para monitorear sus dispositivos de IoT

Puede usarlo AWS IoT Events para monitorear sus dispositivos o procesos y tomar medidas en función de eventos importantes. Para ello, siga estos pasos básicos:

Creación de entradas

Usted debe contar con un método para que sus dispositivos y procesos transmitan datos de telemetría a AWS IoT Events. Esto lo consigue enviando mensajes como entradas a AWS IoT Events. Puede enviar mensajes como entradas de varias maneras, para ello puede:

- Usa la [BatchPutMessage](#) operación.
- Definir una [rule-action de iotEvents](#) para el [motor de reglas de AWS IoT Core](#). La rule-action reenvía los datos del mensaje desde su entrada a AWS IoT Events.

- En AWS IoT Analytics, utilice la [CreateDataset](#) operación para crear un conjunto de datos `contentDeliveryRules`. Estas reglas especifican la AWS IoT Events entrada a la que se envía automáticamente el contenido del conjunto de datos.
- Defina una [iotEventsacción](#) en un `transitionEvents` evento `onExit` o modelo de AWS IoT Events `onInput` detector. La información sobre la instancia del modelo de detector y el evento que inició la acción se devuelven al sistema como una entrada con el nombre que usted especifique.

Antes de que sus dispositivos empiecen a enviar datos de esta forma, debe definir una o más entradas. Para ello, asigne un nombre a cada entrada y especifique qué campos de los datos del mensaje entrante monitorea la entrada. AWS IoT Events recibe su entrada, en forma de JSON carga útil, de muchas fuentes. Se puede actuar sobre cada entrada independiente o combinada con otras entradas para detectar eventos más complejos.

Crear un modelo de detector

Defina un modelo de detector (un modelo de su equipo o proceso) utilizando estados. Para cada estado, defina una lógica condicional (booleana) que evalúa las entradas para detectar eventos significativos. Cuando se detecta un evento, puede cambiar el estado o iniciar acciones personalizadas o predefinidas mediante otros servicios. AWS Puede definir eventos adicionales que inicien acciones al entrar o salir de un estado y, opcionalmente, al cumplirse una condición.

En este tutorial, envías un SNS mensaje de Amazon como acción cuando el modelo entra o sale de un estado determinado.

Cómo supervisar un dispositivo o proceso

Al supervisar varios dispositivos o procesos, usted especifica un campo en cada entrada que identifica el dispositivo o proceso concreto del que procede la entrada. (Vea el campo `key` en `CreateDetectorModel`). Al identificarse un nuevo dispositivo (se ve un nuevo valor en el campo de entrada identificado por la `key`), se crea un detector. (Cada detector es una instancia del modelo de detector). A continuación, el nuevo detector sigue respondiendo a las entradas procedentes de ese dispositivo hasta que su modelo de detector se actualice o elimine.

Si supervisa un único proceso (incluso con varios dispositivos o subprocesos enviando entradas), usted no especifica un campo `key` de identificación único. En este caso, se crea un único detector (instancia) al llegar la primera entrada.

Cómo enviar mensajes como entradas a su modelo de detector

Existen varias formas de enviar un mensaje desde un dispositivo o procesarlo como entrada a un AWS IoT Events detector sin necesidad de formatear el mensaje de forma adicional. En este tutorial, utilizará la AWS IoT consola para escribir una regla de [AWS IoT Events acción](#) para el motor de AWS IoT Core reglas al que reenvía los datos de sus mensajes. AWS IoT Events Para ello, usted identifica la entrada por su nombre. A continuación, seguirá utilizando la AWS IoT consola para generar algunos mensajes que se reenviarán como entradas a AWS IoT Events.

¿Cómo puede saber qué estados necesita en un modelo de detector?

Para determinar qué estados debería tener su modelo de detector, primero decida qué medidas puede usted tomar. Por ejemplo, si su automóvil funciona con gasolina, usted mira el indicador de combustible al iniciar un viaje para ver si necesita repostar. Aquí tiene una acción: decirle al conductor “debe repostar”. Su modelo de detector necesita dos estados: “el coche no necesita combustible” y “el coche sí necesita combustible”. En general, querrá definir un estado para cada acción posible, más otro para cuando no se requiera ninguna acción. Esto funciona incluso si la acción en sí misma es más complicada. Por ejemplo, puede que quiera buscar e incluir información sobre dónde encontrar la gasolinera más cercana o el precio más barato, pero esto lo hace cuando envía el mensaje “debe repostar”.

Para decidir en qué estado debe entrar a continuación, se fijará en las entradas. Las entradas contienen la información que necesita para decidir en qué estado debería estar. Para crear una entrada, usted selecciona uno o varios campos en un mensaje enviado por su dispositivo o proceso que le ayuden a decidir. En este ejemplo, necesita una entrada que le indique el nivel actual de combustible (“porcentaje de lleno”). Es posible que su coche le envíe varios mensajes diferentes, cada uno con varios campos distintos. Para crear esta entrada, debe seleccionar el mensaje y el campo que informa del nivel actual de combustible. La longitud del trayecto que va a realizar (“distancia hasta el destino”) se puede incluir en el código para simplificar las cosas; puede utilizar la distancia media de su viaje. Realizará algunos cálculos basados en la entrada (“¿A cuántos litros equivale ese porcentaje de lleno? ¿Es la distancia media del viaje mayor que los kilómetros que puede recorrer, dados los litros que tiene y su promedio de ‘kilómetros por litro’?”). Usted realiza estos cálculos y envía los mensajes en eventos.

Hasta ahora tiene dos estados y una entrada. Necesita un evento en el primer estado que realice los cálculos en función de la entrada y decida si se pasa al segundo estado. Eso es un evento de transición. (Los `transitionEvents` están en una lista de eventos `onInput` del estado. Al recibirse una entrada en este primer estado, el evento realiza una transición al segundo estado, si se cumple

la `condition` del evento). Al llegar al segundo estado, envía el mensaje en cuanto entra en él. (Usted utiliza un evento `onEnter`. Al entrar en el segundo estado, este evento envía el mensaje. No es necesario esperar a que llegue otra entrada). Hay otros tipos de eventos, pero esto es todo lo que necesita para un ejemplo sencillo.

Los otros tipos de eventos son `onExit` y `onInput`. En cuanto se recibe una entrada, y si se cumple la condición, un evento `onInput` realiza las acciones especificadas. Cuando una operación sale de su estado actual y se cumple la condición, el evento `onExit` realiza las acciones especificadas.

¿Le falta algo? Sí, ¿cómo se vuelve al primer estado “el coche no necesita combustible”? Después de llenar el depósito de gasolina, la entrada muestra el depósito lleno. En el segundo estado necesita un evento de transición para volver al primer estado, que ocurra cuando se reciba la entrada (en los eventos `onInput`: del segundo estado). Debería volver al primer estado si los cálculos muestran que ahora tiene suficiente gasolina para llegar a donde quiere ir.

Eso es lo básico. Algunos modelos de detectores se vuelven más complejos añadiendo estados que reflejan entradas importantes, no solo posibles acciones. Por ejemplo, podría tener tres estados en un modelo de detector que vigile la temperatura: un estado “normal”, un estado “demasiado caliente” y un estado “problema potencial”. Usted pasa al estado de problema potencial cuando la temperatura supera un determinado nivel, pero aún no es demasiado caliente. No querrá enviar una alarma a menos que permanezca a esta temperatura durante más de 15 minutos. Si la temperatura vuelve a la normalidad antes de ese periodo, el detector vuelve al estado normal. Si el temporizador expira, el detector pasa al estado demasiado caliente y envía una alarma, solo por precaución. Podría hacer lo mismo utilizando variables y un conjunto más complejo de condiciones de evento. Sin embargo, a menudo es más fácil utilizar otro estado para, en efecto, almacenar los resultados de sus cálculos.

¿Cómo puede saber si necesita una instancia, o varias, de un detector?

Para decidir cuántas instancias necesita, hágase la siguiente pregunta: “¿Qué le interesa saber?”. Digamos que quiere saber qué tiempo hace hoy. ¿Está lloviendo (estado)? ¿Necesita llevar un paraguas (acción)? Puede tener un sensor que informe de la temperatura, otro que informe de la humedad y otros que informen de la presión barométrica, la velocidad y dirección del viento y la precipitación. Sin embargo, debe monitorear todos estos sensores a la vez para determinar el estado del tiempo (lluvia, nieve, nublado, soleado) y la acción apropiada por realizar (llevar un paraguas o aplicarse protector solar). Sin importar el número de sensores, lo que quiere es una única instancia de detector para monitorear el estado del tiempo e informarle de las medidas que debe tomar.

Pero si usted es el meteorólogo de su región, podría tener varias instancias de estos conjuntos de sensores, situados en diferentes lugares de la región. La gente de cada lugar necesita saber qué

tiempo hace en ese lugar. En este caso, necesita múltiples instancias de su detector. Los datos comunicados por cada sensor en cada ubicación deben incluir un campo que usted haya designado como campo key. Este campo permite a AWS IoT Events crear una instancia del detector para la zona y, a continuación, seguir dirigiendo esta información a esa instancia de detector a medida que siga llegando. ¡Se acabaron los cabellos estropeados o las narices quemadas por el sol!

Básicamente, necesita una única instancia de detector si tiene una sola situación (un proceso o una ubicación) que monitorear. Si tiene muchas situaciones (ubicaciones, procesos) que monitorear, necesitará varias instancias del detector.

step-by-step Ejemplo sencillo

En este ejemplo, utilizamos los AWS CLI comandos de AWS IoT Events APIs uso para crear un detector que modele dos estados de un motor: un estado normal y un estado de sobrepresión.

Cuando la presión medida en el motor supera un determinado umbral, el modelo pasa al estado de sobrepresión y envía un mensaje del Amazon Simple Notification Service (AmazonSNS) para avisar al técnico de la situación. Cuando la presión cae por debajo del umbral durante tres lecturas de presión consecutivas, el modelo vuelve al estado normal y envía otro SNS mensaje de Amazon como confirmación de que la condición ha pasado. Necesitamos tres lecturas consecutivas por debajo del umbral de presión para eliminar posibles tartamudeos de mensajes de sobrepresión/normalidad en caso de una fase de recuperación no lineal o una lectura de recuperación anómala puntual.

A continuación, se resumen los pasos para crear el detector.

Creación de entradas.

Para supervisar sus dispositivos y procesos, deben tener una forma de transferir datos de telemetría a AWS IoT Events. Esto se hace enviando mensajes como entradas a AWS IoT Events. Puede hacer esto de varias formas:

- Usa la [BatchPutMessage](#) operación. Este método es fácil, pero requiere que sus dispositivos o procesos puedan acceder a AWS IoT Events API través de un SDK o el AWS CLI.
- En AWS IoT Core, escribe una regla de [AWS IoT Events acción](#) para el motor de AWS IoT Core reglas al que reenvía los datos de tus mensajes. AWS IoT Events Esto identifica la entrada por su nombre. Utilice este método si sus dispositivos o procesos pueden enviar mensajes a través AWS IoT Core de ellos o ya lo están haciendo. Este método suele requerir menos potencia de cálculo de un dispositivo.

- En AWS IoT Analytics, utilice la [CreateDataset](#) operación para crear un conjunto de datos `contentDeliveryRules` que especifique la AWS IoT Events entrada, donde el contenido del conjunto de datos se envía automáticamente. Utilice este método si desea controlar sus dispositivos o procesos en función de datos agregados o analizados en AWS IoT Analytics.

Para que sus dispositivos puedan enviar datos de este modo, debe definir una o más entradas. Para ello, asigne un nombre a cada entrada y especifique qué campos de los datos del mensaje entrante monitorea la entrada.

Crear un modelo de detector

Cree un modelo de detector (un modelo de su equipo o proceso) utilizando estados. Para cada estado, defina una lógica condicional (booleana) que evalúe las entradas para detectar eventos significativos. Cuando se detecta un evento, puede cambiar el estado o iniciar acciones personalizadas o predefinidas mediante otros AWS servicios. Puede definir eventos adicionales que inicien acciones al entrar o salir de un estado y, opcionalmente, al cumplirse una condición.

Supervisión de varios dispositivos o procesos

Si monitorea varios dispositivos o procesos y desea realizar un seguimiento de cada uno de ellos por separado, especifique un campo en cada entrada que identifique el dispositivo o proceso en particular del que proviene la entrada. Consulte el campo `key` en `CreateDetectorModel`. Al identificarse un nuevo dispositivo (se ve un nuevo valor en el campo de entrada identificado por la `key`), se crea una instancia de detector. La nueva instancia de detector sigue respondiendo a las entradas procedentes de ese dispositivo en particular hasta que se actualice o elimine su modelo de detector. Tendrá tantos detectores únicos (instancias) como valores únicos haya en los campos `key` de entrada.

Monitoreo de un único dispositivo o proceso

Si supervisa un único proceso (incluso con varios dispositivos o subprocesos enviando entradas), usted no especifica un campo `key` de identificación único. En este caso, se crea un único detector (instancia) al llegar la primera entrada. Por ejemplo, puede tener sensores de temperatura en cada habitación de una casa, pero solo una HVAC unidad para calentar o enfriar toda la casa. Por tanto, solo puede controlarla como un proceso único, incluso si el ocupante de cada habitación quiere que prevalezca su voto (opinión).

Envío de mensajes desde sus dispositivos o procesos como entradas a su modelo de detector

Describimos las diversas formas de enviar un mensaje desde un dispositivo o proceso como entrada a un AWS IoT Events detector en entradas. Después de crear las entradas y construir el modelo de detector, ya está listo para empezar a enviar datos.

Note

Al crear un modelo de detector, o actualizar uno existente, transcurren varios minutos antes de que el modelo de detector nuevo o actualizado comience a recibir mensajes y a crear detectores (instancias). Al actualizar el modelo de detector, es posible que durante este periodo siga observando un comportamiento basado en la versión anterior.

Temas

- [Creación de una entrada para capturar datos del dispositivo](#)
- [Creación de un modelo de detector para representar los estados de los dispositivos](#)
- [Envío de mensajes como entradas a un detector](#)

Creación de una entrada para capturar datos del dispositivo

Como ejemplo, suponga que sus dispositivos envían mensajes con el siguiente formato.

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

Puede crear una entrada para capturar los `pressure` datos y los `motorid` (que identifique el dispositivo específico que envió el mensaje) mediante el siguiente AWS CLI comando.

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

El archivo `pressureInput.json` contiene lo siguiente.

```
{
  "inputName": "PressureInput",
  "inputDescription": "Pressure readings from a motor",
  "inputDefinition": {
    "attributes": [
```

```
    { "jsonPath": "sensorData.pressure" },  
    { "jsonPath": "motorid" }  
  ]  
}  
}
```

Cuando cree sus propias entradas, recuerde recopilar primero mensajes de ejemplo como JSON archivos desde sus dispositivos o procesos. Puede utilizarlos para crear una entrada desde la consola o el CLI.

Creación de un modelo de detector para representar los estados de los dispositivos

En [Creación de una entrada para capturar datos del dispositivo](#) creó una input basada en un mensaje que informa de los datos de presión de un motor. Para continuar con el ejemplo, aquí tiene un modelo de detector que responde a un evento de sobrepresión en un motor.

Usted crea dos estados: "Normal" y "Dangerous". Cada detector (instancia) entra en el estado "Normal" cuando se crea. La instancia se crea al llegar una entrada con un valor único para la key "motorid".

Si la instancia del detector recibe una lectura de presión de 70 o más, entra en el estado Dangerous «» y envía un SNS mensaje de Amazon como advertencia. Si las lecturas de presión vuelven a la normalidad (menos de 70) durante tres entradas consecutivas, el detector vuelve al estado «Normal» y envía otro SNS mensaje de Amazon confirmando que todo está bien.

En este ejemplo de modelo de detector se supone que ha creado dos SNS temas de Amazon cuyos nombres de recursos de Amazon (ARNs) se muestran en la definición como "targetArn": "arn:aws:sns:us-east-1:123456789012:underPressureAction" y "targetArn": "arn:aws:sns:us-east-1:123456789012:pressureClearedAction".

Para obtener más información, consulte la [Guía para desarrolladores de Amazon Simple Notification Service](#) y, más específicamente, la documentación de la [CreateTopic](#) operación en la API referencia de Amazon Simple Notification Service.

En este ejemplo también se supone que ha creado un rol AWS Identity and Access Management (IAM) con los permisos adecuados. El rol ARN de este rol se muestra en la definición del modelo de detector como "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole". Siga los pasos que se indican [Configurar permisos para AWS IoT Events](#) para crear esta función y copie ARN la función en el lugar correspondiente de la definición del modelo de detector.

Puede crear el modelo de detector mediante el siguiente AWS CLI comando.

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

El archivo "motorDetectorModel.json" contiene lo siguiente.

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



```

        }
      }
    ]
  },
  "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"
}

```

Envío de mensajes como entradas a un detector

Ahora ha definido una entrada que identifica los campos importantes en los mensajes enviados desde un dispositivo (consulte [Creación de una entrada para capturar datos del dispositivo](#)). En la

sección anterior, creó un detector `model` que responde a un evento de sobrepresión en un motor (consulte [Creación de un modelo de detector para representar los estados de los dispositivos](#)).

Para completar el ejemplo, envíe mensajes desde un dispositivo (en este caso un ordenador con la AWS CLI instalada) como entradas al detector.

Note

Al crear un modelo de detector o actualizar uno existente, transcurren varios minutos antes de que el modelo de detector nuevo o actualizado comience a recibir mensajes y a crear detectores (instancias). Si usted actualiza el modelo de detector, es posible que durante este periodo siga observando un comportamiento basado en la versión anterior.

Utilice el siguiente AWS CLI comando para enviar un mensaje con datos que superen el umbral.

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

El archivo `highPressureMessage.json` contiene lo siguiente.

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

Debe cambiar el `messageId` en cada mensaje enviado. Si no lo cambias, el AWS IoT Events sistema deduplicará los mensajes. AWS IoT Events ignora un mensaje si es `messageID` igual a otro mensaje que se envió en los últimos cinco minutos.

A este punto, se crea un detector (instancia) para monitorear los eventos del motor `Fulton-A32`. Este detector entra en el estado `Normal` cuando se crea. Pero como hemos enviado un valor de presión superior al umbral, pasa de inmediato al estado `Dangerous`. Mientras lo hace, el detector

envía un mensaje al SNS punto final de Amazon en el que se ARN encuentra `arn:aws:sns:us-east-1:123456789012:underPressureAction`.

Ejecute el siguiente AWS CLI comando para enviar un mensaje con datos que estén por debajo del umbral de presión.

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

El archivo `normalPressureMessage.json` contiene lo siguiente.

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

Debe cambiar el `messageId` en el archivo cada vez que invoque el comando `BatchPutMessage` en un periodo de cinco minutos. Envíe el mensaje dos veces más. Después de enviar el mensaje tres veces, el detector (instancia) del motor «Fulton-A32» envía un mensaje al SNS punto final de Amazon `arn:aws:sns:us-east-1:123456789012:pressureClearedAction` y vuelve a entrar en el "Normal" estado.

Note

Puede enviar varios mensajes a la vez con `BatchPutMessage`. Sin embargo, no se garantiza el orden en que se procesan estos mensajes. Para garantizar que los mensajes (entradas) se procesen en orden, envíelos de uno en uno y espere una respuesta satisfactoria cada vez que los API llame.

A continuación se muestran ejemplos de cargas útiles de SNS mensajes creadas por el modelo de detector descrito en esta sección.

en caso de “Superación del umbral de presión”

```
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":{
        "pressureThresholdBreach":3
      },
      "timers":{}
    }
  },
  "eventName":"Pressure Threshold Breached"
}
```

en caso de “Restablecimiento de la presión normal”

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

Si ha definido algún temporizador, su estado actual también se muestra en las cargas útiles de los SNS mensajes.

Las cargas útiles de los mensajes contienen información sobre el estado del detector (instancia) en el momento en que se envió el mensaje (es decir, en el momento en que se ejecutó la SNS acción). Puede utilizar la operación https://docs.aws.amazon.com/iotevents/latest/apireference/API_iotevents-data_DescribeDetector.html para obtener información similar sobre el estado del detector.

Restricciones y limitaciones del modelo de detector

Es importante tener en cuenta los siguientes aspectos al crear un modelo de detector.

Cómo usar el campo **actions**

El campo **actions** es una lista de objetos. Puede tener más de un objeto, pero solo se permite una acción en cada objeto.

Example

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

```
    }  
  }  
]
```

Cómo usar el campo **condition**

La `condition` es obligatoria para `transitionEvents` y opcional en otros casos.

Si el campo `condition` no está presente, equivale a `"condition": true`.

El resultado de la evaluación de una expresión de condición debe ser un valor booleano. Si el resultado no es un valor booleano, equivale a `false` y no se iniciarán las `actions` ni la transición al `nextState` en el evento.

Disponibilidad de valores variables

Por defecto, si se fija el valor de una variable en un evento, no se dispone de su nuevo valor ni se utiliza para evaluar condiciones en otros eventos del mismo grupo. El nuevo valor no está disponible ni se utiliza en una condición de evento en el mismo campo `onInput`, `onEnter` o `onExit`.

Establezca el parámetro `evaluationMethod` en la definición del modelo de detector para cambiar este comportamiento. Si se establece `evaluationMethod` en `SERIAL`, las variables se actualizan y las condiciones de evento se evalúan en el orden en que estén definidos los eventos. Caso contrario, si se establece `evaluationMethod` en `BATCH` o lo asume por defecto, las variables dentro de un estado se actualizan y los eventos dentro de un estado se realizan solo después de que se hayan evaluado todas las condiciones del evento.

En el estado "Dangerous", en el campo `onInput`, `"$variable.pressureThresholdBreach"` se decrementa en uno en el evento "Pressure Okay" al cumplirse la condición (cuando la entrada actual tiene una presión menor o igual que 70).

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

```

    }
  }
]
}

```

El detector debería volver al estado "Normal" cuando "\$variable.pressureThresholdBreach" llegue a 0 (es decir, cuando el detector haya recibido tres lecturas de presión consecutivas menores o iguales que 70). El evento "BackToNormal" en transitionEvents debe comprobar que "\$variable.pressureThresholdBreach" sea menor o igual que 1 (no 0) y además volver a verificar que el valor actual dado por "\$input.PressureInput.sensorData.pressure" sea menor o igual que 70.

```

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

```

Caso contrario, si la condición comprueba solo el valor de la variable, dos lecturas normales seguidas de una lectura de sobrepresión cumplirían la condición y se volvería al estado "Normal". La condición mira el valor que se le dio a "\$variable.pressureThresholdBreach" durante la última vez que se procesó una entrada. El valor de la variable se restablece a 3 en el evento "Overpressurized", pero recuerde que este nuevo valor aún no está disponible para ninguna condition.

Por defecto, cada vez que un control entra en el campo onInput, una condition solo puede ver el valor de una variable tal y como era al inicio del procesamiento de la entrada, antes de que cualquier acción especificada en onInput lo modifique. Lo mismo ocurre para onEnter y onExit. Cualquier cambio realizado en una variable al entrar o salir del estado no está disponible para otras condiciones especificadas en los mismos campos onEnter o onExit.

Latencia al actualizar un modelo de detector

Si actualiza, elimina y vuelve a crear un modelo de detector (consulte [UpdateDetectorModel](#)), pasará un tiempo antes de que se eliminen todos los detectores generados (instancias) y se

utilice el nuevo modelo para volver a crear los detectores. Se recrean una vez que el nuevo modelo de detector entra en función y llegan nuevas entradas. Durante este tiempo, es posible que los detectores generados por la versión anterior del modelo de detector sigan procesando entradas. Durante este período, es posible que siga recibiendo las alertas definidas por el modelo de detector anterior.

Espacios en las claves de entrada

Se permiten espacios en las claves de entrada, pero las referencias a la clave deben estar entre tildes graves, tanto en la definición del atributo de entrada como cuando se hace referencia al valor de la clave en una expresión. Por ejemplo, dada una carga de mensaje como la siguiente:

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

Utilice lo siguiente para definir la entrada.

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

En una expresión condicional, debe hacer referencia al valor de cualquiera de estas claves utilizando también tildes graves.

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

Un ejemplo comentado: el control de temperatura HVAC

Algunos de los siguientes JSON archivos de ejemplo tienen comentarios en línea, por lo que no son válidos JSON. Las versiones completas de estos ejemplos, sin comentarios, están disponibles en [Ejemplo: uso del control de HVAC temperatura](#).

Introducción

En este ejemplo se implementa un modelo de control de termostato que le permite hacer lo siguiente.

- Definir un único modelo de detector que pueda utilizarse para supervisar y controlar múltiples áreas. Se crea una instancia de detector para cada área.
- Ingerir datos de temperatura de múltiples sensores en cada área de control.
- Cambiar el punto de consigna de temperatura para un área.
- Establecer parámetros operativos para cada área y restablecerlos mientras se utiliza la instancia.
- Añadir o eliminar de forma dinámica sensores en un área.
- Especificar un tiempo de ejecución mínimo para proteger las unidades de calefacción y refrigeración.
- Rechazar lecturas anómalas de los sensores.
- Definir puntos de consigna de emergencia que activen de inmediato la calefacción o la refrigeración si alguno de los sensores informa de una temperatura superior o inferior a un umbral determinado.
- Notificar lecturas anómalas y picos de temperatura.

Definiciones de entrada para modelos de detectores

Queremos crear un modelo de detector que podamos utilizar para monitorear y controlar la temperatura en varias áreas diferentes. Cada área puede tener varios sensores que informen de la temperatura. Suponemos que cada área cuenta con una unidad de calefacción y otra de refrigeración que se pueden encender o apagar para controlar la temperatura del área. Cada área está controlada por una instancia del detector.

Dado que las distintas áreas que monitoreamos y controlamos podrían tener características diferentes que exijan parámetros de control distintos, definimos las 'seedTemperatureInput' para que proporcionen esos parámetros para cada área. Cuando enviamos uno de estos mensajes

de entrada a AWS IoT Events, se crea una nueva instancia del modelo de detector que tiene los parámetros que queremos utilizar en esa área. Esta es la definición de esta entrada.

CLI comando:

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

Archivo: 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" }
    ]
  }
}
```

Respuesta:

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

Notas

- Se crea una nueva instancia del detector para cada 'areaId' único recibido en cualquier mensaje. Consulte el campo 'key' en la definición de 'areaDetectorModel'.
- La temperatura media puede variar de la 'desiredTemperature' en un 'allowedError' antes de que se activen las unidades de calefacción o refrigeración del área.
- Si cualquier sensor notifica una temperatura superior a 'rangeHigh', el detector informa de un pico y de inmediato pone en marcha la unidad de refrigeración.
- Si cualquier sensor notifica una temperatura inferior a 'rangeLow', el detector informa de un pico y de inmediato pone en marcha la unidad de calefacción.
- Si cualquier sensor notifica una temperatura superior a 'anomalousHigh' o inferior a 'anomalousLow', el detector informa de una lectura anómala de sensor, pero ignora la lectura de temperatura informada.
- El 'sensorCount' le indica al detector cuántos sensores están emitiendo información del área. El detector calcula la temperatura media del área dando el factor de ponderación apropiado a cada lectura de temperatura que recibe. Por ello, el detector no tendrá que hacer un seguimiento de lo que informa cada sensor y el número de sensores se puede cambiar de manera dinámica según sea necesario. Sin embargo, si un sensor individual deja de emitir, el detector no lo sabrá o no tendrá en cuenta esta circunstancia. Le recomendamos que cree otro modelo de detector específico para monitorear el estado de conexión de cada sensor. Disponer de dos modelos de detector complementarios simplifica el diseño de ambos.
- El valor de 'noDelay' puede ser true o false. Una unidad de calefacción o refrigeración, tras encenderse, debe permanecer encendida durante un tiempo mínimo determinado para proteger la integridad de la unidad y alargar su vida útil. Si 'noDelay' se configura en false, la instancia del detector aplica un retardo antes de apagar las unidades de refrigeración y calefacción a fin de garantizar que funcionen durante el tiempo mínimo. El número de segundos de retardo se ha codificado en la definición del modelo de detector porque no podemos utilizar un valor variable para establecer un temporizador.

Se utiliza 'temperatureInput' para transmitir los datos del sensor a una instancia del detector.

CLI comando:

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

Archivo: temperatureInput.json

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

Respuesta:

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

Notas

- Una instancia del detector de ejemplo no utiliza el 'sensorId' para controlar ni monitorear un sensor de manera directa. Se transfiere automáticamente a las notificaciones enviadas por la instancia del detector. A partir de ahí, se lo puede utilizar para identificar sensores que estén fallando (p. ej., un sensor que envía regularmente lecturas anómalas podría estar a punto de fallar) o que han dejado de emitir (cuando se utiliza como entrada de un modelo de detector adicional que controla el latido del dispositivo). El 'sensorId' también puede ayudar a identificar zonas cálidas o frías en un área si sus lecturas difieren regularmente de la media.
- El 'areaId' se utiliza para dirigir los datos del sensor a la instancia de detector apropiada. Se crea una instancia de detector para cada 'areaId' único recibido en cualquier mensaje. Consulte el campo 'key' en la definición de 'areaDetectorModel'.

Crear una definición de modelo de detector

El ejemplo de 'areaDetectorModel' tiene comentarios en línea.

CLI comando:

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

Archivo: 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:cool0ff"
      }
    },
    {
      "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:cool0n"
          }
        },
        {
          "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
lost.
          // 'timeout()' is used in such mixed conditionals, its value is

          "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:cool0ff"
          }
        },
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:heat0n"
          }
        }
      ],
    }
  ]
}
```

```

        "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:cool0ff"
            }
        },
        {
            "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"
}

```

Respuesta:

```

{
  "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"  
  }  
}
```

Se usa BatchUpdateDetector para actualizar

Puede utilizar la operación `BatchUpdateDetector` para poner una instancia de detector en un estado conocido, incluyendo valores de temporizadores y de variables. En el siguiente ejemplo, la operación `BatchUpdateDetector` restablece los parámetros operativos de un área que está bajo monitoreo y control de temperatura. Esta operación le permite hacerlo sin tener que eliminar y recrear ni actualizar el modelo de detector.

CLI comando:

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

Archivo: `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": [
    ]
}
]
}
}

```

Respuesta:

```

{
  "batchUpdateDetectorErrorEntries": []
}

```

BatchPutMessageÚselo para entradas

Example 1

Utilice la operación BatchPutMessage para enviar un mensaje "seedTemperatureInput" que establezca los parámetros operativos de un área determinada bajo monitoreo y control de la temperatura. Cualquier mensaje AWS IoT Events que reciba un mensaje nuevo "areaId" provoca la creación de una nueva instancia de detector. Sin embargo, la nueva instancia de detector no cambiará de estado a "idle" ni comenzará a supervisar la temperatura ni a controlar las unidades de calefacción o refrigeración hasta que se reciba un mensaje "seedTemperatureInput" para la nueva área.

CLI comando:

```

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

```

Archivo: 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}"
}
]
}

```

Respuesta:

```

{
  "BatchPutMessageErrorEntries": []
}

```

Example

2

Utilice la operación `BatchPutMessage` para enviar un mensaje "temperatureInput" a fin de notificar los datos de un sensor de temperatura en un área de control y monitoreo determinada.

CLI comando:

```

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

```

Archivo: temperatureExample.json

```

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

```

Respuesta:

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

Example 3

Utilice la operación `BatchPutMessage` para enviar un mensaje `"seedTemperatureInput"` a fin de cambiar el valor de la temperatura deseada en un área determinada.

CLI comando:

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

Archivo: seedSetDesiredTemp.json

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

Respuesta:

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

Ingiera mensajes MQTT

Si los recursos informáticos de sus sensores no pueden "BatchPutMessage" API utilizarlos, pero pueden enviar sus datos al intermediario de AWS IoT Core mensajes mediante un MQTT cliente ligero, puede crear una regla AWS IoT Core temática para redirigir los datos de los

mensajes a una AWS IoT Events entrada. La siguiente es una definición de una regla AWS IoT Events temática que toma los campos "areaId" y de "sensorId" entrada del MQTT tema y el "sensorData.temperature" campo del campo de carga "temp" del mensaje e incorpora estos datos a nuestro. AWS IoT Events "temperatureInput"

CLI comando:

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

Archivo: 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/anotherRole"
        }
      }
    ],
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23"
  }
}
```

Respuesta: [none]

Si el sensor envía un mensaje sobre el tema "update/temperature/Area51/03" con la siguiente carga.

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

Esto hace que los datos se ingieran AWS IoT Events como si se hubiera realizado la siguiente "BatchPutMessage" API llamada.

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

Archivo: spooferExample.json

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

Generar SNS mensajes de Amazon

Los siguientes son ejemplos de SNS mensajes generados por la instancia del "Area51" detector.

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

      "detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"}, "eventTime":1557520274729,
      "inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":{
      "stateName":"start","variables":{
      "sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature":24.5}}}, "eventName":"resetHeatCool"}
```

```
Cooling system off command> {"eventTime":1557520274729,"payload":{
  "actionExecutionId":"98f6a1b5-8f40-3cdb-9256-93afd4d66192","detector":{
    "detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"}, "eventTime":1557520274729,
    "inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message"},"state":{
    "stateName":"start","variables":{
    "sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature":24.5}}}, "eventName":"resetHeatCool"}
```

Configure el DescribeDetector API

Puede utilizar la operación `DescribeDetector` para ver el estado actual, los valores de las variables y los temporizadores de una instancia de detector.

CLI comando:

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

Respuesta:

```
{
  "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"
```



```
}
}
```

Usa el motor de AWS IoT Core reglas

Las siguientes reglas vuelven a publicar AWS IoT Core MQTT los mensajes como mensajes de solicitud de actualización paralelos. Suponemos que AWS IoT Core las cosas están definidas para una unidad de calefacción y una unidad de refrigeración para cada área controlada por el modelo de detector. En este ejemplo, hemos definido cosas denominadas "Area51HeatingUnit" y "Area51CoolingUnit".

CLI comando:

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

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

Respuesta: [vacío]

CLI comando:

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

Archivo: 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"
        }
      }
    ]
  }
}
```

Respuesta: [vacío]

CLI comando:

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

Archivo: 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"
        }
      }
    ]
  }
}

```

Respuesta: [vacío]

CLI comando:

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

Archivo: 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"
        }
      }
    ]
  }
}

```

Respuesta: [vacío]

Acciones compatibles para recibir datos y activar acciones

AWS IoT Events puede activar acciones cuando detecta un evento específico o un evento de transición. Puede definir acciones integradas para usar un temporizador o establecer una variable, o enviar datos a otros AWS recursos.

Note

Al definir una acción en un modelo de detector, puede utilizar expresiones para parámetros que sean de tipo cadena de datos. Para obtener más información, consulte [Expresiones](#).

AWS IoT Events admite las siguientes acciones que permiten usar un temporizador o establecer una variable:

- [setTimer](#) para crear un temporizador.
- [resetTimer](#) para restablecer el temporizador.
- [clearTimer](#) para eliminar el temporizador.
- [setVariable](#) para crear una variable.

AWS IoT Events admite las siguientes acciones que le permiten trabajar con AWS los servicios:

- [iotTopicPublish](#) publicar un mensaje sobre un MQTT tema.
- [iotEvents](#) para enviar datos a AWS IoT Events como valor de entrada.
- [iotSiteWise](#) para enviar datos a una propiedad de recurso en AWS IoT SiteWise.
- [dynamoDB](#) para enviar datos a una tabla de Amazon DynamoDB.
- [dynamoDBv2](#) para enviar datos a una tabla de Amazon DynamoDB.
- [firehose](#) para enviar datos a una transmisión de Amazon Data Firehose.
- [lambda](#) para invocar una función de AWS Lambda .
- [sns](#) para enviar datos como notificación de inserción.
- [sqs](#) para enviar datos a una SQS cola de Amazon.

Utilice el temporizador AWS IoT Events incorporado y las acciones variables

AWS IoT Events admite las siguientes acciones que permiten utilizar un temporizador o configurar una variable:

- [setTimer](#) para crear un temporizador.
- [resetTimer](#) para restablecer el temporizador.
- [clearTimer](#) para eliminar el temporizador.
- [setVariable](#) para crear una variable.

Establecer la acción del temporizador

Set timer action

La acción `setTimer` le permite crear un temporizador con una duración en segundos.

More information (2)

Al crear un temporizador, debe especificar los siguientes parámetros.

timerName

El nombre del temporizador.

durationExpression

(Opcional) Duración, en segundos, del temporizador.

El resultado evaluado de una expresión de duración se redondea al número entero menor más cercano. Por ejemplo, si configura el temporizador en 60,99 segundos, el resultado evaluado de la expresión de duración es 60 segundos.

Para obtener más información, consulte [SetTimerAction](#) la AWS IoT Events API Referencia.

Restablecer la acción de temporizador

Reset timer action

La acción `resetTimer` le permite ajustar el temporizador en el resultado previamente evaluado de la expresión de duración.

More information (1)

Al restablecer un temporizador, debe especificar el siguiente parámetro.

timerName

El nombre del temporizador.

AWS IoT Events no reevalúa la expresión de duración al restablecer el temporizador.

Para obtener más información, consulte [ResetTimerAction](#) la AWS IoT Events API Referencia.

Eliminar la acción del temporizador

Clear timer action

La acción `clearTimer` le permite eliminar un temporizador existente.

More information (1)

Al eliminar un temporizador, debe especificar el siguiente parámetro.

timerName

El nombre del temporizador.

Para obtener más información, consulte [ClearTimerAction](#) la AWS IoT Events API Referencia.

Establecer una acción de variable

Set variable action

La acción `setVariable` le permite crear una variable con un valor específico.

More information (2)

Al crear una variable, debe especificar los siguientes parámetros.

variableName

El nombre de la variable.

value

El nuevo valor de la variable.

Para obtener más información, consulte [SetVariableAction](#) la AWS IoT Events API Referencia.

Trabajando con otros AWS servicios

AWS IoT Events admite las siguientes acciones que le permiten trabajar con AWS los servicios:

- [iotTopicPublish](#) publicar un mensaje sobre un MQTT tema.
- [iotEvents](#) para enviar datos a AWS IoT Events como valor de entrada.
- [iotSiteWise](#) para enviar datos a una propiedad de recurso en AWS IoT SiteWise.
- [dynamoDB](#) para enviar datos a una tabla de Amazon DynamoDB.
- [dynamoDBv2](#) para enviar datos a una tabla de Amazon DynamoDB.
- [firehose](#) para enviar datos a una transmisión de Amazon Data Firehose.
- [lambda](#) para invocar una función de AWS Lambda .
- [sns](#) para enviar datos como notificación de inserción.
- [sqs](#) para enviar datos a una SQS cola de Amazon.

Important

- Debes elegir la misma AWS región para ambas AWS IoT Events y los AWS servicios con los que quieres trabajar. Para ver una lista completa de las regiones admitidas, consulte [Puntos de conexión y cuotas de AWS IoT Events](#) en la Referencia general de Amazon Web Services.

- Debe usar la misma AWS región al crear otros AWS recursos para las AWS IoT Events acciones. Si cambias de AWS región, es posible que tengas problemas para acceder a los AWS recursos.

De forma predeterminada, AWS IoT Events genera una carga útil estándar JSON para cualquier acción. Esta carga de acción contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo de detector y el evento que desencadenó la acción. Para configurar la carga de acción, puede usar una expresión de contenido. Para obtener más información, consulte [Expresiones para filtrar, transformar y procesar datos de eventos](#) y el tipo de datos de [carga útil](#) en la AWS IoT Events API Referencia.

AWS IoT Core

IoT topic publish action

La AWS IoT Core acción le permite publicar un MQTT mensaje a través del intermediario de AWS IoT mensajes. Para ver una lista completa de las regiones admitidas, consulte [Puntos de conexión y cuotas de AWS IoT Core](#) en la Referencia general de Amazon Web Services.

El agente de AWS IoT mensajes conecta a AWS IoT los clientes mediante el envío de mensajes desde los clientes publicadores a los clientes suscritos. Para obtener más información, consulte [Protocolos de comunicación de dispositivos](#) en la Guía para AWS IoT desarrolladores.

More information (2)

Al publicar un MQTT mensaje, debe especificar los siguientes parámetros.

mqttTopic

El MQTT tema que recibe el mensaje.

Puede definir el nombre de un MQTT tema de forma dinámica en tiempo de ejecución mediante variables o valores de entrada creados en el modelo de detector.

payload

(Opcional) La carga predeterminada contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo detector y el evento que desencadenó la acción.

También puede personalizar la carga. Para obtener más información, consulte [Carga útil](#) en la AWS IoT Events APIreferencia.

Note

Asegúrese de que la política adjunta a su función de AWS IoT Events servicio conceda el `iot:Publish` permiso. Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [lotTopicPublishAction](#) la AWS IoT Events API Referencia.

AWS IoT Events

IoT Events action

La AWS IoT Events acción le permite enviar datos a AWS IoT Events como entrada. Para ver una lista completa de las regiones admitidas, consulte [Puntos de conexión y cuotas de AWS IoT Events](#) en la Referencia general de Amazon Web Services.

AWS IoT Events le permite monitorizar sus equipos o flotas de dispositivos para detectar fallos o cambios en el funcionamiento y activar acciones cuando se produzcan dichos eventos. Para obtener más información, consulte [¿Qué es? AWS IoT Events](#) en la Guía para AWS IoT Events desarrolladores.

More information (2)

Al enviar datos a AWS IoT Events, debe especificar los siguientes parámetros.

inputName

El nombre de la AWS IoT Events entrada que recibe los datos.

payload

(Opcional) La carga predeterminada contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo detector y el evento que desencadenó la acción. También puede personalizar la carga. Para obtener más información, consulte [Carga útil](#) en la AWS IoT Events API referencia.

Note

Asegúrese de que la política adjunta a su función de AWS IoT Events servicio conceda el `iotevents:BatchPutMessage` permiso. Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [lotEventsAction](#) la AWS IoT Events API Referencia.

AWS IoT SiteWise

IoT SiteWise action

La AWS IoT SiteWise acción le permite enviar datos a una propiedad de un activo en AWS IoT SiteWise. Para ver una lista completa de las regiones admitidas, consulte [Puntos de conexión y cuotas de AWS IoT SiteWise](#) en la Referencia general de Amazon Web Services.

AWS IoT SiteWise es un servicio gestionado que le permite recopilar, organizar y analizar datos de equipos industriales a escala. Para obtener más información, consulte [¿Qué es AWS IoT SiteWise?](#) en la Guía del usuario de AWS IoT SiteWise .

More information (11)

Al enviar datos a una propiedad de un activo AWS IoT SiteWise, debe especificar los siguientes parámetros.

Important

Para recibir los datos, debe utilizar una propiedad de activo existente en AWS IoT SiteWise.

- Si utiliza la AWS IoT Events consola, debe especificar si desea `propertyAlias` identificar la propiedad del activo objetivo.
- Si utiliza la AWS CLI, debe especificar una `propertyAlias` o ambas `assetId` e `propertyId` identificar la propiedad del activo objetivo.

Para obtener más información, consulte [Mapeo de flujos de datos industriales a propiedades de activos](#) en la Guía del usuario de AWS IoT SiteWise .

propertyAlias

(Opcional) El alias de la propiedad de activo. También puede especificar una expresión.

assetId

(Opcional) El ID del activo que tiene la propiedad especificada. También puede especificar una expresión.

propertyId

(Opcional) ID de la propiedad de activo. También puede especificar una expresión.

entryId

(Opcional) Un identificador único para esta entrada. Puede utilizar el ID de entrada para realizar un seguimiento de qué entrada de datos provoca un error en caso de que se produzca un fallo. El valor predeterminado es un nuevo identificador único. También puede especificar una expresión.

propertyValue

Una estructura que contenga los detalles sobre el valor de propiedad.

quality

(Opcional) La calidad del valor de la propiedad de activo. El valor debe ser GOOD, BAD o UNCERTAIN. También puede especificar una expresión.

timestamp

(Opcional) Una estructura que contenga la información de marca temporal. Si no especifica este valor, el predeterminado es el momento del evento.

timeInSeconds

La marca temporal, en segundos, en formato de tiempo Unix. El rango válido es de 1 a 31556889864403199. También puede especificar una expresión.

offsetInNanos

(Opcional) El desplazamiento en nanosegundos convertido de `timeInSeconds`. El rango válido es de 0 a 999999999. También puede especificar una expresión.

value

Estructura que contiene un valor de propiedad de activo.

⚠ Important

Debe especificar uno de los siguientes tipos de valor, dependiendo del valor `dataType` de la propiedad de activo especificada. Para obtener más información, consulte [AssetProperty](#) la AWS IoT SiteWise API Referencia.

booleanValue

(Opcional) El valor de la propiedad de activo es un valor booleano que debe ser TRUE o FALSE. También puede especificar una expresión. Si utiliza una expresión, el resultado evaluado debe ser un valor booleano.

doubleValue

(Opcional) El valor de la propiedad de activo es un valor doble. También puede especificar una expresión. Si utiliza una expresión, el resultado evaluado debe ser un valor doble.

integerValue

(Opcional) El valor de la propiedad de activo es un entero. También puede especificar una expresión. Si utiliza una expresión, el resultado evaluado debe ser un entero.

stringValue

(Opcional) El valor de la propiedad de activo es una cadena. También puede especificar una expresión. Si utiliza una expresión, el resultado evaluado debe ser una cadena.

📘 Note

Asegúrese de que la política asociada a su función de AWS IoT Events servicio le conceda el `iotsitewise:BatchPutAssetPropertyValue` permiso. Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [lotSiteWiseAction](#) la AWS IoT Events API Referencia.

Amazon DynamoDB

DynamoDB action

La acción Amazon DynamoDB le permite enviar datos a una tabla de DynamoDB. Una columna de la tabla de DynamoDB recibe todos los pares atributo-valor de la carga de acción que especifique. Para ver una lista de las regiones admitidas, consulte [Puntos de conexión y cuotas de Amazon DynamoDB](#) en la Referencia general de Amazon Web Services.

Amazon DynamoDB es un servicio SQL sin base de datos totalmente gestionado que proporciona un rendimiento rápido y predecible con una escalabilidad perfecta. Para obtener más información, consulte [¿Qué es DynamoDB?](#) en la Guía para desarrolladores de Amazon DynamoDB.

More information (10)

Al enviar datos a una columna de una tabla de DynamoDB, debe especificar los siguientes parámetros.

tableName

El nombre de la tabla de DynamoDB que recibe los datos. El valor `tableName` debe coincidir con el nombre de tabla de la tabla de DynamoDB. También puede especificar una expresión.

hashKeyField

El nombre de la clave hash (también denominada clave de partición). El valor `hashKeyField` debe coincidir con la clave de partición de la tabla de DynamoDB. También puede especificar una expresión.

hashKeyType

(Opcional) El tipo de datos de la clave hash. El valor del tipo de clave hash debe ser `STRING` o `NUMBER`. El valor predeterminado es `STRING`. También puede especificar una expresión.

hashKeyValue

El valor de la clave hash. La `hashKeyValue` utiliza plantillas de sustitución. Estas plantillas proporcionan datos en tiempo de ejecución. También puede especificar una expresión.

rangeKeyField

(Opcional) El nombre de la clave de rango (también denominada clave de clasificación). El valor `rangeKeyField` debe coincidir con la clave de clasificación de la tabla de DynamoDB. También puede especificar una expresión.

rangeKeyType

(Opcional) El tipo de datos de la clave de rango. El valor del tipo de clave hash debe ser `STRING` o `NUMBER`. El valor predeterminado es `STRING`. También puede especificar una expresión.

rangeKeyValue

(Opcional) El valor de la clave de rango. La `rangeKeyValue` utiliza plantillas de sustitución. Estas plantillas proporcionan datos en tiempo de ejecución. También puede especificar una expresión.

operación

(Opcional) El tipo de operación que se va a realizar. También puede especificar una expresión. El valor de la operación debe ser uno de los siguientes:

- `INSERT`: permite insertar datos como un elemento nuevo en la tabla de DynamoDB. Este es el valor predeterminado.
- `UPDATE`: permite actualizar un elemento existente de la tabla de DynamoDB con nuevos datos.
- `DELETE`: permite eliminar un elemento existente de la tabla de DynamoDB.

payloadField

(Opcional) El nombre de la columna de DynamoDB que recibe la carga de acción. El nombre predeterminado es `payload`. También puede especificar una expresión.

payload

(Opcional) La carga predeterminada contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo detector y el evento que desencadenó la acción. También puede personalizar la carga. Para obtener más información, consulte [Carga útil](#) en la referencia. [AWS IoT Events API](#)

Si el tipo de carga útil especificado es una cadena, `DynamoDBAction` envía datos que no sean JSON a la tabla de DynamoDB como datos binarios. La consola de DynamoDB mostrará los datos como texto codificado en Base64. El valor de `payloadField` es `payload-field_raw`. También puede especificar una expresión.

Note

Asegúrese de que la política asociada a su función de AWS IoT Events servicio conceda el permiso. `dynamodb:PutItem` Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [DynamoDBAction](#) en la AWS IoT Events API referencia.

Amazon DynamoDB(v2)

DynamoDBv2 action

La acción Amazon DynamoDB(v2) le permite escribir datos en una tabla de DynamoDB. Una columna independiente de la tabla de DynamoDB recibe un par atributo-valor de la carga de acción que especifique. Para ver una lista de las regiones admitidas, consulte [Puntos de conexión y cuotas de Amazon DynamoDB](#) en la Referencia general de Amazon Web Services.

Amazon DynamoDB es un servicio SQL sin base de datos totalmente gestionado que proporciona un rendimiento rápido y predecible con una escalabilidad perfecta. Para obtener más información, consulte [¿Qué es DynamoDB?](#) en la Guía para desarrolladores de Amazon DynamoDB.

More information (2)

Al enviar datos a varias columnas de una tabla de DynamoDB, debe especificar los siguientes parámetros.

tableName

El nombre de la tabla de DynamoDB que recibe los datos. También puede especificar una expresión.

payload

(Opcional) La carga predeterminada contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo detector y el evento que desencadenó la acción. También puede personalizar la carga. Para obtener más información, consulte [Carga útil](#) en la referencia.AWS IoT Events API

⚠ Important

El tipo de carga útil debe ser JSON. También puede especificar una expresión.

ℹ Note

Asegúrese de que la política asociada a su función de AWS IoT Events servicio conceda el `dynamodb:PutItem` permiso. Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [DynamoDBv2Action](#) en la AWS IoT Events API referencia.

Amazon Data Firehose

Firehose action

La acción Amazon Data Firehose te permite enviar datos a una cadena de entrega de Firehose. Para ver la lista de regiones compatibles, consulte los [puntos de enlace y las cuotas de Amazon Data Firehose](#) en el. Referencia general de Amazon Web Services

Amazon Data Firehose es un servicio totalmente gestionado para entregar datos de streaming en tiempo real a destinos como Amazon Simple Storage Service (Amazon Simple Storage Service), Amazon Redshift, OpenSearch Amazon OpenSearch Service (Service) y Splunk. Para obtener más información, consulte [¿Qué es Amazon Data Firehose?](#) en la Guía para desarrolladores de Amazon Data Firehose.

More information (3)

Al enviar datos a un flujo de entrega de Firehose, debe especificar los siguientes parámetros.

deliveryStreamName

El nombre del flujo de entrega de Firehose que recibe los datos.

separator

(Opcional) Puede usar un separador de caracteres para separar los datos continuos enviados al flujo de entrega de Firehose. El valor del separador debe ser '\n' (nueva línea), '\t' (tabulador), '\r\n' (nueva línea de Windows) o ',' (coma).

payload

(Opcional) La carga predeterminada contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo detector y el evento que desencadenó la acción. También puede personalizar la carga. Para obtener más información, consulte [Carga útil](#) en la AWS IoT Events API referencia.

Note

Asegúrese de que la política adjunta a su función de AWS IoT Events servicio conceda el `firehose:PutRecord` permiso. Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [FirehoseAction](#) la AWS IoT Events API Referencia.

AWS Lambda

Lambda action

La AWS Lambda acción permite llamar a una función Lambda. Para ver una lista completa de las regiones admitidas, consulte [Puntos de conexión y cuotas de AWS Lambda](#) en la Referencia general de Amazon Web Services.

AWS Lambda es un servicio informático que permite ejecutar código sin aprovisionar ni administrar servidores. Para obtener más información, consulta [¿Qué es? AWS Lambda](#) en la Guía para AWS Lambda desarrolladores.

More information (2)

Al llamar a una función de Lambda, debe especificar los siguientes parámetros.

functionArn

La ARN de la función Lambda a llamar.

payload

(Opcional) La carga predeterminada contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo detector y el evento que desencadenó la acción. También puede personalizar la carga. Para obtener más información, consulte [Carga útil](#) en la AWS IoT Events API referencia.

Note

Asegúrese de que la política adjunta a su función de AWS IoT Events servicio conceda el `lambda:InvokeFunction` permiso. Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [LambdaAction](#) la AWS IoT Events API Referencia.

Amazon Simple Notification Service

SNS action

La acción de publicación de un SNS tema de Amazon te permite publicar un SNS mensaje de Amazon. Para obtener la lista de las regiones admitidas, consulte [Puntos de conexión y cuotas de Amazon Simple Notification Service](#) en la Referencia general de Amazon Web Services.

Amazon Simple Notification Service (Amazon Simple Notification Service) es un servicio web que coordina y gestiona la entrega o el envío de mensajes a los puntos de conexión o clientes suscritos. Para obtener más información, consulta [¿Qué es AmazonSNS?](#) en la guía para desarrolladores de Amazon Simple Notification Service.

Note

La acción de publicación de SNS temas de Amazon no admite los temas de Amazon SNS FIFO (primero en entrar, primero en salir). Como el motor de reglas es un servicio totalmente distribuido, es posible que los mensajes no se muestren en un orden específico cuando se inicie la SNS acción de Amazon.

More information (2)

Cuando publiques un SNS mensaje de Amazon, debes especificar los siguientes parámetros.

targetArn

El ARN del SNS destinatario de Amazon que recibe el mensaje.

payload

(Opcional) La carga predeterminada contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo detector y el evento que desencadenó la acción.

También puede personalizar la carga. Para obtener más información, consulta [Carga útil](#) en la AWS IoT Events APIreferencia.

Note

Asegúrese de que la política adjunta a su función de AWS IoT Events servicio conceda el `sns:Publish` permiso. Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [SNSTopicPublishAction](#) la AWS IoT Events APIReferencia.

Amazon Simple Queue Service

SQS action

La SQS acción Amazon te permite enviar datos a una SQS cola de Amazon. Para obtener una lista de las regiones admitidas, consulte [Puntos de conexión y cuotas de Amazon Simple Queue Service](#) en la Referencia general de Amazon Web Services.

Amazon Simple Queue Service (AmazonSQS) ofrece una cola alojada segura, duradera y disponible que le permite integrar y desacoplar sistemas y componentes de software distribuidos. Para obtener más información, consulte [Qué es Amazon Simple Queue Service](#) en la Guía para desarrolladores de Amazon Simple Queue Service.

Note

La SQS acción de Amazon no admite temas de >Amazon SQS FIFO (primero en entrar, primero en salir). Como el motor de reglas es un servicio totalmente distribuido, es posible que los mensajes no se muestren en un orden específico cuando se inicie la SQS acción de Amazon.

More information (3)

Cuando envíes datos a una SQS cola de Amazon, debes especificar los siguientes parámetros.

queueUrl

El URL de la SQS cola de Amazon que recibe los datos.

useBase64

(Opcional) AWS IoT Events codifica los datos en texto en Base64, si lo especificas. TRUE El valor predeterminado es FALSE.

payload


(Opcional) La carga predeterminada contiene todos los pares atributo-valor que tienen la información sobre la instancia del modelo detector y el evento que desencadenó la acción. También puede personalizar la carga. Para obtener más información, consulte [Carga útil en la referencia](#).AWS IoT Events API

Note

Asegúrese de que la política adjunta a su función de AWS IoT Events servicio conceda el `sqs:SendMessage` permiso. Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

Para obtener más información, consulte [SNSTopicPublishAction](#)la AWS IoT Events APIReferencia.

También puedes usar Amazon SNS y el motor de AWS IoT Core reglas para activar una AWS Lambda función. Esto permite realizar acciones mediante otros servicios, como Amazon Connect, o incluso una aplicación de planificación de recursos empresariales (ERP) de la empresa.

 Note

Para recopilar y procesar grandes flujos de registros de datos en tiempo real, puede utilizar otros AWS servicios, como [Amazon Kinesis](#). Desde allí, puede completar un análisis inicial y, a continuación, enviar los resultados AWS IoT Events como entrada a un detector.

Expresiones para filtrar, transformar y procesar datos de eventos

Las expresiones se utilizan para evaluar los datos entrantes, realizar cálculos y determinar las condiciones en las que deben producirse acciones o transiciones de estado específicas. AWS IoT Events proporciona varias formas de especificar valores al crear y actualizar modelos de detectores. Puede usar expresiones para especificar valores literales o AWS IoT Events puede evaluar las expresiones antes de especificar valores determinados.

Temas

- [Sintaxis para filtrar los datos del dispositivo y definir acciones](#)
- [Ejemplos de expresiones y uso para AWS IoT Events](#)

Sintaxis para filtrar los datos del dispositivo y definir acciones

Puede utilizar literales, operadores, funciones, referencias y plantillas de sustitución en las expresiones de AWS IoT Events .

Literales

- Entero
- Decimal
- Cadena
- Booleano

Operadores

Unario

- Not (Booleano): !
- Not (bit a bit): ~
- Menos (aritmético): -

Cadena

- Concatenación: +

Ambos operandos deben ser cadenas. Los literales de cadena se deben incluir entre comillas simples (').

Por ejemplo: 'my' + 'string' -> 'mystring'

Aritmético

- Adición: +

Ambos operandos deben ser numéricos.

- Sustracción: -
- División: /

El resultado de la división es un valor entero redondeado, a menos que al menos uno de los operandos (divisor o dividendo) sea un valor decimal.

- Multiplicación: *

Bit a bit (entero)

- O BIEN: |

Por ejemplo: 13 | 5 -> 13

- AND: &

Por ejemplo: 13 & 5 -> 5

- XOR: ^

Por ejemplo: 13 ^ 5 -> 8


- NOT: ~

Por ejemplo: ~13 -> -14

Booleano

- Menor que: <
- Menor o igual que: <=
- Igual a: ==
- No igual a: !=
- Mayor o igual que: >=
- Mayor que: >

- AND: &&
- O BIEN: ||

 Note

Cuando una subexpresión de || contiene datos indefinidos, esa subexpresión se trata como false.

Paréntesis

Puede usar paréntesis para agrupar términos dentro de una expresión.

Funciones para usar en expresiones

Funciones integradas

timeout("*timer-name*")

Da como resultado true si ha transcurrido el tiempo especificado. Reemplazar *nombre del temporizador* con el nombre del temporizador que haya definido, entre comillas. En una acción de evento, puede definir un temporizador y luego iniciarlo, restablecerlo o borrar uno que haya definido con anterioridad. Consulte el campo `detectorModelDefinition.states.onInput|onEnter|onExit.events.actions.setTimer.timerName`.

Un temporizador creado en un estado puede ser referenciado en un estado diferente. Debe visitar el estado en el que creó el temporizador antes de entrar en el estado en el que se hace referencia al temporizador.

Por ejemplo, un modelo de detector tiene dos estados, `TemperatureChecked` y `RecordUpdated`. Ha creado un temporizador en el `TemperatureChecked` estado. Primero debe visitar el `TemperatureChecked` estado antes de poder usar el temporizador en el `RecordUpdated` estado.

Para garantizar la precisión, el tiempo mínimo que debe fijarse un temporizador es de 60 segundos.

Note

`timeout()` devuelve `true` solo la primera vez que se comprueba tras el vencimiento real del temporizador y devuelve `false` a partir de entonces.

`convert(type, expression)`

Da como resultado el valor de la expresión convertida al tipo especificado. La *type* el valor debe ser `String`, `Boolean`, o `Decimal`. Utilice una de estas palabras clave o una expresión que dé como resultado una cadena que contenga la palabra clave. Solo las siguientes conversiones tienen éxito y devuelven un valor válido:

- Booleano -> cadena

Devuelve la cadena `"true"` o `"false"`.

- Decimal -> cadena
- Cadena -> Booleano
- Cadena -> decimal

La cadena especificada debe ser una representación válida de un número decimal o `convert()` falla.

Si `convert()` no devuelve un valor válido, la expresión de la que forma parte tampoco es válida. Este resultado equivale a `false` y no activará las `actions` o la transición al `nextState` especificado como parte del evento en el que se produce la expresión.

`isNull(expression)`

Da como resultado `true` si la expresión devuelve un valor nulo. Por ejemplo, si la entrada `MyInput` recibe el mensaje `{ "a": null }`, entonces lo siguiente da como resultado `true`, pero `isUndefined($input.MyInput.a)` da como resultado `false`.

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

`isUndefined(expression)`

Da como resultado `true` si la expresión es indefinida. Por ejemplo, si la entrada `MyInput` recibe el mensaje `{ "a": null }`, entonces lo siguiente da como resultado `false`, pero `isNull($input.MyInput.a)` da como resultado `true`.

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

triggerType("type")

La *type* el valor puede ser "Message" o "Timer". Se evalúa como true si la condición del evento en la que aparece se está evaluando porque un temporizador se ha vencido, como en el siguiente ejemplo.

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

O se ha recibido un mensaje de entrada.

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

currentInput("input")

Da como resultado true si la condición del evento en la que aparece se está evaluando porque se ha recibido el mensaje de entrada especificado. Por ejemplo, si la entrada Command recibe el mensaje { "value": "Abort" }, entonces da como resultado true.

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

Utilice esta función para verificar que la condición se está evaluando porque se ha recibido un mensaje de entrada concreto y no se ha vencido un temporizador, como en la siguiente expresión.

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

Funciones de coincidencia de cadenas

startsWith(*expression1*, *expression2*)

Da como resultado true si la primera expresión de cadena comienza con la segunda expresión de cadena. Por ejemplo, si la entrada MyInput recibe el mensaje { "status": "offline" }, entonces lo siguiente da como resultado true.

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

Ambas expresiones deben dar como resultado un valor de cadena. Si alguna de las expresiones no da como resultado un valor de cadena, entonces el resultado de la función es indefinido. No se realiza ninguna conversión.

endsWith(*expression1*, *expression2*)

Da como resultado `true` si la primera expresión de cadena termina con la segunda expresión de cadena. Por ejemplo, si la entrada `MyInput` recibe el mensaje `{ "status": "offline" }`, entonces lo siguiente da como resultado `true`.

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

Ambas expresiones deben dar como resultado un valor de cadena. Si alguna de las expresiones no da como resultado un valor de cadena, entonces el resultado de la función es indefinido. No se realiza ninguna conversión.

contains(*expression1*, *expression2*)

Da como resultado `true` si la primera expresión de cadena contiene la segunda expresión de cadena. Por ejemplo, si la entrada `MyInput` recibe el mensaje `{ "status": "offline" }`, entonces lo siguiente da como resultado `true`.

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

Ambas expresiones deben dar como resultado un valor de cadena. Si alguna de las expresiones no da como resultado un valor de cadena, entonces el resultado de la función es indefinido. No se realiza ninguna conversión.

Funciones de manipulación de enteros bit a bit

bitor(*expression1*, *expression2*)

Evalúa el OR bit a bit de las expresiones de enteros (la operación OR binaria se realiza en los bits correspondientes de los enteros). Por ejemplo, si la entrada `MyInput` recibe el mensaje `{ "value1": 13, "value2": 5 }`, entonces lo siguiente da como resultado 13.

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

Ambas expresiones deben dar como resultado un valor entero. Si cualquiera de las expresiones no da como resultado un valor entero, el resultado de la función es indefinido. No se realiza ninguna conversión.

bitand(*expression1*, *expression2*)

Evalúa las expresiones enteras a nivel AND de bits (la AND operación binaria se realiza en los bits correspondientes de los números enteros). Por ejemplo, si la entrada MyInput recibe el mensaje { "value1": 13, "value2": 5 }, entonces lo siguiente da como resultado 5.

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

Ambas expresiones deben dar como resultado un valor entero. Si cualquiera de las expresiones no da como resultado un valor entero, el resultado de la función es indefinido. No se realiza ninguna conversión.

bitxor(*expression1*, *expression2*)

Evalúa las expresiones enteras bit XOR a bit (la XOR operación binaria se realiza en los bits correspondientes de los enteros). Por ejemplo, si la entrada MyInput recibe el mensaje { "value1": 13, "value2": 5 }, entonces lo siguiente da como resultado 8.

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

Ambas expresiones deben dar como resultado un valor entero. Si cualquiera de las expresiones no da como resultado un valor entero, el resultado de la función es indefinido. No se realiza ninguna conversión.

bitnot(*expression*)

Evalúa la expresión entera a NOT nivel de bits (la NOT operación binaria se realiza en los bits del entero). Por ejemplo, si la entrada MyInput recibe el mensaje { "value": 13 }, entonces lo siguiente da como resultado -14.

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

Ambas expresiones deben dar como resultado un valor entero. Si cualquiera de las expresiones no da como resultado un valor entero, el resultado de la función es indefinido. No se realiza ninguna conversión.

Referencia para las entradas y variables de las expresiones

Entradas

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

`input-name` es una entrada que se crea mediante la [CreateInput](#) acción.

Por ejemplo, si tiene una entrada llamada `TemperatureInput` para la que ha definido entradas de `inputDefinition.attributes.jsonPath`, los valores podrían aparecer en los siguientes campos disponibles.

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

Para hacer referencia al valor del campo `temperature`, utilice el siguiente comando.

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

Para los campos cuyos valores son matrices, puede hacer referencia a los miembros de la matriz mediante `[n]`. Por ejemplo, dados los siguientes valores:

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

Se puede hacer referencia al valor `78.8` con el siguiente comando.

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

Variables

`$variable.variable-name`

variable-name Es una variable que se definió mediante la [CreateDetectorModel](#) acción.

Por ejemplo, si tiene una variable llamada TechnicianID que ha definido mediante `detectorDefinition.states.onInputEvents.actions.setVariable.variableName`, puede hacer referencia al valor (cadena) dado más recientemente a la variable con el siguiente comando.

```
$variable.TechnicianID
```

Solo puede establecer los valores de las variables mediante la acción `setVariable`. No puede asignar valores a variables en una expresión. Una variable no se puede destruir. Por ejemplo, no puede asignarle el valor `null`.

Note

En las referencias que utilicen identificadores que no sigan el patrón (expresión regular) `[a-zA-Z][a-zA-Z0-9_]*`, debe encerrar dichos identificadores entre tildes graves (```). Por ejemplo, una referencia a una entrada llamada MyInput con un campo llamado `_value` debe especificar este campo como `$input.MyInput.`_value``.

Cuando utilice referencias en expresiones, compruebe lo siguiente:

- Cuando utilice una referencia como operando con uno o más operadores, asegúrese de que todos los tipos de datos a los que hace referencia sean compatibles.

Por ejemplo, en la siguiente expresión, el entero 2 es un operando de los operadores `==` y `&&`. Para asegurarse de que los operandos sean compatibles, `$variable.testVariable + 1` y `$variable.testVariable` deben hacer referencia a un número entero o decimal.

Además, el entero 1 es un operando del operador `+`. Por lo tanto, `$variable.testVariable` debe hacer referencia a un número entero o decimal.

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

- Cuando utilice una referencia como argumento pasado a una función, asegúrese de que la función admita los tipos de datos a los que hace referencia.

Por ejemplo, la siguiente función `timeout("time-name")` requiere una cadena con comillas dobles como argumento. Si utiliza una referencia para la `timer-name` valor, debe hacer referencia a una cadena con comillas dobles.

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

Note

Para la `convert(type, expression)` función, si utiliza una referencia para el `type` valor, el resultado evaluado de la referencia debe ser `StringDecimal`, o `Boolean`.

AWS IoT Events las expresiones admiten tipos de datos enteros, decimales, de cadena y booleanos. En la tabla siguiente se ofrece una lista de pares de tipos incompatibles.

Pares de tipos incompatibles

Entero, cadena

Entero, booleano

Decimal, cadena

Decimal, booleano

Cadena, booleano

Plantillas de sustitución para expresiones

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

`${}` identifica la cadena como una cadena interpolada. `expression` Puede ser cualquier AWS IoT Events expresión. Esto incluye operadores, funciones y referencias.

Por ejemplo, ha utilizado la [SetVariableAction](#) acción para definir una variable. El `variableName` es `SensorID` y el `value` es `10`. Puede crear las siguientes plantillas de sustitución.

Plantilla de sustitución	Cadena de resultados
<code>'\${Sensor ' + \$variable.SensorID}'</code>	"Sensor 10"
<code>'Sensor ' + '\${\$variable.SensorID + 1}'</code>	"Sensor 11"
<code>'Sensor 10: \${\$variable.SensorID == 10}'</code>	"Sensor 10: true"
<code>'{"sensor\":"\${\$variable.SensorID + 1}\}'</code>	"{"sensor\":"11\"}"
<code>'{"sensor\":"\${\$variable.SensorID + 1}}'</code>	"{"sensor\":"11}"

Ejemplos de expresiones y uso para AWS IoT Events

Puede especificar valores en un modelo de detector de las siguientes maneras:

- Introduzca las expresiones compatibles en la AWS IoT Events consola.
- Pase las expresiones a los parámetros AWS IoT Events APIs as.

Las expresiones admiten literales, operadores, funciones, referencias y plantillas de sustitución.

Important

Sus expresiones deben hacer referencia a un valor entero, decimal, de cadena o booleano.

Escribir AWS IoT Events expresiones

Consulta los siguientes ejemplos para ayudarte a escribir tus AWS IoT Events expresiones:

Literal

En los valores literales, las expresiones deben contener comillas simples. Un valor booleano debe ser `true` o `false`.

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

Referencia

En las referencias, debe especificar variables o valores de entrada.

- La siguiente entrada hace referencia a un número decimal, `10.01`.

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

- La siguiente variable hace referencia a una cadena, `Greenhouse Temperature Table`.

```
$variable.TableName
```

Plantilla de sustitución

En una plantilla de sustitución, debe usar `${}` y la plantilla debe estar entre comillas simples. Una plantilla de sustitución también puede contener una combinación de literales, operadores, funciones, referencias y plantillas de sustitución.

- El resultado evaluado de la siguiente expresión es una cadena, `50.018 in Fahrenheit`.

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

- El resultado evaluado de la siguiente expresión es una cadena, `{"sensor_id\":"Sensor_1\","temperature\":"50.018\}"`.

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

Concatenación de cadenas

En una concatenación de cadenas, debe usar `+`. Una concatenación de cadenas también puede contener una combinación de literales, operadores, funciones, referencias y plantillas de sustitución.

- El resultado evaluado de la siguiente expresión es una cadena, Greenhouse Temperature Table 2000-01-01.

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

AWS IoT Events ejemplos de modelos de detectores

Esta página proporciona una lista de ejemplos de casos de uso que muestran cómo configurar varias AWS IoT Events funciones. Los ejemplos van desde detecciones básicas, como los umbrales de temperatura, hasta escenarios más avanzados de detección de anomalías y aprendizaje automático. Cada ejemplo incluye procedimientos y fragmentos de código para ayudarle a configurar las AWS IoT Events detecciones, las acciones y las integraciones. Estos ejemplos muestran la flexibilidad del AWS IoT Events servicio y cómo se puede personalizar para diversas aplicaciones y casos de uso de IoT. Consulte esta página cuando explore AWS IoT Events las capacidades o si necesita orientación para implementar un flujo de trabajo específico de detección o automatización.

Temas

- [Ejemplo: uso del control de HVAC temperatura](#)
- [Ejemplo: una grúa que detecta condiciones](#)
- [Ejemplo: detección de eventos con sensores y aplicaciones](#)
- [Ejemplo: dispositivo HeartBeat para monitorizar las conexiones de los dispositivos](#)
- [Ejemplo: una ISA alarma](#)
- [Ejemplo: construya una alarma sencilla](#)

Ejemplo: uso del control de HVAC temperatura

Antecedentes

Este ejemplo implementa un modelo de control de temperatura (un termostato) con estas características:

- Un modelo de detector que usted define que puede supervisar y controlar múltiples áreas. (Se creará una instancia de detector para cada área).
- Cada instancia de detector recibe datos de temperatura de múltiples sensores dispuestos en cada área de control.
- Puede cambiar la temperatura deseada (el punto de consigna) para cada área en cualquier momento.
- Puede definir los parámetros operativos para cada área y cambiar estos parámetros en cualquier momento.

- Puede añadir sensores a un área o eliminarlos de ella en cualquier momento.
- Puede activar un tiempo mínimo de funcionamiento de las unidades de calefacción y refrigeración para protegerlas de posibles daños.
- Los detectores rechazarán y reportarán las lecturas anómalas de los sensores.
- Puede definir puntos de consigna de temperatura de emergencia. Si cualquier sensor reporta una temperatura superior o inferior a los puntos de consigna que haya definido, las unidades de calefacción o refrigeración se activarán de inmediato y el detector reportará ese pico de temperatura.

Este ejemplo demuestra las siguientes capacidades funcionales:

- Creación de modelos de detectores de eventos.
- Creación de entradas.
- Ingesta de entradas en un modelo de detector.
- Evaluación de las condiciones de activación.
- Consulta de las variables de estado en las condiciones y ajuste de los valores de las variables en función de las condiciones.
- Consulta de los temporizadores en las condiciones y ajuste de los temporizadores en función de las condiciones.
- Realiza acciones para enviar MQTT mensajes SNS y a Amazon.

Introduce definiciones para un HVAC sistema

Se utiliza una `seedTemperatureInput` para crear una instancia de detector para un área y definir sus parámetros operativos.

CLI comando utilizado:

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

Archivo: `seedInput.json`

```
{
  "inputName": "seedTemperatureInput",
  "inputDescription": "Temperature seed values.",
```

```



```

Respuesta:

```

{
  "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."
  }
}

```

Cada sensor de cada área debe enviar una `temperatureInput`, según sea necesario.

CLI comando utilizado:

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

Archivo: `temperatureInput.json`

```

{
  "inputName": "temperatureInput",
  "inputDescription": "Temperature sensor unit data.",
  "inputDefinition": {
    "attributes": [

```

```

    { "jsonPath": "sensorId" },
    { "jsonPath": "areaId" },
    { "jsonPath": "sensorData.temperature" }
  ]
}

```

Respuesta:

```

{
  "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."
  }
}

```

Definición del modelo de detector para un HVAC sistema

El `areaDetectorModel` define cómo funciona cada instancia de detector. Cada instancia de `state machine` recoge las lecturas de los sensores de temperatura, luego cambia el estado y envía mensajes de control en función de estas lecturas.

CLI comando utilizado:

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

Archivo: `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:cool0n"
                }
            }
        }
    ]
}

```

```
    },
    {
      "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:cool0ff"
            }
        },
        {
            "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"
}

```

Respuesta:

```

{
    "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"
  }
}

```

BatchPutMessageejemplos de un HVAC sistema

En este ejemplo, BatchPutMessage se utiliza para crear una instancia de detector para un área y definir los parámetros de funcionamiento iniciales.

CLI comando utilizado:

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

Archivo: 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}"
    }
  ]
}

```

Respuesta:

```

{
  "BatchPutMessageErrorEntries": []
}

```

En este ejemplo, BatchPutMessage se utiliza para reportar las lecturas de los sensores de temperatura, de un único sensor en un área.

CLI comando utilizado:

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

Archivo: temperatureExample.json

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

Respuesta:

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

En este ejemplo, BatchPutMessage se utiliza para cambiar la temperatura deseada de un área.

CLI comando utilizado:

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

Archivo: seedSetDesiredTemp.json

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

```
}
```

Respuesta:

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

Ejemplos de SNS mensajes de Amazon generados por la instancia del Area51 detector:

```
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"
}

```

En este ejemplo, utilizamos el DescribeDetector API para obtener información sobre el estado actual de una instancia de detector.

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

Respuesta:

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

```
        "name": "rangeHigh",
        "value": "30.0"
    },
    {
        "name": "enteringNewState",
        "value": "false"
    },
    {
        "name": "averageTemperature",
        "value": "19.572"
    },
    {
        "name": "allowedError",
        "value": "0.7"
    },
    {
        "name": "anomalousHigh",
        "value": "60.0"
    },
    {
        "name": "reportedTemperature",
        "value": "15.72"
    },
    {
        "name": "goodToGo",
        "value": "false"
    }
    ],
    "stateName": "idle",
    "timers": [
        {
            "timestamp": 1557520454.0,
            "name": "idleTimer"
        }
    ]
},
"keyValue": "Area51",
"detectorModelName": "areaDetectorModel",
"detectorModelVersion": "1"
}
}
```

BatchUpdateDetector ejemplo de un HVAC sistema

En este ejemplo, BatchUpdateDetector se utiliza para cambiar los parámetros operativos de una instancia de detector en funcionamiento.

CLI comando utilizado:

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

Archivo: 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": [
]
}
]
}
```

Respuesta:

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

```
}
```

AWS IoT Core motor de reglas

Las siguientes reglas vuelven a publicar AWS IoT Events MQTT los mensajes como mensajes de solicitud de actualización paralelos. Suponemos que AWS IoT Core las cosas están definidas para una unidad de calefacción y una unidad de refrigeración para cada área controlada por el modelo de detector.

En este ejemplo, hemos definido cosas denominadas `Area51HeatingUnit` y `Area51CoolingUnit`.

CLI comando utilizado:

```
aws iot create-topic-rule --cli-input-json file://ADMShadowCool0ffRule.json
```

Archivo: `ADMShadowCool0ffRule.json`

```
{
  "ruleName": "ADMShadowCool0ff",
  "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": [
      {
        "republsh": {
          "topic": "$$aws/things/${payload.detector.keyValue}CoolingUnit/shadow/update",
          "roleArn": "arn:aws:iam::123456789012:role/service-role/ADMShadowRole"
        }
      }
    ]
  }
}
```

Respuesta: [vacío]

CLI comando utilizado:


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

Archivo: 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"
        }
      }
    ]
  }
}
```

Respuesta: [vacío]

CLI comando utilizado:

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

Archivo: 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"
      }
    }
  ]
}

```

Respuesta: [vacío]

CLI comando utilizado:

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

Archivo: 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"
        }
      }
    ]
  }
}

```

Respuesta: [vacío]

Ejemplo: una grúa que detecta condiciones

Antecedentes

Un operador de muchas grúas desea detectar cuándo las máquinas necesitan mantenimiento o sustitución y activar las notificaciones apropiadas. Cada grúa tiene un motor. Cada motor emite mensajes (entradas) con información sobre presión y temperatura. El operador quiere dos niveles de detectores de eventos:

- Un detector de eventos a nivel de grúa
- Un detector de eventos a nivel de motor

Mediante los mensajes procedentes de los motores (que contienen metadatos que incluyen `craneId` y `motorId`), el operador puede ejecutar ambos niveles de detectores de eventos utilizando el enrutamiento apropiado. Cuando se cumplan las condiciones del evento, las notificaciones deben enviarse a los SNS temas de Amazon correspondientes. El operador puede configurar los modelos de detectores para que no se emitan notificaciones duplicadas.

Este ejemplo demuestra las siguientes capacidades funcionales:

- Cree, lea, actualice y elimine (CRUD) de entradas.
- Cree, lea, actualice, elimine (CRUD) modelos de detectores de eventos y diferentes versiones de detectores de eventos.
- Enrutamiento de una entrada a múltiples detectores de eventos.
- Ingesta de entradas en un modelo de detector.
- Evaluación de las condiciones de activación y de eventos de ciclo de vida.
- Capacidad para consultar variables de estado en las condiciones y establecer sus valores en función de las condiciones.
- Orquestación en tiempo de ejecución con definición, estado, evaluador de desencadenantes y ejecutor de acciones.
- Ejecución de acciones `ActionsExecutor` con un SNS objetivo.

Envía comandos en respuesta a las condiciones detectadas

Esta página proporciona un ejemplo del uso de AWS IoT Events comandos para configurar entradas, crear modelos de detectores y enviar datos de sensores simulados. Los ejemplos muestran cómo aprovechar AWS IoT Events para monitorear equipos industriales como motores y grúas.

```
#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/2000008/2000009/g" messages/*

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

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

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

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

Modelo de detector para el monitoreo de grúas

Supervise sus flotas de equipos o dispositivos para detectar fallas o cambios en la operación y active acciones cuando ocurran dichos eventos. Usted define modelos de detectores en los JSON que se especifican estados, reglas y acciones. Esto le permite monitorear entradas como la temperatura y la presión, rastrear las infracciones de los umbrales y enviar alertas. Los ejemplos muestran modelos de detectores para una grúa y un motor, que detectan problemas de sobrecalentamiento y Amazon notifica SNS cuando se supera un umbral. Puede actualizar los modelos para refinar el comportamiento sin interrumpir la supervisión.

Archivo: craneDetectorModel.json

```
{
  "detectorModelName": "craneDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Running",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
```

```

        "actions": [
            {
                "setVariable": {
                    "variableName": "craneThresholdBreach",
                    "value": "0"
                }
            }
        ]
    },
    ],
    "onInput": {
        "events": [
            {
                "eventName": "Overheated",
                "condition": "$input.TemperatureInput.temperature > 35",
                "actions": [
                    {
                        "setVariable": {
                            "variableName": "craneThresholdBreach",
                            "value": "$variable.craneThresholdBreach + 1"
                        }
                    }
                ]
            },
            {
                "eventName": "Crane Threshold Breached",
                "condition": "$variable.craneThresholdBreach > 5",
                "actions": [
                    {
                        "sns": {
                            "targetArn": "arn:aws:sns:us-
east-1:123456789012:CraneSNSTopic"
                        }
                    }
                ]
            },
            {
                "eventName": "Underheated",
                "condition": "$input.TemperatureInput.temperature < 25",
                "actions": [
                    {
                        "setVariable": {
                            "variableName": "craneThresholdBreach",

```

```

        "value": "0"
      }
    ]
  }
},
"initialStateName": "Running"
},
"key": "craneid",
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

Para actualizar un modelo de detector existente. Archivo: `updateCraneDetectorModel.json`

```

{
  "detectorModelName": "craneDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Running",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "craneThresholdBreach",
                    "value": "0"
                  }
                },
                {
                  "setVariable": {
                    "variableName": "alarmRaised",
                    "value": "'false'"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}

```

```

    ]
  },
  "onInput": {
    "events": [
      {
        "eventName": "Overheated",
        "condition": "$input.TemperatureInput.temperature > 30",
        "actions": [
          {
            "setVariable": {
              "variableName": "craneThresholdBreach",
              "value": "$variable.craneThresholdBreach + 1"
            }
          }
        ]
      },
      {
        "eventName": "Crane Threshold Breached",
        "condition": "$variable.craneThresholdBreach > 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": "craneThresholdBreach",
              "value": "0"
            }
          }
        ]
      }
    ]
  }
}

```



```

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

```

Archivo: motorDetectorModel.json

```

{
  "detectorModelName": "motorDetectorModel",
  "detectorModelDefinition": {
    "states": [
      {
        "stateName": "Running",
        "onEnter": {
          "events": [
            {
              "eventName": "init",
              "condition": "true",
              "actions": [
                {
                  "setVariable": {
                    "variableName": "motorThresholdBreach",
                    "value": "0"
                  }
                }
              ]
            }
          ]
        },
        "onInput": {
          "events": [
            {
              "eventName": "Overheated And Overpressurized",
              "condition": "$input.PressureInput.pressure > 70 &&
$input.TemperatureInput.temperature > 30",
              "actions": [

```

```

        {
            "setVariable": {
                "variableName": "motorThresholdBreach",
                "value": "$variable.motorThresholdBreach + 1"
            }
        }
    ],
},
{
    "eventName": "Motor Threshold Breach",
    "condition": "$variable.motorThresholdBreach > 5",
    "actions": [
        {
            "sns": {
                "targetArn": "arn:aws:sns:us-
east-1:123456789012:MotorSNSTopic"
            }
        }
    ]
}
]
}
},
],
    "initialStateName": "Running"
},
    "key": "motorId",
    "roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}

```

Para actualizar un modelo de detector existente. Archivo: `updateMotorDetectorModel.json`

```

{
    "detectorModelName": "motorDetectorModel",
    "detectorModelDefinition": {
        "states": [
            {
                "stateName": "Running",
                "onEnter": {
                    "events": [
                        {
                            "eventName": "init",
                            "condition": "true",

```

```

        "actions": [
            {
                "setVariable": {
                    "variableName": "motorThresholdBreach",
                    "value": "0"
                }
            }
        ]
    },
    "onInput": {
        "events": [
            {
                "eventName": "Overheated And Overpressurized",
                "condition": "$input.PressureInput.pressure > 70 &&
$input.TemperatureInput.temperature > 30",
                "actions": [
                    {
                        "setVariable": {
                            "variableName": "motorThresholdBreach",
                            "value": "$variable.motorThresholdBreach + 1"
                        }
                    }
                ]
            },
            {
                "eventName": "Motor Threshold Breached",
                "condition": "$variable.motorThresholdBreach > 5",
                "actions": [
                    {
                        "sns": {
                            "targetArn": "arn:aws:sns:us-
east-1:123456789012:MotorSNSTopic"
                        }
                    }
                ]
            }
        ]
    }
},
"initialStateName": "Running"
},

```

```
"roleArn": "arn:aws:iam::123456789012:role/columboSNSRole"
}
```

Entradas para el monitoreo de grúas

Archivo: pressureInput.json

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

Archivo: temperatureInput.json

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

Envíe mensajes de alarma y operativos

Archivo: highPressureMessage.json

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

```
    }  
  ]  
}
```

Archivo: highTemperatureMessage.json

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

Archivo: lowPressureMessage.json

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

Archivo: lowTemperatureMessage.json

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

Ejemplo: detección de eventos con sensores y aplicaciones

Este modelo de detector es una de las plantillas disponibles en la AWS IoT Events consola. Se incluye aquí para su comodidad.

```
{
  "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"
}
}

```

Ejemplo: dispositivo HeartBeat para monitorizar las conexiones de los dispositivos

Este modelo de detector es una de las plantillas disponibles en la AWS IoT Events consola. Se incluye aquí para su comodidad.

```

{
  "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"
}
}

```

Ejemplo: una ISA alarma

Este modelo de detector es una de las plantillas disponibles en la AWS IoT Events consola. Se incluye aquí para su comodidad.

```

{
  "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"
}

```

Ejemplo: construya una alarma sencilla

Este modelo de detector es una de las plantillas disponibles en la AWS IoT Events consola. Se incluye aquí para su comodidad.

```

{
  "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"
                }
            }
        }
    ]
}

```

```
        ],
        "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"
}
```


Monitoreo con alarmas

AWS IoT Events las alarmas le ayudan a supervisar sus datos para detectar cambios. Los datos pueden ser métricas que mide para sus equipos y procesos. Puede crear alarmas que le envíen notificaciones al superarse un umbral. Las alarmas le ayudan a detectar problemas, agilizar el mantenimiento y optimizar el rendimiento de sus equipos y procesos.

Las alarmas son instancias de modelos de alarma. Un modelo de alarma especifica qué detectar, cuándo enviar notificaciones, a quién notificar, etc. También puede especificar una o más [acciones compatibles](#) que se producen cuando cambia el estado de la alarma. AWS IoT Events enruta [los atributos de entrada](#) derivados de sus datos a las alarmas correspondientes. Si los datos que monitorea están fuera del rango especificado, se invoca una alarma. También puede reconocer las alarmas o ponerlas en modo posponer (snooze).

Trabajando con AWS IoT SiteWise

Puede utilizar AWS IoT Events las alarmas para supervisar las propiedades de los activos en AWS IoT SiteWise. AWS IoT SiteWise envía los valores de las propiedades de los activos a AWS IoT Events las alarmas. AWS IoT Events envía el estado de alarma a AWS IoT SiteWise.

AWS IoT SiteWise también admite alarmas externas. Puede elegir alarmas externas si utiliza alarmas externas AWS IoT SiteWise y dispone de una solución que devuelva los datos del estado de la alarma. La alarma externa contiene una propiedad de medición que ingiere los datos del estado de alarma.

AWS IoT SiteWise no evalúa el estado de las alarmas externas. Además, no puede reconocer o posponer una alarma externa si su estado cambia.

Puede utilizar la función SiteWise Monitor para ver el estado de las alarmas externas en los portales SiteWise Monitor.

Para obtener más información, consulte [Supervisión de datos con alarmas](#) en la Guía del AWS IoT SiteWise usuario y [Supervisión con alarmas](#) en la Guía de aplicaciones de SiteWise monitorización.

Flujo de reconocimiento

Al crear un modelo de alarma, puede elegir si desea habilitar el flujo de reconocimiento. Al habilitar el flujo de reconocimiento, su equipo recibirá notificaciones al cambiar el estado de una alarma. Su

equipo puede reconocer la alarma y dejar una nota. Por ejemplo, puede incluir la información de la alarma y las medidas que va a tomar para solucionar el problema. Si los datos que monitorea están fuera del rango especificado, se invoca una alarma.

Las alarmas tienen los siguientes estados:

DISABLED

Cuando la alarma está en el estado DISABLED, no está preparada para evaluar datos. Para activar la alarma, debe cambiarla al estado NORMAL.

NORMAL

Cuando la alarma está en el estado NORMAL, está lista para evaluar datos.

ACTIVE

Si la alarma está en el estado ACTIVE, se invoca la alarma. Los datos que está monitoreando están fuera del rango especificado.

ACKNOWLEDGED

Si la alarma está en el estado ACKNOWLEDGED, se invocó la alarma y usted la reconoció.

LATCHED

Se invocó la alarma, pero usted no la reconoció después de un cierto periodo. La alarma cambia automáticamente al estado NORMAL.

SNOOZE_DISABLED

Cuando la alarma está en el estado SNOOZE_DISABLED, esta se desactiva durante un periodo especificado. Una vez transcurrido el tiempo de aplazamiento, la alarma cambia automáticamente al estado NORMAL.

Creación de un modelo de alarma

Puede utilizar AWS IoT Events las alarmas para supervisar sus datos y recibir notificaciones cuando se supere un umbral. Las alarmas proporcionan parámetros que puede utilizar para crear o configurar un modelo de alarma. Puede usar la AWS IoT Events consola o AWS IoT Events API crear o configurar el modelo de alarma. Al configurar el modelo de alarma, los cambios tienen efecto a medida que llegan nuevos datos.

Requisitos

Al crear un modelo de alarma se imponen los siguientes requisitos.

- Puede crear un modelo de alarma para supervisar un atributo de entrada AWS IoT Events o una propiedad de un activo AWS IoT SiteWise.
 - Si decide monitorizar un atributo de entrada en AWS IoT Events, [Cree una entrada para los modelos](#) antes de crear el modelo de alarma.
 - Si decide supervisar una propiedad de activo, debe [crear un modelo de activo](#) AWS IoT SiteWise antes de crear el modelo de alarma.
- Debe tener una IAM función que permita a la alarma realizar acciones y acceder a AWS los recursos. Para obtener más información, consulte [Configuración de permisos para AWS IoT Events](#).
- Todos los AWS recursos que utilice este tutorial deben estar en la misma AWS región.

Creación de un modelo de alarma (consola)

A continuación, se muestra cómo crear un modelo de alarma para supervisar un AWS IoT Events atributo en la AWS IoT Events consola.

1. Inicie sesión en la [consola de AWS IoT Events](#).
2. En el panel de navegación, seleccione Modelos de alarmas.
3. En la página Modelos de alarmas, seleccione Crear modelo de alarma.
4. En la sección Detalles del modelo de alarma realice lo siguiente:
 - a. Introduzca un nombre único.
 - b. (Opcional) Introduzca una descripción.
5. En la sección Objetivo de la alarma realice lo siguiente:

Important

Para elegir Propiedad de activo de AWS IoT SiteWise , antes debe haber creado un modelo de activo en AWS IoT SiteWise.

- a. Seleccione Atributo de entrada de AWS IoT Events .

- b. Seleccione la entrada.
- c. Seleccione la clave del atributo de entrada. Este atributo de entrada se utiliza como clave para crear la alarma. AWS IoT Events enruta las entradas asociadas a esta clave a la alarma.

 Important

Si la carga útil del mensaje de entrada no contiene esta clave de atributo de entrada, o si la clave no está en la misma JSON ruta especificada en la clave, el mensaje no se podrá introducir correctamente. AWS IoT Events

6. En la sección Definiciones de umbrales, se definen el atributo de entrada, el valor umbral y el operador de comparación que se AWS IoT Events utilizan para cambiar el estado de la alarma.

- a. En Atributo de entrada, elija el atributo que desee monitorear.

Toda vez que este atributo de entrada reciba nuevos datos, estos se evaluarán para determinar el estado de la alarma.

- b. En Operador, elija el operador de comparación. El operador compara su atributo de entrada con el valor umbral de del mismo.

Puede elegir entre estas opciones:

- > mayor que
 - >= mayor o igual que
 - < menor que
 - <= menor o igual a
 - = igual a
 - != distinto de
- c. Para el valor umbral, introduzca un número o elija un atributo en AWS IoT Events las entradas. AWS IoT Events compara este valor con el valor del atributo de entrada que elija.
 - d. (Opcional) Para Gravedad, utilice un número que su equipo comprenda y refleje la gravedad de la alarma.
7. (Opcional) En la sección Configuración de notificaciones, configure los parámetros de notificación de la alarma.

Puede añadir hasta 10 notificaciones. En Notificación 1, haga lo siguiente:

- a. En Protocolo, elija entre las siguientes opciones:
 - Correo electrónico y texto: la alarma envía una SMS notificación y una notificación por correo electrónico.
 - Correo electrónico: La alarma envía una notificación por correo electrónico.
 - Texto: la alarma envía una SMS notificación.
- b. En Remitente, especifique la dirección de correo electrónico que puede enviar notificaciones sobre esta alarma.

Para añadir más direcciones de correo electrónico a la lista de remitentes, seleccione Añadir remitente.

- c. (Opcional) En Destinatario, elija el destinatario.

Para añadir más usuarios a su lista de destinatarios, elija Añadir nuevo usuario. Debe añadir nuevos usuarios a su tienda de IAM Identity Center antes de poder añadirlos a su modelo de alarma. Para obtener más información, consulte [Gestione el acceso al IAM Identity Center de los destinatarios de las alarmas](#).

- d. (Opcional) En Mensaje personalizado adicional, introduzca un mensaje que describa qué detecta la alarma y qué acciones deben realizar los destinatarios.
8. En la sección Instancia, puede activar o desactivar todas las instancias de alarma que se creen basándose en este modelo de alarma.
 9. En la sección Ajustes avanzados, haga lo siguiente:

- a. En Flujo de reconocimiento, puede habilitar o deshabilitar las notificaciones.
 - Si elige Habilitado, recibirá una notificación cuando cambie el estado de la alarma. Debe reconocer la notificación para que el estado de alarma vuelva a normal.
 - Si elige Deshabilitado, no se requiere ninguna acción. La alarma cambia automáticamente al estado normal cuando la medición vuelve al rango especificado.

Para obtener más información, consulte [Flujo de reconocimiento](#).

- b. En Permisos, elija una de las siguientes opciones:

- Puede crear un nuevo rol a partir de plantillas de AWS políticas y crear AWS IoT Events automáticamente un IAM rol para usted.
- Puede utilizar un IAM rol existente que permita a este modelo de alarma realizar acciones y acceder a otros AWS recursos.

Para obtener más información, consulte [Administración de identidad y acceso para AWS IoT Events](#).

- c. Para configurar las notificaciones adicionales, puede editar su AWS Lambda función para gestionar las notificaciones de alarma. Elige una de las siguientes opciones para tu AWS Lambda función:
 - Crear una AWS Lambda función nueva: AWS IoT Events crea una AWS Lambda función nueva para usted.
 - Utilizar una AWS Lambda función existente: utilice una AWS Lambda función existente eligiendo un nombre de AWS Lambda función.

Para obtener más información sobre las acciones posibles, consulte [Trabajando con otros AWS servicios](#).

- d. (Opcional) En la acción Establecer estado, puede añadir una o más AWS IoT Events acciones para llevarlas a cabo cuando cambie el estado de la alarma.
10. (Opcional) Puede añadir Etiquetas para gestionar sus alarmas. Para obtener más información, consulte [Etiquetado de los recursos de AWS IoT Events](#).
 11. Seleccione Crear.

Respuesta a las alarmas

Si activó [Flujo de reconocimiento](#), recibirá notificaciones al cambiar el estado de la alarma. Para responder a la alarma, puede reconocerla, deshabilitarla, habilitarla, restablecerla o posponerla.

Console

A continuación, se muestra cómo responder a una alarma en la consola de AWS IoT Events .

1. Inicie sesión en la [consola de AWS IoT Events](#).
2. En el panel de navegación, seleccione Modelos de alarmas.

3. Seleccione el modelo de alarma deseado.
4. En la sección Lista de alarmas, elija la alarma deseada.
5. En Acciones, puede elegir una de las siguientes opciones:
 - Reconocer: La alarma cambia al estado ACKNOWLEDGED.
 - Deshabilitar: La alarma pasa al estado DISABLED.
 - Habilitar: La alarma cambia al estado NORMAL.
 - Reiniciar: La alarma cambia al estado NORMAL.
 - Posponer, y luego haga lo siguiente:
 1. Elija la Duración de snooze o introduzca una Duración de Snooze personalizada.
 2. Seleccione Guardar.

La alarma cambia al estado SNOOZE_DISABLED

Para obtener más información sobre los estados de alarma, consulte [Flujo de reconocimiento](#).

API

Para responder a una o más alarmas, puede utilizar las siguientes AWS IoT Events API operaciones:

- [BatchAcknowledgeAlarm](#)
- [BatchDisableAlarm](#)
- [BatchEnableAlarm](#)
- [BatchResetAlarm](#)
- [BatchSnoozeAlarm](#)

Administración de las notificaciones de alarma

AWS IoT Events utiliza una función Lambda para gestionar las notificaciones de alarma. Puede utilizar la función Lambda proporcionada por AWS IoT Events o crear una nueva.

Temas

- [Creación de una función de Lambda](#)

- [Uso de la función de Lambda proporcionada por AWS IoT Events](#)
- [Gestione el acceso al IAM Identity Center de los destinatarios de las alarmas](#)

Creación de una función de Lambda

AWS IoT Events proporciona una función Lambda que permite a las alarmas enviar y recibir correos electrónicos y SMS notificaciones.

Requisitos

Se imponen los siguientes requisitos al crear una función de Lambda para las alarmas:

- Si tu alarma envía correos electrónicos o SMS notificaciones, debes tener un IAM rol que te AWS Lambda permita trabajar con Amazon SES y AmazonSNS.

Ejemplo de política:

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

- Debes elegir la misma AWS región para ambos AWS IoT Events y AWS Lambda. Para obtener la lista de regiones admitidas, consulte [Puntos de conexión y cuotas de AWS IoT Events](#) y [Puntos de conexión y cuotas de AWS Lambda](#) en Referencia general de Amazon Web Services.

Despliegue de una función de Lambda

En este tutorial se utiliza una AWS CloudFormation plantilla para implementar una función Lambda. Esta plantilla crea automáticamente un IAM rol que permite que la función Lambda funcione con Amazon y SES Amazon. SNS

A continuación, se muestra cómo usar el AWS Command Line Interface (AWS CLI) para crear una CloudFormation pila.

1. En la terminal de tu dispositivo, ejecuta `aws --version` para comprobar si has instalado el AWS CLI. Para obtener más información, consulte [Installing the AWS CLI](#) en la Guía del usuario de AWS Command Line Interface .
2. Ejecute `aws configure list` para comprobar si ha configurado AWS CLI la AWS región que tiene todos sus AWS recursos para este tutorial. Para obtener más información, consulte [Instalar o actualizar a la última versión de AWS CLI en la](#) Guía del AWS Command Line Interface usuario
3. Descargue la CloudFormation plantilla, [notificationLambda.template.yaml.zip](#).

Note

Si encuentra dificultades para descargar el archivo, la plantilla también está disponible en [CloudFormation plantilla](#).

4. Descomprima el contenido y guárdelo localmente como `notificationLambda.template.yaml`.
5. Abra un terminal en su dispositivo y vaya hasta el directorio en que descargó el archivo `notificationLambda.template.yaml`.
6. Para crear una CloudFormation pila, ejecuta el siguiente comando:

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

Puede modificar esta CloudFormation plantilla para personalizar la función Lambda y su comportamiento.

Note

AWS Lambda Reintenta los errores de función dos veces. Si la función no tiene capacidad suficiente para gestionar todas las solicitudes entrantes, los eventos podrían esperar en la cola durante horas o días para su envío a la función. Puede configurar una cola de mensajes no entregados (DLQ) en la función para capturar los eventos que no se procesaron correctamente. Para obtener más información, consulte [Invocación asíncrona](#) en la Guía para desarrolladores de AWS Lambda .

También puedes crear o configurar la pila en la consola. CloudFormation Para obtener más información, consulte [Trabajo con pilas](#), en la Guía del usuario de AWS CloudFormation .

Creación de una función de Lambda personalizada

Puede crear una función de Lambda o modificar la proporcionada con AWS IoT Events.

Se imponen los siguientes requisitos al crear una función de Lambda personalizada.

- Agregue permisos que permitan a la función Lambda realizar acciones específicas y acceder AWS a los recursos.
- Si usa la función Lambda proporcionada por AWS IoT Events, asegúrese de elegir el motor de ejecución de Python 3.7.

Ejemplo de función de Lambda:

```
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 SS0. 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.utcnow().timestamp(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')
```

Para obtener más información, consulte [¿Qué es AWS Lambda?](#) en la Guía para desarrolladores de AWS Lambda .

CloudFormation plantilla

Utilice la siguiente CloudFormation plantilla para crear la función Lambda.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Notification Lambda for Alarm Model'
Resources:
  NotificationLambdaRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: lambda.amazonaws.com
            Action: sts:AssumeRole
      Path: "/"
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/AWSLambdaExecute'
    Policies:
      - PolicyName: "NotificationLambda"
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: "Allow"
              Action:
                - "ses:GetIdentityVerificationAttributes"
                - "ses:SendEmail"
                - "ses:VerifyEmailIdentity"
```

```

    Resource: "*"
  - Effect: "Allow"
    Action:
      - "sns:Publish"
      - "sns:OptInPhoneNumber"
      - "sns:CheckIfPhoneNumberIsOptedOut"
    Resource: "*"
  - Effect: "Deny"
    Action:
      - "sns:Publish"
    Resource: "arn:aws:sns:*:*:*"

```

NotificationLambdaFunction:

Type: AWS::Lambda::Function

Properties:

Role: !GetAtt NotificationLambdaRole.Arn

Runtime: python3.7

Handler: index.lambda_handler

Timeout: 300

MemorySize: 3008

Code:

```

ZipFile: |
    import boto3
    import json
    import logging
    import datetime
    logger = logging.getLogger()
    logger.setLevel(logging.INFO)
    ses = boto3.client('ses')
    sns = boto3.client('sns')
    def check_value(target):
        if target:
            return True
        return False

```

Check whether email is verified. Only verified emails are allowed to send emails to or from.

```

def check_email(email):
    if not check_value(email):
        return False
    result = ses.get_identity_verification_attributes(Identities=[email])
    attr = result['VerificationAttributes']
    if (email not in attr or attr[email]['VerificationStatus'] != 'Success'):
        logging.info('Verification email for {} sent. You must have all the
emails verified before sending email.'.format(email))

```

```

        ses.verify_email_identity(EmailAddress=email)
        return False
    return True

    # Check whether the phone holder has opted out of receiving SMS messages from
    your account
    def check_phone_number(phone_number):
        try:
            result = sns.check_if_phone_number_is_opted_out(phoneNumber=phone_number)
            if (result['isOptedOut']):
                logger.info('phoneNumber {} is not opt in of receiving SMS messages.
                Phone number must be opt in first.'.format(phone_number))
                return False
            return True
        except Exception as e:
            logging.error('Your phone number {} must be in E.164 format in SSO.
            Exception thrown: {}'.format(phone_number, e))
            return False

    def check_emails(emails):
        result = True
        for email in emails:
            if not check_email(email):
                result = False
        return result

    def lambda_handler(event, context):
        logging.info('Received event: ' + json.dumps(event))
        nep = json.loads(event.get('notificationEventPayload'))
        alarm_state = nep['alarmState']
        default_msg = 'Alarm ' + alarm_state['stateName'] + '\n'
        timestamp =
datetime.datetime.utcnow().timestamp(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')

```

Uso de la función de Lambda proporcionada por AWS IoT Events

Se imponen los siguientes requisitos al utilizar la función de Lambda proporcionada por AWS IoT Events para administrar sus notificaciones de alarma:

- Debe verificar la dirección de correo electrónico que envía las notificaciones por correo electrónico en Amazon Simple Email Service (AmazonSES). Para obtener más información, consulte [Verificación de direcciones de correo electrónico en Amazon SES](#), en la Guía para desarrolladores de Amazon Simple Email Service.

Si recibe un enlace de verificación, pulse el enlace para verificar su dirección de correo electrónico. También debería controlar su carpeta de correo no deseado en busca del correo electrónico de verificación.

- Si la alarma envía SMS notificaciones, debe utilizar el formato E.164 para los números de teléfono internacionales. Este formato contiene `+<country-calling-code><area-code><phone-number>`.

Ejemplos de números de teléfono:

País	Número de teléfono local	Número en formato E.164
Estados Unidos	206-555-0100	+12065550100
Reino Unido	020-1234-1234	+442012341234
Lituania	8+601+12345	+37060112345

Para encontrar el prefijo telefónico de un país, visite countrycode.org.

La función Lambda proporcionada por AWS IoT Events comprueba si se utilizan números de teléfono con formato E.164. Sin embargo, no verifica los números de teléfono. Si te aseguras de haber introducido los números de teléfono correctos pero no has recibido SMS notificaciones, puedes ponerte en contacto con los operadores de telefonía. Es posible que los operadores bloqueen los mensajes.

Gestione el acceso al IAM Identity Center de los destinatarios de las alarmas

AWS IoT Events AWS IAM Identity Center se utiliza para gestionar el SSO acceso de los destinatarios de las alarmas. Para permitir que la alarma envíe notificaciones a los destinatarios, debe activar IAM Identity Center y añadir los destinatarios a su almacén de IAM Identity Center. Para

obtener más información, consulte [Adición de usuarios](#) en la Guía de usuarios de AWS IAM Identity Center .

 Important

- Debe elegir la misma AWS región para AWS IoT Events Identity Center y para IAM Identity Center. AWS Lambda
- AWS Organizations solo admite una región de IAM Identity Center a la vez. Si quiere que IAM Identity Center esté disponible en una región diferente, primero debe eliminar la configuración actual del IAM Identity Center. Para obtener más información, consulte los [datos regionales de IAM Identity Center](#) en la Guía AWS IAM Identity Center del usuario.

Seguridad en AWS IoT Events

La seguridad en la nube AWS es la máxima prioridad. Como AWS cliente, usted se beneficia de una arquitectura de centro de datos y red diseñada para cumplir con los requisitos de las organizaciones más sensibles a la seguridad.

La seguridad es una responsabilidad compartida entre usted AWS y usted. El [modelo de responsabilidad compartida](#) la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta AWS los servicios en la AWS nube. AWS también le proporciona servicios que puede utilizar de forma segura. Auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los [programas de conformidad de AWS](#). Para obtener más información sobre los programas de cumplimiento aplicables AWS IoT Events, consulte los [AWS servicios incluidos en el ámbito de aplicación por programa de cumplimiento](#).
- Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice. Usted también es responsable de otros factores incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y los reglamentos aplicables.

Esta documentación le ayudará a entender cómo aplicar el modelo de responsabilidad compartida cuando lo utilice AWS IoT Events. Los siguientes temas muestran cómo configurarlo AWS IoT Events para cumplir sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros AWS servicios que pueden ayudarle a supervisar y proteger sus AWS IoT Events recursos.

Temas

- [Administración de identidad y acceso para AWS IoT Events](#)
- [Monitorización AWS IoT Events](#)
- [Validación de conformidad para AWS IoT Events](#)
- [Resiliencia en AWS IoT Events](#)
- [Seguridad de la infraestructura en AWS IoT Events](#)

Administración de identidad y acceso para AWS IoT Events

AWS Identity and Access Management (IAM) es un AWS servicio que ayuda al administrador a controlar de forma segura el acceso a AWS los recursos. IAM los administradores controlan quién

puede autenticarse (iniciar sesión) y quién está autorizado (tiene permisos) para usar AWS IoT Events los recursos. IAM es un AWS servicio que puede utilizar sin coste adicional.

Temas

- [Público](#)
- [Autenticación con identidades](#)
- [Administración de acceso mediante políticas](#)
- [Más información](#)
- [¿Cómo AWS IoT Events funciona con IAM](#)
- [AWS IoT Events ejemplos de políticas basadas en la identidad](#)
- [Prevención de la sustitución confusa entre servicios](#)
- [Solución de problemas AWS IoT Events de identidad y acceso](#)

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo en el que se realice AWS IoT Events.

Usuario del servicio: si utiliza el AWS IoT Events servicio para realizar su trabajo, el administrador le proporcionará las credenciales y los permisos que necesita. A medida que vaya utilizando más AWS IoT Events funciones para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en AWS IoT Events, consulte [Solución de problemas AWS IoT Events de identidad y acceso](#).

Administrador de servicios: si estás a cargo de AWS IoT Events los recursos de tu empresa, probablemente tengas acceso total a ellos AWS IoT Events. Su trabajo consiste en determinar a qué AWS IoT Events funciones y recursos deben acceder los usuarios del servicio. A continuación, debe enviar solicitudes al IAM administrador para cambiar los permisos de los usuarios del servicio. Revise la información de esta página para comprender los conceptos básicos del IAM. Para obtener más información sobre cómo su empresa puede utilizar IAM con AWS IoT Events, consulte [¿Cómo AWS IoT Events funciona con IAM](#).

IAM administrador: si es IAM administrador, puede que desee obtener más información sobre cómo puede redactar políticas para administrar el acceso a ellas AWS IoT Events. Para ver ejemplos de

políticas AWS IoT Events basadas en la identidad que puede utilizar IAM, consulte [AWS IoT Events ejemplos de políticas basadas en la identidad](#)

Autenticación con identidades

La autenticación es la forma de iniciar sesión AWS con sus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como IAM usuario o asumiendo un IAM rol.

Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (IAM Identity Center), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, el administrador configuró previamente la federación de identidades mediante roles. IAM Cuando accede AWS mediante la federación, asume indirectamente un rol.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte [Cómo iniciar sesión Cuenta de AWS en su](#) Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS incluye un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte [Firmar AWS API las solicitudes](#) en la Guía del IAM usuario.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactorial (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte [Autenticación multifactorial](#) en la Guía del AWS IAM Identity Center usuario y [Uso de la autenticación multifactorial \(MFA\) AWS en](#) la Guía del IAM usuario.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para

las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de tareas que requieren que inicie sesión como usuario root, consulte [Tareas que requieren credenciales de usuario root](#) en la Guía del IAM usuario.

Usuarios y grupos de IAM

Un [IAMusuario](#) es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos utilizar credenciales temporales en lugar de crear IAM usuarios con credenciales de larga duración, como contraseñas y claves de acceso. Sin embargo, si tiene casos de uso específicos que requieren credenciales a largo plazo con IAM los usuarios, le recomendamos que rote las claves de acceso. Para obtener más información, consulte [Rotar las claves de acceso con regularidad para los casos de uso que requieran credenciales de larga duración](#) en la Guía del IAM usuario.

Un [IAMgrupo](#) es una identidad que especifica un conjunto de IAM usuarios. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos para grandes conjuntos de usuarios. Por ejemplo, puede asignar un nombre a un grupo IAMAdminsy concederle permisos para administrar IAM los recursos.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales de larga duración permanentes; no obstante, los roles proporcionan credenciales temporales. Para obtener más información, consulte [Cuándo crear un IAM usuario \(en lugar de un rol\)](#) en la Guía del IAM usuario.

IAMroles

Un [IAMrol](#) es una identidad dentro de tu Cuenta de AWS que tiene permisos específicos. Es similar a un IAM usuario, pero no está asociado a una persona específica. Puede asumir temporalmente un IAM rol en el AWS Management Console [cambiando de rol](#). Puede asumir un rol llamando a una AWS API operación AWS CLI o utilizando una operación personalizadaURL. Para obtener más información sobre los métodos de uso de roles, consulte [Uso de IAM roles](#) en la Guía del IAM usuario.

IAMlos roles con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al

rol y se le conceden los permisos define el rol. Para obtener información sobre los roles para la federación, consulte [Creación de un rol para un proveedor de identidad externo](#) en la Guía del IAM usuario. Si usa IAM Identity Center, configura un conjunto de permisos. Para controlar a qué pueden acceder sus identidades después de autenticarse, IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM. Para obtener información acerca de los conjuntos de permisos, consulte [Conjuntos de permisos](#) en la Guía del usuario de AWS IAM Identity Center .

- **Permisos IAM de usuario temporales:** un IAM usuario o rol puede asumir un IAM rol para asumir temporalmente diferentes permisos para una tarea específica.
- **Acceso multicuenta:** puedes usar un IAM rol para permitir que alguien (un responsable de confianza) de una cuenta diferente acceda a los recursos de tu cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunos Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para conocer la diferencia entre las funciones y las políticas basadas en recursos para el acceso multicuenta, consulta el tema sobre el acceso a los [recursos entre cuentas IAM en](#) la Guía del IAM usuario.
- **Acceso entre servicios:** algunos Servicios de AWS utilizan funciones en otros. Servicios de AWS Por ejemplo, cuando realizas una llamada en un servicio, es habitual que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
- **Sesiones de acceso directo (FAS):** cuando utilizas un IAM usuario o un rol para realizar acciones en AWS ellas, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama a un Servicio de AWS, junto con los que solicitan, Servicio de AWS para realizar solicitudes a los servicios descendentes. FAS las solicitudes solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información detallada sobre la política a la hora de realizar FAS solicitudes, consulte [Reenviar las sesiones de acceso](#).
- **Función de servicio:** una función de servicio es una [IAM función](#) que un servicio asume para realizar acciones en su nombre. Un IAM administrador puede crear, modificar y eliminar un rol de servicio desde dentro IAM. Para obtener más información, consulte [Crear un rol para delegar permisos Servicio de AWS en un rol](#) en el IAM Manual del usuario.
- **Función vinculada a un servicio:** una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un Servicio de AWS. El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y

son propiedad del servicio. Un IAM administrador puede ver los permisos de los roles vinculados al servicio, pero no editarlos.

- Aplicaciones que se ejecutan en Amazon EC2: puedes usar un IAM rol para administrar las credenciales temporales de las aplicaciones que se ejecutan en una EC2 instancia y que realizan AWS CLI o AWS API solicitudes. Esto es preferible a almacenar las claves de acceso en la EC2 instancia. Para asignar un AWS rol a una EC2 instancia y ponerlo a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite que los programas que se ejecutan en la EC2 instancia obtengan credenciales temporales. Para obtener más información, consulte [Uso de un IAM rol para conceder permisos a aplicaciones que se ejecutan en EC2 instancias de Amazon](#) en la Guía del IAM usuario.

Para saber si se deben usar IAM roles o IAM usuarios, consulte [Cuándo crear un IAM rol \(en lugar de un usuario\)](#) en la Guía del IAM usuario.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como JSON documentos. Para obtener más información sobre la estructura y el contenido de los documentos de JSON políticas, consulte [Descripción general de JSON las políticas](#) en la Guía del IAM usuario.

Los administradores pueden usar AWS JSON las políticas para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Para conceder a los usuarios permiso para realizar acciones en los recursos que necesitan, un IAM administrador puede crear IAM políticas. A continuación, el administrador puede añadir las IAM políticas a las funciones y los usuarios pueden asumir las funciones.

IAM las políticas definen los permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de AWS Management Console AWS CLI, el o el AWS API.

Políticas basadas en identidad

Las políticas basadas en la identidad son documentos de política de JSON permisos que se pueden adjuntar a una identidad, como un IAM usuario, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener información sobre cómo crear una política basada en la identidad, consulte [Creación de IAM políticas](#) en la Guía del usuario. IAM

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y funciones de su empresa. Cuenta de AWS Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para saber cómo elegir entre una política gestionada o una política integrada, consulte [Elegir entre políticas gestionadas y políticas integradas en la Guía del IAM](#) usuario.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- **Límites de permisos:** un límite de permisos es una función avanzada en la que se establecen los permisos máximos que una política basada en la identidad puede conceder a una IAM entidad (IAMusuario o rol). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte los [límites de los permisos para IAM las entidades](#) en la Guía del IAMusuario.
- **Políticas de control de servicios (SCPs):** SCPs son JSON políticas que especifican los permisos máximos para una organización o unidad organizativa (OU) AWS Organizations. AWS Organizations es un servicio para agrupar y administrar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilitas todas las funciones de una organización, puedes aplicar políticas de control de servicios (SCPs) a una o a todas tus cuentas. SCPLimita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una Usuario raíz de la cuenta de AWS. Para obtener más información sobre OrganizationsSCPs, consulte las [políticas de control de servicios](#) en la Guía del AWS Organizations usuario.

- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información, consulte [las políticas de sesión](#) en la Guía del IAM usuario.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo se AWS determina si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la [lógica de evaluación de políticas](#) en la Guía del IAM usuario.

Más información

Para obtener más información sobre la administración de identidades y accesos AWS IoT Events, consulte las páginas siguientes:

- [¿Cómo AWS IoT Events funciona con IAM](#)
- [Solución de problemas AWS IoT Events de identidad y acceso](#)

¿Cómo AWS IoT Events funciona con IAM

Antes de administrar el acceso a AWS IoT Events, debe comprender qué IAM funciones están disponibles para su uso AWS IoT Events. IAM Para obtener una visión general de cómo funcionan AWS los servicios AWS IoT Events y otros serviciosIAM, consulte [AWS los servicios con los que funcionan IAM](#) en la Guía del IAM usuario.

Temas

- [AWS IoT Events políticas basadas en la identidad](#)
- [Políticas de AWS IoT Events basadas en recursos](#)
- [Autorización basada en etiquetas de AWS IoT Events](#)
- [AWS IoT Events IAMroles](#)

AWS IoT Events políticas basadas en la identidad

Con las políticas IAM basadas en la identidad, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. AWS IoT Events admite acciones, recursos y claves de condición específicos. Para obtener información sobre todos los elementos que se utilizan en una JSON política, consulte la [referencia a los elementos de la IAM JSON política](#) en la Guía del IAM usuario.

Acciones

El `Action` elemento de una política IAM basada en la identidad describe la acción o las acciones específicas que la política permitirá o denegará. Las acciones políticas suelen tener el mismo nombre que la operación asociada AWS API. La acción se utiliza en una política para otorgar permisos para realizar la operación asociada.

Las acciones políticas AWS IoT Events utilizan el siguiente prefijo antes de la acción: `iotevents:`. Por ejemplo, para conceder permiso a alguien para crear una AWS IoT Events entrada con la AWS IoT Events `CreateInput` API operación, debes incluir la `iotevents:CreateInput` acción en su política. Para conceder permiso a alguien para enviar una entrada con la AWS IoT Events `BatchPutMessage` API operación, debes incluir la `iotevents-data:BatchPutMessage` acción en su política. Las declaraciones de política deben incluir un `NotAction` elemento `Action` o. AWS IoT Events define su propio conjunto de acciones que describen las tareas que puede realizar con este servicio.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo:

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

Puede utilizar caracteres comodín para especificar varias acciones (*). Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Describe`, incluya la siguiente acción:

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

Para ver una lista de AWS IoT Events acciones, consulte [las acciones definidas por AWS IoT Events](#) en la Guía del IAM usuario.

Recursos

El elemento `Resource` especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Para especificar un recurso, utilice un ARN o un comodín (*) para indicar que la declaración se aplica a todos los recursos.

El recurso del modelo de AWS IoT Events detector tiene lo siguiente: ARN

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

Para obtener más información sobre el formato de ARNs, consulte [Identificar AWS los recursos con los nombres de los recursos de Amazon \(ARNs\)](#).

Por ejemplo, para especificar el modelo de Foobar detector en su declaración, utilice lo siguiente ARN:

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

Para especificar todas las instancias que pertenecen a una cuenta específica, utilice el carácter comodín (*):

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

Algunas AWS IoT Events acciones, como las de creación de recursos, no se pueden realizar en un recurso específico. En dichos casos, debe utilizar el carácter comodín (*).

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

Algunas AWS IoT Events API acciones implican varios recursos. Por ejemplo, `CreateDetectorModel` hace referencia a las entradas en sus declaraciones de condición, por lo que un usuario debe tener permisos para utilizar la entrada y el modelo de detector. Para especificar varios recursos en una sola sentencia, sepárelos ARNs con comas.

```
"Resource": [  
    "resource1",  
    "resource2"
```

Para ver una lista de los tipos de AWS IoT Events recursos y sus respectivos tiposARNs, consulte [los recursos definidos por AWS IoT Events](#) en la Guía del IAM usuario. Para saber con qué acciones puede especificar cada recurso, consulte [Acciones definidas por AWS IoT Events](#). ARN

Claves de condición

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilizan [operadores de condición](#), tales como igual o menor que, para que coincida la condición de la política con valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación AND lógica. Si especifica varios valores para una sola clave de condición, AWS evalúa la condición mediante una OR operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder a un usuario permiso para acceder a un recurso solo si está etiquetado con su nombre de usuario. Para obtener más información, consulte [los elementos IAM de política: variables y etiquetas](#) en la Guía del IAM usuario.

AWS IoT Events no proporciona ninguna clave de condición específica del servicio, pero admite el uso de algunas claves de condición globales. Para ver todas las claves de condición AWS globales, consulte las claves de [contexto de condición AWS globales](#) en la Guía del IAM usuario».

Ejemplos

Para ver ejemplos de políticas AWS IoT Events basadas en la identidad, consulte. [AWS IoT Events ejemplos de políticas basadas en la identidad](#)

Políticas de AWS IoT Events basadas en recursos

AWS IoT Events no es compatible con las políticas basadas en los recursos». Para ver un ejemplo de una página detallada de política basada en recursos, consulte <https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html>.

Autorización basada en etiquetas de AWS IoT Events

Puedes adjuntar etiquetas a AWS IoT Events los recursos o pasarles etiquetas en una solicitud. AWS IoT Events Para controlar el acceso en función de etiquetas, debe proporcionar información

de las etiquetas en el [elemento de condición](#) de una política utilizando las claves de condición `iotevents:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`. Para obtener más información acerca del etiquetado de recursos de AWS IoT Events, consulte [Etiquetar sus recursos AWS IoT Events](#).

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte [Visualización de AWS IoT Events las entradas en función de las etiquetas](#).

AWS IoT Events IAMroles

Un [IAMrol](#) es una entidad dentro de tu Cuenta de AWS que tiene permisos específicos.

Utilizar credenciales temporales con AWS IoT Events

Puede usar credenciales temporales para iniciar sesión con la federación, asumir un IAM rol o asumir un rol multicuenta. Las credenciales de seguridad temporales se obtienen llamando a AWS Security Token Service (AWS STS) API operaciones como [AssumeRole](#) o [GetFederationToken](#).

AWS IoT Events no admite el uso de credenciales temporales.

Roles vinculados al servicio

Las [funciones vinculadas al servicio](#) permiten a AWS los servicios acceder a los recursos de otros servicios para completar una acción en tu nombre. Los roles vinculados al servicio aparecen en tu IAM cuenta y son propiedad del servicio. Un IAM administrador puede ver los permisos de los roles vinculados al servicio, pero no editarlos.

AWS IoT Events no admite los roles vinculados al servicio.

Roles de servicio

Esta característica permite que un servicio asuma un [rol de servicio](#) en su nombre. Este rol permite que el servicio obtenga acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles de servicio aparecen en su IAM cuenta y son propiedad de la cuenta. Esto significa que un IAM administrador puede cambiar los permisos de este rol. Sin embargo, hacerlo podría deteriorar la funcionalidad del servicio.

AWS IoT Events admite funciones de servicio.

AWS IoT Events ejemplos de políticas basadas en la identidad

De forma predeterminada, los usuarios y los roles no tienen permiso para crear o modificar AWS IoT Events recursos. Tampoco pueden realizar tareas con AWS Management Console AWS CLI, o AWS API. IAMEI administrador debe crear IAM políticas que concedan a los usuarios y roles permisos para realizar API operaciones específicas en los recursos específicos que necesitan. El administrador debe asociar esas políticas a los usuarios o grupos que necesiten esos permisos.

Para obtener información sobre cómo crear una política IAM basada en la identidad utilizando estos documentos de JSON política de ejemplo, consulte [Creación de políticas en la JSON pestaña de la Guía del IAM usuario](#).

Temas

- [Prácticas recomendadas sobre las políticas](#)
- [Mediante la consola de AWS IoT Events](#)
- [Cómo permitir a los usuarios que vean sus propios permisos](#)
- [Acceso a una entrada de AWS IoT Events](#)
- [Visualización de AWS IoT Events las entradas en función de las etiquetas](#)

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidad son muy eficaces. Determinan si alguien puede crear AWS IoT Events recursos de tu cuenta, acceder a ellos o eliminarlos. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience a utilizar las políticas AWS gestionadas: para empezar a AWS IoT Events utilizarlas rápidamente, utilice las políticas AWS gestionadas para conceder a sus empleados los permisos que necesitan. Estas políticas ya están disponibles en su cuenta, y las mantiene y actualiza AWS. Para obtener más información, consulte [Cómo empezar a usar permisos con políticas AWS administradas](#) en la Guía del IAM usuario.
- Conceder privilegios mínimos: al crear políticas personalizadas, conceda solo los permisos necesarios para llevar a cabo una tarea. Comience con un conjunto mínimo de permisos y conceda permisos adicionales según sea necesario. Por lo general, es más seguro que comenzar con permisos demasiado tolerantes e intentar hacerlos más estrictos más adelante. Para obtener más información, consulte [Otorgar privilegios mínimos](#) en la Guía del IAM usuario.

- **Habilitar MFA operaciones confidenciales:** para mayor seguridad, solicite a los usuarios que utilicen la autenticación multifactorial (MFA) para acceder a recursos u API operaciones confidenciales. Para obtener más información, consulte [Uso de la autenticación multifactorial \(MFA\) AWS en la Guía del IAM usuario](#).
- **Utilizar condiciones de política para mayor seguridad:** en la medida en que sea práctico, defina las condiciones en las que sus políticas basadas en identidad permitan el acceso a un recurso. Por ejemplo, puede escribir condiciones para especificar un rango de direcciones IP permitidas desde el que debe proceder una solicitud. También puede escribir condiciones para permitir las solicitudes solo dentro de un intervalo de fechas o horas específico, o para requerir el uso de SSL o MFA. Para obtener más información, consulte [los elementos IAM JSON de la política: Condición](#) en la Guía del IAM usuario.

Mediante la consola de AWS IoT Events

Para acceder a la AWS IoT Events consola, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y ver detalles sobre los AWS IoT Events recursos de su cuenta Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

Para garantizar que esas entidades puedan seguir utilizando la AWS IoT Events consola, adjunte también la siguiente política AWS gestionada a las entidades. Para obtener más información, consulte [Añadir permisos a un usuario](#) en la Guía del IAM usuario:

```
{
  "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}"
  }
]
}

```

No es necesario conceder permisos mínimos de consola a los usuarios que realicen llamadas únicamente al AWS CLI o al AWS API. En su lugar, permita el acceso únicamente a las acciones que coincidan con la API operación que está intentando realizar.

Cómo permitir a los usuarios que vean sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para completar esta acción en la consola o mediante programación mediante la AWS CLI tecla o. AWS API

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "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": "*"
}
]
}

```

Acceso a una entrada de AWS IoT Events

En este ejemplo, quieres conceder a un usuario tu Cuenta de AWS acceso a una de tus AWS IoT Events entradas, `exampleInput`. También desea permitir al usuario añadir, actualizar y eliminar entradas.

La política concede los permisos `iotevents:ListInputs`, `iotevents:DescribeInput`, `iotevents>CreateInput`, `iotevents>DeleteInput` y `iotevents:UpdateInput` al usuario. Para ver un ejemplo de tutorial del Amazon Simple Storage Service (Amazon S3) que concede permisos a los usuarios y los prueba mediante la consola, [consulte Controlar el acceso a un bucket](#) con políticas de usuario.

```

{
  "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/*"
  }
]
}

```

Visualización de AWS IoT Events las entradas en función de las etiquetas

Puede utilizar las condiciones de su política basada en identidad para controlar el acceso a los recursos de AWS IoT Events basados en etiquetas. En este ejemplo se muestra cómo se puede crear una política que permita ver un *input*. Sin embargo, el permiso solo se concede si el *input* la etiqueta Owner tiene el valor del nombre de usuario de ese usuario. Esta política también proporciona los permisos necesarios para llevar a cabo esta acción en la consola.

```

{
  "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}"}
    }
  }
]
```

También puede adjuntar esta política al usuario de en su cuenta. Si un usuario llamado `richard-roe` intenta ver un AWS IoT Events `input`, el `input` debe estar etiquetado `Owner=richard-roe owner=richard-roe`. De lo contrario, se le deniega el acceso. La clave de la etiqueta de condición `Owner` coincide con los nombres de las claves de condición `Owner` y `owner` porque no distinguen entre mayúsculas y minúsculas. Para obtener más información, consulte [los elementos IAM JSON de la política: condición](#) en la Guía del IAM usuario.

Prevención de la sustitución confusa entre servicios

Note

- El AWS IoT Events servicio solo permite a los clientes usar roles para iniciar acciones en la misma cuenta en la que se creó el recurso. Esto significa que no se puede realizar un ataque de suplente confuso con este servicio.
- Esta página sirve de referencia para que los clientes vean cómo funciona el problema del suplente confuso y cómo puede prevenirse en caso de que se permitieran recursos entre cuentas en el servicio de AWS IoT Events .

El problema del suplente confuso es una cuestión de seguridad en la que una entidad que no tiene permiso para realizar una acción puede obligar a una entidad con más privilegios a realizar la acción. En AWS, la suplantación de identidad entre servicios puede provocar el confuso problema de un diputado.

La suplantación entre servicios puede producirse cuando un servicio (el servicio que lleva a cabo las llamadas) llama a otro servicio (el servicio al que se llama). El servicio que lleva a cabo las llamadas se puede manipular para utilizar sus permisos a fin de actuar en función de los recursos de otro cliente de una manera en la que no debe tener permiso para acceder. Para evitarlo, AWS proporciona herramientas que le ayudan a proteger los datos de todos los servicios cuyos directores de servicio tengan acceso a los recursos de su cuenta.

Recomendamos utilizar las claves de contexto de condición [aws:SourceAccount](#) global [aws:SourceArn](#) y las claves contextuales en las políticas de recursos para limitar los permisos que se AWS IoT Events otorgan a otro servicio al recurso. Si el `aws:SourceArn` valor no contiene el ID de la cuenta, como un bucket de Amazon S3ARN, debe usar ambas claves de contexto de condición global para limitar los permisos. Si utiliza claves de contexto de condición global y el valor de `aws:SourceArn` contiene el ID de cuenta, el valor de `aws:SourceAccount` y la cuenta en el valor de `aws:SourceArn` deben utilizar el mismo ID de cuenta cuando se utiliza en la misma instrucción de política.

Utilice `aws:SourceArn` si desea que solo se asocie un recurso al acceso entre servicios. Utilice `aws:SourceAccount` si quiere permitir que cualquier recurso de esa cuenta se asocie al uso entre servicios. El valor de `aws:SourceArn` debe ser el modelo de detector o el modelo de alarma asociado a la solicitud `sts:AssumeRole`.

La forma más eficaz de protegerse contra el confuso problema de los adjuntos es utilizar la clave de contexto ARN de la condición `aws:SourceArn` global con todo el recurso. Si no conoce la totalidad ARN del recurso o si está especificando varios recursos, utilice la clave de condición del contexto `aws:SourceArn` global con caracteres comodín (*) para las partes desconocidas del ARN. Por ejemplo, `arn:aws:iotevents:*:123456789012:*`.

En los ejemplos siguientes se muestra cómo utilizar las claves de contexto `aws:SourceArn` y de condición `aws:SourceAccount` global AWS IoT Events para evitar el confuso problema de los diputados.

Temas

- [Ejemplo: acceso a un modelo de detector](#)
- [Ejemplo: acceder a un modelo de alarma](#)
- [Ejemplo: acceder a un recurso en una región específica](#)
- [Ejemplo: opciones de registro](#)

Ejemplo: acceso a un modelo de detector

En este ejemplo, la función proporcionada solo se puede utilizar para acceder al modelo de detector denominado `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"
        }
      }
    }
  ]
}
```

Ejemplo: acceder a un modelo de alarma

El siguiente ejemplo muestra cómo conceder AWS IoT Events acceso para enviar datos a una alarma y, al mismo tiempo, restringir los permisos para modificar la configuración de la alarma mediante una IAM política.

El siguiente rol solo se puede utilizar para acceder a cualquier modelo de alarma.

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

```

Ejemplo: acceder a un recurso en una región específica

El siguiente ejemplo muestra un rol que puede utilizar para acceder a un recurso en una región especificada. La región de este ejemplo es *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:*"
        }
      }
    }
  ]
}

```

```
}
```

Ejemplo: opciones de registro

Para proporcionar una función a las opciones de registro, permita que cada AWS IoT Events recurso la asuma. Utilice un comodín (*) tanto para el tipo como para el nombre del recurso.

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

Solución de problemas AWS IoT Events de identidad y acceso

Utilice la siguiente información como ayuda para diagnosticar y solucionar los problemas más comunes que pueden surgir al trabajar con AWS IoT Events y IAM.

Temas

- [No estoy autorizado a realizar ninguna acción en AWS IoT Events](#)
- [No tengo autorización para realizar iam:PassRole](#)
- [Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS IoT Events recursos](#)

No estoy autorizado a realizar ninguna acción en AWS IoT Events

Si AWS Management Console le indica que no está autorizado a realizar una acción, debe ponerse en contacto con su administrador para obtener ayuda. Su administrador es la persona que le facilitó su nombre de usuario y contraseña.

El siguiente ejemplo de error se produce cuando el usuario IAM mateojackson intenta utilizar la consola para ver los detalles de un *input* pero no tiene `iotevents:ListInputs` permisos.

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

En este caso, Mateo pide a su administrador que actualice sus políticas de forma que pueda obtener acceso al recurso *my-example-input* mediante la acción `iotevents:ListInput`.

No tengo autorización para realizar `iam:PassRole`

Si recibe un error que indica que no tiene autorización para realizar la acción `iam:PassRole`, las políticas deben actualizarse a fin de permitirle pasar un rol a AWS IoT Events.

Algunos Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada a un servicio. Para ello, debe tener permisos para transferir el rol al servicio.

El siguiente ejemplo de error se produce cuando un IAM usuario denominado marymajor intenta utilizar la consola para realizar una acción en ella. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

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

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con AWS el administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis AWS IoT Events recursos

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que respaldan las políticas basadas en recursos o las listas de control de acceso (ACLs), puedes usar esas políticas para permitir que las personas accedan a tus recursos.

Consulte los siguientes temas para determinar sus mejores opciones:

- Para saber si AWS IoT Events es compatible con estas funciones, consulte [¿Cómo AWS IoT Events funciona con IAM.](#)
- Para obtener información sobre cómo proporcionar acceso a los recursos de su propiedad, consulte [Proporcionar acceso a un IAM usuario en otro Cuenta de AWS de su propiedad](#) en la Guía del IAM usuario. Cuentas de AWS
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta [Cómo permitir el acceso a recursos que Cuentas de AWS son propiedad de terceros](#) en la Guía del IAM usuario.
- Para obtener información sobre cómo proporcionar acceso mediante la federación de identidades, consulte [Proporcionar acceso a usuarios autenticados externamente \(federación de identidades\)](#) en la Guía del IAM usuario.
- Para saber la diferencia entre el uso de roles y políticas basadas en recursos para el acceso entre cuentas, consulte el acceso a [recursos entre cuentas IAM en la Guía](#) del usuario. IAM

Monitorización AWS IoT Events

La supervisión es una parte importante del mantenimiento de la confiabilidad, la disponibilidad y el rendimiento de AWS IoT Events sus AWS soluciones. Debe recopilar los datos de supervisión de todas las partes de la AWS solución para poder depurar con mayor facilidad una falla multipunto en caso de que se produzca. Antes de empezar a monitorizar AWS IoT Events, debe crear un plan de monitorización que incluya respuestas a las siguientes preguntas:

- ¿Cuáles son los objetivos de la monitorización?
- ¿Qué recursos va a monitorizar?
- ¿Con qué frecuencia va a monitorizar estos recursos?

- ¿Qué herramientas de monitorización va a utilizar?
- ¿Quién se encargará de realizar las tareas de monitoreo?
- ¿Quién debería recibir una notificación cuando surjan problemas?

El siguiente paso es establecer una línea base para el AWS IoT Events rendimiento normal de su entorno, midiendo el rendimiento en distintos momentos y bajo diferentes condiciones de carga. A medida que monitorice AWS IoT Events, almacene los datos de monitorización históricos para que pueda compararlos con los datos de rendimiento actual, identificar los patrones de rendimiento normal y las anomalías en el rendimiento, así como desarrollar métodos para la resolución de problemas.

Por ejemplo, si utilizas AmazonEC2, puedes supervisar la CPU utilización, la E/S del disco y la utilización de la red de tus instancias. Si el rendimiento no alcanza el nivel de referencia establecido, es posible que necesite reconfigurar u optimizar la instancia para reducir la CPU utilización, mejorar la E/S del disco o reducir el tráfico de red.

Temas

- [Herramientas de monitoreo](#)
- [Monitorización con Amazon CloudWatch](#)
- [Registrar AWS IoT Events API llamadas con AWS CloudTrail](#)

Herramientas de monitoreo

AWS proporciona varias herramientas que puede utilizar para supervisar AWS IoT Events. Puede configurar algunas de estas herramientas para que monitoricen por usted, pero otras herramientas requieren intervención manual. Le recomendamos que automatice las tareas de monitorización en la medida de lo posible.

Herramientas de monitoreo automatizadas

Puede utilizar las siguientes herramientas de supervisión automatizadas para observar AWS IoT Events e informar cuando algo va mal:

- Amazon CloudWatch Logs: supervise, almacene y acceda a sus archivos de registro desde AWS CloudTrail u otras fuentes. Para obtener más información, consulta [Uso de los CloudWatch paneles de Amazon](#) en la Guía del CloudWatch usuario de Amazon.

- AWS CloudTrail Supervisión de registros: comparta archivos de registro entre cuentas, supervise los archivos de CloudTrail registro en tiempo real enviándolos a CloudWatch Logs, cree aplicaciones de procesamiento de registros en Java y valide que sus archivos de registro no hayan cambiado después de su entrega. CloudTrail Para obtener más información, consulte [Trabajar con archivos de CloudTrail registro](#) en la Guía del AWS CloudTrail usuario.

Herramientas de monitoreo manuales

Otra parte importante del monitoreo AWS IoT Events consiste en monitorear manualmente los elementos que las CloudWatch alarmas no cubren. El AWS IoT Events panel de control y otros paneles de la AWS consola proporcionan una at-a-glance vista del estado de su AWS entorno. CloudWatch Le recomendamos que compruebe también los archivos de registro. AWS IoT Events

- La AWS IoT Events consola muestra:
 - Modelos de detector
 - Detectores
 - Entradas
 - Configuración
- La página de CloudWatch inicio muestra:
 - Alarmas y estado actual
 - Gráficos de alarmas y recursos
 - Estado de los servicios

Además, puede CloudWatch hacer lo siguiente:

- Crear [paneles personalizados](#) para monitorizar los servicios que le interesan
- Realizar un gráfico con los datos de las métricas para resolver problemas y descubrir tendencias
- Busque y explore todas sus métricas AWS de recursos
- Crear y editar las alarmas de notificación de problemas

Monitorización con Amazon CloudWatch

Al desarrollar o depurar un modelo de AWS IoT Events detector, necesita saber lo que AWS IoT Events está haciendo y los errores que encuentra. Amazon CloudWatch monitorea tus AWS recursos y las aplicaciones en las que AWS ejecutas en tiempo real. De este CloudWatch modo, obtiene visibilidad en todo el sistema sobre el uso de los recursos, el rendimiento de las aplicaciones y

el estado operativo. [Habilite el CloudWatch registro de Amazon al desarrollar modelos AWS IoT Events de detectores](#) contiene información sobre cómo habilitar el CloudWatch registro de AWS IoT Events. Para generar registros como el que se muestra a continuación, debe establecer el nivel de verbosidad en «Depurar» y proporcionar uno o más objetivos de depuración, con un nombre de modelo de detector y uno opcional. KeyValue

El siguiente ejemplo muestra una entrada de registro de CloudWatch DEBUG nivel generada por. AWS IoT Events

```
{
  "timestamp": "2019-03-15T15:56:29.412Z",
  "level": "DEBUG",
  "logMessage": "Summary of message evaluation",
  "context": "MessageEvaluation",
  "status": "Success",
  "messageId": "SensorAggregate_2th846h",
  "keyValue": "boiler_1",
  "detectorModelName": "BoilerAlarmDetector",
  "initialState": "high_temp_alarm",
  "initialVariables": {
    "high_temp_count": 1,
    "high_pressure_count": 1
  },
  "finalState": "no_alarm",
  "finalVariables": {
    "high_temp_count": 0,
    "high_pressure_count": 0
  },
  "message": "{\"temp\": 34.9, \"pressure\": 84.5}",
  "messageType": "CUSTOMER_MESSAGE",
  "conditionEvaluationResults": [
    {
      "result": "True",
      "eventName": "alarm_cleared",
      "state": "high_temp_alarm",
      "lifeCycle": "OnInput",
      "hasTransition": true
    },
    {
      "result": "Skipped",
      "eventName": "alarm_escalated",
      "state": "high_temp_alarm",
      "lifeCycle": "OnInput",
    }
  ]
}
```

```
    "hasTransition": true,
    "resultDetails": "Skipped due to transition from alarm_cleared event"
  },
  {
    "result": "True",
    "eventName": "should_recall_technician",
    "state": "no_alarm",
    "lifeCycle": "OnEnter",
    "hasTransition": true
  }
]
}
```

Registrar AWS IoT Events API llamadas con AWS CloudTrail

AWS IoT Events está integrado con AWS CloudTrail un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un AWS servicio en AWS IoT Events. CloudTrail captura todas las API llamadas AWS IoT Events como eventos, incluidas las llamadas desde la AWS IoT Events consola y desde las llamadas en código a AWS IoT Events APIs.

Si crea una ruta, puede habilitar la entrega continua de CloudTrail eventos a un bucket de Amazon S3, incluidos los eventos para AWS IoT Events. Si no configura una ruta, podrá ver los eventos más recientes en la CloudTrail consola, en el historial de eventos. Con la información recopilada por usted CloudTrail, puede determinar a AWS IoT Events qué dirección IP se realizó la solicitud, quién la realizó, cuándo se realizó y detalles adicionales.

Para obtener más información CloudTrail, consulte la [Guía AWS CloudTrail del usuario](#).

AWS IoT Events información en CloudTrail

CloudTrail está habilitada en su AWS cuenta al crear la cuenta. Cuando se produce una actividad en AWS IoT Events, esa actividad se registra en un CloudTrail evento junto con otros eventos de AWS servicio en el historial de eventos. Puedes ver, buscar y descargar los eventos recientes en tu AWS cuenta. Para obtener más información, consulte [Trabajar con el historial de CloudTrail eventos](#).

Para tener un registro continuo de los eventos de tu AWS cuenta, incluidos los eventos de tu cuenta AWS IoT Events, crea una ruta. Un rastro permite CloudTrail entregar archivos de registro a un bucket de Amazon S3. De forma predeterminada, cuando crea una ruta en la consola, la ruta se aplica a todas AWS las regiones. La ruta registra los eventos de todas las regiones de la AWS partición y envía los archivos de registro al bucket de Amazon S3 que especifique. Además, puede

configurar otros AWS servicios para analizar más a fondo los datos de eventos recopilados en los CloudTrail registros y actuar en función de ellos. Para obtener más información, consulte:

- [Crear una ruta para tu AWS cuenta](#)
- [CloudTrail servicios e integraciones compatibles](#)
- [Configuración de SNS las notificaciones de Amazon para CloudTrail](#)
- [Recibir archivos de CloudTrail registro de varias regiones](#) y [recibir archivos de CloudTrail registro de varias cuentas](#)

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario lo ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de IAM usuario o raíz.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Para obtener más información, consulte el [CloudTrail userIdentityelemento](#). AWS IoT Events las acciones se documentan en la [AWS IoT Events APIreferencia](#).

Descripción de las entradas de los archivos de AWS IoT Events registro

Un rastro es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que usted especifique. AWS CloudTrail Los archivos de registro contienen una o más entradas de registro. Un evento representa una solicitud única de cualquier fuente e incluye información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. CloudTrail Los archivos de registro no son un registro ordenado de las API llamadas públicas, por lo que no aparecen en ningún orden específico.

Cuando el CloudTrail registro está activado en tu AWS cuenta, la mayoría de las API llamadas a la AWS IoT Events acción se registran en los archivos de CloudTrail registro, donde se escriben junto con otros registros de AWS servicio. CloudTrail determina cuándo crear y escribir en un nuevo archivo en función del período de tiempo y del tamaño del archivo.

Cada entrada de registro contiene información sobre quién generó la solicitud. La información de identidad del usuario en la entrada de registro le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con credenciales de IAM usuario o root.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro AWS servicio.

Puede almacenar sus archivos de registro en su bucket de Amazon S3 durante todo el tiempo que desee, pero también puede definir reglas de ciclo de vida de Amazon S3 para archivar o eliminar archivos de registro automáticamente. De forma predeterminada, los archivos de registro se cifran con el cifrado del lado del servidor de Amazon S3 (SSE).

Para recibir una notificación cuando se entreguen los archivos de registro, puede CloudTrail configurar la publicación de SNS las notificaciones de Amazon cuando se entreguen nuevos archivos de registro. Para obtener más información, consulte [Configuración de SNS las notificaciones de Amazon para CloudTrail](#).

También puede AWS IoT Events agrupar archivos de registro de varias AWS regiones y AWS cuentas en un único bucket de Amazon S3.

Para obtener más información, consulte [Recepción de archivos de CloudTrail registro de varias regiones](#) y [Recepción de archivos de CloudTrail registro de varias cuentas](#).

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la DescribeDetector acción.

```
{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `CreateDetectorModel` acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la CreateInput acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `DeleteDetectorModel` acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la DeleteInput acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `DescribeDetectorModel` acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la DescribeInput acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la DescribeLoggingOptions acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la ListDetectorModels acción.

```

{
  "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": "CkZEZXRL1Y3Rvck1vZGVsM19saXN0ZGV0ZWNo3Jtb2R1bHN0ZXN0X2V1OWJkZTk1YT",
  "maxResults": 3
},
"responseElements": null,
"requestID": "6d70f262-da95-4bb5-94b4-c08369df75bb",
"eventID": "2d01a25c-d5c7-4233-99fe-ce1b8ec05516",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la `ListDetectorModelVersions` acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la ListDetectors acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la ListInputs acción.

```

{
  "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": "CkhjYW5hcnlfdGVzdF9pbnB1dF9saXN0ZGV0ZWNo0b3Jtb2RlbHN0ZXN0ZDU3OGZ",
    "maxResults": 3
  },
  "responseElements": null,
  "requestID": "dd6762a1-1f24-4e63-a986-5ea3938a03da",
  "eventID": "c500f6d8-e271-4366-8f20-da4413752469",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la PutLoggingOptions acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la UpdateDetectorModel acción.

```

{
  "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"
}

```

El siguiente ejemplo muestra una entrada de CloudTrail registro que demuestra la UpdateInput acción.

```

{
  "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"
}

```

Validación de conformidad para AWS IoT Events


Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte [Servicios de AWS Alcance por programa de cumplimiento](#)

[Servicios de AWS](#) de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de [AWS cumplimiento > Programas AWS](#).

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte [Descarga de informes en AWS Artifact](#).

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- [Guías de inicio rápido sobre seguridad y cumplimiento](#): estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en AWS la seguridad y el cumplimiento.
- [Diseñando una arquitectura basada en la HIPAA seguridad y el cumplimiento en Amazon Web Services](#): en este documento técnico se describe cómo pueden utilizar las empresas AWS para crear HIPAA aplicaciones aptas.

 Note

No todos son aptos. Servicios de AWS HIPAA Para obtener más información, consulta la [Referencia de servicios HIPAA aptos](#).

- [AWS Recursos](#) de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- [AWS Guías de cumplimiento para clientes](#): comprenda el modelo de responsabilidad compartida desde la perspectiva del cumplimiento. En las guías se resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y se orientan a los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- [Evaluación de los recursos con reglas](#) en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- [AWS Security Hub](#)— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del

sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la [Referencia de controles de Security Hub](#).

- [Amazon GuardDuty](#): Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, por ejemplo PCIDSS, cumpliendo con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- [AWS Audit Manager](#)— Esto le Servicio de AWS ayuda a auditar continuamente su AWS consumo para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Resiliencia en AWS IoT Events

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte [infraestructura AWS global](#).

Seguridad de la infraestructura en AWS IoT Events

Como servicio gestionado, AWS IoT Events está protegido por la seguridad de la red AWS global. Para obtener información sobre los servicios AWS de seguridad y cómo se AWS protege la infraestructura, consulte [Seguridad AWS en la nube](#). Para diseñar su AWS entorno utilizando las mejores prácticas de seguridad de la infraestructura, consulte [Protección de infraestructuras en un marco](#) de buena AWS arquitectura basado en el pilar de la seguridad.

Utiliza las API llamadas AWS publicadas para acceder a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Necesitamos TLS 1.2 y recomendamos TLS 1.3.

- Cifre suites con perfecto secreto (PFS), como (Ephemeral Diffie-Hellman) o DHE ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben firmarse con un identificador de clave de acceso y una clave de acceso secreta que esté asociada a un director. IAM También puede utilizar [AWS Security Token Service](#) (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

AWS cuotas de servicio para AWS IoT Events los recursos

La Referencia general de AWS Guía proporciona las cuotas predeterminadas AWS IoT Events para una AWS cuenta. A menos que se especifique lo contrario, cada cuota es por AWS región. Para obtener más información, consulte [Puntos de conexión AWS IoT Events y cuotas](#) y [Service Quotas de AWS](#) en la Guía de Referencia general de AWS .

Para solicitar un aumento de Service Quotas, envíe un caso de soporte a la consola del [Centro de soporte](#). Para obtener más información, consulte [Solicitud de un aumento de cuota](#) en la Guía del usuario de Service Quotas.

Note

- Todos los nombres de modelos de detector y entradas deben ser únicos en una misma cuenta.
- No puede cambiar los nombres de los modelos de detector ni de las entradas después de haberlos creado.

Etiquetar sus recursos AWS IoT Events

Para administrar y organizar sus modelos de detectores y entradas con mayor facilidad, puede asignar opcionalmente sus propios metadatos en forma de etiquetas a cada uno de estos recursos. En esta sección se describe qué son las etiquetas y cómo crearlas.

Conceptos básicos de etiquetas

Las etiquetas le permiten clasificar sus AWS IoT Events recursos de diferentes maneras, por ejemplo, por propósito, propietario o entorno. Esto es útil cuando se tienen muchos recursos del mismo tipo. Puede identificar rápidamente un recurso específico según las etiquetas que le haya asignado.

Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Por ejemplo, podría definir un conjunto de etiquetas para sus entradas que le ayuden a hacer un seguimiento de los dispositivos que envían estas entradas según su tipo. Le recomendamos que cree un conjunto de claves de etiqueta que cumpla sus necesidades para cada tipo de recurso. Mediante el uso de un conjunto coherente de claves de etiquetas, podrá administrar los recursos de más fácilmente.

Puede buscar y filtrar los recursos en función de las etiquetas que añade o aplique, utilizarlas para categorizar y realizar un seguimiento de sus costes, y también utilizar etiquetas para controlar el acceso a los recursos, tal y como se describe en la sección [Uso de etiquetas con IAM políticas](#) de la Guía para AWS IoT desarrolladores.

Para facilitar su uso, el editor de etiquetas AWS Management Console proporciona una forma centralizada y unificada de crear y administrar las etiquetas. Para obtener más información, consulte [Introducción al editor de etiquetas](#) en los AWS recursos de etiquetado y en la Guía del usuario del editor de etiquetas.

También puede trabajar con etiquetas utilizando las AWS CLI y las AWS IoT Events API. Puede asociar etiquetas con modelos de detectores y entradas al crearlas a través del campo "Tags" en los siguientes comandos:

- [CreateDetectorModel](#)
- [CreateInput](#)

Puede añadir, modificar o eliminar etiquetas para recursos existentes que admitan el uso de etiquetas con los siguientes comandos:

- [TagResource](#)
- [ListTagsForResource](#)
- [UntagResource](#)

Puede editar las claves y los valores de las etiquetas y también puede eliminar etiquetas de un recurso en cualquier momento. Puede establecer el valor de una etiqueta como una cadena vacía, pero no puede asignarle un valor nulo. Si añade una etiqueta con la misma clave que una etiqueta existente en ese recurso, el nuevo valor sobrescribirá al antiguo. Si elimina un recurso, también se eliminarán todas las etiquetas que este tenga asociadas.

Para obtener más información, consulte [Prácticas recomendadas para etiquetar recursos AWS](#)

Restricciones y limitaciones en las etiquetas

Se aplican las siguientes restricciones básicas a las etiquetas:

- Número máximo de etiquetas por recurso: 50
- Longitud máxima de la clave: 127 caracteres Unicode en -8 UTF
- Longitud máxima del valor: 255 caracteres Unicode en -8 UTF
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas.
- No utilices el "aws:" prefijo en los nombres o valores de las etiquetas porque está reservado para AWS su uso. Los nombres y valores de etiquetas que tienen este prefijo no se pueden editar ni eliminar. Las etiquetas que tengan este prefijo no cuentan para el límite de etiquetas por recurso.
- Si se utiliza su esquema de etiquetado en múltiples servicios y recursos, recuerde que otros servicios pueden tener otras restricciones sobre caracteres permitidos. Por lo general, los caracteres permitidos son: letras, espacios y números representables en UTF -8, y los siguientes caracteres especiales: + - =. _:/@.

Uso de etiquetas con políticas de IAM

Puede aplicar permisos de nivel de recursos basados en etiquetas en IAM las políticas que utilice para las acciones. AWS IoT Events API Esto le ofrece un mejor control sobre los recursos que un usuario puede crear, modificar o utilizar.

El `Condition` elemento (también denominado `Condition` bloque) se utiliza con las siguientes condiciones, claves de contexto y valores en una IAM política para controlar el acceso de los usuarios (permisos) en función de las etiquetas de un recurso:

- Utilice `aws:ResourceTag/<tag-key>: <tag-value>` para permitir o denegar acciones de los usuarios en recursos con etiquetas específicas.
- Se usa `aws:RequestTag/<tag-key>: <tag-value>` para exigir que se use (o no se use) una etiqueta específica al realizar una API solicitud para crear o modificar un recurso que permita etiquetas.
- Se utiliza `aws:TagKeys: [<tag-key>, ...]` para exigir que se utilice (o no se utilice) un conjunto específico de claves de etiquetas al realizar una API solicitud para crear o modificar un recurso que permita etiquetas.

Note

Las condiciones, las claves y los valores de contexto de una IAM política se aplican solo a aquellas AWS IoT Events acciones en las que el identificador de un recurso que se pueda etiquetar es un parámetro obligatorio.

[Control del acceso mediante etiquetas](#) en la Guía del usuario de AWS Identity and Access Management contiene información adicional sobre el uso de etiquetas. La sección de [referencia de IAM JSON políticas](#) de esa guía contiene una sintaxis detallada, descripciones y ejemplos de los elementos, las variables y la lógica de evaluación de JSON las políticas IAM.

La siguiente política de ejemplo aplica dos restricciones basadas en etiquetas. Un usuario restringido por esta política:

- No puede dar a un recurso la etiqueta "env=prod" (en el ejemplo, consulte la línea "aws:RequestTag/env" : "prod").
- No puede modificar ni obtener acceso a un recurso que tenga una etiqueta "env=prod" existente (en el ejemplo, consulte la línea "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": "*"    
  }  
]  
}
```

También puede especificar varios valores de etiqueta para una determinada clave de etiqueta encerrándolos en una lista, como se indica a continuación.

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

Note

Si permite o deniega a los usuarios acceso a recursos en función de etiquetas, debe considerar denegar explícitamente a los usuarios la posibilidad de agregar estas etiquetas o retirarlas de los mismos recursos. De lo contrario, es posible que un usuario eluda sus restricciones y obtenga acceso a un recurso modificando sus etiquetas.

Solución de problemas AWS IoT Events

Esta guía de solución de problemas proporciona soluciones a los problemas más comunes que pueden surgir al utilizarla AWS IoT Events. Examine los temas para identificar y resolver problemas relacionados con la detección de eventos, el acceso a los datos, los permisos, las integraciones de servicios, las configuraciones de los dispositivos y mucho más. Con consejos para la resolución de problemas relacionados con la AWS IoT Events consola APICLI, los errores, la latencia y las integraciones, esta guía tiene como objetivo resolver rápidamente sus problemas para que pueda crear aplicaciones fiables y escalables basadas en eventos.

Temas

- [AWS IoT Events Problemas y soluciones comunes](#)
- [Solución de problemas de un modelo de detector mediante la ejecución de análisis](#)

AWS IoT Events Problemas y soluciones comunes

Consulte la siguiente sección para solucionar los errores y encontrar posibles soluciones para resolverlos. AWS IoT Events

Errores

- [Errores de creación del modelo de detector](#)
- [Actualizaciones de un modelo de detector eliminado](#)
- [Fallo de activación de la acción \(al cumplirse una condición\)](#)
- [Fallo de activación de la acción \(al superar un umbral\)](#)
- [Uso incorrecto de los estados](#)
- [Mensaje de conexión](#)
- [InvalidRequestException mensaje](#)
- [action.setTimerErrores de Amazon CloudWatch Logs](#)
- [Errores de CloudWatch carga útil de Amazon](#)
- [Tipos de datos incompatibles](#)
- [No se pudo enviar el mensaje a AWS IoT Events](#)

Errores de creación del modelo de detector

Aparecen errores cuando intento crear un modelo de detector.

Solución

Al crear un modelo de detector, debe tener en cuenta las siguientes limitaciones.

- Solo se permite una acción en cada campo `action`.
- La `condition` es obligatoria para `transitionEvents`. Es opcional para los eventos `OnEnter`, `OnInput` y `OnExit`.
- Si el campo `condition` está vacío, el resultado evaluado de la expresión de condición equivale a `true`.
- El resultado evaluado de la expresión de condición debe ser un valor booleano. Si el resultado no es un valor booleano, equivale a `false` y no activa las `actions` ni la transición al `nextState` especificados en el evento.

Para obtener más información, consulte [Restricciones y limitaciones del modelo de detector](#).

Actualizaciones de un modelo de detector eliminado

Actualicé o eliminé un modelo de detector hace unos minutos, pero sigo recibiendo actualizaciones del estado del modelo de detector anterior mediante MQTT mensajes o SNS alertas.

Solución

Si actualiza, elimina o vuelve a crear un modelo de detector (consulte [UpdateDetectorModel](#)), pasará un tiempo hasta que se eliminen todas las instancias del detector y se utilice el nuevo modelo. Durante este tiempo, es posible que se sigan procesando entradas mediante las instancias de la versión anterior del modelo de detector. Podría seguir recibiendo alertas definidas por el modelo de detector anterior. Espere al menos siete minutos antes de volver a comprobar la actualización o de reportar un error.

Fallo de activación de la acción (al cumplirse una condición)

El detector no logra activar una acción o una transición a un nuevo estado al cumplirse la condición.

Solución

Compruebe que el resultado evaluado de la expresión condicional del detector sea un valor booleano. Si el resultado no es un valor booleano, equivale a `false` y no activa las `action` ni la transición al `nextState` especificados en el evento. Para obtener más información, consulte [Sintaxis de expresiones condicionales](#).

Fallo de activación de la acción (al superar un umbral)

El detector no activa una acción o una transición de evento cuando la variable en una expresión condicional alcanza un valor especificado.

Solución

Si actualiza `setVariable` para `onInput`, `onEnter`, o `onExit`, el nuevo valor no se utiliza al evaluar cualquier `condition` durante el ciclo de procesamiento actual. En cambio, se utiliza el valor original hasta completarse el ciclo actual. Puede cambiar este comportamiento si configura el parámetro `evaluationMethod` en la definición del modelo de detector. Si `evaluationMethod` se establece en `SERIAL`, las variables se actualizan y las condiciones de eventos se evalúan en el orden en que se definan los eventos. Si `evaluationMethod` se establece en `BATCH` (predeterminado), las variables se actualizan y los eventos se ejecutan solo después de que se evalúen todas las condiciones del evento.

Uso incorrecto de los estados

El detector entra en estados incorrectos cuando intento enviar mensajes a las entradas mediante `BatchPutMessage`.

Solución

Si suele enviar varios mensajes [BatchPutMessage](#) a las entradas, no se garantiza el orden en que se procesan los mensajes o las entradas. Para garantizar el orden, envíe los mensajes uno a la vez y espere cada vez que `BatchPutMessage` confirme el éxito.

Mensaje de conexión

`('Connection aborted.', error(54, 'Connection reset by peer'))` Aparece un error cuando intento llamar o invocar un API.

Solución

Compruebe que Open SSL utilice la versión TLS 1.1 o posterior para establecer la conexión. Este debería ser el método predeterminado en la mayoría de las distribuciones de Linux o en Windows versión 7 y posteriores. Es posible que los usuarios de macOS necesiten actualizar OpenSSL.

InvalidRequestException mensaje

InvalidRequestException Recibo cuando intento llamar `CreateDetectorModel` y `UpdateDetectorModel` APIs.

Solución

Para resolver el problema, compruebe lo siguiente. Para obtener más información, consulte [CreateDetectorModel](#) y [UpdateDetectorModel](#).

- Asegúrese de no utilizar al mismo tiempo `seconds` y `durationExpression` como parámetros de `SetTimerAction`.
- Asegúrese de que su expresión de cadena para `durationExpression` sea válida. La expresión de cadena puede contener números, variables (`$variable.<variable-name>`) o valores de entrada (`$input.<input-name>.<path-to-datum>`).

action.setTimer Errores de Amazon CloudWatch Logs

Puede configurar Amazon CloudWatch Logs para supervisar las instancias AWS IoT Events del modelo de detector. Los siguientes son errores comunes que se AWS IoT Events generan cuando se utiliza `action.setTimer`.

- Error: Su expresión de duración para el temporizador llamado `<timer-name>` no se ha podido evaluar como número.

Solución

Asegúrese de que su expresión de cadena para `durationExpression` pueda convertirse en un número. No se permiten otros tipos de datos, como los booleanos.

- Error: El resultado evaluado de su expresión de duración para el temporizador llamado `<timer-name>` es mayor que 31622440. Para garantizar la precisión, asegúrese de que su expresión de duración haga referencia a un valor comprendido entre 60 y 31622400.

Solución

Asegúrese de que la duración de su temporizador sea menor o igual que 31622400 segundos. El resultado evaluado de la duración se redondea hacia abajo al número entero más próximo.

- Error: El resultado evaluado de su expresión de duración para el temporizador llamado `<timer-name>` es menor que 60. Para garantizar la precisión, asegúrese de que su expresión de duración haga referencia a un valor comprendido entre 60 y 31622400.

Solución

Asegúrese de que la duración de su temporizador sea mayor o igual que 60 segundos. El resultado evaluado de la duración se redondea hacia abajo al número entero más próximo.

- Error: Su expresión de duración para el temporizador llamado `<timer-name>` no se ha podido evaluar. Compruebe los nombres de las variables, los nombres de las entradas y las rutas a los datos para asegurarse de que hace referencia a variables y entradas existentes.

Solución

Asegúrese de que su expresión de cadena haga referencia a variables y entradas existentes. La expresión de cadena puede contener números, variables (`$variable.variable-name`) y valores de entrada (`$input.input-name.path-to-datum`).

- Error: No se ha podido establecer el temporizador llamado `<timer-name>`. Compruebe su expresión de duración e inténtelo de nuevo.

Solución

Consulte la [SetTimerAction](#) acción para asegurarse de haber especificado los parámetros correctos y, a continuación, vuelva a configurar el temporizador.

Para obtener más información, consulte [Habilitar el CloudWatch registro de Amazon al desarrollar modelos de AWS IoT Events detectores](#).

Errores de CloudWatch carga útil de Amazon

Puede configurar Amazon CloudWatch Logs para supervisar las instancias AWS IoT Events del modelo de detector. Los siguientes son los errores y advertencias más comunes que se generan al configurar la carga útil de la acción.

- Error: No se ha podido evaluar su expresión para la acción. Asegúrese de que los nombres de las variables, los nombres de las entradas y las rutas a los datos hagan referencia a variables y valores de entrada existentes. Verifique también que el tamaño de la carga sea inferior a 1 KB, el tamaño máximo permitido para una carga.

Solución

Asegúrese de introducir correctamente los nombres de las variables, los nombres de las entradas y las rutas a los datos. También puede recibir este mensaje de error si la carga de acción es superior a 1 KB.

- Error: No se ha podido interpretar su expresión de contenido para la carga de *<action-type>*. Introduzca una expresión de contenido con la sintaxis correcta.

Solución

La expresión de contenido puede contener cadenas ('*string*'), variables (*\$variable.variable-name*), valores de entrada (*\$input.input-name.path-to-datum*), concatenaciones de cadenas y cadenas que contengan *{}*.

- Error: tu expresión de carga útil *{expression}* no es válida. El tipo de carga útil definido esJSON, por lo que debe especificar una expresión que AWS IoT Events dé como resultado una cadena.

Solución

Si el tipo de carga útil especificado esJSON, AWS IoT Events primero comprueba si el servicio puede evaluar la expresión en una cadena. El resultado evaluado no puede ser un booleano ni un número. Podría recibir este error si falla la validación.

- Advertencia: la acción se ejecutó, pero no pudimos evaluar tu expresión de contenido para que la carga útil de la acción fuera válida. JSON El tipo de carga útil definido es. JSON

Solución

Asegúrese de AWS IoT Events poder evaluar la expresión de su contenido para que la carga útil de la acción sea válidaJSON, si define el tipo de carga como. JSON AWS IoT Events ejecuta la acción aunque no AWS IoT Events pueda evaluar la expresión de contenido como válida. JSON

Para obtener más información, consulte [Habilitar el CloudWatch registro de Amazon al desarrollar modelos de AWS IoT Events detectores](#).

Tipos de datos incompatibles

Se han encontrado tipos de datos incompatibles [<inferred-types>] para <reference> en la siguiente expresión: <expression>

Solución

Podría recibir este error por uno de los siguientes motivos:

- Los resultados evaluados de sus referencias no son compatibles con otros operandos de sus expresiones.
- El tipo del argumento pasado a una función no es compatible.

Cuando utilice referencias en expresiones, compruebe lo siguiente:

- Cuando utilice una referencia como operando con uno o más operadores, asegúrese de que todos los tipos de datos a los que hace referencia sean compatibles.

Por ejemplo, en la siguiente expresión, el entero 2 es un operando de los operadores == y &&. Para asegurarse de que los operandos sean compatibles, `$variable.testVariable + 1` y `$variable.testVariable` deben hacer referencia a un número entero o decimal.

Además, el entero 1 es un operando del operador +. Por lo tanto, `$variable.testVariable` debe hacer referencia a un número entero o decimal.

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

- Cuando utilice una referencia como argumento pasado a una función, asegúrese de que la función admita los tipos de datos a los que hace referencia.

Por ejemplo, la siguiente función `timeout("time-name")` requiere una cadena con comillas dobles como argumento. Si utiliza una referencia para `timer-name` valor, debe hacer referencia a una cadena con comillas dobles.

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

Note

Para la `convert(type, expression)` función, si utiliza una referencia para el *type* valor, el resultado evaluado de la referencia debe ser `StringDecimal`, o `Boolean`.

Para obtener más información, consulte [Referencia para las entradas y variables de las expresiones](#).

No se pudo enviar el mensaje a AWS IoT Events

Mensaje: No se ha podido enviar el mensaje a lot Events

Solución

Podría experimentar este error por las siguientes razones:

- La carga del mensaje de entrada no contiene el `Input attribute Key`.
- No `Input attribute Key` está en la misma JSON ruta que la especificada en la definición de entrada.
- El mensaje de entrada no coincide con el esquema, tal como se define en la AWS IoT Events entrada.

Note

La ingesta de datos de otros servicios también experimentará fallos.

Example

Por ejemplo AWS IoT Core, en, la AWS IoT regla fallará con el siguiente mensaje `Verify the Input Attribute key`.

Para resolver este problema, asegúrese de que el esquema del mensaje de carga útil de entrada se ajuste a la definición AWS IoT Events de entrada y que la `Input attribute Key` ubicación coincida. Para obtener más información, consulte [Cree una entrada para los modelos](#) para aprender a definir AWS IoT Events las entradas.

Solución de problemas de un modelo de detector mediante la ejecución de análisis

AWS IoT Events puede analizar su modelo de detector y generar resultados de análisis sin enviar datos de entrada a su modelo de detector. AWS IoT Events realiza una serie de análisis descritos en esta sección para comprobar el modelo de su detector. Esta solución de problemas avanzada también resume la información de diagnóstico, incluyendo nivel de gravedad y ubicación, para que pueda encontrar y solucionar rápidamente los posibles problemas de su modelo de detector. Para obtener más información sobre los tipos de errores de diagnóstico y los mensajes de su modelo de detector, consulte [Análisis del modelo de detector e información de diagnóstico](#).

Puede utilizar la AWS IoT Events consola, [API](#), [AWS Command Line Interface \(AWS CLI\)](#) o [AWS SDK](#) para ver los mensajes de error de diagnóstico derivados del análisis del modelo de su detector.

Note

- Debe corregir todos los errores antes de poder publicar su modelo de detector.
- Le recomendamos que revise las advertencias y tome las medidas necesarias antes de utilizar su modelo de detector en entornos de producción. De lo contrario, el modelo de detector podría no funcionar como se espera.
- Puede tener hasta 10 análisis en el estado RUNNING al mismo tiempo.

Para obtener información sobre cómo analizar su modelo de detector, consulte [Análisis de un modelo de detector \(consola\)](#) o [Análisis de un modelo de detector \(AWS CLI\)](#).

Temas

- [Análisis del modelo de detector e información de diagnóstico](#)
- [Análisis de un modelo de detector \(consola\)](#)
- [Análisis de un modelo de detector \(AWS CLI\)](#)

Análisis del modelo de detector e información de diagnóstico


Los análisis del modelo de detector recaban la siguiente información de diagnóstico:

- **Nivel:** el nivel de gravedad del resultado del análisis. En función del nivel de gravedad, los resultados de los análisis se clasifican en tres categorías generales:
 - **Información (INFO):** un resultado de información le notifica sobre un campo significativo en su modelo de detector. Este tipo de resultado no suele requerir una acción inmediata.
 - **Advertencia (WARNING):** un resultado de advertencia llama especialmente la atención sobre los campos que podrían causar problemas en su modelo de detector. Le recomendamos que revise las advertencias y tome las medidas necesarias antes de utilizar su modelo de detector en entornos de producción. De lo contrario, el modelo de detector podría no funcionar como se espera.
 - **Error (ERROR):** un resultado de error le notifica que se ha encontrado un problema en su modelo de detector. AWS IoT Events realiza automáticamente este conjunto de análisis al intentar publicar el modelo de detector. Debe corregir todos los errores antes de poder publicar el modelo de detector.
- **Ubicación:** contiene información que puede utilizar para localizar el campo en su modelo de detector al que hace referencia el resultado del análisis. Una ubicación suele incluir el nombre del estado, el nombre del evento de transición, el nombre del evento y la expresión (por ejemplo, `in state TemperatureCheck in onEnter in event Init in action setVariable`).
- **Tipo:** el resultado del tipo de análisis. Los tipos de análisis se dividen en las siguientes categorías:
 - **supported-actions**— AWS IoT Events puede invocar acciones cuando se detecta un evento específico o un evento de transición. Puede definir acciones integradas para utilizar un temporizador o establecer una variable, o enviar datos, a otros servicios de AWS . Debe especificar las acciones que funcionen con otros AWS servicios en una AWS región en la que los AWS servicios estén disponibles.
 - **service-limits**— Las cuotas de servicio, también conocidas como límites, son la cantidad máxima o mínima de recursos u operaciones de servicio para su AWS cuenta. A menos que se indique lo contrario, cada cuota es específica de la región de . En función de sus necesidades empresariales, puede actualizar su modelo de detector para evitar encontrarse con límites o solicitar un aumento de cuota. Puede solicitar aumentos para algunas cuotas, pero otras no se pueden aumentar. Para obtener más información, consulte [Cuotas](#).
- **structure:** el modelo de detector debe tener todos los componentes requeridos, como los estados, y seguir una estructura que AWS IoT Events admita. Un modelo de detector debe tener al menos un estado y una condición que evalúe los datos de entrada a fin de detectar eventos significativos. Al detectar un evento, el modelo de detector realiza una transición al siguiente estado y puede invocar acciones. Estos eventos se conocen como eventos de transición. Un evento de transición debe indicar el siguiente estado al que se debe entrar.

- **expression-syntax:** AWS IoT Events proporciona varias formas de especificar valores al crear y actualizar modelos de detectores. Puede utilizar literales, operadores, funciones, referencias y plantillas de sustitución en las expresiones. Puede usar expresiones para especificar valores literales o AWS IoT Events puede evaluar las expresiones antes de especificar valores determinados. La expresión debe seguir la sintaxis requerida. Para obtener más información, consulte [Expresiones para filtrar, transformar y procesar datos de eventos](#).

Las expresiones del modelo de detector AWS IoT Events pueden hacer referencia a datos específicos o a un recurso.

- **data-type:** AWS IoT Events admite tipos de datos enteros, decimales, de cadena y booleanos. Si AWS IoT Events puede convertir automáticamente los datos de un tipo de datos en otro durante la evaluación de la expresión, estos tipos de datos son compatibles.

 Note

- Los únicos tipos de datos compatibles que AWS IoT Events admite son enteros y decimales.
- AWS IoT Events no puede evaluar expresiones aritméticas porque no AWS IoT Events puede convertir un entero en una cadena.

- **referenced-data:** debe definir los datos referenciados en su modelo de detector antes de poder utilizarlos. Por ejemplo, si desea enviar datos a una tabla de DynamoDB, debe definir una variable que haga referencia al nombre de la tabla antes de poder utilizar la variable en una expresión (`$variable.TableName`).
- **referenced-resource:** los recursos que el modelo de detector utilice deben estar disponibles. Debe definir los recursos antes de poder utilizarlos. Por ejemplo, su deseo es crear un modelo de detector para supervisar la temperatura de un invernadero. Debe definir una entrada (`$input.TemperatureInput`) para dirigir los datos de temperatura entrantes a su modelo de detector antes de poder utilizar la `$input.TemperatureInput.sensorData.temperature` para referenciar la temperatura.

Consulte la siguiente sección para solucionar errores y encontrar posibles soluciones a partir del análisis de su modelo de detector.

Solución de errores del modelo del detector

Los tipos de errores descritos anteriormente proporcionan información de diagnóstico sobre un modelo de detector y corresponden a mensajes que podría recuperar. Utilice estos mensajes y las soluciones sugeridas para resolver los errores de su modelo de detector.

Mensajes y soluciones

- [Location](#)
- [supported-actions](#)
- [service-limits](#)
- [structure](#)
- [expression-syntax](#)
- [data-type](#)
- [referenced-data](#)
- [referenced-resource](#)

Location

Un resultado de un análisis con información sobre Location, corresponde al siguiente mensaje de error:

- Mensaje: Contiene información adicional sobre el resultado del análisis. Este puede ser de información, advertencia o error.

Solución: Podría recibir este mensaje de error si ha especificado una acción que AWS IoT Events actualmente no admite. Para obtener una lista de las acciones admitidas, consulte [Acciones compatibles para recibir datos y activar acciones](#).

supported-actions

Un resultado de un análisis con información sobre supported-actions, corresponde a los siguientes mensajes de error:

- Mensaje: hay un tipo de acción no válido en la definición de la acción: *action-definition*.

Solución: Podría recibir este mensaje de error si ha especificado una acción que AWS IoT Events actualmente no admite. Para obtener una lista de las acciones admitidas, consulte [Acciones compatibles para recibir datos y activar acciones](#).

- Mensaje: DetectorModel la definición tiene un *aws-service* acción, pero el *aws-service* el servicio no es compatible en la región *region-name*.

Solución: es posible que reciba este mensaje de error si la acción que especificó es compatible con su región actual AWS IoT Events, pero no está disponible. Esto puede ocurrir al intentar enviar datos a un AWS servicio que no está disponible en la región. También debes elegir la misma región para ambos AWS IoT Events y los AWS servicios que estás utilizando.

service-limits

Un resultado de un análisis con información sobre `service-limits`, corresponde a los siguientes mensajes de error:

- Mensaje: La expresión de contenido permitida en la carga útil ha superado el límite *content-expression-size* bytes en caso de evento *event-name* en estado *state-name*.

Solución: Podría recibir este mensaje de error si la expresión de contenido de su carga de acción es superior a 1024 bytes. El tamaño de la expresión de contenido de una carga puede ser de hasta 1024 bytes.

- Mensaje: El número de estados permitidos en la definición del modelo de detector ha superado el límite *states-per-detector-model*.

Solución: Podría recibir este mensaje de error si su modelo de detector tiene más de 20 estados. Un modelo de detector puede tener hasta 20 estados.

- Mensaje: La duración del temporizador *timer-name* debe ser al menos *minimum-timer-duration* segundos de duración.

Solución: Podría recibir este mensaje de error si la duración de su temporizador es inferior a 60 segundos. Se recomienda fijar la duración de un temporizador entre 60 y 31622400 segundos. Si especifica una expresión para la duración de su temporizador, el resultado evaluado de la expresión de duración se redondea al número entero menor más cercano.

- Mensaje: El número de acciones permitidas por evento ha superado el límite *actions-per-event* en la definición del modelo de detector

Solución: Podría recibir este mensaje de error si el evento tiene más de 10 acciones. Puede tener hasta 10 acciones por cada evento en su modelo de detector.

- Mensaje: El número de eventos de transición permitidos por estado ha superado el límite *transition-events-per-state* en la definición del modelo de detector.

Solución: Podría recibir este mensaje de error si el estado tiene más de 20 eventos de transición. Puede tener hasta 20 eventos de transición por cada estado en su modelo de detector.

- Mensaje: El número de eventos permitidos por estado ha superado el límite *events-per-state* en la definición del modelo de detector

Solución: Podría recibir este mensaje de error si el estado tiene más de 20 eventos. Puede tener hasta 20 eventos por cada estado en su modelo de detector.

- Mensaje: El número máximo de modelos de detectores que pueden asociarse a una única entrada podría haber alcanzado el límite. Entrada *input-name* se utiliza en *detector-models-per-input* rutas de modelos de detectores.

Solución: Podría recibir este mensaje de advertencia si intentara dirigir una entrada a más de 10 modelos de detectores. Puede tener hasta 10 modelos de detectores diferentes asociados a un único modelo de detector.

structure

Un resultado de un análisis con información sobre `structure`, corresponde a los siguientes mensajes de error:

- Mensaje: Las acciones pueden tener un solo tipo definido, pero se encontró una acción con *number-of-types* tipos. Divídala en acciones separadas.

Solución: puede recibir este mensaje de error si ha especificado dos o más acciones en un solo campo mediante API operaciones para crear o actualizar el modelo de detector. Puede definir una matriz de objetos de `Action`. Asegúrese de definir cada acción como un objeto independiente.

- Mensaje: El `TransitionEvent` *transition-event-name* transiciones a un estado inexistente *state-name*.

Solución: es posible que reciba este mensaje de error si AWS IoT Events no encuentra el siguiente estado al que hacía referencia el evento de transición. Asegúrese de que el siguiente estado esté definido y de haber introducido el nombre de estado correcto.

- Mensaje: `DetectorModelDefinition` Tenían un nombre de estado compartido: estado encontrado *state-name* with *number-of-states* repeticiones.

Solución: Podría recibir este mensaje de error si utiliza el mismo nombre para uno o más estados. Asegúrese de asignar un nombre único a cada estado en su modelo de detector. El nombre del estado debe tener entre 1 y 128 caracteres. Caracteres válidos: a-z, A-Z, 0-9, _ (guion bajo) y - (guion).

- Mensaje: La definición `initialStateName` *initial-state-name* no correspondía a un Estado definido.

Solución: Podría recibir este mensaje de error si el nombre del estado inicial es incorrecto. El modelo de detector permanece en el estado inicial (inicio) hasta que llegue una entrada. En cuanto llega una entrada, el modelo de detector pasa de inmediato al siguiente estado. Asegúrese de que el nombre del estado inicial sea el nombre de un estado definido y de haber introducido el nombre correcto.

- Mensaje: La definición del modelo de detector debe utilizar al menos una entrada en una condición.

Solución: Podría recibir este error si no ha especificado una entrada en una condición. Debe utilizar al menos una entrada en al menos una condición. De lo contrario, AWS IoT Events no evalúa los datos entrantes.

- Mensaje: solo uno de segundos y se `durationExpression` puede configurar `SetTimer`.

Solución: Podría recibir este mensaje de error si utiliza tanto `seconds` como `durationExpression` para su temporizador. Asegúrese de utilizar `seconds` o `durationExpression` como parámetros de `SetTimerAction`. Para obtener más información, consulte [SetTimerAction](#) la AWS IoT Events API Referencia.

- Mensaje: Una acción de su modelo de detector es inaccesible. Compruebe la condición que inicia la acción.

Solución: Si una acción en su modelo de detector es inaccesible, la condición del evento da como resultado falso. Compruebe la condición del evento que contiene la acción a fin de asegurarse de que dé como resultado verdadero. Si la condición del evento da como resultado verdadero, la acción debería estar disponible.

- Mensaje: Se está leyendo un atributo de entrada, pero esto podría deberse al vencimiento de un temporizador.

Solución: El valor de un atributo de entrada se puede leer cuando ocurre una de las siguientes situaciones:

- Se ha recibido un nuevo valor de entrada.
- Cuando un temporizador del detector ha vencido.

Para asegurarse de que un atributo de entrada se evalúe solo al recibirse un nuevo valor para esa entrada, incluya una llamada a la función `triggerType("Message")` en su condición de la siguiente manera:

La condición original que se evalúa en el modelo de detector:

```
if ($input.HeartBeat.status == "OFFLINE")
```

pasaría a ser similar a la siguiente:

```
if ( triggerType("MESSAGE") && $input.HeartBeat.status == "OFFLINE")
```

donde la llamada a la función `triggerType("Message")` viene antes de la entrada inicial proporcionada en la condición. Mediante esta técnica, la función `triggerType("Message")` se evaluará como verdadero y satisfará la condición de recibir un nuevo valor de entrada. Para obtener más información sobre el uso de la función `triggerType`, consulte `triggerType` en la sección [Expresiones](#) de la Guía para desarrolladores de AWS IoT Events

- Mensaje: Un estado de su modelo de detector es inaccesible. Compruebe la condición que provocará una transición al estado deseado.

Solución: Si un estado en su modelo de detector es inaccesible, una condición que cause una transición entrante a ese estado se evaluará como falsa. Compruebe que las condiciones de las transiciones entrantes a ese estado inaccesible en su modelo de detector den como resultado verdadero, para que el estado deseado se vuelva accesible.

- Mensaje: Un temporizador que vence puede provocar el envío de una cantidad inesperada de mensajes.

Solución: Para evitar que su modelo de detector entre en un estado infinito de envío de una cantidad inesperada de mensajes porque un temporizador ha vencido, considere el uso de una llamada a la función `triggerType("Message")` en las condiciones de su modelo de detector de la siguiente manera:

La condición original que se evalúa en el modelo de detector:

```
if (timeout("awake"))
```

se transformaría en una condición similar a la siguiente:

```
if (triggerType("MESSAGE") && timeout("awake"))
```

donde la llamada a la función `triggerType("Message")` viene antes de la entrada inicial proporcionada en la condición.

Este cambio evita que se inicien acciones de temporizador en su detector, lo que impide un bucle infinito de envío de mensajes. Para obtener más información sobre cómo utilizar las acciones de temporizador en su detector, consulte la página [Uso de las acciones integradas](#) de la Guía para desarrolladores de AWS IoT Events

expression-syntax

Un resultado de un análisis con información sobre `expression-syntax`, corresponde a los siguientes mensajes de error:

- Mensaje: Su expresión de carga útil `{expression}` no es válida. El tipo de carga útil definido es JSON, por lo que debe especificar una expresión que AWS IoT Events dé como resultado una cadena.

Solución: si el tipo de carga útil especificado es JSON, AWS IoT Events primero comprueba si el servicio puede evaluar la expresión en una cadena. El resultado evaluado no puede ser un booleano ni un número. Si la validación no tiene éxito, podría recibir este error.

- Mensaje: `SetVariableAction.value` debe ser una expresión. No se pudo analizar el valor `'variable-value'`

Solución: Puede utilizar `SetVariableAction` para definir una variable con un `name` y un `value`. El `value` puede ser una cadena, un número o un valor booleano. También puede especificar una expresión para el `value`. Para obtener más información [SetVariableAction](#), consulte la AWS IoT Events API Referencia.

- Mensaje: No hemos podido analizar la expresión de los atributos (`attribute-name`) para la acción DynamoDB. Introduzca la expresión con la sintaxis correcta.

Solución: Debe utilizar expresiones para todos los parámetros en las plantillas de sustitución de `DynamoDBAction`. Para obtener más información, consulte [DynamoDBAction](#) en la AWS IoT Events API Referencia.

- Mensaje: No hemos podido analizar tu expresión `tableName` para la acción `DynamoDBv2`. Introduzca la expresión con la sintaxis correcta.

Solución: El `tableName` en `DynamoDBv2Action` debe ser una cadena. Debe utilizar una expresión para el `tableName`. Las expresiones aceptan literales, operadores, funciones, referencias y plantillas de sustitución. Para obtener más información, consulte [DynamoDBv2Action](#) en la AWS IoT Events API Referencia.

- Mensaje: No hemos podido evaluar su expresión como válida JSON. La acción `DynamoDBv2` solo admite el tipo de carga útil JSON.

Solución: el tipo de carga útil `DynamoDBv2` debe ser JSON. Asegúrese de que AWS IoT Events pueda evaluar la expresión de su contenido para que la carga útil sea válida. Para obtener más información, consulte [DynamoDBv2Action](#), en la AWS IoT Events API Referencia.

- Mensaje: No hemos podido analizar tu expresión de contenido para ver la carga útil de `action-type`. Introduzca una expresión de contenido con la sintaxis correcta.

Solución: la expresión de contenido puede contener cadenas (`'string'`), variables (`$variable.variable-name`), valores de entrada (`$entrada.input-name.path-to-datum`), concatenaciones de cadenas y cadenas que contienen `${}`.

- Mensaje: Las cargas personalizadas no deben estar vacías.

Solución: es posible que recibas este mensaje de error si has elegido una carga útil personalizada para tu acción y no has introducido ninguna expresión de contenido en la consola. AWS IoT Events Si elige Carga personalizada, debe introducir una expresión de contenido en Carga personalizada. Para obtener más información, consulte [Carga útil](#) en la AWS IoT Events API referencia.

- Mensaje: No se pudo analizar la expresión de duración `'duration-expression'` para el temporizador `'timer-name'`.

Solución: El resultado evaluado de su expresión de duración para el temporizador debe ser un valor comprendido entre 60 y 31622400. El resultado evaluado de la duración se redondea hacia abajo al número entero más próximo.

- Mensaje: No se pudo analizar la expresión `'expression'` para `action-name`

Solución: Podría recibir este mensaje si la expresión para la acción especificada tiene una sintaxis incorrecta. Asegúrese de introducir una expresión con la sintaxis correcta. Para obtener más información, consulte [Sintaxis para filtrar los datos del dispositivo y definir acciones](#).

- Mensaje: Tu *fieldName* porque IotSiteWiseAction no se pudo analizar. Debe utilizar la sintaxis correcta en su expresión.

Solución: es posible que reciba este error si no AWS IoT Events puede analizar su *fieldName* para IotSiteWiseAction. Asegúrese de que *fieldName* usa una expresión que AWS IoT Events pueda analizarse. Para obtener más información, consulte [lotSiteWiseAction](#) la AWS IoT Events APIReferencia.

data-type

Un resultado de un análisis con información sobre data-type, corresponde a los siguientes mensajes de error:

- Mensaje: expresión de duración *duration-expression* para temporizador *timer-name* no es válido, debe devolver un número.

Solución: es posible que reciba este mensaje de error AWS IoT Events si no puede evaluar la expresión de duración del temporizador con un número. Asegúrese de que su *durationExpression* se pueda convertir en un número. Otros tipos de datos, como los booleanos, no son compatibles.

- Mensaje: Expresión *condition-expression* no es una expresión de condición válida.

Solución: es posible que reciba este mensaje de error si AWS IoT Events no puede evaluarla *condition-expression* con un valor booleano. El valor booleano debe ser TRUE o FALSE. Asegúrese de que su expresión de condición se pueda convertir en un valor booleano. Si el resultado no es un valor booleano, es equivalente a FALSE y no invocará las acciones o la transición al *nextState* especificado en el evento.

- Mensaje: Tipos de datos incompatibles [*inferred-types*] encontrado para *reference* en la siguiente expresión: *expression*

Solución:: Todas las expresiones para el mismo atributo de entrada o variable en el modelo de detector deben hacer referencia al mismo tipo de datos.

Utilice la siguiente información para resolver el problema:

- Cuando utilice una referencia como operando con uno o más operadores, asegúrese de que todos los tipos de datos a los que hace referencia sean compatibles.

Por ejemplo, en la siguiente expresión, el entero 2 es un operando de los operadores == y &&. Para asegurarse de que los operandos sean compatibles, `$variable.testVariable + 1` y `$variable.testVariable` deben hacer referencia a un número entero o decimal.

Además, el entero 1 es un operando del operador +. Por lo tanto, `$variable.testVariable` debe hacer referencia a un número entero o decimal.

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

- Cuando utilice una referencia como argumento pasado a una función, asegúrese de que la función admita los tipos de datos a los que hace referencia.

Por ejemplo, la siguiente función `timeout("time-name")` requiere una cadena con comillas dobles como argumento. Si utiliza una referencia para el `timer-name` valor, debe hacer referencia a una cadena con comillas dobles.

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

Note

Para la `convert(type, expression)` función, si utiliza una referencia para el `type` valor, el resultado evaluado de la referencia debe ser `StringDecimal`, o `Boolean`.

Para obtener más información, consulte [Referencia para las entradas y variables de las expresiones](#).

- Mensaje: Tipos de datos incompatibles [*inferred-types*] utilizado con *reference*. Esto puede provocar un error de tiempo de ejecución.

Solución: Podría recibir este mensaje de advertencia si dos expresiones para el mismo atributo de entrada o variable hacen referencia a dos tipos de datos. Asegúrese de que sus expresiones para el mismo atributo de entrada o variable hagan referencia al mismo tipo de datos en el modelo de detector.

- Mensaje: Los tipos de datos [*inferred-types*] que ha introducido para el operador [*operator*] no son compatibles con la siguiente expresión: '*expression*'

Solución: Podría recibir este mensaje de error si su expresión combina tipos de datos que no son compatibles con un operador especificado. Por ejemplo, en la siguiente expresión, el operador `+` es compatible con los tipos de datos entero, decimal y cadena, pero no con los operandos de tipo de datos booleano.

```
true + false
```

Debe asegurarse de que los tipos de datos que utilice con un operador sean compatibles.

- Mensaje: Los tipos de datos [*inferred-types*] encontrado para *input-attribute* no son compatibles y pueden provocar un error de tiempo de ejecución.

Solución: Podría recibir este mensaje de error si dos expresiones para el mismo atributo de entrada hacen referencia a dos tipos de datos ya sea el `OnEnterLifecycle` de un estado, o tanto el `OnInputLifecycle` como el `OnExitLifecycle` de un estado. Asegúrese de que sus expresiones en `OnEnterLifecycle` (o tanto en `OnInputLifecycle` como en `OnExitLifecycle`) hagan referencia al mismo tipo de datos para cada estado de su modelo de detector.

- Mensaje: La expresión de carga útil [*expression*] no es válida. Especifique una expresión que se traduzca en una cadena en tiempo de ejecución, ya que el tipo de carga útil es el JSON formato.

Solución: es posible que reciba este error si el tipo de carga útil especificado lo es JSON, pero no AWS IoT Events puede evaluar su expresión en una cadena. Asegúrese de que el resultado evaluado sea una cadena, no un booleano ni un número.

- Mensaje: tu expresión interpolada {*interpolated-expression*} debe dar como resultado un entero o un valor booleano en tiempo de ejecución. De lo contrario, la expresión de carga útil {*payload-expression*} no se podrá analizar en tiempo de ejecución como válida. JSON

Solución: es posible que reciba este mensaje de error si no AWS IoT Events puede evaluar la expresión interpolada en un valor entero o booleano. Asegúrese de que su expresión interpolada puede convertirse a un valor entero o booleano, dado que otros tipos de datos, como cadena, no son compatibles.

- Mensaje: el tipo de expresión del campo `IotSitewiseAction` *expression* se define como tipo *defined-type* y se deduce como tipo *inferred-type*. El tipo definido y el tipo inferido deben ser iguales.

Solución: Podría recibir este mensaje de error si su expresión en el `propertyValue` de `IotSitetwiseAction` tiene un tipo de datos definido diferente al tipo de datos inferido por AWS IoT Events. Asegúrese de utilizar el mismo tipo de datos para todas las instancias de esta expresión en su modelo de detector.

- Mensaje: Los tipos de datos [*inferred-types*] utilizados para `setTimer` la acción no se evalúan `Integer` para la siguiente expresión: *expression*

Solución: Podría recibir este mensaje de error si el tipo de datos inferido para su expresión de duración no es un valor entero o decimal. Asegúrese de que su `durationExpression` se pueda convertir a un número. Otros tipos de datos, como los booleanos y de cadena, no son compatibles.

- Mensaje: Los tipos de datos [*inferred-types*] utilizados con los operandos del operador de comparación [*operator*] no son compatibles en la siguiente expresión: *expression*

Solución: los tipos de datos inferidos para los operandos del *operator* en la expresión condicional (*expression*) de su modelo de detector no coinciden. Los operandos se deben utilizar con tipos de datos coincidentes en todas las demás partes de su modelo de detector.

Tip

Puede utilizar `convert` para cambiar el tipo de datos de una expresión en su modelo de detector. Para obtener más información, consulte [Funciones para usar en expresiones](#).

referenced-data

Un resultado de un análisis con información sobre `referenced-data`, corresponde a los siguientes mensajes de error:

- Mensaje: Se detectó un temporizador roto: temporizador *timer-name* se usa en una expresión pero nunca se establece.

Solución: Podría recibir este mensaje de error si utiliza un temporizador que no se haya configurado. Debe configurar cualquier temporizador antes de utilizarlo en una expresión. Asegúrese también de introducir el nombre correcto del temporizador.

- Mensaje: Se detectó una variable rota: variable *variable-name* se usa en una expresión pero nunca se establece.

Solución: Podría recibir este mensaje de error si utiliza una variable que no se haya configurado. Debe configurar cualquier variable antes de utilizarla en una expresión. Asegúrese también de introducir el nombre correcto de la variable.

- Mensaje: Se ha detectado una variable inservible: se utiliza una variable en una expresión antes de haberle establecido un valor.

Solución: A cada variable se le debe asignar un valor antes de que se pueda evaluar en una expresión. Establezca el valor de la variable antes de cada uso para poder así recuperarlo. Asegúrese también de introducir el nombre correcto de la variable.

referenced-resource

Un resultado de un análisis con información sobre `referenced-resource`, corresponde a los siguientes mensajes de error:

- Mensaje: La definición del modelo de detector contiene una referencia a una entrada que no existe.

Solución: Podría recibir este mensaje de error si utiliza expresiones para hacer referencia a una entrada que no existe. Asegúrese de que su expresión haga referencia a una entrada existente e introduzca el nombre de entrada correcto. Si no tiene una entrada, primero cree una.

- Mensaje: la definición del modelo de detector no contiene nada válido InputName: *input-name*

Solución: Podría recibir este mensaje de error si su modelo de detector contiene un nombre de entrada no válido. Asegúrese de introducir el nombre de la entrada correcto. El nombre de la entrada debe tener entre 1 y 128 caracteres. Caracteres válidos: a-z, A-Z, 0-9, _ (guion bajo) y - (guion).

Análisis de un modelo de detector (consola)

En los siguientes pasos, se utiliza la AWS IoT Events consola para analizar un modelo de detector.

1. Inicie sesión en la [consola de AWS IoT Events](#).
2. En el panel de navegación, seleccione Modelos de detectores.
3. En Modelos de detectores, seleccione el modelo de detector deseado.
4. En la página del modelo de detector, seleccione Editar.

5. En la esquina superior derecha, seleccione Ejecutar análisis.

The screenshot shows the AWS IoT Events console interface for a detector model named 'temperatureDetectorModel'. The 'Run analysis' button in the top right corner is highlighted with a red rectangular box. The main canvas displays a state machine diagram with a 'Start' state and a transition to a 'TemperatureCheck' state. The 'TemperatureCheck' state is labeled with '2 events'. On the right side, there is a 'Detector model palette' with instructions on how to build a detector model and a search bar for states.

El siguiente es un ejemplo del resultado de un análisis en la AWS IoT Events consola.

This screenshot shows the same detector model interface as above, but with the 'Run analysis' button replaced by 'Rerun analysis'. Below the main canvas, a 'Detector model analysis' panel is visible, highlighted with a red box. This panel shows the analysis results: '(1) All', '(0) Error', '(0) Warning', and '(1) Information'. Below these counts, there is an information icon and a message: 'Info: data-type Message: Inferred data types [Integer] for \$variable.temperatureChecked'.

Note

Una vez que AWS IoT Events comience a analizar el modelo de detector, dispondrá de hasta 24 horas para recuperar los resultados del análisis.

Análisis de un modelo de detector (AWS CLI)

Los siguientes pasos se utilizan AWS CLI para analizar un modelo de detector.

1. Ejecute el siguiente comando para iniciar un análisis.

```
aws iotevents start-detector-model-analysis --cli-input-json file://file-name.json
```

Note

Reemplazar *file-name* con el nombre del archivo que contiene la definición del modelo de detector.

Example Definición del modelo de detector

```
{
  "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"
}
}

```

Si utiliza el AWS CLI para analizar un modelo de detector existente, elija una de las siguientes opciones para recuperar la definición del modelo de detector:

- Si desea utilizar la AWS IoT Events consola, haga lo siguiente:
 1. En el panel de navegación, seleccione Modelos de detectores.
 2. En Modelos de detectores, seleccione el modelo de detector deseado.
 3. Seleccione Exportar modelo de detector en Acción para descargar el modelo de detector. El modelo de detector se guarda enJSON.
 4. Abra el JSON archivo del modelo del detector.
 5. Solo necesita el objeto `detectorModelDefinition`. Elimine lo siguiente:
 - La primera llave (`{`) en la parte superior de la página
 - La línea `detectorModel`
 - El objeto `detectorModelConfiguration`.
 - La última llave (`}`) en la parte inferior de la página
 6. Guarde el archivo.
- Si desea utilizar el AWS CLI, haga lo siguiente:

1. Ejecute el comando siguiente en un terminal.

```
aws iotevents describe-detector-model --detector-model-name detector-model-name
```

2. Reemplazar *detector-model-name* con el nombre del modelo de su detector.
3. Copie el objeto `detectorModelDefinition` en un editor de texto.
4. Añada llaves (`{}`) fuera de `detectorModelDefinition`.
5. Guarde el archivo enJSON.

Example Ejemplo de respuesta

```
{  
  "analysisId": "c1133390-14e3-4204-9a66-31efd92a4fed"  
}
```

2. Copie el ID de análisis de la salida.
3. Ejecute el siguiente comando para recuperar el estado del análisis.

```
aws iotevents describe-detector-model-analysis --analysis-id "analysis-id"
```

Note

Reemplazar *analysis-id* con el identificador de análisis que ha copiado.


Example Ejemplo de respuesta

```
{  
  "status": "COMPLETE"  
}
```

El estado puede ser uno de los siguientes valores:

- **RUNNING**— AWS IoT Events está analizando su modelo de detector. El proceso puede tardar hasta un minuto en completarse.
- **COMPLETE**— AWS IoT Events ha terminado de analizar su modelo de detector.

- FAILED— no AWS IoT Events pudo analizar su modelo de detector. Inténtelo de nuevo más tarde.
4. Ejecute el siguiente comando para recuperar uno o más resultados de análisis del modelo de detector.


 Note

Reemplazar *analysis-id* con el identificador de análisis que ha copiado.

```
aws iotevents get-detector-model-analysis-results --analysis-id "analysis-id"
```

Example Ejemplo de respuesta

```
{
  "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**

Una vez que AWS IoT Events comience a analizar el modelo de detector, dispondrá de hasta 24 horas para recuperar los resultados del análisis.

AWS IoT Events comandos

En este capítulo se explican AWS IoT Events en detalle todas las API operaciones, incluidos ejemplos de solicitudes, respuestas y errores de los protocolos de servicios web compatibles.

AWS IoT Events acciones

Puede usar AWS IoT Events API comandos para crear, leer, actualizar y eliminar entradas y modelos de detectores, y para enumerar sus versiones. Para obtener más información, consulte las [acciones](#) y [los tipos de datos](#) que se admiten AWS IoT Events en la AWS IoT Events API Referencia.

Las [AWS IoT Events secciones](#) de la Referencia de AWS CLI comandos incluyen los AWS CLI comandos que puede utilizar para administrar y manipular AWS IoT Events.

AWS IoT Events datos

Puede utilizar los API comandos de AWS IoT Events datos para enviar entradas a los detectores, hacer una lista de los detectores y ver o actualizar el estado de un detector. Para obtener más información, consulte las [acciones](#) y [los tipos de datos](#) que admite AWS IoT Events Data in the AWS IoT Events API Reference.

[Las secciones de AWS IoT Events datos](#) de la Referencia de AWS CLI comandos incluyen los AWS CLI comandos que puede utilizar para procesar AWS IoT Events datos.

Historial de documentos para AWS IoT Events

En la siguiente tabla se describen los cambios importantes efectuados en la Guía para desarrolladores de AWS IoT Events después del 17 de septiembre de 2020. Para obtener más información sobre las actualizaciones de esta documentación, puedes suscribirte a un RSS feed.

Cambio	Descripción	Fecha
Lanzamiento regional	AWS IoT Events ya está disponible en la región de Asia Pacífico (Bombay).	30 de septiembre de 2021
Lanzamiento regional	AWS IoT Events ya está disponible en la región AWS GovCloud (EE. UU.-Oeste).	22 de septiembre de 2021
Solución de problemas de un modelo de detector mediante la ejecución de análisis	AWS IoT Events ahora puede analizar su modelo de detector y generar resultados de análisis que puede utilizar para solucionar los problemas de su modelo de detector.	23 de febrero de 2021
Lanzamiento regional	Lanzado AWS IoT Events en China (Pekín).	30 de septiembre de 2020
Uso de expresiones	Se han añadido ejemplos para mostrarle cómo se escriben expresiones.	22 de septiembre de 2020
Monitoreo con alarmas	Las alarmas le ayudan a monitorear sus datos a fin de detectar cambios. Puede crear alarmas que le envíen notificaciones al superarse un umbral.	1 de junio de 2020

Actualizaciones anteriores

En la siguiente tabla se describen los cambios importantes efectuados en la Guía para desarrolladores de AWS IoT Events antes del 18 de septiembre de 2020.

Cambio	Descripción	Fecha
Se agregó la validación de tipos a la referencia de expresiones	Se agregó información de validación de tipos a la referencia de expresiones.	3 de agosto de 2020
Se agregó una advertencia de región para otros servicios	Se agregó una advertencia sobre la selección de la misma región AWS IoT Events y otros AWS servicios.	7 de mayo de 2020
Adiciones, actualizaciones	<ul style="list-style-type: none"> • Función de personalización de la carga útil • Nuevas acciones de eventos: Amazon DynamoDB y AWS IoT SiteWise 	27 de abril de 2020
Se agregaron funciones integradas para las expresiones condicionales del modelo de detector	Se agregaron funciones integradas para las expresiones condicionales del modelo de detector.	10 de septiembre de 2019
Se agregaron ejemplos de modelos de detectores	Se agregaron ejemplos para el modelo de detector.	5 de agosto de 2019
Se agregaron nuevas acciones de evento	<p>Se han añadido nuevas acciones de evento para:</p> <ul style="list-style-type: none"> • Lambda • Amazon SQS • Kinesis Data Firehose 	19 de julio de 2019

Cambio	Descripción	Fecha
	<ul style="list-style-type: none"> • AWS IoT Events entrada 	
Adiciones, correcciones	<ul style="list-style-type: none"> • Descripción actualizada de la <code>timeout()</code> función. • Se ha añadido la práctica recomendada en relación con la inactividad de las cuentas. 	11 de junio de 2019
Política de permisos actualizada y opciones de depuración de la consola	<ul style="list-style-type: none"> • Se actualizó la política de permisos de la consola. • Imagen actualizada de la página de opciones de depuración de la consola. 	5 de junio de 2019
Actualizaciones	AWS IoT Events servicio abierto a disponibilidad general.	30 de mayo de 2019
Adiciones, actualizaciones	<ul style="list-style-type: none"> • Información de seguridad actualizada. • Se agregó un ejemplo de modelo de detector anotado. 	22 de mayo de 2019
Se agregaron ejemplos y permisos necesarios	Se agregaron ejemplos de SNS carga útil de Amazon; se agregaron los permisos necesarios para <code>CreateDetectorModel</code> .	17 de mayo de 2019
Se agregó información de seguridad adicional	Se agregó información a la sección de seguridad.	9 de mayo de 2019

Cambio	Descripción	Fecha
Versión inicial de vista previa limitada	Versión inicial de la documentación de vista previa limitada.	28 de marzo de 2019

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la versión original de inglés, prevalecerá la versión en inglés.