



Guida per gli sviluppatori

AWS IoT Events



AWS IoT Events: Guida per gli sviluppatori

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

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

Che cos'è AWS IoT Events?	1
Vantaggi e caratteristiche	1
Casi d'uso	2
Configurazione	4
Configurazione delle autorizzazioni per AWS IoT Events	4
Autorizzazioni di azione	5
Protezione dei dati di input	7
Politica del ruolo CloudWatch di registrazione di Amazon	8
Policy sui ruoli di messaggistica di Amazon SNS	10
Nozioni di base	11
Prerequisiti	13
Crea un input	14
Creare un input nel riquadro di navigazione	15
Creare un input nel modello del rilevatore	15
Creare un modello di rilevatore	15
Invia input per testare il modello del rilevatore	23
Best practice	27
Abilita la CloudWatch registrazione di Amazon durante lo sviluppo di modelli di AWS IoT Events rilevatori	27
Pubblica regolarmente per salvare il modello del rilevatore quando lavori nella console AWS IoT Events	28
Archivia AWS IoT Events i tuoi dati per evitare possibili perdite di dati dovute a un lungo periodo di inattività	28
Tutorial	29
Utilizzo AWS IoT Events per monitorare i dispositivi IoT	29
Come fai a sapere di quali stati hai bisogno in un modello di rilevatore?	31
Come fai a sapere se hai bisogno di una o più istanze di un rilevatore?	32
Esempio semplice step-by-step	33
Crea un input per acquisire i dati del dispositivo	35
Crea un modello di rilevatore per rappresentare gli stati del dispositivo	36
Inviare messaggi come input a un rilevatore	40
Restrizioni e limitazioni del modello di rilevatore	43
Un esempio commentato: controllo della temperatura HVAC	47
Contesto	47

Azioni supportate	84
Utilizzo di azioni integrate	85
Imposta l'azione del timer	85
Reimposta l'azione del timer	86
Cancella l'azione del timer	86
Imposta un'azione variabile	87
Lavorare con altri AWS servizi	87
AWS IoT Core	88
AWS IoT Events	89
AWS IoT SiteWise	90
Amazon DynamoDB	93
Amazon DynamoDB (versione 2)	95
Amazon Data Firehose	96
AWS Lambda	97
Amazon Simple Notification Service	98
Amazon Simple Queue Service	100
Espressioni	102
Sintassi	102
Valori letterali	102
Operatori	102
Funzioni	104
Riferimenti	108
Modelli di sostituzione	111
Utilizzo	112
Scrittura di AWS IoT Events espressioni	112
Esempi di modelli di rivelatori	114
Controllo della temperatura HVAC	114
Storia di fondo	114
Definizioni di input	115
Definizione del modello di rilevatore	117
BatchPutMessageesempi	135
BatchUpdateDetector esempio	140
AWS IoT Coreesempi di motori di regole	142
Gru	145
Storia di fondo	145
Comandi	146

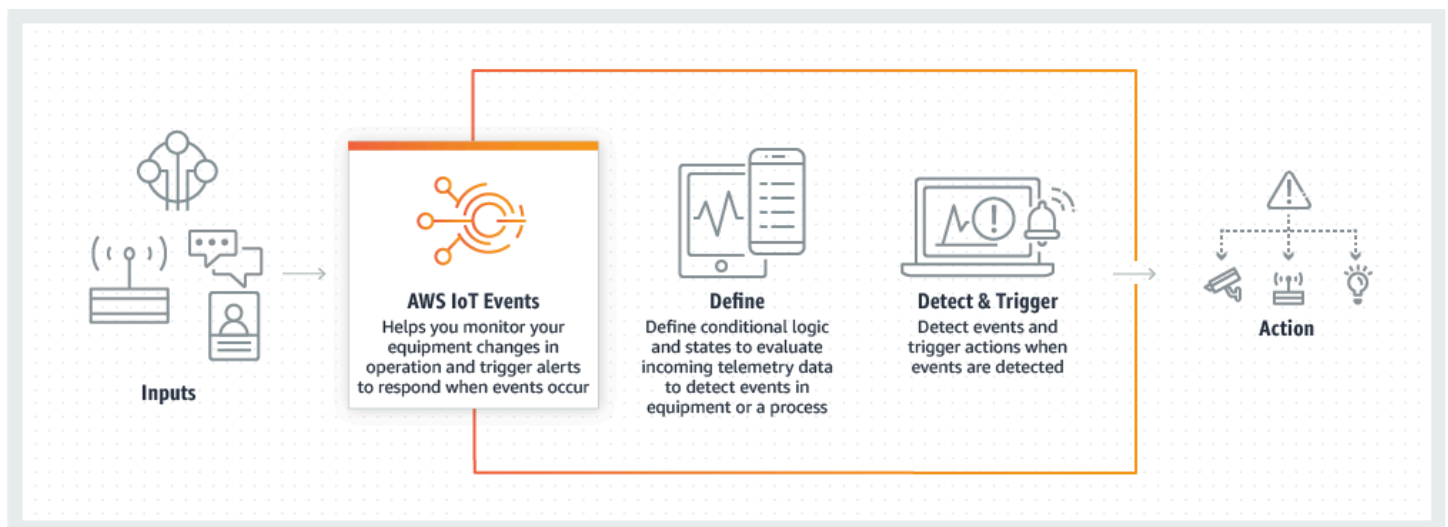
Modelli di rilevatore	147
Input	154
Messaggi	155
Rilevamento di eventi con sensori e applicazioni	156
Dispositivo HeartBeat	158
Allarme ISA	160
Allarme semplice	170
Monitoraggio con allarmi	175
Utilizzo di AWS IoT SiteWise	175
Conferma il flusso	175
Creazione di un modello di allarme	176
Requisiti	177
Creazione di un modello di allarme (console)	177
Risposta agli allarmi	180
Risposta agli allarmi (console)	180
Risposta agli allarmi (API)	181
Gestione delle notifiche di allarme	181
Creazione di una funzione Lambda	182
Utilizzo della funzione Lambda fornita da AWS IoT Events	190
Gestione dei destinatari	191
Sicurezza	193
Gestione dell'identità e degli accessi	193
Destinatari	194
Autenticazione con identità	195
Gestione dell'accesso con policy	198
Ulteriori informazioni	200
Funzionamento di AWS IoT Events con IAM	200
Esempi di policy basate su identità	204
Prevenzione del problema "confused deputy" tra servizi	210
Risoluzione dei problemi	214
Monitoraggio	216
Strumenti di monitoraggio	217
Monitoraggio con Amazon CloudWatch	218
Registrazione delle chiamate API AWS IoT Events con AWS CloudTrail	220
Convalida della conformità	237
Resilienza	238

Sicurezza dell'infrastruttura	239
Quote	240
Assegnazione di tag	241
Nozioni di base sui tag	241
Restrizioni e limitazioni di tag	242
Utilizzo dei tag con policy IAM	242
Risoluzione dei problemi	246
AWS IoT EventsProblemi e soluzioni comuni	246
Errori di creazione del modello di rilevatore	246
Aggiornamenti da un modello di rilevatore eliminato	247
Errore nell'attivazione dell'azione (quando viene soddisfatta una condizione)	247
Errore nell'attivazione dell'azione (quando si supera una soglia)	248
Utilizzo errato dello stato	248
Messaggio di connessione	248
InvalidRequestException messaggio	249
Errori di Amazon CloudWatch Logs <code>action.setTimer</code>	249
Errori del CloudWatch payload di Amazon	250
Tipi di dati incompatibili	251
Impossibile inviare il messaggio a AWS IoT Events	253
Risoluzione dei problemi relativi a un modello di rilevatore	253
Informazioni diagnostiche	254
Analisi di un modello di rilevatore (console)	268
Analisi di un modello di rilevatore () AWS CLI	269
Comandi	274
Operazioni AWS IoT Events	274
Dati AWS IoT Events	274
Cronologia dei documenti	275
Aggiornamenti precedenti	276
.....	cclxxix

Che cos'è AWS IoT Events?

AWS IoT Events consente di monitorare le flotte di apparecchiature o dispositivi per rilevare eventuali guasti o cambiamenti di funzionamento e di attivare azioni quando si verificano tali eventi. AWS IoT Events osserva continuamente i dati dei sensori IoT provenienti da dispositivi, processi, applicazioni e altri AWS servizi per identificare eventi significativi in modo da poter intervenire.

Puoi utilizzarlo AWS IoT Events per creare applicazioni complesse di monitoraggio degli eventi nel AWS cloud a cui puoi accedere tramite la AWS IoT Events console o le API.



Vantaggi e caratteristiche

Accetta input da più fonti

AWS IoT Events accetta input da molte fonti di dati di telemetria IoT. Questi includono dispositivi con sensori, applicazioni di gestione e altri AWS IoT servizi, come e. AWS IoT Core AWS IoT Analytics. È possibile inviare qualsiasi input di dati di telemetria AWS IoT Events utilizzando un'interfaccia API (BatchPutMessageAPI) standard.

Usa espressioni logiche semplici per riconoscere modelli di eventi complessi

AWS IoT Events è in grado di riconoscere modelli di eventi che coinvolgono più input da un singolo dispositivo o applicazione IoT o da diverse apparecchiature e molti sensori indipendenti. Ciò è particolarmente utile perché ogni sensore e applicazione fornisce informazioni importanti. Ma solo combinando diversi dati di sensori e applicazioni è possibile ottenere un quadro completo delle prestazioni e della qualità delle operazioni. È possibile configurare i AWS IoT Events

rilevatori per riconoscere questi eventi utilizzando espressioni logiche semplici anziché codice complesso.

Attiva azioni basate sugli eventi

AWS IoT Events ti consente di attivare azioni direttamente in Amazon Simple Notification Service (Amazon SNS), Lambda AWS IoT Core, Amazon SQS e Amazon Kinesis Firehose. Puoi anche attivare una AWS Lambda funzione utilizzando il motore AWS IoT delle regole che consente di intraprendere azioni utilizzando altri servizi, come Amazon Connect o le tue applicazioni di pianificazione delle risorse aziendali (ERP).

AWS IoT Events include una libreria predefinita di azioni che puoi intraprendere e ti consente anche di definirne di personalizzate.

Scalabilità automatica per soddisfare le esigenze della flotta

AWS IoT Events si ridimensiona automaticamente quando si collegano dispositivi omogenei. È possibile definire un rilevatore una sola volta per un tipo specifico di dispositivo e il servizio ridimensionerà e gestirà automaticamente tutte le istanze del dispositivo a cui si connette. AWS IoT Events

Casi d'uso

Monitoraggio e manutenzione dei dispositivi remoti

È necessario monitorare una flotta di macchine distribuite in remoto. Se una di esse smette di funzionare e non si dispone di ulteriori informazioni sulla causa dell'errore, potrebbe essere necessario sostituire immediatamente l'intera unità o macchina di elaborazione. Ma questo non è sostenibile. Con AWS IoT Events puoi ricevere messaggi da più sensori su ogni macchina e diagnosticare il problema esatto utilizzando i codici di errore che vengono inviati nel tempo. Invece di sostituire tutto, ora disponi delle informazioni necessarie per inviare a un tecnico solo la parte da sostituire. Con milioni di macchine, i risparmi possono arrivare a milioni di dollari, riducendo il costo totale di proprietà o manutenzione di ogni macchina.

Gestisci i robot industriali

Implementate robot all'interno delle vostre strutture per automatizzare lo spostamento dei pacchi. Per ridurre al minimo il costo dei robot, i robot dispongono di sensori semplici e a basso costo che segnalano le informazioni al cloud. Tuttavia, i robot dispongono di dozzine di sensori e centinaia di modalità operative, il che rende difficile rilevare i problemi non appena si verificano. In questo

modo AWS IoT Events, puoi creare un sistema esperto che elabora i dati dei sensori nel cloud e crea avvisi per avvisare automaticamente il personale tecnico se un guasto è imminente.

Tieni traccia dei sistemi di automazione degli edifici

Gestite un gran numero di data center che devono essere monitorati per verificare l'alta temperatura e la bassa umidità per evitare guasti alle apparecchiature che si verificano quando vengono superate queste soglie ambientali. I sensori utilizzati vengono acquistati da molti produttori e ogni tipo è dotato di un proprio software di gestione. Tuttavia, il software di gestione di diversi fornitori non è compatibile, il che rende difficile l'individuazione dei problemi. In questo modo AWS IoT Events, è possibile impostare avvisi per notificare agli analisti operativi i problemi relativi ai sistemi di riscaldamento e raffreddamento con largo anticipo rispetto ai guasti. In questo modo, è possibile evitare l'arresto non programmato del data center, che comporterebbe un costo di migliaia di dollari in termini di sostituzione delle apparecchiature e una potenziale perdita di fatturato.

Configurazione AWS IoT Events

Se non ne hai uno Account AWS, completa i passaggi seguenti per crearne uno.

Per iscriverti a un Account AWS

1. Apri la pagina all'indirizzo <https://portal.aws.amazon.com/billing/signup>.
2. Segui le istruzioni online.

Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

Quando ti iscrivi a un Account AWS, Utente root dell'account AWS viene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, [assegna l'accesso amministrativo a un utente amministrativo](#) e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

Configurazione delle autorizzazioni per AWS IoT Events

Questa sezione descrive i ruoli e le autorizzazioni necessari per utilizzare alcune funzionalità di AWS IoT Events. È possibile utilizzare AWS CLI i comandi o la console AWS Identity and Access Management (IAM) per creare ruoli e politiche di autorizzazione associate per accedere alle risorse o eseguire determinate funzioni. AWS IoT Events

La [IAM User Guide](#) contiene informazioni più dettagliate sul controllo sicuro delle autorizzazioni di accesso AWS alle risorse. Per informazioni specifiche su AWS IoT Events, consulta [Azioni, risorse e chiavi di condizione](#) per. AWS IoT Events

Per utilizzare la console IAM per creare e gestire ruoli e autorizzazioni, consulta il [tutorial IAM: delega l'accesso tra AWS account utilizzando i ruoli IAM](#).

Note

Le chiavi possono contenere da 1 a 128 caratteri e possono includere:

- lettere maiuscole o minuscole a-z
- numeri 0-9

- caratteri speciali -, _ o:.

Autorizzazioni di azione

AWS IoT Events consente di attivare azioni che utilizzano altri AWS servizi. A tal fine, è necessario concedere AWS IoT Events l'autorizzazione a eseguire queste azioni per conto dell'utente. Questa sezione contiene un elenco delle azioni e un esempio di politica che concede il permesso di eseguire tutte queste azioni sulle risorse. Se necessario, modificate i riferimenti alla *regione* e *all'ID dell'account*. Quando possibile, dovresti anche modificare i caratteri jolly (*) per fare riferimento a risorse specifiche a cui accederai. Puoi utilizzare la console IAM per concedere l'autorizzazione AWS IoT Events all'invio di un avviso Amazon SNS che hai definito.

AWS IoT Events supporta le seguenti azioni che consentono di utilizzare un timer o impostare una variabile:

- [setTimer](#) per creare un timer.
- [resetTimer](#) per resettare il timer.
- [clearTimer](#) per eliminare il timer.
- [setVariable](#) per creare una variabile.

AWS IoT Events supporta le seguenti azioni che consentono di lavorare con AWS i servizi:

- [iotTopicPublish](#) per pubblicare un messaggio su un argomento MQTT.
- [iotEvents](#) a cui inviare dati AWS IoT Events come valore di input.
- [iotSiteWise](#) per inviare i dati a una proprietà di asset in AWS IoT SiteWise.
- [dynamoDB](#) per inviare dati a una tabella Amazon DynamoDB.
- [dynamoDBv2](#) per inviare dati a una tabella Amazon DynamoDB.
- [firehose](#) per inviare dati a uno stream Amazon Data Firehose.
- [lambda](#) per richiamare una AWS Lambda funzione.
- [sns](#) per inviare dati come notifica push.
- [sqs](#) per inviare dati a una coda Amazon SQS.

Example Policy

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

Protezione dei dati di input

È importante considerare chi può concedere l'accesso ai dati di input da utilizzare in un modello di rilevatore. Se disponi di un utente o di un'entità di cui desideri limitare le autorizzazioni generali, ma a cui è consentito creare o aggiornare un modello di rilevatore, devi anche concedere l'autorizzazione a tale utente o entità per aggiornare il routing di input. Ciò significa che oltre a concedere l'autorizzazione per `iotevents:CreateDetectorModel` e `iotevents:UpdateDetectorModel`, è necessario concedere anche l'autorizzazione per `iotevents:UpdateInputRouting`.

Example

La seguente politica aggiunge l'autorizzazione per `iotevents:UpdateInputRouting`.

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

Puoi specificare un elenco di Amazon Resource Names (ARN) di input anziché il carattere jolly "*" per "Resource" per limitare questa autorizzazione a input specifici. Ciò consente di limitare l'accesso ai dati di input utilizzati dai modelli di rilevatori creati o aggiornati dall'utente o dall'entità.

Politica del ruolo CloudWatch di registrazione di Amazon

I seguenti documenti politici forniscono la politica sui ruoli e la politica di fiducia AWS IoT Events a cui puoi inviare i log per tuo CloudWatch conto.

Policy del ruolo:

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

Policy di trust:

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

```

    }
  ]
}

```

È inoltre necessaria una politica di autorizzazioni IAM allegata all'utente che consenta all'utente di passare i ruoli, come segue. Per ulteriori informazioni, consulta [Concessione a un utente delle autorizzazioni per il trasferimento di un ruolo a un AWS servizio](#) nella Guida per l'utente IAM.

```

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

```

È possibile utilizzare il comando seguente per inserire la politica delle risorse per CloudWatch i log. Ciò consente di AWS IoT Events inserire gli eventi di registro negli CloudWatch stream.

```

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

```

Utilizzate il seguente comando per inserire le opzioni di registrazione. Sostituisci il ruolo `roleArn` con il ruolo di registrazione che hai creato.

```

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

```

Policy sui ruoli di messaggistica di Amazon SNS

I seguenti documenti politici forniscono la politica di ruolo e la politica di fiducia che consentono di AWS IoT Events inviare messaggi SNS.

Policy del ruolo:

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

Policy di trust:

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


Guida introduttiva alla AWS IoT Events console

[Questa sezione mostra come creare un modello di input e un modello di rilevatore utilizzando la AWS IoT Events console.](#) Si modellano due stati di un motore: uno stato normale e uno di sovrappressione. Quando la pressione misurata nel motore supera una certa soglia, il modello passa dallo stato normale allo stato di sovrappressione. Quindi invia un messaggio Amazon SNS per avvisare un tecnico della condizione. Quando la pressione scende nuovamente al di sotto della soglia per tre letture consecutive della pressione, il modello torna allo stato normale e invia un altro messaggio Amazon SNS come conferma.

Controlliamo tre letture consecutive al di sotto della soglia di pressione per eliminare possibili interruzioni dovute a segnali di sovrappressione o normali, in caso di una fase di recupero non lineare o di una lettura anomala della pressione.

Sulla console, puoi anche trovare diversi modelli di rilevatori predefiniti che puoi personalizzare. Puoi anche utilizzare la console per importare modelli di rilevatori scritti da altri o esportare i tuoi modelli di rilevatori e utilizzarli in diverse regioni. AWS Se importate un modello di rilevatore, assicuratevi di creare gli input richiesti o di ricrearli per la nuova regione e di aggiornare gli eventuali ARN di ruolo utilizzati.

Sulla console puoi anche trovare diversi modelli di rilevatori predefiniti che puoi personalizzare. Puoi anche utilizzare la console per importare modelli di rilevatori scritti da altri o esportare i tuoi modelli di rilevatori e utilizzarli in diversi modi. Regioni AWS Se importate un modello di rilevatore, assicuratevi di creare gli input richiesti o di ricrearli per la nuova regione e di aggiornare gli eventuali ARN di ruolo utilizzati.

Usa la AWS IoT Events console per scoprire quanto segue.

Definisci gli input

Per poterli monitorare, dispositivi e processi devono disporre di un modo per ottenere i dati telemetrici in AWS IoT Events. Questo viene fatto inviando messaggi come input a. AWS IoT Events Ci sono diversi modi per farlo:

- Usa l' [BatchPutMessage](#) operazione.
- In AWS IoT Core, scrivi una regola [AWS IoT Events d'azione](#) per il motore di AWS IoT regole in AWS IoT Events cui inoltrare i dati del messaggio. È necessario identificare l'input per nome.

- In AWS IoT Analytics, usa l' [CreateDataset](#) operazione per creare un set di dati `contentDeliveryRules`. Queste regole specificano l' AWS IoT Events input a cui i contenuti del set di dati vengono inviati automaticamente.

Prima che i dispositivi possano inviare dati in questo modo, è necessario definire uno o più input. A tale scopo, assegnate un nome a ogni input e specificate quali campi dei dati dei messaggi in arrivo vengono monitorati dall'input.

Create un modello di rilevatore

Definite un modello di rilevatore (un modello dell'apparecchiatura o del processo) utilizzando gli stati. Per ogni stato, definite una logica condizionale (booleana) che valuti gli input in ingresso per rilevare eventi significativi. Quando il modello del rilevatore rileva un evento, può modificare lo stato o avviare azioni personalizzate o predefinite utilizzando altri servizi. AWS È possibile definire eventi aggiuntivi che avviano azioni quando si entra o si esce da uno stato e, facoltativamente, quando viene soddisfatta una condizione.

In questo tutorial, invii un messaggio Amazon SNS come azione quando il modello entra o esce da un determinato stato.

Monitora un dispositivo o un processo

Se monitorate più dispositivi o processi, specificate un campo in ogni input che identifichi il particolare dispositivo o processo da cui proviene l'input. Visualizza il key campo in `CreateDetectorModel`. Quando il campo di input identificato da key riconosce un nuovo valore, viene identificato un nuovo dispositivo e viene creato un rilevatore. Ogni rilevatore è un'istanza del modello di rilevatore. Il nuovo rilevatore continua a rispondere agli input provenienti da quel dispositivo fino a quando il modello di rilevatore non viene aggiornato o eliminato.

Se monitorate un singolo processo (anche se diversi dispositivi o sottoprocessi inviano input), non specificate un campo identificativo univoco. key In questo caso, il modello crea un singolo rilevatore (istanza) quando arriva il primo input.

Inviare messaggi come input al vostro modello di rilevatore

Esistono diversi modi per inviare un messaggio da un dispositivo o elaborarlo come input in un AWS IoT Events rilevatore che non richiedono l'esecuzione di una formattazione aggiuntiva del messaggio. In questo tutorial, usi la AWS IoT console per scrivere una regola [AWS IoT Events d'azione](#) per il motore di AWS IoT regole che inoltra i dati dei messaggi. AWS IoT Events

Per fare ciò, identifica l'input per nome e continua a utilizzare la AWS IoT console per generare messaggi che vengono inoltrati come input a. AWS IoT Events

Note

Questo tutorial utilizza la console per creare gli stessi input ed è detector model mostrato nell'esempio all'indirizzo. [Tutorial](#) Puoi usare questo esempio JSON per aiutarti a seguire il tutorial.

Argomenti

- [Prerequisiti](#)
- [Crea un input](#)
- [Creare un modello di rilevatore](#)
- [Invia input per testare il modello del rilevatore](#)

Prerequisiti

Se non hai un AWS account, creane uno.

1. Apri la pagina all'indirizzo <https://portal.aws.amazon.com/billing/signup>.
2. Segui le istruzioni online.

Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

Quando ti registri per un Account AWS, ne Utente root dell'account AWSviene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, [assegna l'accesso amministrativo a un utente amministrativo](#) e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

3. Crea due argomenti su Amazon Simple Notification Service (Amazon SNS).

Questo tutorial (e l'esempio corrispondente) presuppone che tu abbia creato due argomenti di Amazon SNS. Gli ARN di questi argomenti sono mostrati come: `arn:aws:sns:us-east-1:123456789012:underPressureAction` e `arn:aws:sns:us-east-1:123456789012:pressureClearedAction` Sostituisci questi valori con gli ARN degli argomenti di Amazon SNS che crei. Per ulteriori informazioni, consulta la [Guida per gli sviluppatori di Amazon Simple Notification Service](#).

In alternativa alla pubblicazione di avvisi su argomenti di Amazon SNS, puoi fare in modo che i rilevatori inviino messaggi MQTT con un argomento da te specificato. Con questa opzione, puoi verificare che il tuo modello di rilevatore stia creando istanze e che tali istanze inviino avvisi utilizzando la console AWS IoT Core per sottoscrivere e monitorare i messaggi inviati a tali argomenti MQTT. È inoltre possibile definire il nome dell'argomento MQTT in modo dinamico in fase di esecuzione utilizzando un input o una variabile creata nel modello del rilevatore.

4. Scegliete uno che supporti Regione AWS . AWS IoT Events Per ulteriori informazioni, consulta [AWS IoT Events](#) nella Riferimenti generali di AWS. Per assistenza, consulta [Lavorare con il AWS Management Console](#) nella Guida introduttiva a AWS Management Console.

Crea un input

Quando create gli input per i vostri modelli, vi consigliamo di raccogliere file che contengano esempi di payload di messaggi inviati dai dispositivi o dai processi per segnalarne lo stato di salute. La presenza di questi file consente di definire gli input necessari.

È possibile creare un input tramite diversi metodi descritti in questa sezione.

Per iniziare, create un file denominato `input.json` nel file system locale con il seguente contenuto:

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

Ora che avete questo `input.json` file iniziale, potete creare un input. Utilizzate uno degli argomenti di questa sezione per istruzioni sulla creazione di un input utilizzando il pannello di navigazione o utilizzando il modello del rilevatore.

Argomenti

- [Creare un input nel riquadro di navigazione](#)
- [Creare un input nel modello del rilevatore](#)

Creare un input nel riquadro di navigazione

Questo argomento mostra come creare un input, per un modello di allarme o un modello di rilevatore, tramite il pannello di navigazione.

1. Accedi alla [AWS IoT Events console](#) o seleziona l'opzione Crea un nuovo AWS IoT Events account.
2. Nella AWS IoT Events console, nell'angolo in alto a sinistra, seleziona ed espandi il riquadro di navigazione.
3. Nel riquadro di navigazione a sinistra, seleziona Ingressi.
4. Nell'angolo destro della console, scegli Crea input.
5. Per l'input, inserisci una InputNamedescrizione opzionale e scegli Carica file. Nella finestra di dialogo visualizzata, selezionate il `input.json` file creato nella panoramica per [creare un input](#).
6. Per Scegli gli attributi di input, seleziona gli attributi da utilizzare e scegli Crea. In questo esempio, selezioniamo `motorid` e `SensorData.pressure`.

Creare un input nel modello del rilevatore

Questo argomento mostra come definire un input per un modello di rilevatore per ricevere dati o messaggi di telemetria.

1. Apri la [AWS IoT Events console](#).
2. Nella AWS IoT Events console, scegli Crea modello di rilevatore.
3. Scegli Crea nuova.
4. Scegliere Create input (Crea input).
5. Per l'input, inserisci una InputNamedescrizione opzionale e scegli Carica file. Nella finestra di dialogo visualizzata, selezionate il `input.json` file creato nella panoramica per [creare un input](#).
6. Per Scegli gli attributi di input, seleziona gli attributi da utilizzare e scegli Crea. In questo esempio, selezioniamo `motorid` e `SensorData.pressure`.

Creare un modello di rilevatore

In questo argomento, definirete un modello di rivelatore (un modello dell'apparecchiatura o del processo) utilizzando gli stati.

Per ogni stato, si definisce una logica condizionale (booleana) che valuta gli input in ingresso per rilevare un evento significativo. Quando viene rilevato un evento, cambia lo stato e può avviare azioni aggiuntive. Questi eventi sono noti come eventi di transizione.

Nei vostri stati, definite anche eventi che possono eseguire azioni ogni volta che il rilevatore entra o esce da quello stato o quando viene ricevuto un input (questi sono noti come `OnEnter` `OnInput` eventi `OnExit` e). Le azioni vengono eseguite solo se la logica condizionale dell'evento restituisce un risultato positivo. `true`

Per creare un modello di rilevatore

1. Il primo stato del rilevatore è stato creato per te. Per modificarlo, seleziona il cerchio con l'etichetta `State_1` nello spazio di modifica principale.
2. Nel riquadro Stato, inserisci il nome dello stato e `OnEnter` scegli Aggiungi evento.
3. Nella pagina Aggiungi `OnEnter` evento, inserisci il nome dell'evento e la condizione dell'evento. In questo esempio, immettere `true` per indicare che l'evento viene sempre avviato quando viene inserito lo stato.
4. In Azioni relative agli eventi, scegli Aggiungi azione.
5. In Azioni relative agli eventi, procedi come segue:
 - a. Seleziona Imposta variabile
 - b. Per Operazione variabile, scegli Assegna valore.
 - c. In Nome variabile, immettete il nome della variabile da impostare.
 - d. Per Valore variabile, immettere il valore `0` (zero).
6. Selezionare Salva.

Una variabile, come quella che hai definito, può essere impostata (dato un valore) in qualsiasi caso nel modello del rilevatore. È possibile fare riferimento al valore della variabile (ad esempio, nella logica condizionale di un evento) solo dopo che il rilevatore ha raggiunto uno stato ed eseguito un'azione in cui è definito o impostato.

7. Nel riquadro Stato, scegliete la X accanto a Stato per tornare alla palette del modello Detector.
8. Per creare un secondo stato del rilevatore, nella palette del modello Detector, scegli Stato e trascinalo nello spazio di modifica principale. Questo crea uno stato intitolato `untitled_state_1`
9. Pausa sul primo stato (Normale). Viene visualizzata una freccia sulla circonferenza dello stato.

10. Fate clic e trascinate la freccia dal primo stato al secondo stato. Viene visualizzata una linea diretta dal primo stato al secondo stato (etichettata Senza titolo).
11. Seleziona la riga Senza titolo. Nel riquadro Evento di transizione, inserite un nome di evento e una logica di attivazione dell'evento.
12. Nel riquadro Evento di transizione, scegli Aggiungi azione.
13. Nel riquadro Aggiungi azioni relative agli eventi di transizione, scegli Aggiungi azione.
14. Per Scegli un'azione, scegli Imposta variabile.
 - a. Per Operazione variabile, scegli Assegna valore.
 - b. Per Nome variabile, immettete il nome della variabile.
 - c. In Assegna valore, inserisci un valore come: `$variable.pressureThresholdBreached + 3`
 - d. Selezionare Salva.
15. Selezionate il secondo stato `untitled_state_1`.
16. Nel riquadro Stato, inserisci il nome dello stato e per On Enter, scegli Aggiungi evento.
17. Nella pagina Aggiungi OnEnter evento, inserisci il nome dell'evento e la condizione dell'evento. Selezionare Add action (Aggiungi operazione).
18. Per Scegli un'azione, scegli Invia messaggio SNS.
 - a. Per l'argomento SNS, inserisci l'ARN di destinazione del tuo argomento Amazon SNS.
 - b. Selezionare Salva.
19. Continua ad aggiungere gli eventi dell'esempio.
 - a. Per OnInput, scegliete Aggiungi evento e inserite e salvate le seguenti informazioni sull'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. Per OnInput, scegli Aggiungi evento e inserisci e salva le seguenti informazioni sull'evento.

```

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

```

- c. Per OnExit, scegli Aggiungi evento e inserisci e salva le seguenti informazioni sull'evento utilizzando l'ARN dell'argomento Amazon SNS che hai creato.

```

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

```

20. Pausa sul secondo stato (Pericoloso). Viene visualizzata una freccia sulla circonferenza dello stato
21. Fate clic e trascinate la freccia dal secondo stato al primo stato. Viene visualizzata una linea diretta con l'etichetta Untitled.
22. Scegliete la riga Senza titolo e nel riquadro Evento di transizione, inserite il nome dell'evento e la logica di attivazione dell'evento utilizzando le seguenti informazioni.

```

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

```

Per ulteriori informazioni sul motivo per cui testiamo il `$input` valore e il `$variable` valore nella logica di attivazione, consulta la voce relativa alla disponibilità dei valori delle variabili in.

[Restrizioni e limitazioni del modello di rilevatore](#)

23. Seleziona lo stato di inizio. Per impostazione predefinita, questo stato è stato creato quando è stato creato un modello di rilevatore). Nel riquadro Start, scegliete lo stato di destinazione (ad esempio, Normale).
24. Quindi, configura il modello del rilevatore per ascoltare gli input. Nell'angolo in alto a destra, scegli Pubblica.

25. Nella pagina Publish Detector model, procedi come segue.
 - a. Immettete il nome del modello del rilevatore, una descrizione e il nome di un ruolo. Questo ruolo è stato creato per te.
 - b. Scegli Crea un rilevatore per ogni valore chiave univoco. Per creare e utilizzare il tuo ruolo, segui i passaggi indicati [Configurazione delle autorizzazioni per AWS IoT Events](#) e inseriscilo come ruolo qui.
26. Per la chiave di creazione del rilevatore, scegli il nome di uno degli attributi dell'input che hai definito in precedenza. L'attributo scelto come chiave di creazione del rilevatore deve essere presente in ogni messaggio di input e deve essere unico per ogni dispositivo che invia messaggi. Questo esempio utilizza l'attributo motorid.
27. Scegliere Save and publish (Salva e pubblica).

Note

Il numero di rilevatori univoci creati per un determinato modello di rilevatore si basa sui messaggi di input inviati. Quando viene creato un modello di rilevatore, viene selezionata una chiave dagli attributi di input. Questa chiave determina quale istanza del rilevatore utilizzare. Se la chiave non è mai stata vista prima (per questo modello di rilevatore), viene creata una nuova istanza del rilevatore. Se la chiave è già stata vista, utilizziamo l'istanza del rilevatore esistente corrispondente a questo valore chiave.

È possibile creare una copia di backup della definizione del modello di rilevatore (in JSON), ricreare o aggiornare il modello del rilevatore o utilizzarla come modello per creare un altro modello di rilevatore.

È possibile eseguire questa operazione dalla console o utilizzando il seguente comando CLI. Se necessario, modificate il nome del modello di rilevatore in modo che corrisponda a quello usato quando lo avete pubblicato nel passaggio precedente.

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

Questo crea un file (`motorDetectorModel.json`) con contenuti simili al seguente.

```
{
  "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":
"pressureThresholdBreached",
        "value": "0"
    }
    }
    ],
    "condition": "true"
}
]
},
"onExit": {
    "events": []
}
},
{
    "onInput": {
        "transitionEvents": [
            {
                "eventName": "Back to Normal",
                "actions": [],
                "condition": "$variable.pressureThresholdBreached <= 1
&& $input.PressureInput.sensorData.pressure <= 70",
                "nextState": "Normal"
            }
        ],
        "events": [
            {
                "eventName": "Overpressurized",
                "actions": [
                    {
                        "setVariable": {
                            "variableName":
"pressureThresholdBreached",
                            "value": "3"
                        }
                    }
                ],
                "condition": "$input.PressureInput.sensorData.pressure
> 70"
            }
        ],
        {
            "eventName": "Pressure Okay",
            "actions": [
                {

```

```

        "setVariable": {
            "variableName":
"pressureThresholdBreached",
            "value":
"$variable.pressureThresholdBreached - 1"
        }
    ],
    "condition": "$input.PressureInput.sensorData.pressure
<= 70"
}
],
},
"stateName": "Dangerous",
"onEnter": {
    "events": [
        {
            "eventName": "Pressure Threshold Breached",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:MyIoTButtonSNSTopic"
                    }
                }
            ],
            "condition": "$variable.pressureThresholdBreached > 1"
        }
    ],
},
"onExit": {
    "events": [
        {
            "eventName": "Normal Pressure Restored",
            "actions": [
                {
                    "sns": {
                        "targetArn": "arn:aws:sns:us-
west-2:123456789012:IoTVirtualButtonTopic"
                    }
                }
            ],
            "condition": "true"
        }
    ]
}

```

```
    ]
  }
}
],
"initialStateName": "Normal"
}
}
```

Invia input per testare il modello del rilevatore

Esistono diversi modi per ricevere i dati di telemetria in (vedi). AWS IoT Events [Azioni supportate](#)

Questo argomento mostra come creare una AWS IoT regola nella AWS IoT console che inoltri i messaggi come input al rilevatore. AWS IoT Events È possibile utilizzare il client MQTT della AWS IoT console per inviare messaggi di test. È possibile utilizzare questo metodo per ottenere dati di telemetria su AWS IoT Events quando i dispositivi sono in grado di inviare messaggi MQTT utilizzando il broker di messaggi. AWS IoT

Per inviare input per testare il modello del rilevatore

1. Apri la [AWS IoT Core console](#). Nel riquadro di navigazione a sinistra, sotto Gestisci, scegli Routing dei messaggi, quindi scegli Regole.
2. Scegli Crea regola in alto a destra.
3. Nella pagina Crea una regola, completa i seguenti passaggi:
 1. Fase 1: Specificare le proprietà della regola. Completare i seguenti campi:
 - Nome della regola. Inserisci un nome per la regola, ad esempio `MyIoTEventsRule`.

Note

Non utilizzare spazi.

- Descrizione della regola. Si tratta di un'opzione facoltativa.
 - Seleziona Successivo.
2. Fase 2. Configurare l'istruzione SQL. Completare i seguenti campi:
 - Versione SQL. Seleziona l'opzione appropriata dall'elenco.
 - Istruzione SQL. Specificare **`SELECT *, topic(2) as motorid FROM 'motors/+/status'`**.

Seleziona Successivo.

3. Fase 3. Allega azioni alle regole. Nella sezione Azioni relative alle regole, completa quanto segue:

- Azione 1. Seleziona IoT Events. Vengono visualizzati i seguenti campi:
 - a. Nome di input. Seleziona l'opzione appropriata dall'elenco. Se l'input non viene visualizzato, scegli Aggiorna.

Per creare un nuovo input, scegli Create IoT Events input. Completare i seguenti campi:

- Nome di input. Specificare PressureInput.
- Descrizione. Si tratta di un'opzione facoltativa.
- Carica un file JSON. Carica una copia del tuo file JSON. In questa schermata è presente un collegamento a un file di esempio, se non disponi di un file. Il codice include:

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

- Scegli gli attributi di input. Seleziona le opzioni appropriate.
- Tag. Si tratta di un'opzione facoltativa.

Scegli Crea.

Torna alla schermata Crea regola e aggiorna il campo Input name. Seleziona l'input che hai appena creato.

- b. Modalità Batch. Si tratta di un'opzione facoltativa. Se il payload è una matrice di messaggi, selezionate questa opzione.
- c. ID del messaggio. Questo passaggio è facoltativo, ma è consigliato.
- d. Ruolo IAM. Seleziona il ruolo appropriato dall'elenco. Se il ruolo non è elencato, scegli Crea nuovo ruolo.

Digita il nome del ruolo e scegli Crea.

Per aggiungere un'altra regola, scegli **Aggiungi azione** alla regola

- **Azione di errore.** Questa sezione è facoltativa. Per aggiungere un'azione, scegli **Aggiungi azione di errore** e seleziona l'azione appropriata dall'elenco.

Completa i campi visualizzati.

- **Seleziona Successivo.**

4. Fase 4. Rivedi e crea. Controlla le informazioni sullo schermo e scegli **Crea**.

4. Nel riquadro di navigazione a sinistra, sotto **Test**, scegli **MQTT test client**.

5. Scegliere **Publish to a topic** (Pubblica in un argomento). Completare i seguenti campi:

- **Nome dell'argomento.** Inserisci un nome per identificare il messaggio, ad esempio `motoris/Fulton-A32/status`.
- **Payload del messaggio.** Inserisci i seguenti dati:

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

Note

Cambia `messageId` ogni volta che pubblichi un nuovo messaggio.

6. Per **Publish**, mantieni invariato l'argomento, ma modifica il `pressure` payload impostandolo su un valore maggiore del valore di soglia specificato nel modello del rilevatore (ad **85** esempio).

7. Seleziona **Publish** (Pubblica).

L'istanza del rilevatore che hai creato genera e ti invia un messaggio Amazon SNS. Continua a inviare messaggi con valori di pressione superiori o inferiori alla soglia di pressione (70 per questo esempio) per vedere il rilevatore in funzione.

In questo esempio, devi inviare tre messaggi con valori di pressione inferiori alla soglia per tornare allo stato Normale e ricevere un messaggio Amazon SNS che indica che la condizione di sovrappressione è stata superata. Una volta tornato allo stato Normale, un messaggio con una

lettura della pressione superiore al limite fa sì che il rilevatore entri nello stato Pericoloso e invii un messaggio Amazon SNS indicante tale condizione.

Ora che hai creato un semplice modello di input e rilevatore, prova quanto segue.

- Visualizza altri esempi di modelli di rilevatori (modelli) sulla console.
- Segui i passaggi indicati [Esempio semplice step-by-step](#) per creare un modello di input e di rilevatore utilizzando AWS CLI
- Scopri i dettagli del materiale [Espressioni](#) utilizzato negli eventi.
- Ulteriori informazioni su [Azioni supportate](#).
- Se qualcosa non funziona, vedi [Risoluzione dei problemi AWS IoT Events](#).

Le migliori pratiche per AWS IoT Events

Segui queste best practice per trarne il massimo vantaggio. AWS IoT Events

Argomenti

- [Abilita la CloudWatch registrazione di Amazon durante lo sviluppo di modelli di AWS IoT Events rilevatori](#)
- [Pubblica regolarmente per salvare il modello del rilevatore quando lavori nella console AWS IoT Events](#)
- [Archivia AWS IoT Events i tuoi dati per evitare possibili perdite di dati dovute a un lungo periodo di inattività](#)

Abilita la CloudWatch registrazione di Amazon durante lo sviluppo di modelli di AWS IoT Events rilevatori

Amazon CloudWatch monitora AWS le tue risorse e le applicazioni su cui esegui AWS in tempo reale. Con CloudWatch, ottieni visibilità a livello di sistema sull'uso delle risorse, sulle prestazioni delle applicazioni e sullo stato operativo. Quando si sviluppa o si esegue il debug di un modello di AWS IoT Events rilevatore, CloudWatch consente di sapere cosa AWS IoT Events sta facendo e gli eventuali errori riscontrati.

Per abilitare CloudWatch

1. Se non l'hai già fatto, segui i passaggi indicati [Configurazione delle autorizzazioni per AWS IoT Events](#) per creare un ruolo con una policy allegata che conceda l'autorizzazione a creare e gestire CloudWatch i log per. AWS IoT Events
2. Accedere alla [console AWS IoT Events](#).
3. Nel pannello di navigazione scegli Impostazioni.
4. Nella pagina Impostazioni, scegli Modifica.
5. Nella pagina Modifica opzioni di registrazione, nella sezione Opzioni di registrazione, procedi come segue:
 - a. Per Livello di verbosità, selezionate un'opzione.
 - b. Per Seleziona ruolo, seleziona un ruolo con autorizzazioni sufficienti per eseguire le azioni di registrazione che hai scelto.

- c. (Facoltativo) Se avete scelto Debug per il livello di dettaglio, potete aggiungere obiettivi di debug effettuando le seguenti operazioni:
 - i. In Obiettivi di debug, scegliete Aggiungi opzione modello.
 - ii. Immettete il nome del modello del rilevatore e (facoltativo) specificate KeyValuesi modelli di rilevatore e i rilevatori specifici (istanze) da registrare.
6. Scegli Aggiorna.

Le opzioni di registrazione sono state aggiornate correttamente.

Pubblica regolarmente per salvare il modello del rilevatore quando lavori nella console AWS IoT Events

Quando usi la AWS IoT Events console, il lavoro in corso viene salvato localmente nel tuo browser. Tuttavia, è necessario scegliere Pubblica in cui salvare il modello del rilevatore. AWS IoT Events Dopo aver pubblicato un modello di rilevatore, il lavoro pubblicato sarà disponibile in qualsiasi browser che utilizzi per accedere al tuo account.

Note

Se non pubblichi il tuo lavoro, non verrà salvato. Dopo aver pubblicato un modello di rilevatore, non puoi cambiarne il nome. Tuttavia, potete continuare a modificarne la definizione.

Archivia AWS IoT Events i tuoi dati per evitare possibili perdite di dati dovute a un lungo periodo di inattività

Se non li utilizzi AWS IoT Events per un periodo di tempo significativo, i tuoi dati, compresi i modelli dei rilevatori, potrebbero essere eliminati automaticamente. Un periodo di tempo significativo potrebbe significare, ad esempio, che non vengano addebitati costi e che non si creino modelli di rilevatori. Tuttavia, non elimineremo i dati o i modelli di rilevatori senza darti un preavviso di almeno 30 giorni. Se hai bisogno di archiviare i dati per un periodo di tempo prolungato, prendi in considerazione l'utilizzo di [servizi AWS di archiviazione](#).

Tutorial

In questo capitolo viene illustrato come:

- Fatevi aiutare a decidere quali stati includere nel vostro modello di rilevatore e stabilite se avete bisogno di una o più istanze del rilevatore.
- Segui un esempio che utilizza il. AWS CLI
- Crea un input per ricevere dati di telemetria da un dispositivo e un modello di rilevatore per monitorare e generare report sullo stato del dispositivo che invia tali dati.
- Esamina le restrizioni e le limitazioni relative agli input, ai modelli di rilevatori e al servizio. AWS IoT Events
- Guarda un esempio più complesso di modello di rilevatore, con commenti inclusi.

Argomenti

- [Utilizzo AWS IoT Events per monitorare i dispositivi IoT](#)
- [Esempio semplice step-by-step](#)
- [Restrizioni e limitazioni del modello di rilevatore](#)
- [Un esempio commentato: controllo della temperatura HVAC](#)

Utilizzo AWS IoT Events per monitorare i dispositivi IoT

È possibile AWS IoT Events utilizzarlo per monitorare i dispositivi o i processi e intervenire in base a eventi significativi. Per farlo, segui questi passaggi di base:

Crea input

È necessario disporre di un modo per i dispositivi e i processi di inserire i dati di telemetria. AWS IoT Events Puoi farlo inviando messaggi come input a. AWS IoT Events È possibile inviare messaggi come input in diversi modi:

- Usa l' [BatchPutMessage](#) operazione.
- [Definite un'iotEventsazione per il motore delle AWS IoT Core regole.](#) La regola-azione inoltra i dati del messaggio dall'utente a. AWS IoT Events

- In AWS IoT Analytics, usa l'[CreateDataset](#) operazione per creare un set di dati con `contentDeliveryRules`. Queste regole specificano l'AWS IoT Events input a cui i contenuti del set di dati vengono inviati automaticamente.
- Definisci un'[azione IoT Events](#) in un modello o evento di `onInput` un AWS IoT Events rilevatore. `onExit transitionEvents` Le informazioni sull'istanza del modello di rilevatore e sull'evento che ha avviato l'azione vengono restituite al sistema come input con il nome specificato.

Prima che i dispositivi inizino a inviare dati in questo modo, è necessario definire uno o più input. A tale scopo, assegna un nome a ogni input e specifica quali campi dei dati dei messaggi in arrivo vengono monitorati dall'input. AWS IoT Events riceve il proprio input, sotto forma di payload JSON, da molte fonti. Ogni input può essere utilizzato da solo o combinato con altri input per rilevare eventi più complessi.

Crea un modello di rilevatore

Definisci un modello di rilevatore (un modello dell'apparecchiatura o del processo) utilizzando gli stati. Per ogni stato, definisci la logica condizionale (booleana) che valuta gli input in entrata per rilevare eventi significativi. Quando viene rilevato un evento, puoi modificare lo stato o avviare azioni personalizzate o predefinite utilizzando altri servizi. AWS È possibile definire eventi aggiuntivi che avviano azioni quando si entra o si esce da uno stato e, facoltativamente, quando viene soddisfatta una condizione.

In questo tutorial, invia un messaggio Amazon SNS come azione quando il modello entra o esce da un determinato stato.

Monitora un dispositivo o un processo

Se stai monitorando diversi dispositivi o processi, specifichi un campo in ogni input che identifica il particolare dispositivo o processo da cui proviene l'input. (Vedi il key campo `inCreateDetectorModel`.) Quando viene identificato un nuovo dispositivo (viene visualizzato un nuovo valore nel campo di immissione identificato da `key`), viene creato un rilevatore. (Ogni rilevatore è un'istanza del modello di rilevatore.) Quindi il nuovo rilevatore continua a rispondere agli input provenienti da quel dispositivo fino a quando il modello di rilevatore non viene aggiornato o eliminato.

Se stai monitorando un singolo processo (anche se diversi dispositivi o sottoprocessi inviano input), non specifichi un campo identificativo univoco. `key` In questo caso, viene creato un singolo rilevatore (istanza) all'arrivo del primo input.

Invia messaggi come input al tuo modello di rilevatore

Esistono diversi modi per inviare un messaggio da un dispositivo o elaborarlo come input in un AWS IoT Events rilevatore che non richiedono l'esecuzione di una formattazione aggiuntiva del messaggio. In questo tutorial, usi la AWS IoT console per scrivere una regola [AWS IoT Events](#) [d'azione](#) per il motore di AWS IoT Core regole che inoltra i dati dei messaggi. AWS IoT Events Per fare ciò, identificate l'input per nome. Quindi continui a utilizzare la AWS IoT console per generare alcuni messaggi che vengono inoltrati come input a. AWS IoT Events

Come fai a sapere di quali stati hai bisogno in un modello di rilevatore?

Per determinare quali stati deve avere il modello di rilevatore, decidete innanzitutto quali azioni potete intraprendere. Ad esempio, se la vostra automobile funziona a benzina, guardate l'indicatore del livello del carburante quando iniziate un viaggio per vedere se avete bisogno di fare rifornimento. Qui hai un'unica cosa da fare: dire all'autista di «andare a fare benzina». Il modello del rilevatore necessita di due stati: «l'auto non ha bisogno di carburante» e «l'auto ha bisogno di carburante». In generale, si desidera definire uno stato per ogni azione possibile, più uno in più per quando non è richiesta alcuna azione. Funziona anche se l'azione stessa è più complicata. Ad esempio, potresti voler cercare e includere informazioni su dove trovare la stazione di rifornimento più vicina o il prezzo più basso, ma lo fai quando invii il messaggio «vai a prendere benzina».

Per decidere in quale stato entrare successivamente, si esaminano gli input. Gli input contengono le informazioni di cui hai bisogno per decidere in quale stato ti trovi. Per creare un input, selezionate uno o più campi in un messaggio inviato dal dispositivo o dal processo che vi aiuta a decidere. In questo esempio, è necessario un input che indichi il livello attuale del carburante («percentuale di pieno»). Forse la tua auto ti sta inviando diversi messaggi, ognuno con diversi campi. Per creare questo input, è necessario selezionare il messaggio e il campo che riporta il livello attuale dell'indicatore del gas. La durata del viaggio che stai per intraprendere («distanza dalla destinazione») può essere codificata per semplificare le cose; puoi utilizzare la durata media del viaggio. Eseguirai alcuni calcoli in base all'input (in quanti galloni si traduce quella percentuale totale? è la lunghezza media del viaggio superiore alle miglia percorribili, considerando i galloni a disposizione e la media delle «miglia per gallone»). Esegui questi calcoli e invii messaggi negli eventi.

Finora hai due stati e un input. È necessario un evento nel primo stato che esegua i calcoli in base all'input e decida se passare al secondo stato. Questo è un evento di transizione. (transitionEvent sono nell'elenco degli onInput eventi di uno stato. Alla ricezione di un input in questo primo stato, l'evento esegue una transizione al secondo stato, se l'evento condition viene soddisfatto.) Quando raggiungi il secondo stato, invii il messaggio non appena entri nello stato. (Si

utilizza un `onEnter` evento. Entrando nel secondo stato, questo evento invia il messaggio. Non è necessario attendere l'arrivo di un altro input.) Esistono altri tipi di eventi, ma è tutto ciò che serve per un semplice esempio.

Gli altri tipi di eventi sono `onExit` e `onInput`. Non appena viene ricevuto un input e la condizione viene soddisfatta, un `onInput` evento esegue le azioni specificate. Quando un'operazione esce dallo stato corrente e la condizione viene soddisfatta, l'`onExit` evento esegue le azioni specificate.

Ti manca qualcosa? Sì, come si torna al primo stato «l'auto non ha bisogno di carburante»? Dopo aver riempito il serbatoio, l'input mostra che il serbatoio è pieno. Nel secondo stato è necessario un evento di transizione che ritorni al primo stato che si verifica quando viene ricevuto l'input (negli `onInput`: eventi del secondo stato). Dovrebbe tornare al primo stato se i calcoli mostrano che ora avete abbastanza gas per portarvi dove volete andare.

Queste sono le basi. Alcuni modelli di rilevatori diventano più complessi aggiungendo stati che riflettono input importanti, non solo azioni possibili. Ad esempio, potreste avere tre stati in un modello di rilevatore che tiene traccia della temperatura: uno stato «normale», uno stato «troppo caldo» e uno stato «potenziale problema». Si passa allo stato di potenziale problema quando la temperatura supera un certo livello, ma non è ancora diventata troppo alta. Non vuoi inviare un allarme a meno che non rimanga a questa temperatura per più di 15 minuti. Se la temperatura torna alla normalità prima di allora, il rilevatore torna allo stato normale. Se il timer scade, il rilevatore passa allo stato troppo caldo e invia un allarme, giusto per essere prudenti. Potresti fare la stessa cosa usando variabili e un insieme più complesso di condizioni di evento. Ma spesso è più semplice utilizzare un altro stato per, in effetti, memorizzare i risultati dei calcoli.

Come fai a sapere se hai bisogno di una o più istanze di un rilevatore?

Per decidere di quante istanze hai bisogno, chiediti «Cosa ti interessa sapere?» Supponiamo che tu voglia sapere che tempo fa oggi. Piove (stato)? Devi prendere un ombrello (azione)? Puoi avere un sensore che segnala la temperatura, un altro che segnala l'umidità e altri che segnalano la pressione barometrica, la velocità e la direzione del vento e le precipitazioni. Ma è necessario monitorare tutti questi sensori insieme per determinare lo stato delle condizioni meteorologiche (pioggia, neve, cielo coperto, sole) e l'azione appropriata da intraprendere (prendere un ombrello o applicare una protezione solare). Nonostante il numero di sensori, è necessario che un'unica istanza del rilevatore monitori lo stato delle condizioni meteorologiche e indichi le azioni da intraprendere.

Ma se ti occupi delle previsioni meteorologiche della tua regione, potresti avere più istanze di questo tipo di array di sensori, situate in diverse località della regione. Le persone di ogni località devono

sapere che tempo fa in quella località. In questo caso, sono necessarie più istanze del rilevatore. I dati riportati da ciascun sensore in ogni posizione devono includere un campo che avete designato come campokey. Questo campo consente di AWS IoT Events creare un'istanza di rilevatore per l'area e quindi di continuare a indirizzare queste informazioni a quell'istanza del rilevatore man mano che continua ad arrivare. Niente più capelli rovinati o nasi bruciati dal sole!

In sostanza, è necessaria un'istanza del rilevatore se si dispone di una situazione (un processo o una posizione) da monitorare. Se avete molte situazioni (sedi, processi) da monitorare, avete bisogno di più istanze del rilevatore.

Esempio semplice step-by-step

In questo esempio, chiamiamo le AWS IoT Events API utilizzando AWS CLI i comandi per creare un rilevatore che modella due stati di un motore: uno stato normale e una condizione di sovrappressione.

Quando la pressione misurata nel motore supera una certa soglia, il modello passa allo stato di sovrappressione e invia un messaggio Amazon Simple Notification Service (Amazon SNS) per avvisare un tecnico della condizione. Quando la pressione scende al di sotto della soglia per tre letture consecutive della pressione, il modello torna allo stato normale e invia un altro messaggio Amazon SNS per confermare che la condizione è stata risolta. Richiediamo tre letture consecutive al di sotto della soglia di pressione per eliminare la possibile interruzione dei messaggi di sovrappressione/normalità in caso di una fase di ripristino non lineare o di una lettura anomala di ripristino una tantum.

Di seguito è riportata una panoramica dei passaggi per creare il rilevatore.

Crea input.

Per poterli monitorare, dispositivi e processi devono disporre di un modo per ottenere i dati telemetrici in AWS IoT Events. Questa operazione viene eseguita inviando messaggi come input ad AWS IoT Events. Ci sono diversi modi per farlo:

- Usa l' [BatchPutMessage](#) operazione. Questo metodo è semplice ma richiede che i dispositivi o i processi siano in grado di accedere all'AWS IoT Events API tramite un SDK o il AWS CLI.
- In AWS IoT Core, scrivi una regola [AWS IoT Events d'azione](#) per il motore di AWS IoT Core regole in cui inoltrare i dati dei messaggi. AWS IoT Events Questo identifica l'input per nome. Utilizzate questo metodo se i vostri dispositivi o processi possono inviare messaggi, o lo stanno già facendo AWS IoT Core. Questo metodo richiede in genere una minore potenza di calcolo da parte di un dispositivo.

- In AWS IoT Analytics, utilizza l' [CreateDataset](#) operazione per creare un set di dati con `contentDeliveryRules` l'AWS IoT Events input specificato, in cui i contenuti del set di dati vengono inviati automaticamente. Utilizzate questo metodo se desiderate controllare i dispositivi o i processi sulla base di dati aggregati o analizzati in AWS IoT Analytics.

Prima che i dispositivi possano inviare dati in questo modo, è necessario definire uno o più input. A tale scopo, assegnate un nome a ogni input e specificate i campi dei dati dei messaggi in arrivo monitorati dall'input.

Create un modello di rilevatore

Create un modello di rilevatore (un modello dell'apparecchiatura o del processo) utilizzando gli stati. Per ogni stato, definite una logica condizionale (booleana) che valuti gli input in ingresso per rilevare eventi significativi. Quando viene rilevato un evento, può modificare lo stato o avviare azioni personalizzate o predefinite utilizzando altri servizi. AWS È possibile definire eventi aggiuntivi che avviano azioni quando si entra o si esce da uno stato e, facoltativamente, quando viene soddisfatta una condizione.

Monitora diversi dispositivi o processi

Se stai monitorando diversi dispositivi o processi e desideri tenere traccia di ciascuno di essi separatamente, specifica un campo in ogni input che identifichi il particolare dispositivo o processo da cui proviene l'input. Visualizza il key campo in `CreateDetectorModel`. Quando viene identificato un nuovo dispositivo (viene visualizzato un nuovo valore nel campo di input identificato da `key`), viene creata un'istanza del rilevatore. La nuova istanza del rilevatore continua a rispondere agli input provenienti da quel particolare dispositivo fino a quando il relativo modello di rilevatore non viene aggiornato o eliminato. Hai tanti rilevatori (istanze) unici quanti sono i valori univoci nei campi di input. `key`

Monitora un singolo dispositivo o processo

Se stai monitorando un singolo processo (anche se diversi dispositivi o sottoprocessi inviano input), non specifichi un campo identificativo `key` univoco. In questo caso, viene creato un singolo rilevatore (istanza) all'arrivo del primo input. Ad esempio, potreste avere sensori di temperatura in ogni stanza di una casa, ma solo un'unità HVAC per riscaldare o raffreddare l'intera casa. Quindi puoi controllarlo solo come un singolo processo, anche se ogni occupante della stanza desidera che il proprio voto (input) prevalga.

Invia messaggi dai tuoi dispositivi o processi come input al tuo modello di rilevatore

Abbiamo descritto i diversi modi per inviare un messaggio da un dispositivo o da un processo come input in un AWS IoT Events rilevatore in input. Dopo aver creato gli input e creato il modello del rilevatore, sei pronto per iniziare a inviare dati.

Note

Quando crei un modello di rilevatore o ne aggiorni uno esistente, occorrono alcuni minuti prima che il modello di rilevatore nuovo o aggiornato inizi a ricevere messaggi e a creare rilevatori (istanze). Se il modello di rilevatore viene aggiornato, durante questo periodo potresti continuare a vedere il comportamento basato sulla versione precedente.

Argomenti

- [Crea un input per acquisire i dati del dispositivo](#)
- [Crea un modello di rilevatore per rappresentare gli stati del dispositivo](#)
- [Inviare messaggi come input a un rilevatore](#)

Crea un input per acquisire i dati del dispositivo

Ad esempio, supponiamo che i dispositivi inviino messaggi con il seguente formato.

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

È possibile creare un input per acquisire i dati e il `motorid` (che identifica il dispositivo specifico che ha inviato il messaggio) utilizzando il comando seguente AWS CLI.

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

Il file `pressureInput.json` contiene quanto segue.

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

Quando create input personalizzati, ricordatevi di raccogliere innanzitutto messaggi di esempio come file JSON dai vostri dispositivi o processi. Puoi usarli per creare un input dalla console o dalla CLI.

Crea un modello di rilevatore per rappresentare gli stati del dispositivo

Nel [Crea un input per acquisire i dati del dispositivo](#), ne hai creato uno input basato su un messaggio che riporta i dati sulla pressione di un motore. Per continuare con l'esempio, ecco un modello di rilevatore che risponde a un evento di sovrappressione in un motore.

Si creano due stati: "Normal" e "Dangerous". Ogni rilevatore (istanza) entra nello stato "Normal" quando viene creato. L'istanza viene creata quando arriva un input con un valore univoco per key "motorid".

Se l'istanza del rilevatore riceve una lettura della pressione pari o superiore a 70, entra nello stato "Dangerous" e invia un messaggio Amazon SNS come avviso. Se le letture della pressione tornano alla normalità (meno di 70) per tre ingressi consecutivi, il rilevatore torna allo stato "Normal" e invia un altro messaggio Amazon SNS come tutto chiaro.

Questo modello di rilevatore di esempio presuppone che tu abbia creato due argomenti Amazon SNS i cui Amazon Resource Names (ARN) sono mostrati nella definizione come e. "targetArn": "arn:aws:sns:us-east-1:123456789012:underPressureAction" "targetArn": "arn:aws:sns:us-east-1:123456789012:pressureClearedAction"

Per ulteriori informazioni, consulta la [Amazon Simple Notification Service Developer Guide](#) e, più specificamente, la documentazione sull'[CreateTopic](#) operazione nell'Amazon Simple Notification Service API Reference.

Questo esempio presuppone inoltre che tu abbia creato un ruolo AWS Identity and Access Management (IAM) con le autorizzazioni appropriate. L'ARN di questo ruolo è mostrato nella definizione del modello di rivelatore come. "roleArn": "arn:aws:iam::123456789012:role/

IoTEventsRole" Segui i passaggi [Configurazione delle autorizzazioni per AWS IoT Events](#) per creare questo ruolo e copia l'ARN del ruolo nella posizione appropriata nella definizione del modello del rilevatore.

È possibile creare il modello del rilevatore utilizzando il seguente comando. AWS CLI

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

Il file "motorDetectorModel.json" contiene quanto segue.

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

```

        }
      }
    ],
    "nextState": "Dangerous"
  }
]
}
},
{
  "stateName": "Dangerous",
  "onEnter": {
    "events": [
      {
        "eventName": "Pressure Threshold Breached",
        "condition": "$variable.pressureThresholdBreachd > 1",
        "actions": [
          {
            "sns": {
              "targetArn": "arn:aws:sns:us-
east-1:123456789012:underPressureAction"
            }
          }
        ]
      }
    ]
  },
  "onInput": {
    "events": [
      {
        "eventName": "Overpressurized",
        "condition": "$input.PressureInput.sensorData.pressure > 70",
        "actions": [
          {
            "setVariable": {
              "variableName": "pressureThresholdBreachd",
              "value": "3"
            }
          }
        ]
      }
    ]
  },
  {
    "eventName": "Pressure Okay",
    "condition": "$input.PressureInput.sensorData.pressure <= 70",
    "actions": [

```

```

        {
            "setVariable": {
                "variableName": "pressureThresholdBreached",
                "value": "$variable.pressureThresholdBreached - 1"
            }
        }
    ],
    "transitionEvents": [
        {
            "eventName": "BackToNormal",
            "condition": "$input.PressureInput.sensorData.pressure <= 70 &&
$variable.pressureThresholdBreached <= 1",
            "nextState": "Normal"
        }
    ],
    "onExit": {
        "events": [
            {
                "eventName": "Normal Pressure Restored",
                "condition": "true",
                "actions": [
                    {
                        "sns": {
                            "targetArn": "arn:aws:sns:us-
east-1:123456789012:pressureClearedAction"
                        }
                    }
                ]
            }
        ]
    }
},
"initialStateName": "Normal"
},
"key" : "motorid",
"roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole"
}

```

Inviare messaggi come input a un rilevatore

Ora hai definito un input che identifica i campi importanti nei messaggi inviati da un dispositivo (vedi). [Crea un input per acquisire i dati del dispositivo](#) Nella sezione precedente, ne avete creato uno `detector model` che risponde a un evento di sovrappressione in un motore (vedete). [Crea un modello di rilevatore per rappresentare gli stati del dispositivo](#)

Per completare l'esempio, inviate messaggi da un dispositivo (in questo caso un computer su cui è AWS CLI installato il dispositivo) come input al rilevatore.

Note

Quando create un modello di rilevatore o ne aggiornate uno esistente, occorrono alcuni minuti prima che il modello di rilevatore nuovo o aggiornato inizi a ricevere messaggi e a creare rilevatori (istanze). Se aggiorni il modello del rilevatore, durante questo periodo potresti continuare a vedere il comportamento basato sulla versione precedente.

Utilizzate il AWS CLI comando seguente per inviare un messaggio con dati che superano la soglia.

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

Il file "highPressureMessage.json" contiene quanto segue.

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

È necessario modificare il valore `messageId` in ogni messaggio inviato. Se non lo modifichi, il AWS IoT Events sistema deduplica i messaggi. AWS IoT Events ignora un messaggio se contiene lo `messageID` stesso messaggio di un altro messaggio inviato negli ultimi cinque minuti.

A questo punto, viene creato un rilevatore (istanza) per monitorare gli eventi del motore. "Fulton-A32" Questo rilevatore entra "Normal" nello stato al momento della creazione. Ma poiché abbiamo inviato un valore di pressione superiore alla soglia, passa immediatamente allo "Dangerous" stato. Così facendo, il rilevatore invia un messaggio all'endpoint Amazon SNS il cui ARN è. `arn:aws:sns:us-east-1:123456789012:underPressureAction`

Esegui il AWS CLI comando seguente per inviare un messaggio con dati inferiori alla soglia di pressione.

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

Il file `normalPressureMessage.json` contiene quanto segue.

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

È necessario modificare il `messageId` file ogni volta che si richiama il `BatchPutMessage` comando entro un periodo di cinque minuti. Inviare il messaggio altre due volte. Dopo che il messaggio è stato inviato tre volte, il rilevatore (istanza) del motore "Fulton-A32" invia un messaggio all'"`arn:aws:sns:us-east-1:123456789012:pressureClearedAction`" endpoint Amazon SNS e rientra nello stato. "Normal"

Note

Puoi inviare più messaggi contemporaneamente con `BatchPutMessage`. Tuttavia, l'ordine in cui questi messaggi vengono elaborati non è garantito. Per garantire che i messaggi (input) vengano elaborati in ordine, inviateli uno alla volta e aspettate una risposta corretta ogni volta che viene chiamata l'API.

Di seguito sono riportati alcuni esempi di payload di messaggi SNS creati dal modello di rilevatore, descritto in questa sezione.

sull'evento «Pressure Threshold Breached»

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

sull'evento «Normal Pressure Restored»

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



```
    "messageId": "00004",
    "triggerType": "Message"
  },
  "state": {
    "stateName": "Dangerous",
    "variables": {
      "pressureThresholdBreached": 0
    },
    "timers": {}
  }
},
"eventName": "Normal Pressure Restored"
}
```

Se hai definito dei timer, il loro stato attuale viene mostrato anche nei payload dei messaggi SNS.

I payload dei messaggi contengono informazioni sullo stato del rilevatore (istanza) al momento dell'invio del messaggio (ovvero al momento dell'esecuzione dell'azione SNS). È possibile utilizzare l'https://docs.aws.amazon.com/iotevents/latest/apireference/API_iotevents-data_DescribeDetector.html operazione per ottenere informazioni simili sullo stato del rilevatore.

Restrizioni e limitazioni del modello di rilevatore

I seguenti aspetti sono importanti da considerare quando si crea un modello di rilevatore.

Come usare il campo **actions**

Il **actions** campo è un elenco di oggetti. È possibile avere più di un oggetto, ma è consentita una sola azione in ogni oggetto.

Example

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

```

        "variableName": "temperatureIsTooHigh",
        "value": "$variable.temperatureIsTooHigh - 1"
      }
    }
  ]

```

Come usare il **condition** campo

Il **condition** campo è obbligatorio per gli altri casi **transitionEvents** ed è facoltativo.

Se il **condition** campo non è presente, è equivalente a `"condition": true`.

Il risultato della valutazione di un'espressione condizionale deve essere un valore booleano. Se il risultato non è un valore booleano, è equivalente `false` e non avvierà la transizione **actions** o verso il valore specificato nell'**nextState** evento.

Disponibilità di valori variabili

Per impostazione predefinita, se il valore di una variabile è impostato in un evento, il nuovo valore non è disponibile o non viene utilizzato per valutare le condizioni di altri eventi dello stesso gruppo. Il nuovo valore non è disponibile o non è utilizzato in una condizione di evento nello stesso **onInput** **onExit** campo **onEnter** o.

Imposta il **evaluationMethod** parametro nella definizione del modello del rilevatore per modificare questo comportamento. Quando **evaluationMethod** è impostato su **SERIAL**, le variabili vengono aggiornate e le condizioni degli eventi vengono valutate nell'ordine in cui gli eventi sono definiti. Altrimenti, quando è impostato **BATCH** o **evaluationMethod** è impostato come predefinito, le variabili all'interno di uno stato vengono aggiornate e gli eventi all'interno di uno stato vengono eseguiti solo dopo aver valutato tutte le condizioni dell'evento.

"Dangerous" Nello stato, sul **onInput** campo, `"$variable.pressureThresholdBreached"` viene diminuito di uno nel "Pressure Okay" caso in cui la condizione sia soddisfatta (quando la pressione in ingresso corrente è inferiore o uguale a 70).

```

{
  "eventName": "Pressure Okay",
  "condition": "$input.PressureInput.sensorData.pressure <= 70",
  "actions": [
    {
      "setVariable": {

```

```

        "variableName": "pressureThresholdBreached",
        "value": "$variable.pressureThresholdBreached - 1"
    }
}
]
}

```

Il rilevatore dovrebbe tornare allo "Normal" stato quando "\$variable.pressureThresholdBreached" raggiunge lo 0 (ovvero quando il rilevatore ha ricevuto tre letture di pressione contigue inferiori o uguali a 70). L'"BackToNormal" evento in transitionEvents deve verificare che "\$variable.pressureThresholdBreached" sia minore o uguale a 1 (non a 0) e inoltre verificare nuovamente che il valore corrente fornito da "\$input.PressureInput.sensorData.pressure" sia inferiore o uguale a 70.

```

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

```

Altrimenti, se la condizione verifica solo il valore della variabile, due letture normali seguite da una lettura di sovrappressione soddisferebbero la condizione e tornerebbero allo stato. "Normal" La condizione sta esaminando il valore che "\$variable.pressureThresholdBreached" è stato dato durante la precedente elaborazione di un input. Il valore della variabile viene ripristinato a 3 nell'"Overpressurized" evento, ma ricordate che questo nuovo valore non è ancora disponibile per nessuno condition.

Per impostazione predefinita, ogni volta che un controllo entra nel onInput campo, a condition può vedere il valore di una variabile solo com'era all'inizio dell'elaborazione dell'input, prima che venga modificato dalle azioni specificate in onInput. Lo stesso vale per onEnter e onExit. Qualsiasi modifica apportata a una variabile quando entriamo o usciamo dallo stato non è disponibile per altre condizioni specificate nella stessa onEnter o onExit nei campi.

Latenza durante l'aggiornamento di un modello di rilevatore

Se si aggiorna, si elimina e si ricrea un modello di rilevatore (vedi [UpdateDetectorModel](#)), c'è un certo ritardo prima che tutti i rilevatori (istanze) generati vengano eliminati e il nuovo modello

venga utilizzato per ricreare i rilevatori. Vengono ricreati dopo l'entrata in vigore del nuovo modello di rilevatore e l'arrivo di nuovi input. Durante questo periodo gli input potrebbero continuare a essere elaborati dai rilevatori generati dalla versione precedente del modello di rilevatore. Durante questo periodo, potreste continuare a ricevere gli avvisi definiti dal modello di rilevatore precedente.

Spazi nei tasti di input

Gli spazi sono consentiti nelle chiavi di input, ma i riferimenti alla chiave devono essere racchiusi tra parentesi inverse, sia nella definizione dell'attributo di input che quando il valore della chiave è referenziato in un'espressione. Ad esempio, dato un payload di messaggi come il seguente:

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

Utilizzate quanto segue per definire l'input.

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

In un'espressione condizionale, è necessario fare riferimento al valore di qualsiasi chiave di questo tipo utilizzando anche i backtick.

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

Un esempio commentato: controllo della temperatura HVAC

Alcuni dei seguenti file JSON di esempio contengono commenti in linea, il che li rende JSON non validi. Le versioni complete di questi esempi, senza commenti, sono disponibili all'indirizzo. [Controllo della temperatura HVAC](#)

Contesto

Questo esempio implementa un modello di controllo del termostato che consente di effettuare le seguenti operazioni.

- Definite un solo modello di rilevatore che può essere utilizzato per monitorare e controllare più aree. Viene creata un'istanza del rilevatore per ogni area.
- Inserisci i dati di temperatura da più sensori in ogni area di controllo.
- Modifica il punto di impostazione della temperatura per un'area.
- Imposta i parametri operativi per ogni area e ripristina questi parametri mentre l'istanza è in uso.
- Aggiungi o elimina dinamicamente sensori da un'area.
- Specificate un'autonomia minima per proteggere le unità di riscaldamento e raffreddamento.
- Rifiuta le letture anomale del sensore.
- Definisci i set point di emergenza che attivano immediatamente il riscaldamento o il raffreddamento se un sensore riporta una temperatura superiore o inferiore a una determinata soglia.
- Segnala letture anomale e picchi di temperatura.

Definizioni di input

Vogliamo creare un modello di rilevatore che possiamo utilizzare per monitorare e controllare la temperatura in diverse aree. Ogni area può avere diversi sensori che segnalano la temperatura. Partiamo dal presupposto che ogni area sia servita da un'unità di riscaldamento e un'unità di raffreddamento che possono essere accese o spente per controllare la temperatura nell'area. Ogni area è controllata da un'istanza del rilevatore.

Poiché le diverse aree che monitoriamo e controlliamo possono avere caratteristiche diverse che richiedono parametri di controllo diversi, definiamo e forniamo tali parametri per ciascuna area. 'seedTemperatureInput' Quando inviamo uno di questi messaggi di input a AWS IoT Events, viene creata una nuova istanza del modello di rilevatore con i parametri che vogliamo utilizzare in quell'area. Ecco la definizione di quell'input.

Comando CLI:

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

File: 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" }
    ]
  }
}
```

Risposta:

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

Note

- Viene creata una nuova istanza del rilevatore per ogni messaggio univoco 'areaId' ricevuto. Vedi il 'key' campo nella 'areaDetectorModel' definizione.
- La temperatura media può variare di tanto in 'desiredTemperature' tanto 'allowedError' prima che le unità di riscaldamento o raffreddamento vengano attivate nell'area.
- Se un sensore riporta una temperatura superiore a quella 'rangeHigh', il rilevatore segnala un picco e avvia immediatamente l'unità di raffreddamento.
- Se un sensore riporta una temperatura inferiore a 'rangeLow', il rilevatore segnala un picco e avvia immediatamente l'unità di riscaldamento.
- Se un sensore riporta una temperatura superiore 'anomalousHigh' o inferiore a 'anomalousLow', il rilevatore segnala una lettura anomala del sensore, ma ignora la lettura della temperatura riportata.
- 'sensorCount' Indica al rilevatore quanti sensori rilevano i dati relativi all'area. Il rilevatore calcola la temperatura media dell'area assegnando il fattore di peso appropriato a ciascuna lettura della temperatura che riceve. Per questo motivo, il rilevatore non dovrà tenere traccia di ciò che ogni sensore riporta e il numero di sensori può essere modificato dinamicamente, in base alle esigenze. Tuttavia, se un singolo sensore va offline, il rilevatore non lo saprà né lo terrà conto. Ti consigliamo di creare un altro modello di rilevatore specifico per monitorare lo stato della connessione di ciascun sensore. La presenza di due modelli di rilevatori complementari semplifica la progettazione di entrambi.
- Il 'noDelay' valore può essere true o false. Dopo l'accensione, un'unità di riscaldamento o raffreddamento deve rimanere accesa per un certo periodo di tempo minimo per proteggere l'integrità dell'unità e prolungarne la durata operativa. Se 'noDelay' è impostato su false, l'istanza del rilevatore impone un ritardo prima di spegnere le unità di raffreddamento e riscaldamento, per garantire che funzionino per il tempo minimo. Il numero di secondi di ritardo è stato codificato nella definizione del modello di rilevatore perché non siamo in grado di utilizzare un valore variabile per impostare un timer.

'temperatureInput' Viene utilizzato per trasmettere i dati del sensore a un'istanza del rilevatore.

Comando CLI:

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

File: temperatureInput.json

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

Risposta:

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

Note

- 'sensorId' Non viene utilizzato da un'istanza di rilevatore di esempio per controllare o monitorare direttamente un sensore. Viene automaticamente passato alle notifiche inviate dall'istanza del rilevatore. Da lì, può essere utilizzato per identificare i sensori che non funzionano (ad esempio, un sensore che invia regolarmente letture anomale potrebbe essere sul punto di guastarsi) o che sono andati offline (quando viene utilizzato come input per un modello di rilevatore aggiuntivo che monitora il battito cardiaco del dispositivo). Inoltre, 'sensorId' può aiutare a identificare le zone calde o fredde di un'area se le letture differiscono regolarmente dalla media.
- 'areaId' Viene utilizzato per indirizzare i dati del sensore all'istanza del rilevatore appropriata. Viene creata un'istanza del rilevatore per ogni messaggio univoco 'areaId' ricevuto. Vedi il 'key' campo nella 'areaDetectorModel' definizione.

Definizione del modello di rilevatore

L'areaDetectorModel esempio contiene commenti in linea.

Comando CLI:

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

File: 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"
}

```

Risposta:

```

{
  "detectorModelConfiguration": {
    "status": "ACTIVATING",
    "lastUpdateTime": 1557523491.168,
    "roleArn": "arn:aws:iam::123456789012:role/IoTEventsRole",
  }
}

```

```
    "creationTime": 1557523491.168,  
    "detectorModelArn": "arn:aws:iotevents:us-west-2:123456789012:detectorModel/  
areaDetectorModel",  
    "key": "areaId",  
    "detectorModelName": "areaDetectorModel",  
    "detectorModelVersion": "1"  
  }  
}
```

BatchUpdateDetector esempio

È possibile utilizzare l'BatchUpdateDetector operazione per mettere un'istanza del rilevatore in uno stato noto, inclusi i valori del timer e delle variabili. Nell'esempio seguente, l'BatchUpdateDetector operazione ripristina i parametri operativi per un'area sottoposta a monitoraggio e controllo della temperatura. Questa operazione consente di eseguire questa operazione senza dover eliminare, ricreare o aggiornare il modello del rilevatore.

Comando CLI:

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

File: 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": [
    ]
}
]
}
]
}

```

Risposta:

```

{
  "batchUpdateDetectorErrorEntries": []
}

```

BatchPutMessageesempi

Example 1

Utilizzate l'BatchPutMessageoperazione per inviare un "seedTemperatureInput" messaggio che imposta i parametri operativi per una determinata area sottoposta a controllo e monitoraggio della temperatura. Qualsiasi messaggio ricevuto in AWS IoT Events quell'area presenta un nuovo messaggio "areaId" causa la creazione di una nuova istanza del rilevatore. Tuttavia, la nuova istanza del rilevatore non cambierà stato "idle" e non inizierà a monitorare la temperatura e a controllare le unità di riscaldamento o raffreddamento fino a quando non verrà ricevuto un "seedTemperatureInput" messaggio relativo alla nuova area.

Comando CLI:

```

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

```

File: 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}"  
  }  
]
```

Risposta:

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

Example

2

Utilizzate l'BatchPutMessageoperazione per inviare un "temperatureInput" messaggio per riportare i dati del sensore di temperatura per un sensore in una determinata area di controllo e monitoraggio.

Comando CLI:

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

File: temperatureExample.json

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



```
}
```

Risposta:

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

Example 3

Utilizzare l'BatchPutMessageoperazione per inviare un "seedTemperatureInput" messaggio per modificare il valore della temperatura desiderata per una determinata area.

Comando CLI:

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

File: seedSetDesiredTemp.json

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

Risposta:

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

Esempio: importazione di messaggi MQTT

Se le risorse di elaborazione dei sensori non possono utilizzare l'"BatchPutMessage"API, ma possono inviare i dati al broker di AWS IoT Core messaggi utilizzando un client MQTT leggero, puoi

creare una regola AWS IoT Core tematica per reindirizzare i dati dei messaggi a un input. AWS IoT Events Di seguito è riportata una definizione di una regola AWS IoT Events tematica che prende i campi "areaId" e "sensorId" di input dall'argomento MQTT e il "sensorData.temperature" campo dal campo del payload del messaggio e inserisce questi dati "temp" nel nostro. AWS IoT Events "temperatureInput"

Se le risorse di elaborazione dei sensori non possono utilizzare l'"BatchPutMessage"API, ma possono inviare i dati al broker di AWS IoT Core messaggi utilizzando un client MQTT leggero, puoi creare una regola AWS IoT Core tematica per reindirizzare i dati dei messaggi a un input. AWS IoT Events Di seguito è riportata una definizione di una regola AWS IoT Events tematica che prende i campi "areaId" e "sensorId" di input dall'argomento MQTT e il "sensorData.temperature" campo dal campo del payload del messaggio e inserisce questi dati "temp" nel nostro. AWS IoT Events "temperatureInput"

Comando CLI:

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

File: seedSetDesiredTemp.json

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

Risposta: [nessuna]

Se il sensore invia un messaggio sull'argomento "update/temperature/Area51/03" con il seguente payload.

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

Ciò comporta l'inserimento dei dati AWS IoT Events come se fosse stata effettuata la seguente chiamata "BatchPutMessage" API.

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

File: spooferExample.json

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

Esempi: messaggi Amazon SNS generati

Di seguito sono riportati alcuni esempi di messaggi SNS generati dall'istanza del "Area51" rilevatore.

```
Heating system off command> {
  "eventTime":1557520274729,
  "payload":{
    "actionExecutionId":"f3159081-bac3-38a4-96f7-74af0940d0a4",
    "detector":{
      "detectorModelName":"areaDetectorModel","keyValue":"Area51","detectorModelVersion":"1"},
    "inputName":"seedTemperatureInput","messageId":"00001","triggerType":"Message","state":{
      "stateName":"start","variables":{
        "sensorCount":10,"rangeHigh":30.0,"resetMe":false,"enteringNewState":true,"averageTemperature":{}}},
    "eventName":"resetHeatCool"
  }
}
```

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

Esempio: API DescribeDetector

È possibile utilizzare l'DescribeDetector operazione per visualizzare lo stato corrente, i valori delle variabili e i timer per un'istanza del rilevatore.

Comando CLI:

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

Risposta:

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

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

```

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

```

AWS IoT Core esempi di motori di regole

Le seguenti regole ripubblicano i messaggi AWS IoT Core MQTT come messaggi di richiesta di aggiornamento shadow. Partiamo dal presupposto che AWS IoT Core le cose siano definite per un'unità di riscaldamento e un'unità di raffreddamento per ogni area controllata dal modello di rilevatore. In questo esempio, abbiamo definito cose denominate "Area51HeatingUnit" e "Area51CoolingUnit".

Comando CLI:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowCool0ffRule.json
```

File: 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"
        }
      }
    ]
  }
}

```

```
    }
  }
]
}
}
```

Risposta: [vuoto]

Comando CLI:

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

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

Risposta: [vuoto]

Comando CLI:

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

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

Risposta: [vuoto]

Comando CLI:

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

File: 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": [
      {
        "republsh": {
```



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

Risposta: [vuoto]

Azioni supportate

AWS IoT Events può attivare azioni quando rileva un evento o un evento di transizione specifico. È possibile definire azioni integrate per utilizzare un timer o impostare una variabile o inviare dati ad altre AWS risorse.

Note

Quando si definisce un'azione in un modello di rilevatore, è possibile utilizzare espressioni per parametri che sono di tipo stringa. Per ulteriori informazioni, consulta [Espressioni](#).

AWS IoT Events supporta le seguenti azioni che consentono di utilizzare un timer o impostare una variabile:

- [setTimer](#) per creare un timer.
- [resetTimer](#) per resettare il timer.
- [clearTimer](#) per eliminare il timer.
- [setVariable](#) per creare una variabile.

AWS IoT Events supporta le seguenti azioni che consentono di lavorare con AWS i servizi:

- [iotTopicPublish](#) per pubblicare un messaggio su un argomento MQTT.
- [iotEvents](#) a cui inviare dati AWS IoT Events come valore di input.
- [iotSiteWise](#) per inviare i dati a una proprietà di asset in AWS IoT SiteWise.
- [dynamoDB](#) per inviare dati a una tabella Amazon DynamoDB.
- [dynamoDBv2](#) per inviare dati a una tabella Amazon DynamoDB.
- [firehose](#) per inviare dati a uno stream Amazon Data Firehose.
- [lambda](#) per richiamare una AWS Lambda funzione.
- [sns](#) per inviare dati come notifica push.
- [sqs](#) per inviare dati a una coda Amazon SQS.

Utilizzo di azioni integrate

AWS IoT Events supporta le seguenti azioni che consentono di utilizzare un timer o impostare una variabile:

- [setTimer](#) per creare un timer.
- [resetTimer](#) per resettare il timer.
- [clearTimer](#) per eliminare il timer.
- [setVariable](#) per creare una variabile.

Imposta l'azione del timer

Set timer action

L'azione `setTimer` consente di creare un timer con durata in secondi.

More information (2)

Quando si crea un timer, è necessario specificare i seguenti parametri.

timerName

Il nome del timer.

durationExpression

(Facoltativo) La durata del timer, in secondi.

Il risultato valutato di un'espressione di durata viene arrotondato per difetto al numero intero più vicino. Ad esempio, se si imposta il timer su 60,99 secondi, il risultato valutato dell'espressione di durata è 60 secondi.

Per ulteriori informazioni, consulta [SetTimerAction](#) nella documentazione di riferimento dell'API AWS IoT Events .

Reimposta l'azione del timer

Reset timer action

L'`resetTimer` azione consente di impostare il timer sul risultato precedentemente valutato dell'espressione di durata.

More information (1)

Quando si reimposta un timer, è necessario specificare il seguente parametro.

timerName

Il nome del timer.

AWS IoT Events non rivaluta l'espressione della durata quando si reimposta il timer.

Per ulteriori informazioni, consulta [ResetTimerAction](#) nella documentazione di riferimento dell'API AWS IoT Events .

Cancella l'azione del timer

Clear timer action

L'`clearTimer` azione consente di eliminare un timer esistente.

More information (1)

Quando si elimina un timer, è necessario specificare il seguente parametro.

timerName

Il nome del timer.

Per ulteriori informazioni, consulta [ClearTimerAction](#) nella documentazione di riferimento dell'API AWS IoT Events .

Imposta un'azione variabile

Set variable action

L'`setVariable` azione consente di creare una variabile con un valore specificato.

More information (2)

Quando si crea una variabile, è necessario specificare i seguenti parametri.

variableName

Il nome della variabile.

value

Il nuovo valore della variabile.

Per ulteriori informazioni, consulta [SetVariableAction](#) nella documentazione di riferimento dell'API AWS IoT Events .

Lavorare con altri AWS servizi

AWS IoT Events supporta le seguenti azioni che consentono di lavorare con AWS i servizi:

- [iotTopicPublish](#) per pubblicare un messaggio su un argomento MQTT.
- [iotEvents](#) a cui inviare dati AWS IoT Events come valore di input.
- [iotSiteWise](#) per inviare i dati a una proprietà di asset in AWS IoT SiteWise.
- [dynamoDB](#) per inviare dati a una tabella Amazon DynamoDB.
- [dynamoDBv2](#) per inviare dati a una tabella Amazon DynamoDB.
- [firehose](#) per inviare dati a uno stream Amazon Data Firehose.
- [lambda](#) per richiamare una AWS Lambda funzione.
- [sns](#) per inviare dati come notifica push.
- [sqs](#) per inviare dati a una coda Amazon SQS.

Important

- Devi scegliere la stessa AWS regione per entrambi AWS IoT Events e i servizi AWS con cui lavorare. Per l'elenco delle regioni supportate, consulta [Endpoint e quote AWS IoT Events](#) in Riferimenti generali di Amazon Web Services.
- È necessario utilizzare la stessa AWS regione quando si creano altre risorse AWS per le azioni AWS IoT Events. Se cambi regione AWS, potresti avere problemi di accesso alle risorse AWS.

Per impostazione predefinita, AWS IoT Events genera un payload standard in JSON per qualsiasi azione. Questo payload di azioni contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello del rilevatore e sull'evento che ha attivato l'azione. Per configurare il payload dell'azione, è possibile utilizzare un'espressione di contenuto. Per ulteriori informazioni, consulta [Espressioni](#) e il tipo di dati [Payload](#) nell'AWS IoT Events API Reference.

AWS IoT Core

IoT topic publish action

L'azione AWS IoT Core consente di pubblicare un messaggio MQTT tramite il broker di AWS IoT messaggi. Per l'elenco delle regioni supportate, consulta [Endpoint e quote AWS IoT Core](#) in Riferimenti generali di Amazon Web Services.

Il broker di AWS IoT messaggi collega AWS IoT i client inviando messaggi dai client di pubblicazione ai client abbonati. Per ulteriori informazioni, consulta [Message broker AWS IoT nella Guida per](#) gli AWS IoT sviluppatori.

More information (2)

Quando si pubblica un messaggio MQTT, è necessario specificare i seguenti parametri.

mqttTopic

L'argomento MQTT che riceve il messaggio.

È possibile definire il nome di un argomento MQTT in modo dinamico in fase di esecuzione utilizzando variabili o valori di input creati nel modello del rilevatore.

payload

(Facoltativo) Il payload predefinito contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello del rilevatore e sull'evento che ha attivato l'azione. È inoltre possibile personalizzare il payload. [Per ulteriori informazioni, consulta Payload nell'API Reference. AWS IoT Events](#)

Note

Assicurati che la policy allegata al tuo ruolo AWS IoT Events di servizio conceda l'`iot:Publish` autorizzazione. Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, consulta [iotTopicPublishAction](#) nella documentazione di riferimento dell'API AWS IoT Events .

AWS IoT Events

IoT Events action

L' AWS IoT Events azione consente di inviare dati a AWS IoT Events come input. Per l'elenco delle regioni supportate, consulta [Endpoint e quote AWS IoT Events](#) in Riferimenti generali di Amazon Web Services.

AWS IoT Events consente di monitorare le apparecchiature o le flotte di dispositivi per individuare guasti o cambiamenti di funzionamento e di attivare azioni quando si verificano tali eventi. Per ulteriori informazioni, consulta [Cos'è? AWS IoT Events](#) nella Guida per gli AWS IoT Events sviluppatori.

More information (2)

Quando si inviano dati a AWS IoT Events, è necessario specificare i seguenti parametri.

inputName

Il nome dell' AWS IoT Events input che riceve i dati.

payload

(Facoltativo) Il payload predefinito contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello del rilevatore e sull'evento che ha attivato l'azione. È inoltre possibile personalizzare il payload. [Per ulteriori informazioni, consulta Payload nell'API Reference. AWS IoT Events](#)

Note

Assicurati che la policy allegata al tuo ruolo AWS IoT Events di servizio conceda `iotevents:BatchPutMessage` autorizzazione. Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, consulta [lotEventsAction](#) nella documentazione di riferimento dell'API AWS IoT Events .

AWS IoT SiteWise

IoT SiteWise action

L' AWS IoT SiteWise azione consente di inviare dati a una proprietà della risorsa in AWS IoT SiteWise. Per l'elenco delle regioni supportate, consulta [Endpoint e quote AWS IoT SiteWise](#) in Riferimenti generali di Amazon Web Services.

AWS IoT SiteWise è un servizio gestito che consente di raccogliere, organizzare e analizzare i dati provenienti da apparecchiature industriali su larga scala. Per ulteriori informazioni, consulta [Che cos'è AWS IoT SiteWise?](#) nella Guida per l'utente di AWS IoT SiteWise .

More information (11)

Quando inviate dati a una proprietà di un asset in AWS IoT SiteWise, dovete specificare i seguenti parametri.

Important

Per ricevere i dati, è necessario utilizzare una proprietà dell'asset esistente in AWS IoT SiteWise.

- Se utilizzate la AWS IoT Events console, dovete specificare di `propertyAlias` identificare la proprietà dell'asset di destinazione.
- Se si utilizza la AWS CLI, è necessario specificare una delle due `propertyAlias` o entrambe `assetId` e `propertyId` identificare la proprietà dell'asset di destinazione.

Per ulteriori informazioni, consulta la sezione [Mappatura dei flussi di dati industriali alle proprietà degli asset](#) nella Guida per l'utente di AWS IoT SiteWise .

propertyAlias

(Facoltativo) L'alias della proprietà dell'asset. È inoltre possibile specificare un'espressione.

assetId

(Facoltativo) L'ID della risorsa che ha la proprietà specificata. È inoltre possibile specificare un'espressione.

propertyId

(Facoltativo) L'ID della proprietà dell'asset. È inoltre possibile specificare un'espressione.

entryId

(Facoltativo) Un identificatore univoco per questa voce. È possibile utilizzare l'ID voce per tenere traccia dell'immissione di dati che causa un errore in caso di errore. Il valore predefinito è un nuovo identificatore univoco. È inoltre possibile specificare un'espressione.

propertyValue

Una struttura che contiene dettagli sul valore della proprietà.

quality

(Facoltativo) La qualità del valore della proprietà dell'asset. Il valore deve essere GOOD, BAD o UNCERTAIN. È inoltre possibile specificare un'espressione.

timestamp

(Facoltativo) Una struttura che contiene informazioni sul timestamp. Se non si specifica questo valore, l'impostazione predefinita è l'ora dell'evento.

timeInSeconds

Il timestamp, in secondi, nel formato epoch Unix. L'intervallo valido è 1-31556889864403199. È inoltre possibile specificare un'espressione.

offsetInNanos

(Facoltativo) L'offset in nanosecondi convertito da `timeInSeconds`. L'intervallo valido è 0-999999999. È inoltre possibile specificare un'espressione.

value

Struttura che contiene un valore di proprietà di un asset.

Important

È necessario specificare uno dei seguenti tipi di valore, a seconda del `dataType` della proprietà asset specificata. Per ulteriori informazioni, consulta [AssetProperty](#) nella documentazione di riferimento dell'API AWS IoT SiteWise .

booleanValue

(Facoltativo) Il valore della proprietà dell'asset è un valore booleano che deve essere o. TRUE FALSE È inoltre possibile specificare un'espressione. Se si utilizza un'espressione, il risultato valutato deve essere un valore booleano.

doubleValue

(Facoltativo) Il valore della proprietà dell'asset è doppio. È inoltre possibile specificare un'espressione. Se si utilizza un'espressione, il risultato valutato deve essere doppio.

integerValue

(Facoltativo) Il valore della proprietà dell'asset è un numero intero. È inoltre possibile specificare un'espressione. Se si utilizza un'espressione, il risultato valutato deve essere un numero intero.

stringValue

(Facoltativo) Il valore della proprietà dell'asset è una stringa. È inoltre possibile specificare un'espressione. Se si utilizza un'espressione, il risultato valutato deve essere una stringa.

Note

Assicurati che la politica allegata al tuo ruolo AWS IoT Events di servizio conceda l'`iotsitewise:BatchPutAssetPropertyValue` autorizzazione. Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, consulta [lotSiteWiseAction](#) nella documentazione di riferimento dell'API AWS IoT Events .

Amazon DynamoDB

DynamoDB action

L'azione Amazon DynamoDB consente di inviare dati a una tabella DynamoDB. Una colonna della tabella DynamoDB riceve tutte le coppie attributo-valore nel payload dell'azione specificato. Per l'elenco delle regioni supportate, consulta gli [endpoint e le quote di Amazon DynamoDB](#) nel. Riferimenti generali di Amazon Web Services

Amazon DynamoDB è un servizio di database NoSQL interamente gestito che combina prestazioni elevate e prevedibili con una scalabilità ottimale. Per ulteriori informazioni, consulta [Cos'è DynamoDB?](#) nella Amazon DynamoDB Developer Guide.

More information (10)

Quando si inviano dati a una colonna di una tabella DynamoDB, è necessario specificare i seguenti parametri.

tableName

Il nome della tabella DynamoDB che riceve i dati. Il `tableName` valore deve corrispondere al nome della tabella DynamoDB. È inoltre possibile specificare un'espressione.

hashKeyField

Il nome della chiave hash (chiamata anche chiave di partizione). Il `hashKeyField` valore deve corrispondere alla chiave di partizione della tabella DynamoDB. È inoltre possibile specificare un'espressione.

hashKeyType

(Facoltativo) Il tipo di dati della chiave hash. Il valore del tipo di chiave hash deve essere `STRING` o `NUMBER`. Il valore predefinito è `STRING`. È inoltre possibile specificare un'espressione.

hashKeyValue

Valore della chiave hash. `hashKeyValue` Utilizza modelli sostitutivi. Questi modelli offrono i dati in fase di runtime. È inoltre possibile specificare un'espressione.

rangeKeyField

(Facoltativo) Nome della chiave di intervallo (detta anche chiave di ordinamento). Il `rangeKeyField` valore deve corrispondere alla chiave di ordinamento della tabella DynamoDB. È inoltre possibile specificare un'espressione.

rangeKeyType

(Facoltativo) Il tipo di dati della chiave di intervallo. Il valore del tipo di chiave hash deve essere `STRING` o `NUMBER`. Il valore predefinito è `STRING`. È inoltre possibile specificare un'espressione.

rangeKeyValue

(Facoltativo) Valore della chiave di intervallo. `rangeKeyValue` Utilizza modelli sostitutivi. Questi modelli offrono i dati in fase di runtime. È inoltre possibile specificare un'espressione.

operation

(Facoltativo) Il tipo di operazione da eseguire. È inoltre possibile specificare un'espressione. Il valore dell'operazione deve essere uno dei seguenti valori:

- `INSERT`: inserimento di dati come nuovo elemento nella tabella DynamoDB. Si tratta del valore di default.
- `UPDATE`: aggiornare un elemento esistente della tabella DynamoDB con nuovi dati.
- `DELETE`- Eliminare un elemento esistente dalla tabella DynamoDB.

payloadField

(Facoltativo) Il nome della colonna DynamoDB che riceve il payload dell'azione. Il nome predefinito è `payload`. È inoltre possibile specificare un'espressione.

payload

(Facoltativo) Il payload predefinito contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello di rilevatore e sull'evento che ha attivato l'azione. È inoltre possibile personalizzare il payload. [Per ulteriori informazioni, consulta Payload nell'API Reference. AWS IoT Events](#)

Se il tipo di payload specificato è una stringa, DynamoDBAction invia dati non JSON alla tabella DynamoDB come dati binari. La console DynamoDB mostra i dati come testo con codifica Base64. Il valore di payloadField è *payload-field_raw*. È inoltre possibile specificare un'espressione.

Note

Assicurati che la policy allegata al tuo ruolo di AWS IoT Events servizio conceda l'autorizzazione. dynamodb:PutItem Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, vedere [DynamodbAction](#) nell'API Reference. AWS IoT Events

Amazon DynamoDB (versione 2)

DynamoDBv2 action

L'azione Amazon DynamoDB (v2) consente di scrivere dati su una tabella DynamoDB. Una colonna separata della tabella DynamoDB riceve una coppia attributo-valore nel payload dell'azione specificato. Per l'elenco delle regioni supportate, consulta gli [endpoint e le quote di Amazon DynamoDB](#) nel. Riferimenti generali di Amazon Web Services

Amazon DynamoDB è un servizio di database NoSQL interamente gestito che combina prestazioni elevate e prevedibili con una scalabilità ottimale. Per ulteriori informazioni, consulta [Cos'è DynamoDB?](#) nella Amazon DynamoDB Developer Guide.

More information (2)


Quando invii dati a più colonne di una tabella DynamoDB, devi specificare i seguenti parametri.

tableName


Il nome della tabella DynamoDB che riceve i dati. È inoltre possibile specificare un'espressione.

payload

(Facoltativo) Il payload predefinito contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello del rilevatore e sull'evento che ha attivato l'azione. È inoltre possibile personalizzare il payload. [Per ulteriori informazioni, consulta Payload nell'API Reference. AWS IoT Events](#)

 **Important**

Il tipo di payload deve essere JSON. È inoltre possibile specificare un'espressione.

 **Note**

Assicurati che la policy allegata al tuo ruolo di AWS IoT Events servizio conceda l'autorizzazione. `dynamodb:PutItem` Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, consulta [DynamoDBV2Action](#) nell'API Reference. AWS IoT Events

Amazon Data Firehose

Firehose action

L'azione Amazon Data Firehose consente di inviare dati a un flusso di distribuzione Firehose. Per l'elenco delle regioni supportate, consulta gli [endpoint e le quote di Amazon Data Firehose](#) nel. Riferimenti generali di Amazon Web Services

Amazon Data Firehose è un servizio completamente gestito per la distribuzione di dati di streaming in tempo reale a destinazioni come Amazon Simple Storage Service (Amazon Simple Storage Service), Amazon Redshift, OpenSearch Amazon OpenSearch Service (Service) e Splunk. Per ulteriori informazioni, consulta [What is Amazon Data Firehose?](#) nella Amazon Data Firehose Developer Guide.

More information (3)

Quando si inviano dati a un flusso di distribuzione Firehose, è necessario specificare i seguenti parametri.

deliveryStreamName

Il nome del flusso di distribuzione Firehose che riceve i dati.

separator

(Facoltativo) È possibile utilizzare un separatore di caratteri per separare i dati continui inviati al flusso di distribuzione di Firehose. Il valore del separatore deve essere '\n' (nuova riga), '\t' (tab), '\r\n' (nuova riga di Windows) o ',' (virgola).

payload

(Facoltativo) Il payload predefinito contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello del rilevatore e sull'evento che ha attivato l'azione. È inoltre possibile personalizzare il payload. [Per ulteriori informazioni, consulta Payload nell'API Reference. AWS IoT Events](#)

Note

Assicurati che la policy allegata al tuo ruolo AWS IoT Events di servizio conceda l'`firehose:PutRecord` autorizzazione. Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, consulta [FirehoseAction](#) nella documentazione di riferimento dell'API AWS IoT Events .

AWS Lambda

Lambda action

L' AWS Lambda azione consente di chiamare una funzione Lambda. Per l'elenco delle regioni supportate, consulta [Endpoint e quote AWS Lambda](#) in Riferimenti generali di Amazon Web Services.

AWS Lambda è un servizio di elaborazione che consente di eseguire codice senza effettuare il provisioning o la gestione di server. Per ulteriori informazioni, consulta [Cos'è? AWS Lambda](#) nella Guida per gli AWS Lambda sviluppatori.

More information (2)

Quando si chiama una funzione Lambda, è necessario specificare i seguenti parametri.

functionArn

ARN della funzione Lambda da chiamare.

payload

(Facoltativo) Il payload predefinito contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello del rilevatore e sull'evento che ha attivato l'azione. È inoltre possibile personalizzare il payload. [Per ulteriori informazioni, consulta Payload nell'API Reference. AWS IoT Events](#)

Note

Assicurati che la policy allegata al tuo ruolo AWS IoT Events di servizio conceda l'`lambda:InvokeFunction` autorizzazione. Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, consulta [LambdaAction](#) nella documentazione di riferimento dell'API AWS IoT Events .


Amazon Simple Notification Service

SNS action

L'azione di pubblicazione dell'argomento Amazon SNS consente di pubblicare un messaggio Amazon SNS. Per l'elenco delle regioni supportate, consulta gli [endpoint e le quote di Amazon Simple Notification Service](#) nel. Riferimenti generali di Amazon Web Services

Amazon Simple Notification Service (Amazon Simple Notification Service) è un servizio Web che coordina e gestisce la consegna o l'invio di messaggi a endpoint o client abbonati. Per ulteriori

informazioni, consulta [Cos'è Amazon SNS?](#) nella Guida per gli sviluppatori di Amazon Simple Notification Service.

 Note

L'azione di pubblicazione dell'argomento Amazon SNS non supporta gli argomenti [Amazon SNS FIFO \(first in, first out\)](#). Poiché il motore delle regole è un servizio completamente distribuito, i messaggi potrebbero non essere visualizzati in un ordine specifico quando viene avviata l'azione Amazon SNS.

More information (2)


Quando pubblichi un messaggio Amazon SNS, devi specificare i seguenti parametri.

targetArn

L'ARN del target Amazon SNS che riceve il messaggio.

payload

(Facoltativo) Il payload predefinito contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello del rilevatore e sull'evento che ha attivato l'azione. È inoltre possibile personalizzare il payload. [Per ulteriori informazioni, consulta Payload nell'API Reference. AWS IoT Events](#)

 Note

Assicurati che la policy allegata al tuo ruolo AWS IoT Events di servizio conceda l'`sns:Publish` autorizzazione. Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, consulta [SNS TopicPublishAction](#) nell'AWS IoT Events API Reference.

Amazon Simple Queue Service

SQS action

L'azione Amazon SQS consente di inviare dati a una coda Amazon SQS. Per l'elenco delle regioni supportate, consulta gli [endpoint e le quote di Amazon Simple Queue Service nel](#). Riferimenti generali di Amazon Web Services

Amazon Simple Queue Service (Amazon SQS) offre una coda ospitata internamente sicura, durevole e disponibile che consente di integrare e separare i componenti e i sistemi software distribuiti. Per ulteriori informazioni, consulta [What is Amazon Simple Queue Service](#) nella Amazon Simple Queue Service Developer Guide.

Note

L'azione Amazon SQS non supporta gli argomenti [Amazon SQS FIFO \(first in, first out\)](#). Poiché il motore delle regole è un servizio completamente distribuito, i messaggi potrebbero non essere visualizzati in un ordine specificato quando viene avviata l'azione Amazon SQS.

More information (3)

Quando invii dati a una coda Amazon SQS, devi specificare i seguenti parametri.

queueUrl

L'URL della coda Amazon SQS che riceve i dati.

useBase64

(Facoltativo) AWS IoT Events codifica i dati in testo Base64, se specificato. TRUE Il valore predefinito è FALSE.

payload

(Facoltativo) Il payload predefinito contiene tutte le coppie attributo-valore che contengono le informazioni sull'istanza del modello di rilevatore e sull'evento che ha attivato l'azione. È inoltre possibile personalizzare il payload. [Per ulteriori informazioni, consulta Payload nell'API Reference. AWS IoT Events](#)

Note

Assicurati che la policy allegata al tuo ruolo AWS IoT Events di servizio conceda l'`sqs : SendMessage` autorizzazione. Per ulteriori informazioni, consulta [Gestione delle identità e degli accessi per l'AWS IoT Events](#).

Per ulteriori informazioni, consulta [SNS TopicPublishAction](#) nell'AWS IoT Events API Reference.

Puoi anche utilizzare Amazon SNS e il motore delle AWS IoT Core regole per attivare una AWS Lambda funzione. In questo modo è possibile intraprendere azioni utilizzando altri servizi, come Amazon Connect o persino un'applicazione di pianificazione delle risorse aziendali (ERP).

Note

Per raccogliere ed elaborare grandi flussi di record di dati in tempo reale, puoi utilizzare altri AWS servizi, come [Amazon Kinesis](#). Da lì, puoi completare un'analisi iniziale e quindi inviare i risultati AWS IoT Events come input a un rilevatore.

Espressioni

AWS IoT Events offre diversi modi per specificare i valori durante la creazione e l'aggiornamento dei modelli di rilevatori. È possibile utilizzare le espressioni per specificare valori letterali o AWS IoT Events valutare le espressioni prima di specificare valori particolari.

Sintassi

È possibile utilizzare valori letterali, operatori, funzioni, riferimenti e modelli di sostituzione nelle espressioni. AWS IoT Events

Valori letterali

- Numero intero
- Decimale
- Stringa
- Boolean

Operatori

Unario

- Non (booleano): !
- Non (bit per bit): ~
- Meno (aritmetico): -

Stringa

- Concatenazione: +

Entrambi gli operandi devono essere stringhe. Le stringhe letterali devono essere racchiuse tra virgolette singole (').

Ad esempio: -> 'my' + 'string' 'mystring'

Aritmetica

- Aggiunta: +

Entrambi gli operandi devono essere numerici.

- Sottrazione: -
- Divisione: /

Il risultato della divisione è un valore intero arrotondato a meno che almeno uno degli operandi (divisore o dividendo) sia un valore decimale.

- Moltiplicazione: *

Bit per bit (numero intero)

- OPPURE: |

Ad esempio: $13 | 5 \rightarrow 13$

- E: &

Ad esempio: $13 \& 5 \rightarrow 5$

- XOR: ^


Ad esempio: $\rightarrow 13 \wedge 5 \ 8$

- NON: ~

Ad esempio: $\sim 13 \rightarrow -14$

Boolean

- Meno di: <
- Minore o uguale a: <=
- Uguale a: ==
- Non uguale a: !=
- Maggiore o uguale a: >=
- Maggiore di: >
- E: &&
- OPPURE: ||

 Note

Quando una sottoespressione di || contiene dati non definiti, tale sottoespressione viene trattata come `false`

Parentesi

È possibile utilizzare le parentesi per raggruppare i termini all'interno di un'espressione.

Funzioni

Funzioni integrate

timeout("timer-name")

Valuta `true` se il timer specificato è scaduto. Sostituisci "*timer-name*" con il nome di un timer che hai definito, tra virgolette. In un'azione relativa a un evento, è possibile definire un timer e quindi avviarlo, resettarlo o cancellarne uno definito in precedenza. Vedi il `campodetectorModelDefinition.states.onInput|onEnter|onExit.events.actions.setTimer.timerName`.

Un timer impostato in uno stato può essere referenziato in uno stato diverso. È necessario visitare lo stato in cui è stato creato il timer prima di entrare nello stato a cui fa riferimento il timer.

Ad esempio, un modello di rilevatore ha due stati, `TemperatureChecked` e `RecordUpdated`. Hai creato un timer nello `TemperatureChecked` stato. È necessario visitare lo `TemperatureChecked` stato prima di poter utilizzare il timer nello `RecordUpdated` stato.

Per garantire la precisione, il tempo minimo per impostare un timer è di 60 secondi.

Note

`timeout()` restituisce `true` solo la prima volta che viene controllato dopo la scadenza effettiva del timer e ritorna `false` successivamente.

convert(*type*, *expression*)

Restituisce il valore dell'espressione convertita nel tipo specificato. Il valore del *tipo* deve essere `StringBoolean`, o `Decimal`. Utilizzate una di queste parole chiave o un'espressione che restituisca una stringa contenente la parola chiave. Solo le seguenti conversioni hanno esito positivo e restituiscono un valore valido:

- `Boolean` -> stringa

Restituisce la stringa "true" o "false"

- Decimale -> stringa
- Stringa -> Booleano
- Stringa -> decimale

La stringa specificata deve essere una rappresentazione valida di un numero decimale o ha esito negativo. `convert()`

Se `convert()` non restituisce un valore valido, anche l'espressione di cui fa parte non è valida. Questo risultato è equivalente `false` e non attiverà la transizione `actions` o verso lo `nextState` specificato come parte dell'evento in cui si verifica l'espressione.

isNull(*expression*)

Restituisce `true` se l'espressione restituisce `null`. Ad esempio, se l'input `MyInput` riceve il messaggio `{ "a": null }`, quanto segue restituisce `true`, ma `isUndefined($input.MyInput.a)` restituisce `si. false`

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

isUndefined(*expression*)

Restituisce `true` se l'espressione non è definita. Ad esempio, se l'input `MyInput` riceve il messaggio `{ "a": null }`, quanto segue restituisce `false`, ma `isNull($input.MyInput.a)` restituisce `si. true`

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

triggerType(*"type"*)

Il valore del *tipo* può essere `"Message"` o `"Timer"`. Valuta `true` se la condizione dell'evento in cui appare viene valutata perché un timer è scaduto, come nell'esempio seguente.

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

Oppure è stato ricevuto un messaggio di input.

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

currentInput("input")

Valuta `true` se la condizione dell'evento in cui appare viene valutata perché è stato ricevuto il messaggio di input specificato. Ad esempio, se l'input Command riceve il messaggio{ "value": "Abort" }, quanto segue restituisce. `true`

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

Utilizzate questa funzione per verificare che la condizione venga valutata perché è stato ricevuto un determinato input e non è scaduto un timer, come nell'espressione seguente.

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

Funzioni di corrispondenza delle stringhe

startsWith(*expression1*, *expression2*)

Restituisce `true` se la prima espressione stringa inizia con la seconda espressione di stringa. Ad esempio, se input MyInput riceve il messaggio{ "status": "offline"}, viene restituito quanto segue. `true`

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

Entrambe le espressioni devono restituire un valore di stringa. Se nessuna delle espressioni restituisce un valore di stringa, il risultato della funzione non è definito. Non viene eseguita alcuna conversione.

endsWith(*expression1*, *expression2*)

Valuta `true` se la prima espressione stringa termina con la seconda espressione stringa. Ad esempio, se input MyInput riceve il messaggio{ "status": "offline" }, viene restituito quanto segue. `true`

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

Entrambe le espressioni devono restituire un valore di stringa. Se nessuna delle espressioni restituisce un valore di stringa, il risultato della funzione non è definito. Non viene eseguita alcuna conversione.

contains(*expression1*, *expression2*)

Valuta `true` se la prima espressione stringa contiene la seconda espressione stringa. Ad esempio, se input `MyInput` riceve il messaggio `{ "status": "offline" }`, viene restituito quanto segue. `true`

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

Entrambe le espressioni devono restituire un valore di stringa. Se nessuna delle espressioni restituisce un valore di stringa, il risultato della funzione non è definito. Non viene eseguita alcuna conversione.

Funzioni di manipolazione di numeri interi bit a bit

bitor(*expression1*, *expression2*)

Valuta l'OR bit per bit delle espressioni intere (l'operazione OR binaria viene eseguita sui bit corrispondenti dei numeri interi). Ad esempio, se input `MyInput` riceve il messaggio `{ "value1": 13, "value2": 5 }`, quanto segue restituisce. `13`

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

Entrambe le espressioni devono restituire un valore intero. Se nessuna delle espressioni restituisce un valore intero, il risultato della funzione non è definito. Non viene eseguita alcuna conversione.

bitand(*expression1*, *expression2*)

Valuta l'AND bit per bit delle espressioni intere (l'operazione AND binaria viene eseguita sui bit corrispondenti dei numeri interi). Ad esempio, se input `MyInput` riceve il messaggio `{ "value1": 13, "value2": 5 }`, quanto segue restituisce. `5`

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

Entrambe le espressioni devono restituire un valore intero. Se nessuna delle espressioni restituisce un valore intero, il risultato della funzione non è definito. Non viene eseguita alcuna conversione.

bitxor(*expression1*, *expression2*)

Valuta lo XOR bit per bit delle espressioni intere (l'operazione XOR binaria viene eseguita sui bit corrispondenti dei numeri interi). Ad esempio, se input MyInput riceve il messaggio{ "value1": 13, "value2": 5 }, viene restituito quanto segue. 8

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

Entrambe le espressioni devono restituire un valore intero. Se nessuna delle espressioni restituisce un valore intero, il risultato della funzione non è definito. Non viene eseguita alcuna conversione.

bitnot(*expression*)

Valuta il NOT bit per bit dell'espressione intera (l'operazione binaria NOT viene eseguita sui bit del numero intero). Ad esempio, se input MyInput riceve il messaggio{ "value": 13 }, quanto segue restituisce. -14

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

Entrambe le espressioni devono restituire un valore intero. Se nessuna delle espressioni restituisce un valore intero, il risultato della funzione non è definito. Non viene eseguita alcuna conversione.

Riferimenti

Input

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

`input-name` è un input creato utilizzando l'[CreateInput](#) azione.

Ad esempio, se avete un input denominato TemperatureInput per il quale avete definito delle `inputDefinition.attributes.jsonPath` voci, i valori potrebbero apparire nei seguenti campi disponibili.

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

Per fare riferimento al valore del temperature campo, utilizzate il comando seguente.

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

Per i campi i cui valori sono matrici, è possibile fare riferimento ai membri dell'array utilizzando `[n]`. Ad esempio, dati i seguenti valori:

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

78.8 È possibile fare riferimento al valore con il comando seguente.

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

Variables

`$variable.variable-name`

variable-name è una variabile che hai definito utilizzando l'[CreateDetectorModelazione](#).

Ad esempio, se avete una variabile denominata TechnicianID che avete definito utilizzando `detectorDefinition.states.onInputEvents.actions.setVariable.variableName`, potete fare riferimento all'ultimo valore (stringa) assegnato alla variabile con il comando seguente.

```
$variable.TechnicianID
```

È possibile impostare i valori delle variabili solo utilizzando l'`setVariableazione`. Non è possibile assegnare valori alle variabili in un'espressione. Una variabile non può essere annullata. Ad esempio, non puoi assegnarle il valore. `null`

Note

Nei riferimenti che utilizzano identificatori che non seguono il modello (espressione regolare) `[a-zA-Z][a-zA-Z0-9_]*`, è necessario racchiudere tali identificatori in backticks

(). ` Ad esempio, un riferimento a un input denominato MyInput con un campo denominato `_value` deve specificare questo campo come `$input.MyInput._value``

Quando utilizzate riferimenti nelle espressioni, controllate quanto segue:

- Quando utilizzate un riferimento come operando con uno o più operatori, assicuratevi che tutti i tipi di dati a cui fate riferimento siano compatibili.

Ad esempio, nell'espressione seguente, il numero intero 2 è un operando di entrambi gli `==` operatori `and`. `&&` Per garantire che gli operandi siano compatibili `$variable.testVariable + 1` e che `$variable.testVariable` debbano fare riferimento a un numero intero o decimale.

Inoltre, il numero intero 1 è un operando dell'operatore `+`. Pertanto, `$variable.testVariable` deve fare riferimento a un numero intero o decimale.

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

- Quando utilizzate un riferimento come argomento passato a una funzione, assicuratevi che la funzione supporti i tipi di dati a cui fate riferimento.

Ad esempio, la seguente `timeout("time-name")` funzione richiede una stringa con virgolette doppie come argomento. Se si utilizza un riferimento per il valore `timer-name`, è necessario fare riferimento a una stringa tra virgolette doppie.

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

Note

Per la `convert(type, expression)` funzione, se si utilizza un riferimento per il valore del `tipo`, il risultato valutato del riferimento deve essere `String`, `Decimal` o `Boolean`

AWS IoT Events le espressioni supportano i tipi di dati interi, decimali, stringhe e booleani. La tabella seguente fornisce un elenco di coppie di tipi incompatibili.

Coppie di tipi incompatibili

Numero intero, stringa

Numero intero, booleano

Decimale, stringa

Decimale, booleano

Stringa, booleana

Modelli di sostituzione

'*expression*'

`${}` identifica la stringa come stringa interpolata. `expression` può essere qualsiasi espressione. AWS IoT Events Ciò include operatori, funzioni e riferimenti.

Ad esempio, hai usato l'[SetVariableAction](#) azione per definire una variabile. `variableName` è `SensorID`, e `value` è `10`. È possibile creare i seguenti modelli sostitutivi.

Modello sostitutivo	Stringa di risultato
' <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\"}"

Modello sostitutivo	Stringa di risultato
<code>'{\\"sensor\\":\${\$variable.SensorID + 1}}'</code>	<code>"{\\"sensor\\":11}"</code>

Utilizzo delle espressioni

È possibile specificare i valori in un modello di rilevatore nei seguenti modi:

- Immettete le espressioni supportate nella AWS IoT Events console.
- Passa le espressioni alle AWS IoT Events API come parametri.

Le espressioni supportano valori letterali, operatori, funzioni, riferimenti e modelli sostitutivi.

Important

Le espressioni devono fare riferimento a un numero intero, decimale, stringa o valore booleano.

Scrittura di AWS IoT Events espressioni

Guarda i seguenti esempi per aiutarti a scrivere AWS IoT Events le tue espressioni:

Letterale

Per i valori letterali, le espressioni devono contenere virgolette singole. Un valore booleano deve essere `true`. `false`

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

Riferimento

Per i riferimenti, è necessario specificare variabili o valori di input.

- Il seguente input fa riferimento a un numero decimale,. 10.01

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

- La variabile seguente fa riferimento a una stringa,. Greenhouse Temperature Table

```
$variable.TableName
```

Modello di sostituzione

Per un modello di sostituzione, è necessario utilizzare `${}` e il modello deve essere racchiuso tra virgolette singole. Un modello di sostituzione può anche contenere una combinazione di valori letterali, operatori, funzioni, riferimenti e modelli di sostituzione.

- Il risultato valutato della seguente espressione è una stringa,. 50.018 in Fahrenheit

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

- Il risultato valutato della seguente espressione è una stringa,. {"sensor_id": "Sensor_1", "temperature": "50.018"}

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

Concatenamento di stringhe

Per una concatenazione di stringhe, è necessario utilizzare `+`. Una concatenazione di stringhe può anche contenere una combinazione di valori letterali, operatori, funzioni, riferimenti e modelli di sostituzione.

- Il risultato valutato della seguente espressione è una stringa,. Greenhouse Temperature Table 2000-01-01

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

Esempi di modelli di rivelatori

Questa sezione contiene esempi di modelli e ingressi di rivelatori.

Argomenti

- [Controllo della temperatura HVAC](#)
- [Gru](#)
- [Rilevamento di eventi con sensori e applicazioni](#)
- [Dispositivo HeartBeat](#)
- [Allarme ISA](#)
- [Allarme semplice](#)

Controllo della temperatura HVAC

Storia di fondo

Questo esempio implementa un modello di controllo della temperatura (un termostato) con queste caratteristiche:

- Un modello di rivelatore definito dall'utente in grado di monitorare e controllare più aree. (Verrà creata un'istanza del rivelatore per ogni area.)
- Ogni istanza del rivelatore riceve i dati di temperatura da più sensori posizionati in ciascuna area di controllo.
- È possibile modificare la temperatura desiderata (il set point) per ogni area in qualsiasi momento.
- È possibile definire i parametri operativi per ciascuna area e modificarli in qualsiasi momento.
- È possibile aggiungere o eliminare sensori da un'area in qualsiasi momento.
- È possibile abilitare un tempo di funzionamento minimo delle unità di riscaldamento e raffreddamento per proteggerle da eventuali danni.
- I rivelatori rifiuteranno e segnaleranno le letture anomale dei sensori.
- È possibile definire i set point di temperatura di emergenza. Se un sensore riporta una temperatura superiore o inferiore ai set point definiti, le unità di riscaldamento o raffreddamento verranno attivate immediatamente e il rivelatore segnalerà il picco di temperatura.

Questo esempio dimostra le seguenti funzionalità funzionali:

- Crea modelli di rilevatori di eventi.
- Crea input.
- Inserisci gli input in un modello di rilevatore.
- Valuta le condizioni di attivazione.
- Fate riferimento alle variabili di stato nelle condizioni e impostate i valori delle variabili in base alle condizioni.
- Fai riferimento ai timer nelle condizioni e imposta i timer in base alle condizioni.
- Intraprendi azioni per inviare messaggi Amazon SNS e MQTT.

Definizioni di input

A "seedTemperatureInput" viene utilizzato per creare un'istanza di rilevatore per un'area e definirne i parametri operativi.

Comando CLI utilizzato:

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

File: 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" }
    ]
  }
}
```

```
}
```

Risposta:

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

A "temperatureInput" deve essere inviato da ciascun sensore in ogni area, se necessario.

Comando CLI utilizzato:

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

File: temperatureInput.json

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

Risposta:

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

Definizione del modello di rilevatore

"areaDetectorModel" Definisce come funziona ogni istanza del rilevatore. Ogni "state machine" istanza acquisirà le letture del sensore di temperatura, quindi cambierà stato e invierà messaggi di controllo in base a queste letture.

Comando CLI utilizzato:

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

File: areaDetectorModel.json

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

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

```

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

```

```

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

```

```

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

```

```
    }
  ],
  "transitionEvents": [
    {
      "eventName": "anomalousInputArrived",
      "condition": "$input.temperatureInput.sensorData.temperature >=
$variable.anomalousHigh || $input.temperatureInput.sensorData.temperature <=
$variable.anomalousLow",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/anomaly"
          }
        }
      ],
      "nextState": "idle"
    },
    {
      "eventName": "highTemperatureSpike",
      "condition": "$input.temperatureInput.sensorData.temperature >
$variable.rangeHigh",
      "actions": [
        {
          "iotTopicPublish": {
            "mqttTopic": "temperatureSensor/spike"
          }
        },
        {
          "sns": {
            "targetArn": "arn:aws:sns:us-west-2:123456789012:coolOn"
          }
        },
        {
          "iotTopicPublish": {
            "mqttTopic": "hvac/Cooling/On"
          }
        },
        {
          "setVariable": {
            "variableName": "enteringNewState",
            "value": "true"
          }
        }
      ]
    }
  ]
}
```



```

    ],
    "nextState": "cooling"
  },

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

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

```

```

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

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

```

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

```

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

```

```

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

```

```

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

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

```

Risposta:

```

{
    "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"
    }
}

```

BatchPutMessageesempi

In questo esempio, "BatchPutMessage" viene utilizzato per creare un'istanza del rilevatore per un'area e definire i parametri operativi iniziali.

Comando CLI utilizzato:

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

File: 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}"
    }
  ]
}
```

Risposta:

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

In questo esempio, "BatchPutMessage" viene utilizzato per riportare le letture del sensore di temperatura per un singolo sensore in un'area.

Comando CLI utilizzato:

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

File: temperatureExample.json

```
{
```

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

Risposta:

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

In questo esempio, "BatchPutMessage" viene utilizzato per modificare la temperatura desiderata per un'area.

Comando CLI utilizzato:

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

File: seedSetDesiredTemp.json

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

Risposta:

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

Esempi di messaggi Amazon SNS generati dall'istanza del Area51 rilevatore:

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

```
Cooling system off command> {
  "eventTime":1557520274729,
```

```

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

```

In questo esempio, utilizziamo l'"DescribeDetector" API per ottenere informazioni sullo stato corrente di un'istanza del rilevatore.

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

Risposta:


```
{
  "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 esempio

In questo esempio, "BatchUpdateDetector" viene utilizzato per modificare i parametri operativi di un'istanza del rilevatore funzionante.

Comando CLI utilizzato:

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

File: 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": [
]
}
]
}
}

```

Risposta:

```

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

```

AWS IoT Core esempi di motori di regole

Le seguenti regole ripubblicano i messaggi AWS IoT Events MQTT come messaggi di richiesta di aggiornamento shadow. Partiamo dal presupposto che AWS IoT Core le cose siano definite per un'unità di riscaldamento e un'unità di raffreddamento per ogni area controllata dal modello di rilevatore.

In questo esempio, abbiamo definito cose denominate "Area51HeatingUnit" e "Area51CoolingUnit".

Comando CLI utilizzato:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowCool0ffRule.json
```

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

Risposta: [vuoto]

Comando CLI utilizzato:

```
aws iot create-topic-rule --cli-input-json file://ADMSHadowCool0nRule.json
```

File: ADMSHadowCool0nRule.json

```
{
  "ruleName": "ADMSHadowCool0n",
  "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"
        }
      }
    ]
  }
}

```

Risposta: [vuoto]

Comando CLI utilizzato:

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

File: 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"
        }
      }
    ]
  }
}

```

Risposta: [vuoto]

Comando CLI utilizzato:

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

File: 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"
        }
      }
    ]
  }
}
```

Risposta: [vuoto]

Gru

Storia di fondo

Un operatore di molte gru desidera rilevare quando le macchine necessitano di manutenzione o sostituzione e attivare le notifiche appropriate. Ogni gru è dotata di un motore. Un motore emette messaggi (input) con informazioni su pressione e temperatura. L'operatore desidera due livelli di rilevatori di eventi:

- Un rilevatore di eventi a livello di gru
- Un rilevatore di eventi a livello motorio

Utilizzando i messaggi provenienti dai motori (che contengono metadati sia con `motorId` che con `craneId`), l'operatore può eseguire entrambi i livelli di rilevatori di eventi utilizzando il routing appropriato. Quando vengono soddisfatte le condizioni dell'evento, le notifiche devono essere inviate agli argomenti appropriati di Amazon SNS. L'operatore può configurare i modelli del rilevatore in modo che non vengano generate notifiche duplicate.

Questo esempio dimostra le seguenti funzionalità funzionali:

- Crea, leggi, aggiorna, elimina (CRUD) degli input.
- Crea, leggi, aggiorna, elimina (CRUD) di modelli di rilevatori di eventi e diverse versioni di rilevatori di eventi.
- Instradamento di un ingresso verso più rilevatori di eventi.
- Inserimento di input in un modello di rilevatore.
- Valutazione delle condizioni di attivazione e degli eventi del ciclo di vita.
- Capacità di fare riferimento alle variabili di stato nelle condizioni e di impostarne i valori in base alle condizioni.
- Orchestrazione del runtime con definizione, stato, valutatore dei trigger ed esecutore delle azioni.
- Esecuzione di azioni `ActionsExecutor` con un target SNS.

Comandi

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

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

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



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

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

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

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

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

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

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

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

Modelli di rilevatore

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

```

Per aggiornare un modello di rilevatore esistente. File: updateCraneDetectorModel.json

```

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

```



```

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

```

Archivio: 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 Breached",
      "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"
}

```

Per aggiornare un modello di rilevatore esistente. File: `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"
}

```

Input

Archivio: pressureInput.json

```

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

```

Archivio: temperatureInput.json

```

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

```


Messaggi

Archivio: highPressureMessage.json

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

Archivio: highTemperatureMessage.json

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

Archivio: lowPressureMessage.json

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

Archivio: lowTemperatureMessage.json

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

Rilevamento di eventi con sensori e applicazioni

Questo modello di rilevatore è uno dei modelli disponibili nella AWS IoT Events console. È incluso qui per una maggiore comodità.

```
{
  "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"
}
}

```

Dispositivo HeartBeat

Questo modello di rilevatore è uno dei modelli disponibili nella console. AWS IoT Events È incluso qui per una maggiore comodità.

```

{
  "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"
}
}

```

Allarme ISA

Questo modello di rilevatore è uno dei modelli disponibili nella console. AWS IoT Events È incluso qui per una maggiore comodità.

```

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

```

Allarme semplice

Questo modello di rilevatore è uno dei modelli disponibili nella console. AWS IoT Events È incluso qui per una maggiore comodità.

```

{
  "detectorModelDefinition": {
    "states": [
      {
        "onInput": {
          "transitionEvents": [
            {
              "eventName": "not_fixed",
              "actions": [],
            }
          ]
        }
      }
    ]
  }
}

```



```

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

```

```

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

```

```

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

```

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

Monitoraggio con allarmi

AWS IoT Events gli allarmi ti aiutano a monitorare i dati per rilevare eventuali modifiche. I dati possono essere metriche misurate per apparecchiature e processi. È possibile creare allarmi che inviano notifiche quando viene superata una soglia. Gli allarmi consentono di rilevare problemi, semplificare la manutenzione e ottimizzare le prestazioni delle apparecchiature e dei processi.

Gli allarmi sono esempi di modelli di allarme. Il modello di allarme specifica cosa rilevare, quando inviare le notifiche, chi riceve le notifiche e altro ancora. È inoltre possibile specificare una o più [azioni supportate](#) che si verificano quando lo stato dell'allarme cambia. AWS IoT Events indirizza [gli attributi di input](#) derivati dai dati agli allarmi appropriati. Se i dati che stai monitorando non rientrano nell'intervallo specificato, viene richiamato l'allarme. Puoi anche confermare gli allarmi o impostarli sulla modalità snooze.

Utilizzo di AWS IoT SiteWise

È possibile utilizzare gli AWS IoT Events allarmi per monitorare le proprietà degli asset in AWS IoT SiteWise. AWS IoT SiteWise invia i valori delle proprietà degli asset agli AWS IoT Events allarmi. AWS IoT Events invia lo stato di allarme a AWS IoT SiteWise.

AWS IoT SiteWise supporta anche allarmi esterni. Puoi scegliere allarmi esterni se utilizzi allarmi esterni AWS IoT SiteWise e disponi di una soluzione che restituisce i dati sullo stato degli allarmi. L'allarme esterno contiene una proprietà di misurazione che inserisce i dati sullo stato dell'allarme.

AWS IoT SiteWise non valuta lo stato degli allarmi esterni. Inoltre, non è possibile confermare o posticipare un allarme esterno quando lo stato dell'allarme cambia.

È possibile utilizzare la funzione SiteWise Monitor per visualizzare lo stato degli allarmi esterni nei SiteWise portali Monitor.

Per ulteriori informazioni, consulta [Monitoraggio dei dati con allarmi](#) nella Guida per l'AWS IoT SiteWise utente e [Monitoraggio con allarmi](#) nella Guida all'AWS IoT SiteWise applicazione Monitor.

Conferma il flusso

Quando crei un modello di allarme, scegli se abilitare il flusso di conferma. Se abiliti il flusso di conferma, il tuo team riceve una notifica quando lo stato dell'allarme cambia. Il tuo team può

confermare l'allarme e lasciare una nota. Ad esempio, puoi includere le informazioni sull'allarme e le azioni che intendi intraprendere per risolvere il problema. Se i dati che stai monitorando non rientrano nell'intervallo specificato, viene richiamato l'allarme.

Gli allarmi hanno i seguenti stati:

DISABLED

Quando l'allarme è attivo, non è pronto per valutare i dati. **DISABLED** Per abilitare l'allarme, è necessario impostarlo sullo **NORMAL** stato.

NORMAL

Quando l'allarme è attivo **NORMAL**, è pronto per valutare i dati.

ACTIVE

Se l'allarme è nello **ACTIVE** stato, viene richiamato. I dati che stai monitorando non rientrano nell'intervallo specificato.

ACKNOWLEDGED

Quando l'allarme è nello **ACKNOWLEDGED** stato, l'allarme è stato richiamato e tu hai riconosciuto l'allarme.

LATCHED

L'allarme è stato richiamato, ma non l'hai riconosciuto dopo un certo periodo di tempo. L'allarme passa automaticamente allo **NORMAL** stato.

SNOOZE_DISABLED

Quando l'allarme è nello **SNOOZE_DISABLED** stato, viene disabilitato per un periodo di tempo specificato. Dopo l'orario di snooze, la sveglia passa automaticamente allo **NORMAL** stato.

Creazione di un modello di allarme

Puoi utilizzare gli AWS IoT Events allarmi per monitorare i tuoi dati e ricevere notifiche quando viene superata una soglia. Gli allarmi forniscono parametri che puoi utilizzare per creare o configurare un modello di allarme. Puoi utilizzare la AWS IoT Events console o l'AWS IoT Events API per creare o configurare il modello di allarme. Quando configuri il modello di allarme, le modifiche diventano effettive non appena arrivano nuovi dati.

Requisiti

I seguenti requisiti si applicano quando si crea un modello di allarme.

- È possibile creare un modello di allarme per monitorare un attributo di input AWS IoT Events o una proprietà di un asset in AWS IoT SiteWise.
 - Se scegliete di monitorare un attributo di input in AWS IoT Events, effettuate le seguenti operazioni prima di creare il modello di allarme:
 - Fase 1: Leggi la panoramica sulla [creazione di un input](#).
 - Passaggio 2: leggi le istruzioni per [creare un input nel riquadro di navigazione](#).
 - Se scegli di monitorare la proprietà di un asset, devi [creare un modello di asset](#) AWS IoT SiteWise prima di creare il modello di allarme.
- È necessario disporre di un ruolo IAM che consenta all'allarme di eseguire azioni e accedere alle AWS risorse. Per ulteriori informazioni, consulta [Configurazione delle autorizzazioni per AWS IoT Events](#).
- Tutte le AWS risorse utilizzate in questo tutorial devono trovarsi nella stessa AWS regione.

Creazione di un modello di allarme (console)

Di seguito viene illustrato come creare un modello di allarme per monitorare un AWS IoT Events attributo nella AWS IoT Events console.

1. Accedi alla [console AWS IoT Events](#).
2. Nel pannello di navigazione, scegli Modelli di allarme.
3. Nella pagina Modelli di allarme, scegli Crea modello di allarme.
4. Nella sezione Dettagli del modello di allarme, procedi come segue:
 - a. Inserisci un nome univoco.
 - b. (Opzionale) Immettere una descrizione.
5. Nella sezione Obiettivo dell'allarme, procedi come segue:

⚠ Important

Se scegliete la proprietà dell'AWS IoT SiteWiseasset, dovete aver creato un modello di asset inAWS IoT SiteWise.

- a. Scegliete AWS IoT Eventsl'attributo di input.
- b. Scegli l'input.
- c. Scegli la chiave dell'attributo di input. Questo attributo di input viene utilizzato come chiave per creare l'allarme. AWS IoT Eventsindirizza gli ingressi associati a questa chiave verso l'allarme.

⚠ Important

Se il payload del messaggio di input non contiene questa chiave di attributo di input o se la chiave non si trova nello stesso percorso JSON specificato nella chiave, l'inserimento del messaggio non riuscirà. AWS IoT Events

6. Nella sezione Definizioni delle soglie, si definiscono l'attributo di input, il valore di soglia e l'operatore di confronto da AWS IoT Events utilizzare per modificare lo stato dell'allarme.
 - a. Per l'attributo di input, scegliete l'attributo che desiderate monitorare.

Ogni volta che questo attributo di input riceve nuovi dati, viene valutato per determinare lo stato dell'allarme.
 - b. Per Operatore, scegli l'operatore di confronto. L'operatore confronta l'attributo di input con il valore di soglia dell'attributo.

Puoi scegliere tra queste opzioni:

- > maggiore di
- >= maggiore o uguale a
- < minore di
- <= minore o uguale a
- = uguale a
- != diverso da

- c. Per Valore di soglia, inserisci un numero o scegli un attributo negli AWS IoT Events input. AWS IoT Events confronta questo valore con il valore dell'attributo di input scelto.
 - d. (Facoltativo) Per la gravità, utilizzate un numero che il team comprenda per riflettere la gravità di questo allarme.
7. (Facoltativo) Nella sezione Impostazioni di notifica, configura le impostazioni di notifica per l'allarme.

Puoi aggiungere fino a 10 notifiche. Per Notifica 1, procedi come segue:

- a. Per Protocollo, scegli una delle seguenti opzioni:
 - E-mail e testo: l'allarme invia una notifica SMS e una notifica e-mail.
 - E-mail: l'allarme invia una notifica via e-mail.
 - Testo: l'allarme invia una notifica via SMS.
- b. Per Mittente, specifica l'indirizzo e-mail che può inviare notifiche relative a questo allarme.

Per aggiungere altri indirizzi e-mail all'elenco dei mittenti, scegli Aggiungi mittente.

- c. (Facoltativo) Per Destinatario, scegli il destinatario.

Per aggiungere altri utenti all'elenco dei destinatari, scegli Aggiungi nuovo utente. Devi aggiungere nuovi utenti al tuo negozio IAM Identity Center prima di poterli aggiungere al tuo modello di allarme. Per ulteriori informazioni, consulta [Gestione dei destinatari](#).

- d. (Facoltativo) Per Messaggio personalizzato aggiuntivo, inserisci un messaggio che descriva cosa rileva l'allarme e quali azioni devono intraprendere i destinatari.
8. Nella sezione Istanza, puoi abilitare o disabilitare tutte le istanze di allarme create sulla base di questo modello di allarme.
9. Nella sezione Impostazioni avanzate, procedi come segue:
- a. Per il flusso di conferma, puoi abilitare o disabilitare le notifiche.
 - Se scegli Abilitato, ricevi una notifica quando lo stato dell'allarme cambia. È necessario confermare la notifica prima che lo stato di allarme possa tornare alla normalità.
 - Se scegli Disabilitato, non è richiesta alcuna azione. L'allarme passa automaticamente allo stato normale quando la misurazione ritorna all'intervallo specificato.

Per ulteriori informazioni, consulta [Conferma il flusso](#).

- b. Per Autorizzazioni, scegli una delle seguenti opzioni:
- Puoi creare un nuovo ruolo dai modelli di AWS policy e creare AWS IoT Events automaticamente un ruolo IAM per te.
 - Puoi utilizzare un ruolo IAM esistente che consente a questo modello di allarme di eseguire azioni e accedere ad altre AWS risorse.

Per ulteriori informazioni, consulta [Identity and Access Management per AWS IoT Events](#).

- c. Per le impostazioni di notifica aggiuntive, è possibile modificare la AWS Lambda funzione per gestire le notifiche di allarme. Scegliete una delle seguenti opzioni per la vostra AWS Lambda funzione:
- Crea una nuova AWS Lambda funzione: AWS IoT Events crea una nuova AWS Lambda funzione per te.
 - Usa una AWS Lambda funzione esistente: utilizza una AWS Lambda funzione esistente scegliendo il nome di una AWS Lambda funzione.

Per ulteriori informazioni sulle azioni possibili, vedere [Lavorare con altri AWS servizi](#).

- d. (Facoltativo) In Imposta l'azione dello stato, puoi aggiungere una o più AWS IoT Events azioni da intraprendere quando lo stato dell'allarme cambia.
10. (Facoltativo) Puoi aggiungere tag per gestire gli allarmi. Per ulteriori informazioni, consulta [Tagging delle risorse AWS IoT Events](#).
11. Seleziona Create (Crea).

Risposta agli allarmi

Se hai abilitato [il flusso di conferma](#), ricevi notifiche quando lo stato dell'allarme cambia. Per rispondere all'allarme, puoi confermare, disabilitare, abilitare, ripristinare o posticipare l'allarme.

Risposta agli allarmi (console)

Di seguito viene illustrato come rispondere a un allarme nella AWS IoT Events console.

1. Accedi alla [console AWS IoT Events](#).
2. Nel pannello di navigazione, scegli Modelli di allarme.

3. Scegli il modello di allarme di destinazione.
4. Nella sezione Elenco degli allarmi, scegli l'allarme di destinazione.
5. Puoi scegliere una delle seguenti opzioni da Azioni:
 - Conferma: l'allarme passa allo ACKNOWLEDGED stato.
 - Disabilita: l'allarme passa allo DISABLED stato.
 - Abilita: l'allarme passa allo NORMAL stato.
 - Ripristina: l'allarme passa allo NORMAL stato.
 - Snooze, quindi procedi come segue:
 1. Scegli la durata dello snooze o inserisci una lunghezza dello snooze personalizzata.
 2. Seleziona Salva.

L'allarme passa allo stato SNOOZE_DISABLED

Per ulteriori informazioni sugli stati di allarme, vedere [Conferma il flusso](#).

Risposta agli allarmi (API)

Per rispondere a uno o più allarmi, puoi utilizzare le seguenti operazioni API: AWS IoT Events

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

Gestione delle notifiche di allarme

AWS IoT Events utilizza una funzione Lambda per gestire le notifiche di allarme. È possibile utilizzare la funzione Lambda fornita da AWS IoT Events o crearne una nuova.

Creazione di una funzione Lambda

AWS IoT Events fornisce una funzione Lambda che consente agli allarmi di inviare e ricevere notifiche e-mail e SMS.

Requisiti

I seguenti requisiti si applicano quando si crea una funzione Lambda per gli allarmi:

- Se il tuo allarme invia notifiche e-mail o SMS, devi disporre di un ruolo IAM che AWS Lambda consenta di lavorare con Amazon SES e Amazon SNS.

Esempio di politica:

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

```
}
```

- È necessario scegliere la stessa AWS regione per entrambi AWS IoT Events e AWS Lambda. Per l'elenco delle regioni supportate, consulta [AWS IoT Events endpoint e quote](#) e [AWS Lambda endpoint e quote](#) in. Riferimenti generali di Amazon Web Services

Implementazione di una funzione Lambda

Questo tutorial utilizza un AWS CloudFormation modello per distribuire una funzione Lambda. Questo modello crea automaticamente un ruolo IAM che consente alla funzione Lambda di funzionare con Amazon SES e Amazon SNS.

Di seguito viene illustrato come utilizzare la AWS Command Line Interface (AWS CLI) per creare uno CloudFormation stack.

1. Nel terminale del tuo dispositivo, `aws --version` esegui per verificare se hai installato il AWS CLI. Per ulteriori informazioni, consulta [Installazione della AWS CLI](#) nella Guida per l'utente di AWS Command Line Interface.
2. Esegui `aws configure list` per verificare se lo hai configurato AWS CLI nella AWS regione che ha tutte le AWS risorse per questo tutorial. Per ulteriori informazioni, consulta [Configurazione di AWS CLI nella Guida per l'AWS Command Line Interface utente](#)
3. Scarica il CloudFormation modello, [NotificationLambda.template.yaml.zip](#).

Note

Se hai difficoltà a scaricare il file, il modello è disponibile anche in. [CloudFormation modello](#)

4. Decomprimere il contenuto e salvarlo localmente come `notificationLambda.template.yaml`.
5. Apri un terminale sul tuo dispositivo e vai alla directory in cui hai scaricato il `notificationLambda.template.yaml` file.
6. Per creare uno CloudFormation stack, esegui il seguente comando:

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

È possibile modificare questo CloudFormation modello per personalizzare la funzione Lambda e il relativo comportamento.

Note

AWS Lambda prova due volte gli errori di funzione. Se la funzione non dispone di capacità sufficiente per gestire tutte le richieste in entrata, gli eventi possono attendere in coda per ore o giorni per essere inviati alla funzione. È possibile configurare una coda di messaggi non recapitati (DLQ) sulla funzione per acquisire gli eventi che non sono stati elaborati correttamente. Per ulteriori informazioni, consulta [Chiamata asincrona](#) nella AWS Lambda Guida per gli sviluppatori.

Puoi anche creare o configurare lo stack nella console. CloudFormation Per ulteriori informazioni, consulta [Working with stacks](#), nella Guida per l'AWS CloudFormation utente.

Creazione di una funzione Lambda personalizzata

È possibile creare una funzione Lambda o modificare quella fornita da AWS IoT Events

I seguenti requisiti si applicano quando si crea una funzione Lambda personalizzata.

- Aggiungi autorizzazioni che consentono alla funzione Lambda di eseguire azioni specifiche e AWS accedere alle risorse.
- Se usi la funzione Lambda fornita da AWS IoT Events, assicurati di scegliere il runtime Python 3.7.

Esempio di funzione 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')
```

Per ulteriori informazioni, consulta [Cos'è AWS Lambda?](#) nella Guida per gli sviluppatori AWS Lambda.

CloudFormation modello

Usa il seguente CloudFormation modello per creare la tua funzione 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')

```

Utilizzo della funzione Lambda fornita da AWS IoT Events

I seguenti requisiti si applicano quando si utilizza la funzione Lambda fornita da AWS IoT Events per gestire le notifiche di allarme:

- Devi verificare l'indirizzo e-mail che invia le notifiche e-mail in Amazon Simple Email Service (Amazon SES). Per ulteriori informazioni, consulta la sezione [Verifica degli indirizzi e-mail in Amazon SES](#), nella Amazon Simple Email Service Developer Guide.

Se ricevi un link di verifica, fai clic sul link per verificare il tuo indirizzo e-mail. Puoi anche controllare se c'è un'email di verifica nella cartella spam.

- Se la sveglia invia notifiche SMS, è necessario utilizzare la formattazione dei numeri di telefono internazionali E.164 per i numeri di telefono. Questo formato contiene. `+<country-calling-code><area-code><phone-number>`

Numeri di telefono di esempio:

Paese	Numero di telefono locale	Numero formattato E.164
Stati Uniti	206-555-0100	+12065550100
Regno Unito	020-1234-1234	+442012341234
Lituania	8+601+12345	+37060112345

[Per trovare il prefisso internazionale, vai su countrycode.org.](http://countrycode.org)

La funzione Lambda fornita da AWS IoT Events verifica se si utilizzano numeri di telefono in formato E.164. Tuttavia, non verifica i numeri di telefono. Se ti assicuri di aver inserito numeri di telefono corretti ma di non aver ricevuto notifiche SMS, puoi contattare i gestori telefonici. I gestori potrebbero bloccare i messaggi.

Gestione dei destinatari

AWS IoT Events utilizza AWS IAM Identity Center (IAM Identity Center) per gestire l'accesso SSO dei destinatari degli allarmi. Per consentire all'allarme di inviare notifiche ai destinatari, devi abilitare IAM Identity Center e aggiungere i destinatari al tuo archivio IAM Identity Center. Per ulteriori informazioni, consulta [Add Users](#) in AWS IAM Identity Center User Guide.

Important

- Devi scegliere la stessa AWS regione per e per AWS IoT Events IAM Identity Center. AWS Lambda
- AWS Organizations supporta solo una regione IAM Identity Center alla volta. Se desideri rendere IAM Identity Center disponibile in una regione diversa, devi prima eliminare la

configurazione corrente di IAM Identity Center. Per ulteriori informazioni, consulta [IAM Identity Center Region Data](#) in AWS IAM Identity Center User Guide.

Sicurezza in AWS IoT Events

Per AWS, la sicurezza del cloud ha la massima priorità. In quanto cliente AWS, è possibile trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle organizzazioni più esigenti a livello di sicurezza.

La sicurezza è una responsabilità condivisa tra te e AWS. Il [modello di responsabilità condivisa](#) descrive questo come sicurezza del cloud e sicurezza nel cloud:

- La sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che esegue AWS i servizi nel cloud AWS. AWS fornisce, inoltre, servizi utilizzabili in modo sicuro. L'efficacia della nostra sicurezza è regolarmente testata e verificata da revisori di terze parti come parte dei [programmi di conformità AWS](#). Per ulteriori informazioni sui programmi di conformità che si applicano a AWS IoT Events, consulta [Servizi AWS coperti dal programma di compliance](#).
- Sicurezza nel cloud: la tua responsabilità è determinata dal servizio AWS che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda, nonché le leggi e le normative applicabili.

Questa documentazione consente di comprendere come applicare il modello di responsabilità condivisa quando si usa AWS IoT Events. I seguenti argomenti illustrano come configurare AWS IoT Events per soddisfare gli obiettivi di sicurezza e conformità. Imparerai anche a utilizzare altri AWS servizi che possono aiutarti a monitorare e proteggere AWS IoT Events le tue risorse.

Argomenti

- [Gestione delle identità e degli accessi per l'AWS IoT Events](#)
- [Monitoraggio di AWS IoT Events](#)
- [Convalida della conformità per AWS IoT Events](#)
- [Resilienza in AWS IoT Events](#)
- [Sicurezza dell'infrastruttura in AWS IoT Events](#)

Gestione delle identità e degli accessi per l'AWS IoT Events

AWS Identity and Access Management (IAM) è un servizio AWS che consente agli amministratori di controllare in modo sicuro l'accesso alle risorse AWS. Gli amministratori IAM controllano chi è

autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare risorse AWS IoT Events. IAM è un servizio AWS che è possibile utilizzare senza alcun costo aggiuntivo.

Argomenti

- [Destinatari](#)
- [Autenticazione con identità](#)
- [Gestione dell'accesso con policy](#)
- [Ulteriori informazioni](#)
- [Funzionamento di AWS IoT Events con IAM](#)
- [Esempi di policy di AWS IoT Events basate su identità](#)
- [Prevenzione del problema "confused deputy" tra servizi](#)
- [Risoluzione dei problemi di identità e accesso in AWS IoT Events](#)

Destinatari

Le modalità di utilizzo di AWS Identity and Access Management (IAM) cambiano in base alle operazioni eseguite in AWS IoT Events.

Utente del servizio: se utilizzi il servizio AWS IoT Events per eseguire il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di funzionalità AWS IoT Events utilizzate per il lavoro, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di AWS IoT Events, consulta [Risoluzione dei problemi di identità e accesso in AWS IoT Events](#).

Amministratore del servizio: se sei il responsabile delle risorse AWS IoT Events presso la tua azienda, probabilmente disponi dell'accesso completo a AWS IoT Events. Il tuo compito è determinare le caratteristiche e le risorse AWS IoT Events a cui gli utenti del servizio devono accedere. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con AWS IoT Events, consulta [Funzionamento di AWS IoT Events con IAM](#).

Amministratore IAM: un amministratore IAM potrebbe essere interessato a ottenere dei dettagli su come scrivere policy per gestire l'accesso a AWS IoT Events. Per visualizzare policy basate su

identità di AWS IoT Events di esempio che puoi utilizzare in IAM, consulta [Esempi di policy di AWS IoT Events basate su identità](#).

Autenticazione con identità

L'autenticazione è la procedura di accesso ad AWS con le credenziali di identità. Devi essere autenticato (connesso a AWS) come utente root Utente root dell'account AWS, come utente IAM o assumere un ruolo IAM.

Puoi accedere ad AWS come identità federata utilizzando le credenziali fornite attraverso un'origine di identità. Gli utenti AWS IAM Identity Center (Centro identità IAM), l'autenticazione Single Sign-On (SSO) dell'azienda e le credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Se accedi ad AWS tramite la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere alla AWS Management Console o al portale di accesso AWS. Per ulteriori informazioni sull'accesso ad AWS, consulta la sezione [Come accedere al tuo Account AWS](#) nella Guida per l'utente di Accedi ad AWS.

Se accedi ad AWS in modo programmatico, AWS fornisce un Software Development Kit (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le richieste utilizzando le tue credenziali. Se non utilizzi gli strumenti AWS, devi firmare le richieste personalmente. Per ulteriori informazioni sulla firma delle richieste, consulta [Firma delle richieste AWS](#) nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. AWS consiglia ad esempio di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza dell'account. Per ulteriori informazioni, consulta [Autenticazione a più fattori](#) nella Guida per l'utente di AWS IAM Identity Center e [Utilizzo dell'autenticazione a più fattori \(MFA\) in AWS](#) nella Guida per l'utente di IAM.

Utente root di un Account AWS

Quando crei un Account AWS, inizi con una singola identità di accesso che ha accesso completo a tutti i Servizi AWS e le risorse nell'account. Tale identità è detta utente root Account AWS ed è possibile accedervi con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione [Attività che richiedono le credenziali dell'utente root](#) nella Guida per l'utente di IAM.

Utenti e gruppi IAM

Un [utente IAM](#) è una identità all'interno del tuo Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina [Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine](#) nella Guida per l'utente di IAM.

Un [gruppo IAM](#) è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato Amministratori IAM e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta [Quando creare un utente IAM \(invece di un ruolo\)](#) nella Guida per l'utente di IAM.

Ruoli IAM

Un [ruolo IAM](#) è un'identità all'interno di un Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. È possibile assumere temporaneamente un ruolo IAM nella AWS Management Console mediante lo [scambio di ruoli](#). È possibile assumere un ruolo chiamando un'azione AWS CLI o API AWS oppure utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta [Utilizzo di ruoli IAM](#) nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- **Accesso utente federato:** per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta [Creazione di un ruolo per un provider di identità di terza parte](#) nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un

ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per ulteriori informazioni sui set di autorizzazioni, consulta [Set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center.

- **Autorizzazioni utente IAM temporanee:** un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- **Accesso multi-account:** è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, per alcuni dei Servizi AWS, è possibile collegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta [Differenza tra i ruoli IAM e le policy basate su risorse](#) nella Guida per l'utente di IAM.
- **Accesso multi-servizio:** alcuni Servizi AWS utilizzano funzionalità in altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
- **Inoltro delle sessioni di accesso (FAS):** quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, tale utente o ruolo viene considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina [Forward access sessions](#).
- **Ruolo di servizio:** un ruolo di servizio è un [ruolo IAM](#) assunto da un servizio per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione [Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS](#) nella Guida per l'utente di IAM.
- **Ruolo collegato al servizio:** un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: è possibile utilizzare un ruolo IAM per gestire credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 che eseguono richieste di AWS CLI o dell'API AWS. Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un ruolo AWS a un'istanza EC2, affinché sia disponibile per tutte le relative applicazioni, puoi creare un profilo dell'istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta [Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2](#) nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta [Quando creare un ruolo IAM \(invece di un utente\)](#) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Per controllare l'accesso a AWS è possibile creare policy e collegarle a identità o risorse AWS. Una policy è un oggetto in AWS che, quando associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste policy quando un principale IAM (utente, utente root o sessione ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle policy viene archiviata in AWS sotto forma di documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta [Panoramica delle policy JSON](#) nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWSJSON per specificare l'accesso ai diversi elementi. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla AWS Management Console, la AWS CLI o l'API AWS.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Creazione di policy IAM](#) nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo utente, gruppo o ruolo. Le policy gestite sono policy autonome che possono essere collegate a più utenti, gruppi e ruoli in Account AWS. Le policy gestite includono le policy gestite da AWS e le policy gestite dal cliente. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta [Scelta fra policy gestite e policy inline](#) nella Guida per l'utente di IAM.

Altri tipi di policy

AWS supporta altri tipi di policy meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- **Limiti delle autorizzazioni:** un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta [Limiti delle autorizzazioni per le entità IAM](#) nella Guida per l'utente di IAM.
- **Policy di controllo dei servizi (SCP):** le SCP sono policy JSON che specificano il numero massimo di autorizzazioni per un'organizzazione o unità organizzativa (OU) in AWS Organizations. AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata degli Account AWS multipli di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. La SCP limita le autorizzazioni per le entità negli account membri, compreso ogni Utente root dell'account AWS. Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle [Policy di controllo dei servizi](#) nella Guida per l'utente di AWS Organizations.
- **Policy di sessione:** le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate

su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta [Policy di sessione](#) nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta [Logica di valutazione delle policy](#) nella Guida per l'utente di IAM.

Ulteriori informazioni

Per ulteriori informazioni su Identity and Access Management per AWS IoT Events, continua nelle pagine seguenti:

- [Funzionamento di AWS IoT Events con IAM](#)
- [Risoluzione dei problemi di identità e accesso in AWS IoT Events](#)

Funzionamento di AWS IoT Events con IAM

Prima di utilizzare IAM per gestire l'accesso a AWS IoT Events, è necessario comprendere quali caratteristiche IAM sono disponibili per l'uso con AWS IoT Events. Per ottenere un quadro generale del funzionamento di AWS IoT Events e altri servizi AWS con IAM, consulta la sezione [Servizi AWS supportati da IAM](#) nella Guida per l'utente di IAM.

Argomenti

- [Policy AWS IoT Events basate su identità](#)
- [Policy di AWS IoT Events basate sulle risorse](#)
- [Autorizzazione basata su tag AWS IoT Events](#)
- [Ruoli IAM di AWS IoT Events](#)

Policy AWS IoT Events basate su identità

Con le policy basate su identità IAM, puoi specificare operazioni e risorse consentite o rifiutate, nonché le condizioni in base alle quali le operazioni sono consentite o rifiutate. AWS IoT Events

supporta operazioni, risorse e chiavi di condizione specifiche. Per informazioni su tutti gli elementi utilizzati in una policy JSON, consulta [Documentazione di riferimento degli elementi delle policy JSON IAM](#) nella Guida per l'utente IAM.

Azioni

L'elemento `Action` di una policy basata su identità IAM descrive l'operazione o le operazioni specifiche che saranno concesse o rifiutate dalla policy. Le operazioni della policy hanno spesso lo stesso nome dell'operazione API AWS. L'operazione viene utilizzata in una policy per concedere le autorizzazioni di eseguire l'operazione associata.

Le operazioni delle policy in AWS IoT Events utilizzano il seguente prefisso prima dell'operazione: `iotevents:`. Ad esempio, per concedere a qualcuno l'autorizzazione a creare un AWS IoT Events input con l'operazione AWS IoT Events `CreateInput` API, includi `iotevents:CreateInput` nella sua politica. Per concedere a qualcuno l'autorizzazione a inviare un input con l'operazione AWS IoT Events `BatchPutMessage` API, includi `iotevents-data:BatchPutMessage` nella sua politica. Le istruzioni delle policy devono includere un elemento `Action` o `NotAction`. AWS IoT Events definisce un proprio set di operazioni che descrivono le attività che puoi eseguire con questo servizio.

Per specificare più operazioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

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

È possibile specificare più operazioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le operazioni che iniziano con la parola `Describe`, includi la seguente operazione:

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

Per un elenco di operazioni di AWS IoT Events, consulta [Operazioni definite da AWS IoT Events](#) nella Guida per l'utente IAM.

Risorse

L'elemento `Resource` specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Specifica una risorsa utilizzando un ARN o il carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

La risorsa del modello del AWS IoT Events rilevatore ha il seguente ARN:

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

Per ulteriori informazioni sul formato di ARN, consulta [Amazon Resource Name \(ARN\) e spazi dei nomi del servizio AWS](#).

Ad esempio, per specificare il modello del Foobar rilevatore nella dichiarazione, utilizzate il seguente ARN:

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

Per specificare tutti le istanze che appartengono ad un account specifico, utilizza il carattere jolly (*):

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

Alcune operazioni AWS IoT Events, ad esempio quelle per la creazione di risorse, non possono essere eseguite su una risorsa specifica. In questi casi, è necessario utilizzare il carattere jolly (*).

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

Molte operazioni API AWS IoT Events coinvolgono più risorse. Ad esempio, `CreateDetectorModel` fa riferimento agli input nelle sue istruzioni di condizione, quindi un utente deve disporre delle autorizzazioni per utilizzare l'input e il modello del rilevatore. Per specificare più risorse in una singola istruzione, separa gli ARN con le virgole.

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

Per un elenco di tipi di risorse di AWS IoT Events e i relativi ARN, consulta [Risorse definite da AWS IoT Events](#) nella Guida per l'utente IAM. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consulta [Operazioni definite da AWS IoT Events](#).

Chiavi di condizione

L'elemento `Condition` (o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition` è facoltativo. Puoi compilare espressioni

condizionali che utilizzano [operatori di condizione](#), ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione AND logica. Se specifichi più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione OR logica. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi concedere a un utente l'autorizzazione per accedere a una risorsa solo se è stata taggata con il proprio nome utente. Per ulteriori informazioni, consulta [Elementi delle policy IAM: variabili e tag](#) nella Guida per l'utente di IAM.

AWS IoT Events non fornisce chiavi di condizione specifiche del servizio, ma supporta l'utilizzo di alcune chiavi di condizione globali. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di [contesto delle condizioni AWS globali nella Guida](#) per l'utente IAM.»

Esempi

Per visualizzare esempi di policy basate su identità AWS IoT Events, consulta [Esempi di policy di AWS IoT Events basate su identità](#).

Policy di AWS IoT Events basate sulle risorse

AWS IoT Events non supporta policy basate su risorse." Per visualizzare un esempio di una pagina di policy basata su risorse dettagliata, consulta <https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html>.

Autorizzazione basata su tag AWS IoT Events

Puoi collegare i tag alle risorse AWS IoT Events o passare i tag in una richiesta a AWS IoT Events. Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'[elemento condizione](#) di una policy utilizzando le chiavi di condizione `iotevents:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`. Per ulteriori informazioni sul tagging delle risorse di AWS IoT Events, consulta [Tagging delle risorse AWS IoT Events](#).

Per visualizzare una policy basata sulle identità di esempio per limitare l'accesso a una risorsa basata su tag su tale risorsa, consulta [Visualizzazione degli input in base ai tag AWS IoT Events](#).

Ruoli IAM di AWS IoT Events

Un [ruolo IAM](#) è un'entità all'interno di Account AWS che dispone di autorizzazioni specifiche.

Utilizzo di credenziali temporanee con AWS IoT Events

È possibile utilizzare credenziali temporanee per effettuare l'accesso con la federazione, assumere un ruolo IAM o un ruolo multi-account. È possibile ottenere credenziali di sicurezza temporanee chiamando operazioni API AWS Security Token Service (AWS STS) come [AssumeRole](#) o [GetFederationToken](#).

AWS IoT Events non supporta l'utilizzo di credenziali temporanee.

Ruoli collegati ai servizi

[Ruoli collegati al servizio](#) consentono ai servizi AWS di accedere a risorse in altri servizi per completare un'operazione a tuo nome. I ruoli collegati ai servizi sono visualizzati nell'account IAM e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non può modificarle.

AWS IoT Events attualmente non supporta i ruoli collegati ai servizi.

Ruoli dei servizi

Questa caratteristica consente a un servizio di assumere un [ruolo di servizio](#) per conto dell'utente. Questo ruolo consente al servizio di accedere alle risorse in altri servizi per completare un'operazione per conto dell'utente. I ruoli dei servizi sono visualizzati nell'account IAM e sono di proprietà dell'account. Ciò significa che un amministratore IAM può modificare le autorizzazioni per questo ruolo. Tuttavia, questo potrebbe pregiudicare la funzionalità del servizio.

AWS IoT Events supporta i ruoli dei servizi.

Esempi di policy di AWS IoT Events basate su identità

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse AWS IoT Events. Inoltre, non sono in grado di eseguire attività utilizzando la AWS Management Console, AWS CLI o un'API AWS. Un amministratore IAM deve creare policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore deve quindi collegare queste policy a utenti o gruppi che richiedono tali autorizzazioni.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consultare [Creazione di policy nella scheda JSON](#) nella Guida per l'utente di IAM.

Argomenti

- [Best practice delle policy](#)
- [Utilizzo della console di AWS IoT Events](#)
- [Consentire agli utenti di visualizzare le loro autorizzazioni](#)
- [Accedere a un input AWS IoT Events](#)
- [Visualizzazione degli input in base ai tag AWS IoT Events](#)

Best practice delle policy

Le policy basate su identità sono molto efficaci. Determinano se qualcuno può creare, accedere o eliminare risorse AWS IoT Events nell'account. Queste operazioni possono comportare costi aggiuntivi per il proprio Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Guida introduttiva all'uso delle policy gestite da AWS: per iniziare a utilizzare AWS IoT Events rapidamente, utilizza policy gestite da AWS per fornire ai dipendenti le autorizzazioni richieste. Queste policy sono già disponibili nell'account e sono gestite e aggiornate da AWS. Per ulteriori informazioni, consulta [Nozioni di base sull'utilizzo delle autorizzazioni con policy gestite da AWS](#) nella Guida per l'utente di IAM.
- Assegnare il privilegio minimo: quando si creano policy personalizzate, concedere solo le autorizzazioni richieste per eseguire un'attività. Inizia con un set di autorizzazioni minimo e concedi autorizzazioni aggiuntive quando necessario. Questo è più sicuro che iniziare con autorizzazioni che siano troppo permissive e cercare di limitarle in un secondo momento. Per ulteriori informazioni, consulta [Assegnare il privilegio minimo](#) nella Guida per l'utente IAM.
- Abilita l'MFA per operazioni sensibili: per una maggiore sicurezza, richiedi agli utenti di utilizzare l'autenticazione a più fattori (MFA) per accedere a risorse sensibili o operazioni API. Per ulteriori informazioni, consulta [Utilizzo dell'autenticazione a più fattori \(MFA\) in AWS](#) nella Guida per l'utente IAM.
- Utilizzare le condizioni della policy per ulteriore sicurezza – Per quanto possibile, definire le condizioni in cui le policy basate su identità consentono l'accesso a una risorsa. Ad esempio, è possibile scrivere condizioni per specificare un intervallo di indirizzi IP consentiti dai quali deve

provenire una richiesta. È anche possibile scrivere condizioni per consentire solo le richieste all'interno di un intervallo di date o ore specificato oppure per richiedere l'utilizzo di SSL o MFA. Per ulteriori informazioni, consultare [Elementi delle policy JSON di IAM: Condizioni](#) nella Guida per l'utente di IAM.

Utilizzo della console di AWS IoT Events

Per accedere alla console AWS IoT Events è necessario disporre di un insieme di autorizzazioni minimo. Queste autorizzazioni devono consentire di elencare e visualizzare i dettagli relativi alle risorse AWS IoT Events nel tuo Account AWS. Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Per garantire che tali entità possano ancora utilizzare la console AWS IoT Events, collega anche la seguente policy gestita di AWS alle entità. Per ulteriori informazioni, consulta [Aggiungere autorizzazioni a un utente nella Guida per l'utente IAM](#):

```
{
  "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}"
    }
]
}

```

Non sono necessarie le autorizzazioni minime della console per gli utenti che effettuano chiamate solo alla AWS CLI o all'API AWS. Al contrario, puoi accedere solo alle operazioni che soddisfano l'operazione API che stai cercando di eseguire.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. La policy include le autorizzazioni per completare questa azione sulla console o a livello di programmazione utilizzando la AWS CLI o l'API AWS.

```

{
  "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": "*"
}
]
}

```

Accedere a un input AWS IoT Events

In questo esempio, vuoi concedere a un utente che ti appartiene Account AWS l'accesso a uno dei tuoi AWS IoT Events input, `exampleInput`. Desideri inoltre consentire all'utente di aggiungere, aggiornare ed eliminare gli input.

La policy concede le autorizzazioni `iotevents:ListInputs`, `iotevents:DescribeInput`, `iotevents:CreateInput`, `iotevents>DeleteInput` e `iotevents:UpdateInput` all'utente. Per un esempio di procedura dettagliata per Amazon Simple Storage Service (Amazon S3) che concede le autorizzazioni agli utenti e li testa utilizzando la console, [consulta Un esempio di procedura dettagliata](#): `Using user policy to control access to your bucket`.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListInputsInConsole",
      "Effect": "Allow",
      "Action": [
        "iotevents:ListInputs"
      ],
      "Resource": "arn:aws:iotevents:::*"
    },
    {
      "Sid": "ViewSpecificInputInfo",
      "Effect": "Allow",

```

```

    "Action": [
      "iotevents:DescribeInput"
    ],
    "Resource": "arn:aws:iotevents:::exampleInput"
  },
  {
    "Sid": "ManageInputs",
    "Effect": "Allow",
    "Action": [
      "iotevents:CreateInput",
      "iotevents>DeleteInput",
      "iotevents:DescribeInput",
      "iotevents:ListInputs",
      "iotevents:UpdateInput"
    ],
    "Resource": "arn:aws:iotevents:::exampleInput/*"
  }
]
}

```

Visualizzazione degli input in base ai tag AWS IoT Events

Puoi utilizzare le condizioni nella policy basata sulle identità per controllare l'accesso alle risorse di AWS IoT Events in base ai tag. Questo esempio mostra come è possibile creare una politica che consenta la visualizzazione di un *input*. Tuttavia, l'autorizzazione viene concessa solo se il tag di *input* Owner ha il valore del nome utente di quell'utente. Questa policy concede anche le autorizzazioni necessarie per completare questa azione nella console.

```

{
  "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}"}
      }
    ]
  }
```

Puoi collegare questa policy agli utenti nel tuo account. Se un utente denominato `richard-roe` tenta di visualizzare un AWS IoT Events *input*, l'*input* deve essere taggato `Owner=richard-roe` o `owner=richard-roe`. In caso contrario l'accesso è negato. La chiave di tag di condizione `Owner` corrisponde a `Owner` e `owner` perché i nomi delle chiavi di condizione non effettuano la distinzione tra maiuscole e minuscole. Per ulteriori informazioni, consulta la sezione [Elementi delle policy JSON di IAM: condizione](#) nella Guida per l'utente di IAM.

Prevenzione del problema "confused deputy" tra servizi

Note

- Il AWS IoT Events servizio consente ai clienti di utilizzare i ruoli solo per avviare azioni nello stesso account in cui è stata creata una risorsa. Ciò significa che con questo servizio non è possibile eseguire un attacco secondario confuso.
- Questa pagina serve ai clienti come riferimento per vedere come funziona il problema Confused Vice e può essere evitata nel caso in cui nel AWS IoT Events servizio fossero consentite risorse multiaccount.

Con "confused deputy" si intende un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire una certa operazione può costringere un'entità con più privilegi a eseguire tale operazione. In AWS, la rappresentazione cross-service può comportare il problema confused deputy. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per poterti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse del tuo account.

Si consiglia di utilizzare le chiavi di contesto delle condizioni globali [aws:SourceArn](#) e [aws:SourceAccount](#) nelle policy delle risorse per limitare le autorizzazioni con cui AWS IoT

Events fornisce un altro servizio alla risorsa. Se il valore `aws:SourceArn` non contiene l'ID account, ad esempio un ARN di un bucket Amazon S3, è necessario utilizzare entrambe le chiavi di contesto delle condizioni globali per limitare le autorizzazioni. Se si utilizzano entrambe le chiavi di contesto delle condizioni globali e il valore `aws:SourceArn` contiene l'ID account, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` deve utilizzare lo stesso ID account nella stessa dichiarazione di policy.

Utilizzare `aws:SourceArn` se si desidera consentire l'associazione di una sola risorsa all'accesso tra servizi. Utilizzare `aws:SourceAccount` se si desidera consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi. Il valore di `aws:SourceArn` deve essere il modello di rilevatore o il modello di allarme associato alla `sts:AssumeRole` richiesta.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Se non si conosce l'ARN completo della risorsa o si scelgono più risorse, è necessario utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:iotevents:*:123456789012:*`.

Gli esempi seguenti mostrano come utilizzare le chiavi di contesto `aws:SourceArn` e `aws:SourceAccount` global condition AWS IoT Events per prevenire il confuso problema del vice.

Argomenti

- [Esempio 1: Accesso a un modello di rilevatore](#)
- [Esempio 2: Accesso a un modello di allarme](#)
- [Esempio 3: Accesso a una risorsa in una regione specificata](#)
- [Esempio 4: opzioni di registrazione](#)

Esempio 1: Accesso a un modello di rilevatore

Il seguente ruolo può essere utilizzato solo per accedere a un `DetectorModel` nomefoo.

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

```

Esempio 2: Accesso a un modello di allarme

Il seguente ruolo può essere utilizzato solo per accedere a qualsiasi modello di allarme.

```

{
  "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/*"
        }
      }
    }
  ]
}

```

Esempio 3: Accesso a una risorsa in una regione specificata

L'esempio seguente mostra un ruolo che è possibile utilizzare per accedere a una risorsa in una regione specificata. La regione in questo esempio è *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:*"
        }
      }
    }
  ]
}
```

Esempio 4: opzioni di registrazione

Per fornire un ruolo per le opzioni di registrazione, è necessario consentire che venga assunto per ogni risorsa in IoT Events. Di conseguenza, dovrai utilizzare un carattere jolly (*) per il tipo di risorsa e il nome della risorsa.

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

Risoluzione dei problemi di identità e accesso in AWS IoT Events

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di AWS IoT Events e di IAM.

Argomenti

- [Non sono autorizzato a eseguire un'operazione in AWS IoT Events](#)
- [Non sono autorizzato a eseguire iam:PassRole](#)
- [Voglio consentire alle persone esterne al mio account Account AWS di accedere alle mie risorse AWS IoT Events](#)

Non sono autorizzato a eseguire un'operazione in AWS IoT Events

Se la AWS Management Console indica che non hai l'autorizzazione a eseguire un'operazione, devi contattare l'amministratore per ricevere assistenza. L'amministratore è la persona da cui si sono ricevuti il nome utente e la password.

Il seguente errore di esempio si verifica quando l'utente mateojackson IAM tenta di utilizzare la console per visualizzare i dettagli su un *input* ma non dispone iotevents:*ListInputs* delle autorizzazioni.

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

In questo caso, Mateo richiede al suo amministratore di aggiornare le policy per poter accedere alla risorsa *my-example-input* utilizzando l'operazione `iotevents:ListInput`.

Non sono autorizzato a eseguire `iam:PassRole`

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a AWS IoT Events.

Alcuni Servizi AWS consentono di passare un ruolo esistente a tale servizio, invece di creare un nuovo ruolo di servizio o un ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in AWS IoT Events. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

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

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire alle persone esterne al mio account Account AWS di accedere alle mie risorse AWS IoT Events

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per capire se AWS IoT Events supporta queste funzionalità, consulta [Funzionamento di AWS IoT Events con IAM](#).

- Per informazioni su come garantire l'accesso alle risorse negli Account AWS che possiedi, consulta [Fornire l'accesso a un utente IAM in un altro Account AWS in tuo possesso](#) nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso alle risorse ad Account AWS di terze parti, consulta [Fornire l'accesso agli Account AWS di proprietà di terze parti](#) nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta [Fornire l'accesso a utenti autenticati esternamente \(Federazione delle identità\)](#) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consultare [Differenza tra i ruoli IAM e le policy basate su risorse](#) nella Guida per l'utente di IAM.

Monitoraggio di AWS IoT Events

Il monitoraggio è importante per garantire l'affidabilità, la disponibilità e le prestazioni di AWS IoT Events e delle soluzioni AWS. È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica. Prima di iniziare il monitoraggio di AWS IoT Events, è opportuno creare un piano di monitoraggio che includa le risposte alle seguenti domande:

- Quali sono gli obiettivi del monitoraggio?
- Quali risorse verranno monitorate?
- Con quale frequenza eseguirai il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno usati?
- Chi eseguirà i processi di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

La fase successiva consiste nello stabilire una baseline per le prestazioni normali di AWS IoT Events nell'ambiente, misurando le prestazioni in diversi momenti e con condizioni di carico differenti.

Quando monitori AWS IoT Events, archivia i dati di monitoraggio storici per poterli confrontare con i dati sulle prestazioni correnti e per poter identificare i normali modelli di prestazioni e le anomalie e ideare metodi per risolvere i problemi.

Ad esempio, se utilizzi Amazon EC2, puoi monitorare l'utilizzo della CPU, l'I/O del disco e l'utilizzo della rete per le tue istanze. Quando le prestazioni non rientrano nella baseline stabilita, può essere

necessario riconfigurare o ottimizzare l'istanza per ridurre l'utilizzo della CPU, migliorare l'I/O su disco o ridurre il traffico di rete.

Argomenti

- [Strumenti di monitoraggio](#)
- [Monitoraggio con Amazon CloudWatch](#)
- [Registrazione delle chiamate API AWS IoT Events con AWS CloudTrail](#)

Strumenti di monitoraggio

AWS offre vari strumenti che puoi utilizzare per monitorare AWS IoT Events. Alcuni di questi strumenti possono essere configurati in modo che eseguano automaticamente il monitoraggio, mentre altri richiedono l'intervento manuale. Si consiglia di automatizzare il più possibile i processi di monitoraggio.

Strumenti di monitoraggio automatici

Per controllare AWS IoT Events e segnalare l'eventuale presenza di problemi, puoi usare gli strumenti di monitoraggio automatici seguenti:

- Amazon CloudWatch Logs: monitora, archivia e accedi ai tuoi file di registro da o altre fonti. AWS CloudTrail Per ulteriori informazioni, consulta [Monitoring log files](#) nella Amazon CloudWatch User Guide.
- Amazon CloudWatch Events: abbina gli eventi e indirizzali a una o più funzioni o flussi di destinazione per apportare modifiche, acquisire informazioni sullo stato e intraprendere azioni correttive. Per ulteriori informazioni, consulta [What is Amazon CloudWatch Events](#) nella Amazon CloudWatch User Guide.
- AWS CloudTrail Monitoraggio dei log: condividi i file di CloudTrail registro tra account, monitora i file di registro in tempo reale inviandoli a CloudWatch Logs, scrivi applicazioni di elaborazione dei log in Java e verifica che i file di registro non siano cambiati dopo la consegna da parte di. CloudTrail Per ulteriori informazioni, consulta [Lavorare con i file di CloudTrail registro](#) nella Guida per l'utente. AWS CloudTrail

Strumenti di monitoraggio manuali

Un'altra parte importante del monitoraggio AWS IoT Events consiste nel monitorare manualmente gli elementi che gli CloudWatch allarmi non coprono. Le dashboard della AWS console AWS IoT

Events CloudWatch, e altre, forniscono una at-a-glance panoramica dello stato dell'ambiente AWS. Ti consigliamo anche di controllare i file di log in AWS IoT Events.

- La console AWS IoT Events mostra:
 - Modelli di rilevatore
 - Rilevatori
 - Input
 - Impostazioni
- La CloudWatch home page mostra:
 - Stato e allarmi attuali
 - Grafici degli allarmi e delle risorse
 - Stato di integrità dei servizi

Inoltre, è possibile utilizzare CloudWatch per effettuare le seguenti operazioni:

- Crea [pannelli di controllo personalizzati](#) per monitorare i servizi di interesse.
- Creare grafici dei dati dei parametri per la risoluzione di problemi e il rilevamento di tendenze.
- Ricercare e analizzare tutti i parametri delle risorse AWS
- Creare e modificare gli allarmi per ricevere le notifiche dei problemi.

Monitoraggio con Amazon CloudWatch

Quando sviluppi o esegui il debug di un modello di AWS IoT Events rilevatore, devi sapere cosa AWS IoT Events sta facendo e gli eventuali errori riscontrati. Amazon CloudWatch monitora le tue risorse Amazon Web Services (AWS) e le applicazioni su cui esegui AWS in tempo reale. Con CloudWatch, ottieni visibilità a livello di sistema sull'uso delle risorse, sulle prestazioni delle applicazioni e sullo stato operativo. [Abilita la CloudWatch registrazione di Amazon durante lo sviluppo di modelli di AWS IoT Events rilevatori](#) contiene informazioni su come abilitare la CloudWatch registrazione per. AWS IoT Events Per generare registri come quello mostrato di seguito, è necessario impostare il livello di verbosità su «Debug» e fornire uno o più obiettivi di debug, tra cui un nome del modello del rilevatore e un elemento opzionale. KeyValue

L'esempio seguente mostra una voce di registro di livello CloudWatch DEBUG generata da. 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
  }
]
```

}

Registrazione delle chiamate API AWS IoT Events con AWS CloudTrail

AWS IoT Events è integrato con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, ruolo o AWS servizio in AWS IoT Events. CloudTrail acquisisce tutte le chiamate API AWS IoT Events come eventi, incluse le chiamate dalla AWS IoT Events console e dalle chiamate in codice alle AWS IoT Events API.

Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per AWS IoT Events. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare a quale richiesta è stata inviata AWS IoT Events, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta la [Guida AWS CloudTrail per l'utente](#).

AWS IoT Events informazioni in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in AWS IoT Events, tale attività viene registrata in un CloudTrail evento con altri eventi di AWS servizio nella Cronologia degli eventi. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, vedere [Visualizzazione degli eventi con la cronologia degli CloudTrail eventi](#).

Per una registrazione continua degli eventi nell'account AWS che includa gli eventi per AWS IoT Events, creare un trail. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le Regioni nella partizione AWS e distribuisce i file di log nel bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei CloudTrail log. Per ulteriori informazioni, consultare:

- [Panoramica della creazione di un percorso](#)
- [CloudTrail servizi e integrazioni supportati](#)
- [Configurazione delle notifiche Amazon SNS per CloudTrail](#)
- [Ricezione di file di CloudTrail registro da più regioni](#) e [ricezione di file di CloudTrail registro da più account](#)

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali dell'utente IAM o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Per ulteriori informazioni, vedete l'elemento [CloudTrail userIdentity](#). AWS IoT Events le azioni sono documentate nel riferimento all'[AWS IoT Events API](#).

Comprensione delle voci dei file di log di AWS IoT Events

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. AWS CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

Quando CloudTrail la registrazione è abilitata nell'AWS account, la maggior parte delle chiamate API effettuate alle AWS IoT Events azioni vengono registrate nei file di CloudTrail registro, dove vengono scritte insieme ad altri AWS record di servizio. CloudTrail determina quando creare e scrivere su un nuovo file in base al periodo di tempo e alle dimensioni del file.

Ogni voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni sull'identità dell'utente nella voce di log ti permettono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali dell'utente IAM o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Puoi archiviare i file di log nel tuo bucket Amazon S3 per tutto il tempo che desideri, ma puoi anche definire regole del ciclo di vita di Amazon S3 per archiviare o eliminare automaticamente i file di registro. Per impostazione predefinita, i file di registro sono crittografati con la crittografia lato server (SSE) di Amazon S3.

Per ricevere una notifica al momento della consegna dei file di log, puoi CloudTrail configurare la pubblicazione di notifiche Amazon SNS quando vengono consegnati nuovi file di log. Per ulteriori informazioni, consulta l'argomento relativo alla [configurazione delle notifiche Amazon SNS per CloudTrail](#).

Puoi inoltre aggregare file di log AWS IoT Events da più Regioni AWS e da più account AWS in un singolo bucket Amazon S3.

Per ulteriori informazioni, consulta [Ricezione di file di CloudTrail registro da più regioni](#) e [Ricezione di file di CloudTrail registro da più account](#).

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'DescribeDetectorazione.

```
{
  "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"
}
```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'CreateDetectorModelazione.

```
{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'CreateInputazione.

```

{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l>DeleteDetectorModelazione.

```

{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'DeleteInputazione.

```

{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'DescribeDetectorModelazione.

```

{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'DescribeInputazione.

```

{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'DescribeLoggingOptionsazione.

```

{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'ListDetectorModelizzazione.

```

{
  "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": "CkZEZXRlY3Rvck1vZGVsMl9saXN0ZGV0ZWN0b3Jtb2R1bHN0ZXN0X2VlOWJkZTk1YT",
  "maxResults": 3
},
"responseElements": null,
"requestID": "6d70f262-da95-4bb5-94b4-c08369df75bb",
"eventID": "2d01a25c-d5c7-4233-99fe-ce1b8ec05516",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'ListDetectorModelVersionazione.

```

{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'ListDetector azione.

```

{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'ListInputsazione.

```

{
  "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": "CkhjYW5hcnlfdGVzdF9pbnB1dF9saXN0ZGV0ZWNo0b3Jtb2R1bHN0ZXN0ZDU3OGZ",
  "maxResults": 3
},
"responseElements": null,
"requestID": "dd6762a1-1f24-4e63-a986-5ea3938a03da",
"eventID": "c500f6d8-e271-4366-8f20-da4413752469",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'PutLoggingOptionsazione.

```

{
  "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"
}
```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'UpdateDetectorModelazione.

```
{
  "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"
}

```

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'UpdateInputazione.

```

{
  "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"
}
```


Convalida della conformità per AWS IoT Events

Per sapere se il Servizio AWS è coperto da programmi di conformità specifici, consulta i [Servizi AWS coperti dal programma di conformità](#) e scegli il programma di conformità desiderato. Per informazioni generali, consulta [Programmi per la conformità di AWS](#).

È possibile scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta [Download di report in AWS Artifact](#).

La responsabilità di conformità durante l'utilizzo dei Servizi AWS è determinata dalla riservatezza dei dati, dagli obiettivi di conformità dell'azienda e dalle normative vigenti. Per semplificare il rispetto della conformità, AWS mette a disposizione le seguenti risorse:

- [Guide Quick Start per la sicurezza e conformità](#): queste guide all'implementazione illustrano considerazioni relative all'architettura e forniscono la procedura per l'implementazione di ambienti di base su AWS incentrati sulla sicurezza e sulla conformità.
- [Architetture per la sicurezza e la conformità HIPAA su Amazon Web Services](#): questo whitepaper descrive come le aziende possono utilizzare AWS per creare applicazioni conformi alla normativa HIPAA.

 Note

Non tutti i Servizi AWS sono conformi ai requisiti HIPAA. Per ulteriori informazioni, consulta la sezione [Riferimenti sui servizi conformi ai requisiti HIPAA](#).

- [Risorse per la conformità AWS](#): una raccolta di cartelle di lavoro e guide suddivise per settore e area geografica.
- [AWS Guide alla conformità dei clienti](#): comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- [Valutazione delle risorse con le regole](#) nella Guida per gli sviluppatori di AWS Config: il servizio AWS Config valuta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti.
- [AWS Security Hub](#): questo Servizio AWS fornisce una visione completa dello stato di sicurezza all'interno di AWS. La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWS e verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina [Documentazione di riferimento sui controlli della Centrale di sicurezza](#).
- [AWS Audit Manager](#): questo Servizio AWS aiuta a verificare continuamente l'utilizzo di AWS per semplificare la gestione dei rischi e della conformità alle normative e agli standard di settore.

Resilienza in AWS IoT Events

L'infrastruttura globale di AWS è basata su Regioni e zone di disponibilità AWS. AWS Le Regioni forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e velocità effettiva elevata. Con le zone di disponibilità, è possibile progettare e gestire le applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo.

Per ulteriori informazioni sulle Regioni e le Zone di disponibilità AWS, consulta [Infrastruttura globale di AWS](#).

Sicurezza dell'infrastruttura in AWS IoT Events

Come servizio gestito, AWS IoT Events è protetto dalla sicurezza di rete globale AWS. Per informazioni sui servizi di sicurezza AWS e su come AWS protegge l'infrastruttura, consulta la pagina [Sicurezza del cloud AWS](#). Per progettare l'ambiente AWS utilizzando le best practice per la sicurezza dell'infrastruttura, consulta la pagina [Protezione dell'infrastruttura](#) nel Pilastro della sicurezza di AWS Well-Architected Framework.

Utilizza le chiamate API pubblicate da AWS per accedere tramite la rete. I clienti devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, è possibile utilizzare [AWS Security Token Service](#) (AWS STS) per generare le credenziali di sicurezza temporanee per sottoscrivere le richieste.

Quote AWS IoT Events

La Riferimenti generali di AWSGuida fornisce le quote predefinite AWS IoT Events per un AWS account. Se non diversamente specificato, ogni quota è per AWS regione. Per ulteriori informazioni, consulta [AWS IoT EventsEndpoints and Quotas](#) e [Service AWS Quotas](#) nella Guida. Riferimenti generali di AWS

Per richiedere un aumento della quota di servizio, invia una richiesta di supporto nella console del [Support center](#). Per ulteriori informazioni, consulta [Richiesta di un aumento di quota](#) nella Guida per l'utente per Service Quotas.

Note

- Tutti i nomi dei modelli e degli ingressi del rilevatore devono essere univoci all'interno di un account.
- Non è possibile modificare i nomi dei modelli e degli ingressi del rilevatore dopo la loro creazione.

Tagging delle risorse AWS IoT Events

Per aiutarvi a gestire e organizzare i modelli e gli input dei rilevatori, potete facoltativamente assegnare i vostri metadati a ciascuna di queste risorse sotto forma di tag. Questa sezione descrive i tag e mostra come crearli.

Nozioni di base sui tag

I tag consentono di categorizzare le tue risorse AWS IoT Events in modi diversi, ad esempio, per scopo, proprietario o ambiente. Questa funzione è utile quando si dispone di numerose risorse dello stesso tipo. Puoi identificare velocemente una risorsa specifica in base ai tag a questa assegnati.

Ogni tag è formato da una chiave e da un valore opzionale, entrambi personalizzabili. Ad esempio, potreste definire un set di tag per gli input che vi aiutino a tracciare i dispositivi che inviano questi input in base al tipo. Ti consigliamo di creare un set di chiavi di tag in grado di soddisfare i requisiti di ciascun tipo di risorsa. Tramite un set di chiavi di tag coerente la gestione delle risorse risulta notevolmente semplificata.

Puoi cercare e filtrare le risorse in base ai tag che aggiungi o applichi, utilizzare i tag per classificare e tenere traccia dei costi e anche utilizzare i tag per controllare l'accesso alle risorse, come descritto in [Utilizzo dei tag con le politiche IAM](#) nella Developer Guide. AWS IoT

Per una maggiore facilità d'uso, il Tag Editor integrato AWS Management Console fornisce un modo centralizzato e unificato per creare e gestire i tag. Per ulteriori informazioni, consulta [Lavorare con Tag Editor](#) in [Lavorare con](#). AWS Management Console

Puoi lavorare con i tag utilizzando AWS CLI e l'API AWS IoT Events. È possibile associare i tag ai modelli e agli input del rilevatore quando li si crea utilizzando il "Tags" campo nei seguenti comandi:

- [CreateDetectorModel](#)
- [CreateInput](#)

Puoi aggiungere, modificare o eliminare i tag per le risorse esistenti che supportano il tagging utilizzando i comandi seguenti:

- [TagResource](#)
- [ListTagsForResource](#)

- [UntagResource](#)

Puoi modificare chiavi e valori di tag e rimuovere tag da una risorsa in qualsiasi momento. Puoi impostare il valore di un tag su una stringa vuota, ma non su null. Se aggiungi un tag con la stessa chiave di un tag esistente a una risorsa specifica, il nuovo valore sovrascrive quello precedente. Se elimini una risorsa, verranno eliminati anche tutti i tag associati alla risorsa.

Ulteriori informazioni sono disponibili nelle strategie di [AWS etichettatura](#).

Restrizioni e limitazioni di tag

Si applicano le seguenti limitazioni di base ai tag:

- Numero massimo di tag per risorsa: 50
- Lunghezza massima della chiave: 127 caratteri Unicode in UTF-8
- Lunghezza massima del valore: 255 caratteri Unicode in UTF-8
- i valori e le chiavi dei tag rispettano la distinzione tra maiuscole e minuscole;
- Non utilizzate il "aws:" prefisso nei nomi o nei valori dei tag perché è riservato all'uso. AWS Non è possibile modificare né eliminare i nomi o i valori di tag con tale prefisso. I tag con questo prefisso non vengono conteggiati per il limite del numero di tag per risorsa.
- Se lo schema di tagging viene utilizzato in più servizi e risorse, è necessario tenere presente che in altri servizi possono essere presenti limiti sui caratteri consentiti. In generale, i caratteri consentiti sono in genere lettere, spazi e numeri rappresentabili in formato UTF-8, più i caratteri speciali + - = . _ : / @.

Utilizzo dei tag con policy IAM

È possibile applicare autorizzazioni a livello di risorsa basate su tag nelle policy IAM utilizzate per le operazioni API AWS IoT Events. In questo modo è possibile controllare meglio le risorse che un utente può creare, modificare o utilizzare.

Puoi utilizzare l'elemento Condition (denominato anche blocco Condition) con i seguenti valori e chiavi di contesto di condizione in una policy IAM per controllare l'accesso dell'utente (autorizzazione) in base ai tag della risorsa:

- Utilizza `aws:ResourceTag/<tag-key>: <tag-value>` per concedere o negare agli utenti operazioni su risorse con specifici tag.

- Utilizza `aws:RequestTag/<tag-key>: <tag-value>` per richiedere che un tag specifico venga utilizzato (o non utilizzato) durante la creazione di una richiesta API per creare o modificare una risorsa che abilita i tag.
- Utilizza `aws:TagKeys: [<tag-key>, ...]` per richiedere che un set di tag specifico venga utilizzato (o non utilizzato) durante la creazione di una richiesta API per creare o modificare una risorsa che abilita i tag.

Note

Le chiavi di contesto della condizione e i valori all'interno di una policy IAM si applicano solo alle operazioni AWS IoT Events in cui un identificatore per una risorsa in grado di essere taggata è un parametro obbligatorio.

[Il controllo dell'accesso tramite i tag](#) nella Guida AWS Identity and Access Management per l'utente contiene informazioni aggiuntive sull'uso dei tag. La sezione di [riferimento alle policy JSON di IAM](#) di quella guida contiene sintassi, descrizioni ed esempi dettagliati degli elementi, delle variabili e della logica di valutazione delle policy JSON in IAM.

La policy di esempio seguente applica due restrizioni basate su tag. Un utente limitato da questa politica:

- Non può assegnare a una risorsa il tag "env = prod" (nell'esempio, consulta la riga `"aws:RequestTag/env" : "prod"`)
- Non può modificare o accedere a una risorsa con un tag esistente "env = prod" (nell'esempio, consulta la riga `"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": "*"
  }
]
}

```

Puoi anche specificare più valori di tag per una determinata chiave di tag racchiudendoli in un elenco, come segue.

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

Note

Se consenti o neghi a un utente l'accesso a risorse in base ai tag, devi considerare esplicitamente di negare agli utenti la possibilità di aggiungere o rimuovere tali tag dalle stesse risorse. In caso contrario, un utente può eludere le restrizioni e ottenere l'accesso a una risorsa modificandone i tag.

Risoluzione dei problemi AWS IoT Events

Utilizza le informazioni contenute in queste sezioni per risolvere i problemi relativi a AWS IoT Events.

Argomenti

- [AWS IoT Events Problemi e soluzioni comuni](#)
- [Risoluzione dei problemi di un modello di rilevatore eseguendo analisi](#)

AWS IoT Events Problemi e soluzioni comuni

Consulta la sezione seguente per risolvere gli errori e trovare le possibili soluzioni per risolverli. AWS IoT Events

Errori

- [Errori di creazione del modello di rilevatore](#)
- [Aggiornamenti da un modello di rilevatore eliminato](#)
- [Errore nell'attivazione dell'azione \(quando viene soddisfatta una condizione\)](#)
- [Errore nell'attivazione dell'azione \(quando si supera una soglia\)](#)
- [Utilizzo errato dello stato](#)
- [Messaggio di connessione](#)
- [InvalidRequestException messaggio](#)
- [Errori di Amazon CloudWatch Logs action.setTimer](#)
- [Errori del CloudWatch payload di Amazon](#)
- [Tipi di dati incompatibili](#)
- [Impossibile inviare il messaggio a AWS IoT Events](#)

Errori di creazione del modello di rilevatore

Quando tento di creare un modello di rilevatore, ricevo degli errori.

Soluzione

Quando si crea un modello di rilevatore, è necessario considerare le seguenti limitazioni.

- È consentita una sola azione in ogni `action` campo.
- `condition` È richiesto per `transitionEvents`. È facoltativo per `OnEnterOnInput`, ed `OnExit` eventi.
- Se il `condition` campo è vuoto, il risultato valutato dell'espressione della condizione è equivalente a `true`.
- Il risultato valutato dell'espressione della condizione deve essere un valore booleano. Se il risultato non è un valore booleano, è equivalente a `false` e non attiva la transizione `actions` o verso il valore specificato nell'`nextState` evento.

Per ulteriori informazioni, consulta [Restrizioni e limitazioni del modello di rilevatore](#).

Aggiornamenti da un modello di rilevatore eliminato

Ho aggiornato o eliminato un modello di rilevatore qualche minuto fa, ma continuo a ricevere aggiornamenti sullo stato del vecchio modello di rilevatore tramite messaggi MQTT o avvisi SNS.

Soluzione

Se si aggiorna, si elimina o si ricrea un modello di rilevatore (vedi [UpdateDetectorModel](#)), c'è un ritardo prima che tutte le istanze del rilevatore vengano eliminate e venga utilizzato il nuovo modello. Durante questo periodo, gli input potrebbero continuare a essere elaborati dalle istanze della versione precedente del modello di rilevatore. È possibile continuare a ricevere avvisi definiti dal modello di rilevatore precedente. Attendi almeno sette minuti prima di ricontrollare l'aggiornamento o segnalare un errore.

Errore nell'attivazione dell'azione (quando viene soddisfatta una condizione)

Il rilevatore non riesce ad attivare un'azione o passare a un nuovo stato quando la condizione è soddisfatta.

Soluzione

Verificate che il risultato valutato dell'espressione condizionale del rilevatore sia un valore booleano. Se il risultato non è un valore booleano, è equivalente a `false` e non attiva la transizione `action` o verso il valore specificato nell'evento. `nextState` Per ulteriori informazioni, consulta Sintassi delle espressioni [condizionali](#).

Errore nell'attivazione dell'azione (quando si supera una soglia)

Il rilevatore non attiva un'azione o una transizione di eventi quando la variabile in un'espressione condizionale raggiunge un valore specificato.

Soluzione

Se si aggiorna `setVariable` per `onInput`, o `onEnteronExit`, il nuovo valore non viene utilizzato per valutarne uno `condition` durante il ciclo di elaborazione corrente. Al contrario, il valore originale viene utilizzato fino al completamento del ciclo corrente. È possibile modificare questo comportamento impostando il `evaluationMethod` parametro nella definizione del modello del rilevatore. Quando `evaluationMethod` è impostato su `SERIAL`, le variabili vengono aggiornate e le condizioni degli eventi valutate nell'ordine in cui gli eventi sono definiti. Quando `evaluationMethod` è impostata su `BATCH` (impostazione predefinita), le variabili vengono aggiornate e gli eventi vengono eseguiti solo dopo aver valutato tutte le condizioni dell'evento.

Utilizzo errato dello stato

Il rilevatore entra negli stati errati quando tento di inviare messaggi agli ingressi utilizzando.

`BatchPutMessage`

Soluzione

Se si utilizza l'[BatchPutMessage](#) opzione per inviare più messaggi agli input, l'ordine in cui i messaggi o gli input vengono elaborati non è garantito. Per garantire l'ordine, invia i messaggi uno alla volta e attendi ogni volta che confermi l'avvenuta `BatchPutMessage` operazione.

Messaggio di connessione

Ricevo un (`'Connection aborted.'`, `error(54, 'Connection reset by peer')`) errore quando tento di chiamare o richiamare un'API.

Soluzione

Verifica che OpenSSL utilizzi TLS 1.1 o una versione successiva per stabilire la connessione. Questa dovrebbe essere l'impostazione predefinita nella maggior parte delle distribuzioni Linux o Windows versione 7 e successive. Gli utenti di macOS potrebbero dover aggiornare OpenSSL.

InvalidRequestException messaggio

Ricevo `InvalidRequestException` quando tento di chiamare `CreateDetectorModel` e le `UpdateDetectorModel` API.

Soluzione

Controlla quanto segue per risolvere il problema. Per ulteriori informazioni, consultare [CreateDetectorModel](#) e [UpdateDetectorModel](#).

- Assicurati di non utilizzarli entrambi `seconds` e `durationExpression` come parametri `SetTimerAction` contemporaneamente.
- Assicurati che l'espressione stringa for `durationExpression` sia valida. L'espressione stringa può contenere numeri, variabili (`$variable.<variable-name>`) o valori di input (`$input.<input-name>.<path-to-datum>`).

Errori di Amazon CloudWatch Logs `action.setTimer`

Puoi configurare Amazon CloudWatch Logs per monitorare le istanze del modello di AWS IoT Events rilevatore. Di seguito sono riportati gli errori più comuni generati da AWS IoT Events, quando si utilizza `action.setTimer`

- Errore: l'espressione della durata per il timer denominato `<timer-name>` può essere convertita in un numero.

Soluzione

Assicurati che l'espressione stringa per `durationExpression` possa essere convertita in un numero. Altri tipi di dati, come quelli booleani, non sono consentiti.

- Errore: il risultato valutato dell'espressione di durata per il timer denominato `<timer-name>` è maggiore di 31622440. Per garantire la precisione, assicurati che l'espressione di durata si riferisca a un valore compreso tra 60-31622400.

Soluzione

Assicurati che la durata del timer sia inferiore o uguale a 31622400 secondi. Il risultato valutato della durata viene arrotondato per difetto al numero intero più vicino.

- Errore: il risultato valutato dell'espressione di durata per il timer denominato `<timer-name>` è inferiore a 60. Per garantire la precisione, assicurati che l'espressione di durata si riferisca a un valore compreso tra 60-31622400.

Soluzione

Assicurati che la durata del timer sia maggiore o uguale a 60 secondi. Il risultato valutato della durata viene arrotondato per difetto al numero intero più vicino.

- Errore: l'espressione della durata per il timer denominato non `<timer-name>` può essere valutata. Controlla i nomi delle variabili, i nomi di input e i percorsi dei dati per assicurarti di fare riferimento alle variabili e agli input esistenti.

Soluzione

Assicurati che l'espressione stringa si riferisca alle variabili e agli input esistenti. L'espressione stringa può contenere numeri, variabili (`$variable.variable-name`) e valori di input (`$input.input-name.path-to-datum`).

- Errore: impossibile impostare il timer denominato `<timer-name>`. Controlla l'espressione della durata e riprova.

Soluzione

Osservate l'[SetTimerAction](#) azione per assicurarvi di aver specificato i parametri corretti, quindi impostate nuovamente il timer.

Per ulteriori informazioni, consulta [CloudWatch Attivare la registrazione di Amazon durante lo sviluppo di modelli di AWS IoT Events rilevatori](#).

Errori del CloudWatch payload di Amazon

Puoi configurare Amazon CloudWatch Logs per monitorare le istanze del modello di AWS IoT Events rilevatore. Di seguito sono riportati gli errori e gli avvisi comuni generati da AWS IoT Events, quando configuri il payload dell'azione.

- Errore: non è stato possibile valutare la tua espressione per l'azione. Assicurati che i nomi delle variabili, i nomi di input e i percorsi dei dati si riferiscano alle variabili e ai valori di input esistenti. Inoltre, verificate che la dimensione del payload sia inferiore a 1 KB, la dimensione massima consentita di un payload.

Soluzione

Assicurati di inserire i nomi delle variabili, i nomi di input e i percorsi dei dati corretti. Potresti ricevere questo messaggio di errore anche se il payload dell'azione è maggiore di 1 KB.

- Errore: non è stato possibile analizzare la tua espressione di contenuto per il payload di. *<action-type>* Inserisci un'espressione di contenuto con la sintassi corretta.

Soluzione

L'espressione di contenuto può contenere stringhe ('*string*'), variabili (*()*), valori di input (*\$variable.variable-name*), concatenazioni di stringhe e stringhe che contengono.

\$input.input-name.path-to-datum \${}

- Errore: *l'espressione di payload {expression} non è valida*. Il tipo di payload definito è JSON, quindi è necessario specificare un'espressione che AWS IoT Events restituisca una stringa.

Soluzione

Se il tipo di payload specificato è JSON, verifica AWS IoT Events innanzitutto se il servizio è in grado di valutare l'espressione in una stringa. Il risultato valutato non può essere un valore booleano o un numero. Se la convalida fallisce, potresti ricevere questo errore.

- Avviso: l'azione è stata eseguita, ma non siamo riusciti a valutare l'espressione di contenuto per il payload dell'azione in un formato JSON valido. Il tipo di payload definito è JSON.

Soluzione

Assicurati che sia in AWS IoT Events grado di valutare l'espressione di contenuto per il payload dell'azione utilizzando un codice JSON valido, se definisci il tipo di payload come. JSON AWS IoT Events esegue l'azione anche se non AWS IoT Events riesce a valutare l'espressione di contenuto in un formato JSON valido.

Per ulteriori informazioni, consulta [CloudWatch Attivare la registrazione di Amazon durante lo sviluppo di modelli di AWS IoT Events rilevatori](#).

Tipi di dati incompatibili

Messaggio: tipi di dati non compatibili [*<inferred-types>*] trovati *<reference>* nella seguente espressione: *<expression>*

Soluzione

Potresti ricevere questo errore per uno dei seguenti motivi:

- I risultati valutati dei riferimenti non sono compatibili con altri operandi delle espressioni.
- Il tipo di argomento passato a una funzione non è supportato.

Quando utilizzate riferimenti nelle espressioni, controllate quanto segue:

- Quando utilizzate un riferimento come operando con uno o più operatori, assicuratevi che tutti i tipi di dati a cui fate riferimento siano compatibili.

Ad esempio, nell'espressione seguente, il numero intero 2 è un operando di entrambi gli == operatori and. && Per garantire che gli operandi siano compatibili `$variable.testVariable + 1` e che `$variable.testVariable` debbano fare riferimento a un numero intero o decimale.

Inoltre, il numero intero 1 è un operando dell'operatore. + Pertanto, `$variable.testVariable` deve fare riferimento a un numero intero o decimale.

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

- Quando utilizzate un riferimento come argomento passato a una funzione, assicuratevi che la funzione supporti i tipi di dati a cui fate riferimento.

Ad esempio, la seguente `timeout("time-name")` funzione richiede una stringa con virgolette doppie come argomento. Se si utilizza un riferimento per il valore `timer-name`, è necessario fare riferimento a una stringa tra virgolette doppie.

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

Note

Per la `convert(type, expression)` funzione, se si utilizza un riferimento per il valore del `tipo`, il risultato valutato del riferimento deve essere `String`, `Decimal` o `Boolean`.

Per ulteriori informazioni, consulta [Riferimenti](#).

Impossibile inviare il messaggio a AWS IoT Events

Messaggio: Impossibile inviare il messaggio a lot Events

Soluzione

È possibile che si verifichi questo errore per i seguenti motivi:

- Il payload del messaggio di input non contiene. `Input attribute Key`
- Non si `Input attribute Key` trova nello stesso percorso JSON specificato nella definizione di input.
- Il messaggio di input non corrisponde allo schema definito nell'AWS IoT Eventsinput.

Note

Anche l'inserimento di dati da altri servizi potrebbe fallire.

Example

Ad esempio inAWS IoT Core, la AWS IoT regola avrà esito negativo con il seguente messaggio `Verify the Input Attribute key`.

Per risolvere questo problema, assicuratevi che lo schema del messaggio di input del payload sia conforme alla definizione di AWS IoT Events input e che la `Input attribute Key` posizione corrisponda. Per ulteriori informazioni, consulta [the section called “Creare un input nel riquadro di navigazione”](#) per imparare a definire AWS IoT Events gli input.

Risoluzione dei problemi di un modello di rilevatore eseguendo analisi

AWS IoT Events può analizzare il modello del rilevatore e generare risultati di analisi senza inviare dati di input al modello del rilevatore. AWS IoT Events esegue una serie di analisi descritte in questa sezione per verificare il modello del rilevatore. Questa soluzione avanzata per la risoluzione dei problemi riassume anche le informazioni diagnostiche, tra cui il livello di gravità e la posizione, in modo da poter individuare e risolvere rapidamente potenziali problemi nel modello del rilevatore. Per ulteriori informazioni sui tipi e sui messaggi di errore diagnostici per il modello di rilevatore in uso, vedere. [Analisi del modello di rivelatore e informazioni diagnostiche](#)

È possibile utilizzare la AWS IoT Events console, l'[API](#), [AWS Command Line Interface\(AWS CLI\)](#) o l'[AWSSDK](#) per visualizzare i messaggi di errore diagnostici derivanti dall'analisi del modello di rilevatore.

Note

- È necessario correggere tutti gli errori prima di poter pubblicare il modello del rilevatore.
- Si consiglia di esaminare gli avvisi e di intraprendere le azioni necessarie prima di utilizzare il modello di rilevatore negli ambienti di produzione. In caso contrario, il modello del rilevatore potrebbe non funzionare come previsto.
- È possibile avere fino a 10 analisi nello RUNNING stato contemporaneamente.

Per informazioni su come analizzare il modello del rilevatore, consulta [Analisi di un modello di rilevatore \(console\)](#) o [Analisi di un modello di rilevatore \(\) AWS CLI](#)

Argomenti

- [Analisi del modello di rivelatore e informazioni diagnostiche](#)
- [Analisi di un modello di rilevatore \(console\)](#)
- [Analisi di un modello di rilevatore \(\) AWS CLI](#)

Analisi del modello di rivelatore e informazioni diagnostiche

Le analisi dei modelli di rivelatori raccolgono le seguenti informazioni diagnostiche:

- **Livello:** il livello di gravità del risultato dell'analisi. In base al livello di gravità, i risultati dell'analisi rientrano in tre categorie generali:
 - **Informazioni (INFO):** un risultato informativo indica un campo significativo nel modello del rilevatore. Questo tipo di risultato di solito non richiede un'azione immediata.
 - **Avviso (WARNING):** il risultato di un avviso richiama l'attenzione in particolare sui campi che potrebbero causare problemi al modello del rilevatore. Si consiglia di esaminare gli avvisi e intraprendere le azioni necessarie prima di utilizzare il modello di rilevatore negli ambienti di produzione. In caso contrario, il modello del rilevatore potrebbe non funzionare come previsto.
 - **Error (ERROR):** il risultato di un errore segnala un problema riscontrato nel modello del rilevatore. AWS IoT Events segue automaticamente questo set di analisi quando si tenta di pubblicare il

modello del rilevatore. È necessario correggere tutti gli errori prima di poter pubblicare il modello del rilevatore.

- **Posizione:** contiene informazioni che è possibile utilizzare per individuare il campo nel modello del rilevatore a cui fa riferimento il risultato dell'analisi. Una posizione include in genere il nome dello stato, il nome dell'evento di transizione, il nome dell'evento e l'espressione (ad esempio, `in state TemperatureCheck in onEnter in event Init in action setVariable`).
- **Tipo:** il tipo di risultato dell'analisi. I tipi di analisi rientrano nelle seguenti categorie:
 - **supported-actions**— AWS IoT Events può richiamare azioni quando viene rilevato un evento o un evento di transizione specificato. È possibile definire azioni integrate per utilizzare un timer o impostare una variabile o inviare dati ad altri AWS servizi. È necessario specificare azioni che funzionino con altri AWS servizi in una AWS regione in cui i AWS servizi sono disponibili.
 - **service-limits**— Le quote di servizio, note anche come limiti, sono il numero massimo o minimo di risorse o operazioni di servizio per l'AWS account. Salvo diversa indicazione, ogni quota si applica a una regione specifica. A seconda delle esigenze aziendali, è possibile aggiornare il modello di rilevatore per evitare di incontrare limiti o richiedere un aumento della quota. È possibile richiedere aumenti per alcune quote e altre quote non possono essere aumentate. Per ulteriori informazioni, consulta [Quote](#).
- **structure**— Il modello del rilevatore deve avere tutti i componenti richiesti, come gli stati, e seguire una struttura che li supporti. AWS IoT Events Un modello di rilevatore deve avere almeno uno stato e una condizione che valuti i dati di input in ingresso per rilevare eventi significativi. Quando viene rilevato un evento, il modello del rilevatore passa allo stato successivo e può richiamare azioni. Questi eventi sono noti come eventi di transizione. Un evento di transizione deve indicare l'ingresso dello stato successivo.
- **expression-syntax**— AWS IoT Events offre diversi modi per specificare i valori durante la creazione e l'aggiornamento dei modelli di rilevatori. È possibile utilizzare valori letterali, operatori, funzioni, riferimenti e modelli di sostituzione nelle espressioni. È possibile utilizzare le espressioni per specificare valori letterali o AWS IoT Events valutare le espressioni prima di specificare valori particolari. L'espressione deve seguire la sintassi richiesta. Per ulteriori informazioni, consulta [Espressioni](#).

Le espressioni del modello Detector in AWS IoT Events possono fare riferimento a dati specifici o a una risorsa.

- **data-type**— AWS IoT Events supporta tipi di dati interi, decimali, stringhe e booleani. Se AWS IoT Events è possibile convertire automaticamente i dati di un tipo di dati in un altro durante la valutazione delle espressioni, questi tipi di dati sono compatibili.

Note

- I numeri interi e decimali sono gli unici tipi di dati compatibili supportati da AWS IoT Events
 - AWS IoT Events non può valutare le espressioni aritmetiche perché non AWS IoT Events può convertire un numero intero in una stringa.
- **referenced-data**— È necessario definire i dati a cui si fa riferimento nel modello del rilevatore prima di poter utilizzare i dati. Ad esempio, se si desidera inviare dati a una tabella DynamoDB, è necessario definire una variabile che faccia riferimento al nome della tabella prima di poter utilizzare la variabile in un'espressione (). `$variable.TableName`
 - **referenced-resource**— Le risorse utilizzate dal modello di rilevatore devono essere disponibili. È necessario definire le risorse prima di poterle utilizzare. Ad esempio, volete creare un modello di rilevatore per monitorare la temperatura di una serra. È necessario definire un input (`$input.TemperatureInput`) per indirizzare i dati di temperatura in ingresso al modello del rilevatore prima di poterlo utilizzare per fare riferimento `$input.TemperatureInput.sensorData.temperature` alla temperatura.

Consultate la sezione seguente per risolvere gli errori e trovare possibili soluzioni mediante l'analisi del modello del rilevatore.

Risolvi gli errori del modello del rilevatore

I tipi di errori sopra descritti forniscono informazioni diagnostiche su un modello di rilevatore e corrispondono a messaggi che potresti recuperare. Utilizzate questi messaggi e le soluzioni suggerite per risolvere gli errori relativi al modello di rilevatore.

Messaggi e soluzioni

- [Location](#)
- [supported-actions](#)
- [service-limits](#)
- [structure](#)
- [expression-syntax](#)
- [data-type](#)
- [referenced-data](#)

- [referenced-resource](#)

Location

Un risultato dell'analisi con informazioni su `Location`, corrisponde al seguente messaggio di errore:

- Messaggio: contiene informazioni aggiuntive sul risultato dell'analisi. Può trattarsi di un messaggio informativo, di avviso o di errore.

Soluzione: potresti ricevere questo messaggio di errore se hai specificato un'azione che AWS IoT Events al momento non supporta. Per un elenco delle azioni supportate, consulta [Azioni supportate](#).

supported-actions

Un risultato dell'analisi con informazioni su `supported-actions`, corrisponde ai seguenti messaggi di errore:

- Messaggio: tipo di azione non valido presente nella definizione dell'azione: *action-definition*.

Soluzione: potresti ricevere questo messaggio di errore se hai specificato un'azione che AWS IoT Events attualmente non supporta. Per un elenco delle azioni supportate, consulta [Azioni supportate](#).

- Messaggio: la `DetectorModel` definizione contiene un'*aws-service* azione, ma il *aws-service* servizio non è supportato nella regione *region-name*.

Soluzione: potresti ricevere questo messaggio di errore se l'azione specificata è supportata da AWS IoT Events, ma l'azione non è disponibile nella tua regione corrente. Ciò potrebbe verificarsi quando si tenta di inviare dati a un AWS servizio che non è disponibile nella regione. Inoltre, devi scegliere la stessa regione per entrambi AWS IoT Events i AWS servizi che stai utilizzando.

service-limits

Un risultato dell'analisi con informazioni su `service-limits`, corrisponde ai seguenti messaggi di errore:

- Messaggio: *l'espressione di contenuto consentita nel payload ha superato il limite di content-expression-sizebyte nell'event event-name nello state state-name.*

Soluzione: potresti ricevere questo messaggio di errore se l'espressione di contenuto per il payload dell'azione è superiore a 1024 byte. La dimensione dell'espressione di contenuto per un payload può essere fino a 1024 byte.

- Messaggio: il numero di stati consentiti nella definizione del modello di rilevatore ha superato il limite. *states-per-detector-model*

Soluzione: potreste ricevere questo messaggio di errore se il modello del rilevatore ha più di 20 stati. Un modello di rilevatore può avere fino a 20 stati.

- Messaggio: La durata del *nome del timer* deve essere di almeno *minimum-timer-duration* secondi.

Soluzione: potresti ricevere questo messaggio di errore se la durata del timer è inferiore a 60 secondi. È consigliabile che la durata di un timer sia compresa tra 60 e 31622400 secondi. Se si specifica un'espressione per la durata del timer, il risultato valutato dell'espressione di durata viene arrotondato per difetto al numero intero più vicino.

- Messaggio: il numero di azioni consentite per evento ha superato il limite *actions-per-event* nella definizione del modello di rilevatore

Soluzione: potreste ricevere questo messaggio di errore se l'evento ha più di 10 azioni. Puoi avere fino a 10 azioni per ogni evento nel tuo modello di rilevatore.

- Messaggio: il numero di eventi di transizione consentiti per stato ha superato il limite *transition-events-per-state* nella definizione del modello di rilevatore.

Soluzione: potreste ricevere questo messaggio di errore se lo stato presenta più di 20 eventi di transizione. È possibile avere fino a 20 eventi di transizione per ogni stato del modello di rilevatore.

- Messaggio: il numero di eventi consentiti per stato ha superato il limite *events-per-state* nella definizione del modello di rilevatore

Soluzione: potreste ricevere questo messaggio di errore se lo stato presenta più di 20 eventi. È possibile avere fino a 20 eventi per ogni stato del modello di rilevatore.

- Messaggio: il numero massimo di modelli di rilevatore che possono essere associati a un singolo ingresso potrebbe aver raggiunto il limite. Il *nome dell'input* di input viene utilizzato nei *detector-models-per-input* percorsi dei modelli di rilevatori.

Soluzione: potreste ricevere questo messaggio di avviso se provate a indirizzare un input a più di 10 modelli di rilevatore. È possibile associare fino a 10 diversi modelli di rilevatore a un singolo modello di rilevatore.

structure

Un risultato dell'analisi con informazioni su `structure`, corrisponde ai seguenti messaggi di errore:

- Messaggio: le azioni possono avere un solo tipo definito, ma è stata trovata un'azione con *number-of-types* tipi. Suddividi in azioni separate.

Soluzione: potresti ricevere questo messaggio di errore se hai specificato due o più azioni in un unico campo utilizzando le operazioni API per creare o aggiornare il modello del rilevatore. È possibile definire una serie di `Action` oggetti. Assicuratevi di definire ogni azione come un oggetto separato.

- Messaggio: *le TransitionEvent transition-event-name transizioni verso un nome di stato inesistente.*

Soluzione: potresti ricevere questo messaggio di errore se non AWS IoT Events riesci a trovare lo stato successivo a cui ha fatto riferimento l'evento di transizione. Assicurati di aver definito lo stato successivo e di aver inserito il nome dello stato corretto.

- Messaggio: `DetectorModelDefinition` Aveva un nome di stato condiviso: ha trovato il nome dello *stato* con *number-of-states* ripetizioni.

Soluzione: è possibile che venga visualizzato questo messaggio di errore se si utilizza lo stesso nome per uno o più stati. Assicurati di assegnare un nome univoco a ogni stato del tuo modello di rilevatore. Il nome dello stato deve contenere da 1 a 128 caratteri. Caratteri validi: a-z, A-Z, 0-9, _ (trattino basso) e - (trattino).

- Messaggio: le definizioni `initialStateName` *initial-state-name* non corrispondono a uno stato definito.

Soluzione: potresti ricevere questo messaggio di errore se il nome dello stato iniziale non è corretto. Il modello del rilevatore rimane nello stato iniziale (iniziale) fino all'arrivo di un input. Una volta ricevuto un input, il modello del rilevatore passa immediatamente allo stato successivo. Assicuratevi che il nome dello stato iniziale sia il nome di uno stato definito e di inserire il nome corretto.

- Messaggio: `Detector Model Definition` deve utilizzare almeno un input in una condizione.

Soluzione: potresti ricevere questo errore se non hai specificato un input in una condizione. È necessario utilizzare almeno un input in almeno una condizione. Altrimenti, AWS IoT Events non valuta i dati in arrivo.

- Messaggio: è possibile impostare solo uno dei secondi e `DurationExpression`. `SetTimer`

Soluzione: potresti ricevere questo messaggio di errore se li hai utilizzati entrambi `seconds` e come `durationExpression` timer. Assicurati di utilizzare uno dei due `seconds` o `durationExpression` come parametri di `SetTimerAction`. Per ulteriori informazioni, consulta [SetTimerAction](#) nella documentazione di riferimento dell'API AWS IoT Events.

- Messaggio: un'azione nel modello del rilevatore non è raggiungibile. Verifica la condizione che avvia l'azione.

Soluzione: se un'azione nel modello del rilevatore non è raggiungibile, la condizione dell'evento viene valutata falsa. Controllate la condizione dell'evento che contiene l'azione per assicurarvi che risulti vera. Quando la condizione dell'evento risulta vera, l'azione dovrebbe diventare raggiungibile.

- Messaggio: è in corso la lettura di un attributo di input, ma ciò può essere causato dalla scadenza del timer.

Soluzione: il valore di un attributo di input può essere letto quando si verifica una delle seguenti condizioni:

- È stato ricevuto un nuovo valore di input.
- Quando un timer nel rilevatore è scaduto.

Per garantire che un attributo di input venga valutato solo quando viene ricevuto il nuovo valore per quell'input, includi una chiamata alla `triggerType("Message")` funzione nella tua condizione come segue:

La condizione originale oggetto di valutazione nel modello del rilevatore:

```
if ($input.HeartBeat.status == "OFFLINE")
```

diventerebbe simile al seguente:

```
if ( triggerType("MESSAGE") && $input.HeartBeat.status == "OFFLINE")
```

dove una chiamata alla `triggerType("Message")` funzione precede l'input iniziale fornito nella condizione. Utilizzando questa tecnica, la `triggerType("Message")` funzione restituirà `true` e soddisferà la condizione di ricevere un nuovo valore di input. Per ulteriori informazioni sull'utilizzo della `triggerType` funzione, cercate `triggerType` nella sezione [Espressioni](#) della Guida per gli AWS IoT Events sviluppatori

- **Messaggio:** Uno stato nel modello del rilevatore non è raggiungibile. Verificate la condizione che provocherà la transizione allo stato desiderato.

Soluzione: se uno stato del modello del rilevatore non è raggiungibile, una condizione che causa una transizione in entrata a quello stato viene valutata falsa. Verificate che le condizioni delle transizioni in entrata verso lo stato irraggiungibile nel modello del rilevatore risultino vere, in modo che lo stato desiderato possa diventare raggiungibile.

- **Messaggio:** una scadenza del timer può causare l'invio di una quantità imprevista di messaggi.

Soluzione: per evitare che il modello del rilevatore entri in uno stato infinito di invio di una quantità imprevista di messaggi a causa della scadenza di un timer, prendete in considerazione l'utilizzo di una chiamata alla `triggerType("Message")` funzione, nelle seguenti condizioni del modello di rilevatore:

La condizione originale da valutare nel modello del rilevatore:

```
if (timeout("awake"))
```

verrebbe trasformato in una condizione simile alla seguente:

```
if (triggerType("MESSAGE") && timeout("awake"))
```

dove una chiamata alla `triggerType("Message")` funzione precede l'input iniziale fornito nella condizione.

Questa modifica impedisce l'avvio di azioni del timer nel rilevatore, impedendo l'invio di un ciclo infinito di messaggi. Per ulteriori informazioni su come utilizzare le azioni del timer nel rilevatore, consulta la pagina [Utilizzo delle azioni integrate](#) della Guida per gli sviluppatori AWS IoT Events

expression-syntax

Un risultato dell'analisi con informazioni su `expression-syntax`, corrisponde ai seguenti messaggi di errore:

- **Messaggio:** l'espressione di payload `{expression}` non è valida. Il tipo di payload definito è JSON, quindi è necessario specificare un'espressione che AWS IoT Events restituisca una stringa.

Soluzione: se il tipo di payload specificato è JSON, verifica AWS IoT Events innanzitutto se il servizio è in grado di valutare l'espressione in una stringa. Il risultato valutato non può essere un valore booleano o un numero. Se la convalida non riesce, potresti ricevere questo errore.

- Messaggio: `SetVariableAction.value` deve essere un'espressione. Analisi del valore *'variable-value'* non riuscita

Soluzione: è possibile utilizzare `SetVariableAction` per definire una variabile con una `e.name` e un valore `value`. `value` può essere una stringa, un numero o un valore booleano. È inoltre possibile specificare un'espressione per `value`. Per ulteriori informazioni [SetVariableAction](#), consulta la sezione AWS IoT Events API Reference.

- Messaggio: non è stato possibile analizzare l'espressione degli attributi (*nome-attributo*) per l'azione `DynamoDB`. Inserisci l'espressione con la sintassi corretta.

Soluzione: è necessario utilizzare le espressioni per tutti i parametri nei `DynamoDBAction` modelli di sostituzione. Per ulteriori informazioni, consulta [DynamodbAction](#) nell'API Reference. AWS IoT Events

- Messaggio: non siamo riusciti ad analizzare la tua espressione del `TableName` per l'azione `DynamoDBV2`. Inserisci l'espressione con la sintassi corretta.

Soluzione: il `tableName` pin `DynamoDBV2Action` deve essere una stringa. È necessario utilizzare un'espressione per `tableName`. Le espressioni accettano valori letterali, operatori, funzioni, riferimenti e modelli di sostituzione. Per ulteriori informazioni, consulta [DynamoDBV2Action](#) nel riferimento API. AWS IoT Events

- Messaggio: non siamo riusciti a valutare la tua espressione in un formato JSON valido. L'azione `DynamoDBV2` supporta solo il tipo di payload JSON.

Soluzione: il tipo di payload per deve essere JSON. `DynamoDBV2` Assicurati che sia in AWS IoT Events grado di valutare l'espressione di contenuto per il payload in un formato JSON valido. Per ulteriori informazioni, consulta [DynamoDBV2Action](#), nel riferimento API. AWS IoT Events

- Messaggio: *non siamo riusciti ad analizzare la tua espressione di contenuto per il payload di action-type*. Inserisci un'espressione di contenuto con la sintassi corretta.

Soluzione: l'espressione di contenuto può contenere stringhe (*'string'*), variabili (`$variable.variable-name`), *valori di input* (`$input.nome di input.path-to-datum`), concatenazioni di stringhe e stringhe che contengono. `${}`

- Messaggio: i payload personalizzati non devono essere vuoti.

Soluzione: potresti ricevere questo messaggio di errore se hai scelto Custom payload per la tua azione e non hai inserito un'espressione di contenuto nella console. AWS IoT Events Se scegli Payload personalizzato, devi inserire un'espressione di contenuto in Payload personalizzato. Per ulteriori informazioni, consulta [Payload](#) nell'API Reference. AWS IoT Events

- Messaggio: *impossibile analizzare l'espressione di durata 'duration-expression' per il timer 'timer-name'.*

Soluzione: il risultato valutato dell'espressione di durata per il timer deve essere un valore compreso tra 60 e 31622400. Il risultato valutato della durata viene arrotondato per difetto al numero intero più vicino.

- Messaggio: analisi *dell'espressione «espressione» per il nome dell'azione non riuscita*

Soluzione: potresti ricevere questo messaggio se l'espressione per l'azione specificata ha una sintassi errata. Assicurati di inserire un'espressione con la sintassi corretta. Per ulteriori informazioni, consulta [Sintassi](#).

- Messaggio: il tuo *FieldName* IotSiteWiseAction for non può essere analizzato. È necessario utilizzare la sintassi corretta nell'espressione.

Soluzione: potresti ricevere questo errore se non AWS IoT Events riesci ad analizzare il tuo *FieldName for*. IotSiteWiseAction Assicurati che *FieldName* utilizzi un'espressione AWS IoT Events che possa essere analizzata. Per ulteriori informazioni, consulta [IotSiteWiseAction](#) nella documentazione di riferimento dell'API AWS IoT Events.

data-type

Un risultato dell'analisi con informazioni sudate a -type, corrisponde ai seguenti messaggi di errore:

- Messaggio: l'espressione di durata *duration-expression* per timer *timer-name* non è valida, deve restituire un numero.

Soluzione: potresti ricevere questo messaggio di errore se non AWS IoT Events riuscisci a valutare l'espressione della durata del timer in un numero. Assicurati che il tuo `durationExpression` possa essere convertito in un numero. Altri tipi di dati, come quelli booleani, non sono supportati.

- Messaggio: l'espressione *condizionale dell'espressione non è un'espressione condizionale valida*.

Soluzione: potresti ricevere questo messaggio di errore se non AWS IoT Events riuscissi a valutare il tuo valore `condition-expression` come booleano. Il valore booleano deve essere o `TRUE` o `FALSE`. Assicurati che l'espressione della condizione possa essere convertita in un valore booleano. Se il risultato non è un valore booleano, è equivalente `FALSE` e non richiama le azioni o la transizione a quanto specificato nell'`nextState` evento.

- Messaggio: *tipi di dati incompatibili [inferred-types] trovati come riferimento nella seguente espressione: `expression`*

Soluzione: tutte le espressioni per lo stesso attributo o variabile di input nel modello del rilevatore devono fare riferimento allo stesso tipo di dati.

Utilizza le seguenti informazioni per risolvere il problema:

- Quando utilizzate un riferimento come operando con uno o più operatori, assicuratevi che tutti i tipi di dati a cui fate riferimento siano compatibili.

Ad esempio, nell'espressione seguente, il numero intero 2 è un operando di entrambi gli `==` operatori `and`. `&&` Per garantire che gli operandi siano compatibili `$variable.testVariable + 1` e che `$variable.testVariable` debbano fare riferimento a un numero intero o decimale.

Inoltre, il numero intero 1 è un operando dell'operatore `+`. Pertanto, `$variable.testVariable` deve fare riferimento a un numero intero o decimale.

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

- Quando utilizzate un riferimento come argomento passato a una funzione, assicuratevi che la funzione supporti i tipi di dati a cui fate riferimento.

Ad esempio, la seguente `timeout("time-name")` funzione richiede una stringa con virgolette doppie come argomento. Se si utilizza un riferimento per il valore `timer-name`, è necessario fare riferimento a una stringa tra virgolette doppie.

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

Note

Per la `convert(type, expression)` funzione, se si utilizza un riferimento per il valore del *tipo*, il risultato valutato del riferimento deve essere `String`, `Decimal` o `Boolean`.

Per ulteriori informazioni, consulta [Riferimenti](#).

- Messaggio: *tipi di dati incompatibili [inferred-types] utilizzati con riferimento*. Ciò può causare un errore di runtime.

Soluzione: potresti ricevere questo messaggio di avviso se due espressioni per lo stesso attributo di input o variabile fanno riferimento a due tipi di dati. Assicurati che le espressioni per lo stesso attributo o variabile di input facciano riferimento allo stesso tipo di dati nel modello del rilevatore.

- Messaggio: *i tipi di dati [tipi inferiti] che hai inserito per l'operatore [operatore] non sono compatibili per la seguente espressione: 'expression'*

Soluzione: potresti ricevere questo messaggio di errore se l'espressione combina tipi di dati che non sono compatibili con un operatore specificato. Ad esempio, nell'espressione seguente, l'operatore `+` è compatibile con i tipi di dati `Integer`, `Decimal` e `String`, ma non con gli operandi di tipo booleano.

```
true + false
```

È necessario assicurarsi che i tipi di dati utilizzati con un operatore siano compatibili.

- Messaggio: i tipi di dati [tipi *inferiti*] trovati per l'*attributo input* non sono compatibili e possono causare un errore di runtime.

Soluzione: potresti ricevere questo messaggio di errore se due espressioni per lo stesso attributo di input fanno riferimento a due tipi `OnEnterLifecycle` di dati per lo stato o per entrambi gli stati. `OnInputLifecycle` `OnExitLifecycle` Assicurati che le espressioni in `OnEnterLifecycle` (o entrambe `OnExitLifecycle`) `OnInputLifecycle` facciano riferimento allo stesso tipo di dati per ogni stato del modello del rilevatore.

- Messaggio: l'espressione di payload [*espressione*] non è valida. Specificate un'espressione che restituisca una stringa in fase di esecuzione perché il tipo di payload è in formato JSON.

Soluzione: potresti ricevere questo errore se il tipo di payload specificato è JSON, ma non AWS IoT Events riesci a valutarne l'espressione in una stringa. Assicurati che il risultato valutato sia una stringa, non un valore booleano o un numero.

- Messaggio: l'espressione interpolata *{interpolated-expression}* deve restituire un numero intero o un valore booleano in fase di esecuzione. In caso contrario, l'espressione di payload *{payload-expression}* non sarà analizzabile in fase di esecuzione come JSON valido.

Soluzione: potresti ricevere questo messaggio di errore se non AWS IoT Events riesci a valutare l'espressione interpolata con un numero intero o un valore booleano. Assicurati che l'espressione interpolata possa essere convertita in un numero intero o in un valore booleano, poiché altri tipi di dati, come tring, non sono supportati.

- Messaggio: *il tipo di espressione nell'espressione di IotSitetwiseAction campo è definito come tipo definito dal tipo e dedotto come tipo dedotto.* Il tipo definito e il tipo dedotto devono essere uguali.

Soluzione: potresti ricevere questo messaggio di errore se l'espressione in `propertyValue` of `IotSitetwiseAction` ha un tipo di dati definito in modo diverso dal tipo di dati da cui si deduce. AWS IoT Events Assicurati di utilizzare lo stesso tipo di dati per tutte le istanze di questa espressione nel tuo modello di rilevatore.

- Messaggio: *i tipi di dati [inferred-types] utilizzati per l'setTimerazione non restituiscono lo stesso valore per la seguente espressione: Integer espressione*

Soluzione: potresti ricevere questo messaggio di errore se il tipo di dati dedotto per l'espressione di durata non è Integer o Decimal. Assicurati che il tuo `durationExpression` possa essere convertito in un numero. Altri tipi di dati, come Boolean e String, non sono supportati.

- Messaggio: *i tipi di dati [inferred-types] utilizzati con gli operandi dell'operatore di confronto [operator] non sono compatibili nella seguente espressione: espressione*

Soluzione: i tipi di dati dedotti per gli operandi dell'*operatore* nell'espressione condizionale (*espressione*) del modello di rilevatore non corrispondono. Gli operandi devono essere utilizzati con i tipi di dati corrispondenti in tutte le altre parti del modello di rilevatore.

Tip

È possibile utilizzarlo `convert` per modificare il tipo di dati di un'espressione nel modello del rilevatore. Per ulteriori informazioni, consulta [Funzioni](#).

referenced-data

Un risultato dell'analisi con informazioni su `referenced-data`, corrisponde ai seguenti messaggi di errore:

- Messaggio: Rilevato un guasto *Timer: il nome del timer* viene utilizzato in un'espressione ma non viene mai impostato.

Soluzione: è possibile che venga visualizzato questo messaggio di errore se si utilizza un timer non impostato. È necessario impostare un timer prima di utilizzarlo in un'espressione. Inoltre, assicuratevi di inserire il nome corretto del timer.

- Messaggio: Variabile interrotta rilevata: *il nome della variabile* viene utilizzato in un'espressione ma non viene mai impostato.

Soluzione: è possibile che venga visualizzato questo messaggio di errore se si utilizza una variabile non impostata. È necessario impostare una variabile prima di utilizzarla in un'espressione. Inoltre, assicuratevi di inserire il nome della variabile corretto.

- Messaggio: Variabile danneggiata rilevata: una variabile viene utilizzata in un'espressione prima di essere impostata su un valore.

Soluzione: ogni variabile deve essere assegnata a un valore prima di poter essere valutata in un'espressione. Imposta il valore della variabile prima di ogni utilizzo in modo da poterne recuperare il valore. Inoltre, assicuratevi di inserire il nome corretto della variabile.

referenced-resource

Un risultato dell'analisi con informazioni su `referenced-resource`, corrisponde ai seguenti messaggi di errore:

- Messaggio: Detector Model Definition contiene un riferimento a un input che non esiste.

Soluzione: è possibile che venga visualizzato questo messaggio di errore se si utilizzano espressioni per fare riferimento a un input che non esiste. Assicurati che l'espressione faccia riferimento a un input esistente e inserisci il nome di input corretto. Se non disponi di un input, creane prima uno.

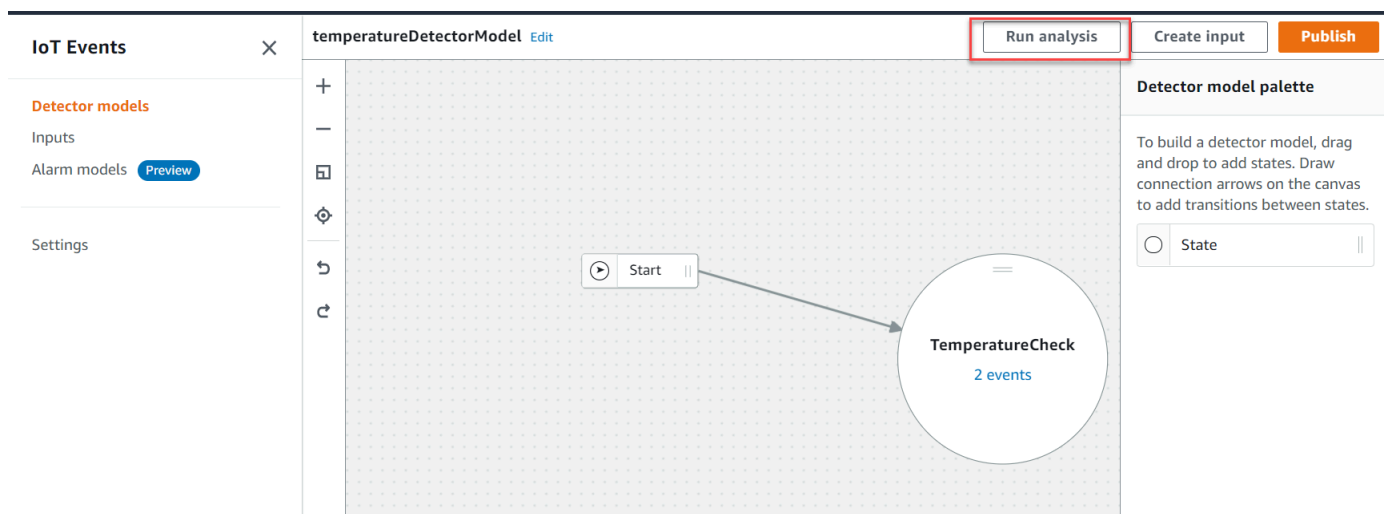
- Messaggio: *Detector Model Definition contiene: input-name non valido InputName*

Soluzione: è possibile che venga visualizzato questo messaggio di errore se il modello del rilevatore contiene un nome di input non valido. Assicurati di inserire il nome di input corretto. Il nome di input deve contenere da 1 a 128 caratteri. Caratteri validi: a-z, A-Z, 0-9, _ (trattino basso) e - (trattino).

Analisi di un modello di rilevatore (console)

I passaggi seguenti utilizzano la AWS IoT Events console per analizzare un modello di rilevatore.

1. Accedi alla [console AWS IoT Events](#).
2. Nel pannello di navigazione, scegli Modelli di rilevatori.
3. In Modelli di rilevatori, scegli il modello di rilevatore di destinazione.
4. Nella pagina del modello del rilevatore, scegli Modifica.
5. Nell'angolo in alto a destra, scegli Esegui analisi.



Di seguito è riportato un esempio di risultato di analisi nella AWS IoT Events console.

The screenshot displays the AWS IoT Events console interface for editing a detector model named 'temperatureDetectorModel'. On the left, a navigation menu includes 'Detector models', 'Inputs', 'Alarm models', and 'Settings'. The main canvas shows a state machine with a 'Start' state (represented by a play button icon) connected to a 'TemperatureCheck' state (represented by a circle with '2 events'). On the right, the 'Detector model palette' shows a 'State' component. At the bottom, a 'Detector model analysis' panel is highlighted with a red box, showing a summary of results: (1) All, (0) Error, (0) Warning, and (1) Information. An information message is displayed: 'Info: data-type Message: Inferred data types [Integer] for \$variable.temperatureChecked'.

Note

Dopo aver AWS IoT Events iniziato ad analizzare il modello del rilevatore, hai fino a 24 ore per recuperare i risultati dell'analisi.

Analisi di un modello di rilevatore () AWS CLI

I passaggi seguenti utilizzano AWS CLI per analizzare un modello di rilevatore.

1. Eseguite il comando seguente per avviare un'analisi.

```
aws iotevents start-detector-model-analysis --cli-input-json file://file-name.json
```

Note

Sostituite *file-name* con il nome del file che contiene la definizione del modello del rilevatore.

Example Definizione del modello di rilevatore

```
{
  "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"
  }
}
```

Se utilizzate il AWS CLI per analizzare un modello di rilevatore esistente, scegliete una delle seguenti opzioni per recuperare la definizione del modello di rilevatore:

- Se desideri utilizzare la AWS IoT Events console, procedi come segue:
 1. Nel riquadro di navigazione, scegli Modelli di rilevatori.
 2. In Modelli di rilevatori, scegli il modello di rilevatore di destinazione.
 3. Scegli Esporta modello di rilevatore da Action per scaricare il modello di rilevatore. Il modello del rilevatore viene salvato in JSON.
 4. Aprire il file JSON del modello del rilevatore.
 5. Ti serve solo l'oggetto. `detectorModelDefinition` Rimuovi quanto segue:
 - La prima parentesi riccia (`{`) nella parte superiore della pagina
 - La linea `detectorModel`
 - L'`detectorModelConfiguration` oggetto
 - L'ultima parentesi riccia (`}`) nella parte inferiore della pagina
 6. Salva il file.
- Se desideri utilizzare il AWS CLI, procedi come segue:
 1. Esegui il comando seguente in un terminale:

```
aws iotevents describe-detector-model --detector-model-name detector-model-name
```

2. Sostituiscilo *detector-model-name* con il nome del modello del tuo rilevatore.
3. Copia l'`detectorModelDefinition` oggetto in un editor di testo.
4. Aggiungete parentesi graffe (`{}`) all'esterno di. `detectorModelDefinition`
5. Salvate il file in JSON.

Example Risposta di esempio

```
"analysisId": "c1133390-14e3-4204-9a66-31efd92a4fed"
}
```

2. Copia l'ID di analisi dall'output.
3. Eseguite il comando seguente per recuperare lo stato dell'analisi.

```
aws iotevents describe-detector-model-analysis --analysis-id "analysis-id"
```

Note

Sostituite *analysis-id* con l'ID di analisi che avete copiato.

Example Risposta di esempio

```
{
  "status": "COMPLETE"
}
```

Lo stato può avere uno dei seguenti valori:

- **RUNNING**— AWS IoT Events sta analizzando il modello del rilevatore. Il completamento di questo processo può richiedere fino a un minuto.
 - **COMPLETE**— AWS IoT Events ha terminato l'analisi del modello del rilevatore.
 - **FAILED**— AWS IoT Events non è stato possibile analizzare il modello del rilevatore. Riprova più tardi.
4. Eseguite il comando seguente per recuperare uno o più risultati di analisi del modello di rilevatore.

Note

Sostituite *analysis-id* con l'ID di analisi che avete copiato.

```
aws iotevents get-detector-model-analysis-results --analysis-id "analysis-id"
```

Example Risposta di esempio

```
{
  "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

Dopo aver AWS IoT Events iniziato ad analizzare il modello del rilevatore, avete fino a 24 ore per recuperare i risultati dell'analisi.

Comandi AWS IoT Events

Questo capitolo fornisce informazioni dettagliate su tutte le operazioni API, incluse le richieste di esempio, le risposte e gli errori per i protocolli di servizi Web supportati. AWS IoT Events

Operazioni AWS IoT Events

È possibile utilizzare i comandi AWS IoT Events API per creare, leggere, aggiornare ed eliminare input e modelli di rilevatori e per elencarne le versioni. Per ulteriori informazioni, consulta le [azioni](#) e i [tipi di dati](#) supportati dall'AWS IoT Events API AWS IoT Events Reference.

Le [AWS IoT Events sezioni](#) del AWS CLI Command Reference includono i AWS CLI comandi che è possibile utilizzare per amministrare e manipolare AWS IoT Events.

Dati AWS IoT Events

È possibile utilizzare i comandi AWS IoT Events Data API per inviare input ai rilevatori, elencare i rilevatori e visualizzare o aggiornare lo stato di un rilevatore. Per ulteriori informazioni, consulta le [azioni](#) e i [tipi di dati](#) supportati da AWS IoT Events Data in the API Reference. AWS IoT Events

Le [sezioni AWS IoT Events dati](#) del AWS CLI Command Reference includono i AWS CLI comandi che è possibile utilizzare per elaborare AWS IoT Events i dati.

Cronologia dei documenti

La tabella seguente descrive le modifiche importanti alla Guida per gli AWS IoT Events sviluppatori dopo il 17 settembre 2020. Per ulteriori informazioni sugli aggiornamenti di questa documentazione, puoi iscriverti a un feed RSS.

Modifica	Descrizione	Data
Lancio della regione	AWS IoT Events è ora disponibile nella regione Asia Pacifico (Mumbai).	30 settembre 2021
Lancio della regione	AWS IoT Events è ora disponibile nella regione AWS GovCloud (Stati Uniti occidentali).	22 settembre 2021
Risolvi i problemi di un modello di rilevatore eseguendo analisi	AWS IoT Events ora puoi analizzare il tuo modello di rilevatore e generare risultati di analisi che puoi utilizzare per risolvere i problemi del tuo modello di rilevatore.	23 febbraio 2021
Lancio della regione	Lanciato AWS IoT Events in Cina (Pechino).	30 settembre 2020
Utilizzo delle espressioni	Sono stati aggiunti esempi per mostrare come scrivere espressioni.	22 settembre 2020
Monitoraggio con allarmi	Gli allarmi consentono di monitorare i dati in caso di modifiche. Puoi creare allarmi che inviano notifiche quando viene superata una soglia.	1 giugno 2020

Aggiornamenti precedenti

La tabella seguente descrive le modifiche importanti alla Guida per gli AWS IoT Events sviluppatori prima del 18 settembre 2020.

Modifica	Descrizione	Data
Aggiunte	È stata aggiunta la convalida del tipo a. Riferimenti	3 agosto 2020
Aggiunte	Sono state aggiunte informazioni sulla regione a. Lavorare con altri AWS servizi	7 maggio 2020
Aggiunte, aggiornamenti	Aggiunta la funzionalità di personalizzazione del payload e nuove azioni relative agli eventi: Amazon DynamoDB e AWS IoT SiteWise	27 aprile 2020
Modifiche	Sono state aggiunte nuove descrizioni dei concetti relativi alle macchine a stati. Modifica generale dei contenuti.	31 ottobre 2019
Aggiunte	Aggiunte nuove funzioni integrate per le espressioni condizionali del modello di rivelatore.	10 settembre 2019
Aggiunte	Aggiunti esempi di modelli di rilevatori.	5 agosto 2019
Aggiunte	Sono state aggiunte nuove azioni relative agli eventi: Lambda, Amazon SQS, Kinesis Data Firehose e input. AWS IoT Events	19 luglio 2019

Modifica	Descrizione	Data
Aggiunte, correzioni	Descrizione corretta della funzione. <code>timeout()</code> Sono state aggiunte le migliori pratiche relative all'inattività dell'account.	11 giugno 2019
Correzioni	Aggiornamento: immagine della pagina delle opzioni di debug della console; politica delle autorizzazioni della console.	5 giugno 2019
Aggiornamenti	AWS IoT Eventsservizio aperto alla disponibilità generale.	30 maggio 2019
Aggiunte, aggiornamenti	Informazioni di sicurezza aggiornate; è stato aggiunto un esempio di modello di rilevatore annotato.	22 maggio 2019
Correzioni	Aggiornato: link al download in anteprima limitato; esempi di payload di Amazon SNS; aggiunte alle autorizzazioni richieste per. <code>CreateDetectorModel</code>	17 maggio 2019
Aggiunte	Sono state aggiunte informazioni sulla sicurezza.	9 maggio 2019
Correzioni	Il link al download limitato dell'anteprima è stato corretto.	19 aprile 2019
Modifiche	Miglioramenti editoriali.	16 aprile 2019

Modifica	Descrizione	Data
Versione di anteprima limitata	Versione di anteprima limitata della documentazione.	28 marzo 2019
Modifiche	Miglioramenti editoriali.	18 maggio 2018

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.