Amazon IoT Core

开发人员指南

Amazon IoT Core: 开发人员指南

Copyright © 2023 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆或者贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其它商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

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

Table of Contents

什么是 Amazon IoT？ 1
您的设备和应用程序如何访问 Amazon IoT 1
Amazon IoT 能做的事情 2
行业中的 IoT 2
家居自动化中的 IoT 3
Amazon IoT 的工作原理 3
IoT 世界 3
Amazon IoT 服务概览 5
Amazon IoT Core 服务 7
了解有关 Amazon IoT 的更多信息 10
Amazon IoT 的培训资源 10
Amazon IoT 资源和指南 10
社交媒体中的 Amazon IoT 11
Amazon IoT Core 规则引擎使用的 Amazon 服务 11
Amazon IoT Core 支持的通信协议 12
Amazon IoT 控制台中的新增功能 12
图例 14
Amazon IoT Core 入门 15
将第一个设备连接到 Amazon IoT Core 15
设置您的Amazon Web Services 账户 16
注册一个 Amazon Web Services 账户 16
保护 IAM 用户 17
打开 Amazon IoT 控制台 17
尝试 Amazon IoT Core 交互式教程 17
连接 IoT 设备 18
保存离线设备状态 19
将设备数据路由到服务 20
试试 Amazon IoT 快速连接 20
第 1 步 开始教程 21
第 2 步 创建一个事物对象 22
第 3 步 将文件下载到您的设备 24
第 4 步 运行示例 26
步骤 5：进一步探索 29
测试与设备数据终端节点的连接 29
通过详细教程了解 Amazon IoT Core 服务 33
哪种设备选项最适合您？ 34
创建 Amazon IoT 资源 34
配置您的设备 37
使用 Amazon IoT MQTT 客户端查看 MQTT 消息 59
查看 MQTT 客户端中的 MQTT 消息 60
从 MQTT 客户端发布 MQTT 消息 62
连接到 Amazon IoT Core 64
Amazon IoT Core - 控制面板终端节点 64
Amazon IoT 设备终端节点 65
Amazon IoT Core用于 LoRa WAN 网关和设备 66
连接到 Amazon IoT Core 服务终端节点 66
Amazon CLI for Amazon IoT Core 67
Amazon SDK 67
AmazonMobile SDK 71
Amazon IoT Core 服务的 REST API 71
将设备连接到 Amazon IoT 72
Amazon IoT 设备数据和服务终端节点 72
Amazon IoT Device SDK 73
设备通信协议 75

MQTT 主题 94
可配置的终端节点 108
连接到 Amazon IoT FIPS 终端节点 112
Amazon IoT Core - 控制面板终端节点 112
Amazon IoT Core - 数据层面终端节点 113
Amazon IoT Device Management - 任务数据终端节点 113
Amazon IoT Device Management - Fleet Hub 终端节点 113
Amazon IoT Device Management - 安全隧道终端节点 113
Amazon IoT 教程 114
用 Amazon IoT Device Client 构建演示 114
使用 Amazon IoT Device Client 构建演示的先决条件 114
Amazon IoT Device Client 的设备准备 116
安装并配置 Amazon IoTDevice Client 124
演示 MQTT 消息与 Amazon IoTDevice Client 通信 133
使用 Amazon IoTDevice Client 演示远程操作（任务） 145
清理 154
使用 Amazon IoT设备开发工具包构建解决方案 160
开始使用 Amazon IoT设备开发工具包构建解决方案 160
使用 Amazon IoT设备开发工具包将设备连接到 Amazon IoT Core 160
创建将设备数据路由到其他服务的 Amazon IoT规则 175
在设备处于离线状态时使用设备影子保持设备状态 201
为 Amazon IoT Core创建自定义授权方 219
使用 Amazon IoT和 Raspberry Pi 监控土壤含水量 230
使用 Amazon IoT 管理设备 239
如何使用注册表管理事物 239
创建事物 239
列出事物 240
描述事物 241
更新事物 242
删除事物 242
将委托人附加到事物 242
将委托人与事物分离 242
事物类型 243
创建事物类型 243
列出事物类型 243
描述事物类型 244
将事物类型与事物相关联 244
弃用事物类型 244
删除事物类型 245
静态事物组 246
创建静态事物组 246
描述事物组 247
将事物添加到静态事物组 248
从静态事物组中删除事物 248
列出事物组中的事物 248
列出事物组 249
列出事物的组 250
更新静态事物组 251
删除事物组 251
将策略附加到静态事物组 251
从静态事物组分离策略 252
列出附加到静态事物组的策略 252
列出策略的组 252
获取事物的有效策略 253
测试 MQTT 操作的授权 253
动态事物组 254
创建动态事物组 255

描述动态事物组 255
更新动态事物组 256
删除动态事物组 256
限制和冲突 257
给您的 Amazon IoT 资源加标签 259
有关标签的基本知识 259
标签限制 260
在 IAM 策略中使用标签 260
账单组 261
查看成本分配和使用率数据 262
安全性 263
Amazon IoT 中的安全性 263
身份验证 264
X.509 证书概览 264
服务器身份验证 264
客户端身份验证 267
自定义身份验证 287
Authorization 298
Amazon IoT Core 策略 299
使用 Amazon IoT Core 凭证提供程序授予直接调用Amazon服务的权限 340
通过 IAM 跨账户访问 344
数据保护 345
Amazon IoT 中的数据加密 346
Amazon IoT 中的传输安全 346
数据加密 347
身份和访问管理 348
受众 348
使用 IAM 身份进行身份验证 349
使用策略管理访问 350
Amazon IoT 如何与 IAM 协同工作 351
基于身份的策略示例 370
Amazon 托管策略 372
问题排查 382
日志记录和监控 384
监控工具 384
合规性验证 385
故障恢复能力 385
将 Amazon IoT Core 与 VPC 终端节点结合使用 385
创建 Amazon IoT Core 的 VPC 终端节点 386
使用 VPC 终端节点控制对 Amazon IoT Core 的访问 386
VPC 终端节点的限制 387
使用 IoT Core 扩缩 VPC 终端节点 387
将自定义域用于 VPC 终端节点 387
适用于 Amazon IoT Core 的 VPC 终端节点的可用性 388
基础设施安全性 388
安全监控 388
安全最佳实践 388
在 Amazon IoT 中保护 MQTT 连接 388
使设备的时钟保持同步 390
验证服务器证书 390
每个设备使用一个身份 390
使用次级Amazon Web Services 区域作为备份 391
使用即时预调配 391
运行 Amazon IoT Device Advisor 测试的权限 391
针对 Device Advisor 防范跨服务混淆代理 392
监控 Amazon IoT 394
配置 Amazon IoT 日志记录 394

配置日志记录角色和策略 395
在 Amazon IoT（控制台）中配置默认日志记录 396
在 Amazon IoT 中配置默认日志记录 (CLI) 397
在 Amazon IoT 中配置资源特定的日志记录 (CLI) 398
日志级别 400
使用 Amazon 监控Amazon IoT告警和指标 CloudWatch 400
使用 Amazon IoT 指标 401
在中创建 CloudWatch 警报Amazon IoT 401
Amazon IoT 指标和维度 404
Amazon IoT使用 CloudWatch 日志进行监控 412
在 CloudWatch 控制台中查看Amazon IoT日志 413
CloudWatch Amazon IoT日志条目 413
将设备端日志上载到 Amazon CloudWatch 432
工作原理 432
使用 Amazon IoT 规则上载设备端日志 432
使用 Amazon CloudTrail 记录 Amazon IoT API 调用 440
Amazon IoT中的信息 CloudTrail 440
了解 Amazon IoT 日志文件条目 440
规则 442
向 Amazon IoT 规则授予所需的访问权限 443
传递角色权限 444
创建 Amazon IoT 规则 445
查看您的规则 448
删除规则 448
Amazon IoT 规则操作 449
Apache Kafka 450
CloudWatch 警报 457
CloudWatch 日志 458
CloudWatch 指标 460
DynamoDB 461
DynamoDBv2 463
Elasticsearch 465
HTTP 467
IoT Analytics 493
IoT Events 495
IoT SiteWise 496
Kinesis Data Firehose 500
Kinesis Data Streams 502
Lambda 503
位置 505
OpenSearch 508
Republish 509
S3 511
Salesforce IoT 513
SNS 514
SQS 515
Step Functions 517
Timestream 518
排查规则问题 522
使用 Amazon IoT 规则访问跨账户资源 522
先决条件 522
Amazon SQS 的跨账户设置 522
Amazon SNS 的跨账户设置 523
Amazon S3 的跨账户设置 524
Amazon Lambda 跨账户设置 526
错误处理（错误操作） 527
错误操作消息格式 527

错误操作示例 528
借助基本摄取功能，降低消息收发成本 529
使用基本摄取功能 529
Amazon IoT SQL 参考 530
SELECT 子句 531
FROM 子句 532
WHERE 子句 533
数据类型 533
运算符 536
函数 541
文本 582
Case 语句 582
JSON 扩展 583
替换模板 585
嵌套对象查询 586
二进制负载 587
SQL 版本 591
Device Shadow 服务 593
使用影子 593
选择使用命名或未命名的影子 593
访问影子 593
在设备、应用程序和其它云服务中使用影子 594
消息顺序 594
修剪影子消息 595
在设备中使用影子 596
首次连接到 Amazon IoT 时初始化设备 597
在设备连接到 Amazon IoT 后处理消息 598
在设备重新连接到 Amazon IoT 时处理消息 598
在应用程序和服务中使用影子 598
在连接到 Amazon IoT 时初始化应用程序或服务 599
在应用程序或服务连接到 Amazon IoT 后处理状态更改 599
检测是否连接了设备 599
模拟 Device Shadow 服务通信 600
设置模拟 601
初始化设备 601
从应用程序中发送更新 603
响应设备中的更新 605
观察应用程序中的更新 608
模拟补充内容 610
与影子交互 610
协议支持 610
请求和报告状态 610
更新影子 610
检索影子文档 613
删除影子数据 614
Device Shadow REST API 616
GetThingShadow 616
UpdateThingShadow 617
DeleteThingShadow 618
ListNamedShadowsForThing 619
Device Shadow MQTT 主题 620
/get 620
/get/accepted 621
/get/rejected 621
/update 622
/update/delta 623
/update/accepted 624

/update/documents 624
/update/rejected 625
/delete 626
/delete/accepted 626
/delete/rejected 627
Device Shadow 服务文档 627
影子文档示例 628
文档属性 632
增量状态 633
对影子文档进行版本控制 634
影子文档中的客户端令牌 634
空影子文档属性 634
影子文档中的数组值 635
Device Shadow 错误消息 636
任务 637
访问 Amazon IoT 任务 637
Amazon IoT Jobs 区域和终端节点 637
Amazon IoT Jobs 是什么？ 637
任务关键概念 638
任务和任务执行状态 640
管理任务 643
任务的代码签名 643
任务文档 643
预签名 URL 644
使用控制台创建和管理任务 645
使用 CLI 创建和管理任务 647
任务模板 654
自定义和 Amazon 托管模板 654
使用 Amazon 托管式模板 655
创建自定义任务模板 667
任务配置 672
任务配置的工作原理 672
指定其他配置 680
设备和任务 685
对设备进行编程以使用任务 687
设备工作流程 687
任务流 688
任务通知 691
Amazon IoT 任务 API 操作 697
任务管理和控制 API 以及数据类型 698
任务设备 MQTT 和 HTTPS API 操作以及数据类型 711
保护 Jobs 的用户和设备 721
Amazon IoT Jobs 所需的策略类型 721
向 Jobs 用户和云服务授权 721
授权设备使用任务 728
任务限制 730
活动和并发任务限制 730
Amazon IoT 安全隧道 732
什么是安全隧道？ 732
安全隧道概念 732
安全隧道的工作原理 733
安全隧道生命周期 733
Amazon IoT 安全隧道教程 734
本部分中的教程 734
打开隧道并启动与远程设备的 SSH 会话 735
为远程设备打开隧道并使用基于浏览器的 SSH 746
本地代理 749

如何使用本地代理 ... 749
为使用 Web 代理的设备配置本地代理 ... 752
多路复用和同步 TCP 连接 ... 757
多路复用多个数据流 ... 758
使用同步 TCP 连接 ... 760
配置远程设备和使用 IoT 代理 ... 761
IoT 代理代码段 ... 762
控制对隧道的访问 ... 763
隧道访问先决条件 ... 763
隧道访问策略 ... 763
解决安全隧道连接问题 ... 768
客户端访问令牌错误无效 ... 768
客户端令牌不匹配错误 ... 769
远程设备连接问题 ... 769
设备预调配 ... 772
在 Amazon IoT 中预调配设备 ... 772
实例集预调配 API ... 773
使用实例集预调配来预调配没有设备证书的设备 ... 774
通过申请进行预调配 ... 774
由可信用户预调配 ... 775
在 Amazon CLI 中使用预先预调配挂钩 ... 777
预调配具有设备证书的设备 ... 779
单个事物预调配 ... 779
即时预调配 ... 780
批量注册 ... 784
预调配模板 ... 784
参数部分 ... 784
资源部分 ... 785
批量注册的模板示例 ... 788
即时预调配（JITP）的模板示例 ... 789
实例集预调配 ... 790
预先预调配挂钩 ... 793
预先预调配挂钩输入 ... 793
预先预调配挂钩返回值 ... 793
预先预调配挂钩 Lambda 示例 ... 794
为安装设备的用户创建 IAM policy 和角色 ... 795
为将安装设备的用户创建 IAM policy ... 795
为将安装设备的用户创建 IAM 角色 ... 796
更新现有策略以向新模板授权 ... 796
设备预调配 MQTT API ... 797
CreateCertificateFromCsr ... 798
CreateKeysAndCertificate ... 799
RegisterThing ... 801
机群索引 ... 803
管理机群索引 ... 803
事物索引 ... 803
事物组索引 ... 804
托管字段 ... 804
自定义字段 ... 806
管理事物索引 ... 806
管理事物组索引 ... 815
查询聚合数据 ... 816
GetStatistics ... 817
GetCardinality ... 818
GetPercentiles ... 819
GetBucketsAggregation ... 821
Authorization ... 821

查询语法 ... 822
支持的功能 ... 822
不支持的 功能 ... 822
注意 ... 822
事物查询示例 ... 823
事物组查询示例 ... 824
机群指标 ... 825
入门教程 ... 825
管理机群指标 ... 830
基于 MQTT 的文件传输 ... 834
什么是流？ ... 834
在Amazon云中管理流 ... 835
向您的设备授予权限 ... 835
将设备连接到 Amazon IoT ... 835
在设备中使用 Amazon IoT 基于 MQTT 的文件传输 ... 836
使用 DescribeStream Dream 获取流数据 ... 836
从流文件中获取数据块 ... 837
处理来自 Amazon IoT 基于 MQTT 的文件传输的错误 ... 840
FreeRTOS OTA 中的一个示例使用案例 ... 842
Amazon IoT Device Defender ... 843
Amazon IoT Device Defender 入门 ... 843
设置 ... 843
审计指南 ... 844
ML Detect 指南 ... 856
自定义查看 Amazon IoT Device Defender 审计结果的时间和方式 ... 879
审核 ... 888
问题严重性 ... 888
后续步骤 ... 889
审核检查 ... 889
审核命令 ... 914
审计查找结果隐藏 ... 938
Detect ... 947
监控未注册设备的行为 ... 948
安全使用案例 ... 949
概念 ... 952
行为 ... 953
ML Detect ... 955
自定义指标 ... 959
设备端指标 ... 964
云端指标 ... 978
使用维度确定安全配置文件中指标的作用域 ... 985
权限 ... 991
Detect 命令 ... 992
如何使用 Amazon IoT Device Defender detect ... 994
缓解操作 ... 995
审计缓解操作 ... 995
检测缓解操作 ... 997
如何定义和管理缓解操作 ... 997
应用缓解操作 ... 999
权限 ... 1003
缓解操作命令 ... 1006
将 Amazon IoT Device Defender 与其它 Amazon 产品结合使用 ... 1007
在运行 Amazon IoT Greengrass 的设备上使用 Amazon IoT Device Defender ... 1007
将 Amazon IoT Device Defender 与 FreeRTOS 和嵌入式设备搭配使用 ... 1007
配合使用 Amazon IoT Device Defender 和 Amazon IoT Device Management ... 1007
Security Hub 集成 ... 1008
跨服务混淆代理问题防范 ... 1013

设备代理的安全最佳实践 1014
Device Advisor 1016
设置 1016
创建 IoT 事物 1016
创建要用作设备角色的 IAM 角色 1017
为 IAM 用户创建自定义托管策略来使用设备顾问 1018
创建 IAM 用户来使用 Device Advisor。 1018
配置您的设备 1019
在控制台中开始使用 Device Advisor 1020
Device Advisor 工作流 1027
先决条件 1027
创建测试套件定义 1027
获取测试套件定义 1029
获取测试终端节点 1029
启动测试套件运行 1029
运行一个测试套件 1030
停止测试套件运行 1030
获取成功的资格测试套件运行的资格报告 1030
Device Advisor 详细控制台工作流 1031
先决条件 1031
创建测试套件定义 1031
启动测试套件运行 1036
停止测试套件运行（可选） 1038
查看测试套件运行详细信息和日志 1039
下载 Amazon IoT 资格报告 1040
长时间测试控制台的工作流程 1040
Device Advisor VPC 端点（Amazon PrivateLink） 1046
Amazon IoT Core Device Advisor VPC 终端节点注意事项 1047
为 Amazon IoT Core Device Advisor 创建接口 VPC 终端节点 1047
通过 VPC 端点控制对 Amazon IoT Core Device Advisor 的访问 1048
Device Advisor 测试用例 1048
用于测试是否符合 Amazon 设备资格认证计划资格的 Device Advisor 测试案例。 1049
TLS 1049
MQTT 1053
影子 1061
任务执行 1062
权限与策略 1064
长时间测试 1064
Amazon IoT Core 设备位置 1076
测量类型和求解器 1076
Amazon IoT Core 设备定位的工作原理 1076
如何使用 Amazon IoT Core 设备位置 1077
解析 IoT 设备的位置 1078
解析设备位置（控制台） 1078
解析设备位置 (API) 1080
对解析位置时出现的错误进行故障排查 1081
位置求解器和设备有效负载 1082
基于 Wi-Fi 的求解器 1082
基于蜂窝的求解器 1083
IP 反向查找求解器 1086
GNSS 求解器 1086
事件消息 1088
事件消息的生成方式 1088
接收事件消息的策略 1088
启用 Amazon IoT 事件 1088
注册表事件 1091
事物事件 1092

事物类型事件 1093
事物组事件 1095
任务事件 1099
生命周期事件 1102
连接/断开连接事件 1102
订阅/取消订阅事件 1104
适用于 LoRaWAN 的 Amazon IoT Core 1106
介绍 1106
如何使用适用于 LoRaWAN 的 Amazon IoT Core 1106
适用于 LoRaWAN 的 Amazon IoT Core 区域和端点 1107
适用于 LoRaWAN 的 Amazon IoT Core 定价 1107
什么是适用于 LoRaWAN 的 Amazon IoT Core？ 1107
什么是 LoRaWAN？ 1108
适用于 LoRaWAN 的 Amazon IoT Core 工作原理 1108
将网关和设备连接到 Amazon IoT Core for LoRaWAN 1110
设备、网关、配置文件和目标的命名约定 1110
将设备数据映射到服务数据 1110
使用控制将您的设备和网关连接到 Amazon IoT Core for LoRaWAN 1110
描述您的 foAmazon IoT Core r LoRa WAN 资源 1111
将您的网关搭载到 Amazon IoT Core for LoRaWAN 1112
将您的设备搭载到 Amazon IoT Core for LoRaWAN 上 1119
配置 LoRaWAN 设备和网关的位置信息 1129
LoRaWAN 设备定位的工作原理 1129
定位工作流概述 1130
配置资源位置 1131
配置 LoRaWAN 网关的位置 1131
配置 LoRaWAN 设备的位置 1133
通过 VPC 接口端点连接到 Amazon IoT Core for LoRaWAN 1136
搭载 Amazon IoT Core for LoRaWAN 控制层面 API 端点 1138
搭载 Amazon IoT Core for LoRaWAN 数据层面 API 端点 1140
使用 for LoRa WANAmazon IoT Core 管理网关 1145
LoRa 基础站软件要求 1145
使用来自Amazon Partner Device Catalog 的合格网关 1146
使用 CUPS 和 LNS 协议 1146
配置 LoRa WAN 网关的信标和筛选功能 1146
使用 Amazon IoT Core for LoRaWAN 的 CUPS 服务更新网关固件 1150
选择网关以接收 LoRa WAN 下行链路数据流量 1159
使用 Amazon IoT Core for LoRaWAN 管理设备 1161
设备注意事项 1161
使用具有可与 Amazon IoT Core for LoRaWAN 搭配使用的网关的设备 1161
LoRa广域网版本 1161
激活模式 1161
设备类 1161
管理 LoRa WAN 设备与之间的通信Amazon IoT 1162
创建多播组向多台设备发送下行链路有效负载 1167
Amazon IoT Core for LoRaWAN 设备的无线固件更新 (FUOTA) 1176
使用网络分析器实时监控无线资源机群 1184
为网络分析器添加必要的 IAM 角色 1185
创建网络分析器配置并添加资源 1186
使用 WebSockets 串流网络分析器跟踪消息 1191
实时查看和监控网络分析器跟踪消息日志 1199
适用于 LoRaWAN 的 Amazon IoT Core 数据安全性 1201
如何在整个系统中保护数据 1202
LoRaWAN 设备和网关传输安全性 1202
Amazon IoT Core适用于 Amazon Side 1203
如何Amazon IoT Core用于亚马逊人行道？ 1203
Amazon IoT Core适用于亚马逊人行道区域和终端节点 1203

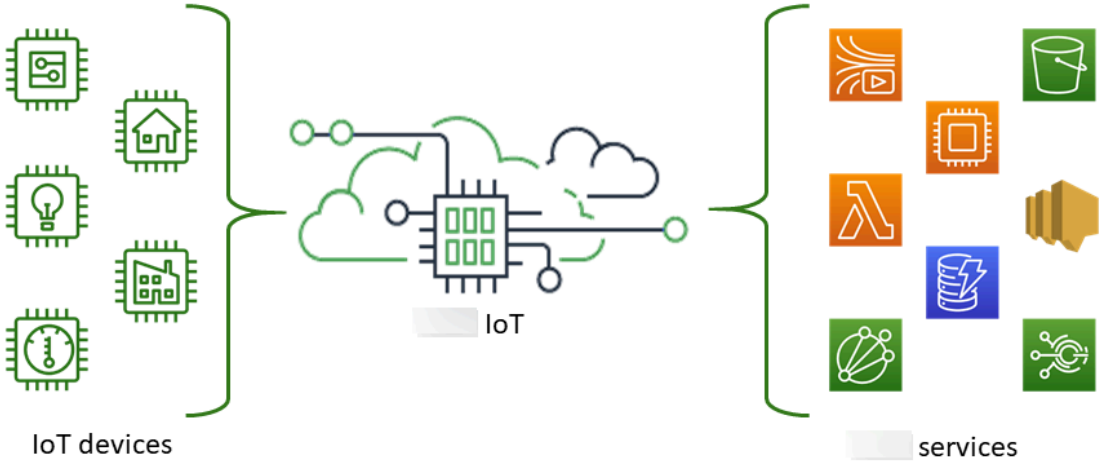
Amazon IoT Core获取亚马逊 Sidewalk 定价 1203
亚马逊人行道有什么Amazon IoT Core用？ 1204
亚马逊人行道的特点Amazon IoT Core 1204
什么是 Amazon Sidewalk？ 1204
亚马逊Amazon IoT Core Sidewalk 的工作原理 1205
亚马逊 SidAmazon IoT Core ewalk 入门 1206
试试传感器监控教程 1207
设置 1207
描述您的Sidewalk 资源 1208
入门使用 Sidewalk 设备简介 1210
将 Sidewalk 设备连接到亚马逊Amazon IoT Core Sidewalk 1212
先决条件 1212
描述您的 Sidewalk 资源 1213
添加您的 Sidewalk 设备 1213
为您的 Sidewalk 设备添加目标 1219
连接 Sidewalk 设备 1223
用Amazon IoT Core于 Amazon Sidewalk 的批量配置设备 1225
亚马逊 Sidewalk 批量配置工作流程 1225
在出厂支持下创建设备配置文件 1228
使用导入任务配置 Sidewalk 设备 1231
Amazon IoT Core用于亚马逊 Sidewalk API 操作 1238
人行道设备配置文件的 API 操作 1238
人行道终端设备的 API 操作 1239
人行道设备目的地的 API 操作 1241
用于批量配置的 API 操作 1242
亚马逊 SidewalAmazon IoT Core k 的监控和事件 1245
Sidewalk 设备活动 1245
监控Sidewalk 设备 1246
监控和记录Amazon IoT Wireless使用亚马逊的情况 CloudWatch 1248
为 Amazon IoT Wireless 配置日志记录 1248
为 Amazon IoT Wireless 创建日志记录角色和策略 1249
为 Amazon IoT Wireless 资源配置日志记录 1251
Amazon IoT Wireless使用 CloudWatch 日志进行监控 1258
查看 CloudWatch Amazon IoT Wireless日志条目 1258
使用 CloudWatch Insights 为筛选日志Amazon IoT Wireless 1263
Amazon IoT Wireless 的事件通知 1266
如何通知资源有关事件 1266
事件和资源类型 1266
用于接收无线事件通知的策略 1266
无线事件的 MQTT 主题的格式 1267
无线事件的定价 1269
启用无线资源的事件 1269
事件配置 1269
先决条件 1269
使用 Amazon Web Services Management Console 启用通知 1270
使用 Amazon CLI 启用通知 1271
LoRaWAN 资源的事件通知 1272
LoRaWAN 资源的事件类型 1272
LoRaWAN 加入事件 1272
连接状态事件 1274
Sidewalk 资源的事件通知 1276
Sidewalk 资源的事件类型 1276
设备注册状态事件 1276
近似事件 1278
消息传输状态事件 1280
Amazon IoT Device SDK、Mobile SDK 和 Amazon IoT Device Client 1283
Amazon IoT Device SDK 1283

Amazon IoT Device SDK for Embedded C 1284
早期 Amazon IoT Device SDK 版本 1284
AmazonMobile SDK 1285
Amazon IoT Device Client 1285
问题排查 1287
诊断连接问题 1287
连接 1287
身份验证 1288
授权 1288
安全和身份 1289
诊断规则问题 1289
配置 CloudWatch Logs 以进行故障排除 1289
诊断外部服务 1290
诊断 SQL 问题 1290
诊断影子问题 1291
诊断 Salesforce 操作问题 1291
执行跟踪 1292
操作成功和失败 1292
机群索引排除指南 1292
队列索引服务的聚合查询疑难解答 1292
机群故障排除指标 1293
解决“您的Amazon账户已超过流限制”的问题 1294
Amazon IoT Device Defender 故障排除指南 1294
Amazon IoT Device Advisor 故障排除指南 1297
排除设备机群断开连接的故障 1298
Amazon IoT 错误 1299
Amazon IoT 配额 1301
..... mcccii

什么是 Amazon IoT？

Amazon IoT 提供云服务将 IoT 设备连接到其它设备和 Amazon 云服务。Amazon IoT 提供设备软件以帮助您将 IoT 设备集成到基于 Amazon IoT 的解决方案。如果您的设备可以连接到 Amazon IoT，则 Amazon IoT 可以将它们连接到 Amazon 提供的云服务。

有关 Amazon IoT 的实践经验介绍，请访问 Amazon IoT Core 入门 (p. 15)。



Amazon IoT 帮助您为解决方案选择最合适和最新的技术。为了帮助您在现场管理和支持 IoT 设备，Amazon IoT Core 支持以下协议：

- MQTT（消息队列和遥测传输） (p. 77)
- 采用 WSS 的 MQTT (Websockets Secure) (p. 77)
- HTTPS（安全超文本传输协议） (p. 91)
- LoRaWAN（远距离广域网） (p. 1106)

Amazon IoT Core 消息代理支持使用 MQTT 和 采用 WSS 的 MQTT 协议发布和订阅消息的设备和客户端。它还支持使用 HTTPS 协议发布消息的设备和客户端。

适用于 LoRaWAN 的 Amazon IoT Core 将帮助您连接和管理无线 LoRaWAN（低功耗远距离广域网）设备。适用于 LoRaWAN 的 Amazon IoT Core 取代了您开发和操作 LoRaWAN 网络服务器 (LNS) 的需求。

如果您不需要 Amazon IoT 功能（例如设备通信、规则 (p. 442)或任务 (p. 637)，请参阅Amazon消息收发以了解可更好地满足您需求的其它 Amazon IoT 消息收发服务的信息。

您的设备和应用程序如何访问 Amazon IoT

Amazon IoT 在 Amazon IoT 教程 (p. 114)中提供了以下几个接口：

- Amazon IoT Device SDK - 构建在设备上运行的应用程序，以便与 Amazon IoT 收发消息。有关更多信息，请参阅Amazon IoT Device SDK、Mobile SDK 和 Amazon IoT Device Client (p. 1283)。

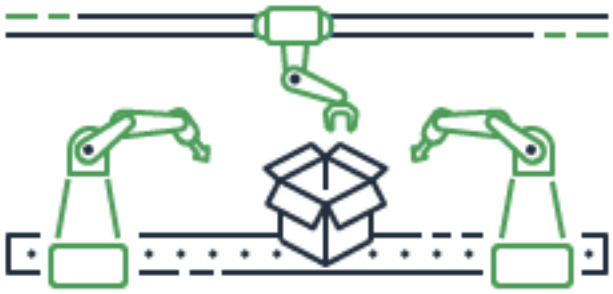
- 适用于 LoRaWAN 的 Amazon IoT Core — 使用适用于 LoRaWAN 的 Amazon IoT Core (p. 1106) 连接和管理远距离 WAN (LoRaWAN) 设备和网关。
- Amazon Command Line Interface (Amazon CLI) — 在 Windows、macOS 和 Linux 上运行 Amazon IoT 的命令。您可以使用这些命令创建并管理事物对象、证书、规则、任务和策略。要开始使用，请参阅 Amazon Command Line Interface 用户指南。有关 Amazon IoT 命令的更多信息，请参阅 Amazon CLI 命令参考中的 iot。
- Amazon IoT API — 使用 HTTP 或 HTTPS 请求构建您的 IoT 应用程序。您可以使用这些 API 操作以编程方式创建和管理事物对象、证书、规则及策略。有关适用于 Amazon IoT 的 API 操作的更多信息，请参阅 Amazon IoT API 参考中的操作。
- Amazon SDK - 使用语言特定 API 构建您的 IoT 应用程序。这些 SDK 中封装了 HTTP/HTTPS API，并且您可以用任何受支持的语言进行编程。有关更多信息，请参阅 Amazon SDK 和工具。

您也可以通过 Amazon IoT 控制台访问 Amazon IoT，它提供了一个图形用户界面 (GUI)，您可以通过该界面配置和管理 IoT 解决方案的事物对象、证书、规则、任务、策略和其它元素。

Amazon IoT 能做的事情

本主题介绍了您可能需要 Amazon IoT 支持的一些解决方案。

行业中的 IoT



以下是一些工业使用案例的示例 Amazon IoT 解决方案，它们应用 IoT 技术来提高工业流程的性能和生产力。

用于工业使用案例的解决方案

- 使用 Amazon IoT 在工业运营中构建预测性质量模型

  了解 Amazon IoT 如何收集和分析工业运营数据以构建预测性质量模型。了解更多
- 使用 Amazon IoT 支持工业运营中的预测性维护

  了解 Amazon IoT 如何帮助规划预防性维护以减少计划外停机时间。了解更多

家居自动化中的 IoT

以下是一些家居自动化使用案例的示例 Amazon IoT 解决方案，应用 IoT 技术构建可扩展的 IoT 应用程序，使用互联家居设备自动化家庭活动。

家居自动化解决方案

- 在您的互联家居中使用 Amazon IoT

  了解 Amazon IoT 如何提供一体式家居自动化解决方案。
- 使用 Amazon IoT 提供家居安全和监控

  了解 Amazon IoT 如何将机器学习和边缘计算应用于家居自动化解决方案。

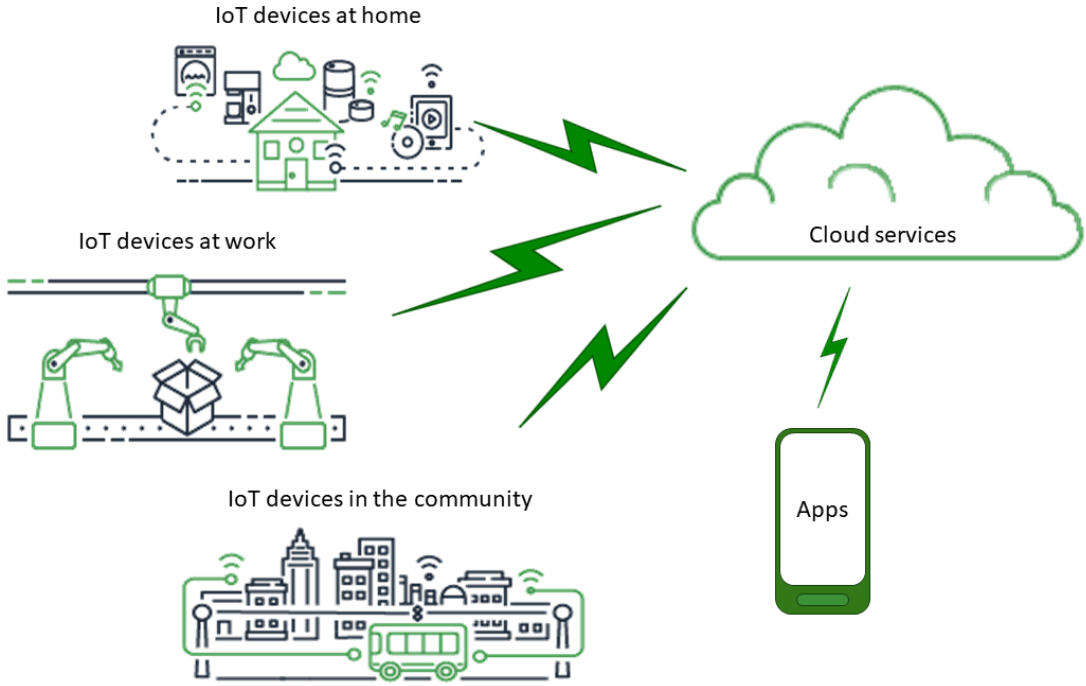
有关适用于工业、消费者和商业使用案例的解决方案列表，请参阅 Amazon IoT 解决方案存储库。

Amazon IoT 的工作原理

Amazon IoT 提供可用于实施 IoT 解决方案的云服务和设备支持。Amazon 提供了许多云服务来支持基于 IoT 的应用程序。因此，为了帮助您了解从哪里开始，本部分提供了基本概念图表和定义，向您介绍 IoT 世界。

IoT 世界

一般来说，物联网 (IoT) 由下图所示的关键组成部分组成。



应用程序

应用程序让最终用户可以访问 IoT 设备以及与这些设备相连的云服务所提供的功能。

云服务

云服务是连接到互联网的分布式大规模数据存储和处理服务。示例包括：

- IoT 连接和管理服务

  Amazon IoT 是 IoT 连接和托管服务的一个示例。
- 计算服务，如 Amazon Elastic Compute Cloud 和 Amazon Lambda
- 数据库服务，例如 Amazon DynamoDB

通信

设备通过使用各种技术和协议与云服务进行通信。示例包括：

- Wi-Fi/宽带互联网
- 宽带蜂窝数据
- 窄带蜂窝数据
- 远距离广域网 (LoRaWAN)
- 专有的射频通信

设备

设备是管理接口和通信的一种硬件。设备通常位于其监控和控制的真实接口附近。设备可以包括计算和存储资源，例如微控制器、CPU、内存。示例包括：

- Raspberry Pi
- Arduino
- 语音接口助手
- LoRaWAN 和设备
- Amazon Sidewalk 设备
- 自定义 IoT 设备

接口

接口是将设备连接到物理世界的组件。

- 用户接口

  允许设备和用户相互通信的组件。

  - 输入接口

    使用户能够与设备通信

    示例：键盘、按钮
  - 输出接口

    使设备能够与用户通信

    示例：字母数字显示、图形显示、指示灯、闹铃
- 传感器

  以设备理解的方式测量或感知外部世界中的某些内容的输入组件。示例包括：

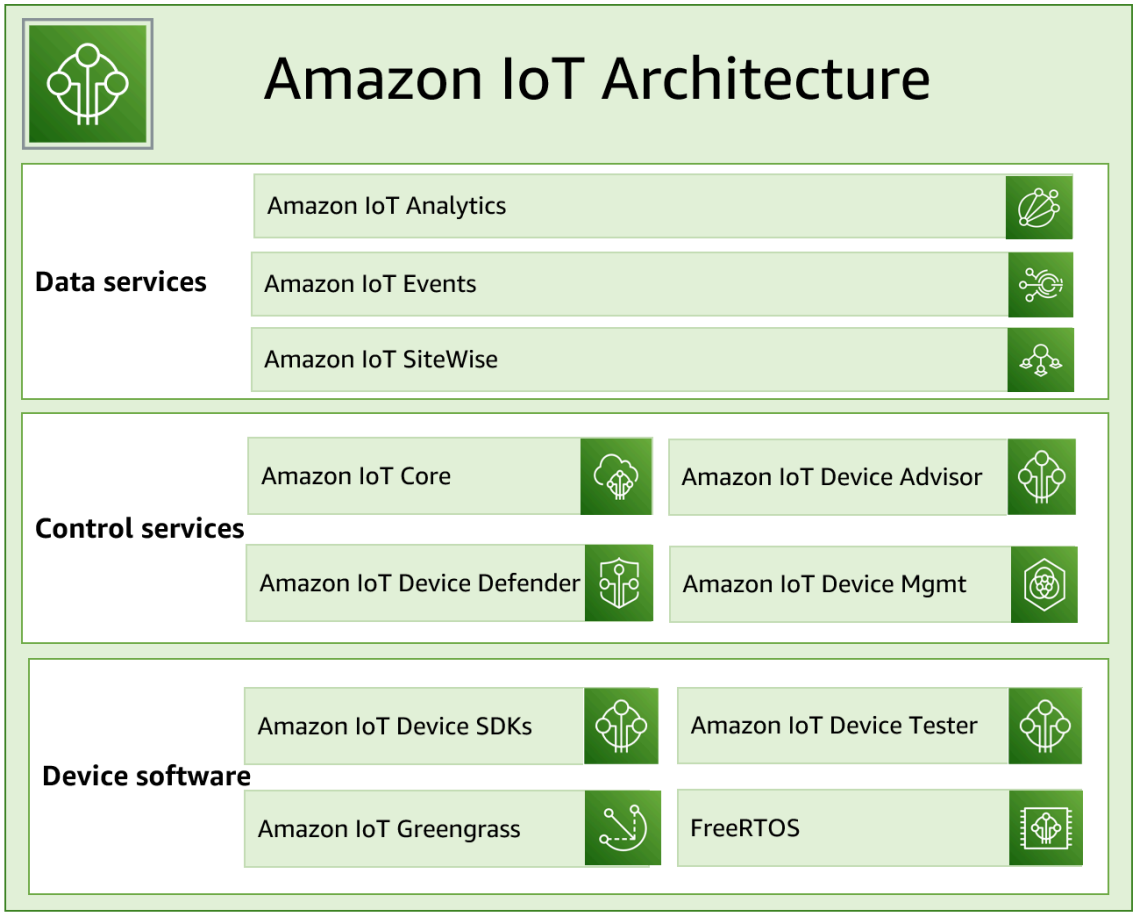
  - 温度传感器（将温度转换为模拟或数字信号）
  - 湿度传感器（将相对湿度转换为模拟数字信号）
  - 模拟到数字转换器（将模拟电压转换为数值）
  - 超声波距离测量装置（将距离转换为数值）
  - 光学传感器（将光度转换为数值）
  - 相机（将图像数据转换为数字数据）
- 执行器

  设备可用于控制外部世界中的某些内容的输出组件。示例包括：

  - 步进电机（将电信号转换为移动）
  - 继电器（控制高电压和电流）

Amazon IoT 服务概览

在 IoT 世界中，Amazon IoT 提供了一些服务来支持与真实世界互动的设备以及在这些设备与 Amazon IoT 之间传递的数据。Amazon IoT 由此图中显示的支持 IoT 解决方案的服务组成。



Amazon IoT 设备软件

Amazon IoT 提供此软件来支持您的 IoT 设备。

Amazon IoT Device SDK

Amazon IoT 设备和移动 SDK (p. 1283) 可帮助您高效地将设备连接到 Amazon IoT。Amazon IoT 设备和移动 SDK 包括开源库、开发人员指南（含示例）和移植指南，便于您在自己选择的硬件平台上构建富有创新精神的 IoT 产品或解决方案。

Amazon IoT Device Tester

适用于 FreeRTOS 的 Amazon IoT Device Tester 和 Amazon IoT Greengrass 是用于微控制器的自动化测试工具。Amazon IoT Device Tester 可以测试您的设备，确定设备是运行 FreeRTOS 还是 Amazon IoT Greengrass，以及是否与 Amazon IoT 服务进行互操作。

Amazon IoT Greengrass

Amazon IoT Greengrass 可将 Amazon IoT 无缝扩展至边缘设备，因此可以在本地操作其生成的数据，并将云用于管理、分析和持久存储。借助 Amazon IoT Greengrass，互联设备可以运行 Amazon Lambda 函数、Docker 容器，或同时运行两者，基于机器学习模型执行预测、使设备数据保持同步以及与其它设备安全通信 – 甚至在没有连接互联网的情况下也可实现这些功能。

FreeRTOS

FreeRTOS 是一款面向微控制器的开源实时操作系统，可让您在 IoT 解决方案中包含小型低功耗边缘设备。FreeRTOS 包括一个内核和越来越多的、支持许多应用程序的软件库。FreeRTOS 系统可以安全地将小型低功耗设备连接到 Amazon IoT，并支持运行 Amazon IoT Greengrass 的更强大的边缘设备。

Amazon IoT 控制服务

连接到以下 Amazon IoT 服务以管理 IoT 解决方案中的设备。

Amazon IoT Core

Amazon IoT Core 是一项托管云服务，它使互联设备能够安全地与云应用程序和其他设备进行交互。Amazon IoT Core 可以支持许多设备和消息，还可以处理这些消息并路由到 Amazon IoT 端点和其它设备。有了 Amazon IoT Core，您的应用程序即使未连接，也可以与所有设备进行交互。

Amazon IoT Core Device Advisor

Amazon IoT Core Device Advisor 是一种基于云的完全托管式测试功能，用于在设备软件开发过程中验证 IoT 设备。Device Advisor 提供预构建的测试，您可以使用这些测试验证 IoT 设备与 Amazon IoT Core 之间的连接是否安全可靠，然后再将设备部署到生产环境。

Amazon IoT Device Defender

Amazon IoT Device Defender 可帮助您保护 IoT 设备机群。Amazon IoTDevice Defender 会不断审计您的 IoT 配置，以确保配置始终遵循安全最佳实践。Amazon IoT如果 Device Defender 在您的 IoT 配置中检测到任何可能引发安全风险的漏洞，例如在多个设备间共享身份证书，或身份证书被吊销的设备试图连接 Amazon IoT Core，该服务会发送提醒。

Amazon IoT Device Management

Amazon IoT Device Management 服务可帮助您跟踪、监控和管理构成设备机群的大量连接设备。Amazon IoTDevice Management 服务可帮助您确保 IoT 设备在部署后能够正常、安全地运行。它们提供安全隧道来访问您的设备、监控设备的运行状况、检测和远程排查问题，还提供管理设备软件和固件更新的服务。

Amazon IoT 数据服务

分析 IoT 解决方案中设备的数据，并使用以下 Amazon IoT 服务采取适当的措施。

Amazon IoT Analytics

Amazon IoT 分析 可让您高效地对大量非结构化 IoT 数据运行和实施复杂分析。Amazon IoT Analytics 可以自动执行分析来自 IoT 设备的数据所需的每个困难步骤。Amazon IoT Analytics 对 IoT 数据进行筛选、转换和丰富，然后再将其存储在时间序列数据存储中以供分析。通过使用内置的 SQL 查询引擎或机器学习，您可以运行一次性查询或计划查询来分析数据。

Amazon IoT Events

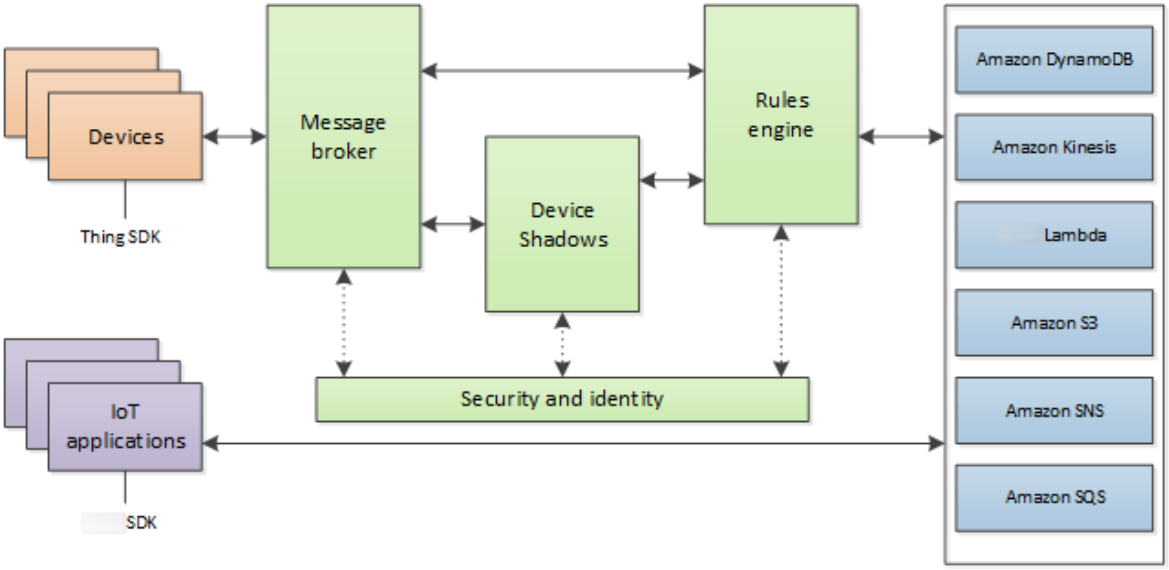
Amazon IoT 事件检测并响应来自 IoT 传感器和应用程序的事件。事件是识别比预期更复杂的情况的数据模式，例如运动检测器使用移动信号来激活灯和监控摄像机。Amazon IoT事件会持续监控来自多个 IoT 传感器和应用程序的数据，并与其它服务（例如 Amazon IoT Core、IoT SiteWise、DynamoDB 等）集成，以实现早期检测并获取独特见解。

Amazon IoT SiteWise

Amazon IoT SiteWise 通过提供在您设施的网关上运行的软件，大规模地收集、存储、组织和监控 MQTT 消息或 API 从工业设备传送的数据。该网关安全地连接到您的本地数据服务器，并自动执行收集和组织数据以及将数据发送到 Amazon 云的过程。

Amazon IoT Core 服务

Amazon IoT Core 提供将您的 IoT 设备连接到 Amazon 云的服务，以便其它云服务和应用程序可以与您的互联网连接设备进行交互。



下一部分将介绍示意图中显示的每个 Amazon IoT Core 服务。

Amazon IoT Core 消息收发服务

Amazon IoT Core 连接服务提供与 IoT 设备的安全通信，并管理它们与 Amazon IoT 之间传递的消息。

设备网关

使设备能够安全高效地与 Amazon IoT 进行通信。设备通信由使用 X.509 证书的安全协议提供保护。

消息代理

提供安全机制以供设备和 Amazon IoT 应用程序用于相互发布和接收消息。进行发布和订阅时，您可以直接使用 MQTT 协议，也可以通过 WebSocket 使用 MQTT 协议。有关受 Amazon IoT 支持的协议的更多信息，请参阅 the section called “设备通信协议” (p. 75)。设备和客户端也可以使用 HTTP REST 接口将数据发布到消息代理。

消息代理将设备数据分发到已订阅设备数据的设备和其它 Amazon IoT Core 服务，例如 Device Shadow 服务和规则引擎。

适用于 LoRaWAN 的 Amazon IoT Core

适用于 LoRaWAN 的 Amazon IoT Core 可以通过将您的 LoRaWAN 设备和网关连接到 Amazon 而无需开发和操作 LoRaWAN 网络服务器 (LNS)，从而设置私有 LoRaWAN 网络。从 LoRaWAN 设备接收的消息将发送到规则引擎，在该引擎中可以对其进行格式化并发送到其它 Amazon IoT 服务。

规则引擎

规则引擎将消息代理中的数据连接到其它 Amazon IoT 服务以进行存储和额外处理。例如，您可以插入、更新或查询 DynamoDB 表，也可以根据在规则引擎中定义的表达式调用 Lambda 函数。您可以使用基于 SQL 的语言选择消息负载中的数据，然后处理数据并将数据发送到其它服务，如 Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB 和 Amazon Lambda。您还可以创建规则以将消息重新发布到消息代理并面向其他订阅者。有关更多信息，请参阅Amazon IoT 规则 (p. 442)。

Amazon IoT Core 控制服务

Amazon IoT Core 控制服务提供设备安全、管理和注册功能。

自定义身份验证服务

您可以定义自定义授权方，从而通过自定义的身份验证服务和 Lambda 函数来管理自己的身份验证和授权策略。自定义授权方使得 Amazon IoT 可以使用持有者令牌身份验证和授权策略，对您的设备进行身份验证和授权操作。

自定义授权方可以实施各种身份验证策略；例如 JSON Web 令牌验证或 OAuth 提供程序标注。它们必须返回设备网关用于授权 MQTT 操作的策略文档。有关更多信息，请参阅自定义身份验证 (p. 287)。

设备预配置服务

让您可使用描述设备所需资源的模板来预配置设备：事物对象、证书以及一个或多个策略。事物对象是注册表中的一个条目，其中包含描述设备的属性。设备使用证书通过 Amazon IoT 进行身份验证。策略确定设备在 Amazon IoT 中可执行的操作。

模板包含可用字典中的值替换的变量 (映射)。您可以使用同一个模板来预配置多个设备，只需在字典中为模板变量传递不同的值。有关更多信息，请参阅设备预调配 (p. 772)。

组注册表

通过将设备按类别分成不同的组，可以使用组来同时管理多台设备。组中还可以包含组，您可以构建组的层次结构。您对父组执行的任何操作都将应用于其子组。同样的操作也应用于父组中的所有设备以及子组中的所有设备。向某个组授予的权限将应用于该组及其子组中的所有设备。有关更多信息，请参阅使用 Amazon IoT 管理设备 (p. 239)。

Jobs 服务

允许您定义一组远程操作，这些操作将被发送到一个或多个连接到 Amazon IoT 的设备并在这些设备上运行。例如，您可以定义一个任务，该任务指示一组设备下载并安装应用程序或固件更新、重启、轮换证书或执行远程故障排除操作。

要创建任务，您需要指定要执行的远程操作的说明，以及应该执行这些操作的目标列表。目标可以是单个设备和/或设备组。有关更多信息，请参阅任务 (p. 637)。

注册表

整理与 Amazon 云中的每台设备关联的资源。您可以注册自己的设备并将每台设备与最多三个自定义属性关联。您还可以将每台设备与相应的证书和 MQTT 客户端 ID 关联，以提高对设备进行管理和故障排除的能力。有关更多信息，请参阅使用 Amazon IoT 管理设备 (p. 239)。

安全和身份服务

在 Amazon 云中共担安全责任。为了安全地将数据发送到消息代理，您的设备必须确保自身凭证的安全。消息代理和规则引擎使用 Amazon 安全功能将数据安全发送到设备或其它 Amazon 服务。有关更多信息，请参阅身份验证 (p. 264)。

Amazon IoT Core 数据服务

即使在不总是连接的设备上，Amazon IoT Core 数据服务也能帮助您的 IoT 解决方案提供可靠的应用程序体验。

Device Shadow

一种 JSON 文档，用于存储和检索设备的当前状态信息。

Device Shadow 服务

Device Shadow 服务会保持设备的状态，以便应用程序可以与设备通信，无论设备是否在线。当设备处于离线状态时，Device Shadow 服务将管理它连接的应用程序的数据。当设备重新连接时，它会将其状态与 Device Shadow 服务中的状态同步。您的设备还可以将有关其当前状态的信息发布到影子，以供不会始终保持连接的应用程序或其它设备使用。有关更多信息，请参阅Amazon IoT Device Shadow 服务 (p. 593)。

Amazon IoT Core 支持服务

适用于 Amazon IoT Core 的 Amazon Sidewalk 集成

Amazon Sidewalk 是一个共享网络，可改进连接选项，帮助设备更好地协同工作。Amazon Sidewalk 支持各种客户设备，例如定位宠物或贵重物品的设备、提供智能家居安全和照明控制的设备，以及为家电和工具提供远程诊断的设备。适用于 Amazon IoT Core 的 Amazon Sidewalk 集成使设备制造商可以将他们的 Sidewalk 设备机群添加到 Amazon IoT 云。

有关更多信息，请参阅 Amazon IoT Core适用于 Amazon Side (p. 1203)

了解有关 Amazon IoT 的更多信息

本主题可以帮助您熟悉 Amazon IoT 的世界。您可以获得有关如何在各种使用案例中应用 IoT 解决方案的一般信息、培训资源、用于 Amazon IoT 的社交媒体链接和所有其它 Amazon 服务，以及 Amazon IoT 使用的服务和通信协议列表。

Amazon IoT 的培训资源

这些培训课程旨在帮助您了解 Amazon IoT 以及如何将它们应用到解决方案设计中。

- Amazon IoT 简介

  Amazon IoT 及其核心服务的视频概览。
- 深入了解 Amazon IoT 身份验证和授权

  探索 Amazon IoT 身份验证和授权概念的高级课程。您将学习如何验证和授权客户端以访问 Amazon IoT 控制平面和数据平面 API。
- 物联网基础系列

  关于不同 IoT 技术和功能的 IoT 电子学习模块的学习路径。

Amazon IoT 资源和指南

这些是关于 Amazon IoT 的具体方面的深入技术资源。

- IoT 详解 – Amazon IoT Well-Architected Framework

  一份描述在 Amazon 上构架 IoT 应用程序的最佳实践的 PDF 文档。
- 设计用于 Amazon IoT Core 的 MQTT 主题

  一份 PDF 文档，介绍在 Amazon IoT Core 中设计 MQTT 主题和借助 MQTT 利用 Amazon IoT Core 功能的最佳实践。
- 在 Amazon IoT Core 中使用 X.509 证书进行设备制造和配置

  一份描述 Amazon IoT 提供预置大型设备机群不同方式的 PDF 文档。
- Amazon IoT Core Device Advisor

  Amazon IoT Core Device Advisor 提供预构建的测试，您可以使用这些测试验证 Amazon IoT Core 的可靠和安全连接最佳实践，然后再将设备部署到生产环境。
- Amazon IoT 资源

  IoT 特定的资源，例如技术指南、参考架构、电子书和精选的博客文章，在可搜索索引中显示。

- IoT Atlas

  有关如何解决常见的 IoT 设计问题的概览。IoT Atlas 深入探讨了您在开发 IoT 解决方案时可能遇到的设计挑战。
- Amazon 白皮书和指南

  我们目前收集的有关 Amazon IoT 和其它 Amazon 技术的白皮书和指南。

社交媒体中的 Amazon IoT

这些社交媒体渠道提供有关 Amazon IoT 和 Amazon 相关主题的信息。

- Amazon IoT 上的物联网 – 官方博客
- Amazon IoTYouTube 上 Amazon Web Services 频道中的视频

这些社交媒体账户涵盖所有 Amazon 服务，包括 Amazon IoT

- YouTube 上的 Amazon Web Services
- Twitter 上的 Amazon Web Services
- Facebook 上的 Amazon Web Services
- Instagram 上的 Amazon Web Services
- LinkedIn 上的 Amazon Web Services

Amazon IoT Core 规则引擎使用的 Amazon 服务

Amazon IoT Core 规则引擎可以连接到这些 Amazon 服务。

- Amazon DynamoDB

  Amazon DynamoDB 是一种可扩展的 NoSQL 数据库服务，可提供快速且可预测的数据库性能。
- Amazon Kinesis

  Amazon Kinesis 可让您轻松收集、处理和分析实时串流数据，以便您及时获得见解并对新信息快速做出响应。Amazon Kinesis 可以摄取视频、音频、应用程序日志和网站点击流等实时数据，也可以获取用于机器学习、分析和其它应用程序的 IoT 遥测数据。
- Amazon Lambda

  利用 Amazon Lambda，您可以运行代码，而无需预置或管理服务器。您可以将您的代码设置为自动从 Amazon IoT 数据和事件触发，或者直接从 Web 或移动应用程序调用。
- Amazon Simple Storage Service

  Amazon Simple Storage Service (Amazon S3) 可以随时从任何位置在 Web 上存储和检索任意数量的数据。Amazon IoT 规则可以将数据发送到 Amazon S3 进行存储。
- Amazon Simple Notification Service

  Amazon Simple Notification Service (Amazon SNS) 是一项 Web 服务，通过该服务，应用程序、终端用户和设备可以发送和接收来自云的通知。
- Amazon Simple Queue Service

  Amazon Simple Queue Service (Amazon SQS) 是一种消息队列服务，可解耦和扩展微服务、分布式系统和无服务器应用程序。
- Amazon OpenSearch Service

Amazon OpenSearch Service (OpenSearch Service) 是一项托管式服务，让用户能够轻松地部署、操作和扩展 OpenSearch（一个流行的开源搜索和分析引擎）。

- Amazon SageMaker

  Amazon SageMaker 可通过查找您 IoT 数据中的模式来创建机器学习 (ML) 模型。然后，该服务会使用这些模型处理新数据并为应用程序生成预测结果。

- Amazon CloudWatch

  Amazon CloudWatch 提供可靠、可扩展且灵活的监控解决方案，帮助您设置、管理和扩展自己的监控系统和基础设施。

Amazon IoT Core 支持的通信协议

这些主题提供了有关 Amazon IoT 使用的通信协议的更多信息。有关 Amazon IoT 使用的协议以及将设备和服务连接到 Amazon IoT 的更多信息，请参阅 连接到 Amazon IoT Core (p. 64)。

- MQTT（消息队列遥测传输）

  MQTT.org 网站的主页，您可以在其中找到 MQTT 协议规范。有关 Amazon IoT 如何支持 MQTT 的更多信息，请参阅 MQTT (p. 77)。

- HTTPS（安全超文本传输协议）

  设备和应用程序可以使用 HTTPS 访问 Amazon IoT 服务。

- LoRaWAN（远距离广域网）

  LoRaWAN 设备和网关可以使用适用于 LoRaWAN 的 Amazon IoT Core 连接到 Amazon IoT Core。

- TLS（传输层安全性）v1.2

  TLS v1.2 (RFC 5246) 的规范。Amazon IoT 使用 TLS v1.2 在设备和 Amazon IoT 之间建立安全连接。

Amazon IoT 控制台中的新增功能

我们正在致力更新 Amazon IoT 控制台的用户界面以提供新的体验。我们正在分阶段更新用户界面，因此控制台中的某些页面将具有新的体验，有些可能同时具有原始体验和新体验，而有些可能只有原始体验。

此表显示了截至 2022 年 1 月 27 日 Amazon IoT 控制台用户界面中各个区域的状态。

Amazon IoT 控制台用户界面状态

控制台页面	原始体验	新体验	注释
监控	不可用	可用	
活动	不可用	可用	
注册 - 入门	不可用	可用	（并未在 CN 区域提供）
注册 - 实例集预调配模板	可用	可用	
管理 — 事物	可用	可用	
管理 — 类型	可用	可用	
管理 — 事物组	可用	可用	

控制台页面	原始体验	新体验	注释
管理 — 账单组	可用	可用	
管理 — 任务	可用	可用	
管理 — 任务模板	不可用	可用	
管理 — 隧道	不可用	可用	
Fleet Hub — 入门	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
Fleet Hub — 应用程序	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
Greengrass — 入门	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
Greengrass — 核心设备	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
Greengrass — 组件	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
Greengrass — 部署	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
Greengrass — 经典 (V1)	可用	可用	
无线连接 — 简介	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
无线连接 — 网关	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
无线连接 — 设备	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
无线连接 — 配置文件	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
无线连接 — 目标	不可用	可用	在所有 Amazon Web Services 区域 中暂不可用
安全 — 证书	可用	可用	
安全 — 策略	可用	可用	

控制台页面	原始体验	新体验	注释
安全 — CA	可用	可用	
安全 — 角色别名	可用	可用	
安全 — 授权方	可用	可用	
防护 — 简介	不可用	可用	
防护 — 审计	不可用	可用	
防护 — 检测	不可用	可用	
防护 — 缓解操作	不可用	可用	
防护 — 设置	不可用	可用	
操作 — 规则	可用	可用	
操作 — 目标	可用	可用	
测试 — Device Advisor	可用	可用	在所有 Amazon Web Services 区域 中暂不可用
测试 — MQTT 测试客户端	可用	可用	
软件	可用	可用	
设置	不可用	可用	
学习	可用	暂不可用	

图例

状态值

- 可用

  此用户界面体验可以使用。
- 不可用

  此用户界面体验无法使用。
- 暂不可用

  正在开发新的用户界面体验，但尚未准备好。
- 正在进行中

  新的用户界面体验正在更新。但是，某些页面可能仍然具有原始用户体验。

Amazon IoT Core 入门

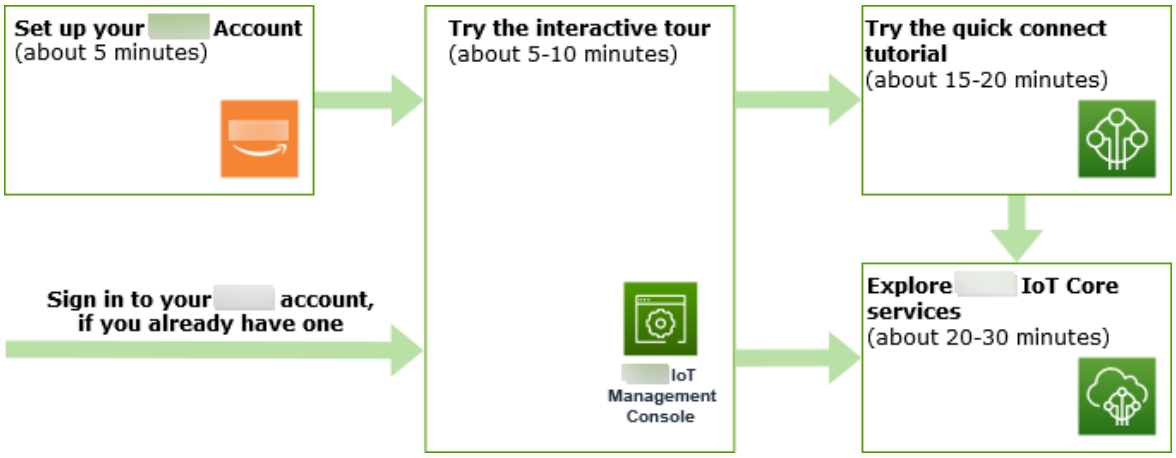
无论您是 IoT 的新手还是拥有多年经验，这些资源都能提供 Amazon IoT 概念和术语，以助您开始使用 Amazon IoT。

- 观看 Amazon IoT 内部及其在 Amazon IoT 的工作原理 (p. 3) 中的组件。
- 利用我们的培训材料和视频集详细了解Amazon IoT (p. 10)。本主题还包括 Amazon IoT 可连接的服务列表、社交媒体链接以及通信协议规范的链接。
- the section called “将第一个设备连接到 Amazon IoT Core” (p. 15).
- 使用 连接到 Amazon IoT Core (p. 64) 开发您的 IoT 解决方案并探索 Amazon IoT 教程 (p. 114)。
- 使用 Device Advisor (p. 1016) 测试和验证 您的 IoT 设备，确保安全可靠的连接。
- 使用诸如 机群索引 (p. 803)、任务 (p. 637) 和 Amazon IoT Device Defender (p. 843) 等 Amazon IoT Core 管理服务管理您的解决方案。
- 使用 Amazon IoT 数据服务 (p. 7) 分析设备的数据。

将第一个设备连接到 Amazon IoT Core

Amazon IoT Core 服务将 IoT 设备连接到 Amazon IoT 服务和其它 Amazon 服务。Amazon IoT Core 包括设备网关和消息代理，用于连接和处理 IoT 设备与云之间的消息。

下面介绍了有关 Amazon IoT Core 和 Amazon IoT 的入门信息。



此部分包含 Amazon IoT Core 的入门路线图，为您介绍其关键服务，并提供了几个示例说明如何将设备连接到 Amazon IoT Core 并在它们之间传递消息。在设备和云之间传递消息对于每个 IoT 解决方案至关重要，也是您的设备与其它 Amazon 服务交互的方式。

- 设置您的Amazon Web Services 账户 (p. 16)

  在您可以使用 Amazon IoT 服务前，您必须设置 Amazon Web Services 账户。如果您已经有Amazon Web Services 账户和 IAM 用户，您可以使用它们并跳过此步骤。
- 尝试交互式教程 (p. 17)

  如果您想要了解基础 Amazon IoT 解决方案在没有连接设备或下载任何软件的情况下可以做什么，这个演示是最好的材料。交互式教程提供一个基于 Amazon IoT Core 服务构建的模拟解决方案，其中说明了它们之间交互的方式。

- 尝试快速连接教程 (p. 20)

  如果您想快速入门 Amazon IoT，并了解其在有限的情况下的工作方式，本教程是最好的材料。在本教程中，您将需要一台设备，并安装一些 Amazon IoT 软件。如果您没有 IoT 设备，可使用 Windows、Linux 或 macOS 个人计算机作为本教程的设备。如果您想尝试 Amazon IoT，但您没有设备，请尝试下一个选项。

- 通过详细教程了解 Amazon IoT Core 服务 (p. 33)

  本教程最适合想要入门 Amazon IoT 的开发人员，在此之后，他们就可以继续探索其它的 Amazon IoT Core 功能，例如规则引擎和影子。本教程遵循的流程类似于快速连接教程，但为每个步骤提供了更多详细信息，以便学员能够更平滑地过渡到更高级的教程。

- 使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)

  了解如何使用 MQTT 测试客户端观看您的第一台设备将 MQTT 消息发布到 Amazon IoT。MQTT 测试客户端可用于监控和排除设备连接故障。

  Note

  如果您想尝试其中的多个入门教程或重复同一教程，则应在开始另一个教程之前删除在前一个教程中创建的事物对象。如果您没有从前面的教程中删除事物对象，则需要在后续教程中使用不同的事物名称。这是因为事物名称在您的账户和Amazon Web Services 区域中必须是唯一的。

有关 Amazon IoT Core 的更多信息，请参阅什么是 Amazon IoT Core (p. 1)？

设置您的Amazon Web Services 账户

首次使用 Amazon IoT Core 前，请完成以下任务：

主题

- 注册一个 Amazon Web Services 账户 (p. 16)
- 保护 IAM 用户 (p. 17)
- 打开 Amazon IoT 控制台 (p. 17)

注册一个 Amazon Web Services 账户

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

注册 Amazon Web Services 账户

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

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

   当您注册 Amazon Web Services 账户 时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅 IAM 用户指南中的为 为 IAM 用户（控制台）启用虚拟 MFA 设备。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅 IAM 用户指南中的以下主题：

- 在您的 Amazon Web Services 账户 中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- IAM 基于身份的策略示例

- 打开 Amazon IoT 控制台 (p. 17)

如果您自己已经有 Amazon Web Services 账户和用户，则可以使用它们并跳到 the section called “打开 Amazon IoT 控制台” (p. 17)。

打开 Amazon IoT 控制台

本部分中的大多数面向控制台的主题都是从 Amazon IoT 控制台开始的。如果您尚未登录Amazon Web Services 账户，请登录，然后打开 Amazon IoT 控制台并继续下一部分以继续 Amazon IoT 入门。

尝试 Amazon IoT Core 交互式教程

该交互式教程展示了基于 Amazon IoT 构建的简单 IoT 解决方案的组成部分。本教程的动画展示了 IoT 设备如何与 Amazon IoT Core 服务交互。本主题提供了 Amazon IoT Core 交互式教程的预览。

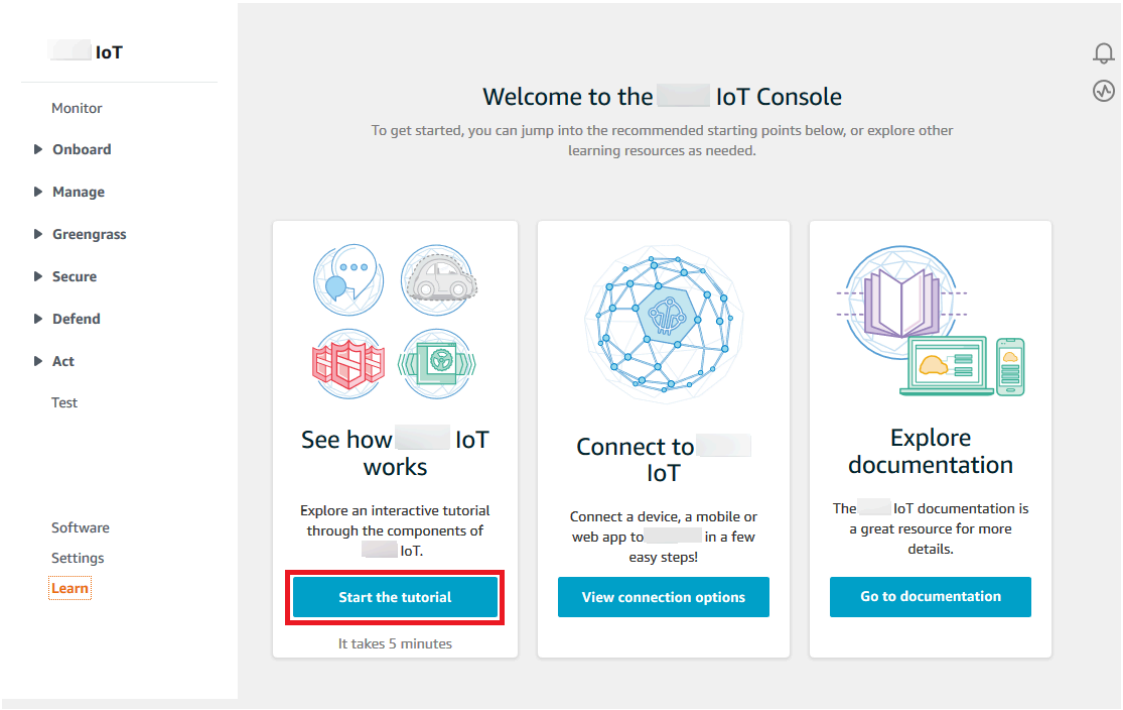
要运行演示，您必须首先the section called “设置您的Amazon Web Services 账户” (p. 16)。但是，本教程不需要任何 Amazon IoT 资源、其它软件或任何编码。

预计需要花大约 5-10 分钟来观看此演示。给自己留 10 分钟将有更多时间考虑每个步骤。

运行 Amazon IoT Core 交互式教程

1. 在 Amazon IoT 控制台中打开学习中心。

   在 Welcome to the Amazon IoT console（欢迎使用 Amazon IoT 控制台）页面的 See how Amazon IoT works（查看 Amazon IoT 工作方式）磁贴中，选择 Start the tutorial（开始教程）。



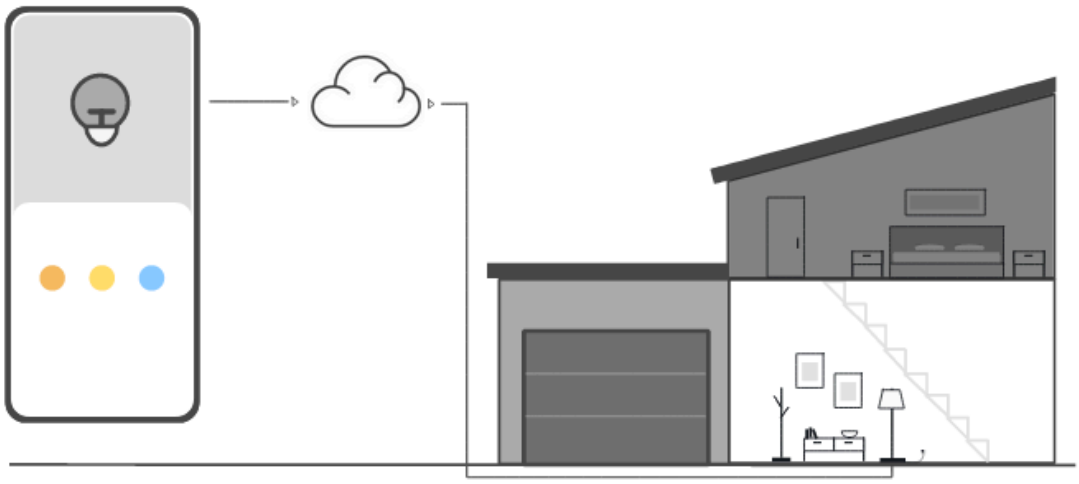
2. 在 Amazon IoT Interactive Tutorial（Amazon IoT 交互式教程）页面中，查看教程的各部分，当您准备好继续时，请选择 Start tutorial（开始教程）。

下面几节介绍 Amazon IoT Interactive Tutorial（Amazon IoT 交互式教程）如何展示这些 Amazon IoT Core 功能：

- 连接 IoT 设备 (p. 18)
- 保存离线设备状态 (p. 19)
- 将设备数据路由到服务 (p. 20)

连接 IoT 设备

了解 IoT 设备如何与 Amazon IoT Core 通信。



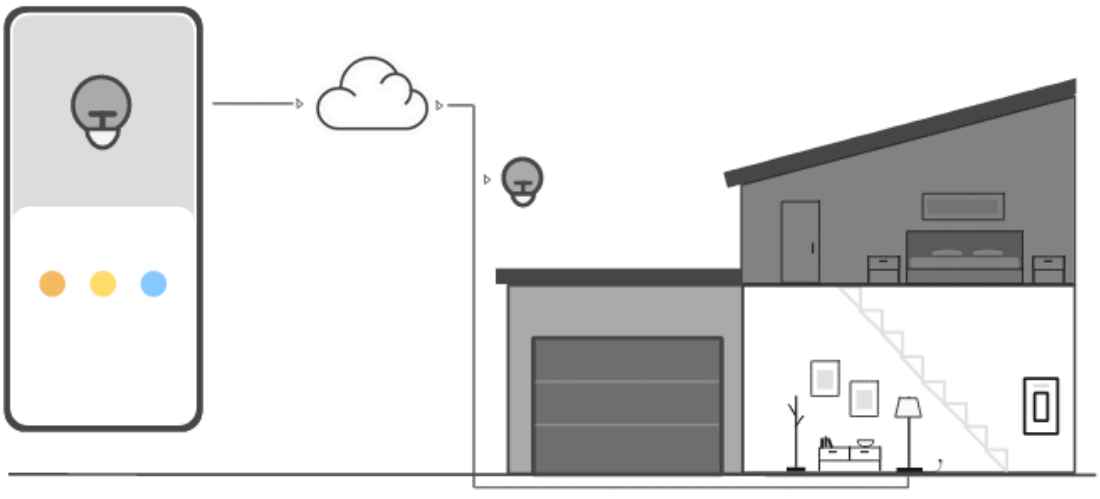
此步骤中的动画展示了两台设备（左边的控制设备和右侧房子里的智能灯）如何连接云中的 Amazon IoT Core 以及与之通信。此动画显示与 Amazon IoT Core 通信的设备并对它们收到的消息做出反应。

控制台中的图像包含未出现在此图像中的动画。

有关将设备连接到 Amazon IoT Core 的更多信息，请参阅连接到 Amazon IoT Core (p. 64)。

保存离线设备状态

了解 Amazon IoT Core 如何在设备或应用程序处于离线状态时保存设备状态。



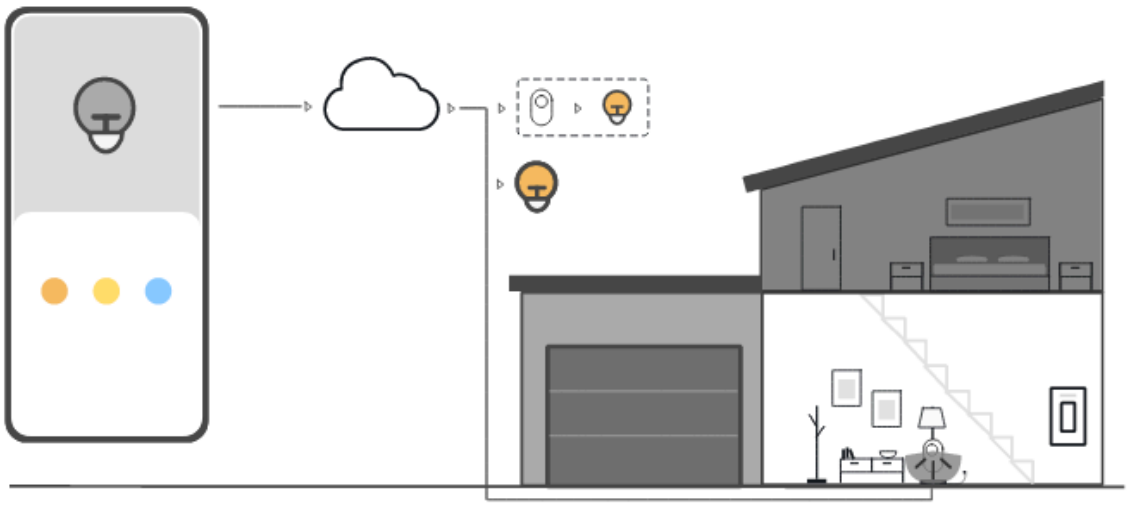
此步骤中的动画显示 Amazon IoT Core 中的设备影子服务如何为控制设备和智能灯保存设备状态信息。当智能灯处于离线状态时，设备影子将保存控制设备中的命令。

当智能灯重新连接到 Amazon IoT Core 时，它会检索这些命令。当控制设备处于离线状态时，设备影子将保存来自智能灯的状态信息。当控制设备重新连接时，它会检索智能灯的当前状态以更新其显示。

有关设备影子的更多信息，请参阅 Amazon IoT Device Shadow 服务 (p. 593)。

将设备数据路由到服务

了解 Amazon IoT Core 如何将设备状态发送给其它 Amazon 服务。



此步骤中的动画显示 Amazon IoT Core 如何使用 Amazon IoT 规则将数据从设备发送到其它 Amazon 服务。Amazon IoT 规则订阅来自设备的特定消息，解释这些消息中的数据，并将解释后的数据路由到其它服务。在此示例中，Amazon IoT 规则解释来自运动传感器的数据并将命令发送到设备影子，然后由设备影子将它们发送到智能灯泡。与前面的示例一样，设备影子存储控制设备的设备状态信息。

有关Amazon IoT规则的更多信息，请参阅Amazon IoT 规则 (p. 442)。

试试 Amazon IoT 快速连接

在本教程中，您将创建您的第一件事物对象，将设备连接到该对象，并观看其发送 MQTT 消息。

您预计需要花费 15-20 分钟来完成本教程。

本教程最适合想要快速入门 Amazon IoT 以了解其在有限场景中工作原理的人士。如果您正在寻找一个能够帮助您入门以便您能够探索更多功能和服务的示例，请尝试 通过详细教程了解 Amazon IoT Core 服务 (p. 33)。

在本教程中，您将在 Amazon IoT Core 中连接到事物资源的设备上下载并运行软件，以作为一个极小的 IoT 解决方案的一部分。该设备可以是 IoT 设备，如 Raspberry Pi，也可以是运行 Linux、OS 和 OSX 或 Windows 的电脑。如果您希望将远程广域网 (WALoRa N) 设备连接到，请参阅教程将设备和网关连接到 Amazon IoT Core适用于 LoRa WAN 的 Device WAN (p. 1106)。Amazon IoT

如果您的设备支持可以运行 Amazon IoT 控制台的浏览器，我们建议您在该设备上完成本教程。

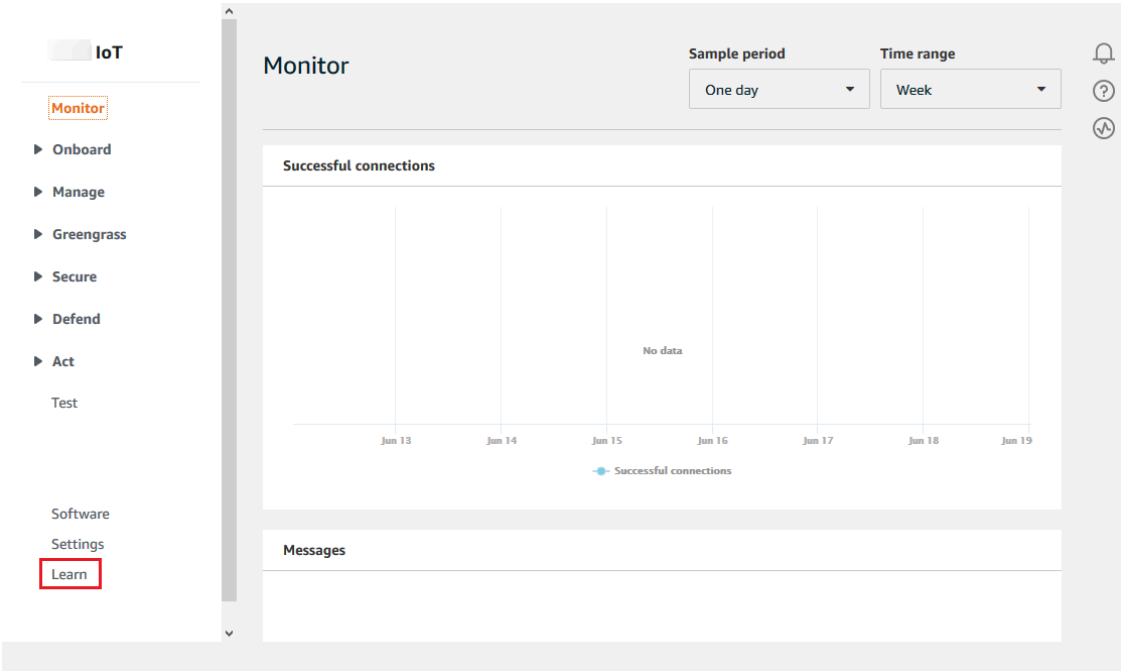
> Note
>
> 如果您的设备没有兼容的浏览器，请在电脑上遵循本教程。当流程要求您下载文件时，请将其下载到您的电脑，然后使用 Secure Copy (SCP) 或类似流程将下载的文件传输到您的设备。

本教程要求您的 IoT 设备与 Amazon Web Services 账户 的设备数据终端节点上的端口 8443 进行通信。要测试它是否可以访问该端口，请尝试测试与设备数据终端节点的连接 (p. 29)中的步骤。

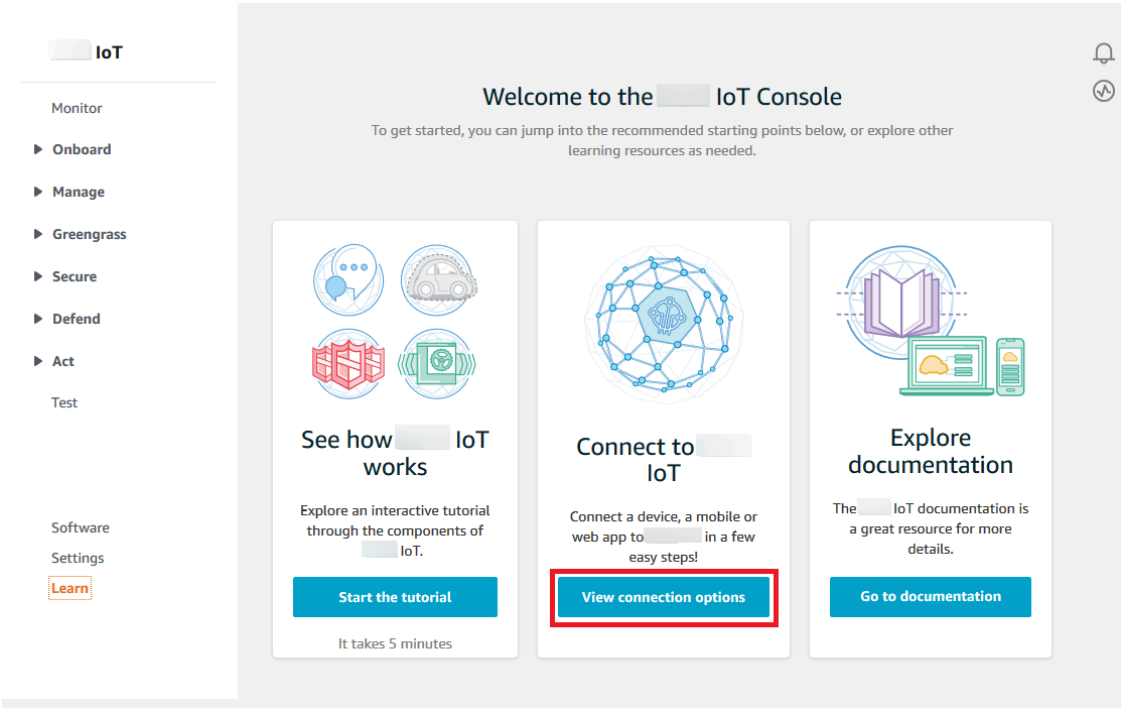
第 1 步 开始教程

如果可能，请在您的设备上完成此流程；否则，请在本流程随后的步骤中准备好将文件传输到您的设备。

1. 登录到 Amazon IoT 控制台。在 Amazon IoT 控制台主页中，从右侧的 Learning resources（学习资源）中，选择 Amazon IoT interactive tutorial（Amazon IoT 交互式教程）中的 Start tutorial（开始教程）。

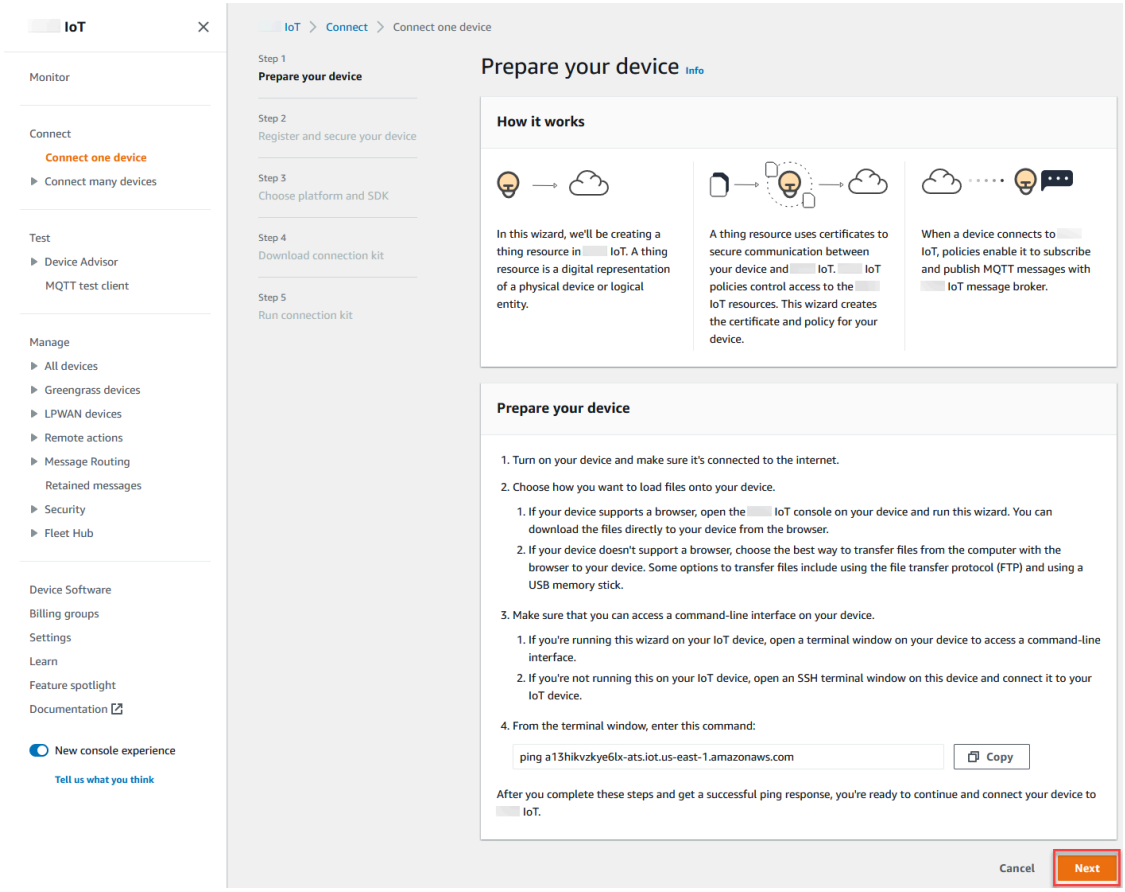


2. 在 Connect to Amazon IoT（连接到 Amazon IoT）磁贴中，选择 View connection options（查看连接选项）。



第 2 步 创建一个事物对象

1. 在 Prepare your device（准备好您的设备）部分中，按照屏幕上的说明准备好要连接到 Amazon IoT 的设备。



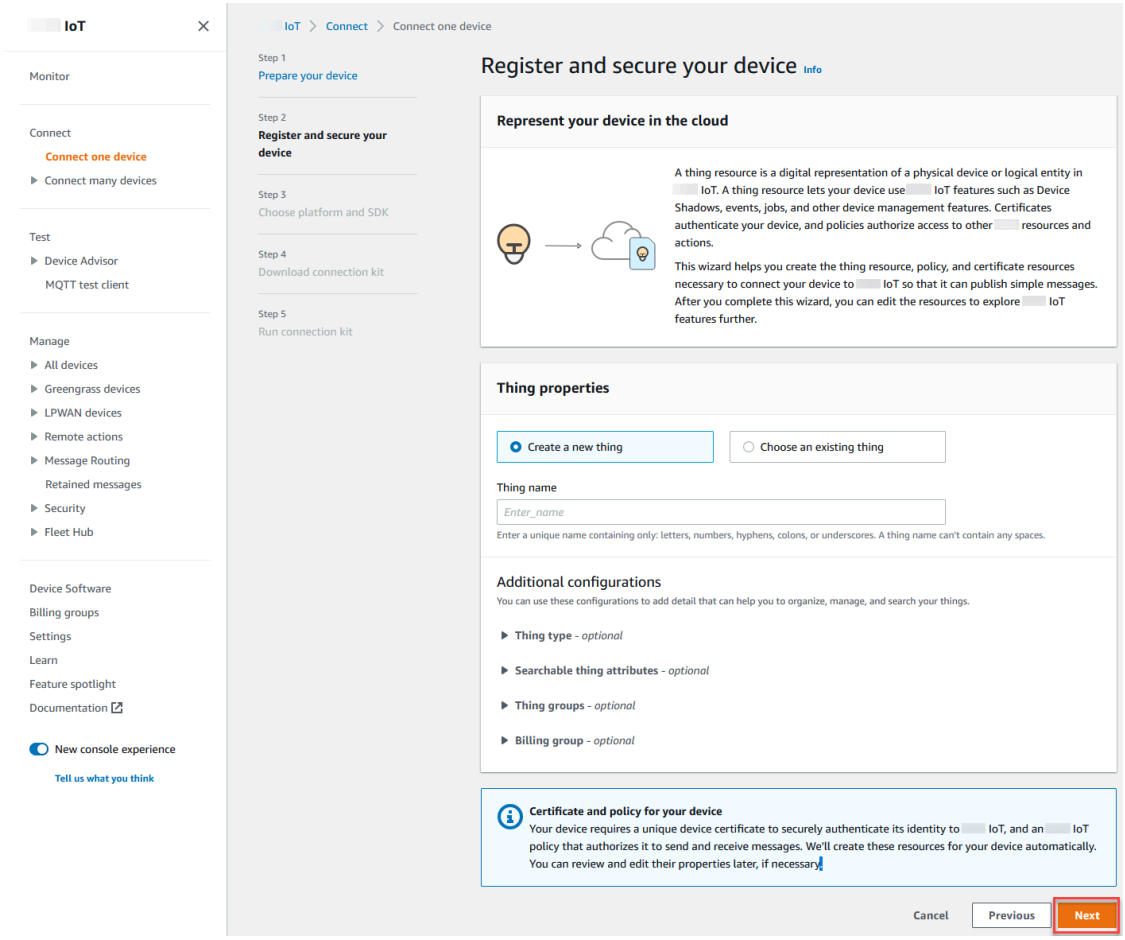
2. 在 Register and secure your device（注册并保护您的设备）部分中，选择 Create a new thing（创建新事物）或 Choose an existing thing（选择现有事物）。在 Thing name（事物名称）字段中，输入事物对象的名称。本例中使用的事物名称为 **`TutorialTestThing`**

   Important

   在继续之前，请仔细检查您的事物名称。
   事物对象创建后便无法更改事物名称。如果要更改事物名称，您必须使用正确的事物名称创建新事物对象，然后删除名称不正确的事物。

   在 Additional configurations（其他配置）部分中，使用列出的可选配置进一步自定义您的事物资源。

   为事物对象提供名称并选择任何其他配置后，选择 Next（下一步）。

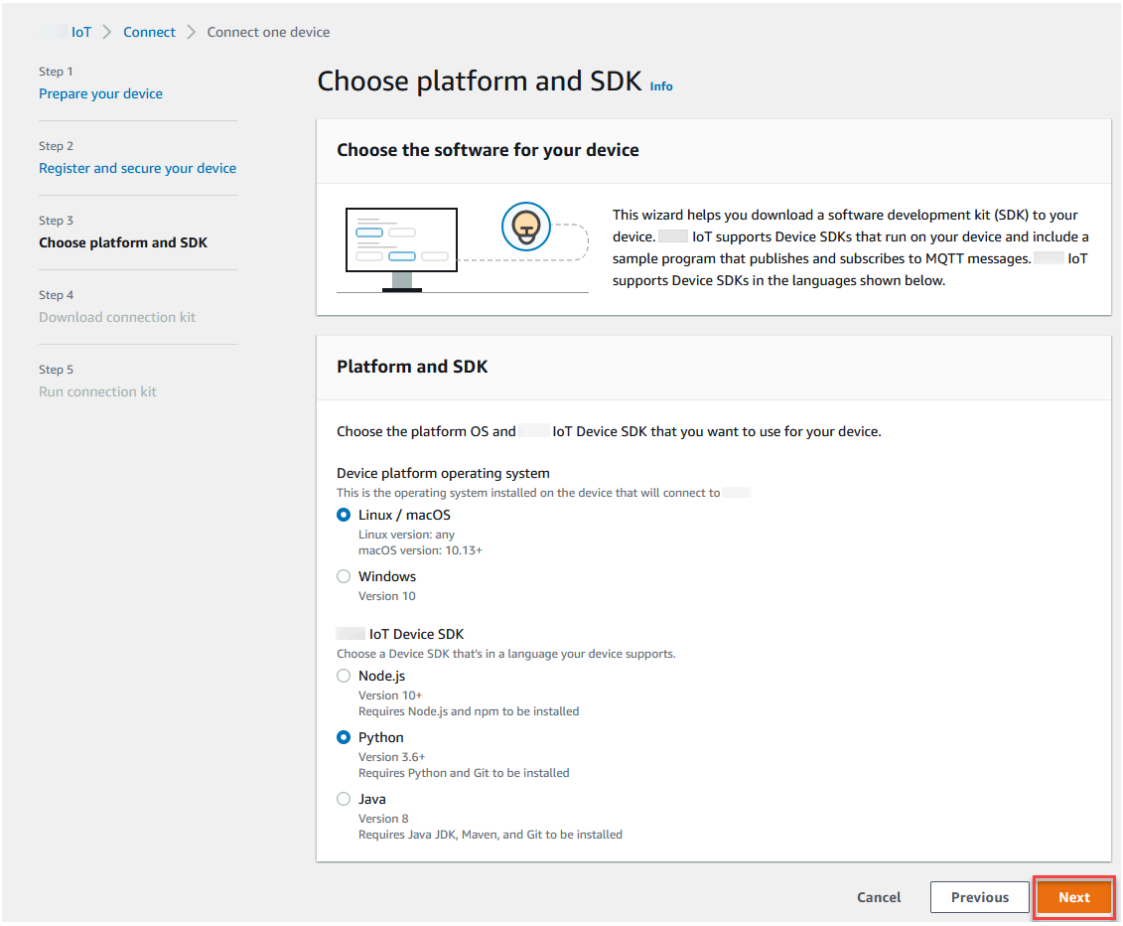


3. 在 Choose platform and SDK（选择平台和 SDK）部分中，选择您要使用的 Amazon IoT 设备 SDK 的平台和语言。本示例使用 Linux/OSX 平台和 Python SDK。请确保在目标设备上安装了 python3 和 pip3，再继续下一步。

   Note

   请务必在控制台页面底部检查所选 SDK 所需的必备软件列表。
   您必须先在目标电脑上安装所需的软件，然后才能继续执行下一步。

   选择好平台和 Device SDK 语言后，选择 Next（下一步）。

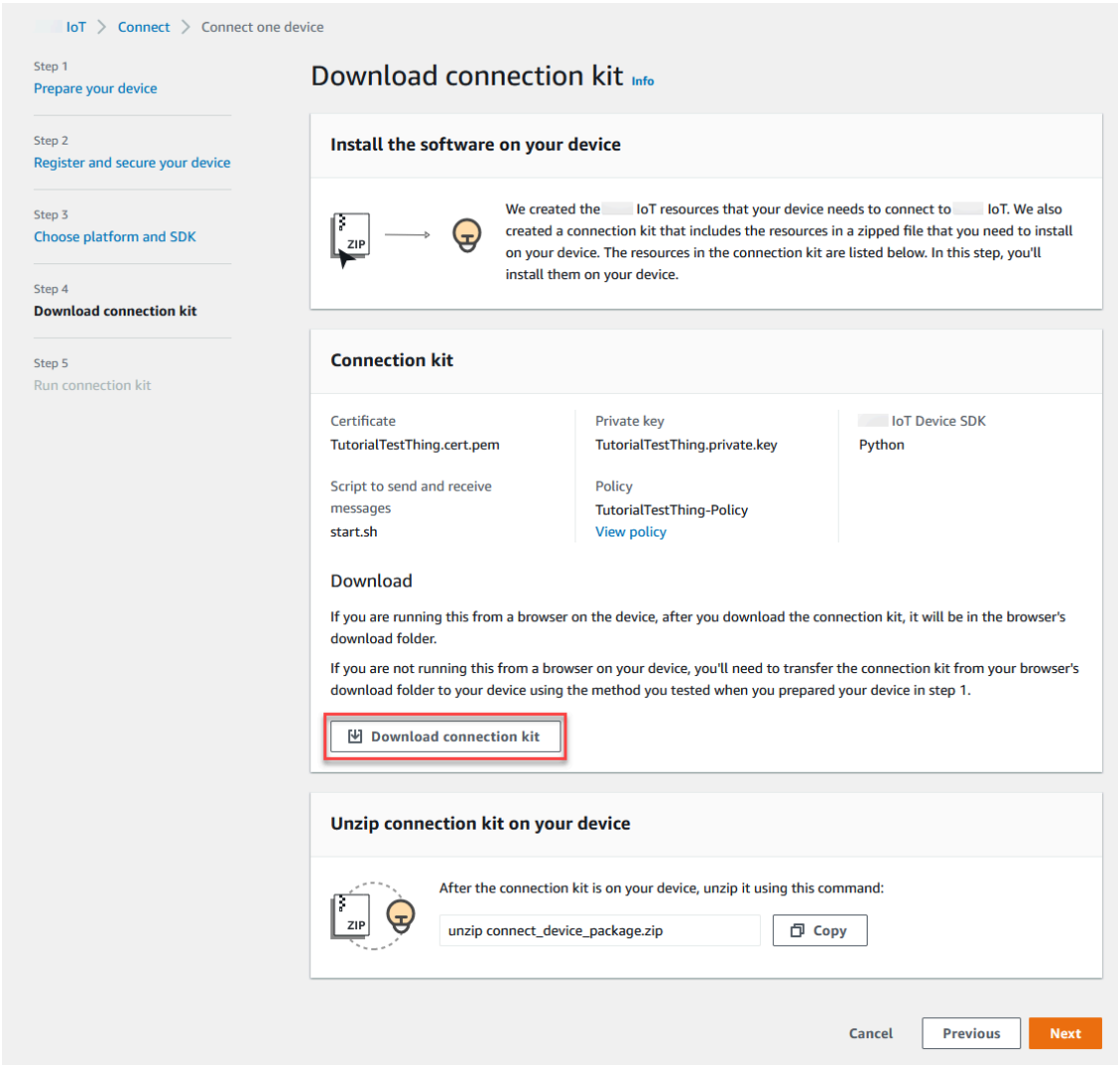


第 3 步 将文件下载到您的设备

此页面将在 Amazon IoT 创建连接工具包后显示，工具包中将包括设备所需的以下文件和资源：

- 用于对设备进行身份验证的事物证书文件
- 一个策略资源，用于授权事物对象与 Amazon IoT 交互
- 用于下载 Amazon Device SDK 并在设备上运行示例程序的脚本

1. 准备好继续后，请选择下载连接套件按钮为您之前选择的平台下载连接工具包。



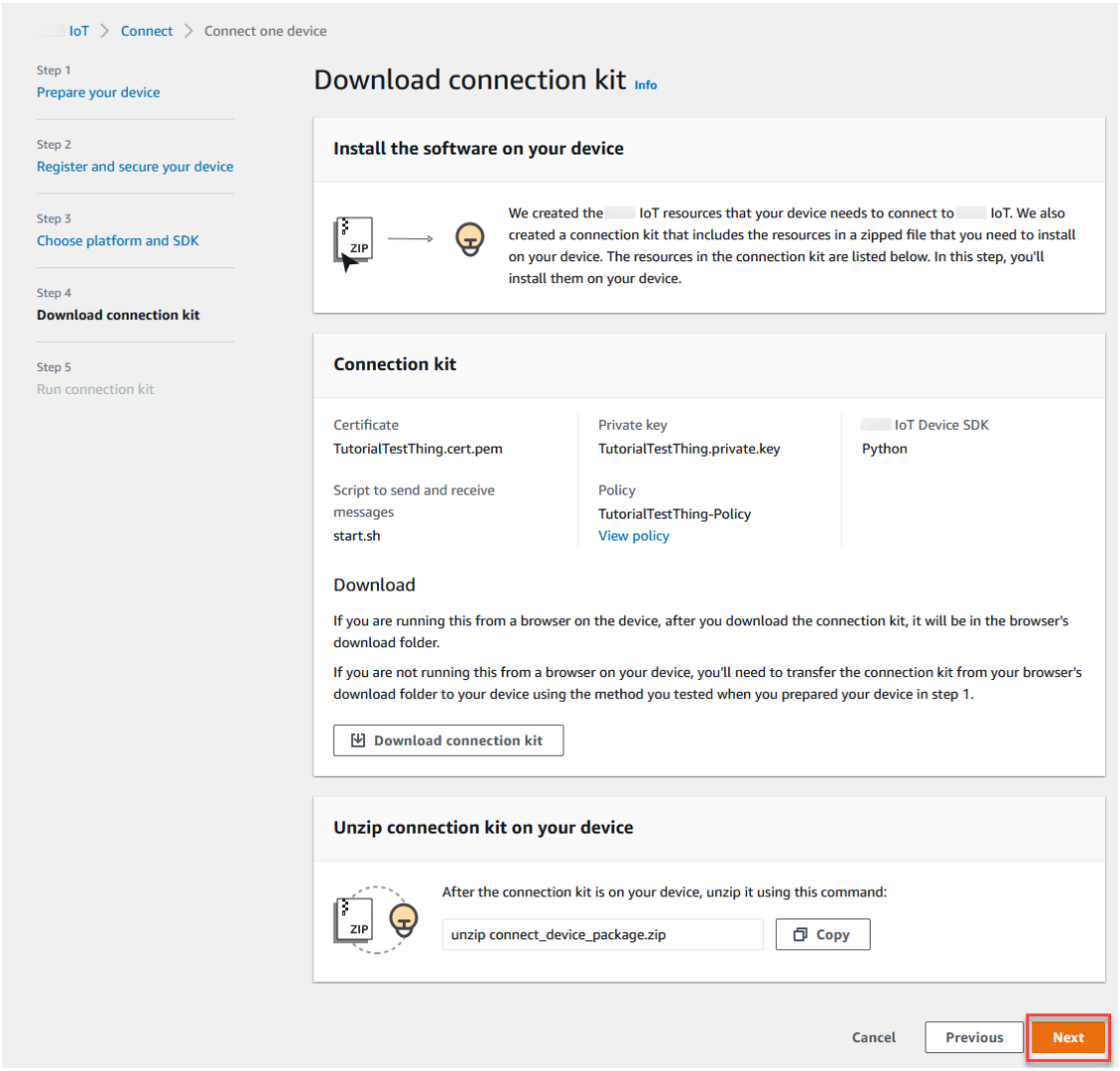
2. 如果您在设备上运行此流程，请将连接工具包文件保存到可运行命令行命令的目录中。

   如果您未在设备上运行此流程，请将连接工具包文件保存到本地目录中，然后将该文件传输到您的设备。

3. 在 Unzip connection kit on your device（在设备上解压缩连接工具包）部分中，在连接工具包文件所在的目录中输入 unzip connect_device_package.zip。

   如果您使用的是 Windows PowerShell 命令窗口，并且unzip命令不起作用，请unzip替换为expand-archive，然后再次尝试命令行。

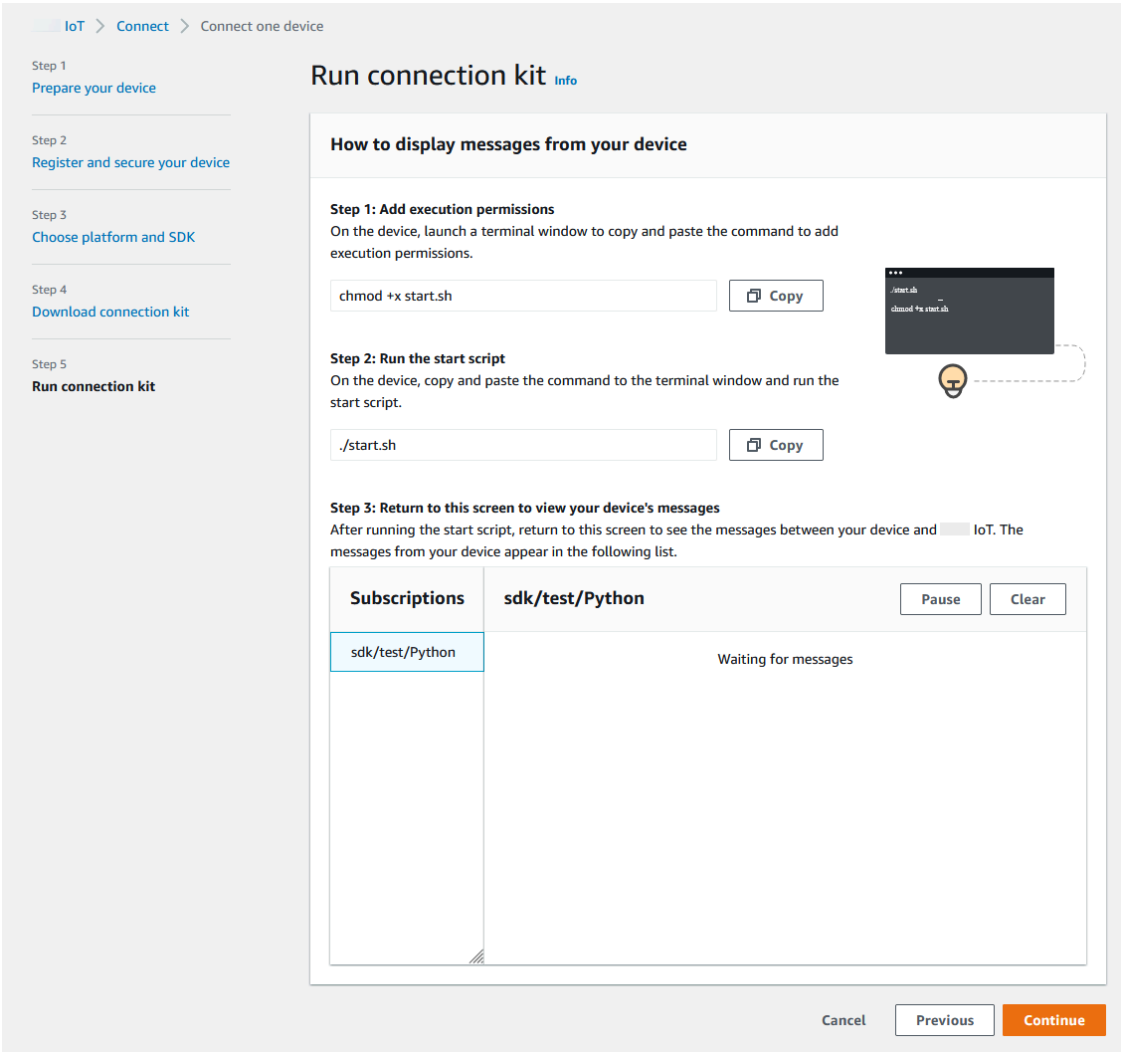
4. 当设备上具有连接工具包文件后，请选择 Next（下一步）继续本教程。



第 4 步 运行示例

您可以在设备上的终端或命令窗口中执行此程序，同时按照控制台中显示的说明进行操作。控制台中显示的命令适用于您在 the section called “第 2 步 创建一个事物对象” (p. 22) 中选择的操作系统。此处显示的内容适用于 Linux/OSX 操作系统。

1. 在设备的终端或命令窗口中，在包含连接工具包文件的目录里，执行 Amazon IoT 控制台中显示的步骤。



2. 在控制台中输入 Step 2（步骤 2）的命令后，您应该会在设备的终端或命令窗口中看到类似以下内容的输出。此输出来自程序发出然后由 Amazon IoT Core 接收的消息。

```
Running pub/sub sample application...
Connecting to a13hikvzkye6lx-ats.iot.us-east-1.amazonaws.com with client ID 'basicPubSub'...
Connected!
Subscribing to topic 'sdk/test/Python'...
Subscribed with QoS.AT_LEAST_ONCE
Sending messages until program killed
Publishing message to topic 'sdk/test/Python': Hello World! [1]
Received message from topic 'sdk/test/Python': b'"Hello World! [1]"'
Publishing message to topic 'sdk/test/Python': Hello World! [2]
Received message from topic 'sdk/test/Python': b'"Hello World! [2]"'
Publishing message to topic 'sdk/test/Python': Hello World! [3]
Received message from topic 'sdk/test/Python': b'"Hello World! [3]"'
```

当示例程序运行时，也将显示测试消息 `Hello World!`。测试消息便会显示在设备的终端或命令窗口中。

Note

有关主题订阅和发布的更多信息，请参阅所选 SDK 的示例代码。

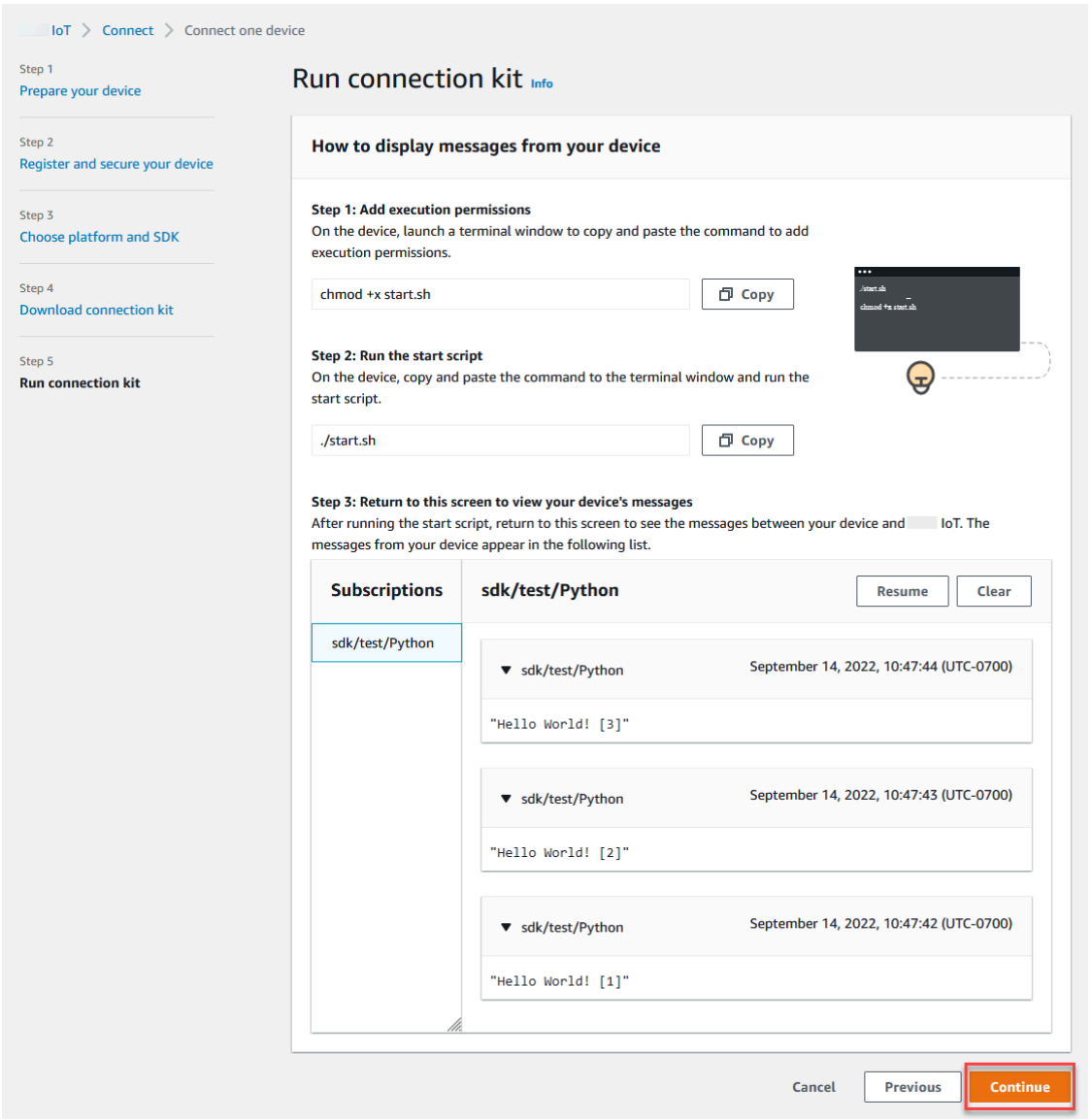
3. 要再次运行示例程序，您可以在控制台中重复本过程的 Step 2（步骤 2）中的命令。

4. （可选）如果要在 Amazon IoT 控制台 中查看来自 IoT 客户端的消息，请在 Amazon IoT 控制台的 Test（测试）页面上打开 MQTT 测试客户端。如果选择 Python SDK，请在 MQTT test client（MQTT 测试客户端）的 Topic filter（主题筛选条件）中输入主题（例如 **sdk/test/*python***），即可订阅来自设备的消息。主题筛选条件区分大小写，并且依赖于您在 Step 1（步骤 1）中选择的 SDK 的编程语言。有关主题订阅和发布的更多信息，请参阅所选 SDK 的代码示例。
5. 订阅测试主题后，请在设备上运行 ./start.sh。有关更多信息，请参阅the section called “使用 Amazon IoT MQTT 客户端查看 MQTT 消息” (p. 59)：

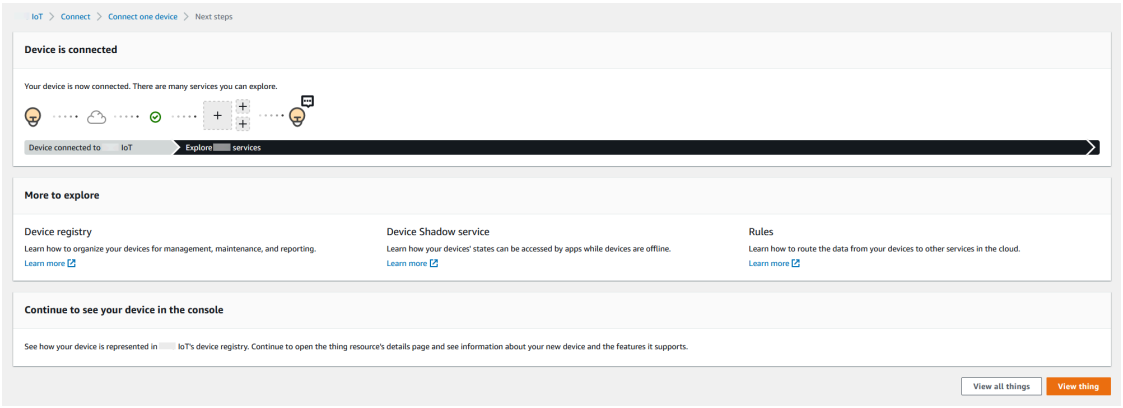
   运行 ./start.sh 后，MQTT 客户端会显示类似以下内容的消息：

```
{
  "message": "Hello World!",
  "sequence": 10
}
```

   每次接收新的 `Hello World` 消息时，`sequence` 数量会递增 1，并在您结束程序时停止。
6. 要完成本教程并查看摘要，请在 Amazon IoT 控制台中选择 Continue（继续）。



7. 现在将显示 Amazon IoT 快速连接教程的摘要。



步骤 5：进一步探索

在您完成入门教程后，可以采用以下建议进一步探索 Amazon IoT。

- 在 MQTT 测试客户端中查看 MQTT 消息

  从 Amazon IoT 控制台，您可以在 Amazon IoT 控制台的 Test（测试）页面上打开 MQTT 客户端。在 MQTT test client（MQTT 测试客户端）中，订阅 #，然后按前一步所述在设备上运行程序 ./start.sh。有关更多信息，请参阅the section called “使用 Amazon IoT MQTT 客户端查看 MQTT 消息” (p. 59)：

- 使用 Device Advisor 在您的设备上运行测试

  使用 Device Advisor 测试您的设备是否可以安全可靠地连接到 Amazon IoT 并与之交互。

- the section called “尝试 Amazon IoT Core 交互式教程” (p. 17)

  要启动交互式教程，请从 Amazon IoT 控制台的 Learn（学习）页面中开始，在 See how Amazon IoT works（了解 IoT 的工作原理）磁贴，选择 Start the tutorial（开始教程）。

- 准备好了解更多教程 (p. 33)

  这个快速入门为您提供了 Amazon IoT 的示例。如果您想进一步探索 Amazon IoT 并了解该强大 IoT 解决方案平台的功能，请从 通过详细教程了解 Amazon IoT Core 服务 (p. 33) 开始准备您的开发平台。

测试与设备数据终端节点的连接

本主题介绍如何测试设备与您账户的设备数据终端节点的连接，该终端节点是 IoT 设备用于连接 Amazon IoT 的终端节点。

在要测试的设备上执行这些步骤，或者使用连接到要测试的设备的 SSH 终端会话来执行这些步骤。

测试设备与设备数据终端节点的连接。

- 查找设备数据终端节点 (p. 30)
- 快速测试连接 (p. 30)
- 让应用程序测试与设备数据终端节点和端口的连接 (p. 31)
- 测试与设备数据终端节点和端口的连接 (p. 33)

查找设备数据终端节点

查找设备数据终端节点

1. 在 Amazon IoT 控制台中，在导航窗格底部附近，选择 Settings（设置）。
2. 在 Settings（设置）页面的 Device data endpoint（设备数据终端节点）容器中，找到 Endpoint（终端节点）值并复制它。终端节点值对于 Amazon Web Services 账户 是唯一的，并且类似于以下示例：`a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`。
3. 保存设备数据终端节点，以便在以下步骤中使用。

快速测试连接

此步骤测试与设备数据终端节点的常规连接，但不测试设备将使用的特定端口。此测试使用通用程序，通常足以了解您的设备是否可以连接到 Amazon IoT。

如果要测试与设备将使用的特定端口的连接，请跳过此步骤并继续 让应用程序测试与设备数据终端节点和端口的连接 (p. 31)。

快速测试设备数据终端节点

1. 在设备的终端或命令行窗口中，将示例设备数据终端节点 (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) 替换为您账户的设备数据终端节点，然后输入此命令。

   Linux

   ```
   ping -c 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

   Windows

   ```
   ping -n 5 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
   ```

2. 如果 `ping` 显示类似于以下内容的输出，则表示它已成功连接到设备数据终端节点。虽然它没有直接与 Amazon IoT 通信，但它确实找到了服务器，并且 Amazon IoT 很可能通过此终端节点可用。

   ```
   PING a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (xx.xx.xxx.xxx) 56(84) bytes of
    data.
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=1 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=2 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=3 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=4 ttl=231 time=127 ms
   64 bytes from ec2-EXAMPLE-218.eu-west-1.compute.amazonaws.com (xx.xx.xxx.xxx):
    icmp_seq=5 ttl=231 time=127 ms
   ```

   如果您对此结果满意，则可以在此处停止测试。

   如果想要测试与 Amazon IoT 使用的特定端口的连接，请继续 让应用程序测试与设备数据终端节点和端口的连接 (p. 31)。
3. 如果 `ping` 没有返回成功输出，请检查终端节点值以确保您拥有正确的终端节点，然后检查设备与互联网的连接。

让应用程序测试与设备数据终端节点和端口的连接

可以通过使用 nmap 来执行更彻底的连接测试。此步骤测试 nmap 是否已安装在设备上。

检查设备上的 **nmap**

1. 在要测试的设备上的终端或命令行窗口中，输入此命令以查看是否已安装 nmap。

```
nmap --version
```

2. 如果您看到类似于以下内容的输出，则表示已安装 nmap，您可以继续 the section called “测试与设备数据终端节点和端口的连接” (p. 33)。

```
Nmap version 6.40 ( http://nmap.org )
Platform: x86_64-koji-linux-gnu
Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-
libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
```

3. 如果您没有看到类似于上一步中所示的响应，则必须在设备上安装 nmap。为设备的操作系统选择相关步骤。

Linux

此步骤要求您具有在计算机上安装软件的权限。

在 Linux 计算机上安装 nmap

1. 在设备的终端或命令行窗口中，输入与所运行的 Linux 版本对应的命令。

   a. Debian 或 Ubuntu：

   ```
   sudo apt install nmap
   ```

   b. CentOS 或 RHEL：

   ```
   sudo yum install nmap
   ```

2. 使用以下命令测试安装：

   ```
   nmap --version
   ```

3. 如果您看到类似于以下内容的输出，则表示已安装 nmap，您可以继续 the section called “测试与设备数据终端节点和端口的连接” (p. 33)。

   ```
   Nmap version 6.40 ( http://nmap.org )
   Platform: x86_64-koji-linux-gnu
   Compiled with: nmap-liblua-5.2.2 openssl-1.0.2k libpcre-8.32 libpcap-1.5.3 nmap-
   libdnet-1.12 ipv6
   Compiled without:
   Available nsock engines: epoll poll select
   ```

macOS

此步骤要求您具有在计算机上安装软件的权限。

在 macOS 计算机上安装 nmap

1. 在浏览器中打开 https://nmap.org/download#macosx，然后下载最新稳定的安装程序。

   出现提示时，选择 Open with DiskImageInstaller。
2. 在安装窗口中，将程序包移到 Applications（应用程序）文件夹。
3. 在 Finder（查找工具）中，找到 Applications（应用程序）文件夹中的 nmap-xxxx-mpkg 程序包，Ctrl-click 该程序包，然后选择 Open（打开）以打开该程序包。
4. 查看安全对话框。如果您已准备好安装 nmap，请选择 Open（打开）以安装 nmap。
5. 在 Terminal 中，使用以下命令测试安装。

   ```
   nmap --version
   ```
6. 如果您看到类似于以下内容的输出，则表示已安装 nmap，您可以继续 the section called “测试与设备数据终端节点和端口的连接” (p. 33)。

   ```
   Nmap version 7.92 ( https://nmap.org )
   Platform: x86_64-apple-darwin17.7.0
   Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 libz-1.2.11
    nmap-libpcre-7.6 nmap-libpcap-1.9.1 nmap-libdnet-1.12 ipv6 Compiled without:
   Available nsock engines: kqueue poll select
   ```

Windows

此步骤要求您具有在计算机上安装软件的权限。

在 Windows 计算机上安装 nmap

1. 在浏览器中打开 https://nmap.org/download#windows，然后下载安装程序的最新稳定版本。

   出现提示时，请选择 Save file（保存文件）。下载文件后，从下载文件夹中打开它。
2. 在安装程序文件完成下载后，打开已下载的 nmap-xxxx-setup.exe 以安装应用程序。
3. 安装程序时接受默认设置。

   对于此测试，您不需要 Npcap 应用程序。如果您不想安装它，则可以取消选择该选项。
4. 在 Command 中，使用以下命令测试安装。

   ```
   nmap --version
   ```
5. 如果您看到类似于以下内容的输出，则表示已安装 nmap，您可以继续 the section called “测试与设备数据终端节点和端口的连接” (p. 33)。

   ```
   Nmap version 7.92 ( https://nmap.org )
   Platform: i686-pc-windows-windows
   Compiled with: nmap-liblua-5.3.5 openssl-1.1.1k nmap-libssh2-1.9.0 nmap-libz-1.2.11
    nmap-libpcre-7.6 Npcap-1.50 nmap-libdnet-1.12 ipv6
   Compiled without:
   Available nsock engines: iocp poll select
   ```

测试与设备数据终端节点和端口的连接

测试设备数据终端节点和端口

1. 在设备的终端或命令行窗口中，将示例设备数据终端节点 (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) 替换为您账户的设备数据终端节点，然后输入此命令。

```
nmap -p 8443 a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com
```

2. 如果 nmap 显示类似于以下内容的输出，nmap 能够通过所选端口成功连接到您的设备数据终端节点。

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-02-18 16:23 Pacific Standard Time
Nmap scan report for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (xx.xxx.147.160)
Host is up (0.036s latency).
Other addresses for a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com (not scanned):
 xx.xxx.134.144 xx.xxx.55.139 xx.xxx.110.235 xx.xxx.174.233 xx.xxx.74.65 xx.xxx.122.179
 xx.xxx.127.126
rDNS record for xx.xxx.147.160: ec2-EXAMPLE-160.eu-west-1.compute.amazonaws.com

PORT     STATE SERVICE
8443/tcp open  https-alt
MAC Address: 00:11:22:33:44:55 (Cimsys)

Nmap done: 1 IP address (1 host up) scanned in 0.91 seconds
```

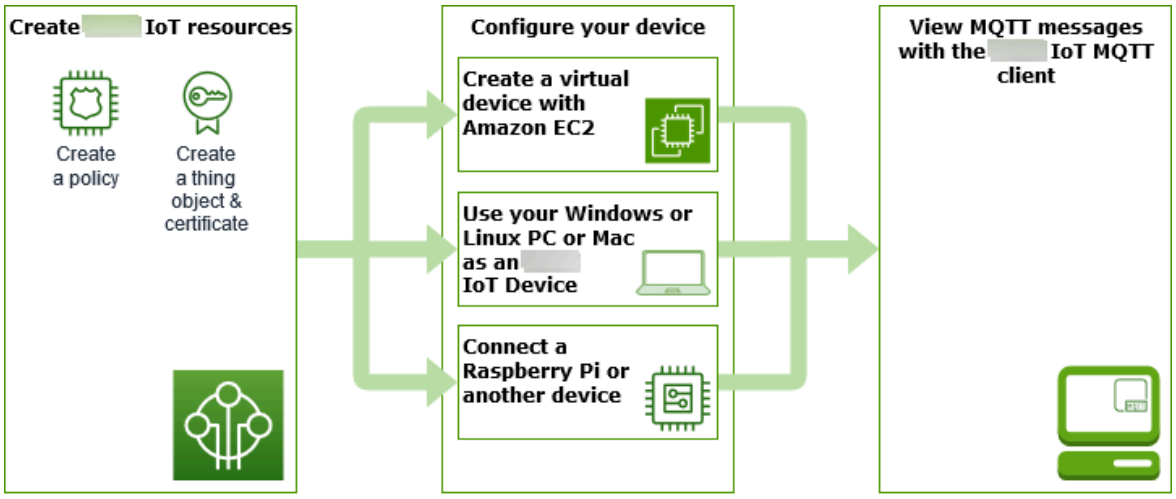
3. 如果 nmap 没有返回成功输出，请检查终端节点值以确保您拥有正确的终端节点，然后检查设备与互联网的连接。

通过将步骤 1 中使用的端口 **8443** 替换为要测试的端口，您可以测试设备数据终端节点上的其他端口，例如主 HTTPS 端口 443。

通过详细教程了解 Amazon IoT Core 服务

在本教程中，您将安装软件并创建 Amazon IoT 资源，以将设备连接到 Amazon IoT Core，使其可以使用 Amazon IoT Core 发送和接收 MQTT 消息。您将在 Amazon IoT 控制台的 MQTT 客户端中看到消息。

您预计需要花费 20-30 分钟来完成本教程。如果您使用的是 IoT 设备或 Raspberry Pi，则您可能需要花费更长时间来完成本教程，因为您可能需要安装操作系统并配置设备。



本教程最适合想要开始使用 Amazon IoT Core 以继续探索更多高级功能（例如规则引擎和影子）的开发人员。本教程将帮助您准备继续学习 Amazon IoT Core 及其如何与其他 Amazon 服务交互，相比于快速入门教程 (p. 20)，本教程将以更加详细的方式介绍各个步骤。如果您只是想要快速、类似于 Hello World 这样的简单教程，请尝试 试试 Amazon IoT 快速连接 (p. 20)。

完成 Amazon Web Services 账户 和 Amazon IoT 控制台的设置后，您将按照以下步骤查看如何连接设备并让设备将消息发送到 Amazon IoT Core。

后续步骤

- 选择最适合您的设备选项 (p. 34)
- 如果您不打算使用 Amazon EC2 创建虚拟设备，请the section called “创建 Amazon IoT 资源” (p. 34)。
- the section called “配置您的设备” (p. 37)
- the section called “使用 Amazon IoT MQTT 客户端查看 MQTT 消息” (p. 59)

有关 Amazon IoT Core 的更多信息，请参阅什么是 Amazon IoT Core (p. 1)？

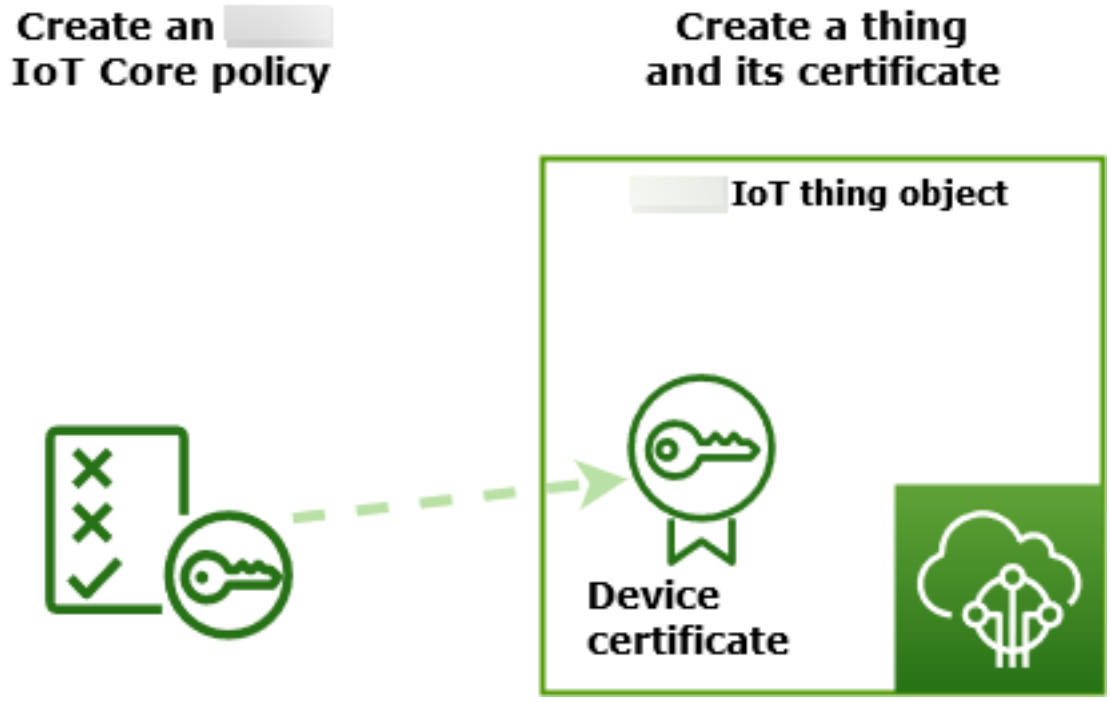
哪种设备选项最适合您？

如果您不确定要选择哪个选项，请参照下表中列出的每个选项的优缺点来帮助您确定最适合的选项。

选项	这可能是一个不错的选择，如果：	这可能不是一个好的选择，如果：
the section called “使用 Amazon EC2 创建虚拟设备” (p. 38)	• 您没有自己的设备可供测试。 • 您不想在自己的系统上安装任何软件。 • 您希望在 Linux OS 上进行测试。	• 您不习惯使用命令行命令。 • 您不想产生任何额外的Amazon费用。 • 您不想在 Linux OS 上进行测试。
the section called “将您的 Windows 或 Linux PC 或 Mac 用作 Amazon IoT 设备” (p. 45)	• 您不想产生任何额外的Amazon费用。 • 您不想配置任何其它设备。	• 您不想在个人电脑上安装任何软件。 • 您需要一个更具代表性的测试平台。
the section called “连接 Raspberry Pi 或其他设备” (p. 51)	• 您想使用实际设备测试 Amazon IoT。 • 您已经拥有要测试的设备。 • 您已有将硬件集成到系统中的经验。	• 您不想为了试用而购买或配置设备。 • 您目前只希望尽可能简单地测试 Amazon IoT。

创建 Amazon IoT 资源

在本教程中，您将创建 Amazon IoT 资源，以便设备连接到 Amazon IoT Core 并交换消息。



1. 创建 Amazon IoT 策略文档，该文档将授权您的设备与 Amazon IoT 服务交互。
2. 在 Amazon IoT 及其 X.509 设备证书中创建事物对象，然后附加策略文档。事物对象是设备在 Amazon IoT 注册表中的虚拟展示。该证书将针对 Amazon IoT Core 为您的设备进行身份验证，而策略文档会授权您的设备与 Amazon IoT 交互。

   Note

   如果您计划 the section called “使用 Amazon EC2 创建虚拟设备” (p. 38)，则可以跳过此页面并继续前往 the section called “配置您的设备” (p. 37)。当您创建虚拟事物时，将会创建这些资源。

本教程使用 Amazon IoT 控制台来创建 Amazon IoT 资源。如果您的设备支持 Web 浏览器，则在设备的 Web 浏览器上运行此流程可能会更容易，因为您可以将证书文件直接下载到您的设备上。如果您在另一台电脑上运行此流程，则需要先将证书文件复制到您的设备，然后才能为示例应用程序所用。

创建 Amazon IoT 策略

设备可以使用 X.509 证书通过 Amazon IoT Core 进行身份验证。该证书附加了 Amazon IoT 策略。这些策略决定了允许设备执行哪些 Amazon IoT 操作，例如订阅或发布 MQTT 主题。您的设备在与 Amazon IoT Core 连接并向其发送消息时会提供其证书。

按照步骤创建一个策略，以允许您的设备执行运行示例程序所需的 Amazon IoT 操作。您必须先创建 Amazon IoT 策略，然后才能将其附加到稍后创建的设备证书。

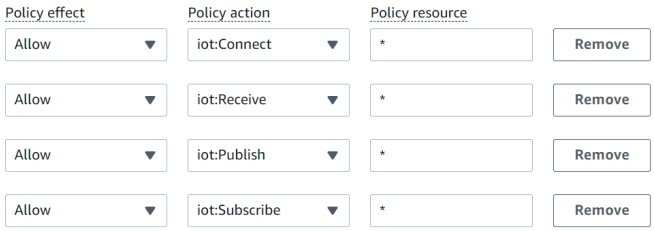
创建 Amazon IoT策略

1. 在左侧菜单中，选择 Security（安全性），然后选择 Policies（策略）。在 You don't have a policy yet（您还没有策略）页面上，选择 Create policy（创建策略）。

   如果您的账户已有现有策略，请选择 Create（创建）。
2. 在 Create policy（创建策略）页面上：

1. 在 Policy properties（策略属性）部分，在 Policy name（策略名称）字段中，输入策略的名称（例如，**My_Iot_Policy**）。请勿在策略名称中使用个人身份信息。
2. 在 Policy document（策略文档）部分，创建用于授予或拒绝资源访问 Amazon IoT Core 操作的策略语句。要创建允许所有客户端执行 **iot:Connect** 的策略语句，请按照以下步骤操作：
   - 在 Policy effect（策略效果）字段中，选择 Allow（允许）。这允许将此策略附加到其证书的所有客户端执行 Policy action（策略操作）字段中列出的操作。
   - 在 Policy action（策略操作）字段中，选择策略操作，例如 **iot:Connect**。策略操作是指设备在从设备 SDK 运行示例程序时需要权限才能执行的操作。
   - 在 Policy resource（策略资源）字段中，输入资源 Amazon Resource Name (ARN) 或 ***。用于选择任何客户端（设备）的 ***。

   要为 **iot:Receive**、**iot:Publish** 和 **iot:Subscribe** 创建策略语句，请选择 Add new statement（添加新语句）并重复相关步骤。



   Note

   在此快速入门教程中，为了简单起见，将使用通配符 (*)。为了提高安全性，您应该指定客户端 ARN 而不是通配符作为资源，从而限定哪些客户端（设备）可以连接和发布消息。客户端 ARN 应采用以下格式：`arn:aws:iot:`*`your-region`*`:`*`your-aws-account`*`:client/`*`my-client-id`*。
   但是，您必须先创建资源（如客户端设备或事物影子），然后才能将其 ARN 分配给策略。有关更多信息，请参阅 Amazon IoT Core 操作资源。

3. 在输入策略的信息后，选择 Create（创建）。

有关更多信息，请参阅Amazon IoT 如何与 IAM 协同工作 (p. 351)：

创建一个事物对象

连接到 Amazon IoT Core 的设备在 Amazon IoT 注册表中由事物对象表示。事物对象表示特定设备或逻辑实体。它可以是物理设备或传感器（例如，灯泡或墙壁上的开关）。此外，它也可以是逻辑实体（如应用程序实例），或没有连接到 Amazon IoT 但与其它连接到 IoT 的设备相关的物理实体（例如，装有发动机传感器的汽车或控制面板）。

要在 Amazon IoT 控制台中创建告警

1. 在 Amazon IoT 控制台的左侧菜单中，依次选择 Manage（管理）和 Things（事物）。
2. 在 Things（事物）页面上，选择 Create things（创建事物）。
3. 在 Create things（创建事物）页面上，选择 Create a single thing（创建单个事物），然后选择 Next（下一步）。
4. 在 Specify thing properties（指定事物属性）页面，对于 Thing name（事物名称），输入事物的名称，如 **MyIotThing**。

   仔细选择事物名称，因为以后无法更改事物名称。

   要更改事物的名称，您必须创建一个新事物，为其指定新名称，然后删除旧事物。

Note

请勿在事物名称中使用个人身份信息。事物名称可以出现在未加密的通信和报告中。

5. 保持此页面上其余字段为空。选择下一步。
6. 在 Configure device certificate - optional（配置证书 - 可选）页面上，选择 Auto-generate a new certificate (recommended)（自动生成新证书（推荐））。选择下一步。
7. 在 Attach policies to certificate - optional（将策略附加到证书 - 可选）页面，选择在上一部分中创建的策略。在该部分中，该策略被命名为 **`My_Iot_Policy`**。选择 Create thing（创建事物）。
8. 在 Download certificates and keys（下载证书和密钥）页面：

   1. 下载每个证书和密钥文件，并保存它们以备稍后使用。您需要在设备上安装这些文件。

      保存证书文件时，请在下表中提供它们的名称。这些是将后续示例中使用的文件名。

      证书文件名

文件	文件路径
私有密钥	`private.pem.key`
公有密钥	（在这些示例中未使用）
设备证书	`device.pem.crt`
根 CA 证书	`Amazon-root-CA-1.pem`

   2. 要为这些文件下载根 CA 文件，请选择根 CA 证书文件的 Download（下载）链接，该链接应与您使用的数据端点和加密套件类型相对应。在本教程中，请选择 RSA 2048 bit key: Amazon Root CA 1（RSA 2048 位密钥：Amazon Root CA 1）右侧的 Download（下载），然后下载 RSA 2048 bit key: Amazon Root CA 1（RSA 2048 位密钥：Amazon Root CA 1）证书文件。

      Important

      您必须在离开此页之前保存证书文件。在控制台中离开此页面后，您将无法再访问证书文件。
      如果忘记下载在此步骤中创建的证书文件，则必须退出此控制台屏幕，转到控制台中的事物列表，删除您创建的事物对象，然后从头重新开始此流程。

   3. 选择完成。

完成此流程后，您应该在事物列表中看到新的事物对象。

配置您的设备

本部分介绍如何配置设备以使其连接到 Amazon IoT Core。如果希望入门 Amazon IoT Core 但还没有设备，您可以使用 Amazon EC2 创建虚拟设备，也可以将 Windows PC 或 Mac 用作 IoT 设备。

选择最适合您的设备选项以尝试 Amazon IoT Core。当然，您可以尝试所有的内容，但一次只能尝试一项。如果您不确定哪个设备选项最适合您，请阅读有关如何选择哪个设备选项最适合 (p. 34)的内容，然后返回此页。

设备选项

- 使用 Amazon EC2 创建虚拟设备 (p. 38)
- 将您的 Windows 或 Linux PC 或 Mac 用作 Amazon IoT 设备 (p. 45)
- 连接 Raspberry Pi 或其他设备 (p. 51)

使用 Amazon EC2 创建虚拟设备

在本教程中，您将创建一个 Amazon EC2 实例，用作云中的虚拟设备。

要完成此教程，您需要一个Amazon Web Services 账户。如果您没有账户，请完成 设置您的Amazon Web Services 账户 (p. 16) 中介绍的步骤然后继续操作。

在本教程中，您将：

- 设置 Amazon EC2 实例 (p. 38)
- 安装 Git、Node.js 并配置 Amazon CLI (p. 39)
- 为您的虚拟设备创建 Amazon IoT 资源 (p. 40)
- 安装以下版本的Amazon IoT设备 SDK JavaScript (p. 43)
- 运行示例应用程序 (p. 43)
- 在 Amazon IoT 控制台中查看来自示例应用程序的消息 (p. 45)

设置 Amazon EC2 实例

以下步骤介绍了如何创建一个 Amazon EC2 实例，以作为您的虚拟设备代替物理设备。

如果这是您首次创建 Amazon EC2 实例，您可能会发现开始使用 Amazon EC2 Linux 实例中的说明更有用。

启动实例

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在左侧的控制台菜单中，展开 Instances（实例）部分并选择 Instances（实例）。在 Instances（实例）控制面板中，选择右侧的 Launch instances（启动实例）以显示基本配置列表。
3. 在 Name and tags（名称和标签）部分中，输入实例的名称并可选添加标签。
4. 在 Application and OS Images (Amazon Machine Image) [应用程序和操作系统映像（Amazon 机器映像）] 下，为您的实例选择一个 AMI 模板，例如 Amazon Linux 2 AMI (HVM)。请注意，此 AMI 标记为“Free tier eligible”(符合条件的免费套餐)。
5. 在 Instance type（实例类型）部分中，您可以选择实例的硬件配置。选择 `t2.micro` 类型 (预设情况下的选择)。请注意，此实例类型适用免费套餐。
6. 在 Key pair (login) [密钥对（登录）] 部分中，从下拉列表中选择密钥对名称，或选择 Create a new key pair（创建新的密钥对）以创建新的密钥对。创建新的密钥对时，确保您下载私钥文件并将其保存在安全的地方，因为这是您下载和保存它的唯一机会。当您启动实例时，您将需要提供密钥对的名称；当您每次连接到实例时，您将需要提供相应的私有密钥。

   Warning

   请勿选择 Proceed without a key pair（在没有密钥对的情况下继续）选项。如果您启动的实例没有密钥对，就不能连接到该实例。
7. 在 Network settings（网络设置）部分和 Configure storage（配置存储）部分中，您可以保留原定设置。准备就绪后，选择 Launch instances（启动实例）。
8. 确认页面会让您知道自己的实例已启动。选择 View Instances 以关闭确认页面并返回控制台。
9. 在实例屏幕上，您可以查看启动状态。启动实例只需很短的时间。启动实例时，其初始状态为 `pending`。实例启动后，其状态变为 `running`，并且会收到一个公有 DNS 名称。(如果 Public DNS (IPv4) 列已隐藏，请选择页面右上角的 Show/Hide Columns (齿轮状图标)，然后选择 Public DNS (IPv4)。)
10. 需要几分钟准备好实例，以便您能连接到实例。检查您的实例是否通过了状态检查；您可以在 Status Checks 列中查看此信息。

    在您的新实例通过其状态检查后，继续执行下一个流程并连接到此实例。

连接到您的实例

您可以通过从 Amazon EC2 控制台中选择实例，然后选择使用 Amazon EC2 Instance Connect 进行连接，以使用基于浏览器的客户端连接到实例。Instance Connect 处理权限并提供成功的连接。

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在左侧菜单上，选择 Instances（实例）。
3. 选择实例，然后选择连接。
4. 依次选择 Amazon EC2 Instance Connect（Amazon EC2 实例连接）和 Connect（连接）。

您现在应该有一个 Amazon EC2 Instance Connect 窗口，该窗口将登录您的新 Amazon EC2 实例。

安装 Git、Node.js 并配置 Amazon CLI

在本部分中，您将在 Linux 实例上安装 Git 和 Node.js。

要安装 Git

1. 在您的 Amazon EC2 Instance Connect 窗口中，使用以下命令更新实例。

```
sudo yum update -y
```

2. 在您的 Amazon EC2 Instance Connect 窗口中，使用以下命令安装 Git。

```
sudo yum install git -y
```

3. 要检查 Git 是否已安装以及 Git 的当前版本，请运行以下命令：

```
git --version
```

要安装 Node.js

1. 在您的 Amazon EC2 Instance Connect 窗口中，使用以下命令安装节点版本管理器 (nvm)。

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
```

由于 nvm 可以安装多个版本的 Node.js 并允许您在各个版本之间切换，我们将使用 nvm 安装 Node.js。

2. 在您的 Amazon EC2 Instance Connect 窗口中，使用此命令激活 nvm。

```
. ~/.nvm/nvm.sh
```

3. 在您的 Amazon EC2 Instance Connect 窗口中，使用以下命令借助 nvm 安装最新版本的 Node.js。

```
nvm install 16
```

> **Note**
>
> 这将安装最新 LTS 版本的 Node.js。

安装 Node.js 还会安装节点程序包管理器 (npm)，以便您根据需要安装其它模块。

4. 在您的 Amazon EC2 Instance Connect 窗口中，使用此命令测试 Node.js 已安装且运行正常。

```
node -e "console.log('Running Node.js ' + process.version)"
```

本教程需要 Node v10.0 或更高版本。有关更多信息，请参阅 Tutorial: Setting Up Node.js on an Amazon EC2 Instance（教程：设置 Amazon EC2 实例上的 Node.js）。

配置 Amazon CLI

您的 Amazon EC2 实例预加载了 Amazon CLI。但您必须完成您的 Amazon CLI 配置文件。有关如何配置 CLI 的更多信息，请参阅 配置 Amazon CLI。

1. 以下示例显示了示例值。将它们替换为您自己的值：您可以在 Amazon 控制台的 My Security Credentials（我的安全凭证）项下的账户信息中找到这些值。

   在您的 Amazon EC2 Instance Connect 窗口中，输入以下命令：

   ```
   aws configure
   ```

   然后在显示的提示处输入您账户中的值。

   ```
   Amazon Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   Amazon Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   Default region name [None]: us-west-2
   Default output format [None]: json
   ```

2. 您可以使用此命令测试您的 Amazon CLI 配置：

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   如果您的 Amazon CLI 配置正确，则该命令应该从您的 Amazon Web Services 账户 返回终端节点地址。

为您的虚拟设备创建 Amazon IoT 资源

此部分描述如何在虚拟设备上使用 Amazon CLI 直接创建事物对象及其证书文件。这是直接在设备上执行的，以避免将它们从另一台电脑复制到设备时可能出现的潜在复杂问题。在本节中，您将为虚拟设备创建以下资源：

- 用于在 Amazon IoT 中表示您的虚拟设备的事物对象。
- 用于验证虚拟设备身份的证书。
- 授权虚拟设备连接到 Amazon IoT 以及发布、接收和订阅消息的策略文档。

要在您的 Linux 实例中创建 Amazon IoT 事物对象

连接到 Amazon IoT 的设备在 Amazon IoT 注册表中由事物对象表示。事物对象表示特定设备或逻辑实体。在这种情况下，事物对象代表您的虚拟设备，即 Amazon EC2 实例。

1. 在您的 Amazon EC2 Instance Connect 窗口中，运行以下命令以创建事物对象。

   ```
   aws iot create-thing --thing-name "MyIotThing"
   ```

2. JSON 响应应该如下所示：

   ```
   {
       "thingArn": "arn:aws:iot:your-region:your-aws-account:thing/MyIotThing",
       "thingName": "MyIotThing",
   ```

```
    "thingId": "6cf922a8-d8ea-4136-f3401EXAMPLE"
}
```

要在您的 Linux 实例中创建和附加 Amazon IoT 密钥和证书

create-keys-and-certificate 命令将创建由 Amazon Root 证书颁发机构签名的客户端证书。此证书用于验证虚拟设备的身份。

1. 在您的 Amazon EC2 Instance Connect 窗口中，创建用于存储证书和密钥文件的目录。

```
mkdir ~/certs
```

2. 在您的 Amazon EC2 Instance Connect 窗口中，使用此命令下载 Amazon 证书颁发机构 (CA) 证书的副本。

```
curl -o ~/certs/Amazon-root-CA-1.pem \
    https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

3. 在您的 Amazon EC2 Instance Connect 窗口中，运行以下命令以创建您的私有密钥、公有密钥和 X.509 证书文件。此命令还用于注册并激活 Amazon IoT 证书。

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile "~/certs/device.pem.crt" \
    --public-key-outfile "~/certs/public.pem.key" \
    --private-key-outfile "~/certs/private.pem.key"
```

响应看起来与以下内容类似。保存 certificateArn，以便您可以在后续命令中使用。您需要使用它来将证书附加到您的事物上，并在稍后的步骤中将策略附加到证书上。

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificateId":
 "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
    "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
VVMxCzAJBgNVBAgEXAMPLEAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAsTC0lBTSEXAMPLE2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
BgkqhkiG9w0BCQEWEG5vb25lQGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCEXAMPLEJBgNVBAgTAldBMRAwDgYD
VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEXAMPLE25lQGFt
YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
rDHudUZEXAMPLELG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb3OhjZnzcvQAEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIlJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp378OD8uTs7fLvjx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC
 KEY-----\nMIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h
\nMMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/
gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWEXAMPLE
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa\nhJLXkX3rHU2xbxJSq7D
+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\nFQIDAQAB\n-----END PUBLIC KEY-----\n",
```

```
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omitted for security
 reasons\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

4. 在您的 Amazon EC2 Instance Connect 窗口中，使用以下命令和之前命令响应中的 *certificateArn* 将您的事物对象附加到您刚刚创建的证书上。

```
aws iot attach-thing-principal \
    --thing-name "MyIotThing" \
    --principal "certificateArn"
```

如果成功，此命令将不会显示任何输出。

要创建并附加策略

1. 在您的 Amazon EC2 Instance Connect 窗口中，复制此策略文档并粘贴到名为 **~/policy.json** 的文件中，以创建策略文件。

如果您没有惯用的 Linux 编辑器，则可以使用此命令打开 nano。

```
nano ~/policy.json
```

将 policy.json 的策略文档粘贴到其中。输入 ctrl-x 退出 nano 编辑器并保存文件。

policy.json 策略文档的内容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:Connect"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

2. 在您的 Amazon EC2 Instance Connect 窗口中，使用以下命令创建策略。

```
aws iot create-policy \
    --policy-name "MyIotThingPolicy" \
    --policy-document "file://~/policy.json"
```

输出：

```
{
    "policyName": "MyIotThingPolicy",
    "policyArn": "arn:aws:iot:your-region:your-aws-account:policy/MyIotThingPolicy",
    "policyDocument": "{
        \"Version\": \"2012-10-17\",
        \"Statement\": [
```

```
            {
                \"Effect\": \"Allow\",
                \"Action\": [
                    \"iot:Publish\",
                    \"iot:Receive\",
                    \"iot:Subscribe\",
                    \"iot:Connect\"
                ],
                \"Resource\": [
                    \"*\"
                ]
            }
        ]
    }",
    "policyVersionId": "1"
}
```

3. 在您的 Amazon EC2 Instance Connect 窗口中，使用以下命令将策略附加到虚拟设备的证书。

```
aws iot attach-policy \
    --policy-name "MyIotThingPolicy" \
    --target "certificateArn"
```

如果成功，此命令将不会显示任何输出。

安装以下版本的Amazon IoT设备 SDK JavaScript

在本部分中，您将安装 DevAmazon IoT ice SDK for，其中包含可使应用程序用于 JavaScript与通信的代码Amazon IoT和示例程序。有关更多信息，请参阅 JavaScript GitHub 存储库的Amazon IoT设备 SDK。

要在您的Amazon IoT Linux 实例 JavaScript 上安装 Device SDK

1. 在您的 Amazon EC2 Instance Connect 窗口中，使用此命令将 Deaws-iot-device-sdk-js-v2 vice SDK for JavaScript 存储库克隆到您的主目录中。Amazon IoT

```
cd ~
git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
```

2. 导航到您在上一步中创建的 aws-iot-device-sdk-js-v2 目录。

```
cd aws-iot-device-sdk-js-v2
```

3. 使用 npm 安装 SDK。

```
npm install
```

运行示例应用程序

下一部分中的命令假定您的密钥和证书文件均已存储在设备上，如下表中所示。

证书文件名

文件	文件路径
私有密钥	~/certs/private.pem.key
设备证书	~/certs/device.pem.crt

文件	文件路径
根 CA 证书	~/certs/Amazon-root-CA-1.pem

在本部分中，您将安装并运行在 Device SDK 的aws-iot-device-sdk-js-v2/samples/node目录中找到Amazon IoT的pub-sub.js示例应用程序 JavaScript。此应用程序显示设备（即您的 Amazon EC2 实例）如何使用 MQTT 库发布和订阅 MQTT 消息。pub-sub.js 示例应用程序订阅主题 topic_1，将 10 条消息发布到该主题，并在从消息代理收到这些消息时予以显示。

要安装和运行示例应用程序

1. 在您的 Amazon EC2 Instance Connect 窗口中，导航到 SDK 创建的 aws-iot-device-sdk-js-v2/samples/node/pub_sub 目录，并使用这些命令安装示例应用程序。

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
npm install
```

2. 在 Amazon EC2 实例Connect 窗口中，使用此命令*your-iot-endpoint*从Amazon IoT获取。

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

3. 在您的 Amazon EC2 Instance Connect 窗口中，*your-iot-endpoint*按照说明插入，然后运行此命令。

```
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

示例应用程序：

1. 连接到您账户的 Amazon IoT Core。
2. 订阅消息主题 topic_1，并显示它收到的有关该主题的消息。
3. 向主题 topic_1 发布 10 条消息。
4. 输出类似于以下内容：

```
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":1}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":2}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":3}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":4}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":5}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":6}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":7}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":8}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":9}
Publish received. topic:"topic_1" dup:false qos:1 retain:false
{"message":"Hello world!","sequence":10}
```

如果您在运行示例应用程序时遇到问题，请查看 the section called “排除示例应用程序的故障” (p. 58)。

您还可以将 `--verbosity debug` 参数添加到命令行，以便示例应用程序显示有关其正在执行的操作的详细消息。该信息可能会为您提供帮助以便您解决问题。

在 Amazon IoT 控制台中查看来自示例应用程序的消息

您可以在Amazon IoT控制台中使用 MQTT test client（MQTT 测试客户端），在示例应用程序的消息通过消息代理时查看它们。

要查看示例应用程序发布的 MQTT 消息

1. 审核使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)。它可以帮助您了解如何使用 Amazon IoT控制台中的 MQTT 测试客户端来查看通过消息代理时的 MQTT 消息。
2. 在Amazon IoT控制台中打开 MQTT 测试客户端。
3. 在 Subscribe to a topic（订阅主题）中，订阅主题 topic_1。
4. 在 Amazon EC2 Instance Connect（Amazon EC2 实例连接）窗口中，再次运行示例应用程序，然后在Amazon IoT控制台的 MQTT 测试客户端中查看消息。

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

有关 MQTT 和 Amazon IoT Core 如何支持此协议的更多信息，请参阅 MQTT。

将您的 Windows 或 Linux PC 或 Mac 用作 Amazon IoT 设备

在本教程中，您将配置个人电脑以用于 Amazon IoT。这些说明支持 Windows 和 Linux PC 以及 Mac。要完成此操作，您需要在电脑上安装某些软件。如果您不想在电脑上安装软件，可以尝试 使用 Amazon EC2 创建虚拟设备 (p. 38)，它将在虚拟机上安装所有软件。

在本教程中，您将：

- 设置您的个人电脑 (p. 45)
- 安装 Git、Python 和 Amazon IoT Device SDK for Python (p. 45)
- 设置策略并运行示例应用程序 (p. 48)
- 在 Amazon IoT 控制台中查看来自示例应用程序的消息 (p. 50)

设置您的个人电脑

要完成本教程，您需要一台连接至互联网的 Windows 或 Linux PC 或 Mac。

在继续下一步之前，请确保您可以在电脑上打开命令行窗口。在 Windows PC 上使用 cmd.exe。在 Linux PC 或 Mac 上，使用 Terminal。

安装 Git、Python 和 Amazon IoT Device SDK for Python

在本部分中，您将在电脑上安装 Python 和 Amazon IoT Device SDK for Python。

安装并使用最新版本的 Git 和 Python

要下载 Git 和 Python 并安装在您的电脑上

1. 检查您的电脑上是否已经安装了 Git。在命令行输入此命令。

```
git --version
```

如果命令显示 Git 版本，则表示已安装 Git，您可以继续执行下一步。

如果命令显示错误，请打开 https://git-scm.com/download 并为您的电脑安装 Git。

2. 检查您是否已安装了 Python。在命令行中输入此命令。

```
python -V
```

Note

如果此命令返回错误：`Python was not found`，这可能是因为您的操作系统将 Python v3.x 可执行文件调用为 `Python3`。在这种情况下，请将 `python` 的所有实例替换为 `python3`，并继续本教程的其余部分。

如果命令显示 Python 版本，则表示已安装 Python。此教程需要 Python v3.7 或更高版本。

3. 如果已安装 Python，您可以跳过本部分的其余步骤。如果没有，请继续。
4. 打开 https://www.python.org/downloads/ 并为您的电脑下载安装程序。
5. 如果下载的程序没有自动开始安装，请运行下载的程序以安装 Python。
6. 验证 Python 的安装。

```
python -V
```

确认该命令显示 Python 版本。如果没有显示 Python 版本，请尝试再次下载并安装 Python。

安装 Amazon IoT Device SDK for Python

要在电脑上安装 Amazon IoT Device SDK for Python

1. 安装 Amazon IoT Device SDK for Python v2 版本。

```
python3 -m pip install awsiotsdk
```

2. 将 DevAmazon IoT ice SDK for Python 存储库克隆到您的主目录的 aws-iot-device-sdk-python-v2 目录。此流程引用您作为###安装的文件的基础目录。

#目录的实际位置取决于您的操作系统。

Linux/macOS

在 macOS 和 Linux 中，#目录为 ~。

```
cd ~
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

Windows

在 Windows 中，您可以在 cmd 窗口中运行此命令来查找#目录路径。

```
echo %USERPROFILE%
cd %USERPROFILE%
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

Note

如果您使用的是 Window PowerShell s 而不是cmd.exe，那么请使用以下命令。

```
echo $home
```

有关更多信息，请参阅适用于 Python 的Amazon IoT Device SDK GitHub 存储库。

准备运行示例应用程序

要准备您的系统以运行示例应用程序

- 创建 certs 目录。进入 certs 目录，复制您在 the section called “创建 Amazon IoT 资源” (p. 34) 中创建和注册事物对象时保存的私有密钥、设备证书和根 CA 证书文件。目标目录中每个文件的文件名应与表中的文件名匹配。

  下一部分中的命令假定您的密钥和证书文件均已存储在设备上，如下表中所示。

  Linux/macOS

  运行此命令以创建您将在运行示例应用程序时使用的 certs 子目录。

  ```
  mkdir ~/certs
  ```

  进入新的子目录，将文件复制到下表所示的目标文件路径。

  证书文件名

文件	文件路径
私有密钥	~/certs/private.pem.key
设备证书	~/certs/device.pem.crt
根 CA 证书	~/certs/Amazon-root-CA-1.pem

  运行此命令以列出 certs 目录中的文件，并将其与表中列出的文件进行比较。

  ```
  ls -l ~/certs
  ```

  Windows

  运行此命令以创建您将在运行示例应用程序时使用的 certs 子目录。

  ```
  mkdir %USERPROFILE%\certs
  ```

  进入新的子目录，将文件复制到下表所示的目标文件路径。

证书文件名

文件	文件路径
私有密钥	`%USERPROFILE%\certs\private.pem.key`
设备证书	`%USERPROFILE%\certs\device.pem.crt`
根 CA 证书	`%USERPROFILE%\certs\Amazon-root-CA-1.pem`

运行此命令以列出 `certs` 目录中的文件，并将其与表中列出的文件进行比较。

```
dir %USERPROFILE%\certs
```

设置策略并运行示例应用程序

在本部分中，您将设置策略并运行在 Amazon IoT Device SDK for Python 的 `aws-iot-device-sdk-python-v2/samples` 目录中找到的 `pubsub.py` 示例脚本。此脚本显示设备将如何使用 MQTT 库发布和订阅 MQTT 消息。

`pubsub.py` 示例应用程序订阅了一个主题，`test/topic`，将 10 条消息发布到该主题，并在从消息代理收到这些消息时予以显示。

要运行 `pubsub.py` 示例应用程序，您需要具有以下信息：

应用程序参数值

参数	在何处查找值
`your-iot-endpoint`	1. 在 Amazon IoT 控制台的左侧菜单中，选择 Settings（设置）。 2. 在 Settings（设置）页面上，您的终端节点将显示在 Device data endpoint（设备数据终端节点）部分。

该*`your-iot-endpoint`*值的格式为:*`endpoint_id`*`-ats.iot.`*`region`*`.amazonaws.com`，例如`a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com`。

在运行脚本之前，请确保事物的策略为示例脚本提供了连接、订阅、发布和接收的权限。

要查找和查看事物资源的策略文档

1. 在 Amazon IoT控制台的 Things（事物）列表中，找到代表设备的事物资源。
2. 选择代表设备的事物资源名称链接来打开 Thing details（事物详细信息）页面。
3. 在Thing details（事物详细信息）页面，Certificates（证书）选项卡中，选择附加到事物资源的证书。列表中应只有一个证书。如果有多个证书，请选择文件安装在设备上并连接到 Amazon IoT Core 上的证书。

   在Certificate（证书）详细信息页面的Policies（策略）选项卡中，选择附加到该证书的策略。应该只有一个策略。如果有多个策略，请重复下一步，确保至少有一个策略授予所需的访问权限。
4. 在策略概述页面，找到 JSON 编辑器然后选择编辑策略文档，根据需要查看和编辑策略文档。

5. 下列示例中显示了 JSON 策略。在"Resource" 元件中，用您的 Amazon Web Services 区域 和每个 Resource 值中的 Amazon Web Services 账户 替换 *region:account*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account:topic/test/topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account:topicfilter/test/topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:region:account:client/test-*"
            ]
        }
    ]
}
```

Linux/macOS

在 Linux/macOS 上运行示例脚本

1. 在命令行窗口中，导航到 SDK 使用这些命令创建的 ~/aws-iot-device-sdk-python-v2/samples/node/pub_sub 目录。

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. 在您的命令行窗口中，*your-iot-endpoint*按照说明进行替换，然后运行此命令。

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file ~/certs/Amazon-root-
CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/private.pem.key
```

Windows

要在 Windows PC 上运行示例应用程序

1. 在您的命令行窗口中，导航到 SDK 创建的 %USERPROFILE%\aws-iot-device-sdk-python-v2\samples 目录，并使用这些命令安装示例应用程序。

```
cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
```

2. 在您的命令行窗口中，*your-iot-endpoint*按照说明进行替换，然后运行此命令。

```
python3 pubsub.py --endpoint your-iot-endpoint --ca_file %USERPROFILE%\certs
\Amazon-root-CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key %USERPROFILE%
\certs\private.pem.key
```

示例脚本：

1. 连接到您账户的 Amazon IoT Core。
2. 订阅消息主题 test/topic，并显示其收到的有关该主题的消息。
3. 向主题 test/topic 发布 10 条消息。
4. 输出类似于以下内容：

```
Connected!
Subscribing to topic 'test/topic'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'test/topic': Hello World! [1]
Received message from topic 'test/topic': b'"Hello World! [1]"'
Publishing message to topic 'test/topic': Hello World! [2]
Received message from topic 'test/topic': b'"Hello World! [2]"'
Publishing message to topic 'test/topic': Hello World! [3]
Received message from topic 'test/topic': b'"Hello World! [3]"'
Publishing message to topic 'test/topic': Hello World! [4]
Received message from topic 'test/topic': b'"Hello World! [4]"'
Publishing message to topic 'test/topic': Hello World! [5]
Received message from topic 'test/topic': b'"Hello World! [5]"'
Publishing message to topic 'test/topic': Hello World! [6]
Received message from topic 'test/topic': b'"Hello World! [6]"'
Publishing message to topic 'test/topic': Hello World! [7]
Received message from topic 'test/topic': b'"Hello World! [7]"'
Publishing message to topic 'test/topic': Hello World! [8]
Received message from topic 'test/topic': b'"Hello World! [8]"'
Publishing message to topic 'test/topic': Hello World! [9]
Received message from topic 'test/topic': b'"Hello World! [9]"'
Publishing message to topic 'test/topic': Hello World! [10]
Received message from topic 'test/topic': b'"Hello World! [10]"'
10 message(s) received.
Disconnecting...
Disconnected!
```

如果您在运行示例应用程序时遇到问题，请查看 the section called “排除示例应用程序的故障” (p. 58)。

您还可以将 `--verbosity Debug` 参数添加到命令行，以便示例应用程序显示有关其正在执行的操作的详细消息。该信息可能会帮助您改正此问题。

在 Amazon IoT 控制台中查看来自示例应用程序的消息

您可以在Amazon IoT控制台中使用 MQTT test client（MQTT 测试客户端），在示例应用程序的消息通过消息代理时查看它们。

要查看示例应用程序发布的 MQTT 消息

1. 审核使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)。它可以帮助您了解如何使用 Amazon IoT控制台中的 MQTT 测试客户端来查看通过消息代理时的 MQTT 消息。

2. 在Amazon IoT控制台中打开 MQTT 测试客户端。
3. 在 Subscribe to a topic（订阅主题）中，订阅主题 test/topic。
4. 在命令行窗口中，再次运行示例应用程序，并在 Amazon IoT 控制台的 MQTT 客户端中查看消息。

   Linux/macOS

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic test/topic --ca_file ~/certs/Amazon-root-CA-1.pem --cert
    ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
   ```

   Windows

   ```
   cd %USERPROFILE%\aws-iot-device-sdk-python-v2\samples
   python3 pubsub.py --topic test/topic --ca_file %USERPROFILE%\certs\Amazon-root-
   CA-1.pem --cert %USERPROFILE%\certs\device.pem.crt --key %USERPROFILE%\certs
   \private.pem.key --endpoint your-iot-endpoint
   ```

有关 MQTT 和 Amazon IoT Core 如何支持此协议的更多信息，请参阅 MQTT。

连接 Raspberry Pi 或其他设备

在本部分，我们将配置一个 Raspberry Pi 以便用于 Amazon IoT。如果您要连接另一台设备，Raspberry Pi 的说明包括可以帮助您调整这些指令以用到您设备上的参考资料。

这通常需要大约 20 分钟，但如果您需要安装许多系统软件升级，则可能需要更长的时间。

在本教程中，您将：

- 设置您的设备 (p. 51)
- 为 Amazon IoT Device SDK 安装所需的工具和库 (p. 52)
- 安装 Amazon IoT Device SDK (p. 53)
- 安装并运行示例应用程序 (p. 55)
- 在 Amazon IoT 控制台中查看来自示例应用程序的消息 (p. 58)

  Important

  将这些指令用于其它设备和操作系统可能会非常困难。您需要充分了解您的设备，以便能够解释这些说明并将它们应用到您的设备上。
  如果您为 Amazon IoT 配置设备时遇到困难，可以尝试使用其他设备选项之一作为替代方案，例如使用 Amazon EC2 创建虚拟设备 (p. 38)或将您的 Windows 或 Linux PC 或 Mac 用作 Amazon IoT 设备 (p. 45)。

设置您的设备

此步骤的目标是收集配置设备所需的信息，以便它可以启动操作系统（OS）、连接到互联网，并允许您在命令行界面与设备进行交互。

要完成本教程，您需要：

- 一个 Amazon Web Services 账户。如果您没有账户，请完成 设置您的Amazon Web Services 账户 (p. 16) 中介绍的步骤然后继续操作。
- Raspberry Pi 3 Model B 或更新型号。这可能适用于早期版本的 Raspberry Pi，但这尚未经过测试。
- Raspberry Pi OS (32 位)或更高版本。我们始终建议使用最新版本的 Raspberry Pi OS。早期版本的操作系统可能正常工作，但这尚未经过测试。

要运行此示例，您不需要使用图形用户界面 (GUI) 安装桌面；但是，如果您不熟悉 Raspberry Pi 并且您的 Raspberry Pi 硬件支持，则使用带有 GUI 的桌面可能会使操作更容易。

- 以太网或 WiFi 连接。
- 键盘、鼠标、显示器、电缆、电源和设备所需的其它硬件。

> **Important**
>
> 您的设备必须先安装、配置和运行操作系统，然后才能继续执行下一步。设备必须连接到互联网，并且您需要能够使用其命令行界面访问设备。命令行访问可以通过直接连接的键盘、鼠标和显示器，也可以使用 SSH 终端远程接口进行访问。

如果您在 Raspberry Pi 上运行具有图形用户界面 (GUI) 的操作系统，请在设备上打开终端窗口，然后在该窗口中执行以下说明。否则，如果您使用远程终端（如 PuTTY）连接到您的设备，请打开设备的远程终端并使用该终端。

为 Amazon IoT Device SDK 安装所需的工具和库

在安装 Amazon IoT Device SDK 和示例代码前，请确保您的系统处于最新状态，并具有安装 SDK 所需的工具和库。

1. 更新操作系统并安装所需的库

   在安装 Amazon IoT Device SDK 前，在设备上的终端窗口中运行这些命令以更新操作系统并安装所需的库。

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```

   ```
   sudo apt-get install cmake
   ```

   ```
   sudo apt-get install libssl-dev
   ```

2. 安装 Git

   如果您设备的操作系统没有安装 Git，则必须安装它，然后才能安装适用于的 Device SDK for 的 Device SDKAmazon IoT for Git JavaScript。

   a. 通过运行此命令测试是否已安装 Git。

      ```
      git --version
      ```

   b. 如果上一个命令返回了 Git 版本，则表示 Git 已经安装，您可以跳到步骤 3。

   c. 如果在运行 git 命令时显示了错误，则请运行此命令来安装 Git。

      ```
      sudo apt-get install git
      ```

   d. 再次运行此命令测试是否已安装 Git。

      ```
      git --version
      ```

   e. 如果已安装 Git，请跳到下一个部分。如果没有，请排除故障并更正错误，然后再继续。您需要 Git 才能安装 DeviceAmazon IoT SDK for JavaScript.

安装 Amazon IoT Device SDK

安装 Amazon IoT Device SDK。

Python

在本部分，您将在您的设备上安装 Python、其开发工具和 Amazon IoT Device SDK for Python。这些说明适用于运行最新版本 Raspberry Pi 操作系统的 Raspberry Pi。如果您有其他设备或使用其它操作系统，则可能需要为设备对以下说明的操作进行调整。

1. 安装 Python 及其开发工具

   Amazon IoT Device SDK for Python 需要您的 Raspberry Pi 上安装了 Python v3.5 或更高版本。

   在设备的终端窗口中，运行这些命令。

   1. 运行此命令以确定设备上安装的 Python 版本。

      ```
      python3 --version
      ```

      如果安装了 Python，它将显示其版本。
   2. 如果显示的版本为 Python 3.5 或更高版本，则您可以跳至步骤 2。
   3. 如果显示的版本低于 Python 3.5，您可以运行此命令来安装正确版本。

      ```
      sudo apt install python3
      ```
   4. 运行此命令以确认现在安装了正确版本的 Python。

      ```
      python3 --version
      ```
2. 针对 pip3 的测试

   在设备的终端窗口中，运行这些命令。

   1. 运行以下命令，查看是否安装了 pip3。

      ```
      pip3 --version
      ```
   2. 如果命令返回了版本号，则表示已安装 pip3，您可以跳到步骤 3。
   3. 如果之前的命令返回了错误，请运行此命令以安装 pip3。

      ```
      sudo apt install python3-pip
      ```
   4. 运行以下命令，查看是否安装了 pip3。

      ```
      pip3 --version
      ```
3. 安装最新的 Amazon IoT Device SDK for Python

   安装 Amazon IoT Device SDK for Python 并将示例应用程序下载到您的设备上。

   在您的设备上运行这些命令。

   ```
   cd ~
   python3 -m pip install awsiotsdk
   ```

```
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

JavaScript

在本部分中，您将在Amazon IoT设备 JavaScript 上安装 Node.js、npm 数据包管理器和 Device SDK for. 这些说明适用于运行 Raspberry Pi 操作系统的 Raspberry Pi。如果您有其他设备或使用其它操作系统，则可能需要为设备对以下说明的操作进行调整。

1. 安装最新版本的 Node.js

   的 DeAmazon IoT vice SDK JavaScript 要求在 Raspberry Pi 上安装 Node.js 和 npm 数据包管理器。

   a. 通过输入此命令以下载最新版本的 Node 存储库。

   ```
   cd ~
   curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
   ```

   b. 安装 Node 和 npm。

   ```
   sudo apt-get install -y nodejs
   ```

   c. 验证 Node 的安装。

   ```
   node -v
   ```

   确认该命令显示了 Node 版本。本教程需要 Node v10.0 或更高版本。如果未显示 Node 版本，请尝试再次下载 Node 存储库。

   d. 验证 npm 的安装。

   ```
   npm -v
   ```

   确认该命令显示了 npm 版本。如果未显示 npm 版本，请尝试再次安装 Node 和 npm。

   e. 重启设备。

   ```
   sudo shutdown -r 0
   ```

   设备重新启动后继续操作。

2. 安装以下版本的Amazon IoT设备 SDK JavaScript

   在Amazon IoT Raspberry Pi JavaScript 上安装 Device SDK f

   a. 将 JavaScript存储库的Amazon IoT设备 SDK 克隆到您的#目录中。`aws-iot-device-sdk-js-v2`在 Raspberry Pi 中，#目录为 `~/`，将在下列命令中用作#目录。如果您的设备使用不同的路径来作为#目录，则必须在以下命令中将 `~/` 替换为您设备的正确路径。

   这些命令会创建 `~/aws-iot-device-sdk-js-v2` 目录并将 SDK 代码复制到其中。

   ```
   cd ~
   git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
   ```

b. 更改为您在之前步骤中创建的 aws-iot-device-sdk-js-v2 目录，然后 运行 npm install 以安装开发工具包。npm install 命令将调用可能需要几分钟才能完成的 aws-crt 库构建。

```
cd ~/aws-iot-device-sdk-js-v2
npm install
```

安装并运行示例应用程序

在本部分，您将安装并运行在 Amazon IoT Device SDK 中找到的 pubsub 示例应用程序。此应用程序将显示您的设备如何使用 MQTT 库发布和订阅 MQTT 消息。示例应用程序订阅了一个主题，topic_1，将 10 条消息发布到该主题，并在从消息代理收到这些消息时予以显示。

安装证书文件

示例应用程序要求在设备上安装对设备进行身份验证的证书文件。

要为示例应用程序安装设备证书文件

1. 运行这些命令，在您的#目录中创建 certs 子目录。

```
cd ~
mkdir certs
```

2. 将您之前在 the section called “创建 Amazon IoT 资源” (p. 34) 中创建的私有密钥、设备证书和根 CA 证书复制到 ~/certs 目录中。

   如何将证书文件复制到设备取决于设备和操作系统，此处未予介绍。但是，如果您的设备支持图形用户界面 (GUI) 并且具有 Web 浏览器，则可以在设备的 Web 浏览器中执行 the section called “创建 Amazon IoT 资源” (p. 34) 中所述的流程，以将生成的文件直接下载到您的设备。

   下一节中的命令假定您的密钥和证书文件存储在设备上，如下表中所示。

   证书文件名

文件	文件路径
根 CA 证书	~/certs/Amazon-root-CA-1.pem
设备证书	~/certs/certificate.pem.crt
私有密钥	~/certs/private.pem.key

要运行示例应用程序，您需要以下信息：

应用程序参数值

参数	在何处查找值
your-iot-endpoint	在 Amazon IoT 控制台中，选择 All devices（所有设备），然后选择 Things（事物）。 选择您为设备创建的 IoT 事物，MyIotThing这是之前使用的名称，然后选择 Interact（交互）。 您的终端节点会显示在事物详细信息页面的 HTTPS 部分中。

参数	在何处查找值
	如果使用新版本 Amazon IoT 控制台，请从 Amazon IoT 菜单中选择 Settings（设置）。终端节点会显示在 Device data endpoint（设备数据终端节点）部分。

该*your-iot-endpoint*值的格式为:*endpoint_id*-ats.iot.*region*.amazonaws.com，例如a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com。

Python

要安装和运行示例应用程序

1. 导航到示例应用程序目录。

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. 在命令行窗口中，*your-iot-endpoint*按照说明进行替换，然后运行此命令。

```
python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert
 ~/certs/certificate.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. 观察示例应用程序：

   1. 为您的账户连接到 Amazon IoT 服务。
   2. 订阅消息主题 topic_1，并显示它收到的有关该主题的消息。
   3. 向主题 topic_1 发布 10 条消息。
   4. 输出类似于以下内容：

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
Subscribing to topic 'topic_1'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 10 message(s)
Publishing message to topic 'topic_1': Hello World! [1]
Received message from topic 'topic_1': b'Hello World! [1]'
Publishing message to topic 'topic_1': Hello World! [2]
Received message from topic 'topic_1': b'Hello World! [2]'
Publishing message to topic 'topic_1': Hello World! [3]
Received message from topic 'topic_1': b'Hello World! [3]'
Publishing message to topic 'topic_1': Hello World! [4]
Received message from topic 'topic_1': b'Hello World! [4]'
Publishing message to topic 'topic_1': Hello World! [5]
Received message from topic 'topic_1': b'Hello World! [5]'
Publishing message to topic 'topic_1': Hello World! [6]
Received message from topic 'topic_1': b'Hello World! [6]'
Publishing message to topic 'topic_1': Hello World! [7]
Received message from topic 'topic_1': b'Hello World! [7]'
Publishing message to topic 'topic_1': Hello World! [8]
Received message from topic 'topic_1': b'Hello World! [8]'
Publishing message to topic 'topic_1': Hello World! [9]
Received message from topic 'topic_1': b'Hello World! [9]'
Publishing message to topic 'topic_1': Hello World! [10]
Received message from topic 'topic_1': b'Hello World! [10]'
10 message(s) received.
Disconnecting...
```

```
Disconnected!
```

如果您在运行示例应用程序时遇到问题，请查看 the section called “排除示例应用程序的故障” (p. 58)。

您还可以将 `--verbosity Debug` 参数添加到命令行，以便示例应用程序显示有关其正在执行的操作的详细消息。该信息可能会为您提供帮助以便您解决问题。

JavaScript

要安装和运行示例应用程序

1. 在您的命令行窗口中，导航到 SDK 创建的 `~/aws-iot-device-sdk-js-v2/samples/node/pub_sub` 目录，并使用这些命令安装示例应用程序。`npm install` 命令将调用可能需要几分钟才能完成的 `aws-crt` 库构建。

```
cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
npm install
```

2. 在命令行窗口中，*your-iot-endpoint*按照说明进行替换，然后运行此命令。

```
node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert
 ~/certs/certificate.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

3. 观察示例应用程序：

   1. 为您的账户连接到 Amazon IoT 服务。
   2. 订阅消息主题 topic_1，并显示它收到的有关该主题的消息。
   3. 向主题 topic_1 发布 10 条消息。
   4. 输出类似于以下内容：

```
Publish received on topic topic_1
{"message":"Hello world!","sequence":1}
Publish received on topic topic_1
{"message":"Hello world!","sequence":2}
Publish received on topic topic_1
{"message":"Hello world!","sequence":3}
Publish received on topic topic_1
{"message":"Hello world!","sequence":4}
Publish received on topic topic_1
{"message":"Hello world!","sequence":5}
Publish received on topic topic_1
{"message":"Hello world!","sequence":6}
Publish received on topic topic_1
{"message":"Hello world!","sequence":7}
Publish received on topic topic_1
{"message":"Hello world!","sequence":8}
Publish received on topic topic_1
{"message":"Hello world!","sequence":9}
Publish received on topic topic_1
{"message":"Hello world!","sequence":10}
```

如果您在运行示例应用程序时遇到问题，请查看 the section called “排除示例应用程序的故障” (p. 58)。

您还可以将 `--verbosity Debug` 参数添加到命令行，以便示例应用程序显示有关其正在执行的操作的详细消息。该信息可能会为您提供帮助以便您解决问题。

在 Amazon IoT 控制台中查看来自示例应用程序的消息

您可以在Amazon IoT控制台中使用 MQTT test client（MQTT 测试客户端），在示例应用程序的消息通过消息代理时查看它们。

要查看示例应用程序发布的 MQTT 消息

1. 审核使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)。它可以帮助您了解如何使用 Amazon IoT控制台中的 MQTT 测试客户端来查看通过消息代理时的 MQTT 消息。
2. 在Amazon IoT控制台中打开 MQTT 测试客户端。
3. 订阅主题 topic_1。
4. 在命令行窗口中，再次运行示例应用程序，并在 Amazon IoT 控制台的 MQTT 客户端中查看消息。

   Python

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert
    ~/certs/certificate.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

   JavaScript

   ```
   cd ~/aws-iot-device-sdk-js-v2/samples/node/pub_sub
   node dist/index.js --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert
    ~/certs/certificate.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
   endpoint
   ```

排除示例应用程序的故障

如果您在尝试运行示例应用程序时遇到错误，请检查以下几个事项。

检查证书

如果证书未处于激活状态，Amazon IoT 将不会接受任何使用它进行授权的连接尝试。创建证书时，很容易忽略 Activate（激活）按钮。幸运的是，您可以在 Amazon IoT 控制台激活您的证书。

要检查证书的激活状态

1. 在 Amazon IoT控制台，在左侧菜单中，选择 Secure（安全），然后选择 Certificates（证书）。
2. 在证书列表中，找到您为练习创建的证书，并在 Status（状态）栏查看其状态。

   如果您不记得证书的名称，请检查 Inactive（未激活）的证书以查看其中是否包含是您正在使用的。

   在列表中选择相应证书以打开其详细信息页面。在详细信息页面中，您可以看到其 Create date（创建日期）以帮助您识别证书。
3. 要激活未激活的证书，请在证书详细信息页面上，选择 Actions（操作），然后选择 Activate（激活）。

如果找到了正确的证书且其处于激活状态，但运行示例应用程序时仍遇到了问题，请按照下一步所述检查其策略。

您还可以按照 the section called “创建一个事物对象” (p. 36) 中所述的步骤尝试创建新事物和新证书。如果您创建了一个新事物，则需要为其指定新事物名称并将新证书文件下载到您的设备上。

检查附加到证书上的策略。

Amazon IoT 中的策略授权操作。如果用于连接 Amazon IoT 的证书没有策略，或者没有允许其连接的策略，则连接将被拒绝，即使证书处于激活状态。

要检查附加到证书的策略

1. 查找上一项中所述的证书，然后打开其详细信息页面。
2. 在证书详细信息页面的左侧菜单中，选择 Policies（策略）以查看附加到证书的策略。
3. 如果证书没有附加策略，请选择 Actions（操作）菜单，然后选择 Attach policy（附加策略）。

   选择您之前在 the section called “创建 Amazon IoT 资源” (p. 34) 中创建的策略。
4. 如果附加了策略，请选择策略磁贴以打开其详细信息页面。

   在详细信息页面上，查看 Policy document（策略文档），以确保它包含与您在 the section called “创建 Amazon IoT 策略” (p. 35) 中创建的策略相同的信息。

检查命令行

请确保为您的系统使用了正确的命令行。Linux 和 macOS 系统上使用的命令通常与在 Windows 系统上使用的命令不同。

检查终端节点地址

查看您输入的命令，并将命令中的终端节点地址对照 Amazon IoT 控制台中的信息仔细检查。

检查证书文件的文件名

将您输入的命令中的文件名与 `certs` 目录中的证书文件的文件名进行对比。

某些系统可能要求将文件名放在引号中才能正常工作。

检查 SDK 安装状况

请确保您的 SDK 已正确完成安装。

如有疑问，请在设备上重新安装 SDK。在大多数情况下，您需要找到教程中标题为安装适用于 ***SDK ##***的 Amazon IoT Device SDK 并再次遵循这一流程。

如果您使用的是 Devic Amazon IoTe SDK JavaScript for，请记住在尝试运行示例应用之前安装这些应用程序。安装 SDK 不会自动安装示例应用程序。示例应用程序必须在安装 SDK 后手动安装。

使用 Amazon IoT MQTT 客户端查看 MQTT 消息

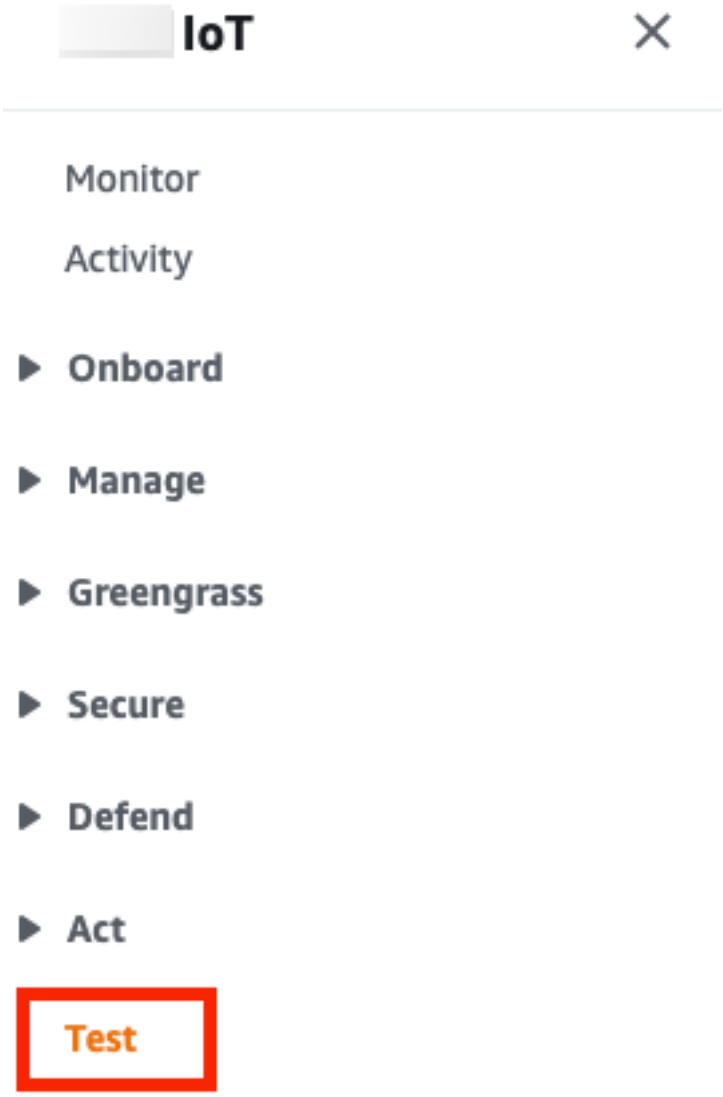
此部分描述如何在 Amazon IoT 控制台中使用 Amazon IoT MQTT 测试客户端以查看 Amazon IoT 发送和接收的 MQTT 消息。本部分中使用的示例涉及 Amazon IoT Core 入门 (p. 15) 使用的示例；但您可以将示例中使用的 *`topicName`* 替换为您的 IoT 解决方案使用的任何主题名称或主题筛选条件 (p. 94)。

设备发布有关主题 (p. 94)的 MQTT 消息以将其状态传达给 Amazon IoT，而 Amazon IoT 发布 MQTT 消息以向设备和应用程序通知更改和事件。您可以使用 MQTT 客户端订阅这些主题，并在出现消息时查看这些消息。您还可以使用 MQTT 测试客户端将 MQTT 消息发布到您 Amazon Web Services 账户中的已订阅设备和服务。

查看 MQTT 客户端中的 MQTT 消息

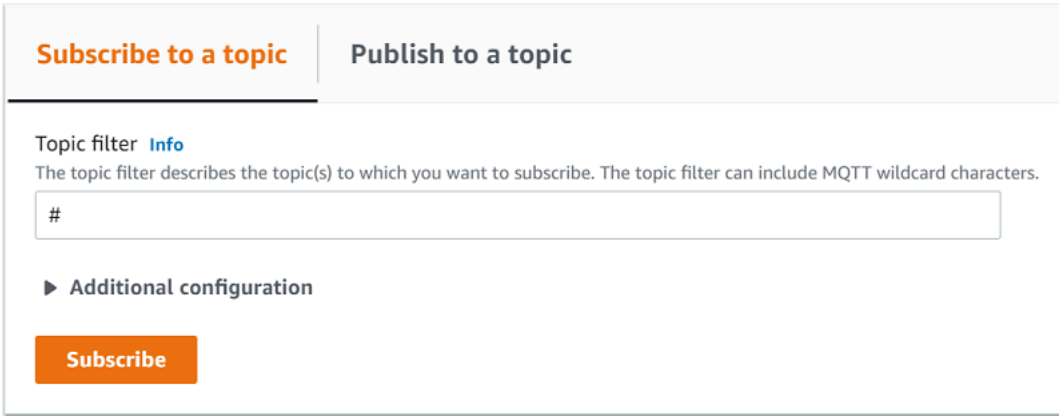
在 MQTT 测试客户端中查看 MQTT 消息

1. 打开 Amazon IoT 控制台，在左侧菜单中选择 Test (测试)，然后选择 MQTT client (MQTT 客户端)。

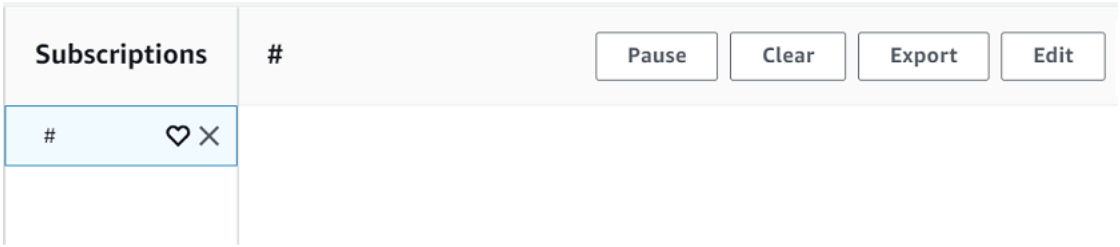


2. 在 Subscribe to a topic（订阅主题）选项卡中，输入*topicName*订阅您的设备发布的主题。对于入门示例应用程序，请订阅 **#**，它订阅了所有消息主题。

   继续使用入门示例操作，在 Subscribe to a topic（订阅主题）选项卡上的 Topic filter（主题筛选条件）字段，输入 **#**，然后选择 Subscribe（订阅）。



主题消息日志页 # 将会打开，且 # 会显示在 Subscriptions（订阅）列表。如果您在 the section called “配置您的设备” (p. 37) 中配置的设备正在运行示例程序，您应该看到它发送到 # 消息日志中的 Amazon IoT 的消息。当 Amazon IoT 收到订阅主题的消息时，消息日志条目将显示在 Publish（发布）部分下。



3. 在 # 消息日志页面，您还可以将消息发布到主题，但需要指定主题名称。您无法发布到 # 主题。

   发布到已订阅主题的消息将在收到时在消息日志中显示，最近的消息将首先显示。

排除 MQTT 消息的故障

使用通配符主题筛选条件

如果您的消息未按预期在消息日志中显示，请尝试按照 主题筛选条件 (p. 95) 中所述订阅通配符主题筛选条件。MQTT 多级通配符主题筛选条件是哈希或井号 (#)，并且可以用作 Subscription topic（订阅主题）字段中的主题筛选条件。

订阅 # 主题筛选条件将订阅消息代理接收的每个主题。您可以将主题筛选条件路径替代为 # 多级通配符或“+”单级通配符，从而缩小筛选条件的范围。

在主题筛选条件中使用通配符时

- 多级通配符必须是主题筛选条件中的最后一个字符。
- 主题筛选条件路径在每个主题级别中只能有一个单级通配符。

例如：

主题筛选条件	显示带有以下内容的消息
#	任何主题名称
topic_1/#	以 topic_1/ 开头的主题名称

主题筛选条件	显示带有以下内容的消息
`topic_1/level_2/#`	以 `topic_1/level_2/` 开头的主题名称
`topic_1/+/level_3`	以 `topic_1/` 开头并以 `/level_3` 结尾的主题名称，并且在之间有一个任意值的元素。

有关主题筛选条件的更多信息，请参阅 主题筛选条件 (p. 95)。

检查主题名称错误

MQTT 主题名称和主题筛选条件区分大小写。如果您的设备将消息发布到 `Topic_1`（带有一个大写的 T）而不是您订阅的主题 `topic_1`，则其消息将不会显示在 MQTT 测试客户端中。但是，订阅通配符主题筛选条件时，会显示设备正在发布消息，并且您会看到它使用的主题名称不是您所期望的主题名称。

从 MQTT 客户端发布 MQTT 消息

向 MQTT 主题发布消息

1. 在 MQTT 测试客户端页面上，在 Publish to a topic（发布到主题）选项卡上的 Topic name（主题名称）字段中，输入您消息的 *topicName*。在此示例中，使用 **my/topic**。

   Note

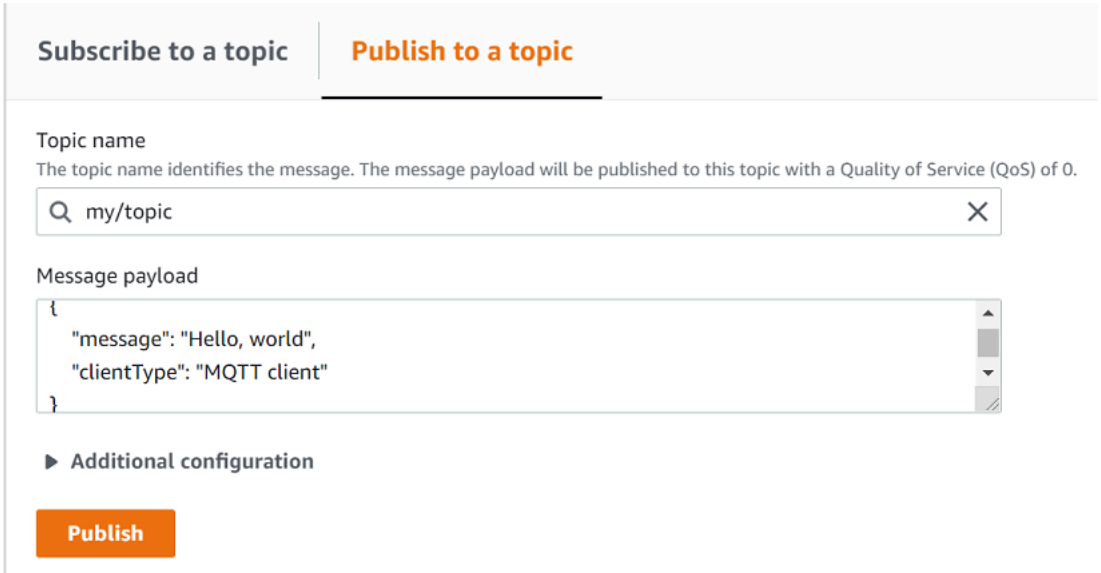
   不要在主题名称中使用个人身份信息，无论您是在测试 MQTT 客户端还是在系统实施中使用这些信息。主题名称可以出现在未加密的通信和报告中。
2. 在消息负载窗口中，输入以下 JSON：

```
{
    "message": "Hello, world",
    "clientType": "MQTT test client"
}
```

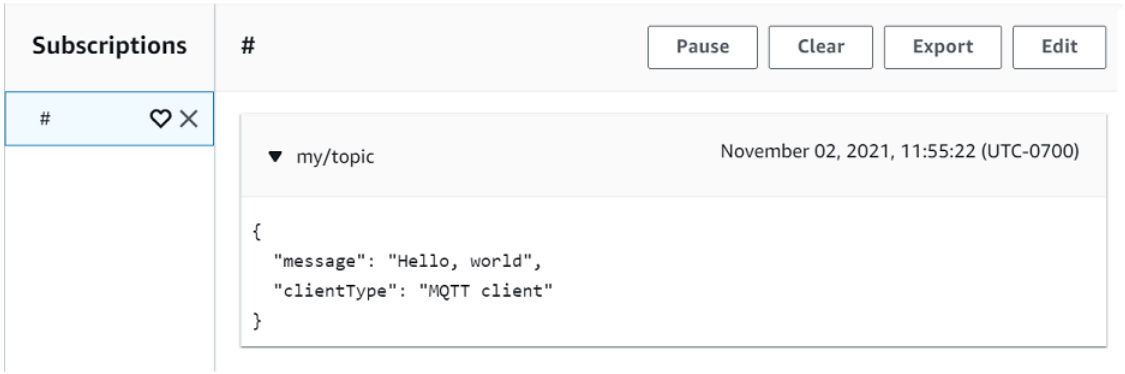
3. 选择 Publish（发布）以将消息发布到 Amazon IoT。

   Note

   确保您已订阅 my/topic 主题，然后再发布您的消息。



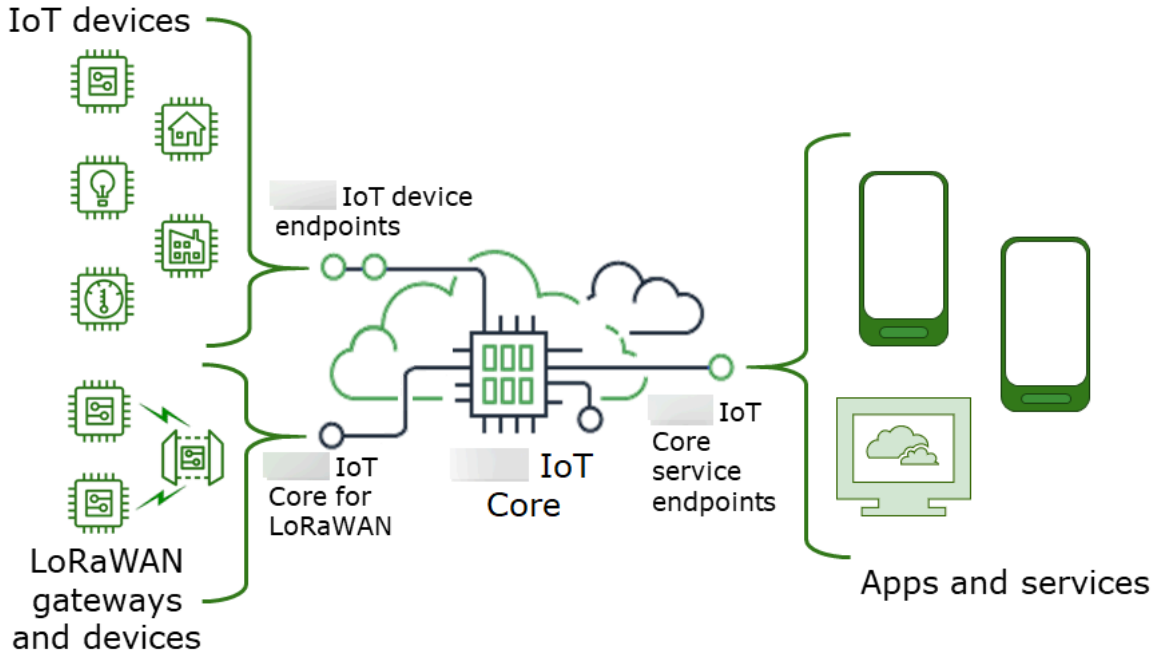
4. 在 Subscription（订阅）列中，选择 my/topic 以查看消息。您应该看到该消息显示在 MQTT 测试客户端中的发布消息有效负载窗口下。



您可以通过更改 Topic name（主题名称）字段中的 *topicName*，然后选择 Publish（发布）按钮，以将 MQTT 消息发布到其它主题。

连接到 Amazon IoT Core

Amazon IoT Core 支持与 IoT 设备、无线网关、服务和应用程序的连接。设备连接到 Amazon IoT Core，这样这些设备就可以与 Amazon IoT 服务和其他设备之间发送和接收数据。应用程序和其它服务也连接到 Amazon IoT Core 以控制和管理 IoT 设备，以及处理来自 IoT 解决方案的数据。本节介绍如何为 IoT 解决方案的各个方面选择与 Amazon IoT Core 连接和通信的最佳方式。



有多种可以和 Amazon IoT 交互的方式。应用程序和服务可以使用 Amazon IoT Core - 控制面板终端节点 (p. 64)，设备可以使用 Amazon IoT 设备终端节点 (p. 65) 或者 Amazon IoT Core用于 LoRa WAN 网关和设备 (p. 66) 连接到 Amazon IoT Core。

Amazon IoT Core - 控制面板终端节点

Amazon IoT Core - 控制面板终端节点提供对控制和管理 Amazon IoT 解决方案的功能的访问权限。

- 终端节点

  Amazon IoT Core - 控制面板和 Amazon IoT Core Device Advisor 控制面板终端节点特定于区域，并列出在 Amazon IoT Core 终端节点和配额中。终端节点的格式如下。

终端节点用途	终端节点格式	供应
Amazon IoT Core - 控制面板	`iot.aws-region.amazonaws.com`	Amazon IoT 控制面板 API
Amazon IoT Core Device Advisor - 控制面板	`api.iotdeviceadvisor.aws-region.amazonaws.com`	Amazon IoT Core Device Advisor 控制面板 API

- SDK 和工具

  Amazon SDK 提供对 Amazon IoT Core API 和其它 Amazon 服务 API 的语言特定支持。Amazon Mobile SDK 为应用程序开发人员提供对 Amazon IoT Core API 和移动设备上其它 Amazon 服务的平台特定支持。

  Amazon CLI 提供对 Amazon IoT 服务终端节点所提供的功能的命令行访问。AmazonT@@ ools for PowerShell 提供工具来管理 PowerShell 脚本环境中的Amazon服务和资源。

- 身份验证

  服务终端节点使用 IAM 用户和Amazon凭证对用户进行身份验证。

- 了解更多信息

  有关更多信息和 SDK 参考链接，请参阅 the section called “连接到 Amazon IoT Core 服务终端节点” (p. 66)。

Amazon IoT 设备终端节点

Amazon IoT 设备终端节点支持 IoT 设备与 Amazon IoT 之间的通信。

- 终端节点

  设备终端节点支持 Amazon IoT Core 和 Amazon IoT Device Management 函数。设备终端节点特定于您的Amazon Web Services 账户，您可以使用 describe-endpoint 命令查看它们。

终端节点用途	终端节点格式	供应
Amazon IoT Core - 数据层面	请参阅??? (p. 72)。	Amazon IoT 数据层面 API
Amazon IoT Device Management - 任务数据	请参阅??? (p. 72)。	Amazon IoT 任务数据层面 API
Amazon IoT Device Advisor - 数据层面	请参阅??? (p. 1019)。	
Amazon IoT Device Management - Fleet Hub		
Amazon IoT Device Management - 安全隧道	`api.tunneling.iot.aws-region.amazonaws.com`	Amazon IoT 安全隧道 API

  有关这些终端节点及其支持的功能的更多信息，请参阅 the section called “Amazon IoT 设备数据和服务终端节点” (p. 72)。

- SDK

  Amazon IoT设备软件开发工具包 (p. 73)提供对消息队列遥测传输 (MQTT) 和安全 (WSS) 协议的语言特定支持消息队列遥测传输 (MQTT) 和 WebSocket 安全 (WSS) 协议进行通信Amazon IoT。 AmazonMobile SDK (p. 71)还支持 MQTT 设备通信、Amazon IoT API 和移动设备上其他Amazon服务的 API。

- 身份验证

  设备终端节点使用 X.509 证书或具有凭证的 Amazon IAM 用户对用户进行身份验证。

- 了解更多信息

  有关更多信息和 SDK 参考链接，请参阅 the section called “Amazon IoT Device SDK” (p. 73)。

Amazon IoT Core用于 LoRa WAN 网关和设备

Amazon IoT Core适用于 LoRa WAN 的将无线网关和设备连接到Amazon IoT Core。

- 端点

  Amazon IoT Core适用于 LoRa WAN 的管理到账户和特定区域Amazon IoT Core终端节点的网关连接。网关可以连接到Amazon IoT Core适用于 LoRa WAN 的为您账户提供的 Configurate Andate Server (CUPS) 终端节点。

终端节点用途	终端节点格式	供应
Configuration and Update Server (CUPS)	*account-specific-prefix*.cups.lorawan.*aws-region*.amazonaws.com:443	网关与Amazon IoT Core适用于 LoRa WAN 的 Configurate Server 之间的网关通信
LoRaWAN 网络服务器 (LNS)	*account-specific-prefix*.gateway.lorawan.*aws-region*.amazonaws.com:443	网关与Amazon IoT Core适用于 LoRa WAN 的 WAN 网络服务器之间的网关通信, 网关与LoRa

- SDK

  AmazonSDK 支持构建 W LoRa AN 的Amazon IoT无线 API。Amazon IoT Core有关更多信息，请参阅 Amazon SDK 和工具包。

- 身份验证

  Amazon IoT Core适用于 LoRa WAN 的设备通信使用 X.509 证书来保护与之间的通信Amazon IoT。

- 了解更多信息

  有关配置和连接无线设备的更多信息，请参阅 适用于 LoRaWAN 的 Amazon IoT Core (p. 1106)。

连接到 Amazon IoT Core 服务终端节点

您可以通过使用 Amazon CLI、首选语言的 Amazon SDK 或直接调用 REST API 来访问 Amazon IoT Core - 控制面板功能。我们建议使用 Amazon CLI 或 Amazon SDK 与 Amazon IoT Core 进行交互，因为它们包含了调用 Amazon 服务的最佳实践。直接调用 REST API 是一种选择，但您必须提供必要的安全凭证，以便能够访问 API。

> Note
>
> IoT 设备应该使用 Amazon IoT Device SDK (p. 73)。Device SDK 经过优化，可在设备上使用，支持与 Amazon IoT 的 MQTT 通信，并支持设备最常使用的 Amazon IoT API。有关 Device SDK 及其提供的功能的更多信息，请参阅 Amazon IoT Device SDK (p. 73)。
> 移动设备应该使用 AmazonMobile SDK (p. 71)。Mobile SDK 支持 Amazon IoT API、MQTT 设备通信和移动设备上其它 Amazon 服务的 API。有关 Mobile SDK 及其提供的功能的更多信息，请参阅 AmazonMobile SDK (p. 71)。

您可以使用网络和移动应用程序中的 Amazon Amplify 工具和资源，以便更轻松地连接到 Amazon IoT Core。获取关于使用 Amplify 连接到 Amazon IoT Core 的更多信息，请参阅 Amplify 文档中的 Pub Sub 入门。

以下各节介绍了您可用来开发 Amazon IoT 和其它 Amazon 服务并与之交互的工具和 SDK。有关可用于在 Amazon 上构建和管理应用程序的 Amazon 工具和开发工具包的完整列表，请参阅用于在Amazon上进行构建的工具。

Amazon CLI for Amazon IoT Core

Amazon CLI 提供对 Amazon API 的命令行访问。

- 安装

  有关如何安装 Amazon CLI 的信息，请参阅安装 Amazon CLI。
- 身份验证

  Amazon CLI 使用来自您的Amazon Web Services 账户的凭证。
- 参考

  有关适用于这些 Amazon IoT Core 服务的 Amazon CLI 命令的信息，请参阅：
  - 适用于 IoT 的 Amazon CLI 命令参考
  - 适用于 IoT 数据的 Amazon CLI 命令参考
  - 适用于 IoT 作业数据的 Amazon CLI 命令参考
  - 适用于 IoT 安全隧道的 Amazon CLI 命令参考

有关用于在 PowerShell 脚本环境中管理Amazon服务和资源的工具，请参阅 T Amazonools for r Tools for r PowerShell r r

Amazon SDK

借助 Amazon SDK，您的应用程序和兼容设备可以调用 Amazon IoT API 和其它 Amazon 服务的 API。本节提供指向 Amazon SDK 和 Amazon IoT Core 服务 API 的 API 参考文档的链接。

Amazon SDK 支持这些 Amazon IoT Core API

- Amazon IoT
- Amazon IoT 数据层面
- Amazon IoT 任务数据层面
- Amazon IoT 安全隧道
- Amazon IoT 无线

C++

要安装 Amazon SDK for C++ 并使用它连接到 Amazon IoT：

1. 按照使用 Amazon C++ SDK 入门中的说明操作

   这些说明描述了如何：
   - 从源文件安装和构建 SDK
   - 提供凭证以通过您的 Amazon Web Services 账户 使用 SDK
   - 在应用程序或服务中初始化和关闭 SDK
   - 创建 CMake 项目以构建应用程序或服务
2. 创建和运行示例应用程序。有关使用 Amazon SDK for C++ 的示例应用程序，请参阅Amazon SDK for C++代码示例。

Amazon SDK for C++ 支持的 Amazon IoT Core 服务的文档

- Amazon::IoTClient" 参考文档
- Aws:: IoTDataPlane:: IoTDataPlaneClient 参考文档

- Aws:: IoTJobsDataPlane:: IoTJobsDataPlaneClient 参考文档
- Aws:: IoTSecureTunneling:: IoTSecureTunnelingClient 参考文档

Go

要安装 Amazon SDK for Go 并使用它连接到 Amazon IoT：

1. 按照 Amazon SDK for Go 入门中的说明操作

   这些说明描述了如何：
   - 安装 Amazon SDK for Go
   - 获取 SDK 的访问密钥以访问您的Amazon Web Services 账户
   - 将程序包导入到我们的应用程序或服务的源代码中
2. 创建和运行示例应用程序。如需使用 Amazon SDK for Go 的示例应用程序，请参阅 Amazon SDK for Go 代码示例。

Amazon SDK for Go 支持的 Amazon IoT Core 服务的文档

- IoT 参考文档
- IoTDataPlane 参考文档
- IoTJobsDataPlane 参考文档
- IoTSecureTunneling 参考文档

Java

要安装 Amazon SDK for Java 并使用它连接到 Amazon IoT：

1. 按照 Amazon SDK for Java 2.x 入门中的说明操作

   这些说明描述了如何：
   - 注册Amazon并创建 IAM 用户
   - 下载 SDK
   - 设置Amazon凭证和区域
   - 将 SDK 与 Apache Maven 结合使用
   - 将 SDK 与 Gradle 结合使用
2. 使用 Amazon SDK for Java 2.x 代码示例之一创建和运行示例应用程序。
3. 查看 SDK API 参考文档

Amazon SDK for Java 支持的 Amazon IoT Core 服务的文档

- IotClient 参考文档
- IotDataPlaneClient 参考文档
- IotJobsDataPlaneClient 参考文档
- IoTSecureTunnelingClient 参考文档

JavaScript

要安装 Amazon SDK for JavaScript 并使用它连接到 Amazon IoT：

1. 按照 Amazon SDK for JavaScript 设置中的说明操作。这些说明适用于在浏览器中以及通过 Node.JS 使用 Amazon SDK for JavaScript。确保按照适用于您的安装的说明进行操作。

这些说明描述了如何：

- 检查先决条件
- 手淫手淫手淫 JavaScript
- 加载以下版本的 SDK JavaScript

2. 如环境的入门选项所述，创建并运行示例应用程序以开始使用 SDK。
   - 开始使用浏览器 JavaScript 中的Amazon SDK，或者
   - 开始使用适用Amazon JavaScript 于 Node.js 的

Amazon SDK for JavaScript 支持的 Amazon IoT Core 服务的文档

- `AWS.Iot reference documentation`
- `AWS.IotData reference documentation`
- `AWS.IotJobsDataPlane reference documentation`
- `AWS.IotSecureTunneling reference documentation`

.NET

要安装 Amazon SDK for .NET 并使用它连接到 Amazon IoT：

1. 按照设置您的 Amazon SDK for .NET 环境中的说明操作
2. 按照设置您的 Amazon SDK for .NET 项目中的说明操作

这些说明描述了如何：

- 启动新项目
- 获取和配置Amazon凭证
- 安装 Amazon SDK

3. 在在 Amazon SDK for .NET 中使用Amazon服务中创建并运行一个示例程序
4. 查看 SDK API 参考文档

Amazon SDK for .NET 支持的 Amazon IoT Core 服务的文档

- Amazon.IoT.Model 参考文档
- 亚马逊。 IotData.模型参考文档
- Amazon.IoTJobsDataPlane .Model 参考文档
- Amazon.IoTSecureTunneling .Model 参考文档

PHP

要安装 Amazon SDK for PHP 并使用它连接到 Amazon IoT：

1. 按照 Amazon SDK for PHP 版本 3 入门中的说明操作

这些说明描述了如何：

- 检查先决条件
- 安装 SDK
- 将 SDK 应用于 PHP 脚本

2. 使用 Amazon SDK for PHP 版本 3 代码示例之一创建和运行示例应用程序

Amazon SDK for PHP 支持的 Amazon IoT Core 服务的文档

- IoTClient 参考文档
- IoTDataPlaneClient 参考文档
- IoTJobsDataPlaneClient 参考文档
- IoTSecureTunnelingClient 参考文档

Python

要安装 Amazon SDK for Python (Boto3) 并使用它连接到 Amazon IoT：

1. 按照 Amazon SDK for Python (Boto3) 快速入门中的说明操作

   这些说明描述了如何：

   - 安装 SDK
   - 配置 SDK
   - 在您的代码中使用 SDK
2. 创建并运行使用 Amazon SDK for Python (Boto3) 的示例程序

   此程序显示账户当前配置的日志记录选项。安装 SDK 并为您的账户配置该包后，您应该能够运行此程序。

   ```
   import boto3
   import json

   # initialize client
   iot = boto3.client('iot')

   # get current logging levels, format them as JSON, and write them to stdout
   response = iot.get_v2_logging_options()
   print(json.dumps(response, indent=4))
   ```

   有关此示例中使用的功能的更多信息，请参阅the section called “配置 Amazon IoT 日志记录” (p. 394)。

Amazon SDK for Python (Boto3) 支持的 Amazon IoT Core 服务的文档

- IoT 参考文档
- IoTDataPlane 参考文档
- IoTJobsDataPlane 参考文档
- IoTSecureTunneling 参考文档

Ruby

要安装 Amazon SDK for Ruby 并使用它连接到 Amazon IoT：

- 按照 Amazon SDK for Ruby 入门中的说明操作

  这些说明描述了如何：

  - 安装 SDK
  - 配置 SDK
- 创建并运行 Hello World 教程

Amazon SDK for Ruby 支持的 Amazon IoT Core 服务的文档

- Aws::IoT::Client 参考文档
- Aws: IoTDataPlane: Aws 参考文档
- Aws: IoTJobsDataPlane: Aws 参考文档
- Aws: IoTSecureTunneling: Aws 参考文档

AmazonMobile SDK

Amazon Mobile SDK 为移动应用程序开发人员提供特定于平台的支持，支持 Amazon IoT Core 服务的 API、使用 MQTT 的 IoT 设备通信以及其它 Amazon 服务的 API。

Android

Amazon Mobile SDK for Android

Amazon Mobile SDK for Android 包含为开发人员提供的库、示例和文档，以便于他们使用Amazon构建联网的移动应用程序。此 SDK 还包括对 MQTT 设备通信和调用 Amazon IoT Core 服务 API 的支持。有关更多信息，请参阅下列内容：

- AmazonGitHobile SDed GitHub
- Amazon Mobile SDK for Android 自述文件
- Amazon Mobile SDK for Android 示例
- Amazon SDK for Android API 参考
- AWSIoTClient 课堂参考文档

iOS

Amazon Mobile SDK for iOS

Amazon Mobile SDK for iOS 是开源的软件开发套件，依据 Apache 开源许可分发。SDK for iOS 为开发人员提供库、代码示例和文档，以便于他们使用 Amazon 构建联网的移动应用程序。此 SDK 还包括对 MQTT 设备通信和调用 Amazon IoT Core 服务 API 的支持。有关更多信息，请参阅下列内容：

- Amazon Mobile SDK for iOS上 GitHub
- Amazon SDK for iOS 自述文件
- Amazon SDK for iOS 示例
- Amazon IoTAmazon SDK for iOS 中的分类参考文档

Amazon IoT Core 服务的 REST API

可以使用 HTTP 请求直接调用 Amazon IoT Core 服务的 REST API。

- 终端节点 URL

  公开 Amazon IoT Core 服务的 REST API 的服务终端节点因区域而异，并在 Amazon IoT Core 终端节点和配额中列出。您必须对具有要访问的 Amazon IoT 资源的区域使用终端节点，因为 Amazon IoT 资源是特定于区域的。
- 身份验证

  Amazon IoT Core 服务的 REST API 使用 Amazon IAM 凭证进行身份验证。有关更多信息，请参阅 Amazon一般参考中的签署 Amazon API 请求。

- API 参考

  有关 Amazon IoT Core 服务 REST API 提供的特定功能的信息，请参阅：
  - IoT 的 API 参考。
  - IoT 数据的 API 参考。
  - IoT 任务数据的 API 参考。
  - IoT 安全隧道的 API 参考。

将设备连接到 Amazon IoT

设备可通过 Amazon IoT 连接到 Amazon IoT Core 和其它服务。通过 Amazon IoT Core，设备使用特定于您账户的设备终端节点发送和接收消息。the section called “Amazon IoT Device SDK” (p. 73) 支持使用 MQTT 和 WSS 协议进行的设备通信。有关设备可以使用的协议的更多信息，请参阅the section called “设备通信协议” (p. 75)。

消息代理

Amazon IoT 通过消息代理管理设备通信。设备和客户端向消息代理发布消息，并且还订阅消息代理发布的消息。消息由应用程序定义的主题 (p. 94) 进行标识。当消息代理收到由设备或客户端发布的消息时，它会将该消息重新发布到订阅了该消息主题的设备和客户端。消息代理还将消息转发到 Amazon IoT 规则 (p. 442)引擎，该引擎可以对消息的内容进行操作。

Amazon IoT 消息安全性

在设备连接到 Amazon IoT 时，使用 the section called “X.509 客户端证书” (p. 267)和Amazon签名 V4 进行身份验证。设备通信受 TLS 1.2 版的保护，并且 Amazon IoT 要求设备在连接时发送服务器名称指示 (SNI) 扩展。有关更多信息，请参阅 Amazon IoT 中的传输安全。

Amazon IoT 设备数据和服务终端节点

> **Important**
>
> 您可以在设备中缓存或存储端点。这意味着您无需在每次连接新设备时查询 `DescribeEndpoint` API。在 Amazon IoT Core 为您的账户创建端点后，这些端点将不会更改。

每个账户都有几个设备终端节点，这些终端节点对账户是唯一的，并支持特定的 IoT 功能。Amazon IoT 设备数据终端节点支持专为 IoT 设备的通信需求而设计的发布/订阅协议；但是，如果其它客户端（如应用程序和服务）的应用程序需要这些终端节点提供的专用功能，则它们也可以使用此接口。Amazon IoT 设备服务终端节点支持以设备为中心的对安全和管理服务的访问。

要了解您账户的设备数据终端节点，您可以在 Amazon IoT Core 控制台的 Settings（设置）页面上找到。

要了解您账户的特定用途的设备终端节点（包括设备数据终端节点），请使用此处显示的 describe-endpoint CLI 命令或 `DescribeEndpoint` REST API，并提供下表中的 *`endpointType`* 参数值。

```
aws iot describe-endpoint --endpoint-type endpointType
```

此命令采用以下格式返回 *`iot-endpoint`*：*`account-specific-prefix`*`.iot.`*`aws-region`*`.amazonaws.com`。

每个客户都有一个 `iot:Data-ATS` 和一个 `iot:Data` 终端节点。每个终端节点都使用 X.509 证书对客户端进行身份验证。我们强烈建议客户使用较新的 `iot:Data-ATS` 终端节点类型，以避免出现与 Symantec 证书颁发机构普遍不信任有关的问题。我们为设备提供`iot:Data`终端节点，用于从使用 VeriSign 证书实现向后兼容的旧终端节点检索数据的终端节点检索数据。有关更多信息，请参阅服务器身份验证。

设备的 Amazon IoT 终端节点

终端节点用途	*endpointType* 值	描述
Amazon IoT Core - 数据层面操作	`iot:Data-ATS`	用于与消息代理、Device Shadow (p. 593) 和 Amazon IoT 的规则引擎 (p. 442)组件之间收发数据。 `iot:Data-ATS` 返回 ATS 签名的数据终端节点。
Amazon IoT Core - 数据层面操作（旧版）	`iot:Data`	`iot:Data`返回为向后兼容而提供的 VeriSign 签名的数据终端节点。Symantec (`iot:Data`) 端点不支持 MQTT 5。
Amazon IoT Core 凭证访问	`iot:CredentialProvider`	用于将设备的内置 X.509 证书交换为临时凭证，以直接与其它 Amazon 服务连接。有关连接到其它 Amazon 服务的更多信息，请参阅授予直接调用 Amazon 服务的权限 (p. 340)。
Amazon IoT Device Management - 任务数据操作	`iot:Jobs`	用于使设备能够使用任务设备 HTTPS API (p. 712)与 Amazon IoT Jobs 服务交互。
Amazon IoT Device Advisor 操作	`iot:DeviceAdvisor`	用于使用 Device Advisor 测试设备的测试终端节点类型。有关更多信息，请参阅??? (p. 1016)：
Amazon IoT Core 数据测试版（预览版）	`iot:Data-Beta`	为测试版预留的终端节点类型。有关当前用法的信息，请参阅 ??? (p. 108)。

您还可以使用自己的完全限定域名 (FQDN)，例如 *example.com*和关联的服务器证书，通过使用 the section called “可配置的终端节点” (p. 108)（目前在公开测试版中提供）将设备连接到 Amazon IoT。

Amazon IoT Device SDK

Amazon IoT Device SDK 可帮助您将 IoT 设备连接到 Amazon IoT Core，并且支持 MQTT 和基于 WSS 的 MQTT 协议。

Amazon IoT Device SDK 与 Amazon SDK 的不同之处在于，Amazon IoT Device SDK 支持 IoT 设备的特殊通信需求，但不支持 Amazon SDK 支持的所有服务。Amazon IoT Device SDK 与支持所有 Amazon 服务的 Amazon SDK 兼容；但是，它们使用不同的身份验证方法并连接到不同的终端节点，这可能使在 IoT 设备上使用 Amazon SDK 变得不切实际。

移动设备

the section called “AmazonMobile SDK” (p. 71)支持 MQTT 设备通信、部分 Amazon IoT 服务 API 和其它 Amazon 服务的 API。如果您在受支持的移动设备上进行开发，请查看其 SDK，了解它是否是开发 IoT 解决方案的最佳选择。

C++

Amazon IoT C++ 设备 SDK

Amazon IoT C++ Device SDK 使开发人员可以使用Amazon和 Amazon IoT Core 服务 API 构建互联应用程序。特别是，此 SDK 面向没有资源限制且需要高级功能（例如，消息队列、多线程支持和最新的语言功能）的设备而设计。有关更多信息，请参阅下列内容：

- Amazon IoTC++ v2 GitHub
- Amazon IoT C++ v2 设备 SDK 自述文件
- Amazon IoT C++ v2 设备 SDK 示例
- Amazon IoT C++ v2 设备 SDK API 文档

Python

适用于 Python 的 Amazon IoT 设备 SDK

借助 DeAmazon IoT vice SDevice SDevice SDevice SDevice SDevice SDevic WebSocket e SDevice SDevice SDevice SDevice SDeviceAmazon IoT 通过将设备连接到 Amazon IoT Core 服务 API，用户可以安全地使用由 Amazon IoT Core 提供的消息代理、规则和 Device Shadow 服务，以及 Amazon Lambda、Amazon Kinesis 和 Amazon S3 等其它Amazon服务。

- Amazon IoTGitHthon v2 GitHub
- 适用于 Python 的 Amazon IoT 设备 SDK v2 自述文件
- 适用于 Python 的 Amazon IoT 设备 SDK v2 示例
- Amazon IoT Python v2 设备 SDK API 文档

JavaScript

Amazon IoT的设备 SDK JavaScript

Amazon IoT Core借助DeAmazon IoT vice SD JavaScript evice SDevice SDevice SDevice SDevice SDevice SDevice SDe WebSocket vice JavaScript 它可用于 Node.js 环境和浏览器应用程序。有关更多信息，请参阅下列内容：

- Amazon IoT适用于 JavaScript v2 的设备 SDK 已启用 GitHub
- Amazon IoT适用于 JavaScript v2 的设备 SDK 自述文件
- Amazon IoT适用于 JavaScript v2 示例的设备 SDK
- Amazon IoT适用于 JavaScript v2 的设备 SDK 文档

Java

适用于 Java 的 Amazon IoT 设备 SDK

借助适用于 Java 的Amazon IoT设备软件开发工具包，Java 开发人员可以Amazon IoT Core通过 MQTT 或基于 MQTT 的 MQTT WebSocket 协议来访问的 API。该 SDK 支持 Device Shadow 服务。您可以使用 HTTP 方法 (包括 GET、UPDATE 和 DELETE) 访问影子。该 SDK 还支持简化的影子访问模型，开发人员可以使用 getter 和 setter 方法与影子交换数据，而不必对任何 JSON 文档进行序列化或反序列化。有关更多信息，请参阅下列内容：

- Amazon IoTSDK for Java v2 GitHub
- 适用于 Java 的 Amazon IoT 设备 SDK v2 自述文件
- 适用于 Java 的 Amazon IoT 设备 SDK v2 示例
- Amazon IoT Java v2 设备 SDK API 文档

Embedded C

适用于嵌入式 C 的 Amazon IoT 设备 SDK

Important

该 SDK 供经验丰富的嵌入式软件开发人员使用。

Amazon IoT Device SDK for Embedded C (C-SDK) 是 MIT 开源许可证下多个 C 语言源文件的集合，可在嵌入式应用程序中使用，以安全地将 IoT 设备连接到 Amazon IoT Core。它包括 MQTT、JSON Parser 和 Amazon IoT Device Shadow 库等。它以源代码的形式分发，用于构建到客户固件和应用程序代码、其它库，以及（可选）RTOS（实时操作系统）中。

Amazon IoT Device SDK for Embedded C通常面向需要优化的 C 语言运行时的资源受限设备。您可以在任何操作系统上使用此 SDK，并将其托管在任何类型的处理器（例如 MCU 和 MPU）上。如果您的设备有足够的可用内存和处理资源，则我们建议您使用其他Amazon IoT设备和移动 SDevice SDevice SDevice SDevice SDevice SDevice SDevice SDK for C++ evice SDevice SDeviceAmazon IoT JavaScript

有关更多信息，请参阅下列内容：

- Amazon IoT适用于嵌入式 C 的设备 SDK GitHub
- 适用于嵌入式 C 的 Amazon IoT 设备 SDK 自述文件
- 适用于嵌入式 C 的 Amazon IoT 设备 SDK 示例

设备通信协议

Amazon IoT Core支持设备和客户端使用 MQTT 和基于 WS WebSocket ecure 的 MQTT 协议发布和订阅消息，使用 HTTPS 协议发布消息。所有协议都支持 IPv4 和 IPv6。本节介绍设备和客户端的不同连接选项。

TLS v1.2

Amazon IoT Core 使用 TLS 1.2 版加密所有通信。客户端还必须发送服务器名称指示 (SNI) TLS 扩展。不包括 SNI 的连接尝试将被拒绝。有关更多信息，请参阅 Amazon IoT 中的传输安全。

Amazon IoT Device SDK (p. 73) 支持 MQTT 和基于 WSS 的 MQTT，并支持客户端连接的安全要求。我们建议使用 Amazon IoT Device SDK (p. 73) 将客户端连接到 Amazon IoT。

协议、端口映射和身份验证

设备或客户端如何使用设备终端节点连接到消息代理，取决于它使用的协议。下表列出了 Amazon IoT 设备终端节点支持的协议及其使用的身份验证方法和端口。

协议、身份验证和端口映射

协议	支持的操作	身份验证	端口	ALPN 协议名称
MQTT WebSocket	发布、订阅	Signature Version 4	443	不适用
MQTT WebSocket	发布、订阅	自定义身份验证	443	不适用
MQTT	发布、订阅	X.509 客户端证书	443†	`x-amzn-mqtt-ca`
MQTT	发布、订阅	X.509 客户端证书	8883	不适用
MQTT	发布、订阅	自定义身份验证	443†	`mqtt`
HTTPS	仅发布	Signature Version 4	443	不适用
HTTPS	仅发布	X.509 客户端证书	443†	`x-amzn-http-ca`

协议	支持的操作	身份验证	端口	ALPN 协议名称
HTTPS	仅发布	X.509 客户端证书	8443	不适用
HTTPS	仅发布	自定义身份验证	443	不适用

应用程序层协议协商 (ALPN)

† 使用 X.509 客户端证书身份验证在端口 443 上连接的客户端必须实施应用程序层协议协商 (ALPN) TLS 扩展，并使用由该客户端 ProtocolNameList 发送的 ALPN 中列出的 ALPN 协议名称作为`ClientHello`消息的一部分。
在端口 443 上，IoT: Jobs (p. 73) 终端节点支持 ALPN x-amzn-http-ca HTTP，但 IoT: Jobs 终端节点不支持 ALPN HTT (p. 73) P。
在使用 ALPN 的端口 443 HTTPS 和端口 443 MQTT 上 x-amzn-mqtt-ca，无法使用自定义身份验证 (p. 287)。

客户端连接到其Amazon Web Services 账户的设备终端节点。有关如何查找您账户的设备终端节点的信息，请参阅 the section called “Amazon IoT 设备数据和服务终端节点” (p. 72)。

Note

Amazon SDK 不需要整个 URL。它们只需要终端节点主机名，例如 Device SDev `pubsub.pyice` SDeAmazon IoT vice SDevice GitHub SD 如下表中所示传递整个 URL 可能会生成错误，例如主机名无效。

连接到 Amazon IoT Core

协议	终端节点或 URL
MQTT	*iot-endpoint*
基于 WSS 的 MQTT	wss://*iot-endpoint*/mqtt
HTTPS	https://*iot-endpoint*/topics

为设备通信选择协议

对于大多数通过设备终端节点进行的 IoT 设备通信，您将希望使用 MQTT 或基于 WSS 的 MQTT 协议；但是，设备终端节点也支持 HTTPS。下表比较了 Amazon IoT Core 使用这两种协议进行设备通信的方式。

Amazon IoT设备协议 side-by-side

功能	MQTT (p. 77)	HTTPS (p. 91)
发布/订阅支持	发布和订阅	仅发布
SDK 支持	Amazon Device SDK (p. 73) 支持 MQTT 和 WSS 协议	不支持 SDK，但您可以使用特定于语言的方法来发出 HTTPS 请求
服务质量支持	MQTT QoS 级别 0 和 1 (p. 79)	通过传递查询字符串参数 ? qos=qos 来支持 QoS，其值可以是 0 或 1。您可以添加此查询字符串,以发布具有所需 QoS 值的消息。
可以接收设备离线时错过的消息	是	否

功能	MQTT (p. 77)	HTTPS (p. 91)
clientId 现场支持	是	否
设备断开检测	是	否
安全通信	是。请参阅 协议、端口映射和身份验证 (p. 75)。	是。请参阅 协议、端口映射和身份验证 (p. 75)。
主题定义	定义的应用程序	定义的应用程序
消息数据格式	定义的应用程序	定义的应用程序
协议开销	小于	提高
功耗	小于	提高

连接持续时间限制

HTTPS 连接的持续时间不一定能超过接收和响应请求所需的时间。

MQTT 连接持续时间取决于您所使用的身份验证功能。下表列出了每个功能在理想条件下的最长连接持续时间。

按身份验证功能列出 MQTT 连接持续时间

功能	最长持续时间 *
X.509 客户端证书	1-2 周
自定义身份验证	1-2 周
Signature Version 4	长达 24 小时

*无法保证

使用 X.509 证书和自定义身份验证，则连接持续时间没有硬性限制，但最短可能只有几分钟。导致连接中断的原因多种多样。以下列表包含一些最常见的原因。

- Wi-Fi 可用性中断
- 互联网服务提供商 (ISP) 连接中断
- 服务修补
- 服务部署
- 服务自动扩展
- 服务主机不可用
- 负载均衡器问题和更新
- 客户端错误

您的设备必须执行检测断开连接的策略和重新连接的策略。获取有关断开连接事件及其处理方式的指导信息，请参阅 ??? (p. 1102) 中的 ??? (p. 1102)。

MQTT

MQTT（消息队列遥测传输）是一种广泛采用的轻型消息传递协议，专为受限制的设备而设计。Amazon IoT Core 对 MQTT 的支持基于 MQTT v3.1.1 规范和 MQTT v5.0 规范，但如 the section called “Amazon IoT

与 MQTT 规范不同” (p. 90) 中所述，有一些差异。作为此标准的最新版本，MQTT 5 引入了多项使基于 MQTT 的系统更强大的关键功能，包括新的可扩展性增强、通过原因代码响应改进的错误报告、消息和会话到期计时器，以及自定义用户消息标头。有关 Amazon IoT Core 支持的 MQTT 5 功能的更多信息，请参阅 MQTT 5 支持的功能 (p. 85)。Amazon IoT Core 还支持跨 MQTT 版本（MQTT 3 和 MQTT 5）通信。MQTT 3 发布者可以向将接收 MQTT 5 发布消息的 MQTT 5 订阅者发送 MQTT 3 消息，反之亦然。

Amazon IoT Core 支持使用 MQTT 协议和 MQTT WSS 协议上并由客户端 ID 标识的设备连接。Amazon IoT Device SDK (p. 73) 支持这两种协议，并且是将设备连接到 Amazon IoT Core 的推荐方法。Amazon IoT 设备 SDK 支持设备和客户端连接和访问 Amazon IoT 服务所需的功能。设备 SDK 支持 Amazon IoT 服务所需的身份验证协议以及 MQTT 协议和 MQTT over WSS 协议所需的连接 ID 要求。有关如何使用 Amazon IoT Device SDK 连接到 Amazon 的信息以及指向受支持语言形式的 Amazon IoT 示例的链接，请参阅 the section called “使用 Amazon IoT Device SDK 与 MQTT 连接” (p. 78)。有关 MQTT 消息的身份验证方法和端口映射的更多信息，请参阅协议、端口映射和身份验证 (p. 75)。

虽然我们建议使用 Amazon IoT Device SDK 连接到 Amazon IoT，但它们并不是必需的。但是，如果您不使用 Amazon IoT Device SDK，则必须提供必要的连接和通信安全性。客户端必须在连接请求中发送服务器名称指示 (SNI) TLS 扩展。不包括 SNI 的连接尝试将被拒绝。有关更多信息，请参阅 Amazon IoT 中的传输安全。使用 IAM 用户和Amazon凭证对客户端进行身份验证的客户端必须提供正确的 Signature Version 4 身份验证。

本主题内容：

- 使用 Amazon IoT Device SDK 与 MQTT 连接 (p. 78)
- MQTT 服务质量 (QoS) 选项 (p. 79)
- MQTT 持久性会话 (p. 79)
- MQTT 保留消息 (p. 81)
- 使用 ConnectAttributes (p. 85)
- MQTT 5 支持的功能 (p. 85)
- MQTT 5 属性 (p. 87)
- MQTT 原因代码 (p. 88)
- Amazon IoT 与 MQTT 规范不同 (p. 90)

使用 Amazon IoT Device SDK 与 MQTT 连接

本节包含指向 Amazon IoT Device SDK 以及演示如何将设备连接到 Amazon IoT 的示例程序源代码的链接。此处链接的示例应用程序演示如何使用 MQTT 协议和基于 WSS 的 MQTT 协议连接到 Amazon IoT。

> **Note**
>
> Amazon IoT 设备 SDK 已在开发者预览版中发布了 MQTT 5 客户端。在预览期间，我们可能会对公共 API 进行向后不兼容的更改，Amazon IoT 设备 SDK 中的服务客户端将继续使用 MQTT 3.1.1 客户端。

C++

使用 Amazon IoT C++ 设备 SDK 连接设备

- 以 C++ 语言显示 MQTT 连接示例的示例应用程序源代码
- Amazon IoTC++ 设备 SDK v2 GitHub

Python

使用适用于 Python 的 Amazon IoT 设备 SDK 连接设备

- 以 Python 语言显示 MQTT 连接示例的示例应用程序源代码
- Amazon IoTGitHthon v2 GitHub

JavaScript

使用Amazon IoT设备 SDK JavaScript 连接设备

- 以 MQTT 连接示例显示的 MQTT 连接示例示例的示例应用程序源代码 JavaScript
- Amazon IoT适用于 JavaScript v2 的设备 SDK 已启用 GitHub

Java

使用适用于 Java 的 Amazon IoT 设备 SDK 连接设备

- 以 Java 语言显示 MQTT 连接示例的示例应用程序源代码
- Amazon IoTSDK for Java v2 GitHub

Embedded C

使用适用于嵌入式 C 的 Amazon IoT 设备 SDK 连接设备

> **Important**
>
> 该 SDK 供经验丰富的嵌入式软件开发人员使用。

- 以嵌入式 C 语言显示 MQTT 连接示例的示例应用程序源代码
- Amazon IoT适用于嵌入式 C 的设备 SDK GitHub

MQTT 服务质量 (QoS) 选项

Amazon IoT 和 Amazon IoT Device SDK 支持 MQTT 服务质量 (QoS) 级别 0 和 1。MQTT 协议定义了第三级 QoS（即级别 2），但 Amazon IoT 不支持它。只有 MQTT 协议支持 QoS 功能。HTTPS 通过传递查询字符串参数 `?qos=qos` 来支持 QoS，其值可以是 0 或 1。

下表描述了每个 QoS 级别如何影响发布到消息代理和由消息代理发布的消息。

QoS 级别为...	消息为...	注释
QoS 级别 0	不发送或发送多次	此级别应该用于通过可靠通信链接发送的消息，或者可以毫无问题地错过的消息。
QoS 级别 1	至少发送一次，然后重复发送，直到收到 PUBACK 响应	在发送方收到指示成功传递的 PUBACK 响应之前，该消息不被认为是完整的。

MQTT 持久性会话

持久性会话存储客户端尚未确认的服务质量 (QoS) 为 1 的客户端订阅和消息。当断开连接的设备重新连接到持久性会话时，该会话将恢复，其订阅也将恢复，并且设备将发送在重新连接之前收到的客户端尚未确认的订阅消息。

创建持久性会话

在 MQTT 3 中，您可以通过发送 `CONNECT` 消息并将 `cleanSession` 标记设置为 `0` 来创建 MQTT 持久性会话。如果发送 `CONNECT` 消息的客户端不存在会话，则会创建一个新的持久性会话。如果客户端已存在会话，则客户端会恢复现有会话。要创建干净会话，您需要发送一条 `CONNECT` 消息，并将 `cleanSession` 标志设置为 `1`，当客户端断开连接时，代理将不存储任何会话状态。

在 MQTT 5 中，您可以通过设置 Clean Start 标志和 Session Expiry Interval 来处理永久性会话。干净启动控制连接会话的开始和上一会话的结束。当您设置 Clean Start = 1 时，会创建一个新会话，如果之前的会话存在，将终止之前的会话。当您设置 Clean Start = 0 时，连接会话将恢复之前的会话（如果存在之前的会话）。会话过期间隔控制连接会话的结束。会话过期间隔指定断开连接后会话将持续的时间（以秒为单位，4 字节整数）。设置 Session Expiry interval=0 可使会话在断开连接后立即终止。如果未在 CONNECT 消息中指定会话过期间隔，默认值为 0。

MQTT 5 干净启动和会话过期

属性值	描述
Clean Start= 1	创建新会话并终止之前的会话（如果存在之前的会话）。
Clean Start= 0	如果存在之前的会话，则恢复会话。
Session Expiry Interval> 0	持续会话。
Session Expiry interval= 0	不持续会话。

在 MQTT 5 中，如果设置 Clean Start = 1 和 Session Expiry Interval = 0，这相当于 MQTT 3 清理会话。如果设置 Clean Start = 0 和 Session Expiry Interval > 0，这相当于 MQTT 3 永久会话。

> Note
>
> 不支持跨 MQTT 版本（MQTT 3 和 MQTT 5）的持久性会话。无法将 MQTT 3 持久性会话恢复为 MQTT 5 会话，反之亦然。

持久性会话期间的操作

客户端使用连接已确认 (CONNACK) 消息中的 sessionPresent 属性确定是否存在持久性会话。如果 sessionPresent 为 1，则存在持久性会话，并且在客户端收到 CONNACK 后立即将为客户端存储的所有消息传送给客户端，如重新连接到持久性会话后的消息流量 (p. 80)中所述。如果 sessionPresent 是 1，则客户端无需重新订阅。但是，如果 sessionPresent 为 0，则不存在持久性会话，并且客户端必须重新订阅主题筛选条件。

客户端加入持久性会话后，它可以发布消息并订阅主题筛选条件，而无需在每个操作上附加任何标记。

重新连接到持久性会话后的消息流量

持久性会话表示客户端与 MQTT 消息代理之间的持续连接。当客户端使用持久性会话连接到消息代理时，消息代理会保存客户端在连接期间所做的所有订阅。当客户端断开连接时，消息代理将未确认的 QoS 1 消息和发布的新 QoS 1 消息存储到客户端订阅的主题。根据账户限制存储消息。超过限制的消息将被删除。有关持久消息限制的更多信息，请参阅 Amazon IoT Core 终端节点和配额。当客户端重新连接到其持久性会话时，将恢复所有订阅，并以每秒 10 条消息的最大速率将所有已存储消息发送到客户端。在 MQTT 5 中，如果具有消息过期间隔的出站 QoS1 在客户端离线时过期，则连接恢复后，客户端将不会收到过期消息。

重新连接后，以每秒限制为 10 条已存储消息的速率将已存储消息与任何当前消息流量一起发送到客户端，直到达到 Publish requests per second per connection 限制。由于已存储消息的传递速率受到限制，如果会话在重新连接后有超过 10 条已存储消息要传递，传递所有已存储的消息将需要几秒钟的时间。

结束持久性会话

持久性会话可通过以下方式结束：

- 持久性会话到期时间已过。当消息代理检测到客户端已断开连接（通过客户端断开连接或连接超时）时，持久性会话到期计时器将启动。

- 客户端发送将 `cleanSession` 标记设置为 `1` 的 `CONNECT` 消息。

在 MQTT 3 中，持久性会话过期时间的默认值为一小时，此设置适用于账户中的所有会话。

在 MQTT 5 中，您可以为 CONNECT 和 DISCONNECT 数据包上的每个会话设置会话过期间隔。

对于 DISCONNECT 数据包上的会话过期间隔：

- 如果当前会话的会话过期间隔为 0，则无法在 DISCONNECT 数据包上将会话过期间隔设置为大于 0。
- 如果当前会话的会话过期间隔大于 0，并且您在 DISCONNECT 数据包上将会话过期间隔设置为 0，会话将在 DISCONNECT 上结束。
- 否则，DISCONNECT 数据包的会话过期间隔将更新当前会话的会话过期间隔。

> **Note**
>
> 会话结束时等待发送给客户端的已存储消息将被丢弃；但是，即使无法发送，也仍按标准消息传递费率计费。有关消息定价的更多信息，请参阅 Amazon IoT Core 定价。您可以配置到期时间间隔。

持久性会话到期后重新连接

如果客户端在其持久性会话到期之前未重新连接到该会话，则该会话结束并丢弃其存储的消息。当客户端在会话到期后重新连接并将 `cleanSession` 标记设置为 `0` 时，服务会创建一个新的持久性会话。上一个会话中的任何订阅或消息都不可用于此会话，因为它们在上一个会话到期时被丢弃。

持久性会话消息费用

当消息代理向客户端或离线永久性会话发送消息时，消息将向您的Amazon Web Services 账户收费。当具有持久会话的离线设备重新连接并恢复其会话时，存储的消息将传递到设备并再次向您的账户收费。有关消息定价的更多信息，请参阅 Amazon IoT Core 定价 - 消息收发。

通过使用标准限制增加过程，可以将默认的持久性会话到期时间延长 1 小时。请注意，增加会话过期时间可能会增加消息费用，因为额外的时间可能允许离线设备存储更多消息，而这些额外的消息费用将按标准消息收发费率计入您的账户。会话到期时间为大约时间，会话的持续时间最多可能比账户限制长 30 分钟；但是，会话时间不会短于账户限制。有关会话限制的更多信息，请参阅 Amazon Service Quotas。

MQTT 保留消息

Amazon IoT Core 支持 MQTT 协议中描述的 RETAIN 标志。当客户端在发布的 MQTT 消息上设置 RETAIN 标志时，Amazon IoT Core 保存该消息。然后可以将其发送给新的订阅者，通过调用 `GetRetainedMessage` 操作进行检索，并在 Amazon IoT 控制台中查看。

使用 MQTT 保留消息的示例

- 作为初始配置消息

  客户端订阅主题后，MQTT 保留消息会被发送给客户端。如果希望订阅主题的所有客户端在订阅后立即接收 MQTT 保留消息，您可以发布设置了RETAIN 标志的配置消息。只要发布新的配置消息，订阅客户端也会收到该配置更新。
- 作为最后一个已知的状态消息

  设备可以在当前状态消息上设置 REATIN 标志，以便 Amazon IoT Core 保存它们。当应用程序连接或重新连接时，它们可以订阅此主题并在订阅保留消息主题后立即获得上一次报告状态。这样，它们就不必等到设备发出下一条消息才能看到当前状态。

本节内容：

- 将 MQTT 保留消息存储在 Amazon IoT Core 中的常见任务 (p. 82)
- 账单和保留消息 (p. 83)

- 比较 MQTT 保留消息和 MQTT 持久会话 (p. 83)
- MQTT 保留了消息和 Amazon IoT Device Shadows (p. 84)

将 MQTT 保留消息存储在 Amazon IoT Core 中的常见任务

Amazon IoT Core 使用 RETAIN 标志设置保存 MQTT 消息。这些 保留消息 被作为普通的 MQTT 消息发送给已订阅该主题的所有客户端，同时也被存储起来发送给该主题的新订阅者。

客户端需要特定的策略操作授权才能访问 MQTT 保留消息。获取有关使用保留消息策略的示例，请参阅 保留的消息策略示例 (p. 330)。

本节介绍了涉及保留消息的常见操作。

- 创建保留消息

  客户端在发布 MQTT 消息时确定是否保留消息。客户端可以在发布消息时使用 设备 SDK (p. 1283) 设置 RETAIN 标志。应用程序和服务可以在使用 `Publish` 操作 发布 MQTT 消息时设置 RETAIN 标志。

  每个主题名称只保留一条消息。发布到主题的带有 RETAIN 标志设置的新消息将替换先前发送到该主题的任何现有保留消息。

  注意： 如果您设置了 RETAIN 标志则不能发布到 预留主题 (p. 96)。
- 订阅保留消息主题

  客户端订阅保留消息主题，就像订阅任何其他 MQTT 消息主题一样。通过订阅保留消息主题接收的保留消息均有 RETAIN 标志设置。

  当客户端将具有 0 字节消息负载的保留消息发布到保留消息主题时，将从 Amazon IoT Core 中删除保留消息。已订阅保留消息主题的客户端也将收到该 0 字节消息。

  订阅包含保留消息主题的通配符主题筛选条件可使客户端接收发布到保留消息主题的后续消息，但在订阅时不会传递保留消息。

  注意： 要在订阅时接收保留消息，订阅请求中的主题筛选条件必须与保留消息主题完全匹配。

  订阅保留消息主题时收到的保留消息有 RETAIN 标志设置。订阅客户端在订阅后收到的保留消息则没有。
- 检索保留消息

  当客户端使用保留消息订阅主题时，保留消息会被自动发送给客户端。要让客户端在订阅时收到保留消息，必须订阅保留消息的确切主题名称。通过订阅包含保留消息主题的通配符主题筛选条件，客户端可以接收发布到保留消息主题的后续消息，但不会在订阅时传送保留消息。

  服务和应用可以通过调用 `ListRetainedMessages` 和 `GetRetainedMessage` 列出和检索保留消息。

  在 未 设置 RETAIN 标志的情况下，客户端不会被阻止发布消息到保留消息主题。这可能会导致意外结果，例如保留消息与订阅主题收到的消息不匹配。

  在 MQTT 5 中，如果保留消息设置了消息过期间隔且保留消息已过期，则订阅该主题的新订阅者在成功订阅后将不会收到该保留消息。
- 列出保留消息主题

  您可以通过调用 `ListRetainedMessages` 列出保留消息，并且可以在 Amazon IoT 控制台 中查看保留消息。
- 获取保留消息详细信息

  您可以通过调用 `GetRetainedMessage` 获取保留消息的详细信息，并且可以在 Amazon IoT 控制台 中查看这些信息。
- 保留 Will 消息

连接设备时创建的 MQTT Will 消息 可以通过设置 `Connect Flag bits` 字段中的 `Will Retain` 标记保留。

- 删除保留消息

  通过将设置了 RETAIN 标志的消息和空（0字节）消息负载发布到要删除的保留消息的主题名称，设备、应用程序和服务可以删除保留消息。从 Amazon IoT Core 中删除保留消息的此类消息被发送给订阅了该主题的客户端，但 Amazon IoT Core 不会保留此类消息。

  还可以通过访问 Amazon IoT 控制台 中的保留消息以交互的方式删除保留消息。通过 Amazon IoT 控制台 删除的保留消息也会向已订阅保留消息主题的客户端发送 0 字节的消息。

  保留消息在删除后无法恢复。客户端需要发布新的保留消息才能取代已删除的消息。

- 保留消息的调试和故障排查

  Amazon IoT 控制台 提供多个工具来帮助您进行保留消息故障排查：

  - 保留消息 页面

    Amazon IoT 控制台中的 保留消息 页面提供您的账户在当前区域中存储的保留消息的分页列表。在此页面上，您可以：

    - 查看每条保留消息的详细信息，例如消息负载、QoS、接收时间。
    - 更新保留消息的内容。
    - 删除保留消息。

  - 打开 MQTT 测试客户端

    Amazon IoT 控制台中的 MQTT 测试客户端 页面可以订阅并发布到 MQTT 主题。发布选项让您可以在发布的消息上设置 REATEIN 标志，以模拟设备的行为方式。

  在 Amazon IoT Core 中如何应用保留消息的这些方面可能会产生一些意外的结果。

  - 保留消息限制

    当帐户存储了最大数量的保留消息时，对于使用 RETAIN 设置且有效负载大于 0 字节的发布消息，Amazon IoT Core 将返回限制响应，直到一些保留消息被删除且保留消息数量低于限制。

  - 保留消息传递顺序

    不保证保留消息和订阅消息传递的顺序。

账单和保留消息

从客户端通过使用 Amazon IoT 控制台或调用 `Publish` 发布设置了 RETAIN 标志的消息会产生 Amazon IoT Core 定价-消息收发 中说明的额外消息收发费用。

除了正常的 API 使用费之外，通过客户端、使用 Amazon IoT 控制台或调用 `GetRetainedMessage` 检索保留消息还会产生消息收发费用。Amazon IoT Core 定价-消息收发 中说明了此类额外费用。

在设备意外断开连接时发布的 MQTT Will消息 会产生 Amazon IoT Core 定价-消息收发 中说明的消息收发费用。

有关消息传递定价的更多信息，请参阅 Amazon IoT Core 定价 - 消息传递。

比较 MQTT 保留消息和 MQTT 持久会话

保留消息和持久性会话是 MQTT 的标准功能，使设备能够接收离线时发布的消息。持久会话中可以发布保留消息。本节介绍这些功能的主要方面及其如何协作。

	保留消息	持久会话
主要功能	在连接大量设备后,保留消息可用于配置或通知这些设备。 如果您希望设备仅接收重新连接后发布到主题的最后一条消息，也可以使用保留消息。	持久会话对于间歇性连接而可能错过多条重要消息的设备非常有用。 设备可以连接持久会话来接收离线时发送的消息。
示例	保留消息可以在设备上线时提供有关其环境的配置信息。初始配置可以包括应该订阅的其他消息主题的列表，或者如何配置本地时区的相关信息。	通过间歇连接的蜂窝网络连接的设备可以使用持久会话，避免丢失在设备超出网络覆盖范围或需要关闭其蜂窝无线电时发送的重要消息。
初始订阅主题时收到的消息	订阅带有保留消息的主题后，会收到最近的保留消息。	在订阅没有保留消息的主题后，向该主题发布消息之前不会收到任何消息。
重新连接后订阅的主题	如果没有持久会话，客户端必须在重新连接后订阅主题。	重新连接后，将恢复订阅的主题。
重新连接后收到的消息	订阅带有保留消息的主题后，会收到最近的保留消息。	在设备断开连接时，所有以 QOS = 1 发布并使用 QOS =1 订阅的消息都将在设备重新连接后发送。
数据会话过期	在 MQTT 3 中，保留消息不会过期。它们将被存储，直至被替换或删除。在 MQTT 5 中，保留消息会在您设置的消息过期间隔后过期。有关更多信息，请参阅消息过期 (p. 85)。	如果未在超时期限内重新连接客户端，持久会话将过期。持久会话过期后，在设备断开连接时以 QOS = 1 发布并使用 QOS =1 订阅的客户端订阅和保存消息将被删除。将不会传送已过期的消息。获取有关持久会话的会话过期时间的更多信息，请参阅 the section called “MQTT 持久性会话” (p. 79)。

有关持久性存储的信息，请参阅 the section called “MQTT 持久性会话” (p. 79)。

使用保留消息，发布客户端可确定是否应在连接后保留消息并将其传送到设备，是否有上一个会话。是否存储消息由发布者自行选择，存储的消息将被发送给所有使用 QoS 0 或 QoS 1 订阅的当前和未来客户端。保留消息一次只保存一条给定主题的消息。

当帐户存储了最大数量的保留消息时，Amazon IoT Core 将对使用 RETAIN 集和大于 0 字节的有效负载发布的消息返回限制响应，直到删除一些保留消息且保留消息数不超过限制。

MQTT 保留了消息和 Amazon IoT Device Shadows

保留消息和 Device Shadows 都保留来自设备的数据，但它们的行为不同，用途也不同。本节介绍它们的相似之处和不同之处。

	保留消息	设备影子
消息有效负载具有预定义的结构或架构	正如实施所定义的。MQTT 没有为其消息负载指定结构或架构。	Amazon IoT 支持特定的数据结构。

	保留消息	设备影子
更新消息负载会生成事件消息	发布保留消息会将消息发送到订阅的客户端，但不会生成其他更新消息。	更新 Device Shadow会出现更新描述更改的消息。
消息更新已编号	保留消息不会自动编号。	Device Shadow 文档具有自动版本号和时间戳。
消息有效负载附加到事物资源上	保留消息不会附加到事物资源。	Device Shadows 附加到事物资源。
更新消息有效负载的单个元素	如果不更新整个消息负载，就无法更改消息的单个元素。	可以更新 Device Shadow 文档的各个元素，而无需更新整个 Device Shadow 文档。
客户端在订阅时收到消息数据	在订阅带保留消息的主题后，客户端会自动收到保留消息。	客户端可以订阅 Device Shadow 更新，但他们必须故意请求当前状态。
索引和可搜索性	保留消息不会编制索引进行搜索。	实例集索引 Device Shadow 数据进行搜索和聚合。

使用 ConnectAttributes

`ConnectAttributes` 允许您在 IAM policy 中指定要在连接消息中使用的属性，如 `PersistentConnect` 和 `LastWill`。借助 `ConnectAttributes`，您可以构建预设情况下不允许设备访问新功能的策略，这在设备受到威胁时很有帮助。

`connectAttributes` 支持以下功能：

`PersistentConnect`

当客户端和代理之间的连接中断时，使用 `PersistentConnect` 功能可以保存客户端在连接期间所做的所有订阅。

`LastWill`

当客户端意外断开连接时，使用 `LastWill` 功能可以向 `LastWillTopic` 发布消息。

默认情况下，您的策略具有非持久性连接，并且没有为此连接传递任何属性。如果您想要具有持久连接，则必须在 IAM 策略中指定持久连接。

有关 `ConnectAttributes` 示例，请参阅连接策略示例 (p. 309)。

MQTT 5 支持的功能

Amazon IoT Core 对 MQTT 5 的支持基于 MQTT v5.0 规范，但有一些差异，如 the section called “Amazon IoT 与 MQTT 规范不同” (p. 90) 中所述。

Important

目前不支持共享订阅。

Amazon IoT Core 支持的 MQTT 5 功能列表包括：

干净启动和会话过期

您可以使用“干净启动”和“会话过期”来更灵活地处理持久性会话。干净启动标志指示是否应在不使用现有会话的情况下启动会话。会话过期间隔指示断开连接后会话保持多长时间。可以在断开连接后修改会话过期间隔。有关更多信息，请参阅the section called “MQTT 持久性会话” (p. 79)：

所有 ACK 上的原因代码

您可以使用原因代码更轻松地调试或处理错误消息。消息代理根据与代理的交互类型（订阅、发布、确认）返回原因代码。有关更多信息，请参阅 MQTT 原因代码 (p. 88)。有关 MQTT 原因代码的完整列表，请参阅 MQTT v5 规范。

主题别名

您可以将主题名称替换为主题别名，主题别名是一个双字节整数。使用主题别名可以优化主题名称的传输，从而有可能降低计量数据服务的数据成本。Amazon IoT Core 默认限制为 8 个主题别名。有关更多信息，请参阅 Amazon 一般参考中的 Amazon IoT Core 端点和配额。

消息过期

您可以向已发布的消息添加消息过期值。这些值表示消息过期间隔（以秒为单位）。如果未在该间隔内将消息发送给订阅者，该消息将过期并被删除。如果不设置消息过期值，消息会过期。

在出站时，订阅者将收到一条消息，其中包含到期间隔内的剩余时间。例如，如果入站发布消息的消息过期间隔为 30 秒，并且在 20 秒后路由到订阅者，则消息过期字段将更新为 10。订阅者收到的消息的更新后的 MEI 可能为 0。这是因为只要剩余时间为 999 毫秒或更短，它就会更新为 0。

在 Amazon IoT Core 中，最短消息过期间隔为 1。如果从客户端将间隔设置为 0，则间隔将调整为 1。最长消息过期间隔为 604800（7 天）。高于此值的任何值都将调整为最大值。

在跨版本通信中，消息过期的行为由入站发布消息的 MQTT 版本决定。例如，通过 MQTT5 连接的会话发送的消息过期消息可能会对订阅 MQTT3 会话的设备过期。下表列出了消息过期如何支持以下类型的发布消息：

发布消息类型	消息过期间隔
定期发布	如果服务器未能在指定时间内传送消息，则过期的消息将被删除，订阅者将无法收到该消息。这包括设备未发布确诊其 QoS 1 消息等情况。
保留	如果保留消息过期，而新客户端订阅了该主题，则该客户端在订阅后将不会收到该消息。
最后遗嘱	最后遗嘱消息间隔是在客户端断开连接，并且服务器尝试向其订阅者传送最后遗嘱消息之后开始的。
已排队消息	如果具有消息过期间隔的出站 QoS1 在客户端离线时过期，则持久性会话 (p. 79)恢复后，客户端将不会收到过期的消息。

服务器断开连接

断开连接时，服务器可以主动向客户端发送 DISCONNECT，以通知连接关闭，同时发送断开连接的原因代码。

请求/响应

发布者可以请求接收方在收到时向发布者指定的主题发送响应。

最大数据包大小

客户端和服务器可以独立指定它们支持的最大数据包大小。

负载格式和内容类型

发布消息时，您可以指定负载格式（二进制、文本）和内容类型。这些消息将被转发给消息的接收者。

Last Will 消息

服务器将对来自客户端的所有 DISCONNECT 数据包发送 Last Will 消息，但正常断开连接数据包（0x00）除外。

MQTT 5 属性

MQTT 5 属性是 MQTT 标准的重要补充，用于支持新的 MQTT 5 功能，例如会话过期和请求/响应模式。在 Amazon IoT Core 中，您可以创建规则来转发出站消息中的属性，也可以使用 HTTP 发布来发布带有一些新属性的 MQTT 消息。

下表列出了 Amazon IoT Core 支持的所有 MQTT 5 属性。

属性	描述	输入类型	数据包
负载格式指示符	一个布尔值，指示负载是否格式化为 UTF-8。	字节	PUBLISH、CONNECT
内容类型	一个描述负载内容的 UTF-8 字符串。	UTF-8 字符串	PUBLISH、CONNECT
回复主题	一个 UTF-8 字符串，描述接收方作为请求-响应流程的一部分应发布到的主题。主题不得包含通配符。	UTF-8 字符串	PUBLISH、CONNECT
关联数据	请求消息的发送者用来指示回复消息回复哪个请求的二进制数据。	二进制	PUBLISH、CONNECT
用户属性	一对 UTF-8 字符串。此属性可以在一个数据包中出现多次。接收者将按照键-值对的发送顺序接收键-值对。	UTF-8 字符串对	CONNECT、PUBLISH、Will Properties、SUBSCRIBE、DISCONNEC
消息过期间隔	一个 4 字节整数，表示消息过期间隔（以秒为单位）。如果此数值不存在，消息永远不会过期。	4 字节整数	PUBLISH、CONNECT
会话过期间隔	一个 4 字节整数，表示会话过期间隔（以秒为单位）。Amazon IoT Core 最多支持 7 天，默认最长一小时。如果您设置的值超过账户的最大值，Amazon IoT Core 将在 CONNACK 中返回调整后的值。	4 字节整数	CONNECT、CONNACK、DISCONNECT
分配的客户端标识符	设备未指定客户端 ID 时，Amazon IoT Core 生成的随机客户端 ID。随机客户端 ID 必须是代理当前管理的任何其他会话都未使用的新客户端标识符。	UTF-8 字符串	CONNACK
服务器保持活动	一个 2 字节整数，表示服务器分配的保持活动时间。如果客户端处于非活动状态的时间超过保持活状态的动时间，服务器将断开客户端的连接。	2 字节整数	CONNACK
请求问题信息	一个布尔值，表示在失败时是发送原因字符串还是用户属性。	字节	连接
接收最大值	一个 2 字节整数，表示在未收到 PUBACK 的情况下可以发送的 PUBLISH QOS > 0 数据包的最大数量。	2 字节整数	CONNECT、CONNACK
主题别名最大值	此值表示将被接受作为主题别名的最大值。默认值为 0。	2 字节整数	CONNECT、CONNACK

属性	描述	输入类型	数据包
最大 QoS 数	Amazon IoT Core 支持的 QoS 的最大值。默认值为 1。Amazon IoT Core 不支持 QoS2。	字节	CONNACK
保留可用	一个布尔值，表示 Amazon IoT Core 消息代理是否支持保留的消息。默认值为 1。	字节	CONNACK
最大数据包大小	Amazon IoT Core 接收和发送的最大数据包大小。不能超过 128 KB。	4 字节整数	CONNECT、CONNACK
通配符订阅可用	一个布尔值，指示 Amazon IoT Core 消息代理是否支持通配符订阅可用。默认值为 1。	字节	CONNACK
订阅标识符可用	一个布尔值，指示 Amazon IoT Core 消息代理是否支持订阅标识符可用。默认值为 0。	字节	CONNACK

MQTT 原因代码

MQTT 5 通过原因代码响应改进了错误报告。Amazon IoT Core 可能会返回原因代码，包括但不限于以下按数据包分组的原因代码。有关 MQTT 支持的原因代码的完整列表，请参阅 MQTT 5 规范。

CONNACK 原因代码

值	十六进制	原因代码名称	描述
0	0x00	成功	连接被接受。
128	0x80	未指定的错误	服务器不希望透露失败的原因，或者其他原因代码都不适用。
133	0x85	客户端标识符无效	客户端标识符是有效的字符串，但服务器不允许使用。
134	0x86	用户名或密码错误	服务器不接受客户端指定的用户名或密码。
135	0x87	未授权	客户端无权连接。
144	0x90	主题名称无效	遗嘱主题名称格式正确，但未被服务器接受。
151	0x97	超出配额	已超出实施或管理施加的限制。
155	0x9B	不支持 QoS	服务器不支持遗嘱 QoS 中设置的 QoS。

PUBACK 原因代码

值	十六进制	原因代码名称	描述
0	0x00	成功	该消息已被接受。继续发布 QoS 1 消息。
128	0x80	未指定的错误	接收者不接受发布，但要么不想透露原因，要么与其他值不匹配。
135	0x87	未授权	PUBLISH 未经授权。
144	0x90	主题名称无效	主题名称格式正确，但不被客户端或服务器接受。

值	十六进制	原因代码名称	描述
145	0x91	正在使用数据包标识符	数据包标识符已在使用中。这可能表示客户端和服务器之间的会话状态不匹配。
151	0x97	超出配额	已超出实施或管理施加的限制。

断开连接原因代码

值	十六进制	原因代码名称	描述
0	0x00	正常断开连接	正常关闭连接。不发送 Will 信息。
129	0x81	数据包格式不正确	收到的数据包不符合此规范。
130	0x82	协议错误	收到意外或无序的数据包。
135	0x87	未授权	请求未经授权。
139	0x8B	服务器正在关闭	服务器正在关闭。
141	0x8D	保持活动状态超时	连接已关闭，因为在 1.5 倍的保持活动状态时间内没有收到任何数据包。
142	0x8E	会话已接管	使用相同 ClientID 的另一个连接已连接，导致此连接关闭。
143	0x8F	主题筛选条件无效	主题筛选条件格式正确，但未被服务器接受。
144	0x90	主题名称无效	主题名称格式正确，但未被该客户端或服务器接受。
147	0x93	超出最大接收数	客户端或服务器收到的发布数超过了未发送 PUBACK 或 PUBCOMP 的最大接收数。
148	0x94	主题别名无效	客户端或服务器收到了一个 PUBLISH 数据包，其中包含的主题别名大于其在 CONNECT 或 CONNACK 数据包中发送的最大主题别名。
151	0x97	超出配额	已超出实施或管理施加的限制。
152	0x98	管理操作	由于管理操作，连接已关闭。
155	0x9B	不支持 QoS	客户端指定的 QoS 大于 CONNACK 的“最大 QoS”中指定的 QoS。
161	0xA1	不支持订阅标识符	服务器不支持订阅标识符；不接受订阅。

PUBACK 原因代码

值	十六进制	原因代码名称	描述
0	0x00	授予的 QoS 0	接受订阅，发送的最大 QoS 将为 QoS 0。这可能比请求的 QoS 低。

值	十六进制	原因代码名称	描述
1	0x01	授予的 QoS 1	接受订阅，发送的最大 QoS 将为 QoS 1。这可能比请求的 QoS 低。
128	0x80	未指定的错误	未接受订阅，要么服务器不希望透露原因，要么其他原因代码都不适用。
135	0x87	未授权	客户端无权进行此订阅。
143	0x8F	主题筛选条件无效	主题筛选条件格式正确，但不允许此客户端使用。
145	0x91	正在使用数据包标识符	指定的数据包标识符已在使用中。
151	0x97	超出配额	已超出实施或管理施加的限制。

UNSUBACK 原因代码

值	十六进制	原因代码名称	描述
0	0x00	成功	订阅将被删除。
128	0x80	未指定的错误	无法完成取消订阅，要么服务器不希望透露原因，要么其他原因代码都不适用。
143	0x8F	主题筛选条件无效	主题筛选条件格式正确，但不允许此客户端使用。
145	0x91	正在使用数据包标识符	指定的数据包标识符已在使用中。

Amazon IoT 与 MQTT 规范不同

尽管消息代理的实施基于 MQTT v3.1.1 规范和 MQTT v5.0 规范，但与这些规范有如下区别：

- Amazon IoT 不支持 MQTT 3 的以下数据包：PUBREC、PUBREL 和 PUBCOMP。
- Amazon IoT 不支持 MQTT 5 的以下数据包：PUBREC、PUBREL 和 PUBCOMP。
- Amazon IoT 不支持 MQTT 5 服务器重定向。
- Amazon IoT 仅支持 MQTT 服务质量 (QoS) 级别 0 和 1。Amazon IoT 不支持使用 QoS 级别 2 进行发布或订阅。在请求 QoS 级别 2 时，消息代理不会发送 PUBACK 或 SUBACK。
- 在 Amazon IoT 中，订阅具有 QoS 级别 0 的主题意味着将消息传送零次或多次。消息可能会多次发送。多次发送的消息在发送时可能会使用不同的数据包 ID。在这些情况下，不会设置 DUP 标志。
- 在响应连接请求时，消息代理将发送 CONNACK 消息。此消息包含一个标志，用于指明该连接是否会恢复上一个会话。
- 在发送其它控制数据包或断开连接请求之前，客户端必须等待从 Amazon IoT 消息代理发送到设备的 CONNACK 消息。
- 当客户端订阅主题时，在消息代理开始发送 SUBACK 和客户端开始收到新的匹配消息之间存在时间延迟。
- 当客户端在订阅主题的主题过滤条件中使用通配符 # 时，主题层次结构中处于及其级别以下的所有字符串均会匹配。但是，父主题不匹配。例如，订阅主题 `sensor/#` 接收发布到主题`sensor/`、`sensor/temperature`、`sensor/temperature/room1` 的消息，但不是发布到 `sensor` 的消息。有关通配符的更多信息，请参阅 主题筛选条件 (p. 95)。

- 消息代理使用客户端 ID 标识每个客户。客户端 ID 作为 MQTT 负载的一部分从客户端传递到消息代理。客户端 ID 相同的两个客户端无法同时连接到消息代理。当某个客户端使用另一客户端正在使用的客户端 ID 连接到消息代理时，会接受新的客户端连接，而之前连接的客户端会断开连接。
- 在极少数情况下，消息代理可能会使用不同的数据包 ID 再次发送相同的逻辑 PUBLISH 消息。
- 订阅包含通配符的主题筛选条件无法接收保留消息。要接收保留消息，订阅请求必须包含与保留消息主题完全匹配的主题筛选条件。
- 消息代理并不保证收到消息和 ACK 的顺序。
- Amazon IoT 可能具有与这些规格不同的限制。有关更多信息，请参阅《Amazon IoT 参考指南》中的Amazon IoT Core消息代理和协议限制与限额。

HTTPS

客户端可使用 HTTP 1.0 或 1.1 协议向 REST API 发出请求来发布消息。有关 HTTP 请求使用的身份验证和端口映射，请参阅协议、端口映射和身份验证 (p. 75)。

> **Note**
>
> HTTPS 不像 MQTT 那样支持 `clientId` 值。`clientId` 在使用 MQTT 时可用，但在使用 HTTPS 时不可用。

HTTPS 消息 URL

设备和客户端通过向特定于客户端的终端节点和特定于主题的 URL 发出 POST 请求来发布消息：

```
https://IoT_data_endpoint/topics/url_encoded_topic_name?qos=1"
```

- *IoT_data_endpoint* 是 Amazon IoT 设备数据终端节点 (p. 72)。您可以在 Amazon IoT 控制台中的事物详细信息页面上或使用 Amazon CLI 命令在客户端上查找该终端节点：

  ```
  aws iot describe-endpoint --endpoint-type iot:Data-ATS
  ```

  该终端节点看起来应如下所示：`a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`
- *url_encoded_topic_name* 是正在发送的消息的完整主题名称 (p. 94)。

HTTPS 消息代码示例

以下是一些如何向 Amazon IoT 发送 HTTPS 消息的示例。

Python (port 8443)

```
import requests
import argparse

# define command-line parameters
parser = argparse.ArgumentParser(description="Send messages through an HTTPS
 connection.")
parser.add_argument('--endpoint', required=True, help="Your Amazon IoT data custom
 endpoint, not including a port. " +
                                                       "Ex: \"abcdEXAMPLExyz-ats.iot.us-
east-1.amazonaws.com\"")
parser.add_argument('--cert', required=True, help="File path to your client
 certificate, in PEM format.")
parser.add_argument('--key', required=True, help="File path to your private key, in PEM
 format.")
```

```
parser.add_argument('--topic', required=True, default="test/topic", help="Topic to
 publish messages to.")
parser.add_argument('--message', default="Hello World!", help="Message to publish. " +
                                                        "Specify empty string to publish
 nothing.")

# parse and load command-line parameter values
args = parser.parse_args()

# create and format values for HTTPS request
publish_url = 'https://' + args.endpoint + ':8443/topics/' + args.topic + '?qos=1'
publish_msg = args.message.encode('utf-8')

# make request
publish = requests.request('POST',
            publish_url,
            data=publish_msg,
            cert=[args.cert, args.key])

# print results
print("Response status: ", str(publish.status_code))
if publish.status_code == 200:
        print("Response body:", publish.text)
```

Python (port 443)

```
import requests
import http.client
import json
import ssl

ssl_context = ssl.SSLContext(protocol=ssl.PROTOCOL_TLS_CLIENT)
ssl_context.minimum_version = ssl.TLSVersion.TLSv1_2

# note the use of ALPN
ssl_context.set_alpn_protocols(["x-amzn-http-ca"])
ssl_context.load_verify_locations(cafile="./<root_certificate>")

# update the certificate and the Amazon endpoint
ssl_context.load_cert_chain("./<certificate_in_PEM_Format>",
 "<private_key_in_PEM_format>")
connection = http.client.HTTPSConnection('<the ats IoT endpoint>', 443,
 context=ssl_context)
message = {'data': 'Hello, I'm using TLS Client authentication!'}
json_data = json.dumps(message)
connection.request('POST', '/topics/device%2Fmessage?qos=1', json_data)

# make request
response = connection.getresponse()

# print results
print(response.read().decode())
```

CURL

您可以使用来自客户端或设备的 curl 向 Amazon IoT 发送消息。

使用 curl 从 Amazon IoT 客户端设备发送消息

1. 检查 curl 版本。

   a. 在客户端上，在命令提示符下运行此命令。

      curl --help

在帮助文本中，查找 TLS 选项。您应看到 `--tlsv1.2` 选项。

b. 如果您看到 `--tlsv1.2` 选项，请继续操作。

c. 如果您没有看到 `--tlsv1.2` 选项或者收到 `command not found` 错误，则您可能需要先在客户端上更新或安装 curl，或者安装 `openssl`，然后再继续。

2. 在客户端上安装证书。

   复制您在 Amazon IoT 控制台中注册客户端（事物）时创建的证书文件。请先确保客户端上有这三个证书文件，然后再继续。

   - CA 证书文件（此示例中为 *Amazon-root-CA-1.pem*）。
   - 客户端的证书文件（此示例中为 *device.pem.crt*）。
   - 客户端的私有密钥文件（此示例中为 *private.pem.key*）。

3. 创建 curl 命令行，并替换您的账户和系统的可替换值。

```
curl --tlsv1.2 \
    --cacert Amazon-root-CA-1.pem \
    --cert device.pem.crt \
    --key private.pem.key \
    --request POST \
    --data "{ \"message\": \"Hello, world\" }" \
    "https://IoT_data_endpoint:8443/topics/topic?qos=1"
```

   --tlsv1.2

   使用 TLS 1.2 (SSL)。

   --cacert *Amazon-root-CA-1.pem*

   用于验证对等方的 CA 证书的文件名和路径（如有必要）。

   --cert *device.pem.crt*

   客户端的证书文件名和路径（如有必要）。

   --key *private.pem.key*

   客户端的私有密钥文件名和路径（如有必要）。

   --request POST

   HTTP 请求的类型（此示例中为 POST）。

   --data "*{ \"message\": \"Hello, world\" }*"

   要发布的 HTTP POST 数据。在此示例中，它是一个 JSON 字符串，其内部引号通过反斜杠字符 (\) 进行转义。

   "https://*IoT_data_endpoint*:8443/topics/*topic*?qos=1"

   客户端的 Amazon IoT 设备数据终端节点的 URL，后跟 HTTPS 端口 `:8443`，它后跟关键字 `/topics/` 和主题名称 `topic`（在此示例中）。指定服务质量作为查询参数 `?qos=1`。

4. 在 Amazon IoT 控制台中打开 MQTT 测试客户端。

   按照 使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59) 中的说明执行操作，配置控制台以订阅消息，该消息应带有 curl 命令中所用的名为 *topic* 的主题，或使用 # 通配符主题筛选条件。

5. 测试命令。

   在 Amazon IoT 控制台的测试客户端中监控主题时，请转到客户端并发出您在步骤 3 中创建的 curl 命令行。控制台中应显示客户端的消息。

MQTT 主题

MQTT 主题将标识 Amazon IoT 消息。Amazon IoT 客户端通过向消息提供主题名称来标识其发布的消息。客户端通过将主题筛选条件注册到 Amazon IoT Core 来标识要订阅（接收）的消息。消息代理使用主题名称和主题筛选条件将消息从发布客户端传输到订阅客户端。

消息代理使用主题来识别使用 MQTT 发送的消息以及使用 HTTP 发送到 HTTPS 消息 URL (p. 91) 的消息。

虽然 Amazon IoT 支持某些保留的系统主题 (p. 96)，但大多数 MQTT 主题是由您（即系统设计人员）创建和管理的。Amazon IoT 使用主题识别从发布客户端接收的消息，并选择要发送到订阅客户端的消息，如以下各节所述。在为系统创建主题命名空间之前，请查看 MQTT 主题的特征，以创建最适合您的 IoT 系统的主题名称层次结构。

主题名称

主题名称和主题筛选条件是 UTF-8 编码的字符串。它们可以使用正斜杠 (/) 字符分隔层次结构的级别来表示信息的层次结构。例如，此主题名称可以引用房间 1 中的温度传感器：

- `sensor/temperature/room1`

在此示例中，其他房间中可能还有其它类型的传感器，其主题名称如下：

- `sensor/temperature/room2`
- `sensor/humidity/room1`
- `sensor/humidity/room2`

> **Note**
>
> 在考虑系统中消息的主题名称时，请记住：
>
> - 主题名称和主题筛选条件区分大小写。
> - 主题名称不得包含个人身份信息。
> - 以 $ 开头的主题名称均为只能由 Amazon IoT Core 使用的预留主题 (p. 96)。
> - Amazon IoT Core 无法在Amazon Web Services 账户或区域之间发送或接收消息。

有关设计主题名称和命名空间的更多信息，请参阅我们的白皮书《为 Amazon IoT Core Core 设计 MQTT 主题》。

有关应用程序如何发布和订阅消息的示例，请从 Amazon IoT Core 入门 (p. 15) 和 Amazon IoT Device SDK、Mobile SDK 和 Amazon IoT Device Client (p. 1283) 开始查阅。

> **Important**
>
> 主题命名空间仅限于Amazon Web Services 账户和区域。例如，一个区域中的某个Amazon Web Services 账户使用的 `sensor/temp/room1` 主题与另一个区域中的同一Amazon账户或任何区域中的任何其它 Amazon Web Services 账户 使用的 `sensor/temp/room1` 主题不同。

主题 ARN

所有主题 ARN (Amazon Resource Name) 都采用以下形式：

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

例如，`arn:aws:iot:us-west-2:123EXAMPLE456:topic/application/topic/device/sensor` 是主题 `application/topic/device/sensor` 的 ARN。

主题筛选条件

订阅客户端会向消息代理注册主题筛选条件，以指定消息代理应将消息发送到的主题。主题筛选条件可以是用于订阅单个主题的单个主题名称，也可以包含通配符以同时订阅多个主题名称。

发布客户端无法在其发布的主题名称中使用通配符。

下表列出了可在主题筛选条件中使用的通配符。

主题通配符

通配符	匹配项	注意
#	主题层次结构中位于其级别及其以下的所有字符串。	必须是主题筛选条件中的最后一个字符。 必须是其主题层次结构级别中的唯一字符。 可以在还包含 + 通配符的主题筛选条件中使用。
+	级别中包含字符的任何字符串。	必须是其主题层次结构级别中的唯一字符。 可在主题筛选条件的多个级别中使用。

将通配符用于之前的传感器主题名称示例：

- `sensor/#` 订阅接收发布到 `sensor/`、`sensor/temperature` 和 `sensor/temperature/room1` 的消息，但不会接收发布到 `sensor` 的消息。
- `sensor/+/room1` 订阅接收发布到 `sensor/temperature/room1` 和 `sensor/humidity/room1` 的消息，但不会接收发布到 `sensor/temperature/room2` 或 `sensor/humidity/room2` 的消息。

主题筛选条件 ARN

所有主题筛选条件 ARN (Amazon Resource Name) 都采用以下形式：

```
arn:aws:iot:aws-region:AWS-account-ID:topicfilter/TopicFilter
```

例如，`arn:aws:iot:us-west-2:123EXAMPLE456:topicfilter/application/topic/+/sensor` 是主题筛选条件 `application/topic/+/sensor` 的 ARN。

MQTT 消息负载

您的 MQTT 消息中发送的消息有效负载并不由 Amazon IoT 指定，除非是用于 the section called “保留的主题” (p. 96) 中的一个。为了满足应用程序的需求，我们建议您在协议的 Amazon IoT Core 服务配额限制范围内为主题定义消息有效负载。

为消息负载使用 JSON 格式可使 Amazon IoT 规则引擎解析消息并将 SQL 查询应用于消息。如果应用程序不需要规则引擎将 SQL 查询应用于消息有效负载，则您可以使用应用程序所需的任何数据格式。有关 SQL 查询中使用的 JSON 文档中的限制和预留字符的信息，请参阅 JSON 扩展 (p. 583)。

有关设计 MQTT 主题及其相应的消息负载的更多信息，请参阅为 Amazon IoT Core 设计 MQTT 主题。

如果消息大小限制超过服务配额，则会导致 CLIENT_ERROR，并显示原因 PAYLOAD_LIMIT_EXCEEDED 和“消息有效负载超过消息类型的大小限制。”有关消息大小限制的更多信息，请参阅 Amazon IoT Core 消息代理限制和配额。

保留的主题

以美元符号 ($) 开头的主题保留供 Amazon IoT 使用。您可以在允许的情况下订阅和发布到这些保留的主题；但是，您不能创建以美元符号开头的新主题。对保留的主题执行不受支持的发布或订阅操作可能会导致连接终止。

资产模型主题

主题	允许的客户端操作	描述
$aws/sitewise/asset-models/*assetModelId*/assets/*assetId*/properties/*propertyId*	Subscribe	Amazon IoT SiteWise 向此主题发布资产属性通知。有关更多信息，请参阅 Amazon IoT SiteWise 用户指南中的与其它 Amazon服务交互。

Amazon IoT Device Defender 主题

这些消息支持简明二进制对象展现 (CBOR) 格式和 JavaScript 对象表示法 (JSON) 格式的响应缓冲区，具体取决于主题的 *payload-format*。Amazon IoT Device Defender 主题仅支持 MQTT 发布。

payload-format	响应格式数据类型
cbor	简洁二进制对象表示法 (CBOR)
json	JavaScript 对象表示法 (JSON)

有关更多信息，请参阅从设备发送指标 (p. 978)。

主题	允许的操作	描述
$aws/things/*thingName*/defender/metrics/*payload-format*	发布	Amazon IoT Device Defender 代理会向此主题发布指标。有关更多信息，请参阅从设备发送指标 (p. 978)。
$aws/things/*thingName*/defender/metrics/*payload-format*/accepted	Subscribe	在 Amazon IoT Device Defender 代理向 $aws/things/*thingName*/defender/metrics/*payload-format* 发布成功消息后，Amazon IoT 将发布到此主题。有关更多信息，请参阅从设备发送指标 (p. 978)。
$aws/things/*thingName*/defender/metrics/*payload-format*/rejected	Subscribe	在 Amazon IoT Device Defender 代理向 $aws/things/*thingName*/defender/metrics/*payload-format* 发布未成功消息后，Amazon IoT 将发布到此主题。有关更

主题	允许的操作	描述
		多信息，请参阅从设备发送指标 (p. 978)。

事件主题

主题	允许的客户端操作	描述
$aws/events/certificates/registered/*caCertificateId*	Subscribe	在 Amazon IoT 自动注册证书以及客户端呈现具有 PENDING_ACTIVATION 状态的证书时，Amazon IoT 发布此消息。有关更多信息，请参阅the section called “配置客户端的首次连接以进行自动注册” (p. 280)。
$aws/events/job/*jobID*/canceled	Subscribe	当取消某个任务时，Amazon IoT 会发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/job/*jobID*/cancellation_in_progress	Subscribe	当某个任务正在被取消时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/job/*jobID*/completed	Subscribe	当完成某个任务时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/job/*jobID*/deleted	Subscribe	当某个任务被删除时，Amazon IoT 将发布此消息。有关更多信息，请参阅 任务事件 (p. 1099)。
$aws/events/job/*jobID*/deletion_in_progress	Subscribe	当某个任务正在被删除时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobID*/canceled	Subscribe	当取消某个任务执行时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobID*/deleted	Subscribe	当删除某个任务执行时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobID*/failed	Subscribe	当某个任务执行失败时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。

主题	允许的客户端操作	描述
$aws/events/jobExecution/*jobID*/rejected	Subscribe	当某个任务执行被拒绝时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobID*/removed	Subscribe	当某个任务执行被删除时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobID*/succeeded	Subscribe	当任务执行成功时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobID*/timed_out	Subscribe	当某个任务执行超时，Amazon IoT 将发布此消息。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/presence/connected/*clientId*	Subscribe	当使用特定客户端 ID 的 MQTT 客户端连接至 Amazon IoT 时，Amazon IoT 将向此主题发布。有关更多信息，请参阅连接/断开连接事件 (p. 1102)。
$aws/events/presence/disconnected/*clientId*	Subscribe	当使用特定客户端 ID 的 MQTT 客户端断开连接至 Amazon IoT 时，Amazon IoT 将向此主题发布。有关更多信息，请参阅连接/断开连接事件 (p. 1102)。
$aws/events/subscriptions/subscribed/*clientId*	Subscribe	当使用特定客户端 ID 的 MQTT 客户端订阅 MQTT 主题时，Amazon IoT 将向此主题发布。有关更多信息，请参阅订阅/取消订阅事件 (p. 1104)。
$aws/events/subscriptions/unsubscribed/*clientId*	Subscribe	当使用特定客户端 ID 的 MQTT 客户端取消订阅 MQTT 主题时，Amazon IoT 将向此主题发布。有关更多信息，请参阅订阅/取消订阅事件 (p. 1104)。
$aws/events/thing/*thingName*/created	Subscribe	当创建 *thingName* 事物时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thing/*thingName*/updated	Subscribe	当 *thingName* 事物更新时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。

主题	允许的客户端操作	描述
$aws/events/thing/*thingName*/deleted	Subscribe	当 *thingName* 事物被删除时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingGroup/*thingGroupName*/created	Subscribe	当创建了事物组 *thingGroupName* 时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingGroup/*thingGroupName*/updated	Subscribe	当更新了事物组 *thingGroupName* 时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingGroup/*thingGroupName*/deleted	Subscribe	当删除了事物组 *thingGroupName* 时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingType/*thingTypeName*/created	Subscribe	当创建了 *thingTypeName* 事物类型时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingType/*thingTypeName*/updated	Subscribe	当更新了 *thingTypeName* 事物类型时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingType/*thingTypeName*/deleted	Subscribe	当删除了 *thingTypeName* 事物类型时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingTypeAssociation/thing/*thingName*/*thingTypeName*	Subscribe	当事物 *thingName* 与事物类型 *thingTypeName* 相关联或与断开关联，时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。

主题	允许的客户端操作	描述
$aws/events/thingGroupMembership/thingGroup/*thingGroupName*/thing/*thingName*/added	Subscribe	当事物 *thingName* 被添加到事物组 *thingGroupName* 时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingGroupMembership/thingGroup/*thingGroupName*/thing/*thingName*/removed	Subscribe	当事物 *thingName* 从事物组 *thingGroupName* 中删除时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingGroupHierarchy/thingGroup/*parentThingGroupName*/childThingGroup/*childThingGroupName*/added	Subscribe	当事物组 *childThingGroupName* 被添加到事物组 *parentThingGroupName* 时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。
$aws/events/thingGroupHierarchy/thingGroup/*parentThingGroupName*/childThingGroup/*childThingGroupName*/removed	Subscribe	当事物组 *childThingGroupName* 从事物组 *parentThingGroupName* 中删除时，Amazon IoT 将发布到此主题。有关更多信息，请参阅the section called “注册表事件” (p. 1091)。

队列预置主题

这些消息支持简洁二进制对象表示法 (CBOR) 格式和 JavaScript 对象表示法 (JSON) 格式的响应缓冲区，具体取决于主题的 *payload-format*。

payload-format	响应格式数据类型
cbor	简洁二进制对象表示法 (CBOR)
json	JavaScript 对象表示法 (JSON)

有关更多信息，请参阅设备预调配 MQTT API (p. 797)。

主题	允许的客户端操作	描述
$aws/certificates/create/*payload-format*	发布	发布到此主题以从证书签名请求 (CSR) 创建证书。
$aws/certificates/create/*payload-format*/accepted	Subscribe	当成功调用 $aws/certificates/create/*payload-format*

主题	允许的客户端操作	描述
		时，Amazon IoT 将发布到此主题。
$aws/certificates/create/*payload-format*/rejected	Subscribe	当未成功调用 $aws/certificates/create/*payload-format* 时，Amazon IoT 将发布此主题。
$aws/certificates/create-from-csr/*payload-format*	发布	发布到此主题以从 CSR 创建证书。
$aws/certificates/create-from-csr/*payload-format*/accepted	Subscribe	Amazon IoT 将对 $aws/certificates/create-from-csr/*payload-format* 的成功调用发布到此主题。
$aws/certificates/create-from-csr/*payload-format*/rejected	Subscribe	Amazon IoT 将对 $aws/certificates/create-from-csr/*payload-format* 的不成功调用发布到此主题。
$aws/events/presence/connected/*clientId*	Subscribe	当使用特定客户端 ID 的 MQTT 客户端连接至 Amazon IoT 时，Amazon IoT 将向此主题发布。有关更多信息，请参阅连接/断开连接事件 (p. 1102)。
$aws/provisioning-templates/*templateName*/provision/*payload-format*	发布	发布到此主题以注册事物。
$aws/provisioning-templates/*templateName*/provision/*payload-format*/accepted	Subscribe	当成功调用 $aws/provisioning-templates/*templateName*/provision/*payload-format* 时，Amazon IoT 将发布到此主题。
$aws/provisioning-templates/*templateName*/provision/*payload-format*/rejected	Subscribe	当未成功调用 $aws/provisioning-templates/*templateName*/provision/*payload-format* 时，Amazon IoT 将发布到此主题。

任务主题

Note

此表中标记为 Receive（接收）的客户端操作表示无论客户端是否订阅了主题，Amazon IoT 都会直接向发出请求的客户端发布的主题。即使客户端尚未订阅这些消息，也会收到这些消息。
这些响应消息不会通过消息代理，也无法由其它客户端或规则订阅。要订阅与工作活动消息相关的任务活动，请使用 `notify` 和 `notify-next` 主题。
当订阅任务和您的机群监控解决方案的 `jobExecution` 事件主题时，您必须首先启用任务和任务执行事件 (p. 1088)接收云端的任何事件。
有关更多信息，请参阅任务设备 MQTT API (p. 712)。

主题	允许的客户端操作	描述
$aws/things/*thingName*/jobs/get	发布	设备向此主题发布一条消息以发出 GetPendingJobExecutions 请求。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/get/accepted	订阅，接收	设备订阅此主题以接收来自 GetPendingJobExecutions 请求的成功响应。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/get/rejected	订阅，接收	当 GetPendingJobExecutions 请求被拒绝时，设备会订阅此主题以接收响应。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/start-next	发布	设备向此主题发布一条消息以发出 StartNextPendingJobExecution 请求。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/start-next/accepted	订阅，接收	设备订阅此主题以接收 StartNextPendingJobExecution 请求的成功响应。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/start-next/rejected	订阅，接收	当 StartNextPendingJobExecution 请求被拒绝时，设备会订阅此主题以接收响应。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/*jobId*/get	发布	设备向此主题发布一条消息以发出 DescribeJobExecution 请求。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/*jobId*/get/accepted	订阅，接收	设备订阅此主题以接收 DescribeJobExecution 请求的成功响应。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/*jobId*/get/rejected	订阅，接收	当 DescribeJobExecution 请求被拒绝时，设备会订阅此主题以接收响应。有关更多信

主题	允许的客户端操作	描述
		息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/*jobId*/update	发布	设备向此主题发布一条消息以发出 UpdateJobExecution 请求。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/*jobId*/update/accepted	订阅，接收	设备订阅此主题以接收 UpdateJobExecution 请求的成功响应。有关更多信息，请参阅任务设备 MQTT API (p. 712)。 注意 仅向 $aws/things/*thingName*/jobs/*jobId*/update 发布消息的设备将收到有关此主题的消息。
$aws/things/*thingName*/jobs/*jobId*/update/rejected	订阅，接收	当 UpdateJobExecution 请求被拒绝时，设备会订阅此主题以接收响应。有关更多信息，请参阅任务设备 MQTT API (p. 712)。 注意 仅向 $aws/things/*thingName*/jobs/*jobId*/update 发布消息的设备将收到有关此主题的消息。
$aws/things/*thingName*/jobs/notify	Subscribe	设备订阅此主题，以在某个事物的待处理执行列表中添加或删除了任务执行时接收通知。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/things/*thingName*/jobs/notify-next	Subscribe	设备订阅此主题，以在事物的下一个待处理任务执行发生更改时接收通知。有关更多信息，请参阅任务设备 MQTT API (p. 712)。
$aws/events/job/*jobId*/completed	Subscribe	当某个任务完成时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。

主题	允许的客户端操作	描述
$aws/events/job/*jobId*/canceled	Subscribe	当取消某个任务时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/job/*jobId*/deleted	Subscribe	当删除某个任务时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/job/*jobId*/cancellation_in_progress	Subscribe	当开始取消某个任务时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/job/*jobId*/deletion_in_progress	Subscribe	当开始删除某个任务时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobId*/succeeded	Subscribe	当任务执行成功时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobId*/failed	Subscribe	当任务执行失败时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobId*/rejected	Subscribe	当任务执行被拒绝时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobId*/canceled	Subscribe	当任务执行取消时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobId*/timed_out	Subscribe	当任务执行超时时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobId*/removed	Subscribe	当任务执行被移除时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。
$aws/events/jobExecution/*jobId*/deleted	Subscribe	当任务执行被删除时，任务服务在此主题上发布事件。有关更多信息，请参阅任务事件 (p. 1099)。

规则主题

主题	允许的客户端操作	描述
$aws/rules/*ruleName*	发布	设备或应用程序向此主题发布消息以直接触发规则。有关更多信息，请参阅借助基本摄取功能，降低消息收发成本 (p. 529)。

安全隧道主题

主题	允许的客户端操作	描述
$aws/things/*thing-name*/tunnels/notify	Subscribe	Amazon IoT 发布此消息，以便 IoT 代理在远程设备上启动本地代理。有关更多信息，请参阅the section called “IoT 代理代码段” (p. 762)。

影子主题

本节中的主题由命名和未命名的影子使用。每个影子使用的主题仅在主题前缀上有所不同。下表显示每种影子类型使用的主题前缀。

ShadowTopicPrefix 值	影子类型
$aws/things/*thingName*/shadow	未命名的（经典）影子
$aws/things/*thingName*/shadow/name/*shadowName*	命名的影子

要创建完整的主题，请为要表示的影子类型选择 *ShadowTopicPrefix*，将 *thingName* 和 *shadowName*（如果适用）替换为相应的值，然后在其后面附加主题存根，如下表中所示。请记住，主题区分大小写。

主题	允许的客户端操作	描述
ShadowTopicPrefix/delete	发布/订阅	设备或应用程序向此主题发布消息以删除影子。有关更多信息，请参阅 /delete (p. 626)。
ShadowTopicPrefix/delete/accepted	Subscribe	当一个影子被删除时，Device Shadow 服务将向此主题发送消息。有关更多信息，请参阅 /delete/accepted (p. 626)。
ShadowTopicPrefix/delete/rejected	Subscribe	当删除影子的请求遭拒时，Device Shadow 服务将向此主题发送消息。有关

主题	允许的客户端操作	描述
		更多信息，请参阅 /delete/rejected (p. 627)。
ShadowTopicPrefix/get	发布/订阅	应用程序或事物向此主题发布空消息来获取影子。有关更多信息，请参阅Device Shadow MQTT 主题 (p. 620)。
ShadowTopicPrefix/get/accepted	Subscribe	当获取影子的请求获批时，Device Shadow 服务将向此主题发送消息。有关更多信息，请参阅 /get/accepted (p. 621)。
ShadowTopicPrefix/get/rejected	Subscribe	当获取影子的请求遭拒时，Device Shadow 服务将向此主题发送消息。有关更多信息，请参阅 /get/rejected (p. 621)。
ShadowTopicPrefix/update	发布/订阅	事物或应用程序向此主题发布消息以更新影子。有关更多信息，请参阅 /update (p. 622)。
ShadowTopicPrefix/update/accepted	Subscribe	当影子更新成功时，Device Shadow 服务将向此主题发送消息。有关更多信息，请参阅 /update/accepted (p. 624)。
ShadowTopicPrefix/update/rejected	Subscribe	当影子更新遭拒时，Device Shadow 服务将向此主题发送消息。有关更多信息，请参阅 /update/rejected (p. 625)。
ShadowTopicPrefix/update/delta	Subscribe	当检测到影子的“reported”部分与“desired”部分之间存在差异时，Device Shadow 服务将向此主题发送消息。有关更多信息，请参阅 /update/delta (p. 623)。
ShadowTopicPrefix/update/documents	Subscribe	每次影子更新成功执行时，Amazon IoT 都会向该主题发布状态文档。有关更多信息，请参阅 /update/documents (p. 624)。

基于 MQTT 的文件传输主题

这些消息支持简洁二进制对象表示法 (CBOR) 格式和 JavaScript 对象表示法 (JSON) 格式的响应缓冲区，具体取决于主题的 *payload-format*。

payload-format	响应格式数据类型
cbor	简洁二进制对象表示法 (CBOR)
json	JavaScript 对象表示法 (JSON)

主题	允许的客户端操作	描述
$aws/things/*ThingName*/streams/*StreamId*/data/*payload-format*	Subscribe	如果接受来自设备的“GetStream”请求，则Amazon基于 MQTT 的文件传输将发布到此主题。负载包含流数据。有关更多信息，请参阅在设备中使用 Amazon IoT 基于 MQTT 的文件传输 (p. 836)。
$aws/things/*ThingName*/streams/*StreamId*/get/*payload-format*	发布	设备向此主题发布消息以执行“GetStream”请求。有关更多信息，请参阅在设备中使用 Amazon IoT 基于 MQTT 的文件传输 (p. 836)。
$aws/things/*ThingName*/streams/*StreamId*/description/*payload-format*	Subscribe	如果接受来自设备的“DescribeStream”请求，则Amazon基于 MQTT 的文件传输将发布到此主题。负载包含流描述。有关更多信息，请参阅在设备中使用 Amazon IoT 基于 MQTT 的文件传输 (p. 836)。
$aws/things/*ThingName*/streams/*StreamId*/describe/*payload-format*	发布	设备向此主题发布消息以执行“DescribeStream”请求。有关更多信息，请参阅在设备中使用 Amazon IoT 基于 MQTT 的文件传输 (p. 836)。
$aws/things/*ThingName*/streams/*StreamId*/rejected/*payload-format*	Subscribe	如果来自设备的“DescribeStream”或“GetStream”请求被拒绝，则Amazon基于 MQTT 的文件传输将发布到此主题。有关更多信息，请参阅在设备中使用 Amazon IoT 基于 MQTT 的文件传输 (p. 836)。

保留的主题 ARN

所有保留的主题 ARN (Amazon Resource Name) 均采用以下形式：

```
arn:aws:iot:aws-region:AWS-account-ID:topic/Topic
```

例如，`arn:aws:iot:us-west-2:123EXAMPLE456:topic/$aws/things/thingName/jobs/get/accepted` 是保留主题 `$aws/things/thingName/jobs/get/accepted` 的 ARN。

可配置的终端节点

Note

此功能在 GovCloud Amazon Web Services 区域 中不可用。

借助 Amazon IoT Core，您可以使用域配置来配置和管理数据端点的行为。您可以生成多个Amazon IoT Core数据端点，还可以使用您自己的完全限定域名（以及关联的服务器证书）和授权方自定义数据端点。有关自定义授权方的更多信息，请参阅 the section called “自定义身份验证” (p. 287)。

Amazon IoT Core 使用服务器名称指示 (SNI) TLS 扩展来应用域配置。设备在连接时必须使用此扩展。它们还必须传递与域配置中指定的域名相同的服务器名称。* 要测试此服务，请使用 GitHub 中的 Amazon IoT Device SDK v2 版本。

Note

如果您在 Amazon Web Services 账户 中创建多个数据端点，它们将共享 Amazon IoT Core 资源，例如 MQTT 主题、设备影子和规则。

您可以使用域配置来简化以下这样的任务。

- 将设备迁移到 Amazon IoT
- 通过为不同的设备类型维护单独的域配置来支持异构设备实例集。
- 在将应用程序基础设施迁移到 Amazon IoT 时，保留品牌标识（例如，通过域名保留）。

您也可以配置完全限定域名 (FQSN) 和关联的服务器证书。您还可以关联自定义授权方。有关更多信息，请参阅自定义身份验证 (p. 287)。

Note

Amazon IoT 使用服务器名称指示 (SNI) TLS 扩展来应用域配置。设备在连接时必须使用此扩展名，并传递与域配置中指定的域名相同的服务器名称。要测试此服务，请使用 GitHub 中的每个 Amazon IoT Device SDK 的 v2 版本。

Note

当您为Amazon IoT自定义域配置提供服务器证书时，证书最多有四个域名。有关更多信息，请参阅 Amazon IoT Core 终端节点和配额。

主题

- 创建和配置 Amazon 托管域 (p. 108)
- 创建和配置自定义域 (p. 109)
- 管理域配置 (p. 111)

创建和配置 Amazon 托管域

您可以使用 CreateDomainConfiguration API 在Amazon托管的域上创建可配置的端点。Amazon 托管域的域配置包括以下内容：

- `domainConfigurationName` — 用于标识域配置的用户定义名称。

  Note

  以 `IoT:` 开头的域配置名称将针对默认终端节点保留，不能使用。此外，此值对您的 Amazon Web Services 区域 必须是唯一的。

- defaultAuthorizerName（可选）— 要在端点上使用的自定义授权方的名称。
- allowAuthorizerOverride — 一个布尔值，指定设备是否可以通过在请求的 HTTP 标头中指定不同的授权方来覆盖默认授权方。如果为 defaultAuthorizerName 指定了值，则需要此值。
- serviceType — Amazon IoT 目前仅支持 DATA 服务类型。如果指定 DATA，则 Amazon IoT 返回端点类型为 iot:Data-ATS 的端点。您无法创建可配置的 iot:Data (VeriSign) 终端节点。

以下 Amazon CLI 命令为 Data 终端节点创建域配置。

```
aws iot create-domain-configuration --domain-configuration-name "myDomainConfigurationName"
 --service-type "DATA"
```

创建和配置自定义域

域配置允许您指定要连接到 Amazon IoT 的自定义完全限定域名 (FQDN)。通过自定义域，您可以管理自己的服务器证书，从而可以管理详细信息，例如用于签署证书的根证书颁发机构 (CA)、签名算法、证书链深度和证书的生命周期。

使用自定义域设置域配置的工作流包括以下三个阶段。

1. 在 Amazon Certificate Manager 中注册服务器证书 (p. 109)
2. 创建域配置 (p. 110)
3. 创建 DNS 记录 (p. 111)

在Amazon证书管理器中注册服务器证书

在使用自定义域创建域配置之前，必须在 Amazon Certificate Manager (ACM) 中注册服务器证书链。您可以使用三种类型的服务器证书。

- ACM 生成的公有证书 (p. 110)
- 由公有 CA 签名的外部证书 (p. 110)
- 由私有 CA 签名的外部证书 (p. 110)

> **Note**
>
> 如果证书包含在 Mozilla 的信任 ca-bundle 中，则 Amazon IoT 会考虑由公共 CA 签署的证书。

证书要求

请参阅导入证书的先决条件，了解将证书导入 ACM 的要求。除了这些要求之外，Amazon IoT Core 还添加了以下要求。

- 叶证书必须包含扩展密钥用法 x509 v3 扩展，且其值应为 serverAuth（TLS Web 服务器身份验证）。如果您从 ACM 请求证书，则会自动添加此扩展。
- 最大证书链深度为 5 个证书。
- 最大证书链大小为 16KB。

对多个域使用一个证书

如果您计划使用一个证书来涵盖多个子域，请在公用名 (CN) 或使用者备用名称 (SAN) 字段中使用通配符域。例如，使用 ***.iot.example.com** 涵盖 cover dev.iot.example.com、qa.iot.example.com 和 prod.iot.example.com。每个 FQDN 都需要自己的域配置，但多个域配置可以使用同一个通配符值。CN 或 SAN 必须涵盖要用作自定义域的 FQDN。如果存在 SAN，会忽略 CN，SAN 必须覆盖您想要用作自定义域

的 FQDN。此涵盖范围可以是完全匹配或通配符匹配。在验证通配符证书并将其注册到账户后，将阻止区域中的其他账户创建与证书重叠的自定义域。

以下各部分介绍如何获取每种类型的证书。每个证书资源都需要一个使用 ACM 注册的 Amazon Resource Name (ARN)，该 ACM 即您在创建域配置时所用的那个。

ACM 生成的公有证书

您可以使用 RequestCertificate API 为您的自定义域生成公有证书。以这种方式生成证书时，ACM 将验证您对自定义域的所有权。有关更多信息，请参阅 Amazon Certificate Manager 用户指南中的请求公有证书。

由公有 CA 签名的外部证书

如果您已经拥有由公有 CA（Mozilla 信任的 ca-bundle 中包含的 CA）签名的服务器证书，则可以使用 ImportCertificate API 将证书链直接导入 ACM。要了解有关此任务以及先决条件和证书格式要求的更多信息，请参阅导入证书。

由私有 CA 签名的外部证书

如果您已经具有由私有 CA 签名或自签名的服务器证书，则可以使用该证书创建域配置，但是还必须在 ACM 中创建一个额外的公有证书以验证域的所有权。若要执行此操作，请在 ACM 中使用 ImportCertificate API 注册您的服务器证书链。要了解有关此任务以及先决条件和证书格式要求的更多信息，请参阅导入证书。

将证书导入到 ACM 后，您可以使用 RequestCertificate API 为您的自定义域生成公有证书。以这种方式生成证书时，ACM 将验证您对自定义域的所有权。有关更多信息，请参阅请求公有证书。创建域配置时，请使用此公有证书作为验证证书。

创建域配置

您可以使用 CreateDomainConfiguration API 在自定义域上创建可配置的端点。自定义域的域配置包括以下内容：

- `domainConfigurationName` — 用于标识域配置的用户定义名称。

  Note

  以 `IoT:` 开头的域配置名称将针对默认终端节点保留，不能使用。此外，此值对您的 Amazon Web Services 区域 必须是唯一的。
- `domainName` – 您的设备用于连接到 Amazon IoT 的 FQDN。

  Note

  Amazon IoT 利用服务器名称指示 (SNI) TLS 扩展来应用域配置。设备在连接时必须使用此扩展名，并传递与域配置中指定的域名相同的服务器名称。
- `serverCertificateArns` — 向 ACM 注册的服务器证书链的 ARN。Amazon IoT Core 当前仅支持一个服务器证书。
- `validationCertificateArn` — 您在 ACM 中生成的用于验证自定义域所有权的公有证书的 ARN。如果您使用公开签名或 ACM 生成的服务器证书，则不需要此参数。
- `defaultAuthorizerName`（可选）— 要在端点上使用的自定义授权方的名称。
- `allowAuthorizerOverride` — 一个布尔值，指定设备是否可以通过在请求的 HTTP 标头中指定不同的授权方来覆盖默认授权方。如果为 `defaultAuthorizerName` 指定了值，则需要此值。
- `serviceType` — Amazon IoT 目前仅支持 `DATA` 服务类型。如果指定 `DATA`，则 Amazon IoT 返回端点类型为 `iot:Data-ATS` 的端点。

以下 Amazon CLI 命令为 iot.example.com 创建域配置。

```
aws iot create-domain-configuration --domain-configuration-name "myDomainConfigurationName"
 --service-type "DATA"
```

```
--domain-name "iot.example.com" --server-certificate-arns serverCertARN --validation-
certificate-arn validationCertArn
```

Note

创建您的域配置后，它可能需要长达 60 分钟才能生效，直到 Amazon IoT 提供您的自定义服务器证书。

创建 DNS 记录

注册服务器证书链并创建域配置后，请创建 DNS 记录，以便您的自定义域指向 Amazon IoT 域。此记录必须指向 `iot:Data-ATS` 类型的 Amazon IoT 终端节点。您可以使用 DescribeEndpoint API 获取测试端点。

以下 Amazon CLI 命令显示了如何获取端点。

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

获取 `iot:Data-ATS` 终端节点后，请创建从您的自定义域到此 Amazon IoT 终端节点的 `CNAME` 记录。如果您在同一个 Amazon Web Services 账户 中创建多个自定义域，请将它们的别名指定为同一个 `iot:Data-ATS` 端点。

问题排查

如果您在将设备连接到自定义域时遇到问题，请确保 Amazon IoT Core 已接受并应用您的服务器证书。您可以使用 Amazon IoT Core 控制台或 Amazon CLI 验证 Amazon IoT Core 是否已接受您的证书。

使用 Amazon IoT Core 控制台中，导航到 Settings（设置）页面，然后选择域配置名称。在 Server certificate details（服务器证书详细信息）部分中，检查状态和状态详细信息。如果证书无效，请将其在 ACM 中替换为符合上一部分中列示的证书要求 (p. 109)的证书。如果证书具有相同的 ARN，Amazon IoT Core 将自动选取并应用。

若要通过使用 Amazon CLI 检查证书状态，调用 DescribeDomainConfiguration API 并指定您的域配置名称。

Note

如果您的证书无效，Amazon IoT Core 将继续提供上一个有效的证书。

您可以使用以下 openssl 命令检查端点上正在提供哪些证书。

```
openssl s_client -connect custom-domain-name:8883 -showcerts -servername
custom-domain-name
```

管理域配置

您可以使用以下 API 管理现有配置的生命周期。

- ListDomainConfigurations
- DescribeDomainConfiguration
- UpdateDomainConfiguration
- DeleteDomainConfiguration

查看域配置

使用 ListDomainConfigurations API 返回您 Amazon Web Services 账户 中所有域配置的分页列表。您可以使用 DescribeDomainConfiguration API 查看特定域配置的详细信息。此 API 采用单个 `domainConfigurationName` 参数并返回指定配置的详细信息。

更新域配置

要更新域配置的状态或自定义授权方，请使用 UpdateDomainConfiguration API。您可以将状态设置为 ENABLED 或 DISABLED。如果您禁用域配置，连接到该域的设备将收到身份验证错误。

Note

目前，您无法更新域配置中的服务器证书。要更改域配置的证书，必须删除并重新创建该证书。

删除域配置

删除域配置之前，请使用 UpdateDomainConfiguration API 将状态设置为 DISABLED。这有助于避免意外删除终端节点。禁用域配置后，请使用 DeleteDomainConfiguration API 将其删除。

Note

您必须将 Amazon 管理的域置于 DISABLED 状态，持续 7 天，然后才可以将其删除。您可以将自定义域置于 DISABLED 状态，然后立即将其删除。

当您删除域配置后，Amazon IoT Core 不再提供与该自定义域关联的服务器证书。

在自定义域中轮换证书

您可能需要定期使用更新的证书替换服务器证书。您执行此操作的速率取决于证书的有效期。如果您使用 Amazon Certificate Manager (ACM) 生成服务器证书，您可以将证书设置为自动续订。当 ACM 续订您的证书时，Amazon IoT Core 会自动接受新证书。您无需执行任何其它操作。如果从其它源导入服务器证书，则可以通过将其重新导入到 ACM 来轮换该证书。有关重新导入证书的信息，请参阅重新导入证书。

Note

Amazon IoT Core 仅在以下条件下获取证书更新。

- 新证书具有与旧证书相同的 ARN。
- 新证书具有与旧证书相同的签名算法、公用名称或备用名称。

连接到 Amazon IoT FIPS 终端节点

Amazon IoT 提供支持美国联邦信息处理标准 (FIPS) 140-2的终端节点。符合 FIPS 的终端节点与标准 Amazon 终端节点不同。要以符合 FIPS 的方式与 Amazon IoT 进行交互，您必须将下面描述的终端节点与符合FIPS的客户端一起使用。Amazon IoT 控制台不符合 FIPS 标准。

下面几节介绍如何使用 REST API、SDK 或 Amazon CLI.访问符合 FIPS 标准 Amazon IoT 终端节点。

Amazon IoT Core - 控制面板终端节点

符合 FIPS 标准Amazon IoT Core - 控制面板支持Amazon IoT操作及其相关CLI 命令列于按服务划分 FIPS 终端节点中。在按服务划分 FIPS 终端节点中，查找Amazon IoT Core - 控制面板服务，然后查找您 Amazon Web Services 区域的终端节点。

要在访问 Amazon IoT 操作时使用符合 FIPS 标准的终端节点，请将 Amazon SDK 或 REST API与 适合您 Amazon Web Services 区域 的终端节点一起使用。

要在运行 aws iot CLI命令时使用符合 FIPS 标准的终端节点，请将具有适合您 Amazon Web Services 区域终端节点的 --endpoint 参数添加到命令中。

Amazon IoT Core - 数据层面终端节点

符合 FIPS 标准的 Amazon IoT Core - 数据层面在按服务划分 FIPS 终端节点中列出终端节点。在按服务划分 FIPS 终端节点中，查找 Amazon IoT Core - 数据层面服务，然后查找您 Amazon Web Services 区域 的终端节点。

通过使用 Amazon IoT 设备 SDK，并为 SDK 的连接功能提供终端节点，以代替帐户的原定设置Amazon IoT Core - 数据层面终端节点，您可以将符合 FIPS 标准的 Amazon Web Services 区域 终端节点用于符合 FIPS 标准的客户端。连接功能特定于 Amazon IoT 设备 SDK。有关连接功能示例，请参阅 Amazon IoT Device SDK for Python 中的连接功能。

> Note
>
> Amazon IoT 不支持符合 FIPS 标准 Amazon Web Services 账户 特定的 Amazon IoT Core-数据层面的终端节点。无法使用需要在服务器名称指示（SNI）(p. 346)中 Amazon Web Services 账户 指定终端节点的服务功能。符合 FIPS 标准Amazon IoT Core - 数据层面终端节点不能支持多帐户注册证书 (p. 268)、自定义域 (p. 109)和自定义授权人 (p. 287)。

Amazon IoT Device Management - 任务数据终端节点

符合 FIPS 标准Amazon IoT Device Management - 任务数据在按服务划分 FIPS 终端节点中列出了终端节点。在按服务划分 FIPS 终端节点中，查找 Amazon IoT Device Management - 任务数据服务，然后查找您 Amazon Web Services 区域 的终端节点。

要在运行 aws iot-jobs-data CLI 命令时使用符合 FIPS 标准的 Amazon IoT Device Management - 任务数据终端节点，请将具有适合您 Amazon Web Services 区域 的终端节点的 --endpoint 参数添加到命令中。您还可以将 REST API 与此终端节点结合使用。

通过使用 Amazon IoT 设备 SDK，并为 SDK 的连接功能提供终端节点，以代替帐户的原定设置Amazon IoT Device Management - 任务数据终端节点，您可以将符合 FIPS 标准的 Amazon Web Services 区域 终端节点用于符合 FIPS 标准的客户端。连接功能特定于 Amazon IoT 设备 SDK。有关连接功能示例，请参阅 Amazon IoT Device SDK for Python 中的连接功能。

Amazon IoT Device Management - Fleet Hub 终端节点

按服务列出的 FIPS 终端节点中列出了与Fleet Hub Amazon IoT 设备管理 CLI 命令共同使用的符合 FIPS 标准的Amazon IoT Device Management - Fleet Hub 终端节点。在按服务划分 FIPS 终端节点中，查找 Amazon IoT Device Management - Fleet Hub服务，然后查找您 Amazon Web Services 区域 的终端节点。

要在运行 aws iotfleethub CLI 命令时使用符合 FIPS 标准的 Amazon IoT Device Management - Fleet Hub 终端节点，请将具有适合您 Amazon Web Services 区域 的终端节点的 --endpoint 参数添加到命令中。您还可以将 REST API 与此终端节点结合使用。

Amazon IoT Device Management - 安全隧道终端节点

Amazon IoT 安全隧道 API 的符合 FIPS 标准的 Amazon IoT Device Management - 安全隧道终端节点与相应 CLI 命令均在 FIPS Endpoints by Service（按服务划分 FIPS 终端节点）中列出。在按服务划分 FIPS 终端节点中，查找 Amazon IoT Device Management - 安全隧道服务，然后查找您 Amazon Web Services 区域 的终端节点。

要在运行 aws iotsecuretunneling CLI 命令时使用符合 FIPS 标准的 Amazon IoT Device Management - 安全隧道终端节点，请将具有适合您 Amazon Web Services 区域 的终端节点的 --endpoint 参数添加到命令中。您还可以将 REST API 与此终端节点结合使用。

Amazon IoT 教程

Amazon IoT 教程分为两种学习路径来支持两个不同的目标。为您的目标选择最佳的学习路径。

- 您想要构建概念验证测试或演示 Amazon IoT 解决方案想法

  要使用您设备上的 Amazon IoT Device Client 展示常规 IoT任务和应用，请选择 the section called “用 Amazon IoT Device Client 构建演示” (p. 114) 学习路径。Amazon IoT Device Client 提供设备软件，您可以使用该软件应用自己的云资源来演示开发程度最低的端到端解决方案

  有关 Amazon IoT Device Client 的信息，请参阅 Amazon IoT Device Client 。
- 您想学习如何构建生产软件来部署解决方案

  要创建满足您特定要求的解决方案软件，请使用 Amazon IoT设备开发工具包，遵循 the section called “使用 Amazon IoT设备开发工具包构建解决方案” (p. 160) 学习路径。

  有关可用的 Amazon IoT设备开发工具包信息，请参阅 ??? (p. 1283)。有关 Amazon设备开发工具包的信息，请参阅在 Amazon上构建的工具。

Amazon IoT 教程学习路径选项

- 用 Amazon IoT Device Client 构建演示 (p. 114)
- 使用 Amazon IoT设备开发工具包构建解决方案 (p. 160)

用 Amazon IoT Device Client 构建演示

此学习路径中的教程将引导您使用 Amazon IoT Device Client 完成开发演示软件的步骤。Amazon IoT Device Client 提供了在 IoT 设备上运行的软件，用于全方位测试和演示在 Amazon IoT 上构建的 IoT 解决方案。

这些教程旨在促进探索和实验，这样在开发设备软件之前您可以有信心 Amazon IoT 支持您的解决方案。

您将在本教程中学到的内容：

- 如何准备 Raspberry Pi 用作 Amazon IoT 的 IoT 设备
- 如何使用设备上的 Amazon IoT Device Client 演示 Amazon IoT 功能

在这个学习路径中，您将在自己的 Raspberry Pi 上安装 Amazon IoT Device Client 并创建云中的 Amazon IoT 资源来演示 IoT 解决方案的概念。尽管此学习路径中的教程通过使用 Raspberry Pi 演示功能，也讲解了帮助您适应其他设备的目标和程序。

使用 Amazon IoT Device Client 构建演示的先决条件

本节介绍在此学习路径中的教程开始之前需要了解的内容。

要完成此学习路径中的教程，您需要：

- 一个 Amazon Web Services 账户

  您可以使用现有的 Amazon Web Services 账户（如有），但是您可能需要添加额外的角色或权限才能使用这些教程的 Amazon IoT 功能。

  如果您需要创建新 Amazon Web Services 账户，请参阅 the section called “设置您的Amazon Web Services 账户” (p. 16)。

- Raspberry Pi 或兼容的 IoT 设备

  由于因数不同，这些教程使用 Raspberry Pi，这是一种常用的演示色斑，成本相对较低。已在 Raspberry Pi 3 Model B+、Raspberry Pi 4 Model B 以及运行 Ubuntu Server 20.04 LTS (HVM) 的 Amazon EC2 实例上测试了教程。要使用 Amazon CLI 并运行命令，我们建议您使用最新版本的 Raspberry Pi OS [Raspberry Pi OS（64 位）或 OS Lite]。早期版本的操作系统可能有用，但我们还没有测试过。

  > Note
  >
  > 这些教程解释了每个步骤的目标，帮助您适应我们没有尝试过的 IoT 硬件；但是，没有具体描述如何适用于其他设备。

- 熟悉 IoT 设备的操作系统

  这些教程中的步骤假设您熟悉使用 Raspberry Pi 支持的命令行界面中的基本 Linux 命令和操作。如果您不熟悉这些操作，可能需要给自己更多的时间来完成教程。

  要完成这些教程，您应已经了解下列操作：

  - 安全地执行基本的设备操作，例如组装和连接组件、将设备连接到所需的电源以及安装和卸载存储卡。
  - 将系统软件和文件上传并下载到设备。如果您的设备不使用可移动存储设备，例如 microSD 卡，您需要知道如何连接到设备以及如何将系统软件和文件上传并下载到设备。
  - 将您的设备连接到计划使用的网络。
  - 用 SSH 终端或类似程序从另一台计算机连接到您的设备。
  - 用命令行界面在设备上创建、复制、移动、重命名和设置文件和目录的权限。
  - 在设备上安装新程序。
  - 使用 FTP 或 SCP 等工具在设备之间传输文件。

- IoT 解决方案的开发和测试环境

  教程描述了所需的软件和硬件；然而，教程假设您能够执行可能没有明确描述的操作。此类硬件和操作的示例包括：

  - 下载并存储文件的本地主机

    对于 Raspberry Pi 而言，通常是可以读写 microSD 存储卡的个人计算机或笔记本电脑。本地主机必须：

    - 连接到互联网。
    - 已经安装并配置 Amazon CLI。
    - 有支持 Amazon控制台的网络浏览器。

  - 能够将本地主机连接到设备进行通信、输入命令以及传输文件

    在 Raspberry Pi 上，通常是使用本地主机上的 SSH 和 SCP 完成的。

  - 连接到 IoT 设备上的显示器和键盘

    这些可能会有所帮助，但并不是完成教程的必需内容。

  - 您本地主机和 IoT 设备能够连接到互联网

    这可能是连接到互联网的路由器或网关的有线网络连接或无线网络连接。本地主机还必须能够连接到 Raspberry Pi。这可能需要它们在同一个局域网上。这些教程无法向您展示如何针对特定设备或设备配置进行设置，但说明了如何测试连通性。

  - 访问局域网的路由器查看连接的设备

    要完成此学习路径中的教程，您需要能够找到 IoT 设备的 IP 地址。

    在局域网上，可以通过访问设备连接到的网络路由器的管理界面来完成此操作。如果您可以在路由器中为设备分配固定 IP 地址，可以在设备每次重新启动后简化重新连接。

    如果键盘和显示器连接到设备上，ifconfig 可以显示设备的 IP 地址。

如果没有使用上述设备，需要在每次重启设备后找到一种方法来识别设备的 IP 地址。

获得所有资料之后，继续 the section called “Amazon IoT Device Client 的设备准备” (p. 116)。

此学习路径中的教程

- 教程： Amazon IoTDevice Client 的设备准备 (p. 116)
- 教程：安装和配置 Amazon IoTDevice Client (p. 124)
- 教程：演示 MQTT 消息与Amazon IoT Device Client 通信 (p. 133)
- 教程：使用 Amazon IoTDevice Client 演示远程操作（任务） (p. 145)
- 教程：运行 Amazon IoTDevice Client 教程后清理 (p. 154)

教程： Amazon IoTDevice Client 的设备准备

本教程将引导您完成 Raspberry Pi 的初始化，为此学习路径中的后续教程做好准备。

本教程的目标是安装当前版本的设备操作系统，确保您可以在开发环境中与设备进行通信。

要开始本教程：

- 请在 the section called “使用 Amazon IoT Device Client 构建演示的先决条件” (p. 114)中列出可供使用的项目。

完成本教程需要大约 90 分钟。

完成本教程后：

- 您的 IoT 设备将安装最新的操作系统。
- 您的 IoT 设备将拥有后续教程所需的额外软件。
- 您了解到设备已连接到互联网。
- 您会在设备上安装所需的证书。

完成本教程后，下一个教程会帮助您的设备准备好使用 Amazon IoTDevice Client。

本教程中的过程

- 步骤 1：安装并更新设备的操作系统 (p. 116)
- 步骤 2：在设备上安装并验证所需的软件 (p. 119)
- 步骤 3：测试您的设备并保存 Amazon CA 证书 (p. 121)

步骤 1：安装并更新设备的操作系统

本节中的过程介绍了如何初始化 Raspberry Pi 用于系统驱动器的 microSD 卡。Raspberry Pi 的 microSD 卡包含其操作系统 (OS) 软件和应用程序文件存储空间。如果您没有使用 Raspberry Pi，请按照设备的说明安装和更新设备的操作系统软件。

完成本部分后，您应能够启动 IoT 设备并从本地主机上的终端程序进行连接。

所需的设备：

- 您的本地开发和测试环境

- 可以连接到互联网的 Raspberry Pi 或您的 IoT 设备
- 至少 8 GB 或足够空间的 microSD 存储卡，用于存储操作系统和所需的软件。

  Note

  在选择 microSD 卡进行练习时，请选择一张存储量较小的卡。
  小容量 SD 卡的备份和更新速度会更快。在 Raspberry Pi 上，这些教程中使用的 microSD 卡容量不需要超过 8 GB。如果您需要更大的存储量用于特定的应用程序，在这些教程中保存的较小的图片文件可以调整存储量较大卡上的文件系统大小，使用所选卡的所有支持空间。

可选设备：

- 连接到 Raspberry Pi 的 USB 键盘
- 用于将显示器连接到 Raspberry Pi 的 HDMI 显示器和电缆

本节中的步骤：

- 将设备的操作系统加载到 microSD 卡内 (p. 117)
- 用新的操作系统启动 IoT 设备 (p. 117)
- 将您的本地主机连接到设备上 (p. 118)

将设备的操作系统加载到 microSD 卡内

此一步骤使用本地主机将设备的操作系统加载到 microSD 卡内。

Note

如果您的设备没有用于操作系统的可移动存储设备，请使用该设备的过程安装操作系统，然后继续 the section called “用新的操作系统启动 IoT 设备” (p. 117)。

要在 Raspberry Pi 上安装操作系统

1. 在本地主机上，下载并解压缩要使用的 Raspberry Pi 操作系统映像。可以从 https://www.raspberrypi.com/software/operating-systems/ 下载最新版本

   选择 Raspberry Pi OS 版本

   本教程使用 Raspberry Pi OS Lite 版本，因为它是在此学习路径中支持这些教程的最小版本。这个版本的 Raspberry Pi 操作系统只有命令行界面，没有图形用户界面。带图形用户界面的最新版本 Raspberry Pi 操作系统也可以使用这些教程；但是，本学习路径中描述的步骤仅使用命令行界面对 Raspberry Pi 执行操作。
2. 将 microSD 卡插到您的计算机上。
3. 使用 SD 卡成像工具，将解压缩的操作系统映像文件写入 microSD 卡。
4. 将 Raspberry Pi OS 映像写入 microSD 卡之后：

   a. 在命令行窗口或文件资源管理器窗口中打开 microSD 卡上的 BOOT 分区。
   b. 在 microSD 卡的 BOOT 分区中的根目录下，创建一个名为 `ssh`的空文件，没有文件扩展名也没有内容。在首次开机时会告知 Raspberry Pi 启用 SSH 通信。
5. 弹出 microSD 卡并从本地主机上安全取出卡片。

您的 microSD 卡已经准备好 the section called “用新的操作系统启动 IoT 设备” (p. 117)。

用新的操作系统启动 IoT 设备

此过程安装 microSD 卡并使用下载的操作系统首次启动 Raspberry Pi。

用新的操作系统启动 IoT 设备

1. 从设备断开电源的情况下，插入上一步中的 microSD 卡，the section called “将设备的操作系统加载到 microSD 卡内” (p. 117)，进入 Raspberry Pi。
2. 将设备连接到有线网络
3. 这些教程将使用 SSH 终端从本地主机与 Raspberry Pi 进行交互。

   如果还想直接与设备交互，您可以：

   a. 将本地主机上的终端窗口 Connect 到 Raspberry Pi 之前，将 HDMI 显示器连接到Raspberry Pi 观看 Raspberry Pi 的控制台消息。
   b. 如果想直接与 Raspberry Pi 交互，请将 USB 键盘连接。
4. 将电源连接到 Raspberry Pi 上，等待大约一分钟后才能初始化。

   如果您的显示器连接到 Raspberry Pi 上，那您可以在显示器上查看启动过程。
5. 找出设备的 IP 地址：

   - 如果您将 HDMI 显示器连接到 Raspberry Pi 上，IP 地址将出现在显示器上的消息中
   - 如果您可以访问 Raspberry Pi 连接的路由器，可以在路由器的管理界面中看到它的地址。

获得 Raspberry Pi IP 地址后，您可以进行 the section called “将您的本地主机连接到设备上” (p. 118)。

将您的本地主机连接到设备上

此过程使用本地主机上的终端程序连接到 Raspberry Pi 上并更改原定设置密码。

要将您的本地主机连接到设备上

1. 在本地主机上打开 SSH 终端程序：

   - Windows: PuTTY
   - Linux/macOS: `Terminal`

   Note

   在 Windows 上没有自动安装 PuTTY。如果计算机上没有，可能需要下载并安装它。
2. 将终端程序连接到 Raspberry Pi 的 IP 地址，然后使用原定设置凭据登录。

```
username: pi
password: raspberry
```

3. 登录 Raspberry Pi 后，请更改 `pi`用户的密码。

```
passwd
```

   按照提示更改密码。

```
Changing password for pi.
Current password: raspberry
New password: YourNewPassword
Retype new password: YourNewPassword
passwd: password updated successfully
```

在终端窗口中有 Raspberry Pi 的命令行提示符并更改密码后，可以继续 the section called “步骤 2：在设备上安装并验证所需的软件” (p. 119)。

步骤 2：在设备上安装并验证所需的软件

本节中的过程从上一节 (p. 116)继续更新您的 Raspberry Pi 操作系统，并在下一节中使用的 Raspberry Pi 上安装该软件，用于构建和安装 Amazon IoTDevice Client。

完成本节后，Raspberry Pi 将拥有最新的操作系统、本学习路径中教程所需的软件，并根据您的位置进行配置。

所需的设备：

- 上一节 (p. 116)中您的本地开发和测试环境
- 上一节 (p. 116)中您使用 的Raspberry Pi
- 上一节 (p. 116)中的 microSD 存储卡

> **Note**
>
> Raspberry Pi Model 3+ 和 Raspberry Pi Model 4 可以执行此学习路径中描述的所有命令。如果您的 IoT 设备无法编译软件或运行 Amazon Command Line Interface，您可能需要在本地主机上安装所需的编译器才能构建软件，然后将其传输到 IoT 设备。有关如何为设备安装和构建软件的更多信息，请参阅设备软件的文档。

本节中的步骤：

- 更新操作系统软件 (p. 119)
- 安装所需的应用程序和库 (p. 120)
- （可选）保存 microSD 卡映像 (p. 121)

更新操作系统软件

这一步骤会更新操作系统软件。

要更新 Raspberry Pi 上的操作系统软件

在本地主机的终端窗口中执行这些步骤。

1. 输入以下命令在您的 Raspberry Pi 上更新系统软件。

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y autoremove
```

2. 更新 Raspberry Pi 的区域设置和时区设置（可选）。

   输入此命令可以更新设备的区域设置和时区设置。

```
sudo raspi-config
```

   a. 要设置设备的区域：

      i. 在 Raspberry Pi Software Configuration Tool（raspi-config）（Raspberry Pi 软件配置工具（raspi-config））屏幕上，选择选项 5。

         5 Localisation Options Configure language and regional settings

使用 Tab键移动到 <Select>，然后按 space bar。

ii. 在本地化选项菜单中，选择选项L1。

L1 Locale Configure language and regional settings

使用 Tab键移动到 <Select>，然后按 space bar。

iii. 在区域设置选项列表中，使用箭头键滚动并选择要在 Raspberry Pi 上安装的语言环境 space bar来标记您想要的选项。

在美国，**en_US.UTF-8** 是一个很好的选择。

iv. 为设备选择语言环境后，请使用 Tab键来选择<OK>，然后按 space bar显示配置区域设置的确认页面。

b. 要设置设备的时区：

i. 在 aspi-config屏幕，选择选项5。

5 Localisation Options Configure language and regional settings

使用 Tab键移动到 <Select>，然后按 space bar。

ii. 在本地化选项菜单中，使用箭头键选择选项L2：

L2 time zone Configure time zone

使用 Tab键移动到 <Select>，然后按 space bar。

iii. 在配置 tzdata菜单中，从列表中选择您的地理区域。

使用 Tab键移动到 <OK>，然后按 space bar。

iv. 在城市列表中，使用箭头键选择时区内的城市。

使要设置时区，用 Tab键移动到 <OK>，然后按 space bar。

c. 更新完设置后，用 Tab键移动到<Finish>，然后按 space bar关闭 aspi-config应用程序。

3. 输入此命令可重启您的 Raspberry Pi。

```
sudo shutdown -r 0
```

4. 等您的 Raspberry Pi 重启。
5. 重新启动 Raspberry Pi 后，将本地主机上的终端窗口重新连接到 Raspberry Pi。

您的 Raspberry Pi 系统软件现已配置完毕，您已准备好继续 the section called “安装所需的应用程序和库” (p. 120)。

安装所需的应用程序和库

此过程将安装后续教程使用的应用程序软件和库。

如果您使用的是 Raspberry Pi，或可以在 IoT 设备上编译所需的软件，请在本地主机上的终端窗口中执行以下步骤。如果必须在本地主机上为 IoT 设备编译软件，请查看 IoT 设备的软件文档，了解有关如何在设备上执行这些步骤的信息。

在 Raspberry Pi 上安装应用程序软件和库

1. 输入此命令安装应用程序软件和库。

```
sudo apt-get -y install build-essential libssl-dev cmake unzip git python3-pip
```

2. 输入这些命令来确认安装了正确版本的软件。

```
gcc --version
cmake --version
openssl version
git --version
```

3. 确认已安装以下版本的应用程序软件：

   - gcc：9.3.0 或更高版本
   - cmake：3.10.x 或更高版本
   - OpenSSL：1.1.1 或更高版本
   - git：2.20.1 或更高版本

如果您的Raspberry Pi拥有所需应用程序软件的可接受版本，您准备好继续 the section called “（可选）保存 microSD 卡映像” (p. 121)。

（可选）保存 microSD 卡映像

在本学习路径中的整个教程中，您会遇到这些过程，将 Raspberry Pi 的 microSD 卡映像的副本保存到本地主机上的文件中。尽管鼓励这样做，但不是必需的任务。通过在建议的位置保存 microSD 卡映像，您可以跳过此学习路径中保存点之前的过程，如果发现需要重试某些内容，这可以节省时间。不定期保存 microSD 卡映像的后果是，如果 microSD 卡损坏或者不小心配置了应用程序或其设置错误，可能必须从头开始重新启动学习路径中的教程。

此时，Raspberry Pi 的 microSD 卡具有更新的操作系统和基本的应用程序软件加载。现在，您可以通过将 microSD 卡的内容保存到文件中来节省完成上述步骤所花的时间。采用拥有设备 microSD 卡映像的当前映像，您可以从这一点开始继续或重试教程或程序，无需从头开始安装和更新软件。

要将 microSD 卡映像保存到文件中

1. 输入此命令关闭 Raspberry Pi。

```
sudo shutdown -h 0
```

2. Raspberry Pi 完全关闭后，请移除电源。
3. 从 Raspberry Pi 中取出 microSD 卡。
4. 在本地主机上：
   a. 插入 microSD 卡。
   b. 使用 SD 卡成像工具，将 microSD 卡映像保存到文件中。
   c. 保存 microSD 卡映像后，从本地主机上弹出该卡。
5. 从 Raspberry Pi 断开电源后，将 microSD 卡插入 Raspberry Pi。
6. 给 Raspberry Pi 供电。
7. 等待大约一分钟后，在本地主机上重新连接已连接到 Raspberry Pi 的本地主机上的终端窗口，然后登录 Raspberry Pi。

步骤 3：测试您的设备并保存 Amazon CA 证书

本节中的过程从上一节 (p. 119)继续，来安装Amazon Command Line Interface以及用于验证您的连接的证书颁发机构证书 Amazon IoT Core。

完成本节后，您会了解 Raspberry Pi 拥有必要的系统软件来安装 Amazon IoTDevice Client 并与互联网有工作连接。

所需的设备：

- 上一节 (p. 119)中您的本地开发和测试环境
- 上一节 (p. 119)中您使用 的Raspberry Pi
- 上一节 (p. 119)中的 microSD 存储卡

本节中的步骤：

- 安装 Amazon Command Line Interface (p. 122)
- 配置 Amazon Web Services 账户凭证 (p. 122)
- 下载 Amazon Root CA 证书 (p. 123)
- （可选）保存 microSD 卡映像 (p. 124)

安装 Amazon Command Line Interface

此过程在您的 Raspberry Pi 安装Amazon CLI。

如果您使用的是 Raspberry Pi 或者可以在 IoT 设备上编译软件，请在本地主机上的终端窗口中执行以下步骤。如果必须在本地主机上为 IoT 设备编译软件，请查看 IoT 设备的软件文档，了解所需库的信息。

要在 Raspberry Pi 上安装 Amazon CLI

1. 运行以下命令下载和并按照安装 Amazon CLI。

```
export PATH=$PATH:~/.local/bin # configures the path to include the directory with the
 Amazon CLI
git clone https://github.com/aws/aws-cli.git # download the Amazon CLI code from GitHub
cd aws-cli && git checkout v2 # go to the directory with the repo and checkout version
 2
pip3 install -r requirements.txt # install the prerequisite software
```

2. 运行此命令安装 Amazon CLI。完成此命令可能最多需要 15 分钟。

```
pip3 install . # install the Amazon CLI
```

3. 运行此命令确认已安装的 Amazon CLI版本正确。

```
aws --version
```

   Amazon CLI 版本应为 2.2或更高版本。

如果 Amazon CLI显示其当前版本，您已准备就绪，可以继续 the section called “配置 Amazon Web Services 账户凭证” (p. 122)。

配置 Amazon Web Services 账户凭证

在此过程中，您将获得 Amazon Web Services 账户凭据并添加它们在 Raspberry Pi 上使用。

要在您的设备上添加 Amazon Web Services 账户凭据

1. 从获取访问密钥ID 和机密访问密钥，从您的 Amazon Web Services 账户验证设备上的 Amazon CLI。

   如果您是首次使用 AmazonIAM， https://aws.amazon.com/premiumsupport/knowledge-center/create-access-key/ 描述在 Amazon控制台中运行创建要在设备上使用的 AmazonIAM 凭据的过程。
2. 在连接到 Raspberry Pi 的本地主机上的终端窗口中。并使用设备的访问密钥 ID 和密钥访问密钥凭据：

a. 使用此命令运行 Amazon配置应用程序。

```
aws configure
```

b. 出现提示时输入您的凭据和配置信息：

```
AWS Access Key ID: your Access Key ID
Amazon Secret Access Key: your Secret Access Key
Default region name: your Amazon Web Services ## code
Default output format: json
```

3. 运行此命令测试设备对 Amazon Web Services 账户和 Amazon IoT Core端点的访问。

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

应返回您 Amazon Web Services 账户特定的Amazon IoT数据端点，例如此示例：

```
{
    "endpointAddress": "a3EXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

如果看见您 Amazon Web Services 账户特定的Amazon IoT数据端点，您的 Raspberry Pi 拥有连接和权限，可以继续 the section called “下载 Amazon Root CA 证书” (p. 123)。

Important

现在，您的 Amazon Web Services 账户凭据存储在 Raspberry Pi 的 microSD 卡上。尽管这样您会轻松使用 Amazon在这些教程中创建的软件交互，但预设情况下，它们也将保存并复制到您在此步骤之后制作的任何 microSD 卡映像中。
为了保护 Amazon Web Services 账户 凭证的安全性，在保存任何其他 microSD 卡镜像之前，请考虑通过再次运行 `aws configure` 擦除凭证，再为 Access Key ID（访问密钥 ID）和 Secret Access Key（秘密访问密钥）输入随机字符，防止您的 Amazon Web Services 账户 凭证遭盗用。如果您发现无意中保存了 Amazon Web Services 账户凭据，可以在 AmazonIAM 控制台中停用它们。

下载 Amazon Root CA 证书

此过程下载并保存 Amazon Root 证书颁发机构（CA）的证书副本。下载此证书可保存以供后续教程中使用，还可测试设备与 Amazon服务的连接。

要下载并保存 Amazon Root CA 证书

1. 运行以下命令为证书创建一个目录。

```
mkdir ~/certs
```

2. 运行此命令下载 Amazon Root CA 证书。

```
curl -o ~/certs/AmazonRootCA1.pem https://www.amazontrust.com/repository/
AmazonRootCA1.pem
```

3. 运行这些命令设置对证书目录及其文件的访问权限。

```
chmod 745 ~
chmod 700 ~/certs
chmod 644 ~/certs/AmazonRootCA1.pem
```

4. 运行此命令查看新目录中的 CA 证书文件。

```
ls -l ~/certs
```

您会看到如下条目。日期和时间会有所不同；但是，文件大小和所有其他信息应与此处显示的相同。

```
-rw-r--r-- 1 pi pi 1188 Oct 28 13:02 AmazonRootCA1.pem
```

如果文件大小不是 1188，检查 curl命令参数。您可能下载了错误的文件。

（可选）保存 microSD 卡映像

此时，Raspberry Pi 的 microSD 卡具有更新的操作系统和基本的应用程序软件加载。

要将 microSD 卡映像保存到文件中

1. 在本地主机上的终端窗口中，清除 Amazon凭证。

   a. 使用此命令运行 Amazon配置应用程序。

   ```
   aws configure
   ```

   b. 出现提示时，替换凭证。您可以离开原定设置区域名称和原定设置输出格式，因为它们是通过按Enter（输入）。

   ```
   AWS Access Key ID [****************YT2H]: XYXYXYXYX
   Amazon Secret Access Key [****************9plH]: XYXYXYXYX
   Default region name [us-west-2]:
   Default output format [json]:
   ```

2. 输入此命令关闭 Raspberry Pi。

```
sudo shutdown -h 0
```

3. Raspberry Pi 完全关闭后，卸下其电源接头。
4. 从设备中取出 microSD 卡。
5. 在本地主机上：

   a. 插入 microSD 卡。
   b. 使用 SD 卡成像工具，将 microSD 卡映像保存到文件中。
   c. 保存 microSD 卡映像后，从本地主机上弹出该卡。
6. 从 Raspberry Pi 断开电源后，将 microSD 卡插入 Raspberry Pi。
7. 对设备供电。
8. 大约一分钟后，在本地主机上重新启动终端窗口会话并登录到设备。

   请勿重新输入 Amazon Web Services 账户凭据。

重新启动并登录 Raspberry Pi 之后，您准备好继续the section called “安装并配置 Amazon IoTDevice Client” (p. 124)。

教程：安装和配置 Amazon IoTDevice Client

本教程将引导您完成 Amazon IoTDevice Client 的安装和配置，以及在此演示和其他演示中使用的 Amazon IoT资源的创建。

要开始本教程：

- 准备好您的本地主机和上一教程 (p. 116)中的 Raspberry Pi。

完成本教程需要大约 90 分钟。

完成本主题后：

- 您的 IoT 设备将准备好在其他 Amazon IoTDevice Client 演示。
- 您将在 Amazon IoT Core中配置 IoT 设备
- 您将已经下载并在设备上安装了 Amazon IoTDevice Client。
- 您将保存设备 microSD 卡的映像，可在后续教程中使用。

所需的设备：

- 上一节 (p. 121)中您的本地开发和测试环境
- 上一节 (p. 121)中您使用 的Raspberry Pi
- 上一节 (p. 121)中您使用的 Raspberry Pi microSD 记忆卡

本教程中的过程

- 步骤 1：下载并保存 Amazon IoTDevice Client (p. 125)
- （可选）保存 microSD 卡映像 (p. 126)
- 步骤 2：在 Amazon IoT中预置 Raspberry Pi (p. 127)
- 步骤 3：配置 Amazon IoTDevice Client 用于测试连接 (p. 130)

步骤 1：下载并保存 Amazon IoTDevice Client

本节中的步骤下载 Amazon IoTDevice Client，编译，然后将其安装在 Raspberry Pi 上。测试安装后，您可以保存 Raspberry Pi的 microSD卡映像，以便以后需要再次尝试教程时使用。

本节中的步骤：

- 下载并构建 Amazon IoTDevice Client (p. 125)
- 创建教程使用的目录 (p. 126)

下载并构建 Amazon IoTDevice Client

此过程安装 Raspberry Pi 中的Amazon IoT Device Client。

在连接到 Raspberry Pi 的本地主机上的终端窗口中执行这些命令。

要在 Raspberry Pi 中安装 Amazon IoTDevice Client

1. 输入以下命令可下载并构建 Raspberry Pi 中的 Amazon IoTDevice Client。

```
cd ~
git clone https://github.com/awslabs/aws-iot-device-client aws-iot-device-client
mkdir ~/aws-iot-device-client/build && cd ~/aws-iot-device-client/build
cmake ../
```

2. 运行此命令来构建 Amazon IoTDevice Client。完成此命令可能最多需要 15 分钟。

```
cmake --build . --target aws-iot-device-client
```

警告消息显示为可以忽略 Amazon IoTDevice Client 的编译文件。

这些教程已经通过 gcc上构建的 Amazon IoTDevice Client测试，gcc 上的 Raspberry Pi OS (bullseye) 2021年10月30日版本 (Raspbian 10.2.1-6+rpi1) 10.2.1 20210110，Raspberry Pi OS (buster) 2021年5月7日版本，(Raspbian 8.3.0-6+rpi1) 8.3.0 版本。

3. Amazon IoT Device Client 完成构建后，通过运行此命令进行测试。

```
./aws-iot-device-client --help
```

如果您看到的命令行帮助 Amazon IoTDevice Client，Amazon IoT Device Client 已成功构建，可供您使用。

创建教程使用的目录

此过程将在 Raspberry Pi 上创建目录，用于存储教程在此学习路径中使用的文件。

要在此学习路径中创建教程使用的目录，请执行以下操作：

1. 运行这些命令创建所需的目录。

```
mkdir ~/dc-configs
mkdir ~/policies
mkdir ~/messages
mkdir ~/certs/testconn
mkdir ~/certs/pubsub
mkdir ~/certs/jobs
```

2. 运行这些命令设置新目录的权限。

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 700 ~/certs/pubsub
chmod 700 ~/certs/jobs
```

创建这些目录并设置权限后，请继续 the section called “（可选）保存 microSD 卡映像” (p. 126)。

（可选）保存 microSD 卡映像

此时，Raspberry Pi 的 microSD 卡具有更新的操作系统和基本的应用程序以及 Amazon IoTDevice Client。

如果您想再次尝试这些练习和教程，可以跳过前面的步骤，使用此步骤保存的 microSD 卡映像写入新的 microSD 卡，然后在 the section called “步骤 2：在 Amazon IoT中预置 Raspberry Pi” (p. 127)中继续教程。

要将 microSD 卡图像保存到文件中，请执行以下操作：

在连接到 Raspberry Pi 的本地主机的终端窗口中：

1. 确认您尚未存储的 Amazon Web Services 账户凭据。

   a. 使用此命令运行 Amazon配置应用程序。

   ```
   aws configure
   ```

   b. 如果您的凭据已存储（如果出现在提示中），输入 **XYXYXYXYX**字符串在提示时，如下所示。保持原定设置区域名称和原定设置输出格式空白。

```
AWS Access Key ID [****************YXYX]: XYXYXYXYX
Amazon Secret Access Key [****************YXYX]: XYXYXYXYX
Default region name:
Default output format:
```

2. 输入此命令可关闭 Raspberry Pi。

```
sudo shutdown -h 0
```

3. Raspberry Pi 完全关闭后，卸下其电源接头。
4. 从设备中取出 microSD 卡。
5. 在本地主机上：

   a. 插入 microSD 卡。
   b. 使用 SD 卡成像工具，将 microSD 卡映像保存到文件中。
   c. 保存 microSD 卡的映像后，从本地主机上弹出该卡。

您可以继续使用这张 microSD 卡在 the section called “步骤 2：在 Amazon IoT中预置 Raspberry Pi” (p. 127)中操作。

步骤 2：在 Amazon IoT中预置 Raspberry Pi

本节中的步骤从安装了 Amazon CLI 和 Amazon IoT Device Client 的已保存 microSD 映像开始，并创建 Amazon IoT 资源和设备证书，以便在 Amazon IoT中为您的 Raspberry Pi 提供资源和设备证书。

在您的 Raspberry Pi 中安装 microSD 卡

此过程安装 microSD 卡，加载必要的软件并配置到 Raspberry Pi 中，配置 Amazon Web Services 账户这样，您就可以继续在这个学习路径中学习教程。

使用来自 the section called “（可选）保存 microSD 卡映像” (p. 126)的 microSD 卡，其中有这个学习路径中进行练习和教程的必要软件。

要在您的 Raspberry Pi 中安装 microSD 卡

1. 从 Raspberry Pi 断电后，将 microSD 卡插入 Raspberry Pi。
2. 给 Raspberry Pi 供电。
3. 大约一分钟后，在本地主机上重新启动终端窗口会话并登录 Raspberry Pi。
4. 在本地主机上、终端窗口中以及访问密钥 ID和您的 Raspberry Pi 上的密钥访问密钥凭证：

   a. 使用此命令运行 Amazon配置应用程序。

```
aws configure
```

   b. 出现提示时，输入您的 Amazon Web Services 账户凭据和配置信息：

```
AWS Access Key ID [****************YXYX]: your Access Key ID
Amazon Secret Access Key [****************YXYX]: your Secret Access Key
Default region name [us-west-2]: your Amazon Web Services ## code
Default output format [json]: json
```

恢复 Amazon Web Services 账户凭证之后，您已准备就绪继续 the section called “在 Amazon IoT Core中配置您的设备” (p. 128)。

在 Amazon IoT Core中配置您的设备

在本节的过程中创建在 Amazon IoT中预置Raspberry Pi 的 Amazon IoT资源。创建这些资源时，系统将要求您记录各种信息。在下一步骤中 Amazon IoTDevice Client 用此信息执行 Device Client 配置。

对于使用 Amazon IoT的 Raspberry Pi，必须进行预置。预置是创建和配置支持 Raspberry Pi 作为 IoT 设备所需 Amazon IoT资源的过程。

启动并重新启动 Raspberry Pi 后，将本地主机上的终端窗口连接到 Raspberry Pi 上并完成这些步骤。

本节中的步骤：

- 创建并下载设备证书文件 (p. 128)
- 创建 Amazon IoT资源 (p. 129)

创建并下载设备证书文件

此过程为此演示创建设备证书文件。

要为 Raspberry Pi 创建和下载设备证书文件

1. 在本地主机的终端窗口中，输入这些命令为您的设备创建设备证书文件。

```
mkdir ~/certs/testconn
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/testconn/device.pem.crt" \
--public-key-outfile "~/certs/testconn/public.pem.key" \
--private-key-outfile "~/certs/testconn/private.pem.key"
```

   此命令会返回类似以下内容的响应。记录 *certificateArn*值以供将来使用。

```
{
    "certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
    "certificateId":
 "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
    "certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----\n",
    "keyPair": {
        "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----\n",
        "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
    }
}
```

2. 输入以下命令设置证书目录及其文件的权限。

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 644 ~/certs/testconn/*
chmod 600 ~/certs/testconn/private.pem.key
```

3. 运行此命令可查看证书目录和文件的权限。

```
ls -l ~/certs/testconn
```

   命令的输出应与您在此处看到的内容相同，但文件日期和时间会有所不同。

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

此时，您已经在 Raspberry Pi 上安装了设备证书文件，可以继续 the section called “创建 Amazon IoT资源” (p. 129)。

创建 Amazon IoT资源

此过程通过在 Amazon IoT中创建您的设备访问 Amazon IoT功能和服务所需的资源为您的设备提供资源。

在 Amazon IoT中预置您的设备

1. 在本地主机的终端窗口中，输入以下命令获取您的 Amazon Web Services 账户设备数据端点的地址。

   ```
   aws iot describe-endpoint --endpoint-type IoT:Data-ATS
   ```

   上面步骤的命令会返回类似以下内容的响应。记录 *endpointAddress*值以供将来使用。

   ```
   {
       "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. 输入此命令为 Raspberry Pi 创建 Amazon IoT事物资源。

   ```
   aws iot create-thing --thing-name "DevCliTestThing"
   ```

   如果您的Amazon IoT事物资源创建后，命令会返回类似此响应。

   ```
   {
       "thingName": "DevCliTestThing",
       "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/DevCliTestThing",
       "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
   }
   ```

3. 在终端窗口中：

   a. 打开文本编辑器，例如 nano。

   b. 复制此 JSON 策略文档并将其粘贴到打开的文本编辑器中。

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "iot:Publish",
                   "iot:Subscribe",
                   "iot:Receive",
                   "iot:Connect"
               ],
               "Resource": [
                   "*"
               ]
           }
       ]
   ```

```
}
```

Note

本策略文档慷慨授予每个资源连接、接收、发布和订阅的权限。通常，策略仅向特定资源授予权限执行特定操作。但是，对于初始设备连通性测试，这种过于笼统和宽容的策略用于尽量减少测试期间出现访问问题的可能性。在随后的教程中，将使用范围更窄的策略文档来展示策略设计中的更好做法。

c. 将文本编辑器中的文件保存为 **~/policies/dev_cli_test_thing_policy.json**。

4. 运行此命令使用之前步骤中的策略文档创建 Amazon IoT策略。

```
aws iot create-policy \
--policy-name "DevCliTestThingPolicy" \
--policy-document "file://~/policies/dev_cli_test_thing_policy.json"
```

如果创建策略，该命令将返回类似此类的响应。

```
{
    "policyName": "DevCliTestThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/DevCliTestThingPolicy",
    "policyDocument": "{\n    \"Version\": \"2012-10-17\",\n    \"Statement\": [\n
     {\n            \"Effect\": \"Allow\",\n            \"Action\": [\n
 \"iot:Publish\",\n                \"iot:Subscribe\",\n                \"iot:Receive\",
\n                \"iot:Connect\"\n            ],\n            \"Resource\": [\n
        \"*\"\n            ]\n        }\n    ]\n}\n",
    "policyVersionId": "1"
}
```

5. 运行此命令将策略附加到设备证书。将 *certificateArn*替换为 您之前保存的 certificateArn值。

```
aws iot attach-policy \
--policy-name "DevCliTestThingPolicy" \
--target "certificateArn"
```

如果成功，该命令不返回任何内容。

6. 运行此命令将设备证书附加到 Amazon IoT事物资源。将 *certificateArn*替换为 您之前保存的 certificateArn值。

```
aws iot attach-thing-principal \
--thing-name "DevCliTestThing" \
--principal "certificateArn"
```

如果成功，该命令不返回任何内容。

在 Amazon IoT中成功配置设备后，您可以继续 the section called “步骤 3：配置 Amazon IoTDevice Client 用于测试连接” (p. 130)。

步骤 3：配置 Amazon IoTDevice Client 用于测试连接

本节中的过程配置 Amazon IoTDevice Client 发布来自 Raspberry Pi 的 MQTT 消息。

本节中的步骤：

- 创建配置文件 (p. 131)
- 打开 MQTT 测试客户端 (p. 132)

- 运行 Amazon IoTDevice Client (p. 132)

创建配置文件

此过程创建配置文件来测试 Amazon IoTDevice Client。

要创建配置文件测试 Amazon IoTDevice Client

- 在连接到 Raspberry Pi 的本地主机的终端窗口中：

  a. 输入这些命令为配置文件创建目录并设置对该目录的权限：

  ```
  mkdir ~/dc-configs
  chmod 745 ~/dc-configs
  ```

  b. 打开文本编辑器，例如 nano。

  c. 复制此 JSON 文档并将其粘贴到打开的文本编辑器中。

  ```
  {
    "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
    "cert": "~/certs/testconn/device.pem.crt",
    "key": "~/certs/testconn/private.pem.key",
    "root-ca": "~/certs/AmazonRootCA1.pem",
    "thing-name": "DevCliTestThing",
    "logging": {
      "enable-sdk-logging": true,
      "level": "DEBUG",
      "type": "STDOUT",
      "file": ""
    },
    "jobs": {
      "enabled": false,
      "handler-directory": ""
    },
    "tunneling": {
      "enabled": false
    },
    "device-defender": {
      "enabled": false,
      "interval": 300
    },
    "fleet-provisioning": {
      "enabled": false,
      "template-name": "",
      "template-parameters": "",
      "csr-file": "",
      "device-key": ""
    },
    "samples": {
      "pub-sub": {
        "enabled": true,
        "publish-topic": "test/dc/pubtopic",
        "publish-file": "",
        "subscribe-topic": "test/dc/subtopic",
        "subscribe-file": ""
      }
    },
    "config-shadow": {
      "enabled": false
    },
    "sample-shadow": {
      "enabled": false,
  ```

```
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

d. 将 ##替换为您在 Amazon Web Services 账户中找到的 the section called “在 Amazon IoT Core中配置您的设备” (p. 128)设备数据端点。

e. 将文本编辑器中的文件保存为 **~/dc-configs/dc-testconn-config.json**。

f. 运行这个命令在新文件上设置权限

```
chmod 644 ~/dc-configs/dc-testconn-config.json
```

保存文件后，您已准备就绪继续 the section called “打开 MQTT 测试客户端” (p. 132)。

打开 MQTT 测试客户端

此过程准备控制台 Amazon IoT中的 MQTT测试客户端订阅 Amazon IoTDevice Client 运行时发布的 MQTT 消息。

准备 MQTT 测试客户端订阅所有 MQTT 消息

1. 在本地主机上，在Amazon IoT控制台，选择MQTT 测试客户端。
2. 在 Subscribe to a topic（订阅主题）选项卡的 Topic filter（主题筛选条件）中输入 **#**（井号），然后选择 Subscribe（订阅）来订阅每个 MQTT 主题。
3. 在订阅标签下，确认您看见 **#**（单独英镑符号）。

让窗口的 MQTT 测试客户端 保持打开，继续 the section called “运行 Amazon IoTDevice Client” (p. 132)。

运行 Amazon IoTDevice Client

此过程运行Amazon IoT Device Client，这样发布一条 MQTT 消息接收和显示 MQTT 测试客户端。

要从 Amazon IoTDevice Client 发送 MQTT 消息

1. 在执行此步骤时，确保连接到 Raspberry Pi 的终端窗口和带 MQTT 测试客户端可见。
2. 在终端窗口中，输入这些命令以使用在 the section called “创建配置文件” (p. 131)中创建的配置文件运行 Amazon IoTDevice Client。

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-testconn-config.json
   ```

   在终端窗口中，Amazon IoT Device Client 显示信息消息以及运行时发生的任何错误。

   如果终端窗口中未显示错误，请查看MQTT 测试客户端.
3. 在 MQTT 测试客户端中，在订阅窗口中，请参阅 Hello World！ 发送到 `test/dc/pubtopic`消息主题。
4. 如果Amazon IoT Device Client 没有显示错误，您会看到 Hello World！发送到MQTT 测试客户端中的 `test/dc/pubtopic`消息，您已经证明了成功的连接。
5. 在终端窗口中，输入 **^C**(Ctrl-C) 可停止 Amazon IoTDevice Client。

在您证明了Amazon IoT Device Client 在Raspberry Pi 上正常运行之后，可以与 Amazon IoT通信，您可以继续 the section called “演示 MQTT 消息与 Amazon IoTDevice Client 通信” (p. 133)。

教程：演示 MQTT 消息与Amazon IoT Device Client 通信

本教程演示了Amazon IoT Device Client 如何订阅和发布 MQTT 消息，这些消息在 IoT 解决方案中常用。

要开始本教程：

- 将本地主机和 Raspberry Pi 配置为在上一节 (p. 124)使用。

  如果您在安装 Amazon IoTDevice Client 之后保存了 microSD 卡映像，您可以将 microSD 卡与Raspberry Pi 一起使用该映像。
- 如果您之前运行过此演示，请查看 ??? (p. 155)删除在之前运行中的所有 Amazon IoT资源，避免重复资源错误。

完成本教程需要大约 45 分钟。

完成本主题后：

- 您将展示 IoT 设备可以通过不同的方式从 Amazon IoT订阅 MQTT 消息然后将 MQTT 消息发布到 Amazon IoT。

所需的设备：

- 上一节 (p. 124)中您的本地开发和测试环境
- 上一节 (p. 124)中您使用 的Raspberry Pi
- 上一节 (p. 124)中您使用的 Raspberry Pi microSD 记忆卡

本教程中的过程

- 步骤 1：准备 Raspberry Pi 来演示 MQTT 消息通信 (p. 133)
- 步骤 2：演示使用 Amazon IoT Device Client 发布消息 (p. 138)
- 步骤 3：演示使用 Amazon IoTDevice Client 订阅消息 (p. 140)

步骤 1：准备 Raspberry Pi 来演示 MQTT 消息通信

此过程在 Amazon IoT中创建资源并在 Raspberry Pi中 演示 使用 Amazon IoTDevice Client 的 MQTT 消息通信。

本节中的步骤：

- 创建证书文件演示 MQTT 通信 (p. 133)
- 预置您的设备演示 MQTT 通信 (p. 134)
- 配置 Amazon IoTDevice Client 配置文件和 MQTT 测试客户端来演示 MQTT 通信 (p. 137)

创建证书文件演示 MQTT 通信

此过程为此演示创建设备证书文件。

要为 Raspberry Pi 创建和下载设备证书文件

1. 在本地主机的终端窗口中，输入以下命令为您的设备创建设备证书文件。

```
mkdir ~/certs/pubsub
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/pubsub/device.pem.crt" \
--public-key-outfile "~/certs/pubsub/public.pem.key" \
--private-key-outfile "~/certs/pubsub/private.pem.key"
```

此命令会返回类似以下内容的响应。保存 *certificateArn*值供稍后使用。

```
{
"certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificateId": "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----\n",
"keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. 输入以下命令设置证书目录及其文件的权限。

```
chmod 700 ~/certs/pubsub
chmod 644 ~/certs/pubsub/*
chmod 600 ~/certs/pubsub/private.pem.key
```

3. 运行此命令可查看证书目录和文件的权限。

```
ls -l ~/certs/pubsub
```

命令的输出应与您在此处看到的内容相同，但文件日期和时间会有所不同。

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

4. 输入这些命令创建日志文件的目录。

```
mkdir ~/.aws-iot-device-client
mkdir ~/.aws-iot-device-client/log
chmod 745 ~/.aws-iot-device-client/log
echo " " > ~/.aws-iot-device-client/log/aws-iot-device-client.log
echo " " > ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
chmod 600 ~/.aws-iot-device-client/log/*
```

预置您的设备演示 MQTT 通信

本节将在 Amazon IoT中创建预置 Raspberry Pi 的 Amazon IoT资源。

在 Amazon IoT中预置您的设备：

1. 在本地主机的终端窗口中，输入以下命令获取您的 Amazon Web Services 账户设备数据端点的地址。

```
aws iot describe-endpoint --endpoint-type IoT:Data-ATS
```

自从您为上一教程运行此命令以来，端点值一直没有更改。再次在此处运行命令是为了方便地查找数据端点值并将其粘贴到本教程中使用的配置文件中。

上面步骤的命令会返回类似以下内容的响应。记录 *endpointAddress*值以供将来使用。

```
{
"endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. 输入此命令为 Raspberry Pi 创建新的 Amazon IoT事物资源。

```
aws iot create-thing --thing-name "PubSubTestThing"
```

因为一个 Amazon IoT事物资源是您的设备在云中的虚拟表示，所以我们可以在 Amazon IoT中创建多个事物资源用于不同的目的。它们都可以由同一物理 IoT 设备使用来表示设备的不同方面。

这些教程一次只能使用一个事物资源来表示 Raspberry Pi。这样，在这些教程中，它们代表不同的演示，在为演示创建 Amazon IoT资源后，可以返回并使用专门为每个演示创建的资源重复演示。

如果您的 Amazon IoT事物资源创建后，命令会返回类似此响应。

```
{
"thingName": "PubSubTestThing",
"thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/PubSubTestThing",
"thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. 在终端窗口中：

   a. 打开文本编辑器，例如 nano。

   b. 复制此 JSON 文档并将其粘贴到打开的文本编辑器中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
```

```
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
      ]
    }
  ]
}
```

c. 在编辑器中，在策略文档的每个 Resource部分中，将 *us-west-2:57EXAMPLE833* 替换为您的 Amazon Web Services 区域、冒号字符 (:) 和 12 位 Amazon Web Services 账户数字。

d. 将文本编辑器中的文件保存为 **~/policies/pubsub_test_thing_policy.json**。

4. 运行此命令使用之前步骤中的策略文档创建 Amazon IoT策略。

```
aws iot create-policy \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_test_thing_policy.json"
```

如果创建策略，该命令将返回类似此类的响应。

```
{
    "policyName": "PubSubTestThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
    "policyDocument": "{\n\"Version\": \"2012-10-17\",\n\"Statement\": [\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\": [\n
\"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},\n{\n\"Effect\":
 \"Allow\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Receive\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
    "policyVersionId": "1"
}
```

5. 运行此命令将策略附加到设备证书。将 *certificateArn*替换为您之前在本部分保存的 certificateArn值。

```
aws iot attach-policy \
--policy-name "PubSubTestThingPolicy" \
--target "certificateArn"
```

如果成功，该命令不返回任何内容。

6. 运行此命令将设备证书附加到 Amazon IoT事物资源。将 *certificateArn*替换为您之前在本部分保存的 certificateArn值。

```
aws iot attach-thing-principal \
--thing-name "PubSubTestThing" \
--principal "certificateArn"
```

如果成功，该命令不返回任何内容。

在 Amazon IoT中成功配置设备后，您准备好继续 the section called “配置 Amazon IoTDevice Client 配置文件和 MQTT 测试客户端来演示 MQTT 通信” (p. 137)。

配置 Amazon IoTDevice Client 配置文件和 MQTT 测试客户端来演示 MQTT 通信

此过程将创建一个配置文件来测试 Amazon IoTDevice Client 。

要创建配置文件测试 Amazon IoTDevice Client

1. 在连接到 Raspberry Pi 的本地主机的终端窗口中：
   a. 打开文本编辑器，例如 nano。
   b. 复制此 JSON 文档并将其粘贴到打开的文本编辑器中。

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/pubsub/device.pem.crt",
  "key": "~/certs/pubsub/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "PubSubTestThing",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
    "file": ""
  },
  "jobs": {
    "enabled": false,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. 将 ##替换为您在 Amazon Web Services 账户中找到的 the section called “在 Amazon IoT Core中配置您的设备” (p. 128)设备数据端点。

d. 将文本编辑器中的文件保存为 **~/dc-configs/dc-pubsub-config.json**。

e. 运行这个命令在新文件上设置权限

```
chmod 644 ~/dc-configs/dc-pubsub-config.json
```

2. 要准备 MQTT 测试客户端订阅所有 MQTT 消息

a. 在本地主机上，在Amazon IoT控制台，选择MQTT 测试客户端。

b. 在订阅主题选项卡中，在主题筛选条件中，输入**#**（一个英镑符号），然后选择订阅。

c. 在订阅标签下，确认您看见 **#**（单独英镑符号）。

继续此教程时，让窗口的 MQTT 测试客户端 保持打开。

保存文件并配置 MQTT 测试客户端后，您便可以继续 the section called “步骤 2：演示使用 Amazon IoT Device Client 发布消息” (p. 138)。

步骤 2：演示使用 Amazon IoT Device Client 发布消息

本节中的过程演示了 Amazon IoTDevice Client 如何发送原定设置和自定义 MQTT 消息。

在上一步中为这些练习创建的策略中的以下策略声明授予 Raspberry Pi 执行以下操作的权限：

- **iot:Connect**

  为客户提供 PubSubTestThing命名，您的 Raspberry Pi 运行 Amazon IoTDevice Client 连接。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Connect"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
  ]
}
```

- **iot:Publish**

  授予 Raspberry Pi 用 `test/dc/pubtopic`MQTT 主题发布消息的全效.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Publish"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
  ]
}
```

  `iot:Publish` 操作提供发布到资源数组中列出的 MQTT 主题权限。那些消息的内容不受策略声明的控制。

使用 Amazon IoT Device Client发布原定设置消息

此过程运行Amazon IoT Device Client，这样发布一条原定设置 MQTT 消息接收和显示 MQTT 测试客户端。

要从 Amazon IoTDevice Client 发送原定设置 MQTT 消息

1. 执行此过程时，请确保连接到 Raspberry Pi 的本地主机上的终端窗口和使用 MQTT测试客户端的窗口都可见。
2. 在终端窗口中，输入这些命令以使用在 Amazon IoT中创建的配置文件运行 the section called “创建配置文件” (p. 131)Device Client。

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-config.json
   ```

   在终端窗口中，Amazon IoT Device Client 显示信息消息以及运行时发生的任何错误。

   如果终端窗口中未显示错误，请查看MQTT 测试客户端.
3. 在 MQTT test client（MQTT 测试客户端）的 Subscriptions（订阅）窗口中，参阅发送到 `test/dc/pubtopic` 消息主题的 Hello World! 消息。
4. 如果Amazon IoT Device Client 没有显示错误，您会看到 Hello World！发送到MQTT 测试客户端中的 `test/dc/pubtopic`消息，您已经证明了成功的连接。
5. 在终端窗口中，输入 **^C**(Ctrl-C) 可停止 Amazon IoTDevice Client。

在您证明了 Amazon IoTDevice Client 发布了原定设置的 MQTT 消息后，您可以继续 the section called “使用 Amazon IoTDevice Client 发布指标消息” (p. 139)。

使用 Amazon IoTDevice Client 发布指标消息

本节中的过程创建一条自定义 MQTT 消息，然后运行 Amazon IoTDevice Client，以便将自定义 MQTT 消息发布一次，以便 MQTT 测试客户端接收和显示。

为Amazon IoT Device Client 创建自定义 MMQTT 消息

在连接到 Raspberry Pi 的本地主机上的终端窗口中执行以下步骤。

为 Amazon IoTDevice Client 创建自定义消息

1. 在终端窗口中，打开文本编辑器，例如nano。
2. 在文本编辑器中，复制并粘贴以下 JSON 文档。这是Amazon IoT Device Client 发布的 MQTT 消息负载。

   ```
   {
     "temperature": 28,
     "humidity": 80,
     "barometer": 1013,
     "wind": {
       "velocity": 22,
       "bearing": 255
     }
   }
   ```
3. 将文本编辑器的内容另存为 **~/messages/sample-ws-message.json**。
4. 输入以下命令设置刚才创建的消息文件的权限。

   ```
   chmod 600 ~/messages/*
   ```

为 Amazon IoTDevice Client 创建用于发送自定义消息的配置文件

1. 在终端窗口中，在文本编辑器中，例如 nano，打开现有 Amazon IoTDevice Client 配置文件：**~/dc-configs/dc-pubsub-config.json**。
2. 编辑 samples对象如下所示。此文件的其他部分不需要更改。

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
      "subscribe-topic": "test/dc/subtopic",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

3. 将文本编辑器的内容另存为 **~/dc-configs/dc-pubsub-custom-config.json**。
4. 运行这个命令在新文件上设置权限

```
chmod 644 ~/dc-configs/dc-pubsub-custom-config.json
```

通过使用Amazon IoT Device Client 发布自定义 MQTT 消息

此更改仅影响 MQTT 消息负载的内容，因此当前策略将继续工作。但是，如果MQTT 主题（由 ~/dc-configs/dc-pubsub-custom-config.json红的 publish-topic值定义）已更改，还需要修改策略语句 iot::Publish，允许 Raspberry Pi 发布到新的 MQTT 主题。

要从 Amazon IoTDevice Client 发送 MQTT 消息

1. 在执行此过程时，确保终端窗口和窗口都有 MQTT 测试客户端可见。此外，请确保 MQTT 测试客户端仍订阅为#主题筛选条件。如果没有订阅，请再次订阅#筛选条件主题。
2. 在终端窗口中，输入这些命令以使用在 Amazon IoT中创建的配置文件运行 the section called “创建配置文件” (p. 131)Device Client。

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
```

   在终端窗口中，Amazon IoT Device Client 显示信息消息以及运行时发生的任何错误。

   如果终端窗口中没有显示错误，请查看 MQTT 测试客户端。
3. 在 MQTT 测试客户端中，在订阅窗口中，请参阅发送到 test/dc/pubtopic消息主题的自定义消息负载。
4. 如果 Amazon IoTDevice Client 未显示任何错误，并且您在 MQTT测试客户端中看到发布到 test/dc/pubtopic消息的自定义消息负载，则表示您已成功发布自定义消息。
5. 在终端窗口中，输入**^C**(Ctrl-C) 可停止Amazon IoT Device Client 。

在您演示了 Amazon IoTDevice Client 发布了自定义消息负载之后，您可以继续 the section called “步骤 3：演示使用 Amazon IoTDevice Client 订阅消息” (p. 140)。

步骤 3：演示使用 Amazon IoTDevice Client 订阅消息

在本节中，您将演示两种类型的消息订阅：

- 单独订阅主题
- 通配符主题订阅

为这些练习创建的策略中的策略语句授予 Raspberry Pi 执行这些操作的权限：

- **`iot:Receive`**

  为 Amazon IoTDevice Client 提供接收与 `Resource`对象中命名主题匹配的 MQTT 主题的权限。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
  ]
}
```

- **`iot:Subscribe`**

  为 Amazon IoTDevice Client 提供订阅与 `Resource`对象中命名相匹配的MQTT主题筛选条件的权限。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
  ]
}
```

订阅单个 MQTT 消息主题

此过程演示 Amazon IoTDevice Client 如何订阅和记录 MQTT 消息。

在连接到 Raspberry Pi 的本地主机上的终端窗口中，列出 **`~/dc-configs/dc-pubsub-custom-config.json`**的内容或者在文本编辑器中打开该文件查看内容。定位 `samples`对象，应如下所示。

```
"samples": {
  "pub-sub": {
    "enabled": true,
    "publish-topic": "test/dc/pubtopic",
    "publish-file": "~/messages/sample-ws-message.json",
    "subscribe-topic": "test/dc/subtopic",
    "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

注意 `subscribe-topic`值是Amazon IoT Device Client 将在运行时订阅的 MQTT 主题。Amazon IoT Device Client 将从此订阅中收到的消息有效负载写入`subscribe-file`值。

要从 Amazon IoT Device Client 订阅 MQTT 消息主题

1. 在执行此过程时，确保终端窗口和窗口都有 MQTT 测试客户端可见。此外，请确保 MQTT 测试客户端仍订阅为#主题筛选条件。如果没有订阅，请再次订阅#筛选条件主题。
2. 在终端窗口中，输入这些命令以使用在 Amazon IoT中创建的配置文件运行 the section called “创建配置文件” (p. 131)Device Client。

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
```

在终端窗口中，Amazon IoT Device Client 显示信息消息以及运行时发生的任何错误。

如果终端窗口中未显示错误，请在 Amazon IoT控制台继续。

3. 在 Amazon IoT控制台中的 MQTT 测试客户端中，选择发布到主题选项卡。
4. 在 主题名称中，输入 **test/dc/subtopic**。
5. 在消息负载中，查看消息内容。
6. 选择 Publish (发布) 发布 MQTT 消息。
7. 在终端窗口中，注意 Amazon IoTDevice Client 的接收消息条目。

```
2021-11-10T16:02:20.890Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
 subscribe topic, size: 45 bytes
```

8. 看见显示已收到消息接收消息的条目之后，输入**^C**(Ctrl-C) 可停止 Amazon IoTDevice Client。
9. 输入此命令可查看消息日志文件的末尾并查看您从 MQTT 测试客户端发布的信息

```
tail ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

通过查看日志文件中的消息，您已经演示了 Amazon IoTDevice Client 收到了您从 MQTT 测试客户端发布的消息。

使用通配符订阅多个 MQTT 消息主题

这些过程演示了 Amazon IoTDevice Client 如何使用通配符订阅和记录 MQTT 消息。要做到这一点，您将：

1. 更新 Amazon IoTDevice Client 用于订阅 MQTT 主题的主题筛选条件。
2. 更新设备使用的策略允许新订阅。
3. 运行 Amazon IoTDevice Client 并从 MQTT 测试控制台发布消息。

使用通配符 MQTT 主题筛选条件创建配置文件订阅多个 MQTT 消息主题

1. 在连接到 Raspberry Pi 的本地主机上的终端窗口中，打开 **~/dc-configs/dc-pubsub-custom-config.json**进行编辑并定位 `samples`对象。
2. 在文本编辑器中，定位 `samples`对象并更新 `subscribe-topic`值如下所示。

```
  "samples": {
    "pub-sub": {
      "enabled": true,
      "publish-topic": "test/dc/pubtopic",
      "publish-file": "~/messages/sample-ws-message.json",
      "subscribe-topic": "test/dc/#",
      "subscribe-file": "~/.aws-iot-device-client/log/pubsub_rx_msgs.log"
```

新 `subscribe-topic`值是 MQTT主题筛选条件 (p. 95)，其末尾带有 MQTT 通配符。这描述了以 `test/dc/`开始的所有 MQTT 主题的订阅。Amazon IoT Device Client 将从该订阅接收的消息有效负载写入名为`subscribe-file` 的文件。

3. 将修改后的配置文件另存为**~/dc-configs/dc-pubsub-wild-config.json**，然后退出编辑器。

修改 Raspberry Pi 使用的策略允许订阅和接收多个 MQTT 消息主题

1. 在连接到 Raspberry Pi 的本地主机上的终端窗口中，在您最喜欢的文本编辑器中打开 **~/policies/pubsub_test_thing_policy.json**进行编辑，然后定位 `iot::Subscribe`和 `iot::Receive`文件中的策略语句。

2. 在iot::Subscribe策略语句中更新 Resource 对象中的字符串用 *替换 subtopic，如下所示。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*"
  ]
}
```

Note

MQTT 主题筛选条件通配符 (p. 95) 是 +（加号）和 #（磅符号）。结尾带有 #的订阅请求订阅以字符前面的字符串开头的 #所有主题（例如，在本用例中为 test/dc/）。
但是，授权此订阅的策略语句中的资源值必须在主题筛选条件ARN中使用 *（星号）代替 #（磅号）。这是因为策略处理器使用的通配符不同于 MQTT 使用的通配符。
有关在策略中为主题和主题筛选条件使用通配符的详细信息，请参阅 在 MQTT 和 Amazon IoT Core 策略中使用通配符 (p. 317)。

3. 在 iot::Receive策略语句中更新 Resource 对象中的字符串用 subtopic替换 *，如下所示。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*"
  ]
}
```

4. 将更新后的策略文档另存为 **~/policies/pubsub_wild_test_thing_policy.json**，然后退出编辑器。

5. 输入此命令可更新本教程的策略使用新的资源定义。

```
aws iot create-policy-version \
--set-as-default \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_wild_test_thing_policy.json"
```

如果命令成功，将返回类似于此类的响应。请注意，policyVersionId 现在是 2，表示这是此策略的第二个版本。

如果成功更新策略，可以继续下一个过程。

```
{
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/PubSubTestThingPolicy",
    "policyDocument": "{\n  \"Version\": \"2012-10-17\",\n  \"Statement\": [\n    {\n
    \"Effect\": \"Allow\",\n      \"Action\": [\n        \"iot:Connect\"\n      ],\n
   \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing
\"\n      ]\n    },\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n
      \"iot:Publish\"\n      ],\n      \"Resource\": [\n        \"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n      ]\n    },\n    {\n      \"Effect
\": \"Allow\",\n      \"Action\": [\n        \"iot:Subscribe\"\n      ],\n
 \"Resource\": [\n        \"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/
*\"\n      ]\n    },\n    {\n      \"Effect\": \"Allow\",\n      \"Action\": [\n
       \"iot:Receive\"\n      ],\n      \"Resource\": [\n        \"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/*\"\n      ]\n    }\n  ]\n}\n",
```

```
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```

如果您收到有太多策略版本无法保存新版本的错误，请输入此命令列出策略的当前版本。查看此命令返回的列表查找可以删除的策略版本。

```
aws iot list-policy-versions --policy-name "PubSubTestThingPolicy"
```

输入此命令删除不再需要的版本。请注意，您无法删除原定设置策略版本。原定设置策略版本为具有 `true`的 `isDefaultVersion`值.

```
aws iot delete-policy-version \
--policy-name "PubSubTestThingPolicy" \
--policy-version-id policyId
```

删除策略版本后，请重试此步骤。

使用更新的配置文件和策略，您可以使用Amazon IoT Device Client 演示通配符订阅。

为了演示 Amazon IoTDevice Client 如何订阅并接收多个 MQTT 消息主题

1. 在 MQTT 测试客户端，检查订阅。如果已在 **#**主题筛选条件中订阅了 MQTT 测试客户端，请继续执行下一步。如果没有订阅，进入 MQTT test client（MQTT 测试客户端）的 Subscribe to a topic（订阅主题）选项卡，在 Topic filter（主题筛选条件）中输入 **#**（井号），然后选择 Subscribe（订阅）进行订阅。
2. 在连接到 Raspberry Pi 的本地主机上的终端窗口中，输入这些命令启动 Amazon IoTDevice Client。

   ```
   cd ~/aws-iot-device-client/build
   ./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-wild-config.json
   ```
3. 在观看在本地主机上终端窗口中的 Amazon IoTDevice Client 输出，返回 MQTT 测试客户端。在发布主题选项卡中，主题名称，输入 **test/dc/subtopic**，然后选择发布。
4. 在终端窗口中，通过查找以下消息确认已收到消息，例如：

   ```
   2021-11-10T16:34:20.101Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
    subscribe topic, size: 76 bytes
   ```
5. 在观看在本地主机的终端窗口中Amazon IoT Device Client 输出，返回 MQTT 测试客户端。在发布主题选项卡中，主题名称，输入 **test/dc/subtopic2**，然后选择发布。
6. 在终端窗口中，通过查找以下消息确认已收到消息，例如：

   ```
   2021-11-10T16:34:32.078Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
    subscribe topic, size: 77 bytes
   ```
7. 看到确认收到两条消息的消息后，请输入 **^C**(Ctrl-C) 可停止 Amazon IoTDevice Client。
8. 输入此命令可查看消息日志文件的末尾并查看您从 MQTT 测试客户端发布的信息

   ```
   tail -n 20 ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
   ```

   Note

   该日志文件仅包含消息有效负载。消息主题不会记录在收到的消息日志文件中。

> 您可能还会在接收到的日志中看到 Amazon IoTDevice Client 发布的消息。这是因为通配符主题筛选条件包含该消息主题，有时候，在发布的消息发送给订阅者之前，消息代理可以处理订阅请求。

日志文件中的条目表明已收到消息。您可以使用其他主题名称重复此过程。主题名称以 `test/dc/`开头的所有消息应该接收并记录。主题名称以任何其他文本开头的消息将被忽略。

在演示Amazon IoT Device Client 如何发布和订阅 MQTT 消息，继续 教程：使用 Amazon IoTDevice Client 演示远程操作（任务）(p. 145)。

教程：使用 Amazon IoTDevice Client 演示远程操作（任务）

在这些教程中，您将配置任务并将其部署到 Raspberry Pi，演示如何向 IoT 设备发送远程操作。

要开始本教程：

- 将本地主机和 Raspberry Pi 配置为在上一节 (p. 133)使用。
- 如果您没有完成上一节中的教程，可以使用 Raspberry Pi 和 microSD 卡来尝试本教程，该卡具有在（可选）保存 microSD 卡映像 (p. 126)中安装 Amazon IoTDevice Client 后保存的图像。
- 如果您之前运行过此演示，请查看??? (p. 155)删除在之前运行中的所有 Amazon IoT资源，避免重复资源错误。

完成本教程需要大约 45 分钟。

完成本主题后：

- 您会演示 IoT 设备可以使用的不同方式使用 Amazon IoT Core运行由 Amazon IoT管理的远程操作。

所需的设备：

- 您在上一节 (p. 124)中测试的本地开发和测试环境
- 您在上一节 (p. 124)中测试的 Raspberry Pi
- 您在上一节 (p. 124)中测试的来自Raspberry Pi 的 microSD 存储卡

本教程中的过程

- 步骤 1：准备 Raspberry Pi 运行任务 (p. 145)
- 步骤 2：在 Amazon IoT中创建并运行任务 (p. 151)

步骤 1：准备 Raspberry Pi 运行任务

本节中的步骤介绍了如何使用Amazon IoT Device Client 准备 Raspberry Pi 运行任务。

> **Note**
>
> 这些过程是特定于设备的。如果要同时对多个设备执行本节中的步骤，每个设备将需要自己的策略和唯一的、特定于设备的证书和设备名称。要为每台设备提供独特的资源，请对每台设备执行一次此过程，同时按照过程中所述更改特定于设备的元素。

本教程中的过程

- 提供您的 Raspberry Pi 来演示任务 (p. 146)

- 配置 Amazon IoTDevice Client 运行任务代理 (p. 149)

提供您的 Raspberry Pi 来演示任务

本节中的过程通过创建 Amazon IoT资源和设备证书在 Amazon IoT中预调配 Raspberry Pi。

创建并下载设备证书文件演示 Amazon IoT任务

此过程为此演示创建设备证书文件。

如果准备了多台设备，必须在每台设备上执行此过程。

要为 Raspberry Pi 创建和下载设备证书文件，请执行以下操作：

在连接到 Raspberry Pi 的本地主机上的终端窗口中，输入以下命令。

1. 输入以下命令为您的设备创建设备证书文件。

```
aws iot create-keys-and-certificate \
--set-as-active \
--certificate-pem-outfile "~/certs/jobs/device.pem.crt" \
--public-key-outfile "~/certs/jobs/public.pem.key" \
--private-key-outfile "~/certs/jobs/private.pem.key"
```

   此命令会返回类似以下内容的响应。保存 *certificateArn*值供稍后使用。

```
{
"certificateArn": "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/76e7e4edb3e52f52334be2f387a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificateId": "76e7e4edb3e52f5233EXAMPLE7a06145b2aa4c7fcd810f3aea2d92abc227d269",
"certificatePem": "-----BEGIN CERTIFICATE-----
\nMIIDWTCCAkGgAwIBAgI_SHORTENED_FOR_EXAMPLE_Lgn4jfgtS\n-----END CERTIFICATE-----\n",
"keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BA_SHORTENED_FOR_EXAMPLE_ImwIDAQAB\n-----END PUBLIC KEY-----\n",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
}
}
```

2. 输入以下命令设置证书目录及其文件的权限。

```
chmod 700 ~/certs/jobs
chmod 644 ~/certs/jobs/*
chmod 600 ~/certs/jobs/private.pem.key
```

3. 运行此命令可查看证书目录和文件的权限。

```
ls -l ~/certs/jobs
```

   命令的输出应与您在此处看到的内容相同，但文件日期和时间会有所不同。

```
-rw-r--r-- 1 pi pi 1220 Oct 28 13:02 device.pem.crt
-rw------- 1 pi pi 1675 Oct 28 13:02 private.pem.key
-rw-r--r-- 1 pi pi  451 Oct 28 13:02 public.pem.key
```

将设备证书文件下载到 Raspberry Pi 之后，您准备好继续 the section called “提供您的 Raspberry Pi 来演示任务” (p. 146)。

创建要演示 Amazon IoT任务的 Amazon IoT资源

创建此设备的 Amazon IoT资源。

如果准备了多台设备，必须在每台设备上执行此过程。

在 Amazon IoT中预调配您的设备：

在连接到 Raspberry Pi 的本地主机的终端窗口中：

1. 输入以下命令获取您的设备数据端点的地址：Amazon Web Services 账户。

   ```
   aws iot describe-endpoint --endpoint-type IoT:Data-ATS
   ```

   自上次运行此命令以来，端点值未发生更改。再次在此处运行该命令可以轻松查找数据端点值并将其粘贴到本教程中使用的配置文件中。

   此 describe-endpoint命令会返回类似以下内容的响应：记录 *endpointAddress*值以供将来使用。

   ```
   {
   "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. 将 *uniqueThingName* 替换为您的设备提供唯一的名称。如果要使用多台设备执行本教程，请为每台设备指定自己的名称。例如，**TestDevice01**、**TestDevice02** 等等。

   输入此命令为 Raspberry Pi 创建新的 Amazon IoT事物资源。

   ```
   aws iot create-thing --thing-name "uniqueThingName"
   ```

   因为一个 Amazon IoT事物资源是您的设备在云中的虚拟表示，所以我们可以在 Amazon IoT中创建多个事物资源用于不同的目的。它们都可以由同一物理 IoT 设备使用来表示设备的不同方面。

   > **Note**
   >
   > 当您想要保护多台设备的策略时，可以使用 ${iot:Thing.*ThingName*} 而不是静态事物名称 *uniqueThingName*。

   这些教程每台设备一次只能使用一件事资源。这样，在这些教程中，它们代表不同的演示，以便在为演示创建 Amazon IoT资源后，可以使用专门为每个演示创建的资源返回并重复演示。

   如果您的 Amazon IoT事物资源创建后，命令会返回类似此响应。记录 *thingArn*值以供稍后创建要在此设备上运行的任务时使用。

   ```
   {
   "thingName": "uniqueThingName",
   "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/uniqueThingName",
   "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
   }
   ```

3. 在终端窗口中：

   a. 打开文本编辑器，例如 nano。
   b. 复制此 JSON 文档并将其粘贴到打开的文本编辑器中。

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

```
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/uniqueThingName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/job/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/jobExecution/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/jobs/
*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/jobExecution/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/
things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/jobs/
*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeJobExecution",
        "iot:GetPendingJobExecutions",
        "iot:StartNextPendingJobExecution",
        "iot:UpdateJobExecution"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName"
      ]
    }
  ]
}
```

c. 在编辑器中，在策略文档的每个 Resource部分中，将 *us-west-2:57EXAMPLE833* 替换为您的 Amazon Web Services 区域、冒号字符 (:) 和 12 位 Amazon Web Services 账户数字。

d. 在编辑器中，在每个策略语句中，将 *uniqueThingName* 替换为您提供给此事物资源的事物名称。

e. 将文本编辑器中的文件保存为 **~/policies/jobs_test_thing_policy.json**。

如果要为多台设备运行此过程，请将文件保存为每台设备上的此文件名。

4. 将 *uniqueThingName* 替换为设备的名称，然后运行此命令创建为该设备定制的 Amazon IoT策略。

```
aws iot create-policy \
--policy-name "JobTestPolicyFor<uniqueThingName>" \
--policy-document "file://~/policies/jobs_test_thing_policy.json"
```

如果创建策略，该命令将返回类似此类的响应。

```
{
    "policyName": "JobTestPolicyFor<uniqueThingName>",
    "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
JobTestPolicyFor<uniqueThingName>",
    "policyDocument": "{\n\"Version\": \"2012-10-17\",\n\"Statement\": [\n{\n
\"Effect\": \"Allow\",\n\"Action\": [\n\"iot:Connect\"\n],\n\"Resource\": [\n
\"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n},\n{\n\"Effect\":
 \"Allow\",\n\"Action\": [\n\"iot:Publish\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Subscribe\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n},\n{\n\"Effect\": \"Allow
\",\n\"Action\": [\n\"iot:Receive\"\n],\n\"Resource\": [\n\"arn:aws:iot:us-
west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
    "policyVersionId": "1"
```

5. 将 *uniqueThingName* 替换为设备 和 *certificateArn*的事物名称以及您在本节之前为此设备保存的 certificateArn值，然后运行此命令将策略附加到设备证书。

```
aws iot attach-policy \
--policy-name "JobTestPolicyFor<uniqueThingName>" \
--target "<certificateArn>"
```

如果成功，该命令不返回任何内容。

6. 将 *uniqueThingName* 替换为设备的事物名称，将 *certificateArn*替换为本节之前保存的 certificateArn值，然后运行此命令将设备证书附加到 Amazon IoT事物资源。

```
aws iot attach-thing-principal \
--thing-name "<uniqueThingName>" \
--principal "<certificateArn>"
```

如果成功，该命令不返回任何内容。

成功预调配 Raspberry Pi 之后，您可以在测试中为另一个 Raspberry Pi 重复此部分，或者，如果所有设备均已预调配，请继续 the section called “配置 Amazon IoTDevice Client 运行任务代理” (p. 149)。

配置 Amazon IoTDevice Client 运行任务代理

此过程为 Amazon IoTDevice Client 创建配置文件运行任务代理：

注意：如果准备多台设备，必须在每台设备上执行此过程。

要创建配置文件测试 Amazon IoTDevice Client ：

1. 在连接到 Raspberry Pi 的本地主机的终端窗口中：

   a. 打开文本编辑器，例如 nano。
   b. 复制此 JSON 文档并将其粘贴到打开的文本编辑器中。

```
{
  "endpoint": "a3qEXAMPLEaffp-ats.iot.us-west-2.amazonaws.com",
  "cert": "~/certs/jobs/device.pem.crt",
  "key": "~/certs/jobs/private.pem.key",
  "root-ca": "~/certs/AmazonRootCA1.pem",
  "thing-name": "uniqueThingName",
  "logging": {
    "enable-sdk-logging": true,
    "level": "DEBUG",
    "type": "STDOUT",
    "file": ""
  },
  "jobs": {
    "enabled": true,
    "handler-directory": ""
  },
  "tunneling": {
    "enabled": false
  },
  "device-defender": {
    "enabled": false,
    "interval": 300
  },
  "fleet-provisioning": {
    "enabled": false,
    "template-name": "",
    "template-parameters": "",
    "csr-file": "",
    "device-key": ""
  },
  "samples": {
    "pub-sub": {
      "enabled": false,
      "publish-topic": "",
      "publish-file": "",
      "subscribe-topic": "",
      "subscribe-file": ""
    }
  },
  "config-shadow": {
    "enabled": false
  },
  "sample-shadow": {
    "enabled": false,
    "shadow-name": "",
    "shadow-input-file": "",
    "shadow-output-file": ""
  }
}
```

c. 将*##*替换为您在 Amazon Web Services 账户中找到的 the section called “在 Amazon IoT Core中配置您的设备” (p. 128)设备数据端点值。

d. 将 *uniqueThingName* 替换为此设备使用的事物名称。

e. 将文本编辑器中的文件保存为 **~/dc-configs/dc-jobs-config.json**。

2. 运行此命令设置新配置文件的文件权限。

```
chmod 644 ~/dc-configs/dc-jobs-config.json
```

此测试没有使用 MQTT 测试客户端。虽然设备将与 Amazon IoT交换任务相关的 MQTT消息，但任务进度消息仅与运行任务的设备交换。由于任务进度消息仅与运行任务的设备交换，因此您无法从其他设备订阅（例如 Amazon IoT控制台。

保存配置文件后，您已准备继续 the section called “步骤 2：在 Amazon IoT中创建并运行任务” (p. 151)。

步骤 2：在 Amazon IoT中创建并运行任务

本节中的过程创建一个任务文档和 Amazon IoT任务资源。创建任务资源后，Amazon IoT 将任务文档发送到指定的任务目标，任务代理将任务文档应用于设备或客户端。

本节中的过程

- 创建并存储任务的任务文档 (p. 151)
- 为一台 IoT 设备在 Amazon IoT中运行任务 (p. 152)

创建并存储任务的任务文档

此过程将创建一个简单的任务文档包括在 Amazon IoT任务资源内。这份任务文档显示“Hello world!” 在任务目标上。

要创建和存储任务文档：

1. 选择要将任务文档保存到的 Amazon S3 存储桶。如果您没有现有 Amazon S3 存储桶可用，需要创建一个。有关如何创建 Amazon S3 存储桶的信息，请参阅 Amazon S3 入门中的主题。
2. 为此任务创建并保存任务文档

   a. 在本地主机上，打开文本编辑器。

   b. 复制此文本并粘贴到编辑器中。

   ```
   {
       "operation": "echo",
       "args": ["Hello world!"]
   }
   ```

   c. 在本地主机上，将编辑器内容保存到名为 **hello-world-job.json**的文件中。

   d. 确认文件已正确保存。一些文本编辑器会自动追加 `.txt`用户保存文本文件时的文件名。如果您的编辑器附加 `.txt`在文件名中，在继续之前更正文件名。

3. 将 *path_to_file* 替换为到 **hello-world-job.json**的路径，如果不在当前目录中，将 *s3_bucket_name* 替换为所选存储桶的 Amazon S3 存储桶路径，然后运行此命令将您的任务文档放入 Amazon S3 存储桶中。

   ```
   aws s3api put-object \
   --key hello-world-job.json \
   --body path_to_file/hello-world-job.json --bucket s3_bucket_name
   ```

   通过替换以下 URL 中的 *s3_bucket_name* and *AWS_region*，可以确定标识存储在 Amazon S3 中的任务文档的任务文档 URL。记录生成的 URL 稍后用作*job_ocument_path*

   ```
   https://s3_bucket_name.s3.AWS_Region.amazonaws.com/hello-world-job.json
   ```

   Note

   Amazon安全防止您无法在 Amazon Web Services 账户外面打开此 URL，例如通过使用浏览器。预设情况下，具有文件访问权限的 Amazon IoT任务引擎使用 URL。在生产环境中，您需要确保 Amazon IoT服务有权访问存储在 Amazon S3 中的任务文档。

保存任务文档的 URL 后，继续 the section called “为一台 IoT 设备在 Amazon IoT中运行任务” (p. 152)。

为一台 IoT 设备在 Amazon IoT中运行任务

本节中的过程启动 Raspberry Pi 上的Amazon IoT Device Client，在设备上运行任务代理，等待任务运行。还在 Amazon IoT中创建了一个任务资源，将将任务发送到 IoT 设备并在物联网设备上运行。

> Note
>
> 此过程仅在一台设备上运行任务。

要在 Raspberry Pi 上启动任务代理：

1. 在连接到 Raspberry Pi 的本地主机上的终端窗口中，运行此命令启动 Amazon IoTDevice Client。

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-jobs-config.json
```

2. 在终端窗口中，确认 Amazon IoTDevice Client 并显示这些消息

```
2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Jobs is enabled
                       .
                       .
                       .
2021-11-15T18:45:56.708Z [INFO]  {Main.cpp}: Client base has been notified that Jobs
 has started
2021-11-15T18:45:56.708Z [INFO]  {JobsFeature.cpp}: Running Jobs!
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 startNextPendingJobExecution accepted and rejected
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 nextJobChanged events
2021-11-15T18:45:56.708Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusAccepted for jobId +
2021-11-15T18:45:56.738Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionAccepted with code {0}
2021-11-15T18:45:56.739Z [DEBUG] {JobsFeature.cpp}: Attempting to subscribe to
 updateJobExecutionStatusRejected for jobId +
2021-11-15T18:45:56.753Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToNextJobChanged with code {0}
2021-11-15T18:45:56.760Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobRejected with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToStartNextJobAccepted with code {0}
2021-11-15T18:45:56.776Z [DEBUG] {JobsFeature.cpp}: Ack received for
 SubscribeToUpdateJobExecutionRejected with code {0}
2021-11-15T18:45:56.777Z [DEBUG] {JobsFeature.cpp}: Publishing
 startNextPendingJobExecutionRequest
2021-11-15T18:45:56.785Z [DEBUG] {JobsFeature.cpp}: Ack received for
 StartNextPendingJobPub with code {0}
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. 在终端窗口中，看到此消息后，继续下一步并创建任务资源。请注意，它可能不是列表中的最后一个条目。

```
2021-11-15T18:45:56.785Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

要创建 Amazon IoT任务资源

1. 在本地主机上：

a. 将 *job_ocument_url* 替换为 the section called “创建并存储任务的任务文档” (p. 151)的任务文档 URL。

b. 将 *thing_arn* 替换为您为设备创建的事物资源的 ARN，然后运行此命令。

```
aws iot create-job \
--job-id hello-world-job-1 \
--document-source "job_document_url" \
--targets "thing_arn" \
--target-selection SNAPSHOT
```

如果成功，该命令将返回类似此结果。

```
{
  "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-1",
  "jobId": "hello-world-job-1"
}
```

2. 在终端窗口中，您应看到 Amazon IoTDevice Client 的输出 。

```
2021-11-15T18:02:26.688Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Job ids differ
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: Executing job: hello-world-job-1
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Attempting to update job execution
 status!
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Assuming executable is in PATH
2021-11-15T18:10:24.890Z [INFO]  {JobsFeature.cpp}: About to execute: echo Hello world!
2021-11-15T18:10:24.890Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.890Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken 3TEWba9Xj6 in the updateJobExecution promises map
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process now running
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Child process about to call execvp
2021-11-15T18:10:24.890Z [DEBUG] {JobEngine.cpp}: Parent process now running, child PID
 is 16737
2021-11-15T18:10:24.891Z [DEBUG] {16737}: Hello world!
2021-11-15T18:10:24.891Z [DEBUG] {JobEngine.cpp}: JobEngine finished waiting for child
 process, returning 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job exited with status: 0
2021-11-15T18:10:24.891Z [INFO]  {JobsFeature.cpp}: Job executed successfully!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Attempting to update job execution
 status!
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stdout with the
 status details
2021-11-15T18:10:24.891Z [DEBUG] {JobsFeature.cpp}: Not including stderr with the
 status details
2021-11-15T18:10:24.892Z [DEBUG] {Retry.cpp}: Retryable function starting, it will
 retry until success
2021-11-15T18:10:24.892Z [DEBUG] {JobsFeature.cpp}: Created EphermalPromise for
 ClientToken GmQ0HTzWGg in the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken 3TEWba9Xj6
 from the updateJobExecution promises map
2021-11-15T18:10:24.905Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
```

```
2021-11-15T18:10:24.917Z [DEBUG] {JobsFeature.cpp}: Ack received for
 PublishUpdateJobExecutionStatus with code {0}
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Removing ClientToken GmQ0HTzWGg
 from the updateJobExecution promises map
2021-11-15T18:10:24.918Z [DEBUG] {JobsFeature.cpp}: Success response after
 UpdateJobExecution for job hello-world-job-1
2021-11-15T18:10:25.861Z [INFO]  {JobsFeature.cpp}: No pending jobs are scheduled,
 waiting for the next incoming job
```

3. 尽管 Amazon IoTDevice Client 正在运行并等待任务，您可以通过更改 `job-id`值并从步骤 1 开始重新运行 create-job。

完成运行任务后，在终端窗口中输入 ^C（control-C）来停止 Amazon IoTDevice Client。

教程：运行 Amazon IoTDevice Client 教程后清理

本教程中的过程将指导您在完成本学习路径中的教程时删除创建的文件和资源。

本教程中的过程

- 步骤 1：使用 Device Client Amazon IoT构建演示后清理设备 (p. 154)
- 步骤 2：使用 Amazon IoT Device Client 构建演示后清理您的 Amazon Web Services 账户 (p. 155)

步骤 1：使用 Device Client Amazon IoT构建演示后清理设备

本教程介绍了在此学习路径中构建演示后如何清理 microSD 卡的两个选项。选择提供所需安全级别的选项。

请注意，清理设备的 microSD 卡不会移除任何您创建的 Amazon IoT资源。要在清理设备的 microSD 卡后清理 Amazon IoT资源，应查看 the section called “使用 Amazon IoT Device Client 构建演示后进行清理” (p. 155)上的教程。

选项 1：通过重写 microSD 卡进行清理

在完成本学习路径中的教程之后，清理 microSD 卡最简单、最彻底的方法是用您在首次准备设备时创建的保存图像文件覆盖 microSD 卡。

此过程使用本地主机将保存的 microSD 卡映像写入 microSD 卡。

> Note
>
> 如果您的设备没有将可移动存储介质用于操作系统，请参阅该设备的步骤。

要向 microSD 卡写入一个新映像

1. 在本地主机上，找到要写入 microSD 卡的已保存 microSD 卡映像。
2. 将 microSD 卡插到您的计算机上。
3. 使用 SD 卡成像工具，将选定的图像文件写入 microSD 卡。
4. 将 Raspberry Pi 操作系统映像写入 microSD 卡后，弹出 microSD 卡并将其从本地主机安全地移除。

您的 microSD 卡已准备就绪，可供使用。

选项 2：通过删除用户目录进行清理

要在完成教程后清理 microSD 卡而不重写 microSD 卡映像，可以单独删除用户目录。这并不像从保存的图像中重写 microSD 卡那么彻底，因为它不会删除可能已安装的任何系统文件。

如果删除用户目录足以满足您的需要，可以按照此步骤操作。

从设备中删除此学习路径的用户目录

1. 运行这些命令可以在连接到设备的终端窗口中删除在此学习路径中创建的用户目录、子目录及其所有文件。

   Note

   删除这些目录和文件后，如果不再完成教程，您将无法运行演示。

```
rm -Rf ~/dc-configs
rm -Rf ~/policies
rm -Rf ~/messages
rm -Rf ~/certs
rm -Rf ~/.aws-iot-device-client
```

2. 在连接到设备的终端窗口中，运行这些命令删除应用程序源目录和文件。

   Note

   这些命令不会卸载任何程序。他们只删除用于构建和安装它们的源文件。删除这些文件后，Amazon CLI 和 Amazon IoTDevice Client 可能不适用。

```
rm -Rf ~/aws-cli
rm -Rf ~/aws
rm -Rf ~/aws-iot-device-client
```

步骤 2：使用 Amazon IoT Device Client 构建演示后清理您的 Amazon Web Services 账户

这些过程可以帮助您识别和删除您在完成此学习路径中的教程时创建的 Amazon资源。

清理 Amazon IoT资源

此过程可以帮助您识别并删除您在完成此学习路径中的教程时创建的 Amazon IoT资源。

在此学习路径中创建的 Amazon IoT资源

教程	事物资源	策略资源
the section called “安装并配置 Amazon IoTDevice Client” (p. 124)	DevclitestTthing	PubSubTestThingPolicy
the section called “演示 MQTT 消息与 Amazon IoTDevice Client 通信” (p. 133)	PubSubTestThing	PubSubTestThingPolicy
the section called “使用 Amazon IoTDevice Client 演示远程操作（任务）” (p. 145)	定义的用户（可能有不止一个）	定义的用户（可能有不止一个）

要删除 Amazon IoT资源，对于您创建的每个事物资源，请按照此过程操作

1. 将 *thing_name* 替换为要删除的对象资源的名称，然后运行此命令列出从本地主机附加到对象资源的证书。

```
aws iot list-thing-principals --thing-name thing_name
```

此命令返回类似这样的响应，其中列出了附加到 *thing_name*的证书。在大多数情况下，列表中只有一个证书。

```
{
    "principals": [
        "arn:aws:iot:us-
west-2:57EXAMPLE833:cert/23853eea3cf0edc7f8a69c74abeafa27b2b52823cab5b3e156295e94b26ae8ac"
    ]
}
```

2. 对于上一个命令列出的每个证书：

   a. 将 *certificate_ID*替换为上一个命令中的证书 ID。证书ID是上一个命令返回的 ARN 中遵循 `cert/`的字母数字字符。然后运行此命令以停用证书。

   ```
   aws iot update-certificate --new-status INACTIVE --certificate-id certificate_ID
   ```

   如果成功，该命令不返回任何内容。

   b. 将 *certificate_ARN*替换为之前返回的证书列表中的证书 ARN，然后运行此命令列出附加到此证书的策略。

   ```
   aws iot list-attached-policies --target certificate_ARN
   ```

   此命令返回类似这样的响应，其中列出了附加到证书的策略。在大多数情况下，列表中只有一个策略。

   ```
   {
       "policies": [
           {
               "policyName": "DevCliTestThingPolicy",
               "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/
   DevCliTestThingPolicy"
           }
       ]
   }
   ```

   c. 对于附加到该证书的每个策略。

      i. 将 *policy_name*替换为上一个命令中的 `policyName`值，将 *certificate_ARN*替换为证书的 ARN，然后运行此命令将策略与证书分离。

      ```
      aws iot detach-policy --policy-name policy_name --target certificate_ARN
      ```

      如果成功，该命令不返回任何内容。

      ii. 将 *policy_name*替换为 `policyName`值，然后运行此命令查看策略是否附加到其他证书。

      ```
      aws iot list-targets-for-policy --policy-name policy_name
      ```

      如果该命令返回这样的空列表，该策略不会附加到任何证书，您将继续列出策略版本。如果还有附加到策略的证书，请继续 detach-thing-principal步骤。

      ```
      {
      ```

```
    "targets": []
}
```

iii. 将 *policy_name*替换为 policyName值，然后运行此命令检查策略版本。要删除策略，它必须只有一个版本。

```
aws iot list-policy-versions --policy-name policy_name
```

如果策略只有一个版本（如此示例），则可以跳至步骤 delete-policy立即删除策略。

```
{
    "policyVersions": [
        {
            "versionId": "1",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:02:46.778000+00:00"
        }
    ]
}
```

如果策略有多个版本，如本例所示，则必须先删除 false值为 isDefaultVersion的策略版本，然后才能删除策略。

```
{
    "policyVersions": [
        {
            "versionId": "2",
            "isDefaultVersion": true,
            "createDate": "2021-11-18T01:52:04.423000+00:00"
        },
        {
            "versionId": "1",
            "isDefaultVersion": false,
            "createDate": "2021-11-18T01:30:18.083000+00:00"
        }
    ]
}
```

如果您需要删除策略版本，将 *policy_name*替换为 policyName值，将 *version_ID*替换为来自上一个命令的 versionId值，然后运行此命令删除策略版本。

```
aws iot delete-policy-version --policy-name policy_name --policy-version-
id version_ID
```

如果成功，该命令不返回任何内容。

删除策略版本后，请重复此步骤，直到该策略只有一个策略版本。

iv. 将 *policy_name*替换为 policyName值，然后运行此命令删除策略。

```
aws iot delete-policy --policy-name policy_name
```

d. 将 *thing_name*替换为事物名称，将 *certificate_ARN*替换为证书的 ARN，然后运行此命令将证书与事物资源分离。

```
aws iot detach-thing-principal --thing-name thing_name --principal certificate_ARN
```

如果成功，该命令不返回任何内容。

e. 将 *certificate_ID*替换为上一个命令中的证书 ID。证书 ID 是上一个命令返回的 ARN 中遵循 cert/的字母数字字符。运行这个命令删除证书资源。

```
aws iot delete-certificate --certificate-id certificate_ID
```

如果成功，该命令不返回任何内容。

3. 将 *thing_name*替换为事物名称，然后运行此命令删除该事物。

```
aws iot delete-thing --thing-name thing_name
```

如果成功，该命令不返回任何内容。

清理 Amazon资源

此过程可以帮助您识别和删除您在完成此学习路径中的教程时创建的其他 Amazon资源。

在此学习路径中创建的其他 Amazon资源

教程	资源类型	资源名称或 ID
the section called “使用 Amazon IoTDevice Client 演示远程操作（任务）” (p. 145)	Amazon S3 对象	hello-world-job.json
the section called “使用 Amazon IoTDevice Client 演示远程操作（任务）” (p. 145)	Amazon IoT 任务资源	定义的用户

要删除在此学习路径中创建的 Amazon资源

1. 要删除在此学习路径中创建的职位

a. 运行此命令列出您的任务Amazon Web Services 账户.

```
aws iot list-jobs
```

此命令将返回您的 Amazon Web Services 账户和 Amazon Web Services 区域的 Amazon IoT任务列表。

```
{
    "jobs": [
        {
            "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-2",
            "jobId": "hello-world-job-2",
            "targetSelection": "SNAPSHOT",
            "status": "COMPLETED",
            "createdAt": "2021-11-16T23:40:36.825000+00:00",
            "lastUpdatedAt": "2021-11-16T23:40:41.375000+00:00",
            "completedAt": "2021-11-16T23:40:41.375000+00:00"
        },
        {
            "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-1",
            "jobId": "hello-world-job-1",
            "targetSelection": "SNAPSHOT",
            "status": "COMPLETED",
            "createdAt": "2021-11-16T23:35:26.381000+00:00",
```

```
            "lastUpdatedAt": "2021-11-16T23:35:29.239000+00:00",
            "completedAt": "2021-11-16T23:35:29.239000+00:00"
        }
    ]
}
```

b. 对于从列表中识别为您在此学习路径中创建任务的每份任务，将 *jobId*替换为要删除的任务 jobId值，然后运行此命令删除 Amazon IoT任务。

```
aws iot delete-job --job-id jobId
```

如果命令成功，该命令不返回任何内容。

2. 要删除您在此学习路径中存储在 Amazon S3 存储桶中的任务文档。

a. 将 *bucket*替换为您使用的存储桶名称，然后运行此命令列出您使用的 Amazon S3 存储桶中的对象。

```
aws s3api list-objects --bucket bucket
```

该命令返回存储桶中 Amazon S3 对象的列表，该列表如下所示。

```
{
    "Contents": [
        {
            "Key": "hello-world-job.json",
            "LastModified": "2021-11-18T03:02:12+00:00",
            "ETag": "\"868c8bc3f56b5787964764d4b18ed5ef\"",
            "Size": 54,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "iot_job_firmware_update.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"7c68c591949391791ecf625253658c61\"",
            "Size": 66,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        },
        {
            "Key": "order66.json",
            "LastModified": "2021-04-13T21:57:07+00:00",
            "ETag": "\"bca60d5380b88e1a70cc27d321caba72\"",
            "Size": 29,
            "StorageClass": "STANDARD",
            "Owner": {
                "DisplayName": "EXAMPLE",
                "ID":
 "e9e3d6ec1EXAMPLEf5bfb5e6bd0a2b6ed03884d1ed392a82ad011c144736a4ee"
            }
        }
    ]
}
```

b. 对于从列表中识别为在此学习路径中创建的对象的每个对象，将 *bucket*替换为要删除的对象的存储桶名称，将 *key*替换为密钥值，然后运行此命令删除 Amazon S3 对象。

```
aws s3api delete-object --bucket bucket --key key
```

如果命令成功，该命令不返回任何内容。

删除在完成此学习路径时创建的所有 Amazon资源和对象，您可以重新开始并重复教程。

使用 Amazon IoT设备开发工具包构建解决方案

本节中的教程帮助您完成开发可使用 Amazon IoT 部署到生产环境的 IoT 解决方案的步骤。

与上一节 the section called “用 Amazon IoT Device Client 构建演示” (p. 114)中的教程相比，这些教程可能需要更多的时间来完成，因为它们使用 Amazon IoT设备软件开发包，并更详细地解释了应用的概念，帮助您创建安全可靠的解决方案。

开始使用 Amazon IoT设备开发工具包构建解决方案

这些教程将引导您完成不同的 Amazon IoT 场景。在适当的情况下，这些教程将使用 Amazon IoT 设备 SDK。

主题

- 教程：使用 Amazon IoT设备开发工具包将设备连接到 Amazon IoT Core (p. 160)
- 创建将设备数据路由到其他服务的 Amazon IoT规则 (p. 175)
- 在设备处于离线状态时使用设备影子保持设备状态 (p. 201)
- 教程：为 Amazon IoT Core创建自定义授权方 (p. 219)
- 教程：使用 Amazon IoT和 Raspberry Pi 监控土壤含水量 (p. 230)

教程：使用 Amazon IoT设备开发工具包将设备连接到 Amazon IoT Core

本教程演示如何将设备连接到 Amazon IoT Core，以便它可以与 Amazon IoT收发数据。完成本教程后，您的设备将配置为连接到 Amazon IoT Core，您将了解设备如何与 Amazon IoT通信。

在本教程中，您将：

1. the section called “为 Amazon IoT准备好您的设备。” (p. 161)
2. the section called “查看 MQTT 协议” (p. 161)
3. the section called “查看 pubsub.py Device SDK 示例应用程序” (p. 162)
4. the section called “连接设备并与 Amazon IoT Core通信” (p. 167)
5. the section called “查看结果” (p. 172)

完成本教程需要大约 1 小时。

在开始本教程之前，请确保您具有：

- 已完成 Amazon IoT Core 入门 (p. 15)

在本教程中，您必须在某部分 the section called “配置您的设备” (p. 37)，请为您的设备选择 the section called “连接 Raspberry Pi 或其他设备” (p. 51)选项，然后使用 Python 语言选项来配置设备。

在该教程中使用的终端窗口保持开启，因为您还将在本教程中使用它。

- 可以运行 Amazon IoTDevice SDK v2 for Python。

  本教程介绍了如何使用 Python 代码示例将设备连接到 Amazon IoT Core，这些示例需要一个功能相对强大的设备。

  如果您使用的是资源受限的设备，则这些代码示例可能无法适用于这些设备。在这种情况下，您可能会通过 the section called “使用 Amazon IoT Device SDK for Embedded C” (p. 172)教程成功完成更多操作。

为 Amazon IoT准备好您的设备。

在 Amazon IoT Core 入门 (p. 15)，您已准备好您的设备以及 Amazon账户，所以它们能够进行通信。本部分回顾了准备工作的各个方面，这些方面涉及与 Amazon IoT Core进行的任何设备通信。

对于要连接到 Amazon IoT Core的设备：

1. 您必须具有 Amazon Web Services 账户。

   如果您没有账户，则 设置您的Amazon Web Services 账户 (p. 16)中的流程将介绍如何创建Amazon Web Services 账户。
2. 在该账户中，您必须具有以下为您在 Amazon Web Services 账户和区域中的设备定义的 Amazon IoT资源。

   创建 Amazon IoT 资源 (p. 34) 中的流程介绍了如何为您的 Amazon Web Services 账户和区域中的设备创建这些资源。

   - 使用 Amazon IoT注册并激活以验证设备的设备证书。

     该证书通常使用 Amazon IoT事物对象创建并随附其上。虽然设备不需要某个事物对象才能连接到 Amazon IoT，但它会使设备获得额外的 Amazon IoT功能。
   - 附加到设备证书的策略，该证书授权其连接到 Amazon IoT Core并执行您希望它执行的所有操作。
3. 能够访问您的 Amazon Web Services 账户的设备端点的互联网连接。

   设备端点在 Amazon IoT 设备数据和服务终端节点 (p. 72)中描述，并且可以在 Amazon IoT控制台中的设置页面看到。
4. 诸如 Amazon IoTDevice SDK 提供的通信软件。本教程使用 Amazon IoT Device SDK v2 for Python。

查看 MQTT 协议

在我们讨论示例应用程序之前，它能有助于了解 MQTT 协议。与其它网络通信协议（如 HTTP）相比，MQTT 协议具有一些优势，这使得它成为了 IoT 设备的常用选择。本部分回顾的是适用于本教程的 MQTT 的主要方面。有关如何将 MQTT 与 HTTP 进行比较的信息，请参阅 为设备通信选择协议 (p. 76)。

MQTT 使用发布/订阅通信模型

MQTT 协议在其主机上使用发布/订阅通信模式。此模式与 HTTP 使用的请求/响应模式不同。借助 MQTT，设备可以与由唯一客户端 ID 标识的主机建立会话。要发送数据，设备会将主题标识的消息发布到主机中的消息代理上。要接收来自消息代理的消息，设备通过在订阅请求中向消息代理发送主题筛选条件，来订阅主题。

MQTT 支持持久性会话

消息代理接收来自设备的消息，并将消息发布到已订阅消息的设备。借助持久性会话 (p. 79)（即使在初始设备断开连接时，仍能保持活动状态的会话），设备可以检索在断开连接时发布的消息。在设备端，MQTT 支持服务质量级别（QoS (p. 79)），以确保主机接收设备发送的消息。

查看 pubsub.py Device SDK 示例应用程序

本部分将回顾本教程中适用的来自 Amazon IoTDevice SDK v2 for Python 中的 `pubsub.py`示例应用程序。在这里，我们将回顾它如何连接到 Amazon IoT Core来发布和订阅 MQTT 消息。下一部分介绍了一些练习，帮助您了解设备如何与 Amazon IoT Core连接及通信。

pubsub.py 示例应用程序演示了 MQTT 与 Amazon IoT Core连接的各个方面：

- 通信协议 (p. 162)
- 持久会话 (p. 164)
- 服务质量 (p. 165)
- 消息发布 (p. 165)
- 消息订阅 (p. 166)
- 设备断开和重新连接 (p. 167)

通信协议

`pubsub.py` 示例演示了使用 MQTT 和 MQTT over WSS 协议进行的 MQTT 连接。Amazon 通用运行时 (Amazon CRT) 库提供低级通信协议支持，并包含在 Amazon IoTDevice SDK v2 for Python 中。

MQTT

`pubsub.py` 示例在 `mqtt_connection_builder` 中调用 `mtls_from_path`（此处显示）以使用 MQTT 协议建立与 Amazon IoT Core 的连接。`mtls_from_path` 使用 X.509 证书和 TLS v1.2 对设备进行身份验证。Amazon CRT 库处理该连接的较低级别的详细信息。

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
    endpoint=args.endpoint,
    cert_filepath=args.cert,
    pri_key_filepath=args.key,
    ca_filepath=args.ca_file,
    client_bootstrap=client_bootstrap,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

您的 Amazon Web Services 账户IoT 设备端点

在示例应用程序中，此值将从命令行传入。

`cert_filepath`

设备证书文件的路径。

在示例应用程序中，此值将从命令行传入。

`pri_key_filepath`

使用其证书文件创建的设备私有密钥文件的路径

在示例应用程序中，此值将从命令行传入。

`ca_filepath`

Root CA 文件的路径。仅当 MQTT 服务器使用信任存储中尚未存在的证书时才需要此项。

在示例应用程序中，此值将从命令行传入。

`client_bootstrap`

处理套接字通信活动的通用运行时对象

在示例应用程序中，此对象会在调用 `mqtt_connection_builder.mtls_from_path`前实例化。

`on_connection_interrupted`, `on_connection_resumed`

当设备连接中断和恢复时调用的回调函数

`client_id`

在 Amazon Web Services 区域中唯一标识此设备的 ID

在示例应用程序中，此值将从命令行传入。

`clean_session`

启动新的持久会话，或者（如果会话已存在）重新连接到现有会话

`keep_alive_secs`

保持活动状态值（以秒为单位），在 CONNECT请求中发送。在此时间间隔内将自动发送 ping。如果服务器在此值的 1.5 倍之后没有收到 ping，假定连接丢失。

基于 WSS 的 MQTT

`pubsub.py` 示例在 `mqtt_connection_builder` 中调用 `websockets_with_default_aws_signing`（此处显示）以使用借助 WSS 的 MQTT 协议建立与 Amazon IoT Core 的连接。`websockets_with_default_aws_signing` 使用 Signature V4 使用借助 WSS 的 MQTT 连接对设备进行身份验证。

```
mqtt_connection = mqtt_connection_builder.websockets_with_default_aws_signing(
    endpoint=args.endpoint,
    client_bootstrap=client_bootstrap,
    region=args.signing_region,
    credentials_provider=credentials_provider,
    websocket_proxy_options=proxy_options,
    ca_filepath=args.ca_file,
    on_connection_interrupted=on_connection_interrupted,
    on_connection_resumed=on_connection_resumed,
    client_id=args.client_id,
    clean_session=False,
    keep_alive_secs=6
)
```

`endpoint`

您的 Amazon Web Services 账户IoT 设备端点

在示例应用程序中，此值将从命令行传入。

`client_bootstrap`

处理套接字通信活动的通用运行时对象

在示例应用程序中，此对象会在调用 `mqtt_connection_builder.websockets_with_default_aws_signing`前实例化。

`region`

Signature V4 身份验证使用的 Amazon签名区域。在 `pubsub.py`中，它将在命令行中传递输入的参数。

在示例应用程序中，此值将从命令行传入。

`credentials_provider`

Amazon 提供用于身份验证的凭证

在示例应用程序中，此对象会在调用 `mqtt_connection_builder.websockets_with_default_aws_signing`前实例化。

`websocket_proxy_options`

HTTP 代理选项（如果使用代理主机）

在示例应用程序中，此值在调用 `mqtt_connection_builder.websockets_with_default_aws_signing`前初始化。

`ca_filepath`

Root CA 文件的路径。仅当 MQTT 服务器使用信任存储中尚未存在的证书时才需要此项。

在示例应用程序中，此值将从命令行传入。

`on_connection_interrupted`, `on_connection_resumed`

当设备连接中断和恢复时调用的回调函数

`client_id`

在 Amazon Web Services 区域中唯一标识此设备的 ID。

在示例应用程序中，此值将从命令行传入。

`clean_session`

启动新的持久会话，或者（如果会话已存在）重新连接到现有会话

`keep_alive_secs`

保持活动状态值（以秒为单位），在 `CONNECT`请求中发送。在此时间间隔内将自动发送 ping。如果服务器在此值的 1.5 倍之后没有收到 ping，则假定连接丢失。

HTTPS

HTTPS 怎么样？Amazon IoT Core 支持发布 HTTPS 请求的设备。从编程的角度来看，设备会像其它应用程序一样将 HTTPS 请求发送到 Amazon IoT Core。有关从设备发送 HTTP 消息的 Python 程序示例，请参阅使用 Python 的 `requests`库的 HTTPS 代码示例 (p. 91)。本示例将使用 HTTPS 将消息发送到 Amazon IoT Core，以便 Amazon IoT Core将其解释为 MQTT 消息。

尽管 Amazon IoT Core支持来自设备的 HTTPS 请求，请务必查看 为设备通信选择协议 (p. 76)的消息，以便您可以对要使用哪种协议进行设备通信做出明智决定。

持久会话

在示例应用程序中，将 `clean_session`参数设置为 `False`即表示连接应该是持久连接。实际上，这意味着此调用打开的连接将重新连接到现有持久会话（如果存在）。否则，它会创建并连接到新的持久会话。

对于持久会话，如果设备未连接，则发送到设备的消息将由消息代理存储。当设备重新连接到持久会话时，消息代理会向设备发送它已订阅的所有已存储的消息。

如果没有持久会话，设备将不会接收设备未连接时发送的消息。使用哪个选项取决于您的应用程序，以及是否必须传达设备未连接时发生的消息。有关更多信息，请参阅 MQTT 持久性会话 (p. 79)。

服务质量

当设备发布和订阅消息时，可以设置首选的服务质量 (QoS)。Amazon IoT 支持用于发布和订阅操作的 QoS 级别 0 和 1。有关 Amazon IoT中 QoS 级别的更多信息，请参阅 MQTT 服务质量 (QoS) 选项 (p. 79)。

适用于 Python 的 AmazonCRT 运行时为其支持的 QoS 级别定义了这些常量：

Python QoS 级别

MQTT QoS 级别	SDK 使用的 Python 符号值	描述
QoS 级别 0	`mqtt.QoS.AT_MOST_ONCE`	无论是否收到消息，均只会尝试发送一次消息。如果出现诸如设备未连接或存在网络错误的情况，则消息可能根本不会发送。
QoS 级别 1	`mqtt.QoS.AT_LEAST_ONCE`	消息将重复发送，直至收到PUBACK确认。

在示例应用程序中，发布和订阅请求的 QoS 级别为 1 (`mqtt.QoS.AT_LEAST_ONCE`)。

- 发布时的 QoS

  当设备发布 QoS 级别为 1 的消息时，它会重复发送消息，直至从消息代理收到 PUBACK响应。如果设备未连接，消息将在重新连接后排队等待发送。
- 订阅时的 QoS

  当设备订阅 QoS 级别为 1 的消息时，消息代理会保存设备订阅的消息，直到这些消息可以发送到设备。消息代理会重新发送消息，直至收到设备发出的 PUBACK响应。

消息发布

成功建立同 Amazon IoT Core的连接后，设备可以发布消息。`pubsub.py` 示例通过调用 `mqtt_connection`对象的 `publish`操作来完成此操作。

```
mqtt_connection.publish(
    topic=args.topic,
    payload=message,
    qos=mqtt.QoS.AT_LEAST_ONCE
)
```

`topic`

标识消息的消息主题名称

在示例应用程序中，这是从命令行传入的。

`payload`

格式化为字符串的消息负载（例如，JSON 文档）

在示例应用程序中，这是从命令行传入的。

JSON 文档是一种常见的负载格式，并且由其它 Amazon IoT服务所认可；但是，消息负载的数据格式，可以是发布者和订阅者同意的任何内容。但在某些情况下，对于大多数操作来说，其它 Amazon IoT服务只能识别 JSON 和 CBOR。

`qos`

此消息的 QoS 级别

消息订阅

从 Amazon IoT和其它服务及设备接收消息时，设备会按其主题名称订阅这些消息。设备可以通过指定主题名称 (p. 94)来订阅单个消息，也可以通过指定主题筛选条件 (p. 95)来订阅一组消息（筛选条件中可以包含通配符）。`pubsub.py` 示例使用此处显示的代码订阅消息并注册回调函数，以便在收到消息后处理消息。

```
subscribe_future, packet_id = mqtt_connection.subscribe(
    topic=args.topic,
    qos=mqtt.QoS.AT_LEAST_ONCE,
    callback=on_message_received
)
subscribe_result = subscribe_future.result()
```

`topic`

要订阅的主题。这可以是主题名称或主题筛选条件。

在示例应用程序中，这是从命令行传入的。

`qos`

在设备断开连接时，消息代理是否应该存储这些消息。

`mqtt.QoS.AT_LEAST_ONCE` 的值（QoS 级别 1），创建连接时需要指定持久性会话 (`clean_session=False`)。

`callback`

要调用以处理已订阅消息的函数。

`mqtt_connection.subscribe` 函数返回未来和数据包 ID。如果订阅请求成功初始化，则返回的数据包 ID 大于 0。要确保消息代理已接收并注册订阅，您必须等待异步操作的结果返回，如代码示例所示。

回调函数

`pubsub.py` 示例中的回调函数会在设备接收订阅的消息时处理这些消息。

```
def on_message_received(topic, payload, **kwargs):
    print("Received message from topic '{}': {}".format(topic, payload))
    global received_count
    received_count += 1
    if received_count == args.count:
        received_all_event.set()
```

`topic`

消息的主题

这是收到的消息的特定主题名称，即使您订阅了主题筛选条件也是如此。

`payload`

消息负载

此格式是特定于应用程序的格式。

`kwargs`

可能的其它实际参数，如 `mqtt.Connection.subscribe`中所述。

在 `pubsub.py`示例中，`on_message_received` 仅显示主题及其负载。它还会计算在达到限制后收到的结束程序的消息。

您的应用程序将评估主题和负载，以确定要执行的操作。

设备断开和重新连接

`pubsub.py` 示例包括在设备断开连接和重新建立连接时调用的回调函数。您的设备对这些事件采取的操作是特定于应用程序的。

当设备首次连接时，它必须订阅主题才能接收。如果设备在重新连接时存在会话，则会恢复其订阅，并在设备重新连接后，将来自这些订阅的所有存储消息发送到设备。

如果设备的会话在重新连接时不再存在，则必须重新订阅其订阅。持久会话的生命周期有限，当设备断开连接太长时间时，可能会过期。

连接设备并与 Amazon IoT Core通信

本部分介绍了一些练习，可帮助您探索将设备连接到 Amazon IoT Core的不同方面。在这些练习中，您将使用 Amazon IoT控制台中的 MQTT 测试客户端查看您的设备发布的内容，并将消息发布到您的设备。这些练习使用来自 Amazon IoTDevice SDK v2 for Python 的 `pubsub.py`示例并根据您在 Amazon IoT Core 入门 (p. 15)教程中的经验构建。

在本部分中，您将：

- 订阅通配符主题筛选条件 (p. 167)
- 处理主题筛选条件订阅 (p. 169)
- 从您的设备发布消息 (p. 170)

对于这些练习，您将从 `pubsub.py`示例程序开始操作。

> **Note**
>
> 这些练习假定您已完成 Amazon IoT Core 入门 (p. 15)教程并使用该教程中的设备终端窗口。

订阅通配符主题筛选条件

在本练习中，您将修改用于调用 `pubsub.py`的命令行以订阅通配符主题筛选条件，并根据消息主题处理收到的消息。

练习流程

在本练习中，假设您的设备包含温度控制和光控。它使用这些主题名称来标识有关的消息。

1. 在开始练习之前，请尝试根据 Amazon IoT Core 入门 (p. 15)教程在您的设备上运行此命令，确保一切都准备就绪，可正常进行练习。

```
cd ~/aws-iot-device-sdk-python-v2/samples
python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/
certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
```

您看到的输出应该与您在入门教程 (p. 55)看到的一样。

2. 在本练习中，请更改这些命令行参数。

操作	命令行参数	效果
添加	--message ""	配置 pubsub.py以仅侦听
添加	--count 2	收到两条消息后结束程序
更改	--topic device/+/details	设定要订阅的主题筛选条件

对初始命令行进行这些更改将得到此命令行。在设备的终端窗口中输入此命令。

```
python3 pubsub.py --message "" --count 2 --topic device/+/details --ca_file ~/certs/
Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --
endpoint your-iot-endpoint
```

程序应该类似如下所示：

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-24d7cdcc-cc01-458c-8488-2d05849691e1'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
```

如果您在终端上看到类似的内容，则表示您的设备已准备就绪，并侦听主题名称以 topic-1/开始并以结束 /detail的消息。所以，让我们来测试吧。

3. 以下是您的设备可能会收到的几条消息。

主题名称	消息负载
device/temp/details	{ "desiredTemp": 20, "currentTemp": 15 }
device/light/details	{ "desiredLight": 100, "currentLight": 50 }

4. 在 Amazon IoT控制台中使用 MQTT 测试客户端，将上一步中描述的消息发送到您的设备。

   a. 在 Amazon IoT控制台中打开 MQTT 测试客户端。

   b. 在 Subscribe to topic（订阅主题）中，在 Subscription topic field（订阅主题字段）输入主题筛选条件：**device/+/details**，然后选择 Subscribe to topic（订阅主题）。

   c. 在 MQTT 测试客户端中的 Subscriptions（订阅）列中，选择device/+/details。

   d. 对于上表中的每个主题，请在 MQTT 测试客户端中执行以下操作：

      1. 在 Publish（发布）中，在表中输入 Topic name（主题名称）列的值。
      2. 在主题名称下方的消息负载字段中，在表中输入 Message payload（消息负载）列的值。
      3. 观看 pubsub.py运行所在的终端窗口，并在 MQTT 测试客户端中选择 Publish to topic（发布到主题）。

      您应该看到该消息是由终端窗口中的 pubsub.py接收的。

练习结果

有了这个，pubsub.py，使用通配符主题筛选条件订阅消息，接收消息并在终端窗口中显示这些消息。请注意您如何订阅单个主题筛选条件，并调用回调函数来处理具有两个不同主题的消息。

处理主题筛选条件订阅

在上一练习的基础上，修改 pubsub.py示例应用程序以评估消息主题并根据主题处理订阅的消息。

练习流程

评估消息主题

1. 将 pubsub.py复制到 pubsub2.py。
2. 在您常用的文本编辑器或 IDE 中打开 pubsub2.py。
3. 在 pubsub2.py中，查找 on_message_received函数。
4. 在 on_message_received中，在以 print("Received message开头的行之后和在以 global received_count开头的行之前插入以下代码。

```
    topic_parsed = False
    if "/" in topic:
        parsed_topic = topic.split("/")
        if len(parsed_topic) == 3:
            # this topic has the correct format
            if (parsed_topic[0] == 'device') and (parsed_topic[2] == 'details'):
                # this is a topic we care about, so check the 2nd element
                if (parsed_topic[1] == 'temp'):
                    print("Received temperature request: {}".format(payload))
                    topic_parsed = True
                if (parsed_topic[1] == 'light'):
                    print("Received light request: {}".format(payload))
                    topic_parsed = True
    if not topic_parsed:
        print("Unrecognized message topic.")
```

5. 保存更改并使用此命令行运行修改后的程序。

```
python3 pubsub2.py --message "" --count 2 --topic device/+/details --ca_file ~/certs/
Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --
endpoint your-iot-endpoint
```

6. 在 Amazon IoT控制台中打开 MQTT 测试客户端。
7. 在 Subscribe to topic（订阅主题）中，在 Subscription topic field（订阅主题字段）输入主题筛选条件：**device/+/details**，然后选择 Subscribe to topic（订阅主题）。
8. 在 MQTT 测试客户端中的 Subscriptions（订阅）列中，选择device/+/details。
9. 对于此表中的每个主题，请在 MQTT 测试客户端中执行以下操作：

主题名称	消息负载
device/temp/details	{ "desiredTemp": 20, "currentTemp": 15 }
device/light/details	{ "desiredLight": 100, "currentLight": 50 }

1. 在 Publish（发布）中，在表中输入 Topic name（主题名称）列的值。

2. 在主题名称下方的消息负载字段中，在表中输入 Message payload（消息负载）列的值。
3. 观看 pubsub.py运行所在的终端窗口，并在 MQTT 测试客户端中选择 Publish to topic（发布到主题）。

您应该看到该消息是由终端窗口中的 pubsub.py接收的。

您应在终端窗口中看到类似的内容。

```
Connecting to a3qexamplesffp-ats.iot.us-west-2.amazonaws.com with client ID 'test-
af794be0-7542-45a0-b0af-0b0ea7474517'...
Connected!
Subscribing to topic 'device/+/details'...
Subscribed with QoS.AT_LEAST_ONCE
Waiting for all messages to be received...
Received message from topic 'device/light/details': b'{ "desiredLight": 100,
 "currentLight": 50 }'
Received light request: b'{ "desiredLight": 100, "currentLight": 50 }'
Received message from topic 'device/temp/details': b'{ "desiredTemp": 20, "currentTemp":
 15 }'
Received temperature request: b'{ "desiredTemp": 20, "currentTemp": 15 }'
2 message(s) received.
Disconnecting...
Disconnected!
```

练习结果

在本练习中，您添加了代码，以便示例应用程序能够识别并处理回调函数中的多条消息。有了这个，您的设备便可以接收消息并对它们采取行动。

您的设备接收和处理多条消息的另一种方式是单独订阅不同的消息，并将每个订阅分配给自己的回调函数。

从您的设备发布消息

您可以使用 pubsub.py 示例应用程序从您的设备发布消息。虽然它会按原样发布消息，但消息不能作为 JSON 文档读取。本练习修改了示例应用程序，以便能够在可以被 Amazon IoT Core读取的消息负载中发布 JSON 文档。

练习流程

在本练习中，以下消息将以 device/data主题发布。

```
{
    "timestamp": 1601048303,
    "sensorId": 28,
    "sensorData": [
        {
        "sensorName": "Wind speed",
        "sensorValue": 34.2211224
        }
    ]
}
```

准备 MQTT 测试客户端以监控本练习中的消息

1. 在 Subscribe to topic（订阅主题）中，在 Subscription topic field（订阅主题字段）输入主题筛选条件：**device/data**，然后选择 Subscribe to topic（订阅主题）。
2. 在 Subscriptions（订阅）列中，选择 device/data（设备/数据）。

3. 将 MQTT 测试客户端窗口保持打开状态，等待来自设备的消息。

使用 pubsub.py 示例应用程序发送 JSON 文档

1. 在您的设备上，将 pubsub.py复制到 pubsub3.py。
2. 编辑 pubsub3.py来更改其所发布消息的格式。

   a. 在文本编辑器中打开 pubsub3.py。
   b. 查找此代码行：

      ```
      message = "{} [{}]".format(args.message, publish_count)
      ```

   c. 将其更改为：

      ```
      message = "{}".format(args.message)
      ```

   d. 保存您的更改。
3. 在您的设备上运行此命令以发送消息两次。

```
python3 pubsub3.py  --ca_file ~/certs/Amazon-root-CA-1.pem  --cert ~/certs/
device.pem.crt  --key ~/certs/private.pem.key  --topic device/data  --count 2 --
message '{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'  --endpoint your-iot-endpoint
```

4. 在 MQTT 测试客户端中，检查它是否已解释并格式化了消息负载中的 JSON 文档，如下所示：

预设情况下，pubsub3.py 也会订阅它发送的消息。您应该看到它在应用程序的输出中收到了消息。终端窗口应类似如下所示。

```
Connecting to a3qEXAMPLEsffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-5cff18ae-1e92-4c38-a9d4-7b9771afc52f'...
Connected!
Subscribing to topic 'device/data'...
Subscribed with QoS.AT_LEAST_ONCE
Sending 2 message(s)
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
Publishing message to topic 'device/data':
 {"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}
Received message from topic 'device/data':
 b'{"timestamp":1601048303,"sensorId":28,"sensorData":[{"sensorName":"Wind
 speed","sensorValue":34.2211224}]}'
```

```
2 message(s) received.
Disconnecting...
Disconnected!
```

练习结果

有了这个，您的设备可以生成要发送到 Amazon IoT Core的消息以测试基本连通性并提供 Amazon IoT Core来处理的设备消息。例如，您可以使用此应用程序从您的设备发送测试数据以测试 Amazon IoT规则操作。

查看结果

本教程中的示例为您提供了有关设备如何与 Amazon IoT Core（这是 Amazon IoT解决方案的一个基础部分）通信的基础知识实践经验。当您的设备能够与 Amazon IoT Core通信时，他们可以将消息传递给Amazon服务及其可以执行操作的其它设备 同样，Amazon 服务和其它设备可以处理消息发回到您的设备所产生的信息。

如果您准备好进一步探索 Amazon IoT Core，请尝试以下教程：

- the section called “发送 Amazon SNS 通知” (p. 182)
- the section called “将设备数据存储在 DynamoDB 表中” (p. 188)
- the section called “使用 Amazon Lambda函数格式化通知” (p. 194)

教程：使用 Amazon IoT Device SDK for Embedded C

此部分描述如何运行 Amazon IoT Device SDK for Embedded C。

本节中的过程

- 步骤 1：安装 Amazon IoT Device SDK for Embedded C (p. 172)
- 步骤 2：配置示例应用 (p. 173)
- 步骤 3：构建并运行示例应用程序 (p. 174)

步骤 1：安装 Amazon IoT Device SDK for Embedded C

Amazon IoT Device SDK for Embedded C通常面向需要优化的 C 语言运行时的资源受限设备。您可以在任何操作系统上使用此 SDK，并将其托管在任何类型的处理器（例如 MCU 和 MPU）上。如果您有更多的可用内存和处理资源，我们建议您使用更高级的 Amazon IoT设备和移动 SDK 之一（例如，C++、Java、JavaScript 和 Python）。

一般而言，Amazon IoT Device SDK for Embedded C适用于以下系统：这些系统使用 MCU 或运行嵌入式操作系统的低端 MPU。对于本部分中的编程示例，我们假定您的设备使用 Linux。

Example

1. 从 GitHub 将 Amazon IoT Device SDK for Embedded C下载到您的设备。

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-embedded-c.git --recurse-submodules
   ```

   这将在当前目录中创建一个名为 `aws-iot-device-sdk-embedded-c`的目录。
2. 前往到该目录并签出最新版本。请参阅 github.com/aws/aws-iot-device-sdk-embedded-C/tags 以获取最新版本标签。

   ```
   cd aws-iot-device-sdk-embedded-c
   git checkout latest-release-tag
   ```

3. 安装 OpenSSL 1.1.0 或更高版本。当通过软件包管理器安装时，OpenSSL 开发库通常被称为“libssl-dev”或“openssl-devel”。

```
sudo apt-get install libssl-dev
```

步骤 2：配置示例应用

Amazon IoT Device SDK for Embedded C 包含供您试用的示例应用程序。为了简单起见，本教程使用的是 `mqtt_demo_mutual_auth`应用程序，此应用程序演示如何连接到 Amazon IoT Core消息代理并订阅和发布到 MQTT 主题。

1. 将您在 Amazon IoT Core 入门 (p. 15)中创建的证书和私有密钥复制到 `build/bin/certificates`目录中。

   Note

   设备和根 CA 证书可能会过期或被吊销。如果您的证书过期或被吊销，则您必须将新的 CA 证书或私有密钥和设备证书复制到您的设备上。

2. 您必须使用个人 Amazon IoT Core终端节点、私有密钥、证书和根 CA 证书配置示例。导航到 `aws-iot-device-sdk-embedded-c/demos/mqtt/mqtt_demo_mutual_auth`目录。

   如果您已安装 Amazon CLI，则可使用此命令查找您的账户端点 URL。

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

   如果您尚未安装 Amazon CLI，请打开 Amazon IoT控制台。在导航窗格中，依次选择 Manage (管理) 和 Things (事物)。为您的设备选择 IoT 事物，然后选择 Interact（交互）。您的终端节点显示在事物详细信息页面的 HTTPS 部分中。

3. 打开 `demo_config.h`文件并更新以下各项的值：

   AWS_IOT_ENDPOINT

   您的私有终端节点。

   CLIENT_CERT_PATH

   您的证书文件路径，例如 `certificates/device.pem.crt`"。

   CLIENT_PRIVATE_KEY_PATH

   您的私有密钥文件名，例如 `certificates/private.pem.key`。

   例如：

```
// Get from demo_config.h
// =================================================
#define AWS_IOT_ENDPOINT                "my-endpoint-ats.iot.us-east-1.amazonaws.com"
#define AWS_MQTT_PORT                   8883
#define CLIENT_IDENTIFIER               "testclient"
#define ROOT_CA_CERT_PATH               "certificates/AmazonRootCA1.crt"
#define CLIENT_CERT_PATH                "certificates/my-device-cert.pem.crt"
#define CLIENT_PRIVATE_KEY_PATH         "certificates/my-device-private-key.pem.key"
// =================================================
```

4. 使用此命令检查您的设备上是否安装了 CMake。

```
cmake --version
```

如果您看到编译器的版本信息，则可以继续下一部分。

如果出现错误或看不到任何信息，则需要使用此命令安装 cmake 软件包。

```
sudo apt-get install cmake
```

再次运行 cmake --version命令，确认 CMake 已安装并且您已准备好继续操作。

5. 使用此命令检查您的设备上是否安装了开发工具。

```
gcc --version
```

如果您看到编译器的版本信息，则可以继续下一部分。

如果出现错误或看不到任何编译器信息，则需要使用此命令安装 build-essential软件包。

```
sudo apt-get install build-essential
```

再次运行 gcc --version命令，确认构建工具已安装并且您已准备好继续操作。

步骤 3：构建并运行示例应用程序

运行 Amazon IoT Device SDK for Embedded C示例应用程序

1. 导航到 aws-iot-device-sdk-embedded-c并创建目录。

```
mkdir build && cd build
```

2. 输入以下 CMake 命令以生成 Makefiles 构建所需的文件。

```
cmake ..
```

3. 输入以下命令以构建可执行应用程序文件。

```
make
```

4. 使用此命令运行 mqtt_demo_mutual_auth应用程序。

```
cd bin
./mqtt_demo_mutual_auth
```

您应该可以看到类似于如下所示的输出内容：

```
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:584] Establishing a TLS session to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com:8883.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1264] Creating an MQTT connection to a2zk5tjv9x07ct-ats.iot.us-west-2.amazonaws.com.
[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt_serializer.c:970] CONNACK session present bit not set.
[INFO] [MQTT] [core_mqtt_serializer.c:912] Connection accepted.
[INFO] [MQTT] [core_mqtt.c:1526] Received MQTT CONNACK successfully from broker.
[INFO] [MQTT] [core_mqtt.c:1792] MQTT connection established with the broker.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1033] MQTT connection successfully established with broker.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1296] A clean MQTT connection is established. Cleaning up all the stored outgoing publishes.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1314] Subscribing to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1097] SUBSCRIBE sent for topic testclient/example/topic to broker.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=3.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:921] Subscribed to the topic testclient/example/topic. with maximum QoS 1.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1358] Sending Publish to the MQTT topic testclient/example/topic.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:1195] PUBLISH sent for topic testclient/example/topic to broker with packet ID 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=2.
[INFO] [MQTT] [core_mqtt.c:1126] Ack packet deserialized with result: MQTTSuccess.
[INFO] [MQTT] [core_mqtt.c:1139] State record updated. New state=MQTTPublishDone.
[INFO] [DEMO] [mqtt_demo_mutual_auth.c:946] PUBACK received for packet id 2.


[INFO] [DEMO] [mqtt_demo_mutual_auth.c:672] Cleaned up outgoing publish packet with packet id 2.


[INFO] [MQTT] [core_mqtt.c:855] Packet received. ReceivedBytes=40.
[INFO] [MQTT] [core_mqtt.c:1015] De-serialized incoming PUBLISH packet: DeserializerResult=MQTTSuccess.
```

设备现已使用 Amazon IoT Device SDK for Embedded C连接到 Amazon IoT。

您还可以使用 Amazon IoT控制台查看示例应用程序正在发布的 MQTT 消息。有关如何在 Amazon IoT控制台中使用 MQTT 客户端的信息，请参阅the section called “使用 Amazon IoT MQTT 客户端查看 MQTT 消息” (p. 59)。

创建将设备数据路由到其他服务的 Amazon IoT规则

这些教程介绍如何使用一些更常见的规则操作创建并测试 Amazon IoT规则。

Amazon IoT 规则会将数据从您的设备发送到其它 Amazon服务。它们侦听特定的 MQTT 消息，格式化消息负载中的数据，并将结果发送到其它 Amazon服务.

我们建议您按照这里显示的顺序尝试这些功能，即使您的目标是使用 Lambda 或更复杂的函数创建一个规则。这些教程是按照从基本到复杂的顺序提供的。它们以递增方式呈现新概念，从而帮助您了解可用于创建没有特定教程的规则操作的概念。

> **Note**
>
> Amazon IoT 规则可帮助您将数据从 IoT 设备发送到其它 Amazon服务。但是，要成功完成此操作，您需要了解要向其发送数据的其它服务的工作知识。虽然这些教程提供了完成任务所需的信息，但您可能会发现在解决方案中使用数据之前，了解有关要向其发送数据的服务的详细信息会很有帮助。其它 Amazon服务的详细解释不在这些教程的范围之内。

教程场景概览

这些教程的场景是定期发布数据的天气传感器设备。在这个虚构系统中有许多这样的传感器设备。但是，本部分中的教程侧重于单个设备，同时展示了如何容纳多个传感器。

本部分中的教程介绍如何使用 Amazon IoT规则借助这个虚构的气象传感器设备系统来完成以下任务，

- 教程：重新发布 MQTT 消息 (p. 176)

  本教程介绍如何将从天气传感器收到的 MQTT 消息重新发布为仅包含传感器 ID 和温度值的消息。它只使用 Amazon IoT Core服务，并演示简单 SQL 查询以及如何使用 MQTT 客户端来测试您的规则。

- 教程：发送 Amazon SNS 通知 (p. 182)

  本教程介绍如何在天气传感器设备的值超过特定值时发送 SNS 消息。它建立在上一教程中所介绍概念的基础上，并添加了如何使用另一个 Amazon服务，Amazon Simple Notification Service (Amazon SNS) 的说明。

  如果您是 Amazon SNS 的新用户，请查看其入门练习，然后再开始本教程。
- 教程：将设备数据存储在 DynamoDB 表中 (p. 188)

  本教程介绍如何将来自气象传感器设备的数据存储在数据库表中。它使用规则查询语句和替代模板来设置目标服务的消息数据的格式，Amazon DynamoDB。

  如果您是的 DynamoDB 用户，请查看其入门练习，然后再开始本教程。
- 教程：使用 Amazon Lambda函数格式化通知 (p. 194)

  本教程介绍如何调用 Lambda 函数来重新格式化设备数据，然后将其作为文本消息发送。它在 Amazon Lambda函数中添加了 Python 脚本和 AmazonSDK 函数，以使用来自气象传感器设备的消息负载数据进行格式化并发送文本消息。

  如果您是 Lambda 的新用户，请查看 Lambda 的入门练习，然后再开始本教程。

Amazon IoT 规则概览

所有这些教程都创建了 Amazon IoT规则。

对于 Amazon IoT规则将数据从设备发送到另一个 Amazon服务，它使用：

- 规则查询语句，由以下内容组成：
  - 一个 SQL SELECT 子句，用于从消息负载中选择数据并设置其格式
  - 标识要使用的消息的主题筛选条件（规则查询语句中的 FROM 对象）
  - 可选条件语句（SQL WHERE 子句），用于指定执行操作的特定条件
- 至少一个规则操作

设备会向主题发布 MQTT 消息。SQL SELECT 语句中的主题筛选条件标识要应用规则的 MQTT 主题。SQL SELECT 语句中指定的字段将来自传入 MQTT 消息负载的数据格式化，以供规则的操作使用。有关规则操作的完整列表，请参阅 Amazon IoT规则操作 (p. 449)。

本部分中的教程

- 教程：重新发布 MQTT 消息 (p. 176)
- 教程：发送 Amazon SNS 通知 (p. 182)
- 教程：将设备数据存储在 DynamoDB 表中 (p. 188)
- 教程：使用 Amazon Lambda函数格式化通知 (p. 194)

教程：重新发布 MQTT 消息

本教程演示如何创建 Amazon IoT规则，该规则在收到指定的 MQTT 消息时发布 MQTT 消息。可以在发布之前通过规则修改传入的消息负载。这样就可以创建针对特定应用程序量身定制的消息，而无需更改设备或固件。您还可以使用规则的筛选功能仅在满足特定条件时发布消息。

通过规则重新发布的消息行为类似于任何其它 Amazon IoT设备或客户端发送的消息。设备可以订阅重新发布的消息，就像订阅任何其它 MQTT 消息主题一样。

您将在本教程中学到的内容：

- 如何在规则查询语句中使用简单的 SQL 查询和函数
- 如何使用 MQTT 客户端来测试 Amazon IoT规则

完成本教程需要大约 30 分钟。

在本教程中，您将：

- 查看 MQTT 主题和 Amazon IoT规则 (p. 177)
- 步骤 1：创建重新发布 MQTT 消息的 Amazon IoT规则 (p. 177)
- 步骤 2：测试您的新规则 (p. 179)
- 步骤 3：查看结果和后续步骤 (p. 182)

在开始本教程之前，请确保您具有：

- 设置您的Amazon Web Services 账户 (p. 16)

  您将需要用到您的 Amazon Web Services 账户和 Amazon IoT控制台以完成本教程。
- 审核 使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)

  请确保您可以使用 MQTT 客户端订阅和发布主题。您将使用 MQTT 客户端在此流程中测试新规则。

查看 MQTT 主题和 Amazon IoT规则

在谈论 Amazon IoT规则之前，它有助于了解 MQTT 协议。在 IoT 解决方案中，MQTT 协议相比于其它网络通信协议（如 HTTP）更具优势，这使得它成为 IoT 设备使用的常用选择。本部分回顾了 MQTT 的关键方面，因为它们适用于本教程。有关如何将 MQTT 与 HTTP 进行比较的信息，请参阅 为设备通信选择协议 (p. 76)。

MQTT 协议

MQTT 协议在其主机上使用发布/订阅通信模式。要发送数据，设备会将主题标识的消息发布到 Amazon IoT 消息代理。要接收来自消息代理的消息，设备通过在订阅请求中向消息代理发送主题筛选条件，来订阅其将接收的主题。Amazon IoT 规则引擎从消息代理接收 MQTT 消息。

Amazon IoT 规则

Amazon IoT 规则由规则查询语句和一个或多个规则操作组成。当 Amazon IoT规则引擎接收 MQTT 消息，则这些元素将按如下方式操作消息。

- 规则查询语句

  规则的查询语句描述了要使用的 MQTT 主题，解释消息负载中的数据，并按类似于常用 SQL 数据库使用的语句的 SQL 语句的描述设置数据的格式。查询语句的结果是发送到规则操作的数据。
- 规则操作

  规则中的每个规则操作都会对规则的查询语句生成的数据进行操作。Amazon IoT 支持许多规则操作 (p. 449)。但是，在本教程中，您将专注于 Republish (p. 509)规则操作，该操作将查询语句的结果发布为带有特定主题的 MQTT 消息。

步骤 1：创建重新发布 MQTT 消息的 Amazon IoT规则

您将在本教程中为 `device/`*`device_id`*`/data`MQTT 主题创建的 Amazon IoT规则，其中 *`device_id`* 是发送消息的设备的 ID。这些主题由主题筛选条件 (p. 95)描述为 `device/+/data`，其中，`+` 是匹配两个正斜杠字符之间的任何字符串的通配符。

当规则收到来自匹配主题的消息时，它会重新发布 device_id和 temperature的值，以 device/data/temp主题发布为新的 MQTT 消息。

例如，具有 device/22/data主题的 MQTT 消息负载如下所示：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

该规则将 temperature值从消息负载中取出，将 device_id从主题中取出，并将其作为 MQTT 消息以 device/data/temp为主题和类似下面这样的消息负载重新发布：

```
{
  "device_id": "22",
  "temperature": 28
}
```

使用此规则，只需要设备 ID 和温度数据的设备会订阅 device/data/temp主题以仅接收该信息。

创建重新发布 MQTT 消息的规则

1. 打开 Amazon IoT控制台的 Rules（规则）页面。
2. 在 Rules（规则）中，选择 Create（创建）并开始创建新规则。
3. 在 Create a rule（创建规则）顶部：
   a. 在 Name（名称）下，输入规则的名称。在本教程中，将其命名为 **republish_temp**。

      请记住，规则名称在您的账户和区域中必须唯一，并且不能包含任何空格 我们在此名称中使用了下划线字符来分隔规则名称中的两个单词。
   b. 在 Description（说明）中，描述规则。

      有意义的描述可以帮助您记住此规则的作用以及您创建它的原因。描述可以根据需要延长，因此请尽可能详细。
4. 在 Create a rule（创建规则）的 Rule query statement（规则查询语句）中：
   a. 在 Using SQL version（使用 SQL 版本）中，选择 **2016-03-23**。
   b. 在 Rule query statement（规则查询语句）编辑框中，输入语句：

      ```
      SELECT topic(2) as device_id, temperature FROM 'device/+/data'
      ```

      本语句：

      - 侦听主题符合 device/+/data主题筛选条件的 MQTT 消息。
      - 从主题字符串中选择第二个元素，并将其分配给 device_id字段。
      - 从消息负载中选择 temperature字段的值，并将其分配给 temperature字段。
5. 在 Set one or more actions（设置一个或多个操作）中：
   a. 要打开此规则的规则操作列表，请选择 Add action（添加操作）。
   b. 在 Select an action（选择操作）中，选择 Republish a message to an Amazon IoTtopic（将消息重新发布至 IoT 主题）。

c. 在操作列表底部，选择 Configure action（配置操作）以打开选定操作的配置页面。

6. 在 Configure action（配置操作）中：

   a. 在 Topic（主题）中输入 **device/data/temp**。这是此规则将发布的消息 MQTT 主题。

   b. 在 Quality of Service（服务质量）中，选择 0 - The message is delivered zero or more times（0- 消息传递零次或多次）。

   c. 在Choose or create a role to grant Amazon IoTaccess to perform this action（选择或创建角色以向 IoT 授予执行此操作的权限）下:

      i. 选择 Create Role(创建角色)。Create a new role（创建新角色）对话框打开。

      ii. 输入描述新角色的名称。在本教程中，请使用 **republish_role**。

      创建新角色时，将创建用于执行规则操作的正确策略并将其附加到新角色。如果更改此规则操作的主题或在其它规则操作中使用此角色，则必须更新该角色的策略以授权新主题或操作。要更新现有角色，请在本部分中选择 Update role（更新角色）。

      iii. 选择 Create Role（创建角色）以创建角色并关闭对话框。

   d. 选择 Add action（添加操作）将操作添加到规则并返回到 Create a rule（创建规则）页。

7. Republish a message to an Amazon IoTtopic（将消息重新发布到 IoT 主题）操作现在列于 Set one or more actions（设置一个或多个操作）。

   在新动作的磁贴中，在 Republish a message to an Amazon IoTtopic（将消息重新发布至 IoT 主题）下，您可以看到重新发布操作将发布到的主题。

   这是您将添加到此规则的唯一规则操作。

8. 在 Create a rule（创建规则）中，向下滚动到底部，然后选择 Create rule（创建规则）以创建规则并完成此步骤。

步骤 2：测试您的新规则

要测试新规则，您将使用 MQTT 客户端发布和订阅此规则所使用的 MQTT 消息。

在新窗口中打开 Amazon IoT控制台中的 MQTT 客户端。这将允许您在不丢失 MQTT 客户端配置的情况下编辑规则。如果您让其转到控制台中的其它页面，MQTT 客户端不会保留任何订阅或消息日志。

如需使用 MQTT 客户端来测试您的规则

1. 在 Amazon IoT控制台中的 MQTT 客户端中，订阅输入主题，在本例中为 `device/+/data`。

   a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择 Subscribe to a topic（订阅主题）。

   b. 在 Subscription topic（订阅主题）中，输入输入主题筛选条件的主题，**device/+/data**。

   c. 将其它字段保留为原定设置。

   d. 选择 Subscribe to topic（订阅主题）。

      在 Subscriptions（订阅）栏，在 Publish to a topic（发布到主题）项下，将会显示 **device/+/data**。

2. 订阅您的规则将发布的主题：`device/data/temp`。

   a. 在 Subscriptions（订阅）中，再次选择 Subscribe to a topic（订阅主题），并在 Subscription topic（订阅主题）中，输入重新发布消息的主题，**device/data/temp**。

   b. 将其它字段保留为原定设置。

   c. 选择 Subscribe to topic（订阅主题）。

      在 Subscriptions（订阅）栏，在 device/+/data 下，将会显示 **device/data/temp**。

3. 使用特定设备 ID **device/22/data**向输入主题发布消息。您无法发布到包含通配符的 MQTT 主题。

a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择 Subscribe to a topic（订阅主题）。
b. 在 Publish（发布）字段中，输入输入主题名称，**device/22/data**。
c. 复制此处显示的示例数据，然后在主题名称下方的编辑框中粘贴示例数据。

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. 要发送您的 MQTT 消息，请选择 Publish to topic（发布到主题）。

4. 查看已发送的消息。

a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，您之前订阅的两个主题旁边有一个绿点。

绿色圆点表示自您上次查看它们以来已收到一条或多条新消息。

b. 在 Subscriptions（订阅）中，选择 device/+/data 来检查消息负载是否与刚刚发布的内容匹配，如下所示：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

c. 在 Subscriptions（订阅）中，选择 device/data/temp 来检查重新发布的消息负载，如下所示：

```
{
  "device_id": "22",
  "temperature": 28
}
```

请注意，`device_id` 值是一个带引号的字符串，`temperature` 值为数字。这是因为 topic()函数从输入消息的主题名称中提取字符串，而 `temperature`值使用输入消息负载中的数值。

如果您想使 `device_id`值为数值，请在规则查询语句中将 `topic(2)`替换为：

```
cast(topic(2) AS DECIMAL)
```

请注意，将 `topic(2)`值设置为数字值仅在主题的该部分仅包含数字字符时才起作用。

5. 如果您看到正确的消息已发布到 device/data/temp 主题，那么您的规则已起作用。请在下一部分中了解有关“重新发布规则”操作的更多信息。

如果您没有看到正确的消息发布到 device/+/data 或者 device/data/temp 主题中，请查看故障排除提示。

重新发布消息规则疑难解答

如果您没有看到期望的结果，可检查以下事项。

- 您收到了一个错误的广告条

  如果在您发布输入消息时出现错误，请先更正该错误。以下步骤可帮助您更正此错误。
- 您未在 MQTT 客户端中看到输入消息

  每次将输入消息发布到 `device/22/data`主题时，如果您如流程中所述订阅了 `device/+/data`主题筛选条件，则消息应当会在 MQTT 客户端中显示。

  要检查的事项

  - 检查您订阅的主题筛选条件

    如果您如流程中所述订阅了输入消息主题，则每次发布输入消息时都应看到该输入消息的副本。

    如果没有看到该消息，请检查您订阅的主题名称，并将其与您发布的主题进行比较。主题名称区分大小写，您订阅的主题必须与发布消息负载的主题相同。
  - 检查消息发布函数

    在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择device/+/data，检查发布消息的主题，然后选择 Publish to topic（发布到主题）。您应该会看到来自主题下方编辑框中的消息负载出现在消息列表中。
- 您未在 MQTT 客户端中看到您的重新发布的消息

  要使规则起作用，规则必须具有授权其接收和重新发布消息的正确策略，并且必须接收消息。

  要检查的事项

  - 检查 MQTT 客户端的 Amazon Web Services 区域和您创建的规则

    您在其中运行 MQTT 客户端的控制台必须位于您所创建规则的同一 Amazon区域。
  - 检查规则查询语句中的输入消息主题

    要使规则起作用，它必须收到一条消息，消息主题名称应与规则查询语句 FROM 子句中的主题筛选条件相匹配。

    检查规则查询语句中主题筛选条件的拼写与 MQTT 客户端中主题的拼写。主题名称区分大小写，消息的主题必须与规则查询语句中的主题筛选条件匹配。
  - 检查输入消息负载的内容

    要使规则起作用，则必须在 SELECT 语句中声明的消息负载中找到数据字段。

    检查规则查询语句 `temperature`字段中的拼写与 MQTT 客户端中消息负载的拼写。字段名称区分大小写，规则查询语句中 `temperature`字段必须与消息负载中的 `temperature`字段相同。

    确保消息负载中的 JSON 文档格式正确。如果 JSON 有任何错误（例如缺少逗号），则规则将无法读取它。
  - 在规则操作中检查重新发布的消息主题

    重新发布规则操作向其发布新消息的主题必须与您在 MQTT 客户端中订阅的主题匹配。

    在控制台中打开您创建的规则，然后检查规则操作将重新发布消息的主题。
  - 检查规则所使用的角色

    规则操作必须具有接收原始主题和发布新主题的权限。

授权规则接收消息数据并重新发布的策略特定于所使用的主题。如果更改用于重新发布消息数据的主题，则必须更新规则操作的角色，以更新其策略来匹配当前主题。

如果怀疑这里出现了问题，请编辑“重新发布”规则操作并创建新角色。规则操作创建的新角色将接收执行这些操作所需的授权。

步骤 3：查看结果和后续步骤

在本教程中：

- 您在规则查询语句中使用了一个简单的 SQL 查询和几个函数来生成新的 MQTT 消息。
- 您创建了重新发布该新消息的规则。
- 您可以使用 MQTT 客户端来测试您的 Amazon IoT规则。

后续步骤

使用此规则重新发布几条消息后，请尝试使用该规则进行实验，以了解更改教程的某些方面如何影响重新发布的消息。以下想法可以帮助您开始操作。

- 在输入消息的主题中更改 *device_id*，并观察重新发布的消息负载中的效果。
- 更改规则查询语句中选定的字段，并观察重新发布的消息负载中的效果。
- 尝试本系列中的下一个教程，了解如何 教程：发送 Amazon SNS 通知 (p. 182)。

本教程中使用的“重新发布规则”操作也可以帮助您调试规则查询语句。例如，您可以将此操作添加到规则中，以查看其规则查询语句如何设置规则操作所使用的数据的格式。

教程：发送 Amazon SNS 通知

本教程演示如何创建 Amazon IoT规则，该规则将 MQTT 消息数据发送到 Amazon SNS 主题，以便它可以作为 SMS 文本消息发送。

在本教程中，您将创建一条规则，以在温度超过规则中设置的值时将消息数据从天气传感器发送到 Amazon SNS 主题的所有订阅者。当报告的温度超过规则设置的值时，规则会检测到，并创建一个新的消息负载，其中仅包含设备 ID、报告的温度和超出的温度限制。规则将新消息负载作为 JSON 文档发送到 SNS 主题，该主题会通知 SNS 主题的所有订阅者。

您将在本教程中学到的内容：

- 如何创建和测试 Amazon SNS 通知
- 如何从 Amazon IoT规则调用 Amazon SNS 通知
- 如何在规则查询语句中使用简单的 SQL 查询和函数
- 如何使用 MQTT 客户端来测试 Amazon IoT规则

完成本教程需要大约 30 分钟。

在本教程中，您将：

- 步骤 1：创建 Amazon SNS 主题，:发送 SMS 文本消息 (p. 183)
- 步骤 2：创建 Amazon IoT规则发送文本消息 (p. 184)
- 步骤 3：测试 Amazon IoT规则和 Amazon SNS 通知 (p. 185)
- 步骤 4：查看结果和后续步骤 (p. 188)

在开始本教程之前，请确保您具有：

- 设置您的Amazon Web Services 账户 (p. 16)

  您将需要用到您的 Amazon Web Services 账户和 Amazon IoT控制台以完成本教程。
- 审核 使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)

  请确保您可以使用 MQTT 客户端订阅和发布主题。您将使用 MQTT 客户端在此流程中测试新规则。
- 审核 Amazon Simple Notification Service

  如果您以前未使用过 Amazon SNS，请查看 设置 Amazon SNS 的访问权限。如果您已完成了其它 Amazon IoT教程，您的 Amazon Web Services 账户应该已经正确配置。

步骤 1：创建 Amazon SNS 主题，:发送 SMS 文本消息

创建发送 SMS 文本消息的 Amazon SNS 主题

1. 创建 Amazon SNS 主题。
   a. 登录 Amazon SNS 控制台。
   b. 在左侧导航窗格中，选择主题。
   c. 在 Topics（主题）页面上，选择 Create topic（创建主题）。
   d. 在 Details（详细信息）中，选择 Standard（标准）类型。预设情况下，控制台会创建 FIFO 主题。
   e. 在 Name（名称）中，输入 SNS 主题名称。在本教程中，请输入 **high_temp_notice**。
   f. 滚动到页面底部，然后选择 Create topic（创建主题）。

      控制台会打开新主题的详细信息页面。
2. 创建 Amazon SNS 订阅。

   Note

   您将在本教程中发送的消息可能会使您在此订阅中使用的电话号码产生短信费用。

   a. 在 high_temp_notice 主题详细信息页面上，选择 Create subscription（创建订阅）。
   b. 在 Create subscripition（创建订阅），在 Details（详细信息）部分中，在 Protocol（协议）列表中，选择 SMS。
   c. 在 Endpoint（端点）中，输入可接收文本消息的电话号码。请务必输入号码，以 +开头，包含国家和地区代码，并且不包括任何其它标点符号字符。
   d. 选择 Create subscription（创建订阅）。
3. 测试 Amazon SNS 通知。
   a. 在 Amazon SNS 控制台中，左侧导航窗格内，选择 Topics（主题）。
   b. 要打开主题的详细信息页面，请在 Topics（主题），在主题列表中，选择 high_temp_notice。
   c. 打开 Publish message to topic（将消息发布到主题）页面中的 high_temp_notice详细信息页面，选择 Publish message（发布消息）。
   d. 在 Publish message to topic（将消息发布到主题），在 Message body（消息正文）部分，Message body to send to the endpoint（要发送到端点的消息正文）中，输入一条简短的消息。
   e. 滚动到页面底部并选择 Publish message（发布消息）。
   f. 在使用您之前创建订阅时使用的号码的电话上，确认已收到消息。

   如果您没有收到测试消息，请仔细检查电话号码和手机的设置。

确保您可以从 Amazon SNS 控制台发布测试消息，然后继续本教程。

步骤 2：创建 Amazon IoT规则发送文本消息

您将在本教程中创建的 Amazon IoT规则将订阅 `device/`*`device_id`*`/data`MQTT 主题，其中 *`device_id`*是发送消息的设备的 ID。这些主题在主题筛选条件中描述为 `device/+/data`，其中，`+` 是匹配两个正斜杠字符之间的任何字符串的通配符。此规则还测试消息负载中的 `temperature`字段值。

当规则收到来自匹配主题的消息时，它会从主题名称中取 *`device_id`*，从消息负载中取 `temperature`值，并为正在测试的限制添加一个常量值，并将这些值作为 JSON 文档发送到 Amazon SNS 通知主题。

例如，来自气象传感器设备编号 32 的 MQTT 消息使用 `device/32/data`主题，并具有如下所示的消息负载：

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

规则的规则查询语句从消息负载中取 `temperature`值，从主题名称中取 *`device_id`*，并添加常量 `max_temperature`值以向 Amazon SNS 主题发送类似于以下内容的消息负载：

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30
}
```

创建 Amazon IoT规则来检测超限温度值并创建要发送到 Amazon SNS 主题的数据：

1. 打开 Amazon IoT控制台的 Rules（规则）页面。
2. 如果这是您的第一条规则，请选择 Create（创建），或者 Create a rule（创建规则）。
3. 在 Create a rule（创建规则）中：
   a. 在 Name（名称）中，输入 **`temp_limit_notify`**。

      请记住，规则名称在您的 Amazon Web Services 账户和区域中必须唯一，并且不能包含任何空格。我们在此名称中使用了下划线字符来分隔规则名称中的单词。
   b. 在 Description（说明）中，描述规则。

      有意义的描述可以更容易地记住此规则的作用以及您创建它的原因。描述可以根据需要延长，因此请尽可能详细。
4. 在 Create a rule（创建规则）的 Rule query statement（规则查询语句）中：
   a. 在 Using SQL version（使用 SQL 版本）中，选择 2016-03-23。
   b. 在 Rule query statement（规则查询语句）编辑框中，输入语句：

```
SELECT topic(2) as device_id,
    temperature as reported_temperature,
    30 as max_temperature
```

```
FROM 'device/+/data'
WHERE temperature > 30
```

本语句：

- 侦听主题与 `device/+/data`主题筛选条件相符，并且 `temperature`值大于 30 的 MQTT 消息。
- 从主题字符串中选择第二个元素，并将其分配给 `device_id`字段。
- 从消息负载中选择 `temperature`字段的值，并将其分配给 `reported_temperature`字段。
- 创建常数值 `30`来表示限制值，并将其赋值给 `max_temperature`字段。

5. 要打开此规则的规则操作列表，请在 Set one or more actions（设置一个或多个操作）中，选择 Add action（添加操作）。
6. 在 Select an action（选择操作）页面上，选择 Send a message as an SNS push notification（将消息发送为 SNS 推送通知）。
7. 要打开选定操作的配置页面，请在操作列表底部选择 Configure action（配置操作）。
8. 在 Configure action（配置操作）中：

   a. 在 SNS target（SNS target）中，选择 Select（选择），找到您名为 high_temp_notice 的 SNS 主题，然后选择 Select（选择）。

   b. 对于 Message format（消息格式），请选择 RAW。

   c. 在 Choose or create a role to grant Amazon IoTaccess to perform this action（选择或创建角色以向 IoT 授予执行此操作的权限）下，选择 Create Role（创建角色）。

   d. 在 Create a new role（创建新角色）中，Name（名称）项下，输入新角色的唯一名称。在本教程中，请使用 **`sns_rule_role`**。

   e. 选择 Create role（创建角色）。

   如果要重复本教程或重复使用现有角色，请选择 Update role（更新角色）然后再继续。这将更新角色的策略文档，以便与 SNS 目标配合使用。
9. 选择 Add action（添加操作），然后返回到 Create a rule（创建规则）页。

   在新操作的磁贴中，在 Send a message as an SNS push notification（将消息作为 SNS 推送通知发送）项下，您可以看到规则将调用的 SNS 主题。

   这是您将添加到此规则的唯一规则操作。
10. 要创建规则并完成此步骤，请在 Create a rule（创建规则）中，向下滚动到底部，然后选择 Create rule（创建规则）。

步骤 3：测试 Amazon IoT规则和 Amazon SNS 通知

要测试新规则，您将使用 MQTT 客户端发布和订阅此规则所使用的 MQTT 消息。

在新窗口中打开 Amazon IoT控制台中的 MQTT 客户端。这将允许您在不丢失 MQTT 客户端配置的情况下编辑规则。如果您让 MQTT 客户端转到控制台中的其它页面，它将不会保留任何订阅或消息日志。

如需使用 MQTT 客户端来测试您的规则

1. 在 Amazon IoT控制台中的 MQTT 客户端中，订阅输入主题，在本例中为 `device/+/data`。

   a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择 Subscribe to a topic（订阅主题）。

   b. 在 Subscription topic（订阅主题）中，输入输入主题筛选条件的主题，**`device/+/data`**。

   c. 将其它字段保留为原定设置。

   d. 选择 Subscribe to topic（订阅主题）。

在 Subscriptions（订阅）栏，在 Publish to a topic（发布到主题）项下，将会显示 **device/+/data**。

2. 使用特定设备 ID **device/32/data**向输入主题发布消息。您无法发布到包含通配符的 MQTT 主题。

   a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择 Subscribe to a topic（订阅主题）。

   b. 在 Publish（发布）字段中，输入输入主题名称，**device/32/data**。

   c. 复制此处显示的示例数据，然后在主题名称下方的编辑框中粘贴示例数据。

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. 选择 Publish to topic（发布到主题）以发布您的 MQTT 消息。

3. 确认文本消息已发送。

   a. 在 MQTT 客户端，Subscriptions（订阅）项下，您之前订阅的主题旁边会有一个绿色圆点。

   绿色圆点表示自上次查看消息以来已收到一条或多条新消息。

   b. 在 Subscriptions（订阅）中，选择 device/+/data 来检查消息负载是否与刚刚发布的内容匹配，如下所示：

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   c. 检查您用于订阅 SNS 主题的电话，并确认消息负载的内容，如下所示：

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

   请注意，`device_id` 值是一个带引号的字符串，`temperature` 值为数字。这是因为 topic()函数从输入消息的主题名称中提取字符串，而 `temperature`值使用输入消息负载中的数值。

   如果您想使 `device_id`值为数值，请在规则查询语句中将 `topic(2)`替换为：

```
cast(topic(2) AS DECIMAL)
```

   请注意，将 `topic(2)`值设置为数字，`DECIMAL` 值仅在主题的该部分仅包含数字字符时才起作用。

4. 尝试发送温度未超过限制的 MQTT 消息。

   a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择 Subscribe to a topic（订阅主题）。

   b. 在 Publish（发布）字段中，输入输入主题名称，**device/33/data**。

   c. 复制此处显示的示例数据，然后在主题名称下方的编辑框中粘贴示例数据。

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. 要发送您的 MQTT 消息，请选择 Publish to topic（发布到主题）。

您应该会看到您在 **device/+/data**订阅中发送的消息。但是，由于温度值低于规则查询语句中的最大温度，因此您应无法收到文本消息。

如果您未看到正确的行为，请检查故障排除提示。

对 SNS 消息规则进行故障排除

如果您没有看到期望的结果，可以检查以下事项。

- 您收到了一个错误的广告条

  如果在您发布输入消息时出现错误，请先更正该错误。以下步骤可帮助您更正此错误。
- 您未在 MQTT 客户端中看到输入消息

  每次将输入消息发布到 device/22/data主题时，如果您如流程中所述订阅了 device/+/data主题筛选条件，则消息应当会在 MQTT 客户端中显示。

  ### 要检查的事项

  - 检查您订阅的主题筛选条件

    如果您如流程中所述订阅了输入消息主题，则每次发布输入消息时都应看到该输入消息的副本。

    如果没有看到该消息，请检查您订阅的主题名称，并将其与您发布的主题进行比较。主题名称区分大小写，您订阅的主题必须与发布消息负载的主题相同。
  - 检查消息发布函数

    在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择device/+/data，检查发布消息的主题，然后选择 Publish to topic（发布到主题）。您应该会看到来自主题下方编辑框中的消息负载出现在消息列表中。
- 您没有收到 SMS 消息

  要使规则起作用，规则必须具有授权其接收消息和发送 SNS 通知的正确策略，并且必须接收消息。

  ### 要检查的事项

  - 检查 MQTT 客户端的 Amazon Web Services 区域和您创建的规则

    您在其中运行 MQTT 客户端的控制台必须位于您所创建规则的同一 Amazon区域。
  - 检查消息负载中的温度值是否超过测试阈值

    如果温度值小于或等于 30（如规则查询语句中定义），则规则将不会执行其任何操作。
  - 检查规则查询语句中的输入消息主题

要使规则起作用，它必须收到一条消息，消息主题名称应与规则查询语句 FROM 子句中的主题筛选条件相匹配。

检查规则查询语句中主题筛选条件的拼写与 MQTT 客户端中主题的拼写。主题名称区分大小写，消息的主题必须与规则查询语句中的主题筛选条件匹配。

- 检查输入消息负载的内容

  要使规则起作用，则必须在 SELECT 语句中声明的消息负载中找到数据字段。

  检查规则查询语句 `temperature`字段中的拼写与 MQTT 客户端中消息负载的拼写。字段名称区分大小写，规则查询语句中 `temperature`字段必须与消息负载中的 `temperature`字段相同。

  确保消息负载中的 JSON 文档格式正确。如果 JSON 有任何错误（例如缺少逗号），则规则将无法读取它。

- 在规则操作中检查重新发布的消息主题

  重新发布规则操作向其发布新消息的主题必须与您在 MQTT 客户端中订阅的主题匹配。

  在控制台中打开您创建的规则，然后检查规则操作将重新发布消息的主题。

- 检查规则所使用的角色

  规则操作必须具有接收原始主题和发布新主题的权限。

  授权规则接收消息数据并重新发布的策略特定于所使用的主题。如果更改用于重新发布消息数据的主题，则必须更新规则操作的角色，以更新其策略来匹配当前主题。

  如果怀疑这里出现了问题，请编辑“重新发布”规则操作并创建新角色。规则操作创建的新角色将接收执行这些操作所需的授权。

步骤 4：查看结果和后续步骤

在本教程中：

- 您创建并测试了 Amazon SNS 通知主题和订阅。
- 您在规则查询语句中使用了简单 SQL 查询和函数来为您的通知创建新消息。
- 您创建了 Amazon IoT规则发送使用您的自定义消息负载的 Amazon SNS 通知。
- 您可以使用 MQTT 客户端来测试您的 Amazon IoT规则。

后续步骤

使用此规则发送了几条文本消息后，请尝试使用它来了解更改教程的某些方面如何影响消息以及发送消息的时间。以下想法可以帮助您开始操作。

- 在输入消息主题中更改 *`device_id`*，并观察文本消息内容中的效果。
- 更改规则查询语句中选定的字段，并观察文本消息内容中的效果。
- 将规则查询语句中的测试更改为测试最低温度而不是最高温度。记得更改 `max_temperature`名称！
- 添加重新发布规则操作，以便在发送 SNS 通知时发送 MQTT 消息。
- 尝试本系列中的下一个教程，了解如何 教程：将设备数据存储在 DynamoDB 表中 (p. 188)。

教程：将设备数据存储在 DynamoDB 表中

本教程演示如何创建 Amazon IoT规则，以将消息数据发送到 DynamoDB 表。

在本教程中，您将创建一条规则来将消息数据从假想气象传感器设备发送到 DynamoDB 表。该规则将为来自许多气象传感器的数据设置格式，以便将其添加到单个数据库表中。

在本教程中您将学到

- 如何创建 DynamoDB 表
- 如何将消息数据从 Amazon IoT规则发送到 DynamoDB 表
- 如何将替换模板用于 Amazon IoT规则
- 如何在规则查询语句中使用简单的 SQL 查询和函数
- 如何使用 MQTT 客户端来测试 Amazon IoT规则

完成本教程需要大约 30 分钟。

在本教程中，您将：

- 步骤 1：为本教程创建 DynamoDB 表 (p. 189)
- 步骤 2：创建 Amazon IoT规则:将数据发送到 DynamoDB 表 (p. 190)
- 步骤 3：测试 Amazon IoT规则和 DynamoDB 表 (p. 191)
- 步骤 4：查看结果和后续步骤 (p. 193)

在开始本教程之前，请确保您具有：

- 设置您的Amazon Web Services 账户 (p. 16)

  您将需要用到您的 Amazon Web Services 账户和 Amazon IoT控制台以完成本教程。
- 审核 使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)

  请确保您可以使用 MQTT 客户端订阅和发布主题。您将使用 MQTT 客户端在此流程中测试新规则。
- 审核 Amazon DynamoDB 概览

  如果您之前没有使用过 DynamoDB，请查看 DynamoDB 入门来熟悉 DynamoDB 的基本概念和操作。

步骤 1：为本教程创建 DynamoDB 表

在本教程中，您将创建一个具有以下属性的 DynamoDB 表，以记录来自虚拟气象传感器设备的数据：

- `sample_time` 是主键，描述了记录样本的时间。
- `device_id` 是排序键，描述了提供示例的设备
- `device_data` 是从设备接收并由规则查询语句格式化的数据

如需为本教程创建 DynamoDB 表

1. 打开 DynamoDB 控制台，然后选择 Create table（创建表格）。
2. 在 Create table（创建表）中：

   a. 在 Table name（表名称）框中输入表名：**`wx_data`**。
   b. 在 Partition key（分区键）中，输入 **`sample_time`**，然后在字段旁边的选项列表中，选择 **`Number`**。
   c. 在 Sort key（排序键）中，输入 **`device_id`**，然后在字段旁边的选项列表中，选择 **`Number`**。
   d. 在页面底部，选择 Create（创建）。

您将在稍后定义 `device_data`，即您配置 DynamoDB 规则操作时。

步骤 2：创建 Amazon IoT规则:将数据发送到 DynamoDB 表

在此步骤中，您将使用规则查询语句格式化来自虚拟气象传感器设备的数据，以便写入数据库表。

从天气传感器设备接收的示例消息负载如下所示：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

对于数据库条目，您将使用规则查询语句将消息负载的结构展平如下：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind_velocity": 22,
  "wind_bearing": 255
}
```

在这个规则中，您还将使用几个 替换模板 (p. 585)。替代模板是用于从函数和消息数据中插入动态值的表达式。

创建 Amazon IoT规则以将数据发送到 DynamoDB 表

1. 打开 Amazon IoT 控制台的规则中心。
2. 要开始在 Rules（规则）中创建新规则，请选择 Create rule（创建规则）。
3. 在 Rule properties（规则属性）中：

   a. 在 Rule name（规则名称）中，输入 **wx_data_ddb**。

      请记住，规则名称在您的Amazon Web Services 账户和区域中必须唯一，并且不能包含任何空格。我们在此名称中使用了下划线字符来分隔规则名称中的两个单词。

   b. 在 Rule description（规则描述）中，描述规则。

      有意义的描述可以更容易地记住此规则的作用以及您创建它的原因。描述可以根据需要延长，因此请尽可能详细。
4. 选择 Next（下一步）以继续。
5. 在 SQL statement（SQL 语句）中：

   a. 在 SQL version（SQL 版本）中，选择 **2016-03-23**。

   b. 在 SQL statement（SQL 语句）编辑框中，输入语句：

      ```
      SELECT temperature, humidity, barometer,
        wind.velocity as wind_velocity,
        wind.bearing as wind_bearing,
      FROM 'device/+/data'
      ```

      本语句：

      - 侦听主题符合 `device/+/data`主题筛选条件的 MQTT 消息。
      - 将 `wind`属性的元素格式化为单个属性。

- 传递 `temperature`、`humidity`、`barometer` 属性，使其保持不变。

6. 选择 Next（下一步）以继续。
7. 在 Rule actions（规则操作）中：
   a. 要打开此规则的规则操作列表，请在 Action 1（操作 1）中，选择 **DynamoDB**。
   b. 在 Table name（表名称）中，选择在上一步中创建的 DynamoDB 表的名称：**wx_data**。

      Partition key type（分区键类型）和 Sort key type（排序键类型）字段将使用 DynamoDB 表中的值填充。
   c. 在分区键中，输入 **sample_time**。
   d. 在分区键值中，输入 **${timestamp()}**。

      这是您将在此规则中使用第一个 替换模板 (p. 585)。它将使用 timestamp 函数返回的值，而不是消息负载中的值。要了解更多信息，请参阅《Amazon IoT Core 开发人员指南》中的时间戳 (p. 577)。
   e. 在 Sort key（排序键）中，输入 **device_id**。
   f. 在排序键值中，输入 **${cast(topic(2) AS DECIMAL)}**。

      这是您将在此规则中使用的第二个 替换模板 (p. 585)。它将第二个元素的值插入到主题名称，即设备的 ID，后面其将投射为十进制值以匹配密钥的数字格式。要了解有关主题的更多信息，请参阅《Amazon IoT Core 开发人员指南》中的主题 (p. 577)。或者，要了解有关强制转换的更多信息，请参阅《Amazon IoT Core 开发人员指南》中的强制转换 (p. 547)。
   g. 在 Write message data to this column (将消息数据写入到此列) 中，输入 **device_data**。

      这将在 DynamoDB 表中创建 `device_data`列。
   h. 将 Operation (操作) 留空。
   i. 在 IAM Role（IAM 角色）中，选择 Create new role（创建新角色）。
   j. 在 Create role（创建角色）对话框中，为 Role name（角色名称）输入 wx_ddb_role。这个新角色将自动包含一个前缀为 “aws-iot-rule” 的策略，该策略将允许该**wx_data_ddb**规则向您创建的**wx_data** DynamoDB 表发送数据。
   k. 在 IAM role（IAM 角色）中，选择 **wx_ddb_role**。
   l. 在页面底部，选择 Next。
8. 在 Review and create（审核和创建）页面的底部，选择 Create（创建）以创建规则。

步骤 3：测试 Amazon IoT规则和 DynamoDB 表

要测试新规则，您将使用 MQTT 客户端发布和订阅此测试中使用的 MQTT 消息。

在新窗口中打开 Amazon IoT控制台中的 MQTT 客户端。这将允许您在不丢失 MQTT 客户端配置的情况下编辑规则。如果您让其转到控制台中的其它页面，MQTT 客户端不会保留任何订阅或消息日志。您还需要一个单独的控制台窗口打开到 Amazon IoT控制台中的 DynamoDB 表中心以查看规则发送的新条目。

如需使用 MQTT 客户端来测试您的规则

1. 在 Amazon IoT控制台中的 MQTT 客户端中，订阅输入主题，`device/+/data`。
   a. 在 MQTT 客户端中，选择 Subscribe to a topic（订阅主题）。
   b. 对于 Topic filter（主题筛选条件）中，输入输入主题筛选条件的主题，**device/+/data**。
   c. 选择 Subscribe。
2. 现在，使用特定设备 ID **device/22/data**向输入主题发布消息。您无法发布到包含通配符的 MQTT 主题。
   a. 在 MQTT 客户端中，选择 Publish to a topic（发布到主题）。

b. 对于 Topic name（主题名称），输入输入主题名称 **device/22/data**。

c. 对于 Message payload（消息负载），输入以下示例数据。

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. 要发布 MQTT 消息，请选择 Publish（发布）。

e. 现在，在 MQTT 客户端中，选择 Subscribe to a topic（订阅主题）。在 Subscribe（订阅）列中，选择 **device/+/data**订阅。确认上一步骤中的示例数据是否显示在此处。

3. 检查以查看您的规则创建的 DynamoDB 表中的行。

a. 在 Amazon IoT 控制台的 DynamoDB 表中心中，选择 wx_data，然后选择 Items（项目）选项卡。

如果您已经在 Items（项目）选项卡上，则可能需要通过选择表格标题右上角的刷新图标来刷新显示。

b. 请注意，表中的 sample_time 值是链接，请打开一个。如果您刚刚发送了第一条消息，那么这将是列表中唯一的消息。

此链接显示表的该行中所有的数据。

c. 展开 device_data 条目以查看规则查询语句生成的数据。

d. 浏览此显示中可用数据的不同表示形式。您也可以编辑此显示中的数据。

e. 查看完此行数据后，要保存所做的任何更改，请选择 Save（保存），或者要退出而不保存任何更改，请选择 Cancel（取消）。

如果您未看到正确的行为，请检查故障排除提示。

故障排除 DynamoDB 规则

如果您没有看到期望的结果，可检查以下事项。

- 您收到了一个错误的广告条

  如果在您发布输入消息时出现错误，请先更正该错误。以下步骤可帮助您更正此错误。

- 您未在 MQTT 客户端中看到输入消息

  每次将输入消息发布到 `device/22/data`主题时，如果您如流程中所述订阅了 `device/+/data`主题筛选条件，则消息应当会在 MQTT 客户端中显示。

  #### 要检查的事项

  - 检查您订阅的主题筛选条件

    如果您如流程中所述订阅了输入消息主题，则每次发布输入消息时都应看到该输入消息的副本。

    如果没有看到该消息，请检查您订阅的主题名称，并将其与您发布的主题进行比较。主题名称区分大小写，您订阅的主题必须与发布消息负载的主题相同。

  - 检查消息发布函数

在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择device/+/data，检查发布消息的主题，然后选择 Publish to topic（发布到主题）。您应该会看到来自主题下方编辑框中的消息负载出现在消息列表中。

- 您在 DynamoDB 表中看不到您的数据

  要做的第一件事是通过选择表格标题右上角的刷新图标来刷新显示。如果没有显示您要查找的数据，请检查以下内容。

  要检查的事项

  - 检查 MQTT 客户端的 Amazon Web Services 区域和您创建的规则

    您在其中运行 MQTT 客户端的控制台必须位于您所创建规则的同一 Amazon区域。

  - 检查规则查询语句中的输入消息主题

    要使规则起作用，它必须收到一条消息，消息主题名称应与规则查询语句 FROM 子句中的主题筛选条件相匹配。

    检查规则查询语句中主题筛选条件的拼写与 MQTT 客户端中主题的拼写。主题名称区分大小写，消息的主题必须与规则查询语句中的主题筛选条件匹配。

  - 检查输入消息负载的内容

    要使规则起作用，则必须在 SELECT 语句中声明的消息负载中找到数据字段。

    检查规则查询语句 `temperature`字段中的拼写与 MQTT 客户端中消息负载的拼写。字段名称区分大小写，规则查询语句中 `temperature`字段必须与消息负载中的 `temperature`字段相同。

    确保消息负载中的 JSON 文档格式正确。如果 JSON 有任何错误（例如缺少逗号），则规则将无法读取它。

  - 检查规则操作中使用的键名和字段名称

    主题规则中使用的字段名称必须与已发布消息的 JSON 消息负载中找到的字段名称匹配。

    打开在控制台中创建的规则，并检查规则操作配置中的字段名称与 MQTT 客户端中使用的字段名称。

  - 检查规则所使用的角色

    规则操作必须具有接收原始主题和发布新主题的权限。

    授权规则接收消息数据和更新 DynamoDB 表的策略特定于所使用的主题。如果更改规则使用的主题或 DynamoDB 表名称，则必须更新规则操作的角色以更新其策略来匹配。

    如果您怀疑存在问题，请编辑规则操作并创建新角色。规则操作创建的新角色将接收执行这些操作所需的授权。

步骤 4：查看结果和后续步骤

使用此规则向 DynamoDB 表发送几条消息后，请尝试对其进行试验，以了解更改教程中的某些方面如何影响写入表的数据。以下想法可以帮助您开始操作。

- 在输入消息主题中更改 *`device_id`*，并观察对数据的影响。您可以使用它来模拟从多个气象传感器接收数据。
- 更改规则查询语句中选定的字段，并观察对数据的影响。您可以使用它筛选存储在表中的数据。
- 添加重新发布规则操作，以便为添加到表中的每一行发送 MQTT 消息。您可以使用它进行调试。

完成本教程后，检查 the section called “使用 Amazon Lambda函数格式化通知” (p. 194)。

教程：使用 Amazon Lambda函数格式化通知

本教程将演示如何将 MQTT 消息数据发送到 Amazon Lambda操作来格式化并发送到另一个 Amazon服务。在本教程中，Amazon Lambda 操作使用 AmazonSDK 将格式化的消息发送到您在介绍如何the section called “发送 Amazon SNS 通知” (p. 182)的教程中创建的 Amazon SNS 主题。

在有关如何 the section called “发送 Amazon SNS 通知” (p. 182)的教程中，则由规则的查询语句生成的 JSON 文档将作为文本消息的正文发送。结果是一条文本消息，示例如下：

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

在本教程中，您将使用 Amazon Lambda规则操作来调用 Amazon Lambda函数，该函数将规则查询语句中的数据格式化为更便于使用的格式，示例如下：

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

您将在此教程中格式化的 Amazon Lambda函数将使用规则查询语句中的数据来设置消息字符串的格式，并调用 AmazonSDK 的 SNS 发布函数来创建通知。

在本教程中您将学到

- 如何创建和测试 Amazon Lambda函数
- 如何使用 Amazon Lambda函数中的 AmazonSDK 来发布 Amazon SNS 通知
- 如何在规则查询语句中使用简单的 SQL 查询和函数
- 如何使用 MQTT 客户端来测试 Amazon IoT规则

完成本教程需要大约 45 分钟。

在本教程中，您将：

- 步骤 1：创建发送文本消息的 Amazon Lambda函数 (p. 195)
- 步骤 2：使用 Amazon Lambda规则操作创建 Amazon IoT规则 (p. 197)
- 步骤 3：测试 Amazon IoT规则和 Amazon Lambda规则操作 (p. 198)
- 步骤 4：查看结果和后续步骤 (p. 201)

在开始本教程之前，请确保您具有：

- 设置您的Amazon Web Services 账户 (p. 16)

  您将需要用到您的 Amazon Web Services 账户和 Amazon IoT控制台以完成本教程。
- 审核 使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)

  请确保您可以使用 MQTT 客户端订阅和发布主题。您将使用 MQTT 客户端在此流程中测试新规则。
- 完成本部分中的其它规则教程

  本教程需要您在如何 the section called “发送 Amazon SNS 通知” (p. 182)的教程中创建的 SNS 通知主题。本教程还假定您已完成本部分中其它与规则相关的教程。
- 审核 Amazon Lambda概览

  如果您未使用过 Amazon Lambda，请查看 Amazon Lambda和 Lambda 入门以学习它的术语和概念。

步骤 1：创建发送文本消息的 Amazon Lambda函数

本教程中的 Amazon Lambda函数会接收规则查询语句的结果，将元素插入到文本字符串中，然后将结果字符串作为通知中的消息发送到 Amazon SNS。

与有关如何 the section called “发送 Amazon SNS 通知” (p. 182)的教程不同，该教程中使用 Amazon IoT规则操作来发送通知，而本教程则通过使用 AmazonSDK 的函数来从 Lambda 函数发送通知。但是，本教程中使用的实际 Amazon SNS 通知主题与您在有关如何the section called “发送 Amazon SNS 通知” (p. 182)的教程中使用的主题相同。

创建发送文本消息的 Amazon Lambda函数

1. 创建新的 Amazon Lambda函数
   a. 在 Amazon Lambda控制台中，选择 Create function（创建函数）。
   b. 在 Create function（创建函数）中，选择 Use a blueprint（使用蓝图）。

      搜索并选择 **hello-world-python**蓝图，然后选择 Configure（配置）。
   c. 在 Basic information（基本信息）中：
      i. 在 Function name（函数名称）中，输入函数的名称，**format-high-temp-notification**。
      ii. 对于 Execution role（执行角色），请选择 Create a new role from Amazonpolicy templates（从亚马逊云科技策略模板创建新角色）。
      iii. 在 Role name（角色名称）中，为新角色输入名称，**format-high-temp-notification-role**。
      iv. 在 Policy templates - optional（策略模板 — 可选）中，搜索并选择 Amazon SNS publish popolicy（Amazon SNS 发布策略）。
      v. 选择 Create function（创建函数）。
2. 修改蓝图代码以格式化并发送 Amazon SNS 通知。
   a. 创建函数后，您应该会看到 detification（format-high-temp-notification详细信息页面）。如果您不这样做，请从 Lambda Functions（Lambda 函数）页打开它。
   b. 在format-high-temp-notification详细信息页面中，选择配置选项卡并滚动到函数代码面板。
   c. 在 Function code（函数代码）窗口中的 Environment（环境）窗格中，选择 Python 文件，`lambda_function.py`。
   d. 在 Function code（函数代码）窗口中，从蓝图中删除所有原始程序代码，并将其替换为此代码。

```
import boto3
#
#   expects event parameter to contain:
#   {
#       "device_id": "32",
#       "reported_temperature": 38,
#       "max_temperature": 30,
#       "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
#   }
#
#   sends a plain text string to be used in a text message
#
#      "Device {0} reports a temperature of {1}, which exceeds the limit of {2}."
#
#   where:
#       {0} is the device_id value
#       {1} is the reported_temperature value
#       {2} is the max_temperature value
#
def lambda_handler(event, context):
```

```
    # Create an SNS client to send notification
    sns = boto3.client('sns')

    # Format text message from data
    message_text = "Device {0} reports a temperature of {1}, which exceeds the
 limit of {2}.".format(
            str(event['device_id']),
            str(event['reported_temperature']),
            str(event['max_temperature'])
        )

    # Publish the formatted message
    response = sns.publish(
            TopicArn = event['notify_topic_arn'],
            Message = message_text
        )

    return response
```

e. 选择 Deploy（部署）。

3. 在新窗口中，从关于如何the section called “发送 Amazon SNS 通知” (p. 182)的教程中查找 Amazon SNS 主题的 Amazon Resource Name (ARN)。

   a. 在新窗口中，打开 Topics page of the Amazon SNS console（Amazon SNS 控制台主题页面）。

   b. 在 Topics（主题）页面上，在 Amazon SNS 主题列表中找到 high_temp_notice通知主题。

   c. 查找 high_temp_notice 通知主题的 ARN，以在下一步中使用。

4. 为您的 Lambda 函数创建一个测试用例。

   a. 在控制台的 Lambda Func ti on 页面，在format-high-temp-notification详细信息页面，选择页面右上角 “选择测试事件”（尽管看起来已禁用），然后选择 “配置测试事件”。

   b. 在 Configure test event（配置测试事件）中，选择 Create new test event（创建新测试事件）。

   c. 在 Event name（事件名称）中，输入 **`SampleRuleOutput`**。

   d. 在 Event name（事件名称）下方的 JSON 编辑器中，请粘贴此示例 JSON 文档。这是您的 Amazon IoT规则将发送到 Lambda 函数的示例内容。

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

   e. 请参阅具有 high_temp_notice通知主题的 ARN 的窗口并复制 ARN 值。

   f. 在 JSON 编辑器中将 `notify_topic_arn`值替换成您通知主题中的 ARN。

      保持此窗口打开状态，以便在创建 Amazon IoT规则时再次使用该 ARN 值。

   g. 选择 Choose（创建）。

5. 使用示例数据测试函数。

   a. 在 Test（format-high-temp-notification详细信息）页面，在页面右上角，确认 SampleRuleOutputTest（测试）按钮旁。如果没有，请从可用测试事件列表中选择它。

   b. 要将示例规则输出消息发送到您的函数，请选择 Test（测试）。

如果函数和通知都有效，您将在订阅通知的手机上收到一条文本消息。

如果您在手机上没有收到文本消息，请检查操作结果。在 Function code（函数代码）面板中的 Execution result（执行结果）选项卡上，查看响应以查找发生的任何错误。请勿继续执行下一步，直到您的函数可以将通知发送到您的手机。

步骤 2：使用 Amazon Lambda规则操作创建 Amazon IoT规则

在此步骤中，您将使用规则查询语句格式化来自虚拟气象传感器设备的数据，以发送到 Lambda 函数，该函数将格式化并发送文本消息。

从天气设备接收的示例消息负载如下所示：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

在此规则中，您将使用规则查询语句为 Lambda 函数创建消息负载，如下所示：

```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

这包含 Lambda 函数格式化和发送正确文本消息所需的所有信息。

创建一个 Amazon IoT规则来调用 Lambda 函数

1. 打开 Amazon IoT控制台的 Rules（规则）页面。
2. 要在 Rule（规则）中创建新规则，请选择 Create（创建）。
3. 在 Create a rule（创建规则）顶部：

   a. 在 Name（名称）中，输入规则的名称，**wx_friendly_text**。

      请记住，规则名称在您的Amazon Web Services 账户和区域中必须唯一，并且不能包含任何空格。我们在此名称中使用了下划线字符来分隔规则名称中的两个单词。

   b. 在 Description（说明）中，描述规则。

      有意义的描述可以更容易地记住此规则的作用以及您创建它的原因。描述可以根据需要延长，因此请尽可能详细。
4. 在 Create a rule（创建规则）的 Rule query statement（规则查询语句）中：

   a. 在 Using SQL version（使用 SQL 版本）中，选择 **2016-03-23**。

   b. 在 Rule query statement（规则查询语句）编辑框中，输入语句：

```
SELECT
  cast(topic(2) AS DECIMAL) as device_id,
  temperature as reported_temperature,
  30 as max_temperature,
  'arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice' as notify_topic_arn
FROM 'device/+/data' WHERE temperature > 30
```

本语句：

- 侦听主题与 `device/+/data`主题筛选条件相符，并且 `temperature`值大于 30 的 MQTT 消息。
- 从主题字符串中选择第二个元素，将其转换为十进制数，然后将其分配给 `device_id`字段。
- 从消息负载中选择 `temperature`字段的值，并将其分配给 `reported_temperature`字段。
- 创建常数值，`30`，以表示限制值，并将其分配给 `max_temperature`字段。
- 为 `notify_topic_arn`字段创建常数值。

c. 请参阅具有 high_temp_notice 通知主题的 ARN 的窗口并复制 ARN 值。

d. 使用通知主题的 ARN 替换 ARN 值 (*arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice*)。

5. 在 Set one or more actions（设置一个或多个操作）中：

a. 要打开此规则的规则操作列表，请选择 Add action（添加操作）。

b. 在 Select an action（选择操作）中，选择 Send a message to a Lambda function（向 Lambda 函数发送消息）。

c. 要打开选定操作的配置页面，请在操作列表底部选择 Configure action（配置操作）。

6. 在 Configure action（配置操作）中：

a. 在 Function name（函数名称）中，选择 Select（选择）。

b. 选择format-high-temp-notification。

c. 在 Configure action（配置操作）底部，选择 Add action（添加操作）。

d. 要创建规则，请在 Create a rule（创建一条规则）底部，选择 Create rule（创建规则）。

步骤 3：测试 Amazon IoT规则和 Amazon Lambda规则操作

要测试新规则，您将使用 MQTT 客户端发布和订阅此规则所使用的 MQTT 消息。

在新窗口中打开 Amazon IoT控制台中的 MQTT 客户端。现在，您可以在不丢失 MQTT 客户端配置的情况下编辑规则。如果您离开 MQTT 客户端并转到控制台中的其它页面，则将丢失您的订阅或消息日志。

如需使用 MQTT 客户端来测试您的规则

1. 在 Amazon IoT控制台中的 MQTT 客户端中，订阅输入主题，在本例中为 `device/+/data`。

a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择 Subscribe to a topic（订阅主题）。

b. 在 Subscription topic（订阅主题）中，输入输入主题筛选条件的主题，**`device/+/data`**。

c. 将其它字段保留为原定设置。

d. 选择 Subscribe to topic（订阅主题）。

在 Subscriptions（订阅）栏，在 Publish to a topic（发布到主题）项下，将会显示 **`device/+/data`**。

2. 使用特定设备 ID **`device/32/data`**向输入主题发布消息。您无法发布到包含通配符的 MQTT 主题。

a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择 Subscribe to a topic（订阅主题）。

b. 在 Publish（发布）字段中，输入输入主题名称，**`device/32/data`**。

c. 复制此处显示的示例数据，然后在主题名称下方的编辑框中粘贴示例数据。

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
```

```
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

d. 如需发布 MQTT 消息，请选择 Publish to topic（发布到主题）。

3. 确认文本消息已发送。

   a. 在 MQTT 客户端，Subscriptions（订阅）项下，您之前订阅的主题旁边会有一个绿色圆点。

      绿色圆点表示自上次查看消息以来已收到一条或多条新消息。

   b. 在 Subscriptions（订阅）中，选择 device/+/data 来检查消息负载是否与刚刚发布的内容匹配，如下所示：

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   c. 检查您用于订阅 SNS 主题的电话，并确认消息负载的内容，如下所示：

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

      如果您更改消息主题中的主题 ID 元素，请记住，将 `topic(2)`值投射为数字值仅在消息主题中的元素仅包含数字字符时才起作用。

4. 尝试发送温度未超过限制的 MQTT 消息。

   a. 在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择 Subscribe to a topic（订阅主题）。

   b. 在 Publish（发布）字段中，输入输入主题名称，**`device/33/data`**。

   c. 复制此处显示的示例数据，然后在主题名称下方的编辑框中粘贴示例数据。

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

   d. 要发送您的 MQTT 消息，请选择 Publish to topic（发布到主题）。

      您应在 **`device/+/data`**订阅中看到您发送的消息；但是，由于温度值低于规则查询语句中的最大温度，因此您不应收到文本消息。

      如果您未看到正确的行为，请检查故障排除提示。

对您的 Amazon Lambda规则和通知进行故障排除

如果您没有看到期望的结果，可以检查以下事项。

- 您收到了一个错误的广告条

  如果在您发布输入消息时出现错误，请先更正该错误。以下步骤可帮助您更正此错误。

- 您未在 MQTT 客户端中看到输入消息

  每次将输入消息发布到 `device/32/data`主题时，如果您如流程中所述订阅了 `device/+/data`主题筛选条件，则消息应当会在 MQTT 客户端中显示。

  要检查的事项

  - 检查您订阅的主题筛选条件

    如果您如流程中所述订阅了输入消息主题，则每次发布输入消息时都应看到该输入消息的副本。

    如果没有看到该消息，请检查您订阅的主题名称，并将其与您发布的主题进行比较。主题名称区分大小写，您订阅的主题必须与发布消息负载的主题相同。

  - 检查消息发布函数

    在 MQTT 客户端中，在 Subscriptions（订阅）项下，选择device/+/data，检查发布消息的主题，然后选择 Publish to topic（发布到主题）。您应该会看到来自主题下方编辑框中的消息负载出现在消息列表中。

- 您没有收到 SMS 消息

  要使规则起作用，规则必须具有授权其接收消息和发送 SNS 通知的正确策略，并且必须接收消息。

  要检查的事项

  - 检查 MQTT 客户端的 Amazon Web Services 区域和您创建的规则

    您在其中运行 MQTT 客户端的控制台必须位于您所创建规则的同一 Amazon区域。

  - 检查消息负载中的温度值是否超过测试阈值

    如果温度值小于或等于 30（如规则查询语句中定义），则规则将不会执行其任何操作。

  - 检查规则查询语句中的输入消息主题

    要使规则起作用，它必须收到一条消息，消息主题名称应与规则查询语句 FROM 子句中的主题筛选条件相匹配。

    检查规则查询语句中主题筛选条件的拼写与 MQTT 客户端中主题的拼写。主题名称区分大小写，消息的主题必须与规则查询语句中的主题筛选条件匹配。

  - 检查输入消息负载的内容

    要使规则起作用，则必须在 SELECT 语句中声明的消息负载中找到数据字段。

    检查规则查询语句 `temperature`字段中的拼写与 MQTT 客户端中消息负载的拼写。字段名称区分大小写，规则查询语句中 `temperature`字段必须与消息负载中的 `temperature`字段相同。

    确保消息负载中的 JSON 文档格式正确。如果 JSON 有任何错误（例如缺少逗号），则规则将无法读取它。

  - 查看 Amazon SNS 通知

    在 步骤 1：创建 Amazon SNS 主题，:发送 SMS 文本消息 (p. 183)中，请参阅步骤 3，该步骤介绍如何测试 Amazon SNS 通知并测试通知以确保通知有效。

  - 检查 Lambda 函数

    在 步骤 1：创建发送文本消息的 Amazon Lambda函数 (p. 195)中，请参阅步骤 5，该步骤介绍如何使用测试数据测试 Lambda 函数并测试 Lambda 函数。

  - 检查规则所使用的角色

规则操作必须具有接收原始主题和发布新主题的权限。

授权规则接收消息数据并重新发布的策略特定于所使用的主题。如果更改用于重新发布消息数据的主题，则必须更新规则操作的角色，以更新其策略来匹配当前主题。

如果怀疑这里出现了问题，请编辑“重新发布”规则操作并创建新角色。规则操作创建的新角色将接收执行这些操作所需的授权。

步骤 4：查看结果和后续步骤

在本教程中：

- 您创建了 Amazon IoT规则来调用 Lambda 函数，该函数发送使用您的自定义消息负载的 Amazon SNS 通知。
- 您在规则查询语句中使用了简单的 SQL 查询和函数来为 Lambda 函数创建新的消息负载。
- 您可以使用 MQTT 客户端来测试您的 Amazon IoT规则。

后续步骤

使用此规则发送了几条文本消息后，请尝试使用它来了解更改教程的某些方面如何影响消息以及发送消息的时间。以下想法可以帮助您开始操作。

- 在输入消息主题中更改 *device_id*，并观察文本消息内容中的效果。
- 更改规则查询语句中选定的字段，更新 Lambda 函数以在新消息中使用这些字段，并观察文本消息内容中的效果。
- 将规则查询语句中的测试更改为测试最低温度而不是最高温度。更新 Lambda 函数以格式化新消息，并记得更改 `max_temperature`名称。
- 要了解有关如何查找在开发和使用 Amazon IoT 规则过程中可能会出现的错误，请参阅 监控 Amazon IoT (p. 394)。

在设备处于离线状态时使用设备影子保持设备状态

这些教程介绍如何使用 Amazon IoTDevice Shadow 服务来存储和更新设备的状态信息。影子文档是 JSON 文档，根据设备、本地应用程序或服务发布的消息显示设备状态的更改。在本教程中，影子文档将显示灯泡颜色的变化。这些教程还显示影子如何存储此信息，即使设备与互联网断开连接，并在设备重新联机并请求此信息时将最新状态信息传递回设备。

我们建议您按照这里显示的顺序尝试这些教程，从需要创建的 Amazon IoT资源以及必要的硬件设置开始，这也有助于您逐步学习这些概念。这些教程展示了如何配置和连接 Raspberry Pi 设备，以便与 Amazon IoT 搭配使用。如果您没有所需的硬件，您可以按照这些教程将它们适配于您选择的设备或使用 Amazon EC2 创建虚拟设备 (p. 38)。

教程场景概览

这些教程的场景是一个本地应用程序或服务，该应用程序或服务可更改灯泡的颜色，并将其数据发布到预留的影子主题。这些教程类似于交互式入门教程 (p. 17)中描述的 Device Shadow 功能，并在Raspberry Pi设备上实现。本部分中的教程侧重于单个经典影子，同时将展示如何容纳已命名的影子或多个设备。

以下教程将帮助您了解如何使用 Amazon IoTDevice Shadow 服务。

- 教程：准备 Raspberry Pi 运行影子应用程序 (p. 203)

  本教程介绍如何设置 Raspberry Pi 设备，用于连接 Amazon IoT。您还将创建一个 Amazon IoT策略文档和事物资源，下载证书，然后将策略附加到该事物资源。完成本教程需要大约 30 分钟。

- 教程：安装设备软件开发包并运行 Device Shadow 示例应用程序 (p. 207)

  本教程介绍如何安装所需的工具、软件和 Amazon IoTDevice SDK for Python，然后运行示例影子应用程序。本教程基于 连接 Raspberry Pi 或其他设备 (p. 51)中介绍的概念所打造，完成需要 20 分钟。
- 教程：示例应用程序及 MQTT 测试客户端与 Device Shadow 交互 (p. 213)

  本教程介绍了如何使用 `shadow.py`示例应用程序和 Amazon IoT控制台来观察 Amazon IoTDevice Shadow 和灯泡状态变化之间的交互。本教程还介绍了如何将 MQTT 消息发送到 Device Shadow 的预留主题。完成本教程需要大约 45 分钟。

Amazon IoT Device Shadow 概览

Device Shadow 是由您在 Amazon IoT注册表中创建的事物资源 (p. 239)所管理的设备的持续性虚拟化展示。设备的影子是用于存储和检索设备的当前状态信息的 JSON 或 JavaScript 注释文档。您可以使用该影子通过 MQTT 主题或 HTTP REST APIs 获取和设置设备的状态，无论该设备是否连接到互联网。

影子文档包含一个 `state`属性，以描述设备状态的以下方面。

- `desired`：应用程序更新 `desired`对象以指定设备属性的所需状态。
- `reported`：设备在 `reported`对象中报告其当前状态。
- `delta`：Amazon IoT 在 `delta`对象中报告所需的状态和报告的状态之间的差异。

以下为示例影子状态文档：

```
{
  "state": {
    "desired": {
      "color": "green"
      },
    "reported": {
      "color": "blue"
      },
    "delta": {
      "color": "green"
      }
   }
}
```

要更新设备的影子文档，可以使用预留的 MQTT 主题 (p. 105)，Device Shadow REST API (p. 616)支持 `GET`、`UPDATE`，和 `DELETE`操作，以及 Amazon IoTCLI。

在前面的示例中，假设您希望将 `desired`颜色改为 `yellow`。要执行此操作，请将请求发送到 UpdateThingShadow (p. 617)API 或将消息发布到更新 (p. 622)主题，`$aws/things/THING_NAME/shadow/update`。

```
{
  "state": {
    "desired": {
      "color": yellow
    }
  }
}
```

更新仅影响请求中指定的字段。成功更新 Device Shadow 后，Amazon IoT 将发布新 `desired`状态至 `delta`主题。`$aws/things/THING_NAME/shadow/delta`在这种情况下，影子文档如下所示：

```
{
```

```
  "state": {
    "desired": {
      "color": yellow
    },
    "reported": {
      "color": green
    },
    "delta": {
      "color": yellow
      }
  }
}
```

然后，新状态将使用 Update主题 $aws/things/THING_NAME/shadow/update报告给 Amazon IoTDevice Shadow，其中包含以下 JSON 消息：

```
{
  "state": {
    "reported": {
      "color": yellow
    }
  }
}
```

如果要获取当前状态信息，请将请求发送至 GetThingShadow (p. 616)API 或将 MQTT 消息发布到 Get (p. 620)（获取）主题，$aws/things/THING_NAME/shadow/get。

有关使用 Device Shadow 服务的更多信息，请参阅 Amazon IoT Device Shadow 服务 (p. 593)。

有关在设备、应用程序和服务中使用 Device Shadow 的更多信息，请参阅 在设备中使用影子 (p. 596)和 在应用程序和服务中使用影子 (p. 598)。

有关与 Amazon IoT影子交互的信息，请参阅 与影子交互 (p. 610)。

有关 MQTT 预留主题和 HTTP REST API 的信息，请参阅 Device Shadow MQTT 主题 (p. 620)和 Device Shadow REST API (p. 616)。

教程：准备 Raspberry Pi 运行影子应用程序

本教程演示如何设置和配置 Raspberry Pi 设备，并创建设备连接和交换 MQTT 消息所需的 Amazon IoT资源。

> Note
>
> 如果您计划 the section called “使用 Amazon EC2 创建虚拟设备” (p. 38)，您可以跳过此页面并继续执行 the section called “配置您的设备” (p. 37)。创建虚拟事物时，您将创建这些资源。如果您想使用不同的设备，而不是 Raspberry Pi，可以尝试按照这些教程进行操作，将它们调整到您选择的设备。

在本教程中，您将学习如何：

- 设置 Raspberry Pi 设备，并进行配置，以与 Amazon IoT搭配使用。
- 创建 Amazon IoT策略文档，该文档将授权您的设备与 Amazon IoT服务交互。
- 在 Amazon IoTX.509 设备证书中创建事物资源，然后附加策略文档。

事物是您的设备在 Amazon IoT注册表中的虚拟展示。该证书将对您的设备进行身份验证以访问 Amazon IoTCore，并且策略文档授权您的设备与 Amazon IoT交互。

如何运行本教程？

要为 Device Shadow 运行 `shadow.py`示例应用程序，您将需要一个连接到 Amazon IoT的 Raspberry Pi 设备。我们建议您按照此处显示的顺序遵循本教程，首先设置 Raspberry Pi 及其配件，然后创建策略并将策略附加到您创建的事物资源。然后，您可以遵循此教程，使用 Raspberry Pi 支持的图形用户界面 (GUI) 在设备的 Web 浏览器上打开 Amazon IoT控制台，这也使您可以更轻松地将证书直接下载到您的 Raspberry Pi，以便连接到 Amazon IoT。

在开始本教程之前，请确保您具有：

- 一个 Amazon Web Services 账户。如果您没有账户，请完成 设置您的Amazon Web Services 账户 (p. 16)中介绍的步骤然后继续操作。您将需要用到您的 Amazon Web Services 账户和 Amazon IoT控制台以完成本教程。
- Raspberry Pi 及其必要的配件。您将需要：
  - Raspberry Pi 3 Model B 或更新型号。本教程可能适用于早期版本的 Raspberry Pi，但我们还没有测试过。
  - Raspberry Pi OS (32 位)或更高版本。我们始终建议使用最新版本的 Raspberry Pi OS。早期版本的操作系统可能有用，但我们还没有测试过。
  - 以太网或 Wi-Fi 连接。
  - 键盘、鼠标、显示器、电缆和电源。

完成本教程需要大约 30 分钟。

步骤 1：设置和配置 Raspberry Pi 设备

在本部分中，我们将配置一个 Raspberry Pi 设备，用于 Amazon IoT。

> **Important**
>
> 将这些指令用于其它设备和操作系统可能会非常困难。您需要充分了解您的设备，以便能够解释这些说明并将它们应用到您的设备上。如果遇到困难，可以尝试使用其它设备选项之一作为替代方案，例如 使用 Amazon EC2 创建虚拟设备 (p. 38)或者 将您的 Windows 或 Linux PC 或 Mac 用作 Amazon IoT 设备 (p. 45)。

您需要配置您的 Raspberry Pi，以便它可以启动操作系统 (OS)，连接到互联网，并允许您在命令行界面与它进行交互。您还可以使用 Raspberry Pi 支持的图形用户界面 (GUI) 打开 Amazon IoT控制台并运行本教程的其余部分。

要设置 Raspberry Pi

1. 将 SD 卡插入 Raspberry Pi 上的 MicroSD 卡槽。有些 SD 卡预装了一个安装管理器，将为您显示在启动主板后安装操作系统的菜单。您还可以使用 Raspberry Pi 在您的卡上安装操作系统。
2. 将 HDMI TV 或显示器 Connect 到 HDMI 电缆，然后连接到 Raspberry Pi 的 HDMI 端口。
3. 将键盘和鼠标连接到 Raspberry Pi 的 USB 端口，然后插入电源适配器以启动主板。

Raspberry Pi 启动后，如果 SD 卡预先加载了安装管理器，则会出现一个菜单来帮助您安装操作系统。如果您在安装操作系统时遇到问题，请尝试以下步骤。有关设置 Raspberry Pi 的更多信息，请参阅设置您的 Raspberry Pi。

如果您在设置 Raspberry Pi 时遇到问题：

- 在启动主板之前，请检查是否插入了 SD 卡。如果在启动主板后插入 SD 卡，则可能不会显示安装菜单。
- 确保电视或显示器已打开，并且选择了正确的输入信号源。
- 确保您使用的是兼容 Raspberry Pi 的软件。

安装操作系统后，配置 Raspberry Pi 操作系统后，打开 Raspberry Pi 的网络浏览器并导航到 Amazon IoT Core控制台继续本教程中的其余步骤。

如果您能打开 Amazon IoT Core控制台，您的 Raspberry Pi 已经准备就绪。可以继续 the section called “在 Amazon IoT中预置设备 ” (p. 205)。

如果您遇到问题或需要其它帮助，请参阅为您的 Raspberry Pi 获取帮助。

教程：在 Amazon IoT中预置设备

本节将创建教程将使用的 Amazon IoT Core资源。

在 Amazon IoT中配置设备的步骤

- 步骤 1：创建 Device Shadow 的 Amazon IoT策略 (p. 205)
- 步骤 2：创建事物资源并将策略附加到事物 (p. 206)
- 步骤 3：查看结果和后续步骤 (p. 207)

步骤 1：创建 Device Shadow 的 Amazon IoT策略

X.509 证书使用 Amazon IoT Core对您的设备进行身份验证。Amazon IoT 策略附加到允许设备执行 Amazon IoT操作（如订阅或发布 Device Shadow 服务使用的 MQTT 预留主题）的证书。您的设备在与 Amazon IoT Core连接并向其发送消息时会提供其证书。

在此流程中，您将创建一个策略以允许您的设备执行运行示例程序所必要的 Amazon IoT操作。建议您创建一个策略以仅授予执行任务所需的许可。您可以先创建 Amazon IoT策略，然后将其附加到稍后将创建的设备证书。

创建 Amazon IoT策略

1. 在左侧菜单中，选择 Secure（安全），然后选 Policies（策略）。如果您的账户已有策略，请选择 Create（创建），否则，在 You don’t have a policy yet（您还没有策略）页面上，选择 Create a policy（创建策略）。
2. 在 Create a policy（创建策略）页面上：

   a. 在 Name（名称）字段，输入策略的名称（例如 **`My_Device_Shadow_policy`**）。请勿在策略名称中使用个人身份信息。

   b. 在策略文档中，您描述了授予设备发布和订阅 MQTT 保留主题权限的连接、订阅、接收和发布操作。

      复制以下策略并将其粘贴到策略文档中。将 `thingname`替换为您要创建的事物名称（例如 `My_light_bulb`），将 `region`替换为您正在使用这些服务的 Amazon IoT区域，将 `account`替换为您的 Amazon Web Services 账户数量。有关 Amazon IoT策略的更多信息，请参阅 Amazon IoT Core 策略 (p. 299)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/get",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/update"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
```

```
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/get/
accepted",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/get/
rejected",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/update/
accepted",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/update/
rejected",
        "arn:aws:iot:region:account:topic/$aws/things/thingname/shadow/update/
delta"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/get/
accepted",
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/get/
rejected",
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/
update/accepted",
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/
update/rejected",
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingname/shadow/
update/delta"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:region:account:client/test-*"
    }
  ]
}
```

步骤 2：创建事物资源并将策略附加到事物

连接到 Amazon IoT的设备在 Amazon IoT注册表中由事物资源表示。事物资源表示特定设备或逻辑实体，如本教程中的灯泡。

要了解如何在 Amazon IoT创建事物，请执行 创建一个事物对象 (p. 36)中所述的步骤。以下是您在遵循该教程中的步骤时需要注意的几个关键事项：

1. 选择 Create a single thing（创建单个事物），并在 Name（名称）字段中，输入 与您在之前创建策略时指定的 `thingname`相同的事物名称（例如，`My_light_bulb`)。

   事物名称一旦创建，便不可更改。如果您给它一个不同于 `thingname`的名字，请以 `thingname`为名称创建新的事物，并删除旧的事物。

   > **Note**
   >
   > 请勿在事物名称中使用个人身份信息。事物名称可以出现在未加密的通信和报告中。

2. 我们建议您将 Certificate created!（已创建证书！）页面的每个证书文件下载到您可以轻松找到的位置。您需要安装这些文件才能运行示例应用程序。

   建议您将文件下载到 Raspberry Pi 中您的 home目录下的 `certs`子目录中，并使用一个更简单的名称命名各个文件，如下表所示。

证书文件名

文件	文件路径
根 CA 证书	`~/certs/Amazon-root-CA-1.pem`
设备证书	`~/certs/device.pem.crt`
私有密钥	`~/certs/private.pem.key`

3. 激活证书以启用与 Amazon IoT的连接后，请选择 Attach a policy（附加策略）并确保将您之前创建的策略（例如，**`My_Device_Shadow_policy`**）附加到事物。

   创建事物后，您可以看到您的事物资源在 Amazon IoT控制台中的事物列表里显示。

步骤 3：查看结果和后续步骤

在本教程中，您学习了如何：

- 设置和配置 Raspberry Pi 设备。
- 创建 Amazon IoT策略文档，以授权您的设备与 Amazon IoT服务交互。
- 创建事物资源和关联的 X.509 设备证书，并将策略文档附加到其中。

后续步骤

您现在可以安装 Amazon IoTDevice SDK for Python，运行 `shadow.py`示例应用程序，然后使用 Device Shadow 控制状态。有关如何使用该教程的更多信息，请参阅 教程：安装设备软件开发包并运行 Device Shadow 示例应用程序 (p. 207)。

教程：安装设备软件开发包并运行 Device Shadow 示例应用程序

本部分介绍如何安装所需的软件和 Amazon IoTDevice SDK for Python，并运行 `shadow.py`示例应用程序来编辑影子文档并控制影子的状态。

在本教程中，您将学习如何：

- 使用已安装的软件和 Amazon IoTDevice SDK for Python 以运行示应用程序。
- 了解使用示例应用程序输入值如何在 Amazon IoT控制台发布预期的值。
- 查看 `shadow.py`示例应用程序以及它如何使用 MQTT 协议来更新影子的状态。

在运行本教程之前：

您必须已设置好您的 Amazon Web Services 账户，配置好您的 Raspberry Pi 设备，并已创建 Amazon IoT事物和策略，授予设备发布和订阅 MQTT 预留的 Device Shadow 服务主题的权限。有关更多信息，请参阅教程：准备 Raspberry Pi 运行影子应用程序 (p. 203)。

您必须还安装了 Git、Python 和 Amazon IoTDevice SDK for Python。本教程基于教程 连接 Raspberry Pi 或其他设备 (p. 51)中介绍的概念执行。如果您尚未尝试该教程，我们建议您按照该教程中描述的步骤安装证书文件和 Device SDK，然后返回本教程运行 `shadow.py`示例应用程序。

在本教程中，您将：

- 步骤 1：运行 shadow.py 示例应用程序 (p. 208)
- 步骤 2：查看 shadow.py 设备软件开发包示例应用 (p. 210)
- 步骤 3：借助 shadow.py示例应用程序排查问题 (p. 211)

- 步骤 4：查看结果和后续步骤 (p. 212)

完成本教程需要大约 20 分钟。

步骤 1：运行 shadow.py 示例应用程序

在运行 shadow.py示例应用程序前，除了安装的证书文件的名称和位置之外，还需要以下信息。

应用程序参数值

参数	在何处查找值
your-iot-thing-name	您之前在 the section called “步骤 2：创建事物资源并将策略附加到事物” (p. 206)中创建的 Amazon IoT事物的名称。 要查找此值，请在 Amazon IoT控制台中，选择 Manage（管理），然后选择 Things（事物）。
your-iot-endpoint	*your-iot-endpoint* 值的格式为：*endpoint_id*-ats.iot.*region*.amazonaws.com，例如，a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com。要查找此值： 1. 在 Amazon IoT控制台中，选择 Manage（管理），然后选择 Things（事物）。 2. 选择您为您的设备创建的 IoT 事物，您在之前使用过的 My_light_bulb，然后选择 Interact（交互）。您的端点会显示在事物详细信息页面的 HTTPS 部分中。

安装并运行示例应用程序

1. 导航到示例应用程序目录。

```
cd ~/aws-iot-device-sdk-python-v2/samples
```

2. 在命令行窗口中，按照指示替换 *your-iot-endpoint* 和 *your-iot-thing-name* 的值并运行此命令。

```
python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt
 --key ~/certs/private.pem.key --endpoint your-iot-endpoint --thing_name your-iot-
thing-name
```

3. 观察示例应用程序：

   1. 为您的账户连接到 AmazonIoT 服务。
   2. 订阅 Delta事件和 Update及 Get响应。
   3. 提示您在终端中输入所需的值。
   4. 输出类似于以下内容：

```
Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
Connected!
```

```
Subscribing to Delta events...
Subscribing to Update responses...
Subscribing to Get responses...
Requesting current shadow state...
Launching thread to read user input...
Finished getting initial shadow state.
Shadow contains reported value 'off'.
Enter desired value:
```

Note

如果您在运行 shadow.py示例应用程序时遇到问题，请查看 the section called “步骤 3：借助 shadow.py示例应用程序排查问题” (p. 211)。若要获取可能帮助您更正此问题的其它信息，请将 --verbosity debug参数添加到命令行，以便示例应用程序显示有关其正在执行的操作的详细消息。

在影子文档中输入值并观察更新

您可以在终端中输入值来指定 desired值，该值还会更新 reported值。假设您在终端中输入颜色 yellow。reported 值也会更新为颜色 yellow。下面显示了终端中显示的消息：

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

发布此更新请求时，Amazon IoT 将为事物资源创建一个默认的经典影子。您可以在 Amazon IoT控制台中观察您发布到 reported和 desired中的值，方法是查看您创建的事物资源的影子文档（例如 My_light_bulb)。要查看影子文档中的更新：

1. 在 Amazon IoT控制台中，选择 Manage（管理），然后选择 Things（事物）。
2. 在显示的事物列表中，选择您创建的事物，选择 Shadows（影子），然后选择 Classic Shadow（经典影子）。

影子文档应类似于以下内容，显示为颜色 yellow设置的 reported和 desired值。您可以在文档的 Shadow state（影子状态）部分看到这些值。

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "yellow"
},
"reported": {
  "welcome": "aws-iot",
  "color": "yellow"
}
}
```

您还可以看到 Metadata（元数据）部分，其中包含请求的时间戳信息和版本号。

您可以使用状态文档版本来确保正在更新的设备影子文档为最新版本。如果您发送另一个更新请求，则版本号将增加 1。当您为更新请求提供版本时，如果状态文档的当前版本与提供的版本不符，该服务将显示 HTTP 409 冲突响应代码并拒绝请求。

```
{
"metadata": {
```

```
  "desired": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  }
},
"version": 10
}
```

要了解有关影子文档的详细信息并观察状态信息的更改，请继续阅读下一教程 教程：示例应用程序及 MQTT 测试客户端与 Device Shadow 交互 (p. 213)，如本教程的 步骤 4：查看结果和后续步骤 (p. 212)部分所述。或者，您也可以在下一部分中了解 shadow.py示例代码以及它如何使用 MQTT 协议。

步骤 2：查看 shadow.py 设备软件开发包示例应用

本部分将回顾本教程中适用的来自 Amazon IoTDevice SDK v2 for Python 中的 shadow.py示例应用程序。在这里，我们将回顾它如何使用 MQTT 和采用 WSS 协议的 MQTT 来连接到 Amazon IoT Core。Amazon 公共运行时 (Amazon-CRT) 库提供低级通信协议支持，并包含在 Amazon IoTDevice SDK v2 for Python 中。

虽然本教程使用 MQTT 和 采用 WSS 的 MQTT，Amazon IoT 支持发布 HTTPS 请求的设备。有关从设备发送 HTTP 消息的 Python 程序示例，请参阅使用 Python 的 requests库的 HTTPS 代码示例 (p. 91)。

有关如何对设备通信使用何种协议作出明智决定的信息，请查看 为设备通信选择协议 (p. 76)。

MQTT

shadow.py 示例在 mqtt_connection_builder 中调用 mtls_from_path（此处显示）以使用 MQTT 协议建立与 Amazon IoT Core 的连接。mtls_from_path 使用 X.509 证书和 TLS v1.2 对设备进行身份验证。Amazon-CRT 库处理该连接的较低级别的详细信息。

```
mqtt_connection = mqtt_connection_builder.mtls_from_path(
  endpoint=args.endpoint,
  cert_filepath=args.cert,
  pri_key_filepath=args.key,
  ca_filepath=args.ca_file,
  client_bootstrap=client_bootstrap,
  on_connection_interrupted=on_connection_interrupted,
  on_connection_resumed=on_connection_resumed,
  client_id=args.client_id,
  clean_session=False,
  keep_alive_secs=6
)
```

- endpoint 是您从命令行传入的 Amazon IoT端点，而 client_id是在 Amazon Web Services 区域中唯一标识此设备的 ID。
- cert_filepath、pri_key_filepath 和 ca_filepath是指向设备证书和私钥文件以及根 CA 文件的路径。
- client_bootstrap 是处理套接字通信活动的通用运行时对象，并在调用 前实例化。mqtt_connection_builder.mtls_from_path

- on_connection_interrupted 和 on_connection_resumed是在设备连接中断和恢复时调用的回调函数。
- clean_session 表示是否启动新的持久会话，或者如果会话已存在，则重新连接到现有会话。keep_alive_secs 是保持活动状态的值（以秒为单位），以发送到 CONNECT请求中。在此时间间隔内将自动发送 ping。如果服务器在此值的 1.5 倍之后没有收到 ping，则假定连接丢失。

shadow.py 示例还调用 mqtt_connection_builder中的 websockets_with_default_aws_signing以使用采用 WSS 的 MQTT 协议与 Amazon IoT Core建立连接。采用 WSS 的 MQTT 也使用与 MQTT 相同的参数，并采用以下附加参数：

- region 是 Signature V4 身份验证使用的Amazon区域，而 credentials_provider是 Amazon 凭证，用于进行身份验证。区域是从命令行传递的，credentials_provider 对象在调用 mqtt_connection_builder.websockets_with_default_aws_signing前进行实例化。
- websocket_proxy_options 是 HTTP 代理选项（如果使用代理主机）。在 shadow.py示例应用程序中，此值在调用 mqtt_connection_builder.websockets_with_default_aws_signing前进行实例化。

订阅影子主题和活动

shadow.py 示例尝试建立连接并等待完全连接。如果未连接，命令将排队。连接后，示例将订阅增量事件并更新和获取消息，并发布服务质量 (QoS) 级别为 1 的消息 (mqtt.QoS.AT_LEAST_ONCE)。

当设备订阅 QoS 级别 1 的消息时，消息代理会保存该设备订阅的消息，直到这些消息可以发送到设备。消息代理会重新发送消息，直至收到设备发出的 PUBACK响应。

有关 MQTT 协议的更多信息，请参阅 查看 MQTT 协议 (p. 161)和 MQTT (p. 77)。

有关本教程中如何使用 MQTT、采用 WSS 的 MQTT、持久性会话和 QoS 级别的详细信息，请参阅 查看 pubsub.py Device SDK 示例应用程序 (p. 162)。

步骤 3：借助 shadow.py示例应用程序排查问题

当您运行 shadow.py示例应用程序时，您应该看到终端中显示的一些消息以及输入 desired值的提示。如果程序抛出错误，那么请调试以解决错误，您可以从检查系统是否运行了正确的命令开始。

在某些情况下，错误消息可能会指示连接问题，并且看起来类似于：Host name was invalid for dns resolution 或者 Connection was closed unexpectedly。在这种情况下，以下是您可以检查的几个事项：

- 检查命令中的端点地址

  查看您在命令中输入用来运行示例应用程序的 endpoint参数（例如 a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com），然后在 Amazon IoT控制台中检查此值。

  要检查是否使用了正确的值：

  1. 在 Amazon IoT控制台中，选择 Manage（管理），然后选择 Things（事物）。
  2. 选择您为示应用程序创建的事物（例如 My_light_bulb），然后选择 Interact（交互）。

  您的端点会显示在事物详细信息页面的 HTTPS 部分中。您还会看到一条消息，写着：This thing already appears to be connected.
- 检查证书激活

  证书通过 Amazon IoT Core对您的设备进行身份验证。

  要检查您的证书是否处于活动状态：

  1. 在 Amazon IoT控制台中，选择 Manage（管理），然后选择 Things（事物）。

2. 选择您为示应用程序创建的事物（例如 My_light_bulb），然后选择 Security（安全）。
3. 选择证书，然后从证书的详细信息页面中选择 Actions（操作）。

如果在下拉列表中 Activate（激活）不可用，且您只能选择 Deactivate（停用），则表示您的证书处于活动状态。如果并非如此，请选择 Activate（激活）并重新运行示例程序。

如果程序仍未运行，请检查 `certs`文件夹中的证书文件名。

- 检查附加到事物资源的策略

  当证书对您的设备进行身份验证时，Amazon IoT 策略允许设备执行 Amazon IoT操作，例如订阅或发布 MQTT 预留主题。

  要检查是否附加了正确的策略：

  1. 查找如前所述的证书，然后选择 Policies（策略）。
  2. 选择显示的策略，并检查它是否描述 `connect`、`subscribe`、`receive` 和 `publish`操作，授予设备发布和订阅 MQTT 预留主题的权限。

     如需了解示例策略，请参阅 步骤 1：创建 Device Shadow 的 Amazon IoT策略 (p. 205)。

  如果您看到错误消息，指示与 Amazon IoT的连接出现问题，这可能是由于您正在为策略使用的权限所导致的。如果是这种情况，建议您从向 Amazon IoT资源提供完全访问权限的策略开始，然后重新运行示例程序。您可以编辑当前策略，也可以选择当前策略，选择 Detach（分离），然后创建另一个提供完全访问权限的策略并将其附加到您的事物资源。您可以稍后将策略限制为运行程序所需的操作和策略。

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

- 检查您的设备 Device SDK 安装

  如果程序仍未运行，您可以重新安装 Device SDK 包以确保 SDK 安装完整且正确。

步骤 4：查看结果和后续步骤

在本教程中，您学习了如何：

- 安装所需的软件、工具和 Amazon IoTDevice SDK for Python。
- 了解示例应用程序 `shadow.py`如何使用 MQTT 协议检索和更新影子的当前状态。
- 运行 Device Shadow 的示例应用程序，并在 Amazon IoT控制台中观察和更新影子文档。您还学习了如何在运行程序时解决任何问题和修复错误。

后续步骤

您现在可以运行 `shadow.py`示例应用程序，并使用 Device Shadow 控制状态。您可以在 Amazon IoT控制台中观察并更新影子文档，并观察示例应用程序相应的增量事件。使用 MQTT 测试客户端，您可以订阅预留影子主题并在运行示例程序时观察主题收到的消息。有关如何使用该教程的更多信息，请参阅 教程：示例应用程序及 MQTT 测试客户端与 Device Shadow 交互 (p. 213)。

教程：示例应用程序及 MQTT 测试客户端与 Device Shadow 交互

若要与 shadow.py示例应用程序交互，请在终端中为 desired值输入一个值。例如，您可以向请求指定类似于红绿灯的颜色和 Amazon IoT响应，并更新报告的值。

在本教程中，您将学习如何：

- 使用 shadow.py示例应用程序来指定所需的状态并更新影子的当前状态。
- 编辑影子文档以观察增量事件以及 shadow.py示例应用程序如何响应。
- 使用 MQTT 测试客户端订阅影子主题并在运行示例程序时观察更新。

运行本教程之前，您必须具有：

设置您的Amazon Web Services 账户，配置您的 Raspberry Pi 设备，并创建 Amazon IoT事物和策略。您还必须安装了所需的软件、Device SDK、证书文件，并在终端中运行示例程序。有关更多信息，请参阅教程 教程：准备 Raspberry Pi 运行影子应用程序 (p. 203)和 步骤 1：运行 shadow.py 示例应用程序 (p. 208)。如果您尚未完成这些教程，请先予完成。

在本教程中，您将：

- 步骤 1：使用 shadow.py示例应用程序更新所需值和报告值 (p. 213)
- 步骤 2：查看来自 shadow.py中 MQTT 测试客户端中示例应用程序的消息 (p. 214)
- 步骤 3：排除 Device Shadow 交互中的错误 (p. 218)
- 步骤 4：查看结果和后续步骤 (p. 219)

完成本教程需要大约 45 分钟。

步骤 1：使用 shadow.py示例应用程序更新所需值和报告值

在前面的教程 步骤 1：运行 shadow.py 示例应用程序 (p. 208)中，您学习了在按照部分 教程：安装设备软件开发包并运行 Device Shadow 示例应用程序 (p. 207)中所述输入必要数值后，如何在 Amazon IoT控制台中观察发布到影子文档的消息。

在前面的示例中，我们将所需的颜色设置为 yellow。输入每个值后，终端将提示您输入另一个 desired值。如果您再次输入相同的值 (yellow)，应用程序会识别这一点，并提示您输入一个新的 desired值。

```
Enter desired value:
yellow
Local value is already 'yellow'.
Enter desired value:
```

现在，假设您输入了颜色 green。Amazon IoT 响应请求，并将 reported值更新为 green。这就是当 desired状态不同于 reported状态时产生更新的方式，从而导致增量。

shadow.py 示例应用程序模拟 Device Shadow 交互：

1. 在终端输入 desired值（例如 yellow）来发布所需的状态。
2. 由于 desired状态不同于 reported状态（比如说颜色 green），则会产生增量，并且订阅增量的应用程序会收到此消息。
3. 应用程序响应消息并将其状态更新为 desired值，yellow。
4. 然后，应用程序会发布一条更新消息，其中包含设备状态的新报告值，yellow。

下面显示了终端中显示的消息，其展示了更新申请是如何发布的。

```
Enter desired value:
green
Changed local shadow value to 'green'.
Updating reported shadow value to 'green'...
Update request published.
Finished updating reported shadow value to 'green'.
```

在 Amazon IoT控制台中，影子文档将 reported和 desired字段更新后的值反映到 green，然后版本号会增加 1。例如，如果以前的版本号显示为 10，则当前版本号将显示为 11。

> **Note**
>
> 删除影子不会将版本号重置为 0。当您发布更新请求或创建另一个具有相同名称的影子时，您将看到影子版本递增 1。

编辑影子文档以观察增量事件

shadow.py 示例应用程序也订阅 delta事件，并在 desired值出现更改时予以响应。例如，您可以将 desired值更改为颜色 red。为此，请在 Amazon IoT控制台中，单击 Edit（编辑）以编辑 Shadow 文档，然后在 JSON 中将 desired设置为 red，同时将 reported值保留为 green。保存更改之前，请保持 Raspberry Pi 上的终端打开状态，因为当出现更改时，您会看到终端中显示的消息。

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

保存新值后，shadow.py 示例应用程序响应此更改，并在终端中显示指示增量的消息。然后，您应该看到以下消息出现在输入 desired值的提示下方。

```
Enter desired value:
Received shadow delta event.
Delta reports that desired value is 'red'. Changing local value...
Changed local shadow value to 'red'.
Updating reported shadow value to 'red'...
Finished updating reported shadow value to 'red'.
Enter desired value:
Update request published.
Finished updating reported shadow value to 'red'.
```

步骤 2：查看来自 shadow.py中 MQTT 测试客户端中示例应用程序的消息

您可以使用 Amazon IoT控制台中的 MQTT 测试客户端来监控传递在 Amazon Web Services 账户中的 MQTT 消息。通过订阅 Device Shadow 服务使用的保留 MQTT 主题，您可以观察在运行示例应用程序时主题收到的消息。

如果尚未使用 MQTT 测试客户端，您可以查看 使用 Amazon IoT MQTT 客户端查看 MQTT 消息 (p. 59)。它可以帮助您了解如何使用 Amazon IoT控制台中的 MQTT 测试客户端来查看通过消息代理时的 MQTT 消息。

1. 打开 MQTT 测试客户端

   打开 Amazon IoT控制台中的 MQTT 测试客户端，以便您可以观察 MQTT 主题收到的消息，而不会丢失 MQTT 测试客户端的配置。如果您让 MQTT 测试客户端进入控制台中的其它页面，则 MQTT 测试客户端不会保留任何订阅或消息日志。对于本教程的这一部分，您可以在单独的窗口中打开 Amazon IoT事物的影子文档和 MQTT 测试客户端，以便更轻松地观察与 Device Shadow 的交互。

2. 订阅 MQTT 预留的影子主题

   您可以使用 MQTT 测试客户端输入 Device Shadow 的 MQTT 预留主题的名称，并在运行 `shadow.py`示例应用程序时订阅它们以接收更新。要订阅主题：

   a. 在 Amazon IoT控制台中的 MQTT 测试客户端中，选择 Subscribe to a topic（订阅主题）。
   b. 在 Topic filter（主题筛选条件）部分，输入：$aws/things/***thingname***/shadow/update/#。在这里，`thingname` 是您之前创建的事物资源的名称（例如，`My_light_bulb`）。
   c. 保留附加配置设置的默认值，然后选择 Subscribe（订阅）。

   通过使用 # 通配符，您可以同时订阅多个 MQTT 主题，并在单个窗口中观察设备与其影子之间交换的所有消息。有关通配符及其用法的更多信息，请参阅 MQTT 主题 (p. 94)。

3. 运行 **`shadow.py`**示例程序和观察消息

   在 Raspberry Pi 的命令行窗口中，如果您已经断开程序的连接，请再次运行示例应用程序，并在 Amazon IoT控制台中的 MQTT 测试客户端观察消息。

   a. 运行以下命令以重新启动示程序。将 *`your-iot-thing-name`* 和 *`your-iot-endpoint`* 替换为您早前创建的 Amazon IoT事物名称（例如，`My_light_bulb`）以及与设备交互的端点名称。

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   python3 shadow.py --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/
   device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint --
   thing_name your-iot-thing-name
   ```

   `shadow.py` 示例应用程序随后会开始运行并检索当前影子状态。如果已删除影子或清除当前状态，程序会将当前值设置为 `off`，然后提示您输入 `desired` 值。

   ```
   Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID
    'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
   Connected!
   Subscribing to Delta events...
   Subscribing to Update responses...
   Subscribing to Get responses...
   Requesting current shadow state...
   Launching thread to read user input...
   Finished getting initial shadow state.
   Shadow document lacks 'color' property. Setting defaults...
   Changed local shadow value to 'off'.
   Updating reported shadow value to 'off'...
   Update request published.
   Finished updating reported shadow value to 'off'...
   Enter desired value:
   ```

   另一方面，如果程序正在运行，并且您将程序重新启动，您将看到终端中报告的最新颜色值。在 MQTT 测试客户端中，您将看到主题 $aws/things/***thingname***/shadow/get 和 $aws/things/***thingname***/shadow/get/accepted 的更新。

   假设报告的最新颜色为 `green`。下面显示的是 $aws/things/***thingname***/shadow/get/accepted JSON 文件的内容。

   ```
   {
   "state": {
     "desired": {
       "welcome": "aws-iot",
       "color": "green"
     },
     "reported": {
   ```

```
    "welcome": "aws-iot",
    "color": "green"
  }
},
"metadata": {
  "desired": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620161643
    }
  }
},
"version": 10,
"timestamp": 1620173908
}
```

b. 在终端中输入 desired值，例如 yellow。shadow.py 示例应用程序响应并在终端中显示以下消息，这些消息显示了 reported值变更为 yellow的情况。

```
Enter desired value:
yellow
Changed local shadow value to 'yellow'.
Updating reported shadow value to 'yellow'...
Update request published.
Finished updating reported shadow value to 'yellow'.
```

在 Amazon IoT 控制台中的 MQTT 测试客户端中，您会在 Subscriptions（订阅）下看到以下主题收到一条消息：

- $aws/things/***thingname***/shadow/update：显示两个 desired和 updated值更改为颜色 yellow。
- $aws/things/***thingname***/shadow/update/accepted：显示 desired和 reported状态当前的值及其元数据和版本信息。
- $aws/things/***thingname***/shadow/update/documents：显示 desired和 reported状态之前以及当前的值，及其元数据和版本信息。

由于文档 $aws/things/***thingname***/shadow/update/documents 还包含其它两个主题中包含的信息，我们可以查看该文档以查看状态信息。上一个状态显示报告的值设置为 green、其元数据和版本信息，而当前的状态显示报告的值更新为 yellow。

```
{
"previous": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "green"
    },
    "reported": {
      "welcome": "aws-iot",
```

```
      "color": "green"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297898
      }
    }
  },
  "version": 10
},
"current": {
  "state": {
    "desired": {
      "welcome": "aws-iot",
      "color": "yellow"
    },
    "reported": {
      "welcome": "aws-iot",
      "color": "yellow"
    }
  },
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    },
    "reported": {
      "welcome": {
        "timestamp": 1617297888
      },
      "color": {
        "timestamp": 1617297904
      }
    }
  },
  "version": 11
},
"timestamp": 1617297904
}
```


c. 现在，如果您输入另一个 desired值，您可以看到 reported值的进一步更改和这些主题收到的消息更新。版本号也会增加 1。例如，如果您输入值 green，之前的状态会报告值 yellow，而当前状态报告值 green。

4. 编辑影子文档以观察增量事件

   要观察对增量主题的更改，请在 Amazon IoT控制台编辑影子文档。例如，您可以将 desired值更改为颜色 red。为此，请在 Amazon IoT控制台中，选择 Edit（编辑），然后在 JSON 中将 desired值设置

为红色，同时保持 reported设置为 green。在保存更改之前，请保持终端处于开启状态，因为您将看到终端中报告的增量消息。

```
{
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

shadow.py 示例应用程序响应此更改，并在终端中显示指示增量的消息。在 MQTT 测试客户端中，update 主题将收到一条消息，显示对 desired和 reported值的更改。

您还可以看到主题 $aws/things/***thingname***/shadow/update/delta 收到一条消息。要查看消息，请选择该主题，其列于 Subscriptions（订阅）下方。

```
{
"version": 13,
"timestamp": 1617318480,
"state": {
  "color": "red"
},
"metadata": {
  "color": {
    "timestamp": 1617318480
  }
}
}
```

步骤 3：排除 Device Shadow 交互中的错误

运行影子示例应用程序时，在观察与 Device Shadow 服务的交互过程中可能会遇到问题。

如果程序运行成功并提示您输入 desired值，您应该能够通过使用前述影子文档和 MQTT 测试客户端来观察 Device Shadow 交互。但是，如果您无法查看交互，可以检查以下事项：

- 在 Amazon IoT控制台中检查事物名称及其影子

  如果您在影子文档中没有看到这些消息，请查看该命令，并确保它与 Amazon IoT控制台 中的事物名称相符。您还可以通过选择事物资源，然后选择 Shadows（影子）来查看您是否拥有经典影子。本教程主要侧重于与经典影子的交互。

  您还可以确认您使用的设备已连接到互联网。在 Amazon IoT控制台，选择您之前创建的事物，然后选择 Interact（交互）。在事物详细信息页面上，您应该会在此处看到一条消息：`This thing already appears to be connected.`
- 检查您订阅的 MQTT 预留主题

  如果在 MQTT 测试客户端中看不到消息，请检查您订阅的主题是否格式正确。MQTT Device Shadow 主题具有格式 $aws/things/***thingname***/shadow/ 并且可能有 update、get 或者 delete 跟随其后，具体取决于要对影子执行的操作。本教程使用主题 $aws/things/***thingname***/shadow/#，因此请确保您在测试客户端的 Topic filter（主题筛选条件）部分订阅主题时正确输入。

  输入主题名称时，请确保 *thingname* 的名称与您之前创建的 Amazon IoT事物的名称相同。您还可以订阅其它 MQTT 主题，以查看是否已成功执行更新。例如，您可以订阅主题 $aws/things/***thingname***/

shadow/update/rejected 以在更新请求失败时收到一条消息，让您能够调试连接问题。有关预留主题的更多信息，请参阅 the section called “影子主题” (p. 105)和 Device Shadow MQTT 主题 (p. 620)。

步骤 4：查看结果和后续步骤

在本教程中，您学习了如何：

- 使用 shadow.py示例应用程序来指定所需的状态并更新影子的当前状态。
- 编辑影子文档以观察增量事件以及 shadow.py示例应用程序如何响应。
- 使用 MQTT 测试客户端订阅影子主题并在运行示例程序时观察更新。

后续步骤

您可以订阅其它 MQTT 预留主题，以观察影子应用程序的更新。例如，如果您只订阅主题 $aws/things/***thingname***/shadow/update/accepted，则在成功执行更新时只能看到当前状态信息。

您还可以订阅其它影子主题以调试问题或了解有关 Device Shadow 交互的详细信息，还可以调试与 Device Shadow 交互有关的任何问题。有关更多信息，请参阅 the section called “影子主题” (p. 105)和 Device Shadow MQTT 主题 (p. 620)。

您还可以选择通过使用命名影子或使用为 LED 而连接到 Raspberry Pi 的附加硬件来扩展您的应用程序，并使用从终端发送的消息观察其状态的变化。

有关 Device Shadow 服务以及在设备、应用程序和服务中使用该服务的详细信息，请参阅 Amazon IoT Device Shadow 服务 (p. 593)、在设备中使用影子 (p. 596) 和 在应用程序和服务中使用影子 (p. 598)。

教程：为 Amazon IoT Core创建自定义授权方

本教程演示了使用 Amazon CLI创建、验证和使用自定义身份验证的步骤。或者，使用本教程，您可以借助 HTTP Publish API 使用 Postman 将数据发送到 Amazon IoT Core。

本教程介绍如何创建一个示例 Lambda 函数，该函数将在启用令牌签名的情况下，借助 create-authorizer调用实现授权和身份验证逻辑。然后，授权方会使用 test-invoke-authorizer进行验证，最后您可以使用 HTTP Publish API 将数据发送到 Amazon IoT Core来测试 MQTT 主题。示例请求将使用`x-amz-customauthorizer-name`标头并传递和请求标头中的 authorizer 来指定要调用的授权方。 token-key-name `x-amz-customauthorizer-signature`

您将在本教程中学到的内容：

- 如何创建一个 Lambda 函数作为自定义授权方处理程序
- 如何在启用令牌签名的情况下使用使用 Amazon CLI创建自定义授权方。
- 如何使用 test-invoke-authorizer命令测试您的自定义授权方
- 如何使用 Postman 发布 MQTT 主题并使用您的自定义授权方验证请求

完成本教程需要大约 60 分钟。

在本教程中，您将：

- 步骤 1：为自定义授权方创建 Lambda 函数 (p. 220)
- 步骤 2：为自定义授权方创建公有密钥和私有密钥对 (p. 222)
- 步骤3：创建客户授权方资源及其授权 (p. 223)
- 步骤 4：通过调用测试授权方 test-invoke-authorizer (p. 225)
- 步骤 5：测试使用 Postman 发布 MQTT 消息 (p. 227)
- 步骤 6：在 MQTT 测试客户端中查看消息 (p. 228)

- 步骤 7：查看结果和后续步骤 (p. 229)
- 步骤 8：清除 (p. 229)

在开始本教程之前，请确保您具有：

- 设置您的Amazon Web Services 账户 (p. 16)

  您将需要用到您的 Amazon Web Services 账户和 Amazon IoT控制台以完成本教程。

  当您用于本教程的账户中至少包含这些Amazon托管策略时，账户的效果最佳：
  - `IAMFullAccess`
  - `AWSIoTFullAccess`
  - `AWSLambda_FullAccess`

  Important

  本教程中使用的 IAM 策略比您在生产实施中应遵循的更宽松。在生产环境中，请确保您的账户和资源策略仅授予必要的权限。
  在您为生产创建 IAM 策略时，请确定用户和角色需要的访问权限，然后设计允许他们仅执行这些任务的策略。
  有关更多信息，请参阅 IAM 安全最佳实践。
- 已安装 Amazon CLI

  有关如何安装 Amazon CLI的信息，请参阅安装 AmazonCLI。本教程需要 Amazon CLI版本 `aws-cli/2.1.3 Python/3.7.4 Darwin/18.7.0 exe/x86_64`或更高版本。
- OpenSSL 工具

  本教程中的示例使用 LibreSSL 2.6.5。您还可以在本教程中使用 OpenSSL v1.1.1i 工具。
- 查看 Amazon Lambda概览

  如果您未使用过 Amazon Lambda，请查看 Amazon Lambda和 Lambda 入门以学习它的术语和概念。
- 回顾了如何在 Postman 中构建请求

  有关更多信息，请参阅构建请求。
- 从以前的教程中删除了自定义授权方

  您的Amazon Web Services 账户一次只能配置有限数量的自定义授权方。有关如何创建和删除自定义授权方的信息，请参阅 the section called “步骤 8：清除” (p. 229)。

步骤 1：为自定义授权方创建 Lambda 函数

Amazon IoT Core 中的定制身份验证使用授权方资源来对客户端进行身份验证和授权。您将在本部分中创建的函数将在连接到 Amazon IoT Core和访问 Amazon IoT资源时对客户端进行身份验证和授权。

Lambda 函数执行以下操作：

- 如果请求来自 test-invoke-authorizer，则它会返回一个带有 `Deny`操作的 IAM 策略。
- 如果请求来自使用 HTTP 的 Passport，且 `actionToken`参数的值为 `allow`，则它会返回一个带有 `Allow`操作的 IAM 策略。否则，它会返回带有 `Deny`操作的 IAM 策略。

要为自定义授权方创建 Lambda 函数：

1. 在 Lambda 控制台，打开函数。
2. 选择 Create function (创建函数)。

3. 确认 Author from scratch（从头开始编写）处于选中状态。
4. 在 Basic information 下：

   a. 在 Function name (函数名称) 中，输入 **`custom-auth-function`**。

   b. 在运行时确认 Node.js 18.x
5. 选择 Create function（创建函数）。

   Lambda 创建 Node.js 函数和执行角色，该角色将授予函数上载日志的权限。在您调用函数时，Lambda 函数将承担执行角色，并使用该执行角色为 AmazonSDK 创建凭证，并从事件源读取数据。
6. 要在Amazon Cloud9编辑器中查看函数的代码和配置，请在设计窗口选择，然后在编辑器的导航窗格中选择 index.js。custom-auth-function

   对于脚本语言（如 Node.js），Lambda 包含返回成功响应的基本函数。您可以使用 Amazon Cloud9编辑器来编辑您的函数，只要源代码不超过 3MB 即可。
7. 在编辑器中使用以下代码替换 index.js 代码：

```
// A simple Lambda function for an authorizer. It demonstrates
// How to parse a CLI and Http password to generate a response.

exports.handler = async (event, context, callback) {

    //Http parameter to initiate allow/deny request
    const HTTP_PARAM_NAME='actionToken';
    const ALLOW_ACTION = 'Allow';
    const DENY_ACTION = 'Deny';

    //Event data passed to Lambda function
    var event_str = JSON.stringify(event);
    console.log('Complete event :'+ event_str);

    //Read protocolData from the event json passed to Lambda function
    var protocolData = event.protocolData;
    console.log('protocolData value---> ' + protocolData);

    //Get the dynamic account ID from function's ARN to be used
    // as full resource for IAM policy
    var ACCOUNT_ID = context.invokedFunctionArn.split(":")[4];
    console.log("ACCOUNT_ID---"+ACCOUNT_ID);

    //Get the dynamic region from function's ARN to be used
    // as full resource for IAM policy
    var REGION = context.invokedFunctionArn.split(":")[3];
    console.log("REGION---"+REGION);

    //protocolData data will be undefined if testing is done via CLI.
    // This will help to test the set up.
    if (protocolData === undefined) {

        //If CLI testing, pass deny action as this is for testing purpose only.
        console.log('Using the test-invoke-authorizer cli for testing only');
        callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

    } else{

        //Http Testing from Postman
        //Get the query string from the request
        var queryString = event.protocolData.http.queryString;
        console.log('queryString values -- ' + queryString);
        /*         global URLSearchParams       */
        const params = new URLSearchParams(queryString);
        var action = params.get(HTTP_PARAM_NAME);
```

```
        if(action!=null && action.toLowerCase() === 'allow'){

            callback(null, generateAuthResponse(ALLOW_ACTION,ACCOUNT_ID,REGION));

        }else{

            callback(null, generateAuthResponse(DENY_ACTION,ACCOUNT_ID,REGION));

        }

    }

};

// Helper function to generate the authorization IAM response.
var generateAuthResponse = function(effect,ACCOUNT_ID,REGION) {

    var full_resource = "arn:aws:iot:"+ REGION + ":" + ACCOUNT_ID + ":*";
    console.log("full_resource---"+full_resource);

    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'principalId';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17';
    policyDocument.Statement = [];
    var statement = {};
    statement.Action = 'iot:*';
    statement.Effect = effect;
    statement.Resource = full_resource;
    policyDocument.Statement[0] = statement;
    authResponse.policyDocuments = [policyDocument];
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 600;

    console.log('custom auth policy function called from http');
    console.log('authResponse --> ' + JSON.stringify(authResponse));
    console.log(authResponse.policyDocuments[0]);

    return authResponse;
}
```

8. 选择 Deploy（部署）。
9. 在 Changes deployed（更改已部署）出现在编辑器上方后：
   a. 滚动到编辑器上方的 Function overview（函数概览）部分。
   b. 复制 Function ARN（函数 ARN）并保存以在本教程稍后的部分中使用。
10. 测试 函数。
    a. 选择 Test（测试）选项卡。
    b. 使用默认测试设置，选择 Invoke（调用）。
    c. 如果测试成功，则在 Execution results（执行结果）下，打开 Details（详细信息）视图。您应该看到函数返回的策略文档。

       如果测试失败或没有看到策略文档，请查看代码以查找并更正错误。

步骤 2：为自定义授权方创建公有密钥和私有密钥对

您的自定义授权方需要公有密钥和私有密钥来对其进行身份验证。本部分中的命令使用 OpenSSL 工具来创建此密钥对。

要为自定义授权方创建公有和私有密钥对，请执行以下操作

1. 创建私有密钥文件。

```
openssl genrsa -out private-key.pem 4096
```

2. 验证您刚创建的私有密钥文件。

```
openssl rsa -check -in private-key.pem -noout
```

如果命令未显示任何错误，则私有密钥文件是有效的。

3. 创建公有密钥文件。

```
openssl rsa -in private-key.pem -pubout -out public-key.pem
```

4. 验证公有密钥文件。

```
openssl pkey -inform PEM -pubin -in public-key.pem -noout
```

如果命令未显示任何错误，则公有密钥文件是有效的。

步骤3：创建客户授权方资源及其授权

Amazon IoT 自定义授权方是将在前面步骤中创建的所有元素联系在一起的资源。在本部分中，您将创建一个自定义授权方资源，并授予其运行您之前创建的 Lambda 函数的权限。您可以通过使用 Amazon IoT控制台、Amazon CLI 或 AmazonAPI 来创建自定义授权方资源。

在本教程中，您只需要创建一个自定义授权方。本部分描述了如何使用 Amazon IoT控制台和 Amazon CLI进行创建，以便您可以使用最方便的方法操作。以两种方法创建的自定义授权方资源之间没有区别。

创建自定义授权方资源

选择以下选项之一以创建自定义授权方资源

- 使用 Amazon IoT控制台创建自定义授权方 (p. 223)
- 使用 Amazon CLI创建自定义授权方 (p. 224)

要创建自定义授权方（控制台）

1. 打开Amazon IoT控制台的自定义授权者页面，然后选择创建授权者。
2. 在“创建授权者”中：
   a. 在授权者名称中，输入**my-new-authorizer**。
   b. 在“授权者状态”中，选中“有效”。
   c. 在 Authorizer function（授权方函数）中，选择您之前创建的 Lambda 函数。
   d. 在 Token validation - optional（令牌验证 — 可选）中：
      i. 开启代币验证。
      ii. 在令牌密钥名称中，输入**tokenKeyName**。
      iii. 选择 Add key (添加密钥)。
      iv. 在密钥名称中，输入**FirstKey**。

v. 在公钥中，输入public-key.pem文件的内容。请务必包含文件中的行以及 -----BEGIN PUBLIC KEY-----和 -----END PUBLIC KEY-----，并且不要从文件内容中添加或删除任何换行符、回车符或其它字符。您输入的字符串应类似下面这个示例。

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----
```

3. 选择 Create Authorizer（创建授权方）。
4. 如果创建了自定义授权方资源，您将看到自定义授权方列表，并且您的新自定义授权方应显示在列表中，然后您便可以继续下一部分展开测试了。

   如果您看到错误，请检查错误并尝试再次创建自定义授权方，然后仔细检查条目。请注意，每个自定义授权方资源必须具有唯一的名称。

要创建自定义授权方（Amazon CLI）

1. 将您的值替换为 authorizer-function-arn和 token-signing-public-keys，然后运行以下命令：

```
aws iot create-authorizer \
--authorizer-name "my-new-authorizer" \
--token-key-name "tokenKeyName" \
--status ACTIVE \
--no-signing-disabled \
--authorizer-function-arn "arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-
function" \
--token-signing-public-keys FirstKey="-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBzOk4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQOe5tyzz1MsKdmtFGxMqQ3qjEXAMPLEOmqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU5OSAnpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csalS/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N5ON1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZO8OzcobLngJ0Ibw9KkcUdklW
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7l7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFl2yOBmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXSOfUCAwEAAQ==
-----END PUBLIC KEY-----"
```

其中：

- authorizer-function-arn 值是您为自定义授权方创建的 Lambda 函数的 Amazon Resource Name (ARN)。

- token-signing-public-keys 值包含密钥的名称、**FirstKey** 以及 public-key.pem文件的内容。请务必包含文件中的行以及 -----BEGIN PUBLIC KEY-----和 -----END PUBLIC KEY-----，并且不要从文件内容中添加或删除任何换行符、回车符或其它字符。

  注意：输入公有密钥时要非常小心，因为对公有密钥值的任何更改都会使其无法使用。

2. 如果创建了自定义授权方，则该命令将返回新资源的名称和 ARN，如下所示。

```
{
    "authorizerName": "my-new-authorizer",
    "authorizerArn": "arn:aws:iot:Region:57EXAMPLE833:authorizer/my-new-authorizer"
}
```

保存 authorizerArn值，以供下一步使用。

请记住，每个自定义授权方资源必须具有唯一的名称。

授权自定义授权方资源

在本部分中，您将为刚创建的自定义授权方资源授予权限，以运行 Lambda 函数。要授予权限，您可以使用 add-permiss ion CLI 命令。

使用 Amazon CLI将权限授予您的 Lambda 函数

1. 插入您的值后，输入以下命令。请注意，statement-id 值必须唯一。如果您之前运行过本教程，或者如果您遇到了 ResourceConflictException错误，请将 *Id-1234*替换为另一个值。

```
aws lambda add-permission  \
--function-name "custom-auth-function" \
--principal "iot.amazonaws.com" \
--action "lambda:InvokeFunction" \
--statement-id "Id-1234" \
--source-arn authorizerArn
```

2. 如果命令成功，则返回一个权限语句，如本示例。您可以继续到下一部分来测试自定义授权方。

```
{
    "Statement": "{\"Sid\":\"Id-1234\",\"Effect\":\"Allow\",\"Principal\":{\"Service
\":\"iot.amazonaws.com\"},\"Action\":\"lambda:InvokeFunction\",\"Resource\":
\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function\",\"Condition\":
{\"ArnLike\":{\"AWS:SourceArn\":\"arn:aws:lambda:Region:57EXAMPLE833:function:custom-
auth-function\"}}}"
}
```

如果命令不成功，则返回错误，如本示例。在继续操作之前，您需要查看并更正错误。

```
An error occurred (AccessDeniedException) when calling the AddPermission operation:
 User: arn:aws:iam::57EXAMPLE833:user/EXAMPLE-1 is not authorized to perform:
 lambda:AddPer
mission on resource: arn:aws:lambda:Region:57EXAMPLE833:function:custom-auth-function
```

步骤 4：通过调用测试授权方 test-invoke-authorizer

在本部分中，由于定义了所有资源，您将 test-invoke-authorizer从命令行调用，以测试授权通过。

请注意，在通过命令行调用授权方时，protocolData 并未定义，因此授权方将始终返回 DENY 文档。但是，此测试将确认您的自定义授权方和 Lambda 函数是否已正确配置，即使它并未完全测试 Lambda 函数。

要使用 Amazon CLI测试您的自定义授权方及其 Lambda 函数

1. 在包含您在上一步中创建的 private-key.pem文件的目录中，请运行以下命令。

```
echo -n "tokenKeyValue" | openssl dgst -sha256 -sign private-key.pem | openssl base64 -
A
```

此命令将创建要在下一步中使用的签名字符串。签名字符串如下所示：

```
dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V
+MMWu09YSA86+64Y3Gt4tOykpZqn9mn
VB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeeh
bQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePgnuOBvMGCEFE09jGjj
szEHfgAUAQIWXiVGQj16BU1xKpTGSiTAwheLKUjITOEXAMPLECK3aHKYKY
+d1vTvdthKtYHBq8MjhzJ0kggbt29V
QJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuX
f3LzCwQQF/YSUy02u5XkWn+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o
+K
EWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+flFeloZlAWQFH
xRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

复制此签名字符串以在下一步中使用。请注意，不要包含任何额外的字符或遗漏任何字符。

2. 在此命令中，将 token-signature值替换为上一步中的签名字符串，并运行此命令来测试您的授权程序。

```
aws iot test-invoke-authorizer \
--authorizer-name my-new-authorizer \
--token tokenKeyValue \
--token-signature dBwykzlb+fo+JmSGdwoGr8dyC2qB/IyLefJJr
+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4tOykpZqn9mnVB1wyxp
+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm90OyZ94R4bdJsHNig1JePg
+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/
r/F3VV8ITtQp3aXiUtcspACi6ca+tsDuXf3LzCwQQF/YSUy02u5XkWn
+sto6KCkpNlkD0wU8gl3+kOzxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o
+KEWckTe91I1mokDr5sJ4JXixvnJTVSx1li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+flFeloZlAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDJplpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

如果命令成功，它将返回自定义授权方函数生成的信息，如此示例。

```
{
    "isAuthenticated": true,
    "principalId": "principalId",
    "policyDocuments": [
        "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Action\":\"iot:*\",\"Effect\":
\"Deny\",\"Resource\":\"arn:aws:iot:Region:57EXAMPLE833:*\"}]}"
    ],
    "refreshAfterInSeconds": 600,
    "disconnectAfterInSeconds": 3600
}
```

如果命令返回错误，请查看错误并仔细检查您在本节中使用的命令。

步骤 5：测试使用 Postman 发布 MQTT 消息

1. 要从命令行获取设备数据端点，请如此处所示调用 describe-endpoint

```
aws iot describe-endpoint --output text --endpoint-type iot:Data-ATS
```

保存此地址以在后续步骤中用作 *device_data_endpoint_address*。

2. 打开一个新的 Postman 窗口并创建一个新的 HTTP POST 请求。

   a. 在您的计算机中，打开 Postman 应用程序。

   b. 在 Postman 的 File（文件）菜单中，选择 New…（新建）。

   c. 在 New（新建）对话框中，选择 Request（请求）。

   d. 在“保存”请求中，

      i. 在 Request name（请求名称）中，输入 **Custom authorizer test request**。

      ii. 在 Select a collection or folder to save to:（选择要保存到的收藏夹或文件夹：）中选择或创建要保存此请求的收藏夹。

      iii. 选择 Save to ***collection_name***（保存到“collection_name”）。

3. 创建 POST 请求以测试您的自定义授权方。

   a. 在 URL 字段旁边的请求方法选择器中，选择 POST。

   b. 在 URL 字段中，通过使用以下 URL 和来自 describe-endpoint 命令的 *device_data_endpoint_address* 为您的请求创建 URL。

```
https://device_data_endpoint_address:443/topics/test/cust-auth/topic?
qos=0&actionToken=allow
```

   请注意，此 URL 包含 `actionToken=allow`查询参数，该参数将告诉您的 Lambda 函数返回允许访问 Amazon IoT的策略文档。输入 URL 后，查询参数也会显示在 Postman 的 Params（参数）选项卡上。

   c. 在 Auth（身份验证）选项卡上的 Type（类型）字段中，选择 No Auth（无身份验证）。

   d. 在标头选项卡中：

      i. 如果选择了 Host（主机）密钥，请取消选中此项。

      ii. 在标头列表的底部添加这些新标头并确认它们被选中。使用您在上一节中用于 test-invoke-authorize命令的签名字符串，将 **Host**值替换为您的 *device_data_endpoint_address* 和 **x-amz-customauthorizer-signature**值。

键	值
x-amz-customauthorizer-name	**my-new-authorizer**
Host	*########*
tokenKeyName	**tokenKeyValue**
x-amz-customauthorizer-signature	*dbwykzlb+fo+jmsGdwoGr 8dyc2qb/ IyLef jjr+rbcvmu9jl4khaa9dg+v +v+mmwu09ysa86+64Y3GT4tOykpZqn 9mnvb1wyxp+0bdzh8Hmquauh3fwi3 cfPjBv a4cwnulqbZzbCvsluv 7i2iMjEg +cpy0zrwt1jr9rwi bikggpDxWkjaeehb qhhto357TegKs*

键	值
	9pp30uf4TrxypNmFsw a5k7qic01n4biyrtm90oyz94r4r4bdjJePgnu shnig1 obvmgcefe09jjjszehfgjszehfgbtqj16bu1xkptgsit +d1 yhbtvTvdthKt 8mjhzj0kgbt29vqqj1 jcb8riln/ p5+vcvnisxwpplyb5jkys9uvg08reoy64AtizfUhvSul r/f3vv8iTtQp 3aXiUtcsp aci6ca +tsDuXf 3LzCw qqf/ysuy02U5XkWn +sto6KCkpNlkd0wu8gl3+kozxrthnq8geajd5iylx230 o3osjpha7jdywm5O+ke 91i1mokdr5sj4jxixvnjtvs1li49ialw4en1dakc1a0s + aflFeloZl wqfHxRlXsPqi vks1ziucl1 azwprh/ord bJplpiWfBg iogokjidgp9gwhxiik7zWrGm wpmk9o=

e. 在“正文”选项卡中：

i. 在数据格式选项框中，选择 Raw（原始）。

ii. 在数据类型列表中，选择JavaScript。

iii. 在文本字段中，为测试消息输入此 JSON 消息负载：

```
{
    "data_mode": "test",
    "vibration": 200,
    "temperature": 40
}
```

4. 选择 Send（发送）以提交请求。

如果请求成功，则返回：

```
{
    "message": "OK",
    "traceId": "ff35c33f-409a-ea90-b06f-fbEXAMPLE25c"
}
```

成功的响应表明您的自定义授权方允许与 Amazon IoT的连接并且测试消息已经传送到 Amazon IoT Core中的代理处。

如果返回错误，请查看错误消息、*device_data_endpoint_address*、签名字符串和其它标头值。

将此请求保留在 Postman 中以供下一节使用。

步骤 6：在 MQTT 测试客户端中查看消息

在上一步中，您使用 Postman 将模拟设备消息发送到了 Amazon IoT。成功的响应表明您的自定义授权方允许与 Amazon IoT的连接并且测试消息已经传送到 Amazon IoT Core中的代理处。在本部分中，您将使用 Amazon IoT控制台中的 MQTT 测试客户端，以像其它设备和服务一样查看该消息中的消息内容。

要查看自定义授权方授权的测试消息

1. 在 Amazon IoT控制台中打开 MQTT 测试客户端。
2. 在 Subscribe to topic（订阅主题）选项卡中的 Topic filter（主题筛选条件）中，输入 **test/cust-auth/topic**，这是前一节中 Postman 示例使用的消息主题。

3. 选择 Subscribe。

   在下一步中，保持此窗口可见。

4. 在 Postman 中，在您为上一节创建的请求中，选择 Send（发送）。

   查看响应以确保响应成功。如果没有，请按照上一节所述对错误进行故障排除。

5. 在 MQTT test client（MQTT 测试客户端），您应该看到一个新条目，其中显示消息主题以及来自从 Postman 发送的请求（如果已展开）的消息负载。

   如果您未在 MQTT test client（MQTT 测试客户端）中看到您的消息，可以检查以下事项：

   - 确保您的 Postman 请求成功返回。如果 Amazon IoT拒绝连接并返回错误，则请求中的消息不会传递给消息代理。
   - 请确保用于打开 Amazon IoT控制台的 Amazon Web Services 账户和 Amazon Web Services 区域与您在 Postman URL 中使用的相同。
   - 请确保您已在 MQTT test client（MQTT 测试客户端）中正确输入主题。主题筛选条件区分大小写。如有疑问，您还可以订阅 #主题，该主题订阅通过Amazon Web Services 账户和Amazon Web Services 区域用于打开 Amazon IoT控制台的消息代理传递的所有 MQTT 消息。

步骤 7：查看结果和后续步骤

在本教程中：

- 您创建了一个 Lambda 函数作为自定义授权方处理程序
- 您创建了启用令牌签名的自定义授权方
- 您使用 test-invoke-authorizer命令测试了自定义授权方
- 您使用 Postman 发布了一个 MQTT 主题通过并使用您的自定义授权方验证请求
- 您使用了 MQTT test client（MQTT 测试客户端）查看 Postman 测试发送的消息

后续步骤

从 Postman 发送一些消息以验证自定义授权方是否正常工作后，请尝试试验以了解更改本教程的不同方面会对结果产生何种影响。以下是一些入门示例。

- 更改签名字符串，使其不再有权查看未经授权的连接尝试的处理方式。您应该得到一个错误响应，如此所示，并且该消息不应出现在 MQTT test client（MQTT 测试客户端）中。

```
{
    "message": "Forbidden",
    "traceId": "15969756-a4a4-917c-b47a-5433e25b1356"
}
```

- 要了解有关如何查找在开发和使用 Amazon IoT规则过程中可能会出现的错误，请参阅 监控 Amazon IoT (p. 394)。

步骤 8：清除

如果您想重复本教程，可能需要删除一些自定义授权方。您的 Amazon Web Services 账户一次只能配置有限数量的自定义授权方，当您尝试在不删除现有自定义授权方的情况下添加新的授权方时，您可以获得 `LimitExceededException`。

要删除自定义授权方（控制台）

1. 打开 Amazon IoT控制台的自定义授权方页面，然后在自定义授权方列表中找到要删除的自定义授权方。
2. 打开“自定义授权方详细信息”页面，然后从 Actions（操作）菜单中，选择 Edit（编辑）。
3. 取消选中 Activate authorizer（激活授权方），然后选择 Update（更新）。

   您无法在自定义授权方处于活动状态时将其删除。
4. 在“自定义授权方详细信息”页面，打开 Actions（操作）菜单，然后选择 Delete（删除）。

要删除自定义授权方（Amazon CLI）

1. 列出您已安装的自定义授权方，并查找要删除的自定义授权方的名称。

   ```
   aws iot list-authorizers
   ```
2. 在将 *Custom_Auth_Name* 替换为要删除的自定义授权方的 authorizerName后，运行此命令，以将自定义授权方设置为 inactive。

   ```
   aws iot update-authorizer --status INACTIVE --authorizer-name Custom_Auth_Name
   ```
3. 在将 *Custom_Auth_Name* 替换为要删除的自定义授权方的 authorizerName后，运行此命令，以删除自定义授权方。

   ```
   aws iot delete-authorizer --authorizer-name Custom_Auth_Name
   ```

教程：使用 Amazon IoT和 Raspberry Pi 监控土壤含水量

本教程介绍了如何使用 Raspberry Pi、含水量传感器和 Amazon IoT监测室内植物或花园的土壤含水量。Raspberry Pi 运行的代码从传感器读取含水量和温度，然后将数据发送到 Amazon IoT。您在 Amazon IoT中创建一条规则：当含水量低于阈值时，向订阅 Amazon SNS 主题的地址发送电子邮件。

> **Note**
>
> 本教程可能不是最新版本。自本主题最初发布以来，一些参考文献可能已被取代。

目录

- 先决条件 (p. 230)
- 设置 Amazon IoT (p. 231)
  - 步骤 1：创建 Amazon IoT策略 (p. 231)
  - 步骤 2：创建 Amazon IoT事物、证书和私有密钥 (p. 232)
  - 步骤 3：创建 Amazon SNS 主题和订阅 (p. 233)
  - 步骤 4：创建 Amazon IoT规则来发送电子邮件 (p. 233)
- 设置您的 Raspberry Pi 和含水量传感器 (p. 234)

先决条件

要完成此教程，需要：

- 一个 Amazon Web Services 账户。

- 具有管理员权限的 IAM 用户。
- 运行 Windows、macOS、Linux 或 Unix 的开发计算机（用于访问 Amazon IoT控制台）。
- 运行最新 Raspbian OS 的 Raspberry Pi 3B 或 4B。有关安装说明，请参阅 Rasberry Pi 网站上的安装操作系统映像。
- 用于 Raspberry Pi 的显示器、键盘、鼠标和 Wi-Fi 网络或以太网连接。
- 与 Raspberry Pi 兼容的含水量传感器。本教程中使用的传感器是 Adafruit STEMMA I2C 电容式含水量传感器，带有 JST 4 针转母插座电缆接头。

设置 Amazon IoT

要完成本教程，您需要创建以下资源。要将设备连接到 Amazon IoT，您需要创建 IoT 事物、设备证书和 Amazon IoT策略。

- Amazon IoT 事物。

  事物代表物理设备（在本例中为 Rasberry Pi），包括有关该设备的静态元数据。
- 设备证书。

  所有设备都必须拥有设备证书才能连接到 Amazon IoT和通过其身份验证。
- Amazon IoT 策略。

  每个设备证书都有一个或多个与其关联的 Amazon IoT策略。这些策略确定设备可以访问哪些 Amazon IoT 资源。
- Amazon IoT 根 CA 证书。

  设备和其它客户端使用 Amazon IoT根 CA 证书来对与其通信的 Amazon IoT服务器进行身份验证。有关更多信息，请参阅 服务器身份验证 (p. 264)。
- Amazon IoT 规则。

  规则包含查询和一个或多个规则操作。查询从设备消息中提取数据，以确定是否应处理该消息数据。规则操作指定当数据与查询匹配时该做些什么。
- 创建 Amazon SNS 主题和主题订阅。

  该规则侦听来自 Raspberry Pi 的含水量数据。如果该值低于阈值，它会向 Amazon SNS 主题发送消息。Amazon SNS 会将该消息发送到订阅该主题的所有电子邮件地址。

步骤 1：创建 Amazon IoT策略

创建 Amazon IoT策略，以允许您的 Raspberry Pi 连接到 Amazon IoT和向其发送消息。

1. 在 Amazon IoT控制台中，如果显示 Get started (开始使用) 按钮，请选择该按钮。否则，请在导航窗格中展开 Secure（安全），然后选择 Policies（策略）。
2. 如果显示您还没有任何策略对话框，请选择创建策略。否则，选择 Create。
3. 输入 Amazon IoT策略的名称（例如 **`MoistureSensorPolicy`**）。
4. 在添加声明部分中，将现有策略替换为以下 JSON。使用您的Amazon Web Services 区域和Amazon Web Services 账户账号替换*##*和*##*。

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

```
            "Action": "iot:Connect",
            "Resource": "arn:aws:iot:region:account:client/RaspberryPi"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Publish",
            "Resource": [
                "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/update",
                "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/delete",
                "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": [
                "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/update/
accepted",
                "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/delete/
accepted",
                "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/get/
accepted",
                "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/update/
rejected",
                "arn:aws:iot:region:account:topic/$aws/things/RaspberryPi/shadow/delete/
rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
update/accepted",
                "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
delete/accepted",
                "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/get/
accepted",
                "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
update/rejected",
                "arn:aws:iot:region:account:topicfilter/$aws/things/RaspberryPi/shadow/
delete/rejected"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow"
            ],
            "Resource": "arn:aws:iot:region:account:thing/RaspberryPi"

        }
    ]
}
```

5. 选择 Create（创建）。

步骤 2：创建 Amazon IoT事物、证书和私有密钥

在 Amazon IoT注册表中创建事物来代表您的 Raspberry Pi。

1. 在 Amazon IoT控制台的导航窗格中，依次选择 Manage (管理) 和 Things (事物)。

2. 如果显示您还没有任何事物对话框，请选择注册事物。否则，选择 Create。
3. 在创建 Amazon IoT事物页面上，选择创建单个事物。
4. 在 Add your device to the device registry (将您的设备添加到设备注册表) 页面上，输入您的 IoT 事物的名称（例如 **RaspberryPi**），然后选择 Next (下一步)。您无法在创建事物后更改其名称。要更改事物的名称，您必须创建一个新事物，为其指定新名称，然后删除旧事物。
5. 在添加事物的证书页面上，选择创建证书。
6. 选择下载链接以下载证书、私有密钥和根 CA 证书。

   Important

   这是您下载证书和私有密钥的唯一机会。
7. 选择 Activate（激活）来激活您的证书。证书必须处于活动状态，设备才能连接到 Amazon IoT。
8. 选择附加策略。
9. 对于 Add a policy for your thing (添加事物的策略)，选择 MoistureSensorPolicy，然后选择 Register Thing (注册事物)。

步骤 3：创建 Amazon SNS 主题和订阅

创建 Amazon SNS 主题和订阅。

1. 从 AmazonSNS 控制台的导航窗格中，选择 Topics（主题），然后选择 Create topic（创建主题）。
2. 输入主题的名称（例如，**MoistureSensorTopic**）。
3. 输入主题的显示名称（例如，**Moisture Sensor Topic**）。这是在 Amazon SNS 控制台中为您的主题显示的名称。
4. 选择 Create topic（创建主题）。
5. 在 Amazon SNS 主题详细信息页面上，选择 Create subscription（创建订阅）。
6. 对于协议，选择电子邮件。
7. 对于 Endpoint (终端节点)，输入您的电子邮件地址。
8. 选择 Create subscription（创建订阅）。
9. 打开您的电子邮件客户端，查找主题为 **MoistureSensorTopic**的消息。打开这封电子邮件，然后单击 Confirm subscription (确认订阅) 链接。

   Important

   在确认订阅之前，您不会收到来自该 Amazon SNS 主题的任何电子邮件告警。

您应该会收到一封电子邮件，其中包含您输入的文本。

步骤 4：创建 Amazon IoT规则来发送电子邮件

Amazon IoT 规则定义了查询以及从设备收到消息时要执行的一个或多个操作。Amazon IoT 规则引擎侦听设备发送的消息，并使用消息中的数据确定是否应采取某些操作。有关更多信息，请参阅 Amazon IoT 规则 (p. 442)。

在本教程中，您的 Raspberry Pi 在 `aws/things/RaspberryPi/shadow/update`上发布消息。这是设备和 Thing Shadow 服务使用的内部 MQTT 主题。Raspberry Pi 发布具有以下形式的消息：

```
{
    "reported": {
        "moisture" : moisture-reading,
        "temp" : temperature-reading
    }
}
```

创建查询来从传入消息中提取含水量和温度数据。创建 Amazon SNS 操作来在含水量读数低于阈值时获取数据并将其发送给 Amazon SNS 主题订阅者。

创建 Amazon SNS 规则

1. 在 Amazon IoT控制台的导航窗格中，选择 Act (行动)。如果显示您还没有任何规则对话框，请选择创建规则。否则，选择 Create。
2. 在创建规则页面上，为此规则输入名称（例如，**MoistureSensorRule**）。
3. 对于描述，请为此规则提供一个简短的描述（例如，**Sends an alert when soil moisture level readings are too low**）。
4. 在 Rule query statement (规则查询语句) 下，选择 SQL 版本 2016-03-23，然后输入以下 Amazon IoTSQL 查询语句：

```
SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE
 state.reported.moisture < 400
```

当 moisture读数小于 400时，该语句将触发规则操作。

> **Note**
>
> 您可能需要使用其它值。在 Raspberry Pi 上运行代码之后，您可以通过触摸传感器、将其放入水中或放入花盆中来查看从传感器获得的值。

5. 在 Set one or more actions (设置一个或多个操作) 下，选择 Add action (添加操作)。
6. 在选择操作页面上，选择将消息发送为 SNS 推送通知。
7. 滚动到页面底部，然后选择 Configure action (配置操作)。
8. 在配置操作页面上，对于 SNS 目标，选择 Select (选择)，然后选择 LowMoistureTopic。
9. 对于 Message format，请选择 RAW。
10. 在 Choose or create a role to grant Amazon IoTaccess to perform this action（选择或创建角色以向 IoT 授予执行此操作的权限）下，选择 Create Role（创建角色）。输入角色的名称（例如 **LowMoistureTopicRole**），然后选择 Create role (创建角色)。
11. 选择添加操作。
12. 请选择 Create rule (创建规则)。

设置您的 Raspberry Pi 和含水量传感器

将 Micro SD 卡插入 Raspberry Pi 中，连接显示器、键盘、鼠标以及以太网电缆（如果您未使用 Wi-Fi）。先不要连接电源线。

将 JST 跳线连接到含水量传感器。跳线的另一端有四根电线：

- 绿色：I2C SCL
- 白色：I2C SDA
- 红色：电源 (3.5 V)
- 黑色：接地

握住 Raspberry Pi（以太网插孔位于右侧）。以该方向握持时，可看到顶部有两排 GPIO 引脚。按照以下顺序将含水量传感器的电线连接到下排引脚。从最左侧的引脚开始，连接红色（电源）、白色 (SDA) 和绿色 (SCL) 电线。跳过一个引脚，然后连接黑色（接地）电线。有关更多信息，请参阅 Python 计算机接线。

将电源线连接到 Raspberry Pi，然后将另一端插入墙壁插座以将其打开。

配置您的 Raspberry Pi

1. 在 Welcome to Raspberry Pi (欢迎使用 Raspberry Pi) 页面上，选择 Next (下一步)。
2. 选择您的国家/地区、语言、时区和键盘布局。选择 Next (下一步) 。
3. 输入 Raspberry Pi 的密码，然后选择 Next (下一步)。
4. 选择您的 Wi-Fi 网络，然后选择 Next (下一步)。如果不使用 Wi-Fi 网络，则选择 Skip (跳过)。
5. 选择 Next (下一步) 检查软件更新。完成更新后，选择 Restart (重启) 重启您的 Raspberry Pi。

在 Raspberry Pi 启动后，启用 I2C 接口。

1. 在 Raspbian 桌面的左上角，单击 Raspberry 图标，选择 Preferences (首选项)，然后选择 Raspberry Pi Configuration (Raspberry Pi 配置)。
2. 在 Interfaces (接口) 选项卡上，对于 I2C，选择 Enable (启用)。
3. 选择 OK (确定) 。

Adafruit STEMMA 含水量传感器的库是为 CircuitPython 编写的。要在 Raspberry Pi 上运行它们，需要安装最新版本的 Python 3。

1. 在命令提示符中运行以下命令来更新 Raspberry Pi 软件：

   ```
   sudo apt-get update
   ```

   ```
   sudo apt-get upgrade
   ```
2. 运行以下命令，更新您的 Python 3 安装：

   ```
   sudo pip3 install --upgrade setuptools
   ```
3. 运行以下命令，安装 Raspberry Pi GPIO 库：

   ```
   pip3 install RPI.GPIO
   ```
4. 运行以下命令，安装 Adafruit Blinka 库：

   ```
   pip3 install adafruit-blinka
   ```

   有关更多信息，请参阅在 Raspberry Pi 上安装 CircuitPython 库。
5. 运行以下命令，安装 Adafruit Seesaw 库：

   ```
   sudo pip3 install adafruit-circuitpython-seesaw
   ```
6. 运行以下命令，安装 Amazon IoTDevice SDK for Python：

   ```
   pip3 install AWSIoTPythonSDK
   ```

Raspberry Pi 现已具备所有必需的库。创建一个名为 **moistureSensor.py**的文件，并将以下 Python 代码复制到该文件中：

```
from adafruit_seesaw.seesaw import Seesaw
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTShadowClient
from board import SCL, SDA

import logging
import time
import json
import argparse
import busio
```

```
# Shadow JSON schema:
#
# {
#   "state": {
#       "desired":{
#           "moisture":<INT VALUE>,
#           "temp":<INT VALUE>
#       }
#   }
# }

# Function called when a shadow is updated
def customShadowCallback_Update(payload, responseStatus, token):

    # Display status and data from update request
    if responseStatus == "timeout":
        print("Update request " + token + " time out!")

    if responseStatus == "accepted":
        payloadDict = json.loads(payload)
        print("~~~~~~~~~~~~~~~~~~~~~~~~")
        print("Update request with token: " + token + " accepted!")
        print("moisture: " + str(payloadDict["state"]["reported"]["moisture"]))
        print("temperature: " + str(payloadDict["state"]["reported"]["temp"]))
        print("~~~~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Update request " + token + " rejected!")

# Function called when a shadow is deleted
def customShadowCallback_Delete(payload, responseStatus, token):

     # Display status and data from delete request
    if responseStatus == "timeout":
        print("Delete request " + token + " time out!")

    if responseStatus == "accepted":
        print("~~~~~~~~~~~~~~~~~~~~~~~~")
        print("Delete request with token: " + token + " accepted!")
        print("~~~~~~~~~~~~~~~~~~~~~~~~\n\n")

    if responseStatus == "rejected":
        print("Delete request " + token + " rejected!")


# Read in command-line parameters
def parseArgs():

    parser = argparse.ArgumentParser()
    parser.add_argument("-e", "--endpoint", action="store", required=True, dest="host",
 help="Your device data endpoint")
    parser.add_argument("-r", "--rootCA", action="store", required=True, dest="rootCAPath",
 help="Root CA file path")
    parser.add_argument("-c", "--cert", action="store", dest="certificatePath",
 help="Certificate file path")
    parser.add_argument("-k", "--key", action="store", dest="privateKeyPath", help="Private
 key file path")
    parser.add_argument("-p", "--port", action="store", dest="port", type=int, help="Port
 number override")
    parser.add_argument("-n", "--thingName", action="store", dest="thingName",
 default="Bot", help="Targeted thing name")
    parser.add_argument("-id", "--clientId", action="store", dest="clientId",
 default="basicShadowUpdater", help="Targeted client id")

    args = parser.parse_args()
    return args
```

```
# Configure logging
# AWSIoTMQTTShadowClient writes data to the log
def configureLogging():

    logger = logging.getLogger("AWSIoTPythonSDK.core")
    logger.setLevel(logging.DEBUG)
    streamHandler = logging.StreamHandler()
    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
    streamHandler.setFormatter(formatter)
    logger.addHandler(streamHandler)


# Parse command line arguments
args = parseArgs()

if not args.certificatePath or not args.privateKeyPath:
    parser.error("Missing credentials for authentication.")
    exit(2)

# If no --port argument is passed, default to 8883
if not args.port:
    args.port = 8883


# Init AWSIoTMQTTShadowClient
myAWSIoTMQTTShadowClient = None
myAWSIoTMQTTShadowClient = AWSIoTMQTTShadowClient(args.clientId)
myAWSIoTMQTTShadowClient.configureEndpoint(args.host, args.port)
myAWSIoTMQTTShadowClient.configureCredentials(args.rootCAPath, args.privateKeyPath,
 args.certificatePath)

# AWSIoTMQTTShadowClient connection configuration
myAWSIoTMQTTShadowClient.configureAutoReconnectBackoffTime(1, 32, 20)
myAWSIoTMQTTShadowClient.configureConnectDisconnectTimeout(10) # 10 sec
myAWSIoTMQTTShadowClient.configureMQTTOperationTimeout(5) # 5 sec

# Initialize Raspberry Pi's I2C interface
i2c_bus = busio.I2C(SCL, SDA)

# Intialize SeeSaw, Adafruit's Circuit Python library
ss = Seesaw(i2c_bus, addr=0x36)

# Connect to Amazon IoT
myAWSIoTMQTTShadowClient.connect()

# Create a device shadow handler, use this to update and delete shadow document
deviceShadowHandler = myAWSIoTMQTTShadowClient.createShadowHandlerWithName(args.thingName,
 True)

# Delete current shadow JSON doc
deviceShadowHandler.shadowDelete(customShadowCallback_Delete, 5)

# Read data from moisture sensor and update shadow
while True:

    # read moisture level through capacitive touch pad
    moistureLevel = ss.moisture_read()

    # read temperature from the temperature sensor
    temp = ss.get_temp()

    # Display moisture and temp readings
    print("Moisture Level: {}".format(moistureLevel))
    print("Temperature: {}".format(temp))
```

```
    # Create message payload
    payload = {"state":{"reported":{"moisture":str(moistureLevel),"temp":str(temp)}}}

    # Update shadow
    deviceShadowHandler.shadowUpdate(json.dumps(payload), customShadowCallback_Update, 5)
    time.sleep(1)
```

将文件保存到您可以找到的位置。在命令行中运行 moistureSensor.py及以下参数：

endpoint

自定义 Amazon IoT终端节点。有关更多信息，请参阅 Device Shadow REST API (p. 616)。

rootCA

您的 Amazon IoT根 CA 证书的完整路径。

cert

您的 Amazon IoT设备证书的完整路径。

key

您的 Amazon IoT设备证书私有密钥的完整路径。

thingName

您的事物的名称（在本例中为 RaspberryPi）。

clientId

MQTT 客户端 ID。使用 RaspberryPi。

命令行应如下所示：

```
python3 moistureSensor.py --endpoint your-endpoint --rootCA ~/certs/
AmazonRootCA1.pem --cert ~/certs/raspberrypi-certificate.pem.crt --key ~/certs/
raspberrypi-private.pem.key --thingName RaspberryPi --clientId RaspberryPi
```

尝试触摸传感器、将其放入花盆或放入一杯水中，查看传感器对各种含水量有何反应。如果需要，可以在 MoistureSensorRule中更改阈值。当含水量传感器的读数低于规则的 SQL 查询语句中指定的值时，Amazon IoT 会向 Amazon SNS 主题发布消息。您应会收到一封包含含水量和温度数据的电子邮件。

确认收到来自 Amazon SNS 的电子邮件后，按 CTRL + C 停止 Python 程序。该 Python 程序发送的消息应该不足以产生费用，但完成后最好将其停止。

使用 Amazon IoT 管理设备

Amazon IoT 提供了注册表来帮助您管理事物。事物是特定设备或逻辑实体的表示形式。它可以是物理设备或传感器（例如，灯泡或墙壁上的开关）。此外，它也可以是逻辑实体（如应用程序实例），或没有连接到 Amazon IoT 但与其它连接到的设备相关的物理实体（例如，装有发动机传感器的汽车或控制面板）。

事物的相关信息均以 JSON 数据形式存储在 Registry 中。以下是一个事物示例：

```
{
     "version": 3,
    "thingName": "MyLightBulb",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "LightBulb",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

事物由名称进行标识。此外，事物还可以具有名称-值对形式的属性，您可以利用这些属性存储事物的相关信息，如事物的序列号或制造商。

典型的设备使用案例使用事物名称作为默认的 MQTT 客户端 ID。虽然我们没有强制在事物的注册表名称和它使用的 MQTT 客户端 ID、证书或影子状态之间进行映射，但建议您选择一个事物名称，并将其同时用作注册表和 Device Shadow 服务的 MQTT 客户端 ID。这不仅为您的 IoT 实例集带来了有序性和便利性，还保留了基础设备证书模型或影子的灵活性。

您无需在注册表中创建事物便能将设备连接到 Amazon IoT。将事物添加到注册表可让您更轻松地管理和搜索设备。

如何使用注册表管理事物

您可以使用 Amazon IoT 控制台、Amazon IoT API 或 Amazon CLI 与注册表进行交互。以下各部分展示了如何使用 CLI 与 Registry 进行交互。

命名事物对象时：

- 请勿在事物名称中使用个人可识别信息。事物名称可以出现在未加密的通信和报告中。

创建事物

以下命令说明如何使用 Amazon IoTCreateThing 命令从 CLI 创建事物。您无法在创建事物后更改其名称。要更改事物的名称，您必须创建一个新事物，为其指定新名称，然后删除旧事物。

```
$ aws iot create-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

CreateThing 命令显示新事物的名称和 Amazon Resource Name (ARN)：

```
{
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
```

```
    "thingName": "MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678"
}
```

Note

我们建议不要在您的事物名称中使用个人身份信息。

有关更多信息，请参阅《Amazon CLI 命令参考》中的 create-thing。

列出事物

您可以使用 ListThings 命令列出您的账户中的所有事物：

```
$ aws iot list-things
```

```
{
    "things": [
       {
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyLightBulb"
        },
        {
            "attributes": {
                "numOfStates":"3"
             },
            "version": 11,
            "thingName": "MyWallSwitch"
        }
    ]
}
```

您可以使用 ListThings 命令搜索某个事物类型的所有事物：

```
$  aws iot list-things --thing-type-name "LightBulb"
```

```
{
    "things": [
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
```

```
        }
    ]
}
```

您可以使用 ListThings 命令搜索某个属性为特定值的所有事物。此命令最多可搜索三个属性。

```
$  aws iot list-things --attribute-name "wattage" --attribute-value "75"
```

```
{
    "things": [
        {
            "thingTypeName": "StopLight",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 3,
            "thingName": "MyLightBulb"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MyRGBLight"
        },
        {
            "thingTypeName": "LightBulb",
            "attributes": {
                "model": "123",
                "wattage": "75"
            },
            "version": 1,
            "thingName": "MySecondLightBulb"
        }
    ]
}
```

有关更多信息，请参阅《Amazon CLI 命令参考》中的 list-things。

描述事物

您可以使用 DescribeThing 命令获取某事物的更多详情：

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "version": 3,
    "thingName": "MyLightBulb",
    "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
    "thingId": "12345678abcdefgh12345678ijklmnop12345678",
    "defaultClientId": "MyLightBulb",
    "thingTypeName": "StopLight",
    "attributes": {
        "model": "123",
        "wattage": "75"
    }
}
```

有关更多信息，请参阅《Amazon CLI 命令参考》中的 describe-thing。

更新事物

您可以使用 UpdateThing 命令更新事物。请注意，此命令仅更新事物的属性。您无法更改事物的名称。要更改事物的名称，您必须创建一个新事物，为其指定新名称，然后删除旧事物。

```
$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\":
 {\"wattage\":\"150\", \"model\":\"456\"}}"
```

UpdateThing 命令不会生成任何输出。您可以使用 DescribeThing 命令查看结果：

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
    "attributes": {
        "model": "456",
        "wattage": "150"
    },
    "version": 2,
    "thingName": "MyLightBulb"
}
```

有关更多信息，请参阅《Amazon CLI 命令参考》中的 update-thing。

删除事物

您可以使用 DeleteThing 命令删除事物：

```
$ aws iot delete-thing --thing-name "MyThing"
```

如果删除成功或者您指定的事物不存在，此命令将成功返回且没有错误。

有关更多信息，请参阅《Amazon CLI 命令参考》中的 delete-thing。

将委托人附加到事物

物理设备必须具有 X.509 证书才能与 Amazon IoT 通信。您可以将设备上的证书与 Registry 中代表该设备的事物关联。要将证书附加到您的事物，请使用 AttachThingPrincipal 命令：

```
$ aws iot attach-thing-principal --thing-name "MyLightBulb" --principal "arn:aws:iot:us-
east-1:123456789012:cert/a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

AttachThingPrincipal 命令不会生成任何输出。

有关更多信息，请参阅《Amazon CLI 命令参考》中的 attach-thing-principal。

将委托人与事物分离

您可以使用 DetachThingPrincipal 命令将证书与事物分离：

```
$ aws iot detach-thing-principal --thing-name "MyLightBulb" --principal "arn:aws:iot:us-
east-1:123456789012:cert/a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

DetachThingPrincipal 命令不会生成任何输出。

有关更多信息，请参阅《Amazon CLI 命令参考》中的 detach-thing-principal。

事物类型

通过定义事物类型，您可以存储与同一事物类型相关联的所有事物共有的描述和配置信息，这将简化 Registry 中的事物管理。例如，您可以定义 LightBulb 事物类型。所有与 LightBulb 事物类型相关联的事物均具有一组共同的属性：序列号、制造商和功率。创建 LightBulb 类型的事物（或将现有事物的类型更改为 LightBulb）时，您可以指定在 LightBulb 事物类型中定义的每个属性的值。

虽然事物类型是可选项，但使用它们可以更容易地搜索事物。

- 具有事物类型的事物最多可以具有 50 个属性。
- 未设置事物类型的事物最多可以具有三个属性。
- 一个事物只能与一种事物类型相关联。
- 在账户中可以创建的事物类型数量不限。

事物类型是不可变的。事物类型一旦创建，便不可更改。您随时可以弃用某事物类型，以防止新事物与之关联。您也可以删除没有与任何事物关联的事物类型。

创建事物类型

您可以使用 CreateThingType 命令创建事物类型：

```
$ aws iot create-thing-type

                --thing-type-name "LightBulb" --thing-type-properties
 "thingTypeDescription=light bulb type, searchableAttributes=wattage,model"
```

CreateThingType 命令会返回一个含有事物类型及其 ARN 的响应：

```
{
    "thingTypeName": "LightBulb",
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb"
}
```

列出事物类型

您可以使用 ListThingTypes 命令列出事物类型：

```
$ aws iot list-thing-types
```

ListThingTypes 命令会返回一个列有您的Amazon Web Services 账户中定义的事物类型的列表：

```
{
    "thingTypes": [
        {
            "thingTypeName": "LightBulb",
            "thingTypeProperties": {
                "searchableAttributes": [
                    "wattage",
                    "model"
                ],
```

```
                "thingTypeDescription": "light bulb type"
            },
            "thingTypeMetadata": {
                "deprecated": false,
                "creationDate": 1468423800950
            }
        }
    ]
}
```

描述事物类型

您可以使用 DescribeThingType 命令获取某事物类型的相关信息：

```
$ aws iot describe-thing-type --thing-type-name "LightBulb"
```

DescribeThingType 命令返回有关指定类型的信息：

```
{
    "thingTypeProperties": {
        "searchableAttributes": [
            "model",
            "wattage"
        ],
        "thingTypeDescription": "light bulb type"
    },
    "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
    "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb",
    "thingTypeName": "LightBulb",
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1544466338.399
    }
}
```

将事物类型与事物相关联

创建事物时，可以使用 CreateThing 命令指定事物类型：

```
$ aws iot create-thing --thing-name "MyLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\":\"75\", \"model\":\"123\"}}"
```

您随时可以使用 UpdateThing 命令更改与某个事物关联的事物类型：

```
$ aws iot update-thing --thing-name "MyLightBulb"
                --thing-type-name "LightBulb" --attribute-payload  "{\"attributes\":
 {\"wattage\":\"75\", \"model\":\"123\"}}"
```

您还可以使用 UpdateThing 命令取消事物与事物类型之间的关联。

弃用事物类型

事物类型是不可变的。一旦定义了事务类型，便不可更改。然而，您可以通过弃用某种事物类型来防止用户将新事物与之关联。所有与该事物类型相关联的现有事物将保持不变。

要弃用某事物类型，请使用 DeprecateThingType 命令：

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType"
```

您可以使用 DescribeThingType 命令查看结果：

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type",
    },
    "thingTypeMetadata": {
        "deprecated": true,
        "creationDate": 1468425854308,
        "deprecationDate": 1468446026349
    }
}
```

弃用事物类型是可逆操作。您可以通过在 `--undo-deprecate` CLI 命令中使用 DeprecateThingType 标志来撤消弃用：

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType" --undo-deprecate
```

您可以使用 DescribeThingType CLI 命令查看结果：

```
$ aws iot describe-thing-type --thing-type-name "StopLight":
```

```
{
    "thingTypeName": "StopLight",
    "thingTypeArn": "arn:aws:iot:us-east-1:123456789012:thingtype/StopLight",
    "thingTypeId": "12345678abcdefgh12345678ijklmnop12345678"
    "thingTypeProperties": {
        "searchableAttributes": [
            "wattage",
            "numOfLights",
            "model"
        ],
        "thingTypeDescription": "traffic light type"
    },
    "thingTypeMetadata": {
        "deprecated": false,
        "creationDate": 1468425854308,
    }
}
```

删除事物类型

只有在弃用事物类型后，才能将其删除。要删除某事物类型，请使用 DeleteThingType 命令：

```
$ aws iot delete-thing-type --thing-type-name "StopLight"
```

Note

弃用某事物类型后，必须等待 5 分钟，然后才能将其删除。

静态事物组

通过将事物分类成组，您能够使用静态事物组同时管理几个事物。静态事物组包含使用控制台、CLI 或 API 管理的一组事物。另一方面，动态事物组 (p. 254)包含与指定查询匹配的事物。静态事物组还可以包含其它静态组 — 您可以构建组的层次结构。您可以将策略附加到某个父组，该策略将由其所有子组和该组及其子组中的所有事物继承。这使得控制大量事物的权限非常简单。

下面是您可以对静态事物组执行的操作：

- 创建、描述或删除组。
- 将事物添加到一个或多个组。
- 从组中删除事物。
- 列出您已创建的组。
- 列出组的所有子组 (其直接和间接后代)。
- 列出组中的事物，包括其子组中的所有事物。
- 列出组的所有原级组 (其直接和间接父级)。
- 添加、删除或更新组的属性。(属性是您可用于存储与组相关的信息的名称/值对。)
- 从组中附加或分离策略。
- 列出附加到组的策略。
- 列出由某个事物继承的策略 (借助附加到其组或其父组之一的策略)。
- 为组中的事物配置日志记录选项。请参阅配置 Amazon IoT 日志记录 (p. 394)。
- 创建将发送到某个组及其子组中的所有事物并在这些事物上执行的任务。请参阅任务 (p. 637)。

下面是静态事物组的一些限制：

- 一个组最多可以有一个直接父级。
- 如果一个组是另一个组的子组，则必须在创建该子组时指定这一关系。
- 以后无法更改组的父组；因此，务必先规划组的层次结构并创建父组，然后再创建父组中将包含的任何子组。
- 一个事物可以属于的组数受到限制。
- 您无法将事物添加到相同层次结构的多个组中。（换言之，您无法将事物添加到父级相同的两个组中。）
- 您无法重命名组。
- 事物组名称不能包含国际字符，如 û、é 和 ñ。
- 请勿在您的事物组名称中使用个人可识别信息。事物组名称可以出现在未加密的通信和报告中。

在组上附加和分离策略能够以多种非常有效的方式增强 Amazon IoT 操作的安全性。针对各个设备将策略附加到证书，然后附加到事物的方法非常费时，并且难于在整个设备实例集中快速更新或更改策略。在需要轮换某个事物上的证书时，将策略附加到事物的组可以节省步骤。并且，在更改组成员资格时，策略会动态应用到事物，因此您无需在每次组中设备成员资格更改时重新创建复杂的权限集。

创建静态事物组

使用 CreateThingGroup 命令创建静态事物组：

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

CreateThingGroup 命令返回包含静态事物组名称、其 ID 和 ARN 的响应：

```
{
    "thingGroupName": "LightBulbs",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
}
```

> **Note**
>
> 我们建议不要在您的事物组名称中使用个人身份信息。

以下示例在创建静态事物组时指定其父级：

```
$ aws iot create-thing-group --thing-group-name RedLights --parent-group-name LightBulbs
```

和之前一样，CreateThingGroup 命令返回包含静态事物组名称、其 ID 和 ARN 的响应：

```
{
    "thingGroupName": "RedLights",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

> **Important**
>
> 创建事物组层次结构时，请注意以下限制：
>
> - 一个事物组只可以有一个直接父级。
> - 事物组可以拥有的直接子组数量受到限制。
> - 组层次结构的最大深度受到限制。
> - 事物组可以拥有的属性数量受到限制。(属性是您可用于存储与组相关的信息的名称/值对。) 每个属性名称和每个值的长度也受到限制。

描述事物组

您可以使用 DescribeThingGroup 命令获取某事物组的相关信息：

```
$ aws iot describe-thing-group --thing-group-name RedLights
```

DescribeThingGroup 命令返回有关指定组的信息：

```
{
    "thingGroupName": "RedLights",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
    "thingGroupId": "12345678abcdefgh12345678ijklmnop12345678",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1478299948.882
        "parentGroupName": "Lights",
        "rootToParentThingGroups": [
            {
```

```
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/ShinyObjects",
                "groupName": "ShinyObjects"
            },
            {
                "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs",
                "groupName": "LightBulbs"
            }
        ]
    },
    "thingGroupProperties": {
        "attributePayload": {
            "attributes": {
                "brightness": "3400_lumens"
            },
        },
        "thingGroupDescription": "string"
    },
}
```

将事物添加到静态事物组

您可以使用 AddThingToThingGroup 命令将事物添加到静态事物组：

```
$ aws iot add-thing-to-thing-group --thing-name MyLightBulb --thing-group-name RedLights
```

AddThingToThingGroup 命令不会生成任何输出。

> **Important**
>
> 您可以将事物添加到最多 10 个组。但是，您无法将事物添加到相同层次结构的多个组中。(换而言之，您无法将事物添加到共享相同父级的两个组中。)
> 如果一个事物属于尽可能多的事物组，并且其中一个或多个组是动态事物组，则可以使用 overrideDynamicGroups 标志使静态组优先于动态组。

从静态事物组中删除事物

您可以使用 RemoveThingFromThingGroup 命令从组中删除事物：

```
$ aws iot remove-thing-from-thing-group --thing-name MyLightBulb --thing-group-name
 RedLights
```

RemoveThingFromThingGroup 命令不会生成任何输出。

列出事物组中的事物

您可以使用 ListThingsInThingGroup 命令列出属于某个组的事物：

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs
```

ListThingsInThingGroup 命令返回指定组中的事物列表：

```
{
    "things":[
        "TestThingA"
    ]
}
```

使用 `--recursive` 参数，您可以列出属于某个组及其所有子组中的事物：

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs --recursive
```

```
{
    "things":[
        "TestThingA",
        "MyLightBulb"
    ]
}
```

Note

此操作具有最终一致性。换句话说，可能不会立即反映事物组的更改。

列出事物组

您可以使用 ListThingGroups 命令列出您账户的事物组：

```
$ aws iot list-thing-groups
```

ListThingGroups 命令会返回您Amazon Web Services 账户中的事物组的列表：

```
{
    "thingGroups": [
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/ReplaceableObjects"
        }
    ]
}
```

使用可选的筛选条件列出具有指定组作为父级 (`--parent-group`) 的组，或者名称以指定前缀 (`--name-prefix-filter`) 开头的组。使用 `--recursive` 参数可以列出所有子组，而不仅仅是事物组的直接子组：

```
$ aws iot list-thing-groups --parent-group LightBulbs
```

在这种情况下，ListThingGroups 命令返回在您的Amazon Web Services 账户中定义的事物组的直接子组列表：

```
{
```

```
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        }
    ]
}
```

将 `--recursive` 参数与 ListThingGroups 命令结合使用可以列出事物组的所有子组，而不仅仅是直接子组：

```
$ aws iot list-thing-groups --parent-group LightBulbs --recursive
```

ListThingGroups 命令返回事物组的所有子组的列表：

```
{
    "childGroups":[
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "RedLEDLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLEDLights"
        },
        {
            "groupName": "RedIncandescentLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedIncandescentLights"
        }
    ]
}
```

> **Note**
>
> 此操作具有最终一致性。换句话说，可能不会立即反映事物组的更改。

列出事物的组

您可以使用 ListThingGroupsForThing 命令列出事物所属的直接组：

```
$ aws iot list-thing-groups-for-thing --thing-name MyLightBulb
```

ListThingGroupsForThing 命令会返回此事物所属的直接事物组列表：

```
{
    "thingGroups":[
        {
            "groupName": "LightBulbs",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
        },
        {
            "groupName": "RedLights",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
        },
        {
            "groupName": "ReplaceableObjects",
            "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/ReplaceableObjects"
        }
```

```
    ]
}
```

更新静态事物组

您可以使用 UpdateThingGroup 命令更新静态事物组的属性：

```
$ aws iot update-thing-group --thing-group-name "LightBulbs" --thing-group-properties
 "thingGroupDescription=\"this is a test group\", attributePayload=\"{\"attributes
\"={\"Owner\"=\"150\",\"modelNames\"=\"456\"}}"
```

UpdateThingGroup 命令返回一个响应，其中包含该组更新后的版本号：

```
{
    "version": 4
}
```

> **Note**
>
> 事物可以拥有的属性数量受到限制。

删除事物组

要删除事物组，请使用 DeleteThingGroup 命令：

```
$ aws iot delete-thing-group --thing-group-name "RedLights"
```

DeleteThingGroup 命令不会生成任何输出。

> **Important**
>
> 如果您尝试删除具有子事物组的事物组，则会收到错误：
>
> ```
> A client error (InvalidRequestException) occurred when calling the
>  DeleteThingGroup
> operation: Cannot delete thing group : RedLights when there are still child groups
>  attached to it.
> ```
>
> 您必须先删除所有子组，然后才能删除该组。

您可以删除具有子事物的组，但由组中的成员资格授予事物的任何权限将不再适用。删除附加了策略的组之前，请认真检查，确保移除这些权限不会导致组中的事物无法正常运行。此外请注意，在更新云中的记录时，显示事物所属的组的命令（例如，ListGroupsForThing）可能会继续显示该组。

将策略附加到静态事物组

您可以使用 AttachPolicy 命令将策略附加到静态事物组，这样可以扩展到该组中的所有事物及其所有子组中的事物：

```
$ aws iot attach-policy \
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \
  --policy-name "myLightBulbPolicy"
```

AttachPolicy 命令不会生成任何输出

Important

您可以向组附加最多 2 个策略。

Note

我们建议不要在您的策略名称中使用个人身份信息。

`--target` 参数可以是事物组 ARN (如上所示)、证书 ARN 或 Amazon Cognito Identity。有关策略、证书和身份验证的更多信息，请参阅身份验证 (p. 264)。

有关更多信息，请参阅 Amazon IoT Core 策略。

从静态事物组分离策略

您可以使用 DetachPolicy 命令从组中分离策略，这样会更进一步地从该组中的所有事物及其所有子组中的事物分离：

```
$ aws iot detach-policy --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
 --policy-name "myLightBulbPolicy"
```

DetachPolicy 命令不会生成任何输出。

列出附加到静态事物组的策略

您可以使用 ListAttachedPolicies 命令列出附加到静态事物组的策略：

```
$ aws iot list-attached-policies --target "arn:aws:iot:us-west-2:123456789012:thinggroup/
RedLights"
```

`--target` 参数可以是事物组 ARN (如上所示)、证书 ARN 或 Amazon Cognito 身份。

添加可选的 `--recursive` 参数可以包括附加到组的父组的所有策略。

ListAttachedPolicies 命令返回策略列表：

```
{
    "policies": [
        "MyLightBulbPolicy"
        ...
    ]
}
```

列出策略的组

您可以使用 ListTargetsForPolicy 命令列出策略附加到的目标，包括任何组：

```
$ aws iot list-targets-for-policy --policy-name "MyLightBulbPolicy"
```

添加可选的 `--page-size` *`number`* 参数以指定为每个查询返回的最大结果数，并在后续调用上添加 `--marker` *`string`* 参数以检索下一组结果 (如果有)。

ListTargetsForPolicy 命令返回目标的列表以及用于检索更多结果的令牌：

```
{
```

```
    "nextMarker": "string",
    "targets": [ "string" ... ]
}
```

获取事物的有效策略

您可以使用 GetEffectivePolicies 命令列出对事物生效的策略，包括附加到事物所属任意组的策略 (不论该组是直接父组还是间接原级)：

```
$ aws iot get-effective-policies \
  --thing-name "MyLightBulb" \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

使用 `--principal` 参数指定附加到事物的证书的 ARN。如果您使用的是 Amazon Cognito 身份验证，请使用 `--cognito-identity-pool-id` 参数，并（可选）添加 `--principal` 参数来指定特定的 Amazon Cognito 身份。如果您仅指定 `--cognito-identity-pool-id`，则返回未经身份验证用户的与该身份池角色关联的策略。如果同时使用两个参数，则返回经过身份验证用户的与该身份池角色关联的策略。

`--thing-name` 参数是可选的，可用于替代 `--principal` 参数。在使用时，将返回附加到事物所属任意组的策略以及附加到这些组的任意父组（向上直至层次结构中的根组）的策略。

GetEffectivePolicies 命令返回策略列表：

```
{
    "effectivePolicies": [
        {
            "policyArn": "string",
            "policyDocument": "string",
            "policyName": "string"
        }
        ...
    ]
}
```

测试 MQTT 操作的授权

您可以使用 TestAuthorization 命令测试某个事物是否允许 MQTT 操作 (`Publish`、`Subscribe`)：

```
aws iot test-authorization \
    --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847" \
    --auth-infos "{\"actionType\": \"PUBLISH\", \"resources\": [ \"arn:aws:iot:us-
east-1:123456789012:topic/my/topic\"]}"
```

使用 `--principal` 参数指定附加到事物的证书的 ARN。如果使用 Amazon Cognito Identity 身份验证，请指定 Cognito Identity 作为 `--principal` 并且/或者使用 `--cognito-identity-pool-id` 参数。如果您仅指定 `--cognito-identity-pool-id`，则考虑未经身份验证用户的与该身份池角色关联的策略。如果同时使用两个参数，则考虑经过身份验证用户的与该身份池角色关联的策略。

通过列出资源和操作类型的组，并且后跟 `--auth-infos` 参数，来指定一个或多个要测试的 MQTT 操作。`actionType` 字段应包含“PUBLISH”、“SUBSCRIBE”、“RECEIVE”或“CONNECT”。`resources` 字段应包含资源 ARN 的列表。参阅 Amazon IoT Core 策略 (p. 299) 了解更多信息。

您可以通过将这些参数与 `--policy-names-to-add` 参数一起指定，测试添加策略的效果。或者，您可以通过 `--policy-names-to-skip` 参数来测试移除这些策略的效果。

您可以使用可选的 `--client-id` 参数来进一步优化结果。

TestAuthorization 命令返回您指定的每组 `--auth-infos` 查询允许或拒绝的操作的详细信息：

```
{
    "authResults": [
        {
            "allowed": {
                "policies": [
                    {
                        "policyArn": "string",
                        "policyName": "string"
                    }
                ]
            },
            "authDecision": "string",
            "authInfo": {
                "actionType": "string",
                "resources": [ "string" ]
            },
            "denied": {
                "explicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                },
                "implicitDeny": {
                    "policies": [
                        {
                            "policyArn": "string",
                            "policyName": "string"
                        }
                    ]
                }
            },
            "missingContextValues": [ "string" ]
        }
    ]
}
```

动态事物组

动态事物组通过搜索查询更新组成员资格。使用动态对象组，您可以根据对象的连接、注册表、影子或设备数据更改与对象的交互方式。由于动态事物组与实例集索引相关联，您必须启用实例集索引才能使用它们。您可以先预览动态事物组中的事物，然后再使用实例集索引搜索查询创建该组。有关更多信息，请参阅 机群索引 (p. 803) 和 查询语法 (p. 822)。

您可以将动态事物组指定为任务的目标。只有满足定义动态事物组的条件的事物才能执行任务。

例如，假设您要更新设备上的固件，但为将更新中断的几率降至最低，您只希望更新电池电量大于 80% 的设备上的固件。您可以创建仅包括报告的电池电量大于 80% 的设备的动态事物组，并且可以使用该动态事物组作为固件更新任务的目标。只有满足您的电池电量条件的设备才能收到固件更新。在设备达到 80% 的电池电量条件后，它们会添加到动态事物组并接收固件更新。

有关将事物组指定为任务目标的更多信息，请参阅 CreateJob。

动态事物组与静态事物组在以下方面不同：

- 未显式定义事物成员资格。要创建动态事物组，您必须定义一个查询字符串 (p. 823)，用于定义组成员资格。
- 动态事物组无法成为层次结构的一部分。
- 不能将策略应用于动态事物组。
- 您使用一组不同的命令来创建、更新和删除动态事物组。对于所有其它操作，用于与静态事物组交互的相同命令可用于与动态事物组交互。
- 单个账户可以拥有的动态组数量受到限制。

- 请勿在您的事物组名称中使用个人可识别信息。事物组名称可以出现在未加密的通信和报告中。

有关静态事物组的更多信息，请参阅静态事物组 (p. 246)。

例如，假设我们要创建一个动态组，其中包含仓库中温度高于 60 华氏度的所有房间。当房间温度为 61 度或更高时，它将被添加到 RoomTooWarm 动态事物组。RoomTooWarm 动态事物组中的所有房间都会打开冷却风扇。当一个房间的温度降至 60 度或更低时，将从动态事物组中移除该房间，并且其风扇将关闭。

创建动态事物组

使用 CreateDynamicThingGroup 命令创建动态事物组。要为房间太热的场景创建动态事物组，您可以使用 create-dynamic-thing-group CLI 命令：

```
$ aws iot create-dynamic-thing-group --thing-group-name "RoomTooWarm" --query-string
 "attributes.temperature>60"
```

> **Note**
>
> 我们建议不要在您的动态事物组名称中使用个人身份信息。

CreateDynamicThingGroup 命令返回包含索引名称、查询字符串、查询版本，事物组名称、事物组 ID 和事物组 Amazon Resource Name (ARN) 的响应：

```
{
    "indexName": "AWS_Things",
    "queryVersion": "2017-09-30",
    "thingGroupName": "RoomTooWarm",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RoomTooWarm",
    "queryString": "attributes.temperature>60\n",
    "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwx"
}
```

动态事物组创建不是即时发生的。动态事物组回填需要一些时间才能完成。在创建动态事物组时，该组的状态设置为 BUILDING。在回填完成后，状态变为 ACTIVE。要查看动态事物组的状态，请使用 DescribeThingGroup 命令。

描述动态事物组

使用 DescribeThingGroup 命令获取有关动态事物组的信息：

```
$ aws iot describe-thing-group --thing-group-name "RoomTooWarm"
```

DescribeThingGroup 命令返回有关指定组的信息：

```
{
```

```
    "status": "ACTIVE",
    "indexName": "AWS_Things",
    "thingGroupName": "RoomTooWarm",
    "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RoomTooWarm",
    "queryString": "attributes.temperature>60\n",
    "version": 1,
    "thingGroupMetadata": {
        "creationDate": 1548716921.289
    },
    "thingGroupProperties": {},
    "queryVersion": "2017-09-30",
    "thingGroupId": "84dd9b5b-2b98-4c65-84e4-be0e1ecf4fd8"
}
```

在动态事物组上运行 DescribeThingGroup 返回特定于动态事物组的属性，如 queryString 和状态。

动态事物组的状态可以采用以下值：

ACTIVE

动态事物组已准备就绪，可供使用。

BUILDING

正在创建动态事物组，并且正在处理事物成员资格。

REBUILDING

正在按照组的搜索查询的调整更新动态事物组的成员资格。

Note

在创建动态事物组后，您可以使用该组，而不必考虑其状态。只有 ACTIVE 状态的动态事物组包括与该动态事物组的搜索查询匹配的所有事物。BUILDING 和 REBUILDING 状态的动态事物组可能未包括与搜索查询匹配的所有事物。

更新动态事物组

使用 UpdateDynamicThingGroup 命令更新动态事物组的属性，包括组的搜索查询。以下命令更新事物组描述和查询字符串，同时将成员资格条件更改为温度 > 65：

```
$ aws iot update-dynamic-thing-group --thing-group-name "RoomTooWarm" --thing-group-
properties "thingGroupDescription=\"This thing group contains rooms warmer than 65F.\"" --
query-string "attributes.temperature>65"
```

UpdateDynamicThingGroup 命令返回一个响应，其中包含该组更新后的版本号：

```
{
    "version": 2
}
```

动态事物组更新不是即时发生的。动态事物组回填需要一些时间才能完成。在更新动态事物组时，该组的状态变为 REBUILDING，同时该组更新其成员资格。在回填完成后，状态变为 ACTIVE。要查看动态事物组的状态，请使用 DescribeThingGroup 命令。

删除动态事物组

使用 DeleteDynamicThingGroup 命令删除动态事物组：

```
$ aws iot delete-dynamic-thing-group --thing-group-name "RoomTooWarm"
```

DeleteDynamicThingGroup 命令不会生成任何输出。

在更新云中的记录时，显示事物所属的组的命令（例如，ListGroupsForThing）可能会继续显示该组。

限制和冲突

动态事物组与静态事物组共用这些限制：

- 事物组可以拥有的属性数量受到限制。
- 一个事物可以属于的组数受到限制。
- 无法重命名事物组。
- 事物组名称不能包含国际字符，如 û、é 和 ñ。

在使用动态事物组时，请记住以下内容。

必须已启用实例集索引服务

必须已启用实例集索引服务并且实例集索引回填必须已完成，然后您才能创建并使用动态事物组。启用实例集索引服务后，预计会有延迟。回填可能需要一些时间才能完成。您注册的事物越多，回填过程所需的时间就越长。在为动态事物组启用实例集索引服务后，您将无法禁用它，直到您删除所有动态事物组。

> **Note**
>
> 如果您有权查询实例集索引，可以访问整个实例集的事物数据。

动态事物组的数量受到限制

动态组的数量受到限制。

成功的命令可能记录错误

创建或更新动态事物组时，有些事物可能符合处于动态事物组中的条件，但尚未添加到动态事物组中。在这些情况下，尽管会记录错误和生成 AddThingToDynamicThingGroupsFailed 指标 (p. 405)，但创建或更新动态事物组的命令依旧会获得成功。

当无法将符合条件的事物添加到动态事物组或从动态事物组中移除事物以将其添加到另一个组时，会为每个事物创建 CloudWatch 日志中的错误日志条目。当事物无法添加到动态组时，也会创建一个 AddThingToDynamicThingGroupsFailed 指标 (p. 405)；但一个指标可以表示多个日志条目。

当某个事物有资格添加到动态事物组时，将考虑以下事项：

- 事物是否已经位于尽可能多的组中了？（请参阅限制）
  - 否：该事物被添加到动态事物组中。
  - 是：该事物是任何动态事物组的成员吗？
    - 否：无法将该事物添加到动态事物组，记录错误，并生成 AddThingToDynamicThingGroupsFailed 指标 (p. 405)。
    - 是：要加入的动态事物组是否早于该事物已成为其成员的任何动态事物组？
      - 否：无法将该事物添加到动态事物组，记录错误，并生成 AddThingToDynamicThingGroupsFailed 指标 (p. 405)。
      - 是：从该事物所属的最新的动态事物组中删除该事物，记录错误，然后将该事物添加到所需的动态事物组中。这会生成一个错误，并针对从中删除该事物的动态事物组生成一个 AddThingToDynamicThingGroupsFailed 指标 (p. 405)。

当动态事物组中的事物不再满足搜索查询时，将从动态事物组中删除它。同样，当某个事物已更新以满足动态事物组的搜索查询时，它会被添加到前面介绍的组中。这些添加和删除操作是正常的，不会产生错误记录条目。

在启用 overrideDynamicGroups 的情况下，静态组优先于动态组

一个事物可以属于的组数受到限制。当您使用 AddThingToThingGroup 或 UpdateThingGroupsForThing 命令更新事物成员资格时，添加 `--overrideDynamicGroups` 参数会使静态事物组优先于动态事物组。

将事物添加到静态事物组时，应考虑以下因素：

- 事物是否已经属于最大数量的组？
  - 否：该事物被添加到静态事物组中。
  - 是：该事物是否在任何动态组中？
    - 否：该事物无法添加到该事物组中。该命令引发异常。
    - 是：是否已启用 --overrideDynamicGroups？
      - 否：该事物无法添加到该事物组中。该命令引发异常。
      - 是：从最近创建的动态事物组中删除该事物，记录错误，并针对从中删除该事物的动态事物组生成 AddThingToDynamicThingGroupsFailed 指标 (p. 405)。然后，该事物被添加到静态事物组。

旧动态事物组优先于新动态事物组

一个事物可以属于的组数受到限制。当某个事物因创建或更新操作而有资格添加到动态事物组，而该事物已经在尽可能多的组中时，可以从另一个动态事物组中删除该事物以允许此添加。有关如何发生这种情况的更多信息，请参阅成功的命令可能记录错误 (p. 257)和在启用 overrideDynamicGroups 的情况下，静态组优先于动态组 (p. 258)了解示例。

当从动态事物组中删除事物时，将记录错误，并引发事件。

无法将策略应用于动态事物组

尝试将策略应用于动态事物组会生成异常。

动态事物组成员资格具有最终一致性

只为注册表评估事物的最终状态。如果状态快速更新，则可跳过中间状态。避免将规则、任务、 与其成员资格依赖中间状态的动态事物组相关联。

给您的 Amazon IoT 资源加标签

为了帮助您管理和组织事物组、事物类型、主题规则、任务、计划的审核和安全配置文件，您可以选择将自己的元数据以标签的形式分配给其中每个资源。本部分介绍标签并说明如何创建标签。

为了帮助您管理与事物相关的成本，您可以创建包含事物的账单组 (p. 261)。然后，可以将包含您的元数据的标签分配给其中每个账单组。本部分还讨论账单组以及可用于创建和管理它们的命令。

有关标签的基本知识

您可以使用标签按照不同方式（例如按用途、所有者或环境）对 Amazon IoT 资源进行分类。这在您具有相同类型的许多资源时会很有用 — 您可以根据分配给资源的标签快速识别资源。每个标签都包含您定义的一个键和一个可选值。例如，您可以为事物类型定义一组标签来帮助您按类型跟踪设备。我们建议您为每类资源创建一组可满足您的需求的标签键。使用一组连续的标签键，管理资源时会更加轻松。

您可以根据添加或应用的标签搜索和筛选资源。您还可以使用账单组标签对成本进行分类和跟踪。您还可以使用标签控制对资源的访问，如在 IAM 策略中使用标签 (p. 260)中所述。

为便于使用，Amazon管理控制台中的 Tag Editor 提供了一种用于创建和管理标签的集中而统一的方法。有关更多信息，请参阅使用Amazon管理控制台中的使用 Tag Editor。

您也可以使用 Amazon CLI 和 Amazon IoT API 处理标签。当您在以下命令中使用 Tags 字段创建标签时，可以将标签与事物组、事物类型、主题规则、任务、安全配置文件、策略和账单组相关联：

- CreateBillingGroup
- CreateDestination
- CreateDeviceProfile
- CreateDynamicThingGroup
- CreateJob.
- CreateOTAUpdate
- CreatePolicy
- CreateScheduledAudit
- CreateSecurityProfile
- CreateServiceProfile
- CreateStream
- CreateThingGroup
- CreateThingType
- CreateTopicRule
- CreateWirelessGateway
- CreateWirelessDevice

您可以使用以下命令为支持标记的现有资源添加、修改或删除标签：

- TagResource
- ListTagsForResource
- UntagResource

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

标签限制

下面是适用于标签的基本限制：

- 每个资源的最大标签数 - 50
- 最大密钥长度 - 127 个 Unicode 字符（采用 UTF-8 格式）
- 最大值长度 - 255 个 Unicode 字符（采用 UTF-8 格式）
- 标签键和值区分大小写。
- 请勿在标签名称或值中使用 `aws:` 前缀。它保留供 Amazon 使用。您无法编辑或删除带此前缀的标签名称或值。具有此前缀的标签不计入每个资源的标签数限制。
- 如果在多个服务和资源中使用您的标记方案，请记住，其它服务可能对允许使用的字符有限制。允许使用的字符包括：可用 UTF-8 格式表示的字母、空格和数字以及以下特殊字符：+ - = . _ : / @。

在 IAM 策略中使用标签

可以在用于 Amazon IoT API 操作的 IAM 策略中应用基于标签的资源级权限。这可让您更好地控制用户可创建、修改或使用哪些资源。在 IAM 策略中将 `Condition` 元素（也称作 `Condition` 块）与以下条件上下文键和值结合使用来基于资源标签控制用户访问（权限）：

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

> **Note**
>
> IAM 策略中的条件上下文键和值仅适用于能够标记的资源的标识符是必需参数的那些 Amazon IoT 操作。例如，将不允许或拒绝基于条件上下文键和值使用 DescribeEndpoint，因为此请求中引用了不可标记的资源（事物组、事物类型、主题规则、任务或安全配置文件）。有关可标记 Amazon IoT 资源及其支持的条件键的更多信息，请阅读 Amazon IoT 的操作、资源和条件键。

有关使用标签的更多信息，请参阅 Amazon Identity and Access Management 用户指南中的使用标签控制访问权限。该指南的 IAM JSON 策略参考一部分包含 IAM 中的 JSON 策略的元素、变量和评估逻辑的详细语法、描述和示例。

以下示例策略应用 `ThingGroup` 操作两个基于标签的限制。受此策略限制的 IAM 用户：

- 无法创建标签 “env=prod” 的事物组（在示例中，请参阅行 `"aws:RequestTag/env" : "prod"`）。
- 无法修改或访问具有现有标签“env=prod”的资源（在示例中，请参阅行 `"aws:ResourceTag/env" : "prod"`）。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "iot:CreateThingGroup",
```

```
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Deny",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/env": "prod"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:UpdateThingGroup"
      ],
      "Resource": "*"
    }
  ]
}
```

您还可以为给定标签键指定多个标签值，方法是将它们括在列表中，如下所示：

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

Note

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

账单组

Amazon IoT 不允许您直接将标签应用于单个事物，但允许您将事物放入账单组并将标签应用于这些账单组。对于 Amazon IoT，基于标签的成本分配和使用率数据局限于账单组。

Amazon IoT Core无法将适用于 LoRaWAN 的 资源（如无线设备和网关）添加到账单组中。但是，它们可以与 Amazon IoT 事物关联，这些内容可以添加到账单组中。

提供了以下命令：

- AddThingToBillingGroup 将事物添加到账单组。
- CreateBillingGroup 创建账单组。

- DeleteBillingGroup 删除账单组。
- DescribeBillingGroup 返回有关账单组的信息。
- ListBillingGroups 列出您已创建的账单组。
- ListThingsInBillingGroup 列出您已添加到给定账单组的事物。
- RemoveThingFromBillingGroup 从账单组删除给定事物。
- UpdateBillingGroup 从账单组删除给定事物。
- CreateThing 允许您在创建事物时为其指定账单组。
- DescribeThing 返回事物（包括事物所属的账单组，如果有）的描述。

Amazon IoT Wireless API 提供这些操作来将无线设备和网关与 Amazon IoT 事物关联。

- AssociateWirelessDeviceWithThing
- AssociateWirelessGatewayWithThing

查看成本分配和使用率数据

您可以使用账单组标签对成本进行分类和跟踪。当您将标签应用于账单组（从而应用于其包括的事物）时，Amazon将以逗号分隔值 (CSV) 文件格式生成一份成本分配报告，其中包括按标签汇总的使用率和成本。您可以设置代表业务类别（例如成本中心、应用程序名称或所有者）的标签，以便整理多种服务的成本。有关将标签用于成本分配的更多信息，请参阅《Amazon 账单和成本管理用户指南》中的使用成本分配标签。

> **Note**
>
> 要准确地将使用率和成本数据与您放入账单组的那些事物相关联，每个设备或应用程序必须：
>
> - 在 Amazon IoT 中注册为事物。有关更多信息，请参阅 使用 Amazon IoT 管理设备 (p. 239)。
> - 仅使用事物的名称作为客户端 ID，通过 MQTT 连接到 Amazon IoT 消息代理。有关更多信息，请参阅 the section called “设备通信协议” (p. 75)。
> - 使用与事物关联的客户端证书进行身份验证。

为账单组提供了以下定价维度（基于与账单组关联的事物的活动）：

- 连接（基于要连接的用作客户端 ID 的事物名称）。
- 消息收发（基于事物的入站和出站消息；仅限 MQTT）。
- 影子操作（基于其消息触发了影子更新的事物）。
- 触发的规则（基于其入站消息触发规则的事物；不适用于 MQTT 生命周期事件触发的那些规则）。
- 事物索引更新（基于添加到索引的事物）。
- 远程操作（基于已更新的事物）。
- Detect (p. 947)报告（基于报告其活动的事物）。

（为账单组报告的）基于标签的成本和使用率数据不会反映以下活动：

- 设备注册表操作（包括更新事物、事物组和事物类型）。有关更多信息，请参阅 使用 Amazon IoT 管理设备 (p. 239)）。
- 事物组索引更新（在添加事物组时）。
- 索引搜索查询。
- 设备预调配 (p. 772).
- 审核 (p. 888) 报告。

Amazon IoT 中的安全性

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

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

- 云的安全性 – Amazon负责保护在Amazon云中运行Amazon服务的基础设施。Amazon还向您提供可安全使用的服务。作为 Amazon 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于 Amazon IoT 的合规性计划，请参阅合规性计划范围内的 Amazon 服务。
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon IoT 时应用责任共担模式。以下主题说明如何配置 Amazon IoT 以实现您的安全性和合规性目标。您还会了解如何使用其它 Amazon 服务以帮助您监控和保护 Amazon IoT 资源。

主题

- Amazon IoT 安全性 (p. 263)
- 身份验证 (p. 264)
- Authorization (p. 298)
- Amazon IoT Core 中的数据保护 (p. 345)
- 适用于 Amazon IoT 的 Identity and Access Management (p. 348)
- 日志记录和监控 (p. 384)
- Amazon IoT Core 的合规性验证 (p. 385)
- Amazon IoT Core 中的故障恢复能力 (p. 385)
- 将 Amazon IoT Core 和接口 VPC 终端节点一起使用 (p. 385)
- Amazon IoT 中的基础设施安全性 (p. 388)
- 对生产实例集或带 Amazon IoT Core 的设备进行安全监控 (p. 388)
- Amazon IoT Core 中的安全最佳实践 (p. 388)

Amazon IoT 安全性

每个连接的设备或客户端都必须具有与 Amazon IoT 进行交互所需的凭证。进出 Amazon IoT 的所有流量通过传输层安全性 (TLS) 安全发送。数据在 Amazon IoT 和其它 Amazon 服务之间移动时，Amazon 云安全机制可为数据提供保护。

- 您负责管理 Amazon IoT 中的设备证书（X.509 证书、Amazon凭证、Amazon Cognito 身份、联合身份或自定义身份验证令牌）和策略。有关更多信息，请参阅Amazon IoT 中的密钥管理 (p. 348)：您负责将唯一身份分配给每台设备并管理每个设备或每组设备的权限。
- 您的设备使用 X.509 证书或 Amazon Cognito 身份通过安全 TLS 连接来与 Amazon IoT 建立连接。在研究和开发期间，对于一些进行 API 调用或使用的应用程序，您还可以使用 WebSockets IAM 用户和组或自定义身份验证令牌进行身份验证。有关更多信息，请参阅IAM 用户、组和角色 (p. 286)：
- 在使用 Amazon IoT 身份验证时，消息代理负责对设备进行身份验证、安全地接收设备数据以及授予或拒绝您使用 Amazon IoT 策略为设备指定的访问权限。
- 在使用自定义身份验证时，自定义授权方负责对您的设备进行身份验证，并授予或拒绝您使用 Amazon IoT 或 IAM policy 为您的设备指定的访问权限。
- Amazon IoT 规则引擎根据您定义的规则将设备数据转发到其他设备或其它 Amazon 服务。它使用 Amazon Identity and Access Management 将数据安全地传输到最终目的地。有关更多信息，请参阅适用于 Amazon IoT 的 Identity and Access Management (p. 348)：

身份验证

身份验证是用于验证客户端或服务器的身份的机制。服务器身份验证是设备或其它客户端确保它们与实际 Amazon IoT 终端节点进行通信的过程。客户端身份验证是设备或其它客户端使用 Amazon IoT 对自己进行身份验证的过程。

X.509 证书概览

X.509 证书是一个数字证书，它使用 X.509 公有密钥基础设施标准将公有密钥与证书中包含的身份相关联。X.509 证书由一家名为证书颁发机构 (CA) 的可信实体颁发。CA 持有一个或多个名为 CA 证书的特殊证书，它使用这种证书来颁发 X.509 证书。只有证书颁发机构才有权访问 CA 证书。X.509 证书链既用于客户端进行的服务器身份验证，又用于服务器进行的客户端身份验证。

服务器身份验证

当您的设备或其它客户端尝试连接 Amazon IoT Core 时，Amazon IoT Core 服务器将发送您的设备用来对服务器进行身份验证的 X.509 证书。身份验证通过 X.509 证书链 (p. 267)的验证在 TLS 层进行。这与浏览器在您访问 HTTPS URL 时使用的方法相同。如果要使用您自己的证书颁发机构提供的证书，请参阅管理 CA 证书 (p. 270)。

当您的设备或其他客户端与 Amazon IoT Core 终端节点建立 TLS 连接时，Amazon IoT Core 会显示一个证书链，设备可使用该证书链来确认它们正在与 Amazon IoT Core 进行通信，而不是与模拟 Amazon IoT Core

的其它服务器进行通信。显示的链取决于设备连接到的终端节点类型和客户端与 Amazon IoT Core 在 TLS 握手期间协商的密码套件 (p. 346)。

终端节点类型

Amazon IoT Core支持两种不同的数据端点类型，iot:Data和iot:Data-ATS。 iot:Data端点提供由3VeriSign 类公共主要 G5 根 CA 证书签名的证书。 iot:Data-ATS终端节点提供由亚马逊信任服务 CA 签名的服务器证书。

ATS 终端节点提供的证书由 Starfield 进行交叉签名。某些 TLS 客户端实现要求验证信任根，并要求将 Starfield CA 证书安装到客户端的信任存储中。

> Warning
>
> 建议不要使用对整个证书（包括颁发者名称等）进行哈希处理的证书固定方法，因为这将导致证书验证失败，因为我们提供的 ATS 证书由 Starfield 进行交叉签名并具有其它颁发者名称。

除非您的设备需要 Symantec 或 Verisign CA 证书，否则请使用 iot:Data-ATS 终端节点。Symantec 和 Verisign 证书已被弃用，并且大多数 Web 浏览器不再支持这两类证书。

您可以使用 describe-endpoint 命令创建 ATS 终端节点。

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

describe-endpoint 命令采用以下格式返回终端节点。

```
account-specific-prefix.iot.your-region.amazonaws.com
```

首次调用 describe-endpoint 时，会创建一个终端节点。对 describe-endpoint 的所有后续调用将返回同一终端节点。

为了实现向后兼容，Amazon IoT Core 仍支持 Symantec 终端节点。有关更多信息，请参阅 Amazon IoT Core 如何帮助客户应对即将发生的 Symantec 证书颁发机构不受信任情况。ATS 终端节点上运行的设备与同一账户中 Symantec 终端节点上运行的设备可完全互操作，并且不需要任何重新注册。

> Note
>
> 要在 Amazon IoT Core 控制台中查看 iot:Data-ATS 终端节点，请选择 Settings (设置)。控制台仅显示 iot:Data-ATS 终端节点。预设情况下，describe-endpoint 命令显示 iot:Data 终端节点以实现向后兼容。要查看 iot:Data-ATS 终端节点，请指定 --endpointType 参数，如上例所示。

使用 Amazon SDK for Java 创建 IotDataPlaneClient

预设情况下，Amazon SDK for Java - 版本 2 使用 IotDataPlaneClient 终端节点创建 iot:Data。要创建使用 iot:Data-ATS 终端节点的客户端，您必须执行以下操作。

- 使用 DescribeEndpointAPI 创建iot:Data-ATS终端节点。
- 在创建 IotDataPlaneClient 时指定该终端节点。

以下示例执行这两项操作。

```
public void setup() throws Exception {
        IotClient client =
 IotClient.builder().credentialsProvider(CREDENTIALS_PROVIDER_CHAIN).region(Region.US_EAST_1).build();
```

```
        String endpoint = client.describeEndpoint(r -> r.endpointType("iot:Data-
ATS")).endpointAddress();
        iot = IotDataPlaneClient.builder()
                                .credentialsProvider(CREDENTIALS_PROVIDER_CHAIN)
                                .endpointOverride(URI.create("https://" + endpoint))
                                .region(Region.US_EAST_1)
                                .build();
}
```

用于服务器身份验证的 CA 证书

根据您使用的数据终端节点的类型以及您协商的密码套件，Amazon IoT Core 服务器身份验证证书由以下根 CA 证书之一进行签名：

VeriSign 终端节点（旧版）

- RSA 2048 位密钥：3VeriSign 类公共主要 G5 根 CA 证书

Amazon Trust Services 终端节点（首选）

> **Note**
>
> 您可能需要右键单击这些链接，然后选择 Save link as... (将链接另存为...) 将这些证书另存为文件。

- RSA 2048 位密钥：Amazon Root CA 1。
- RSA 4096 位密钥：Amazon Root CA 2。留待将来使用。
- ECC 256 位密钥：Amazon Root CA 3。
- ECC 384 位密钥：Amazon Root CA 4。留待将来使用。

这些证书都由 Starfield 根 CA 证书进行交叉签名。从 2018 年 5 月 9 日在亚太（孟买）区域中推出 Amazon IoT Core 开始，所有新的 Amazon IoT Core 区域都将仅处理 ATS 证书。

服务器身份验证指南

有很多变量会影响设备验证 Amazon IoT Core 服务器身份验证证书的能力。例如，设备的内存可能太有限，无法容纳所有可能的根 CA 证书，或者设备可能会实施非标准的证书验证方法。由于这些原因，我们建议遵循以下准则：

- 我们建议您使用 ATS 终端节点并安装所有受支持的 Amazon Root CA 证书。
- 如果您无法在设备上存储所有这些证书，并且您的设备未使用基于 ECC 的验证，则可以省略 Amazon Root CA 3 和 Amazon Root CA 4 ECC 证书。如果您的设备未实施基于 RSA 的证书验证，则可以省略 Amazon Root CA 1 和 Amazon Root CA 2 RSA 证书。您可能需要右键单击这些链接，然后选择 Save link as... (将链接另存为...) 将这些证书另存为文件。
- 如果在连接到 ATS 终端节点时遇到服务器证书验证问题，请尝试将相关的交叉签名的 Amazon Root CA 证书添加到您的信任存储中。您可能需要右键单击这些链接，然后选择 Save link as... (将链接另存为...) 将这些证书另存为文件。
  - 交叉签名 Amazon Root CA 1
  - 交叉签名的 Amazon Root CA 2 - 预留供将来使用。
  - 交叉签名 Amazon Root CA 3
  - 交叉签名的 Amazon Root CA 4 - 预留供将来使用。
- 如果您遇到服务器证书验证问题，则您的设备可能需要明确信任根 CA。尝试将 Starfield Root CA Certificate 添加到您的信任存储。
- 如果您在执行上述步骤后仍遇到问题，请联系Amazon开发人员支持。

Note

CA 证书具有一个过期日期，在该日期后，这些证书将无法用于验证服务器的证书。可能必须在 CA 证书的过期日期前替换这些证书。请确保可以更新所有设备或客户端上的根 CA 证书，以确保持续的连接并保持最新的安全最佳实践。

Note

在设备代码中连接到 Amazon IoT Core 时，请将证书传递到您用于连接的 API 中。您使用的 API 因 SDK 而异。有关更多信息，请参阅 Amazon IoT Core Device SDK (p. 1283)。

客户端身份验证

Amazon IoT 支持三种类型的身份委托人执行设备或客户端身份验证：

- X.509 客户端证书 (p. 267)
- IAM 用户、组和角色 (p. 286)
- Amazon Cognito 身份 (p. 287)

这些身份可与设备、移动、Web 或桌面应用程序结合使用。用户甚至可以通过键入 Amazon IoT 命令行界面 (CLI) 命令来使用它们。通常，Amazon IoT 设备使用 X.509 证书，移动应用程序使用 Amazon Cognito 身份。Web 应用程序和桌面应用程序使用 IAM 或联合身份。Amazon CLI 命令使用 IAM。有关 IAM 身份的更多信息，请参阅 适用于 Amazon IoT 的 Identity and Access Management (p. 348)。

X.509 客户端证书

X.509 证书向 Amazon IoT 提供对客户端和设备连接进行身份验证的功能。客户端证书必须先注册到 Amazon IoT，然后客户端才能与 Amazon IoT 进行通信。可以在同一Amazon Web Services 区域的多个 Amazon Web Services 账户中注册客户端证书，以便在同一区域的Amazon Web Services 账户之间移动设备。参阅 通过多账户注册在多个Amazon Web Services 账户中使用 X.509 客户端证书 (p. 268) 了解更多信息。

我们建议为每个设备或客户端提供一个唯一的证书，以便进行精细的客户端管理操作，包括证书吊销。设备和客户端还必须支持证书轮换和更换，以帮助确保在证书过期时平稳运行。

有关使用 X.509 证书支持多个设备的信息，请参阅 设备预调配 (p. 772) 以查看 Amazon IoT 支持的不同证书管理和预调配选项。

Amazon IoT 支持以下类型的 X.509 客户端证书：

- Amazon IoT 生成的 X.509 证书
- 由注册到 Amazon IoT 的 CA 签发的 X.509 证书。
- 由未注册到 Amazon IoT 的 CA 签发的 X.509 证书。

本节介绍如何在 Amazon IoT 中管理 X.509 证书。您可以使用 Amazon IoT 控制台或 Amazon CLI 执行以下证书操作：

- 创建 Amazon IoT 客户端证书 (p. 269)
- 创建您自己的客户端证书 (p. 270)
- 注册客户端证书 (p. 277)
- 激活或停用客户端证书 (p. 281)
- 吊销客户端证书 (p. 283)

有关执行这些操作的 Amazon CLI 命令的更多信息，请参阅 Amazon IoT CLI 参考。

使用 X.509 客户端证书

X.509 证书对连接到 Amazon IoT 的客户端和设备进行身份验证。与其它身份和身份验证机制相比，X.509 证书具有多项优势。有了 X.509 证书，可以将非对称密钥用于设备。例如，您可以将私有密钥刻录到设备上的安全存储中，这样敏感的加密材料永远不会离开设备。X.509 证书可以通过用户名和密码或持有者令牌等其它方案提供更可靠的客户端身份验证，因为私有密钥永远不会离开设备。

Amazon IoT 使用 TLS 协议的客户端身份验证模式来验证客户端证书。TLS 支持适用于多种编程语言和操作系统并且通常用于为数据加密。在 TLS 客户端身份验证中，Amazon IoT 请求 X.509 客户端证书并依照证书注册表验证证书的状态和Amazon Web Services 账户。随后，将要求客户端提供与证书所含公有密钥相对应的私有密钥的所有权证明。Amazon IoT 要求客户端将服务器名称指示 (SNI) 扩展发送到传输层安全性 (TLS) 协议。有关配置 SNI 扩展的更多信息，请参阅 Amazon IoT 中的传输安全 (p. 346)。

X.509 证书可以根据可信证书颁发机构 (CA) 进行验证。您可以创建使用 Amazon Root CA 的客户端证书，也可以使用由其它 CA 签发的自己的客户端证书。有关使用您自己的 X.509 证书的更多信息，请参阅创建您自己的客户端证书 (p. 270)。

对于 CA 证书签发的证书过期日期和时间，将在创建 CA 证书时设置。Amazon IoT 生成的 X.509 证书在 UTC 2049 年 12 月 31 日(2049-12-31T23:59:59Z) 到期。有关使用 Amazon IoT 控制台创建使用 Amazon Root CA 的证书的更多信息，请参阅创建 Amazon IoT 客户端证书 (p. 269)。

通过多账户注册在多个Amazon Web Services 账户中使用 X.509 客户端证书

多账户注册能够在同一区域或不同区域的 Amazon Web Services 账户 之间移动设备。您可以在预生产账户中注册、测试和配置设备，然后在生产账户中注册并使用相同的设备和设备证书。您还可以在设备上注册客户端证书，或在没有已向 Amazon IoT 注册的 CA 的情况下注册设备证书。有关更多信息，请参阅 Register a client certificate signed by an unregistered CA (CLI) (p. 279)（注册由未注册的 CA 签发的客户端证书（CLI））。

> Note
>
> 用于多账户注册的证书在 `iot:Data-ATS`、`iot:Data`（旧式）、`iot:Jobs` 和 `iot:CredentialProvider` 终端节点类型中获得支持。有关 Amazon IoT 设备终端节点的更多信息，请参阅 Amazon IoT 设备数据和服务终端节点 (p. 72)。

使用多账户注册的设备必须将服务器名称指示 (SNI) 扩展发送到传输层安全性 (TLS) 协议，并在其连接到 `host_name` 时在 Amazon IoT 字段提供完整端点地址。Amazon IoT 会使用 `host_name` 中的端点地址将连接路由到正确的 Amazon IoT 账户。未发送 `host_name` 中的有效终端节点地址的现有设备将继续工作，但它们将无法使用需要此信息的功能。有关 SNI 扩展以及如何识别 `host_name` 字段的终端节点地址的更多信息，请参阅Amazon IoT 中的传输安全 (p. 346)。

使用多账户注册

1. 您可以向 CA 注册设备证书。您可以在 `SNI_ONLY` 模式下在多个账户中注册签名 CA，并使用该 CA 向多个账户注册相同的客户端证书。有关更多信息，请参阅在 SNI_ONLY 模式下注册 CA 证书（CLI）(p. 273)：
2. 您可以在没有 CA 的情况下注册设备证书。请参阅注册由未注册的 CA (CLI) 签发的客户端证书 (p. 279)。注册 CA 是可选的。您不需要向 Amazon IoT 注册已对设备证书进行签名的 CA。

Amazon IoT 支持的证书签名算法

Amazon IoT 支持以下证书签名算法：

- SHA256WITHRSA
- SHA384WITHRSA
- SHA512WITHRSA
- DSA_WITH_SHA256

- ECDSA-WITH-SHA256
- ECDSA-WITH-SHA384
- ECDSA-WITH-SHA512

有关证书身份验证和安全性的更多信息，请参阅设备证书密钥质量 (p. 891)。

Note

证书签名请求（CSR）必须包含一个公有密钥，该密钥可以是长度至少为 2048 位的 RSA 密钥，也可以是来自 NIST P-256 或 NIST P-384 曲线的 ECC 密钥。有关更多信息，请参阅 CreateCertificateFromCsrAPI 参考指南中的Amazon IoT API。

创建 Amazon IoT 客户端证书

Amazon IoT 提供由 Amazon Root 证书颁发机构 (CA) 签发的客户端证书。

本主题介绍如何创建由 Amazon Root 证书颁发机构签发的客户端证书，以及如何下载证书文件。创建客户端证书文件后，您必须在客户端上安装这些文件。

Note

Amazon IoT 提供的每个 X.509 客户端证书都保存了在证书创建时设置的发布者和主题属性。只有在创建证书后，证书属性才是不可改变的。

您可以使用 Amazon IoT 控制台或 Amazon CLI 创建由 Amazon Root 证书颁发机构签发的 Amazon IoT 证书。

创建 Amazon IoT 证书（控制台）

使用 Amazon IoT 控制台创建 Amazon IoT 证书

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧的导航窗格中，选择安全性，选择证书，然后选择创建。
3. 选择一键式创建证书 (建议)，然后选择创建证书。
4. 从证书已创建！页面，将事物、公有密钥和私有密钥的客户端证书文件下载到安全位置。Amazon IoT 生成的这些证书仅在与 Amazon IoT 服务配合使用时可用。

   如果您还需要 Amazon Root CA 证书文件，此页面也包含指向可用于下载文件的页面链接。
5. 此时客户端证书已创建并注册到 Amazon IoT。在客户端中使用证书之前，您必须激活证书。

   选择 Activate（激活）以立即激活客户端证书。如果您不想立即激活证书，激活客户端证书（控制台）(p. 281) 中介绍了如何在以后激活证书。
6. 如果要将策略附加到证书，请选择 Attach a policy（附加策略）。

   如果您不希望立即附加策略，请选择 Done（完成）以完成操作。您可以在以后附加策略。

完成此流程后，在客户端上安装证书文件。

创建 Amazon IoT 证书 (CLI)

Amazon CLI 提供 create-keys-and-certificate 命令，用于创建由 Amazon Root 证书颁发机构签名的客户端证书。但是，此命令不会下载 Amazon Root CA 证书文件。您可以从用于服务器身份验证的 CA 证书 (p. 266)下载 Amazon Root CA 证书文件。

此命令创建私有密钥、公有密钥和 X.509 证书文件，将证书注册到 Amazon IoT 并激活。

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

如果您不想在创建和注册证书时激活证书，则此命令会创建私有密钥、公有密钥和 X.509 证书文件并注册证书，但不会激活它。激活客户端证书 (CLI) (p. 281) 介绍了以后如何激活证书。

```
aws iot create-keys-and-certificate \
    --no-set-as-active \
    --certificate-pem-outfile certificate_filename.pem \
    --public-key-outfile public_filename.key \
    --private-key-outfile private_filename.key
```

在客户端上安装证书文件。

创建您自己的客户端证书

Amazon IoT 支持由任何根或中间证书颁发机构（CA）签署的客户端证书。Amazon IoT 使用 CA 证书来验证证书的所有权。要使用由非 Amazon CA 的 CA 签署的设备证书，此 CA 的证书必须向 Amazon IoT 注册，以便我们可以验证设备证书的所有权。

Amazon IoT 支持多种自带证书（BYOC）的方式：

- 首先，注册用于签署客户端证书的 CA，然后注册各个客户端证书。如果要在设备或客户端首次连接到 Amazon IoT 时将设备或客户端注册到其客户端证书（也称为即时预调配），您必须向 Amazon IoT 注册签名 CA 并激活自动注册。
- 如果您无法注册签名 CA，则可以选择在没有 CA 的情况下注册客户端证书。对于未使用 CA 注册的设备，当您将它们连接到 Amazon IoT 时，您需要出示服务器名称指示（SNI）。

> **Note**
>
> 要使用 CA 注册客户端证书，必须将签名 CA（而不是层次结构中的任何其他 CA）注册到 Amazon IoT。

> **Note**
>
> 在 DEFAULT 模式下，一个 CA 证书只能由一个区域中的一个账户注册。在 SNI_ONLY 模式下，一个 CA 证书可以由一个区域中的多个账户注册。

有关使用 X.509 证书支持多个设备的更多信息，请参阅 设备预调配 (p. 772) 以查看 Amazon IoT 支持的不同证书管理和预调配选项。

主题

- 管理 CA 证书 (p. 270)
- 使用您的 CA 证书创建客户端证书 (p. 276)

管理 CA 证书

本节介绍管理您自己的证书颁发机构 (CA) 证书的常见任务。

如果您使用由 Amazon IoT 无法识别的证书颁发机构 (CA) 签发的客户端证书，则需要将您的 CA 注册到 Amazon IoT。

如果您希望客户端在首次连接时自动将其客户端证书注册到 Amazon IoT，则必须将签发该客户端证书的 CA 注册到 Amazon IoT。否则，您不需要注册对客户端证书签发的 CA 证书。

Note

在 DEFAULT 模式下，一个 CA 证书只能由一个区域中的一个账户注册。在 SNI_ONLY 模式下，一个 CA 证书可以由一个区域中的多个账户注册。

- 创建 CA 证书 (p. 271)，如果您需要一个。

  创建下一步骤所需的证书和密钥文件。
- 注册 CA 证书 (p. 271)

  使用 Amazon IoT 注册 CA 证书
- 停用 CA 证书 (p. 275)

创建 CA 证书

如果没有 CA 证书，则可以使用 OpenSSL v1.1.1i 工具创建一个。

Note

您无法在 Amazon IoT 控制台中执行此流程。

使用 OpenSSL v1.1.1i 工具创建 CA 证书

1. 生成密钥对。

```
openssl genrsa -out root_CA_key_filename.key 2048
```

2. 使用密钥对中的私有密钥生成 CA 证书。

```
openssl req -x509 -new -nodes \
    -key root_CA_key_filename.key \
    -sha256 -days 1024 \
    -out root_CA_cert_filename.pem
```

注册 CA 证书

这些程序说明如何注册来自 Amazon CA 以外的证书颁发机构（CA）的证书。 Amazon IoT Core使用 CA 证书来验证证书的所有权。要使用由非 Amazon CA 的 CA 签署的设备证书，您必须向注册 CA 证书，Amazon IoT Core以便设备证书的所有权。

注册 CA 证书（控制台）

Note

若要在控制台中注册 CA 证书，请在控制台中的 Register CA certificate（注册 CA 证书）处开始注册。您可以在多账户模式下注册您的 CA，无需提供验证证书或访问私钥。在多账户模式下，可以在同一个账户Amazon Web Services 账户中多个 CA 注册Amazon Web Services 区域。您可以通过提供验证证书和 CA 私钥所有权证明，在单账户模式下注册您的 CA。可以在多账户模式下一合一Amazon Web Services 账户注册 CAAmazon Web Services 区域。

注册 CA 证书 (CLI)

您可以在 DEFAULT 模式或 SNI_ONLY 模式下注册 CA 证书。CA 可以在DEFAULT模式下一合Amazon Web Services 账户一注册Amazon Web Services 区域。一个 CA 可以在同一个SNI_ONLY模式下由多

Amazon Web Services 账户个 CA 注册Amazon Web Services 区域。有关 CA 证书模式的更多信息，请参阅 certificateMode。

在 DEFAULT 模式下注册 CA 证书（CLI）

先决条件

继续操作之前，请确保电脑满足以下条件：

- 根 CA 的证书文件（在以下示例中引用为*root_CA_cert_filename.pem*）
- 根 CA 证书的私有密钥文件（在以下示例中引用为*root_CA_key_filename.key*）
- OpenSSL v1.1.1i 或更高版本

在 **DEFAULT** 模式下使用 Amazon CLI 注册 CA 证书

1. 要从中获取注册代码Amazon IoT，请使用get-registration-code。保存返回的 `registrationCode`，将其用作私有密钥验证证书的 `Common Name`。有关更多信息，请参阅《Amazon CLI 命令参考》中的 get-registration-code。

   ```
   aws iot get-registration-code
   ```

2. 为私有密钥验证证书生成密钥对：

   ```
   openssl genrsa -out verification_cert_key_filename.key 2048
   ```

3. 为私有密钥验证证书创建证书签名请求 (CSR)。将证书的 `Common Name` 字段设置为 get-registration-code 返回的 `registrationCode`。

   ```
   openssl req -new \
       -key verification_cert_key_filename.key \
       -out verification_cert_csr_filename.csr
   ```

   将提示您输入一些信息，包括证书的`Common Name`。

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [AU]:
       State or Province Name (full name) []:
       Locality Name (for example, city) []:
       Organization Name (for example, company) []:
       Organizational Unit Name (for example, section) []:
       Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
       Email Address []:

       Please enter the following 'extra' attributes
       to be sent with your certificate request
       A challenge password []:
       An optional company name []:
   ```

4. 使用 CSR 创建私有密钥验证证书：

   ```
   openssl x509 -req \
       -in verification_cert_csr_filename.csr \
   ```

```
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out verification_cert_filename.pem \
    -days 500 -sha256
```

5. 在 Amazon IoT 中注册 CA 证书。将 CA 证书文件名和私有密钥验证证书文件名传递给 register-ca-certificate 命令，如下所示：有关更多信息，请参阅《Amazon CLI 命令参考》中的 register-ca-certificate。

```
aws iot register-ca-certificate \
    --ca-certificate file://root_CA_cert_filename.pem \
    --verification-cert file://verification_cert_filename.pem
```

   如果成功，此命令将返回 *certificateId*。

6. 此时，CA 证书已向 Amazon IoT 注册，但未激活。CA 证书必须处于活跃状态，然后您才能注册由其签发的任何客户端证书。

   此步骤将激活 CA 证书。

   要激活 CA 证书，请使用 update-certificate 命令，如下所示。有关更多信息，请参阅 Amazon CLI 命令参考中的 update-certificate。

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

要查看 CA 证书的状态，请使用 describe-ca-certificate 命令。有关更多信息，请参阅《Amazon CLI 命令参考》中的 describe-ca-certificate。

在 SNI_ONLY 模式下注册 CA 证书（CLI）

先决条件

继续操作之前，请确保电脑满足以下条件：

- 根 CA 的证书文件（在以下示例中引用为*root_CA_cert_filename.pem*）
- OpenSSL v1.1.1i 或更高版本

在 **SNI_ONLY** 模式下使用 Amazon CLI 注册 CA 证书

1. 在 Amazon IoT 中注册 CA 证书。使用register-ca-certificate命令输入 CA 证书文件名。有关更多信息，请参阅《Amazon CLI 命令参考》中的 register-ca-certificate。

```
aws iot register-ca-certificate \
    --ca-certificate file://root_CA_cert_filename.pem \
    --certificate-mode SNI_ONLY
```

   如果成功，该命令将返回certific *ateId*。

2. 此时，CA 证书已向注册，Amazon IoT但处于非活动状态。CA 证书必须处于活跃状态，然后您才能注册由其签发的任何客户端证书。

   此步骤将激活 CA 证书。

   要激活 CA 证书，请使用 update-certificate 命令，如下所示。有关更多信息，请参阅 Amazon CLI 命令参考中的 update-certificate。

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

要查看 CA 证书的状态，请使用 describe-ca-certificate 命令。有关更多信息，请参阅《Amazon CLI 命令参考》中的 describe-ca-certificate。

创建 CA 验证证书以在控制台中注册 CA 证书

Note

此过程仅适用于从 Amazon IoT 控制台注册 CA 证书。
如果未从 Amazon IoT 控制台执行此过程，请在控制台中的 Register CA certificate（注册 CA 证书）处启动 CA 证书注册过程。

继续操作之前，请确保同一台电脑满足以下条件：

- 根 CA 的证书文件（在以下示例中引用为*root_CA_cert_filename.pem*）
- 根 CA 证书的私有密钥文件（在以下示例中引用为*root_CA_key_filename.key*）
- OpenSSL v1.1.1i 或更高版本

使用命令行界面创建 CA 验证证书，以在控制台中注册 CA 证书

1. *verification_cert_key_filename.key*替换为要创建的验证证书密钥文件的名称（例如**verification_cert.key**）。然后运行此命令，为私有密钥验证证书生成key pair：

   ```
   openssl genrsa -out verification_cert_key_filename.key 2048
   ```

2. *verification_cert_key_filename.key*替换为在步骤 1 中创建的密钥文件的名称。

   将 *verification_cert_csr_filename.csr* 替换为要创建的证书签名请求 (CSR) 文件的名称。例如，**verification_cert.csr**。

   运行此命令以创建 CSR 文件。

   ```
   openssl req -new \
       -key verification_cert_key_filename.key \
       -out verification_cert_csr_filename.csr
   ```

   该命令会提示您输入其他信息（稍后说明）。
3. 在 Amazon IoT 控制台的 Verification certificate（验证证书）容器中，复制注册码。
4. openssl 命令提示您输入的信息显示在以下示例中。除此Common Name字段外，您可以输入自己的值或将其留空。

   将您在上一步骤中复制的注册码粘贴到该Common Name字段中。

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [AU]:
       State or Province Name (full name) []:
   ```

```
    Locality Name (for example, city) []:
    Organization Name (for example, company) []:
    Organizational Unit Name (for example, section) []:
    Common Name (e.g. server FQDN or YOUR name) []:your_registration_code
    Email Address []:

    Please enter the following 'extra' attributes
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

完成后，命令会创建 CSR 文件。

5. 将 *verification_cert_csr_filename.csr* 替换为在上一步骤中使用的 *verification_cert_csr_filename.csr*。

   *root_CA_cert_filename.pem*替换为要注册的 CA 证书的文件名。

   *root_CA_key_filename.key*替换为 CA 证书的私有密钥文件的文件名。

   *verification_cert_filename.pem*替换为要创建的验证证书的文件名。例如，**verification_cert.pem**。

```
openssl x509 -req \
    -in verification_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out verification_cert_filename.pem \
    -days 500 -sha256
```

6. OpenSSL 命令完成后，您应该准备好这些文件，以便在返回控制台时使用。

   - 您的 CA 证书文件（上一条命令中使用的 *root_CA_cert_filename.pem*）
   - 您在上一步骤中创建的验证证书（上一条命令中使用的 *verification_cert_filename.pem*）

停用 CA 证书

启用证书颁发机构 (CA) 证书以自动注册客户端证书时，Amazon IoT检查 CA 证书以确保 CA 是ACTIVE。如果 CA 证书状态为 INACTIVE，Amazon IoT 不允许注册客户端证书。

通过将 CA 证书设置为INACTIVE，可以防止 CA 颁发的任何新客户证书自动注册。

> **Note**
>
> 除非您明确吊销由受损的 CA 证书签发的每个已注册客户端证书，否则所有此类证书均将继续工作。

停用 CA 证书（控制台）

使用 Amazon IoT 控制台停用 CA 证书

1. 登录 Amazon Web Services Management Console并打开 Amazon IoT 控制台。
2. 在左侧的导航窗格中，依次选择安全、CA。
3. 在证书颁发机构列表中，找到要停用的证书颁发机构，然后选择省略号图标打开选项菜单。
4. 在选项菜单上，选择停用。

证书颁发机构应在列表中显示为停用。

Note

Amazon IoT 控制台没有提供方法来列出由您停用的 CA 签发的证书。有关列出这些证书的 Amazon CLI 选项，请参阅停用 CA 证书 (CLI) (p. 276)。

停用 CA 证书 (CLI)

Amazon CLI 提供 update-ca-certificate 命令用于停用 CA 证书。

```
aws iot update-ca-certificate \
    --certificate-id certificateId \
    --new-status INACTIVE
```

使用 list-certificates-by-ca 命令获取已由指定 CA 签名的所有已注册客户端证书的列表。对于由指定 CA 证书签名的每个客户端证书，请使用 update-certificate 命令吊销该客户端证书以避免使用它。

使用 describe-ca-certificate 命令查看 CA 证书的状态。

使用您的 CA 证书创建客户端证书

您可以使用自己的证书颁发机构 (CA) 创建客户端证书。必须先在 Amazon IoT 中注册客户端证书，然后才能使用。有关客户端证书的注册选项的信息，请参阅注册客户端证书 (p. 277)。

创建客户端证书 (CLI)

Note

您无法在 Amazon IoT 控制台中执行此流程。

使用 Amazon CLI 创建客户端证书

1. 生成密钥对。

   ```
   openssl genrsa -out device_cert_key_filename.key 2048
   ```

2. 为客户端证书创建 CSR。

   ```
   openssl req -new \
       -key device_cert_key_filename.key \
       -out device_cert_csr_filename.csr
   ```

   系统将提示您输入一些信息，如下所示：

   ```
   You are about to be asked to enter information that will be incorporated
   into your certificate request.
   What you are about to enter is what is called a Distinguished Name or a DN.
   There are quite a few fields but you can leave some blank
   For some fields there will be a default value,
   If you enter '.', the field will be left blank.
   -----
   Country Name (2 letter code) [AU]:
       State or Province Name (full name) []:
       Locality Name (for example, city) []:
       Organization Name (for example, company) []:
       Organizational Unit Name (for example, section) []:
       Common Name (e.g. server FQDN or YOUR name) []:
       Email Address []:

       Please enter the following 'extra' attributes
   ```

```
    to be sent with your certificate request
    A challenge password []:
    An optional company name []:
```

3. 从 CSR 创建客户端证书。

```
openssl x509 -req \
    -in device_cert_csr_filename.csr \
    -CA root_CA_cert_filename.pem \
    -CAkey root_CA_key_filename.key \
    -CAcreateserial \
    -out device_cert_filename.pem \
    -days 500 -sha256
```

此时已创建客户端证书，但尚未注册到 Amazon IoT。有关注册客户端证书的方式和时机的信息，请参阅注册客户端证书 (p. 277)。

注册客户端证书

客户端证书必须注册到 Amazon IoT 才能启用客户端与 Amazon IoT 之间的通信。您可以手动注册每个客户端证书，也可以将客户端证书配置为在客户端首次连接到 Amazon IoT 时自动注册。

如果您希望客户端和设备在首次连接时注册其客户端证书，则必须在要使用的区域中拥有使用 Amazon IoT 签署客户端证书所需的 注册 CA 证书 (p. 271)。Amazon Root CA 自动注册到 Amazon IoT。

客户端证书可按Amazon Web Services 账户和区域共享。在您要使用客户端证书的每个账户和区域中，必须执行这些主题中的流程。在一个账户或区域中注册的客户端证书，不会在另一个账户或区域中自动识别。

> Note
>
> 使用传输层安全性 (TLS) 协议连接到 Amazon IoT 的客户端必须支持 TLS 的服务器名称指示 (SNI) 扩展。有关更多信息，请参阅Amazon IoT 中的传输安全 (p. 346)：

主题

- 手动注册客户端证书 (p. 277)
- 在客户端连接到注册 (JITR) 时Amazon IoT just-in-time 注册客户端证书 (p. 279)

手动注册客户端证书

您可以使用 Amazon IoT 控制台和 Amazon CLI 手动注册客户端证书。

要使用的注册流程取决于证书是否将由 Amazon Web Services 账户和区域共享。在一个账户或区域中注册的客户端证书，不会在另一个账户或区域中自动识别。

在您要使用客户端证书的每个账户和区域中，必须执行此主题中的流程。客户端证书可以按Amazon Web Services 账户和区域共享，但仅限由未注册到 Amazon IoT 的证书颁发机构 (CA) 签发的客户端证书。

注册由已注册的 CA（控制台）签发的客户端证书

> Note
>
> 在执行此流程之前，请确保您具有客户端证书的 .pem 文件，并且客户端证书由您注册到 Amazon IoT (p. 271) 的 CA 签发。

使用控制台注册将现有证书注册到 Amazon IoT

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。

2. 在导航窗格的 Manage（管理）部分下，选择 Security（安全性），然后选择 Certificates（证书）。
3. 在 Certificates（证书）对话框的 Certificates（证书）页上，选择 Add certificate（添加证书），然后选择 Register certificates（注册证书）。
4. 在 Certificates to upload（要上载的证书）对话框的 Register certificate（注册证书）页上，执行以下操作：
   - 选择 CA is registered with Amazon IoT（CA 注册到 Amazon IoT）。
   - 从 Choose a CA certificate（选择 CA 证书）中，选择您的 Certification authority（证书颁发机构）。
     - 选择 Register a new CA（注册新 CA）以注册未向 Amazon IoT 注册的新 Certification authority（证书颁发机构）。
     - 如果 Amazon Root certificate authority（Amazon 根证书颁发机构）是您的证书颁发机构，请将 Choose a CA certificate（选择 CA 证书）留空。
   - 选择最多 10 个证书进行上载并向 Amazon IoT 注册。
     - 使用您在创建 Amazon IoT 客户端证书 (p. 269)和使用您的 CA 证书创建客户端证书 (p. 276)中创建的证书文件。
   - 选择 Activate（激活）或 Deactivate（停用）。如果选择 Deactive（停用），激活或停用客户端证书 (p. 281)介绍了如何在证书注册后激活证书。
   - 选择 Regster（注册）。

在 Certificates（证书）对话框的 Certificates（证书）页上，现在将显示您注册的证书。

注册由未注册的 CA（控制台）签发的客户端证书

Note

在执行此流程之前，请确保您拥有客户端证书 .pem 文件。

使用控制台注册将现有证书注册到 Amazon IoT

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧的导航窗格中，选择安全，选择证书，然后选择创建。
3. 在 Create a certificate（创建证书）上，找到 Use my certificate（使用我的证书）条目，然后选择 Get started（入门）。
4. 在 Select CA（选择 CA）上，选择 Next（下一步）。
5. 在 Register exsiting device certificates（注册现有的设备证书）上，选择 Select certificates（选择证书），并选择最多 10 个要注册的证书文件。
6. 关闭文件对话框后，选择在注册客户端证书时是要激活还是吊销客户端证书。

   如果您在注册证书时未激活证书，激活客户端证书（控制台） (p. 281) 中介绍了以后如何激活证书。

   如果证书在注册时被吊销，则以后无法激活。

   选择要注册的证书文件，选择注册后要执行的操作，然后选择 Register certificates（注册证书）。

成功注册的客户端证书将显示在证书列表中。

注册由已注册的 CA (CLI) 签发的客户端证书

Note

在执行此流程之前，请确保您具有证书颁发机构 (CA) .pem 和客户端证书的 .pem 文件。客户端证书必须由您已注册到 Amazon IoT (p. 271) 的证书颁发机构 (CA) 签发。

使用 register-certificate 命令注册客户端证书但不激活。

```
aws iot register-certificate \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

客户端证书已注册到 Amazon IoT，但尚未激活。有关如何在以后激活的信息，请参阅 激活客户端证书 (CLI) (p. 281)。

您也可以在使用此命令注册客户端证书时进行激活。

```
aws iot register-certificate \
    --set-as-active \
    --certificate-pem file://device_cert_filename.pem \
    --ca-certificate-pem file://ca_cert_filename.pem
```

有关激活证书以使其可用于连接到 Amazon IoT 的更多信息，请参阅激活或停用客户端证书 (p. 281)

注册由未注册的 CA (CLI) 签发的客户端证书

Note

在执行此流程之前，请确保您拥有证书 .pem 文件。

使用 register-certificate-without-ca 命令注册客户端证书但不激活。

```
aws iot register-certificate-without-ca \
    --certificate-pem file://device_cert_filename.pem
```

客户端证书已注册到 Amazon IoT，但尚未激活。有关如何在以后激活的信息，请参阅 激活客户端证书 (CLI) (p. 281)。

您也可以在使用此命令注册客户端证书时进行激活。

```
aws iot register-certificate-without-ca \
    --status ACTIVE \
    --certificate-pem file://device_cert_filename.pem
```

有关激活证书以使其可用于连接到 Amazon IoT 的更多信息，请参阅 激活或停用客户端证书 (p. 281)。

在客户端连接到注册 (JITR) 时Amazon IoT just-in-time 注册客户端证书

您可以配置 CA 证书，以便在客户端首次连接到 Amazon IoT 时，允许使用它签发的客户端证书自动注册到 Amazon IoT。

要在客户端首次连接到 Amazon IoT 时注册客户端证书，您必须启用 CA 证书进行自动注册，并配置客户端在首次连接时提供所需的证书。

配置 CA 证书以支持自动注册（控制台）

使用 Amazon IoT 控制台配置 CA 证书以支持自动客户端证书注册

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧的导航窗格中，依次选择安全、CA。
3. 在证书颁发机构列表中，找到要启用自动注册的颁发机构，然后使用省略号图标打开选项菜单。
4. 在选项菜单上，选择启用自动注册。

Note

证书颁发机构列表中不显示自动注册状态。要查看证书颁发机构的自动注册状态，您必须打开证书颁发机构的详细信息页面。

配置 CA 证书以支持自动注册 (CLI)

如果您已将 CA 证书注册到 Amazon IoT，请使用 update-ca-certificate 命令将 CA 证书的 `autoRegistrationStatus` 设置为 `ENABLE`。

```
aws iot update-ca-certificate \
--certificate-id caCertificateId \
--new-auto-registration-status ENABLE
```

如果您要在注册 CA 证书时启用 `autoRegistrationStatus`，请使用 register-ca-certificate 命令。

```
aws iot register-ca-certificate \
--allow-auto-registration  \
--ca-certificate file://root_CA_cert_filename.pem \
--verification-cert file://verification_cert_filename.pem
```

使用 describe-ca-certificate 命令查看 CA 证书的状态。

配置客户端的首次连接以进行自动注册

当客户端首次尝试连接到 Amazon IoT 时，作为 TLS 握手的一部分，必须提供一个文件，其中包含由 CA 证书签发的客户端证书。

在客户端连接到 Amazon IoT 时，将 *client_certificate_filename* 文件用作证书文件。Amazon IoT 会将 CA 证书识别为已注册的 CA 证书，注册客户端证书并将其状态设置为 `PENDING_ACTIVATION`。这意味着，已自动注册客户端证书，该证书正在等待激活。客户端证书的状态必须为 `ACTIVE`，然后才能将其用于连接到 Amazon IoT。

Note

您可以使用Amazon IoT Core just-in-time 注册 (JITR) 功能预调配设备，而无需将设备首次连接上的整个信任链发送到Amazon IoT Core。可以出示 CA 证书，但连接时设备需要发送服务器名称指示 (SNI)扩展。

当 Amazon IoT 自动注册证书时或者当客户端提供处于 `PENDING_ACTIVATION` 状态的证书时，Amazon IoT 将向以下 MQTT 主题发布消息：

`$aws/events/certificates/registered/`*caCertificateId*

其中 *caCertificateId* 是颁发客户端证书的 CA 证书的 ID。

发布到该主题的消息具有以下结构：

```
{
        "certificateId": "certificateId",
        "caCertificateId": "caCertificateId",
        "timestamp": timestamp,
        "certificateStatus": "PENDING_ACTIVATION",
        "awsAccountId": "awsAccountId",
        "certificateRegistrationTimestamp": "certificateRegistrationTimestamp"
}
```

您可以创建一项规则，以侦听此主题并执行一些操作。我们建议您创建一项 Lambda 规则，以验证客户端证书不在证书吊销列表 (CRL) 中，激活该证书，创建策略并将其附加到证书中。该策略将确定客户端能够访问

的资源。有关如何创建可侦听$aws/events/certificates/registered/*caCertificateID*主题并执行这些操作的 Lambda 规则的更多信息，请参阅上的客户端证书just-in-time 注册Amazon IoT。

如果在自动注册客户端证书期间发生任何错误或异常，将向您的 Logs 日志Amazon IoT发送相关事件或消息。 CloudWatch 有关如何为您的账户设置日志的更多信息，请参阅亚马逊 CloudWatch 文档。

激活或停用客户端证书

Amazon IoT 在对连接进行身份验证时，验证客户端证书是否处于活动状态。

您可以创建和注册客户端证书但不激活，这样直到您想要使用时才让这些证书可用。您还可以停用活动的客户端证书以暂时禁用它们。最后，您可以吊销客户端证书，防止以后使用这些证书。

激活客户端证书（控制台）

使用 Amazon IoT 控制台激活客户端证书

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择安全，然后选择证书。
3. 在证书列表中，找到要激活的证书，然后使用省略号图标打开选项菜单。
4. 在选项菜单中，选择激活。

证书应在证书列表中显示为活动。

停用客户端证书（控制台）

使用 Amazon IoT 控制台停用客户端证书

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择安全，然后选择证书。
3. 在证书列表中，找到要停用的证书，然后使用省略号图标打开选项菜单。
4. 在选项菜单中，选择停用。

证书应在证书列表中显示为停用。

激活客户端证书 (CLI)

Amazon CLI 提供 update-certificate 命令用于激活证书。

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status ACTIVE
```

如果命令成功，证书的状态将为 ACTIVE。运行 describe-certificate 以查看证书的状态。

```
aws iot describe-certificate \
    --certificate-id certificateId
```

停用客户端证书 (CLI)

Amazon CLI 提供 update-certificate 命令用于停用证书。

```
aws iot update-certificate \
```

```
    --certificate-id certificateId \
    --new-status INACTIVE
```

如果命令成功，证书的状态将为 INACTIVE。运行 describe-certificate 以查看证书的状态。

```
aws iot describe-certificate \
    --certificate-id certificateId
```

将事物或策略附加到客户端证书

当您独立于 Amazon IoT 事物创建和注册证书时，它没有授权任何 Amazon IoT 操作的任何策略，也不会与任何 Amazon IoT 事物对象相关联。本节介绍如何将这些关系添加到注册的证书中。

> **Important**
>
> 要完成这些流程，您必须具有要附加到证书的已创建事物或策略。

该证书通过 Amazon IoT 对设备进行身份验证，以便它可以连接。将证书附加到事物资源可建立设备（通过证书的方式）与事物资源之间的关系。授权设备执行 Amazon IoT 操作（如允许设备连接和发布消息），则必须将适当的策略附加到设备的证书。

将事物附加到客户端证书（控制台）

您需要事物对象的名称才能完成此流程。

将事物对象附加到已注册的证书

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择安全，然后选择证书。
3. 在证书列表中，找到要附加策略的证书，选择省略号图标以打开证书的选项菜单，然后选择附加事物。
4. 在弹出窗口中，找到要附加到证书的事物的名称，选中其复选框，然后选择 Attach（附上）。

现在，事物对象应显示在证书详细信息页面上的事物列表中。

将策略附加到客户端证书（控制台）

您需要策略对象的名称才能完成此流程。

将策略对象附加到已注册的证书

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择安全，然后选择证书。
3. 在证书列表中，找到要附加策略的证书，选择省略号图标以打开证书的选项菜单，然后选择附加策略。
4. 在弹出窗口中，找到要附加到证书的策略的名称，选中其复选框，然后选择 Attach（附上）。

现在，策略对象应显示在证书详细信息页面上的策略列表中。

将事物附加到客户端证书 (CLI)

Amazon CLI 提供了 attach-thing-principal 命令用于将事物对象附加到证书。

```
aws iot attach-thing-principal \
    --principal certificateArn \
```

```
    --thing-name thingName
```

将策略附加到客户端证书 (CLI)

Amazon CLI 提供了 attach-policy 命令用于将策略对象附加到证书。

```
aws iot attach-policy \
    --target certificateArn \
    --policy-name policyName
```

吊销客户端证书

如果您在已注册的客户端证书上检测到可疑活动，则可以吊销该证书，使其无法再次使用。

吊销客户端证书（控制台）

使用 Amazon IoT 控制台吊销客户端证书

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择安全，然后选择证书。
3. 在证书列表中，找到要吊销的证书，然后使用省略号图标打开选项菜单。
4. 在选项菜单中，选择吊销。

如果证书已成功吊销，它将在证书列表中显示为已吊销。

吊销客户端证书 (CLI)

Amazon CLI 提供 update-certificate 命令用于吊销证书。

```
aws iot update-certificate \
    --certificate-id certificateId \
    --new-status REVOKED
```

如果命令成功，证书的状态将为 REVOKED。运行 describe-certificate 以查看证书的状态。

```
aws iot describe-certificate \
    --certificate-id certificateId
```

将设备证书传输到其它账户。

属于一个Amazon Web Services 账户的 X.509 证书可以传输到另一个Amazon Web Services 账户。

若要将 X.509 证书从一个Amazon Web Services 账户传输到另一个账户

1. the section called “开始证书传输” (p. 284)

   在启动传输之前，必须停用证书并从所有策略和事物中分离。
2. the section called “接受或拒绝证书传输” (p. 285)

   接收账户必须明确接受或拒绝传输的证书。接收账户户接受证书后，必须激活证书才能使用。
3. the section called “取消证书传输” (p. 286)

   如果证书未被接受，原始账户可以取消传输。

开始证书传输

您可以使用 Amazon IoT 控制台或 Amazon CLI 开始将证书传输到另一个Amazon Web Services 账户。

开始证书传输（控制台）

要完成此流程，您需要提供您要传输的证书 ID。

在带有要传输证书的账户执行此流程。

要开始将证书传输到另一个Amazon Web Services 账户。

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择安全，然后选择证书。

   选择您打算传输并带有 Active（活动）或者 Inactive（非活动）状态的证书并打开其详细信息页面。
3. 在证书 Details（详细信息）页面的 Actions（操作）菜单中，如果 Deactivate（停用）选项可用，则选择 Deactivate（停用）选项来停用证书。
4. 在证书的 Details（详细信息）页面中，在左侧菜单中，选择 Policies（策略）。
5. 在证书的 Policies（策略）页面上，如果证书附加了任何策略，请打开策略的选项菜单，然后选择 Detach（分离）以分离各个证书。

   在您继续操作前，该证书不能附加有任何策略。
6. 在证书的 Policies（策略）页面中，在左侧菜单中，选择 Thins（事物）。
7. 在证书的 Things（事物）页面，如果证书附加了任何事物，请打开事物的选项菜单并选择 Detach（分离）以分离各个事物。

   在您继续操作前，该证书不能附加有任何事物。
8. 在证书的 Things（事物）页面中，在左侧菜单中，选择 Details（详细信息）。
9. 在证书 Details（详细信息）页面的 Actions（操作）菜单中，选择 Start transfer（开始传输）来打开 Start transfer（开始传输）对话框。
10. 在 Start transfer（开始传输）对话框中，输入接收证书的Amazon Web Services 账户账号和可选的短消息。
11. 选择 Start transfer（开始传输）以传输证书。

控制台应显示一条消息，指示传输成功或失败。如果传输开始，证书的状态将更新为 Transferred（已传输）。

开始证书传输 (CLI)

要完成此流程，您需要使用待传输证书的 *certificateId* 和 *certificateArn*。

在带有要传输证书的账户执行此流程。

要开始将证书传输到另一个Amazon账户

1. 使用 update-certificate 命令来停用证书。

   ```
   aws iot update-certificate --certificate-id certificateId --new-status INACTIVE
   ```
2. 分离所有策略。

   1. 使用 list-attached-policies 命令列出附加到该证书的策略。

```
aws iot list-attached-policies --target certificateArn
```

2. 对于每个附加的策略，请使用 detach-policy 命令予以分离。

```
aws iot detach-policy --target certificateArn --policy-name policy-name
```

3. 分离所有事物。

1. 使用 list-principal-things 命令列示附加到证书上的事物。

```
aws iot list-principal-things --principal certificateArn
```

2. 对于每个附加的事物，请使用 detach-thing-principal 命令予以分离。

```
aws iot detach-thing-principal --principal certificateArn --thing-name thing-name
```

4. 使用 transfer-certificate 命令启动证书传输。

```
aws iot transfer-certificate --certificate-id certificateId --target-aws-
account account-id
```

接受或拒绝证书传输

您可以使用 Amazon IoT 控制台或 Amazon CLI 接受或拒绝证书从另一个Amazon Web Services 账户传输到您的Amazon Web Services 账户。

接受或拒绝证书传输（控制台）

要完成此流程，您需要提供已转移到您的账户的证书 ID。

在接收已转移证书的账户执行此流程。

接受或拒绝转移到您Amazon Web Services 账户的证书

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择安全，然后选择证书。

   选择您想要接受或拒绝且状态为 Pending transfer（待转移）的证书，并打开其详细信息页面。
3. 在证书的 Details（详细信息）页面上，在 Actions（操作）菜单中，

   - 如要接受证书，请选择 Accept transfer（接受传输）。
   - 如不接受证书，请选择 Reject transfer（拒绝传输）。

接受或拒绝证书传输 (CLI)

要完成此流程，您需要想要接受或拒绝的证书传输的 *certificateId*。

在接收已转移证书的账户执行此流程。

接受或拒绝转移到您Amazon Web Services 账户的证书

1. 使用 accept-certificate-transfer 命令以接受证书。

```
aws iot accept-certificate-transfer --certificate-id certificateId
```

2. 使用 reject-certificate-transfer 命令以拒绝证书。

```
aws iot reject-certificate-transfer --certificate-id certificateId
```

取消证书传输

您可以使用 Amazon IoT 控制台或 Amazon CLI 在证书传输被接受之前将其取消。

取消证书传输（控制台）

要完成此流程，您需要提供要取消的证书传输的 ID。

在启动证书传输的账户执行此流程。

取消证书传输

1. 登录 Amazon 管理控制台，打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择安全，然后选择证书。

   选择状态为 Transferred（已传输）且您想要取消传输的证书，并打开其选项菜单。
3. 在证书的选项菜单上，选择 Revoke transfer（撤销传输）选项来取消证书传输。

   > **Important**
   >
   > 请谨慎处理，不要将 Revoke transfer（撤销传输）选项与 Revoke（撤销）选项混淆。Revoke transfer（撤销传输）选项用于取消证书传输，而 Revoke（撤销）选项使证书无法为 Amazon IoT 所用，且操作不可逆。

取消证书传输 (CLI)

要完成此流程，您需要想要取消的证书传输的 *certificateId*。

在启动证书传输的账户执行此流程。

使用 cancel-certificate-transfer 命令取消证书传输。

```
aws iot cancel-certificate-transfer --certificate-id certificateId
```

IAM 用户、组和角色

IAM 用户、组和角色是用于在Amazon中管理身份和身份验证的标准机制。您可以使用它们连接到使用 Amazon SDK 和 Amazon CLI 的 Amazon IoT HTTP 界面。

IAM 角色还允许 Amazon IoT 代表您访问您账户中的其它 Amazon 资源。例如，如果您希望设备将自身状态发布到 DynamoDB 表，则 IAM 角色允许 Amazon IoT 与 Amazon DynamoDB 交互。有关更多信息，请参阅 IAM 角色。

对于通过 HTTP 的消息代理连接，将使用 SignAmazon IoT ature Version 4 签名流程对用户、组和角色执行身份验证。有关信息，请参阅签署 Amazon API 请求。

将 Amazon Signature Version 4 与 Amazon IoT 配合使用时，客户端必须支持在 TLS 实施中使用以下各项：

- TLS 1.2、TLS 1.1、TLS 1.0
- SHA-256 RSA 证书签名验证

- 来自 TLS 密码套件支持部分的密码套件之一

有关信息，请参阅 适用于 Amazon IoT 的 Identity and Access Management (p. 348)。

Amazon Cognito 身份

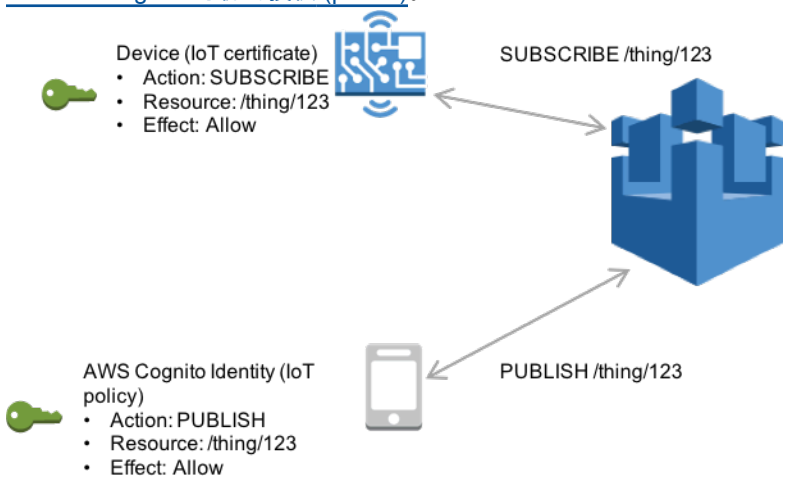
Amazon Cognito Identity 使您能够创建临时的、有限特权的Amazon凭证，以便在移动和 Web 应用程序中使用。在使用 Amazon Cognito Identity 时，您会创建身份池，这些身份池将为您的用户创建唯一身份并通过 Login with Amazon、Facebook 和 Google 这样的身份提供商对其进行身份验证。您还可以将 Amazon Cognito 身份与您自己的经开发人员验证的身份结合使用。有关更多信息，请参阅 Amazon Cognito Identity。

要使用 Amazon Cognito Identity，您需要定义与 IAM 角色相关联的 Amazon Cognito 身份池。IAM 角色与一个 IAM policy 相关联，该策略向您的身份池中的身份授予访问Amazon资源（如调用的Amazon服务）的权限。

Amazon Cognito Identity 可创建未经身份验证的身份和经过身份验证的身份。未经身份验证的身份用于希望在不登录的情况下使用该应用程序的移动或 Web 应用程序中的访客用户。未经身份验证的用户仅授予与身份池关联的 IAM policy 中指定的权限。

当您使用经过身份验证的身份时，除了附加到身份池的 IAM 策略外，您还必须使用 AttachPolicyAPI 将 Amazon IoT策略附加到 Amazon Cognito Identity，并向Amazon IoT应用程序的单个用户提供权限。您可以使用Amazon IoT策略以为特定客户及其设备分配精细权限。

经过身份验证和未经身份验证的用户是不同的身份类型。如果您没有将 Amazon IoT 策略附加到 Amazon Cognito Identity，则经过身份验证的用户无法在 Amazon IoT 中获得授权并无权访问 Amazon IoT 资源和操作。有关为 Amazon Cognito 身份创建策略的更多信息，请参阅 发布/订阅策略示例 (p. 316) 和 使用 Amazon Cognito 身份的授权 (p. 337)。



自定义身份验证

Amazon IoT Core 允许您定义自定义授权方，以便您管理自己的客户端身份验证和授权。当您需要使用除 Amazon IoT Core 本地支持的身份验证机制以外的机制时，这将很有帮助。（有关本地支持机制的更多信息，请参阅 the section called “客户端身份验证” (p. 267)）。

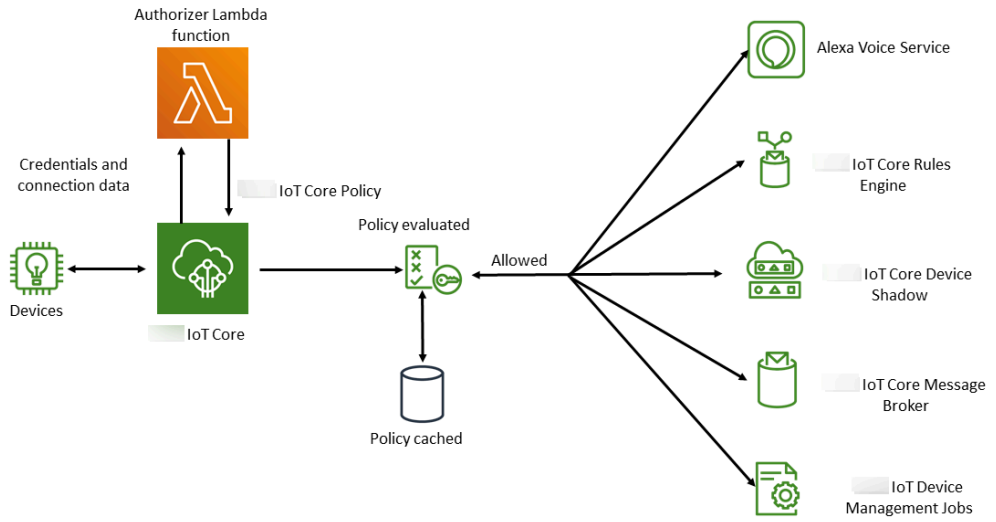
例如，如果要将现场中的现有设备迁移到 Amazon IoT Core，并且这些设备使用自定义承载令牌或 MQTT 用户名和密码进行身份验证，您可以将它们迁移到 Amazon IoT Core，而不必为其预调配新的身份。您可以将自定义身份验证与 Amazon IoT Core 支持的任何通信协议搭配使用。有关受 Amazon IoT Core 支持的协议的更多信息，请参阅 the section called “设备通信协议” (p. 75)。

主题

- 了解自定义身份验证工作流 (p. 288)
- 创建和管理自定义授权方 (p. 289)
- 使用自定义身份验证连接到 Amazon IoT Core (p. 294)
- 授权方故障排除 (p. 296)

了解自定义身份验证工作流

自定义身份验证使您能够定义如何通过使用授权方资源对客户端进行身份验证和授权。 每个授权方都包含对客户托管的 Lambda 函数的引用、用于验证设备凭证的可选公有密钥以及其它配置信息。下图说明了 Amazon IoT Core 中自定义身份验证的授权工作流。



Amazon IoT Core 自定义身份验证和授权工作流

下面的列表说明了自定义身份验证和授权工作流中的每个步骤。

1. 设备使用支持的 the section called “设备通信协议” (p. 75) 之一连接到客户的 Amazon IoT Core 数据终端节点。设备在请求的标头字段或查询参数（适用于 HTTP Publish 或通过 WebSockets 协议的 MQTT）或 MQTT CONNECT 消息的用户名和密码字段中传递凭证（适用于 MQTT 和 MQTT 基于 WebSockets协议的 MQTT）。
2. Amazon IoT Core 检查以下两个条件之一：
   - 传入的请求指定授权方。
   - 接收请求的 Amazon IoT Core 数据终端节点具有已为其配置的默认授权方。

   如果 Amazon IoT Core 以上任一方式查找授权方，Amazon IoT Core 将触发与授权方关联的 Lambda 函数。
3. （可选）如果您已启用令牌签名，Amazon IoT Core 在触发 Lambda 函数之前，将通过使用存储在授权方中的公有密钥来验证请求签名。如果验证失败，Amazon IoT Core 将停止请求而不调用 Lambda 函数。
4. Lambda 函数接收请求中的凭证和连接元数据，并做出身份验证决策。
5. Lambda 函数返回身份验证决策的结果以及 Amazon IoT Core 策略文档，该文档指定允许在连接中执行哪些操作。Lambda 函数还通过调用 Lambda 函数返回指定 Amazon IoT Core 重新验证请求中凭证的频率。

6. Amazon IoT Core 根据从 Lambda 函数接收的策略评估连接上的活动。

扩展注意事项

由于 Lambda 函数为您的授权方处理身份验证和授权，因此该函数受 Lambda 定价和服务限制的约束，例如并发执行率。有关 Lambda 定价的更多信息，请参阅 Lambda 定价。您可以通过调整 Lambda 函数响应中的 `refreshAfterInSeconds` 和 `disconnectAfterInSeconds` 参数管理您 Lambda 函数上的负载。有关 Lambda 函数响应内容的更多信息，请参阅 the section called “定义您的 Lambda 函数” (p. 289)。

> Note
>
> 如果启用签名，则可以防止无法识别的客户端过度触发 Lambda。在禁用授权方的签名之前，请考虑这一点。

> Note
>
> 自定义授权方的 Lambda 函数超时限制为 5 秒。

创建和管理自定义授权方

Amazon IoT Core 使用授权方资源实现自定义身份验证和授权方案。各授权方均包括以下组件：

- 名称：用户定义的唯一字符串，用于标识授权方。
- Lambda 函数 ARN：Lambda 函数的 Amazon Resource Name (ARN)，用于实现授权和身份验证逻辑。
- 令牌密钥名称：用于从 HTTP 标头、查询参数或 MQTT CONNECT 用户名中提取令牌以执行签名验证的键名称。如果在授权方中启用了签名，则需要此值。
- 签名禁用标志（可选）：指定是否禁用凭证签名要求的布尔值。这对于签名凭证没有意义的情况非常有用，例如使用 MQTT 用户名和密码的身份验证方案。默认值为 `false`，因此预设情况下签名将处于启用状态。
- 令牌签名公有密钥：Amazon IoT Core 用于验证令牌签名的公有密钥。其最小长度为 2048 位。如果在授权方中启用了签名，则需要此值。

Lambda 将根据 Lambda 函数的运行次数以及函数中代码执行所需的时间向您收取费用。有关 Lambda 定价的更多信息，请参阅 Lambda 定价。有关创建 Lambda 函数的更多信息，请参阅 Lambda 开发人员指南。

> Note
>
> 如果启用签名，则可以防止无法识别的客户端过度触发 Lambda。在禁用授权方的签名之前，请考虑这一点。

> Note
>
> 自定义授权方的 Lambda 函数超时限制为 5 秒。

定义您的 Lambda 函数

Amazon IoT Core 调用您的授权方时，它会触发与授权方关联的 Lambda 以及包含以下 JSON 对象的事件。示例 JSON 对象包含所有可能的字段。不包括与连接请求无关的任何字段。

```
{
    "token" :"aToken",
    "signatureVerified": Boolean, // Indicates whether the device gateway has validated the
 signature.
    "protocols": ["tls", "http", "mqtt"], // Indicates which protocols to expect for the
 request.
```

```
    "protocolData": {
        "tls" : {
            "serverName": "serverName" // The server name indication (SNI) host_name
 string.
        },
        "http": {
            "headers": {
                "#{name}": "#{value}"
            },
            "queryString": "?#{name}=#{value}"
        },
        "mqtt": {
            "username": "myUserName",
            "password": "myPassword", // A base64-encoded string.
            "clientId": "myClientId" // Included in the event only when the device sends
 the value.
        }
    },
    "connectionMetadata": {
        "id": UUID // The connection ID. You can use this for logging.
    },
}
```

Lambda 函数应使用此信息对传入连接进行身份验证，并决定允许在连接中执行哪些操作。函数应发送包含以下值的响应。

- `isAuthenticated`：一个布尔值，指示是否已对请求进行身份验证。
- `principalId`：字母数字字符串，它充当自定义授权请求发送的令牌的标识符。该值必须是包含至少一个字符且不超过 128 个字符的字母数字字符串，并与此正则表达式（正则表达式）模式匹配：`([a-zA-Z0-9]){1,128}`。
- `policyDocuments`：JSON 格式的 Amazon IoT Core 策略文档的列表。有关创建 Amazon IoT Core 策略的更多信息，请参阅 the section called “Amazon IoT Core 策略” (p. 299)。策略文档的数量最多为 10 个。每个策略文档最多可以包含 2048 个字符。
- `disconnectAfterInSeconds`：指定与 Amazon IoT Core 网关的连接的最大持续时间（以秒为单位）的整数。最小值为 300 秒，最大值为 86400 秒。
- `refreshAfterInSeconds`：指定策略刷新之间间隔的整数。当此时间间隔过去时，Amazon IoT Core 将调用 Lambda 函数以允许策略刷新。最小值为 300 秒，最大值为 86400 秒。

以下 JSON 对象包含 Lambda 函数可以发送的响应示例。

```
{
"isAuthenticated":true, //A Boolean that determines whether client can connect.
"principalId": "xxxxxxxx",  //A string that identifies the connection in logs.
"disconnectAfterInSeconds": 86400,
"refreshAfterInSeconds": 300,
 "policyDocuments": [
      {
        "Version": "2012-10-17",
        "Statement": [
           {
              "Action": "iot:Publish",
              "Effect": "Allow",
              "Resource": "arn:aws:iot:us-east-1:<your_aws_account_id>:topic/
customauthtesting"
            }
         ]
       }
    ]
}
```

policyDocument 值必须包含一个有效的 Amazon IoT Core 策略文档。有关Amazon IoT Core策略的更多信息，请参阅the section called “Amazon IoT Core 策略” (p. 299)。 在通过 TLS 的 MQTT 和通过 WebSockets连接的Amazon IoT Core MQTT 中，将根据refreshAfterInSeconds字段值中指定的间隔缓存此策略。在 HTTP 连接的情况下，每个授权请求都会调用 Lambda 函数，除非您的设备使用 HTTP 持久连接（也称为 HTTP 保持活动状态或 HTTP 连接重用），否则您可以在配置授权方时选择启用缓存。在此间隔期间,Amazon IoT Core 针对此缓存策略授权已建立的连接中的操作，而不会再次触发 Lambda 函数。如果自定义身份验证过程中出现故障，则 Amazon IoT Core 将终止连接。如果连接打开的时间超过 disconnectAfterInSeconds 参数中指定的值，则 Amazon IoT Core 也会终止连接。

以下内容 JavaScript 包含一个示例 Node.js Lambda 函数，该函数在值为的 MQTT Connect 消息中查找密码，test并返回授予权限Amazon IoT Core以连接到名为的客户端myClientName和发布到包含相同客户端名称的主题。如果未找到预期密码，则返回拒绝这两个操作的策略。

```
// A simple Lambda function for an authorizer. It demonstrates
// how to parse an MQTT password and generate a response.

exports.handler = function(event, context, callback) {
    var uname = event.protocolData.mqtt.username;
    var pwd = event.protocolData.mqtt.password;
    var buff = new Buffer(pwd, 'base64');
    var passwd = buff.toString('ascii');
    switch (passwd) {
        case 'test':
            callback(null, generateAuthResponse(passwd, 'Allow'));
        default:
            callback(null, generateAuthResponse(passwd, 'Deny'));
    }
};

// Helper function to generate the authorization response.
var generateAuthResponse = function(token, effect) {
    var authResponse = {};
    authResponse.isAuthenticated = true;
    authResponse.principalId = 'TEST123';

    var policyDocument = {};
    policyDocument.Version = '2012-10-17';
    policyDocument.Statement = [];
    var publishStatement = {};
    var connectStatement = {};
    connectStatement.Action = ["iot:Connect"];
    connectStatement.Effect = effect;
    connectStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:client/myClientName"];
    publishStatement.Action = ["iot:Publish"];
    publishStatement.Effect = effect;
    publishStatement.Resource = ["arn:aws:iot:us-east-1:123456789012:topic/telemetry/
myClientName"];
    policyDocument.Statement[0] = connectStatement;
    policyDocument.Statement[1] = publishStatement;
    authResponse.policyDocuments = [policyDocument];
    authResponse.disconnectAfterInSeconds = 3600;
    authResponse.refreshAfterInSeconds = 300;

    return authResponse;
}
```

上述 Lambda 函数在收到 MQTT Connect 消息中的预期密码 test 时返回以下 JSON。password 和 principalId 属性的值将是来自 MQTT Connect 消息的值。

```
{
```

```
  "password": "password",
  "isAuthenticated": true,
  "principalId": "principalId",
  "policyDocuments": [
    {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": "iot:Connect",
          "Effect": "Allow",
          "Resource": "*"
        },
        {
          "Action": "iot:Publish",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        },
        {
          "Action": "iot:Subscribe",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topicfilter/telemetry/${iot:ClientId}"
        },
        {
          "Action": "iot:Receive",
          "Effect": "Allow",
          "Resource": "arn:aws:iot:region:accountId:topic/telemetry/${iot:ClientId}"
        }
      ]
    }
  ],
  "disconnectAfterInSeconds": 3600,
  "refreshAfterInSeconds": 300
}
```

创建授权方

您可以使用 CreateAuthorizerAPI 创建授权者。 以下示例说明了该命令。

```
aws iot create-authorizer
--authorizer-name MyAuthorizer
--authorizer-function-arn arn:aws:lambda:us-
west-2:<account_id>:function:MyAuthorizerFunction  //The ARN of the Lambda function.
[--token-key-name MyAuthorizerToken //The key used to extract the token from headers.
[--token-signing-public-keys FirstKey=
 "-----BEGIN PUBLIC KEY-----
  [...insert your public key here...]
  -----END PUBLIC KEY-----"
[--status ACTIVE]
[--tags <value>]
[--signing-disabled | --no-signing-disabled]
```

您可以使用 `signing-disabled` 参数选择退出适用于授权方的每次调用的签名验证。除有必要，否则我们强烈建议您不要禁用签名。签名验证可防止来自未知设备的 Lambda 函数过多调用。您无法在创建授权方后更新其 `signing-disabled` 状态。要更改此行为，您必须创建 `signing-disabled` 参数具有不同值的其它自定义授权方。

如果您已禁用签名，则 `tokenKeyName` 和 `tokenSigningPublicKeys` 参数的值是可选的。如果启用了签名，则它们是必需的值。

创建 Lambda 函数和自定义授权方后，必须明确授予 Amazon IoT Core 服务权限代表您调用函数。您可使用以下命令执行此操作。

```
aws lambda add-permission --function-name <lambda_function_name> --principal
iot.amazonaws.com --source-arn <authorizer_arn> --statement-id Id-123 --action
"lambda:InvokeFunction"
```

测试授权方

您可以使用 TestInvokeAuthorizerAPI 来测试授权方的调用和返回值。 API 使您能够指定协议元数据并在授权方中测试签名验证。

以下选项卡显示如何使用 Amazon CLI 来测试您的授权方。

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER \
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER ^
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER `
--token TOKEN_VALUE --token-signature TOKEN_SIGNATURE
```

`token-signature` 参数的值是签名令牌。要了解如何获取此值，请参阅 the section called “签名令牌” (p. 295)。

如果您的授权方使用了用户名和密码，您可以使用 `--mqtt-context` 参数传递此信息。以下选项卡显示如何使用 `TestInvokeAuthorizer` API 将包含用户名、密码和客户端名称的 JSON 对象发送到您的自定义授权方。

Unix-like

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  \
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows CMD

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  ^
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

Windows PowerShell

```
aws iot test-invoke-authorizer --authorizer-name NAME_OF_AUTHORIZER  `
--mqtt-context '{"username": "USER_NAME", "password": "dGVzdA==",
 "clientId":"CLIENT_NAME"}'
```

密码必须采用 base64 编码。以下示例说明如何在类 Unix 的环境中对密码进行编码。

```
echo -n PASSWORD | base64
```

管理自定义授权方

您可以使用以下 API 管理授权方。

- ListAuthorizers：显示您账户中的所有授权者。
- DescribeAuthorizer：显示指定授权者的属性。这些值包括创建日期、上次修改日期和其它属性。
- SetDefaultAuthorizer：为您的Amazon IoT Core数据端点指定默认授权者。 Amazon IoT Core如果设备未传递Amazon IoT Core凭证且未指定授权方，则将使用此授权方。有关使用 Amazon IoT Core 凭证的更多信息，请参阅 the section called “客户端身份验证” (p. 267)。
- UpdateAuthorizer：更改指定授权方的状态、令牌密钥名称或公有密钥。
- DeleteAuthorizer：删除指定的授权者。

> **Note**
>
> 您无法更新授权方的签名要求。这意味着您无法禁用需要签名的现有授权方的中签名。您也不能在不需要签名的现有授权方中要求签名。

使用自定义身份验证连接到 Amazon IoT Core

设备可以借助任何 Amazon IoT Core 支持进行设备消息收发的协议，使用自定义身份验证连接到 Amazon IoT Core。有关受支持的通信协议的更多信息，请参阅 the section called “设备通信协议” (p. 75)。 您传递给授权方 Lambda 函数的连接数据取决于您使用的协议。有关创建授权方 Lambda 函数的更多信息，请参阅 the section called “定义您的 Lambda 函数” (p. 289)。以下部分说明如何使用每个支持的协议连接到身份验证。

HTTPS

使用 HTTP Publish API 将数据发送给 Amazon IoT Core 的设备可以通过请求标头或 HTTP POST 请求中的查询参数传递凭证。设备可以使用 `x-amz-customauthorizer-name` 标头或查询参数指定要调用的授权方。如果您在授权方中启用了令牌签名，则必须使用请求标头或查询参数传递 *`token-key-name`* 和 `x-amz-customauthorizer-signature`。请注意，在浏览器中使用 JavaScript 时，该*`token-signature`*值必须为 URL 编码值。

以下示例请求显示了如何使用请求标头和查询参数传递这些参数。

```
//Passing credentials via headers
POST /topics/topic?qos=qos HTTP/1.1
Host: your-endpoint
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
x-amz-customauthorizer-name: authorizer-name

//Passing credentials via query parameters
POST /topics/topic?qos=qos&x-amz-customauthorizer-signature=token-signature&token-key-name=token-value HTTP/1.1
```

MQTT

使用 MQTT 连接连接到 Amazon IoT Core 的设备可以通过 MQTT 消息中的 `username` 和 `password` 字段传递凭证。`username` 值也可以选择将其他值（包括令牌、签名和授权方名称）传递给授权方的查询字符串包含在内。如果要使用基于令牌的身份验证方案而不是 `username` 和 `password` 值，则可以使用此查询字符串。

Note

密码字段中的数据由 Amazon IoT Core 进行 base64 编码。您的 Lambda 函数必须对其进行解码。

以下示例包含一个 `username` 字符串，其中带有指定令牌和签名的额外参数。

```
username?x-amz-customauthorizer-name=authorizer-name&x-amz-customauthorizer-
signature=token-signature&token-key-name=token-value
```

为了调用授权方，使用 MQTT 连接到 Amazon IoT Core 的设备和自定义身份验证必须连接端口 443。它们还必须传递带有值 `mqtt` 的应用程序层协议协商 (ALPN) TLS 扩展和带有其 Amazon IoT Core 数据终端节点的服务器名称指示 (SNI) 扩展。为避免潜在的错误，`x-amz-customauthorizer-signature` 的值应采用 URL 编码。我们还强烈建议 `x-amz-customauthorizer-name` 和 `token-key-name` 的值采用 URL 编码。有关这些值的更多信息，请参阅the section called “设备通信协议” (p. 75)。V2 Amazon IoT Device SDK、Mobile SDK 和 Amazon IoT Device Client (p. 1283) 可以配置这两个扩展。

MQTT 结束了 WebSockets

使用 MQTT 连接到Amazon IoT Core的设备 WebSockets 可以通过以下两种方式之一传递凭证。

- 通过 HTTP UPGRADE 请求中的请求标头或查询参数建立 WebSockets 连接。
- 通过 `username` 和 `password` 字段中的 MQTT CONNECT 消息。

如果您通过 MQTT 连接消息传递凭证，则需要使用 ALPN 和 SNI TLS 扩展。有关这些扩展的更多信息，请参阅 the section called “MQTT” (p. 294)。以下示例演示如何通过 HTTP Upgrade 请求传递凭证。

```
GET /mqtt HTTP/1.1
Host: your-endpoint
Upgrade: WebSocket
Connection: Upgrade
x-amz-customauthorizer-signature: token-signature
token-key-name: token-value
sec-WebSocket-Key: any random base64 value
sec-websocket-protocol: mqtt
sec-WebSocket-Version: websocket version
```

签名令牌

必须使用 `create-authorizer` 调用中用到的公私密钥对的私有秘钥签署令牌。以下示例说明如何使用类 UNIX 命令和创建令牌签名 JavaScript。它们使用 SHA-256 哈希算法对签名进行编码。

Command line

```
echo -n TOKEN_VALUE | openssl dgst -sha256 -sign PEM encoded RSA private key | openssl
 base64
```

JavaScript

```
const crypto = require('crypto')

const key = "PEM encoded RSA private key"

const k = crypto.createPrivateKey(key)
let sign = crypto.createSign('SHA256')
sign.write(t)
sign.end()
const s = sign.sign(k, 'base64')
```

授权方故障排除

本主题介绍可能导致自定义身份验证工作流中出现故障的常见问题以及解决这些问题的步骤。要最有效地解决问题，请启用 CloudWatch 日志，Amazon IoT Core然后将日志级别设置为 DEBUG。你可以在Amazon IoT Core控制台中启用 CloudWatch 日志 (https://console.aws.amazon.com/iot/)。有关启用和配置适用于 Amazon IoT Core 的日志的更多信息，请参阅 the section called “配置 Amazon IoT 日志记录” (p. 394)。

> Note
>
> 如果将日志级别长时间保留在 DEBUG， CloudWatch 可能会存储大量的日志记录数据。这可能会增加您的 CloudWatch费用。考虑使用基于资源的日志记录来增加仅针对特定事物组中设备的详细程度。有关基于资源的日志记录的更多信息，请参阅 the section called “配置 Amazon IoT 日志记录” (p. 394)。此外，当您完成故障排除时，将日志级别降到详细程度较低的级别。

在您开始故障排除之前，请查看 the section called “了解自定义身份验证工作流” (p. 288) 获取自定义身份验证过程的高度概要视图。这有助于您了解应该在何处查找问题的根源。

本主题将讨论以下两个方面供您调查。

- 与授权方 Lambda 函数相关的问题。
- 与您的设备相关的问题。

检查授权方的 Lambda 函数中是否存在问题

执行以下步骤以确保设备的连接尝试正在调用 Lambda 函数。

1. 验证哪个 Lambda 函数与您的授权方相关联。

   您可以通过调用 DescribeAuthorizerAPI 或通过单击Amazon IoT Core控制台安全部分中所需的授权方进行查证。
2. 检查 Lambda 函数的调用指标。为此，请执行以下步骤。

   a. 打开 Amazon Lambda 控制台 (https://console.aws.amazon.com/lambda/)，然后选择与您的授权方关联的函数。

   b. 选择监控选项卡并查看与您的问题相关的时间范围的指标。
3. 如果您没有看到任何调用，请验证 Amazon IoT Core 有权调用 Lambda 函数。如果您看到调用，则跳到下一步。执行以下步骤以验证 Lambda 函数是否具有所需的权限。

   a. 在 Amazon Lambda 控制台中选择您函数的权限选项卡。

   b. 在页面底部找到基于资源的策略部分。如果您的 Lambda 函数具有所需的权限，则策略类似于以下示例。

```
{
  "Version": "2012-10-17",
  "Id": "default",
  "Statement": [
    {
      "Sid": "Id123",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:111111111111:function:FunctionName",
      "Condition": {
```

```
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:iot:us-east-1:111111111111:authorizer/
AuthorizerName"
        },
        "StringEquals": {
          "AWS:SourceAccount": "111111111111"
        }
      }
    }
  ]
}
```

c. 此策略向您的函数授予对 Amazon IoT Core 委托人的 `InvokeFunction` 权限。如果您没有看到它，则必须使用 AddPermissionAPI 进行添加。以下示例说明如何使用 Amazon CLI 执行此操作。

```
aws lambda add-permission --function-name FunctionName --principal
 iot.amazonaws.com --source-arn AuthorizerARn --statement-id Id-123 --action
 "lambda:InvokeFunction"
```

4. 如果您看到调用，请验证没有错误。出现错误则可能表明 Lambda 函数未正确处理 Amazon IoT Core 发送给它的连接事件。

   有关在 Lambda 函数中处理事件的信息，请参阅 the section called “定义您的 Lambda 函数” (p. 289)。您可以将该测试功能用于 Amazon Lambda 控制台 (https://console.aws.amazon.com/lambda/) 对函数中的测试值进行硬编码，以确保函数正确处理事件。
5. 如果您看到没有错误的调用，但您的设备无法连接（或发布、订阅和接收消息），则问题可能是 Lambda 函数返回的策略未授予设备尝试执行的操作的权限。执行以下步骤以确定函数返回的策略是否存在任何问题。

   a. 使用 Amazon CloudWatch Logs Insights 查询可在短时间内扫描日志以检查故障。以下示例查询按时间戳对事件进行排序并查找失败。

   ```
   display clientId, eventType, status, @timestamp | sort @timestamp desc | filter
    status = "Failure"
   ```

   b. 更新您的 Lambda 函数以记录它返回到 Amazon IoT Core 的数据和触发函数的事件。您可以使用这些日志检查函数创建的策略。
6. 如果您看到调用并没有错误，但您的设备无法连接（或发布、订阅和接收消息），则另一个原因可能是 Lambda 函数超出了超时限制。自定义授权方的 Lambda 函数超时限制为 5 秒。您可以在 CloudWatch 日志或指标中检查函数持续时间。

调查设备问题

如果您发现调用 Lambda 函数或函数返回的策略不存在任何问题，请查找设备连接尝试是否存在问题。格式错误的连接请求可能会导致 Amazon IoT Core 不会触发您的授权方。TLS 层和应用程序层均可能出现连接问题。

TLS 层可能存在的问题：

- 客户必须在所有自定义身份验证请求中传递主机名标头（HTTP、MQTT 通过 WebSockets）或服务器名称指示 TLS 扩展（HTTP、MQTT 通过 WebSockets、MQTT 通过、MQTT）。在这两种情况下，传递的值必须与您账户的 Amazon IoT Core 数据终端节点相匹配。这些是执行以下 CLI 命令时返回的终端节点。
  - `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
  - `aws iot describe-endpoint --endpoint-type iot:Data`（适用于传统 VeriSign端点）
- 在 MQTT 连接中使用自定义身份验证的设备还必须发送应用程序层协议协商 (ALPN) TLS 扩展，并具有值 `mqtt`。
- 自定义身份验证当前仅在端口 443 上可用。

应用层可能存在的问题：

- 如果启用了签名（`signingDisabled` 字段在您的授权方为 false），请查找以下签名问题。
  - 请确保使用 `x-amz-customauthorizer-signature` 标头或查询字符串参数传递令牌签名。
  - 确保服务没有签署令牌以外的值。
  - 请确保使用您在授权方的 `token-key-name` 字段中指定的标头或查询参数中传递令牌。
- 请确保您使用 `x-amz-customauthorizer-name` 标头或查询字符串参数传递的授权方名称是有效的，或者您已为您的账户定义了默认授权方。

Authorization

授权是向经过身份验证的身份授予权限的过程。您可以使用 Amazon IoT Core 和 IAM policy 授予 Amazon IoT Core 中的权限。本主题介绍了 Amazon IoT Core 策略。有关 IAM policy 的更多信息，请参阅 适用于 Amazon IoT 的 Identity and Access Management (p. 348) 和 Amazon IoT 如何与 IAM 协同工作 (p. 351)。

Amazon IoT Core 策略确定经过身份验证的身份可执行的操作。经身份验证的身份由设备、移动应用程序、Web 应用程序和桌面应用程序使用。经过身份验证的身份甚至可以是键入 Amazon IoT Core CLI 命令的用户。仅当身份具有向其授予这些操作的权限的策略时，才能执行 Amazon IoT Core 操作。

Amazon IoT Core 策略和 IAM policy 都可用于 Amazon IoT Core 来控制身份（也称作委托人）可以执行的操作。您使用的策略类型取决于向 Amazon IoT Core 进行身份验证时使用的身份类型。

Amazon IoT Core 操作分为两组：

- 控制面板 API 允许您执行诸如创建或更新证书、事物、规则等管理任务。
- 数据层面 API 允许您向 Amazon IoT Core 发送数据以及从中接收数据。

您使用的策略类型取决于您使用的是控制面板 API 还是数据层面 API。

下表显示了身份类型、它们使用的协议和可用于授权的策略类型。

Amazon IoT Core 数据层面 API 和策略类型

协议和身份验证机制	SDK	身份类型	策略类型		
基于 TLS/TCP、TLS 双向身份验证的 MQTT（端口 8883 或 443）[†](#) (p. 75)	Amazon IoT Device SDK	X.509 证书	Amazon IoT Core 策略		
通过 HTTPS 的 MQTTWebSocket、Sigv4 身份验证的 MQTT（端口 443）	Amazon Mobile SDK	经过身份验证的 Amazon Cognito 身份	IAM 和 Amazon IoT Core 策略		
		未经身份验证的 Amazon Cognito 身份	IAM policy		
		IAM 或联合身份	IAM policy		

协议和身份验证机制	SDK	身份类型	策略类型		
HTTPS、Amazon Signature Version 4 身份验证（端口 443）	Amazon CLI	Amazon Cognito、IAM 或联合身份	IAM policy		
HTTPS、TLS 双向身份验证（端口 8443）	不支持 SDK	X.509 证书	Amazon IoT Core 策略		
基于自定义身份验证的 HTTPS（端口 443）	Amazon IoT Device SDK	自定义授权方	自定义授权方策略		

Amazon IoT Core 控制面板 API 和策略类型

协议和身份验证机制	SDK	身份类型	策略类型		
HTTPS Amazon Signature Version 4 身份验证（端口 443）	Amazon CLI	Amazon Cognito 身份	IAM policy		
		IAM 或联合身份	IAM policy		

Amazon IoT Core 策略附加到 X.509 证书或 Amazon Cognito 身份。IAM policy 附加到 IAM 用户、组或角色。如果您使用 Amazon IoT 控制台或 Amazon IoT Core CLI 附加策略（到证书或 Amazon Cognito Identity），则需要使用 Amazon IoT Core 策略。否则，您应使用 IAM policy。

基于策略的授权功能强大。它使您能够完全控制设备、用户或应用程序可在 Amazon IoT Core 中执行的操作。例如，以使用证书连接到 Amazon IoT Core 的设备为例。您可以允许设备访问所有 MQTT 主题，也可以限制它的访问权限，只允许它访问一个主题。再举一例，假设用户在命令行中键入 CLI 命令。通过使用策略，您可以允许或拒绝用户访问任何命令或 Amazon IoT Core 资源。此外，您还可以控制应用程序对 Amazon IoT Core 资源的访问。

对策略所做的更改可能需要几分钟才能生效，具体取决于 Amazon IoT 缓存策略文档的方式。也就是说，访问最近被授予访问权限的资源可能需要几分钟时间，并且资源可能在撤销访问权限后几分钟内仍可访问。

Amazon IoT Core 策略

Amazon IoT Core 策略是 JSON 文档。它们与 IAM policy 遵循相同的约定。Amazon IoT Core 支持命名策略，因此，许多身份都可以引用相同的策略文档。命名策略采用版本化，以便可以轻松回滚。

Amazon IoT Core 策略允许您控制对 Amazon IoT Core 数据层面的访问。Amazon IoT Core 数据层面由以下操作组成，允许您连接到 Amazon IoT Core 消息代理，发送和接收 MQTT 消息以及获取或更新事物的 Device Shadow。

Amazon IoT Core 策略是包含一个或多个策略语句的 JSON 文档。每个语句包含：

- `Effect`，指定是允许还是拒绝该操作。
- `Action`，用于指定策略允许或拒绝的操作。

- `Resource`，用于指定允许或拒绝对其执行操作的资源。

对策略所做的更改可能需要几分钟才能生效，具体取决于 Amazon IoT 缓存策略文档的方式。也就是说，访问最近被授予访问权限的资源可能需要几分钟时间，并且资源可能在撤销访问权限后几分钟内仍可访问。

主题

- Amazon IoT Core 策略操作 (p. 300)
- Amazon IoT Core 操作资源 (p. 302)
- Amazon IoT Core 策略变量 (p. 303)
- 防止跨服务混淆代理 (p. 308)
- Amazon IoT Core 策略示例 (p. 309)
- 使用 Amazon Cognito 身份的授权 (p. 337)

Amazon IoT Core 策略操作

Amazon IoT Core 定义了以下策略操作：

MQTT 策略操作

`iot:Connect`

表示连接到 Amazon IoT Core 消息代理所需的权限。每次向代理发送 `iot:Connect` 请求时，均检查 CONNECT 权限。消息代理禁止两个具有相同 ID 的客户端同时保持连接状态。在第二个客户端连接后，代理将关闭现有连接。`iot:Connect` 权限可用来确保仅使用特定客户端 ID 的授权客户端可以连接。

`iot:GetRetainedMessage`

代表获取保留单条消息内容的权限。保留的邮件是使用设置的保留标志发布并由 Amazon IoT Core 存储的邮件。有关获取账户所有保留消息列表的权限，请参阅 `iot:ListRetainedMessages` (p. 300)。

`iot:ListRetainedMessages`

表示检索有关帐户保留邮件的摘要信息（而不是邮件内容）的权限。保留的邮件是使用设置的保留标志发布并由 Amazon IoT Core 存储的邮件。为此操作指定的资源 ARN 必须为 `*`。有关获取保留单条消息内容的权限，请参阅 `iot:GetRetainedMessage` (p. 300)。

`iot:Publish`

代表向 MQTT 主题发布内容的权限。每次向代理发送 PUBLISH 请求时，均检查该权限。您可以使用此权限让客户端发布到特定主题模式。

> Note
>
> 要授予 `iot:Publish` 权限，还必须授予 `iot:Connect` 权限。

`iot:Receive`

表示从 Amazon IoT Core 接收消息所需的权限。每次向客户端交付消息时，都确认 `iot:Receive` 权限。由于每次交付时都会检查此权限，因此，可以使用它来吊销当前已订阅某个主题的客户端的权限。

物联网：RetainPublish

表示使用 RETAIN 标志集发布 MQTT 消息的权限。

> Note
>
> 要授予 `iot:RetainPublish` 权限，还必须授予 `iot:Publish` 权限。

`iot:Subscribe`

代表订阅主题筛选条件的权限。每次向代理发送 SUBSCRIBE 请求时，均检查该权限。用此权限让客户端可以订阅与特定主题模式相符的主题。

Note

要授予 `iot:Subscribe` 权限，还必须授予 `iot:Connect` 权限。

Device Shadow 策略操作

`iot:DeleteThingShadow`

表示删除事物的 Device Shadow 的权限。每次提出请求删除事物的 Device Shadow 内容时，都会检查 `iot:DeleteThingShadow` 权限。

`iot:GetThingShadow`

表示检索事物的 Device Shadow 的权限。每次请求检索 Device Shadow 的内容时，都会检查 `iot:GetThingShadow` 权限。

`iot:ListNamedShadowsForThing`

表示列出名为 Shadows 的事物的权限。每次请求列示名为 Shadows 的事物时，都会检查 `iot:ListNamedShadowsForThing` 权限。

`iot:UpdateThingShadow`

表示更新设备的影子的权限。每次请求更新事物的 Device Shadow 的内容时，都会检查 `iot:UpdateThingShadow` 权限。

Note

任务执行策略操作仅适用于 HTTP TLS 终端节点。如果您使用了 MQTT 终端节点，则必须使用本主题中定义的 MQTT 策略操作。
有关演示此操作的任务执行策略的示例，请参阅使用 MQTT 协议的 the section called “基本任务策略示例” (p. 336)。

任务执行 Amazon IoT Core 策略操作

`iot:DescribeJobExecution`

表示为给定事物检索任务执行的权限。每次请求获取任务执行时，都会检查 `iot:DescribeJobExecution` 权限。

`iot:GetPendingJobExecutions`

表示一个权限，用于为事物检索未处于最终状态的任务的列表。每次请求检索该列表时，都会检查 `iot:GetPendingJobExecutions` 权限。

`iot:UpdateJobExecution`

表示更新任务执行的权限。每次请求更新任务执行的状态时，都会检查 `iot:UpdateJobExecution` 权限。

`iot:StartNextPendingJobExecution`

表示一个权限，用于为事物获取和启动下一个待处理任务执行。（即，将状态为 QUEUED 的任务执行更新为状态 IN_PROGRESS。）每次请求启动下一个待处理任务执行，都会检查 `iot:StartNextPendingJobExecution` 权限。

Amazon IoT Core 凭证提供程序策略操作

`iot:AssumeRoleWithCertificate`

表示调用 Amazon IoT Core 凭证提供程序以使用基于证书的身份验证代入 IAM 角色的权限。每次向 Amazon IoT Core 凭证提供程序请求代入角色时，都会检查 `iot:AssumeRoleWithCertificate` 权限。

Amazon IoT Core 操作资源

要为 Amazon IoT Core 策略操作指定一项资源，则必须使用该资源的 ARN。所有资源 ARN 均采用以下形式：

```
arn:aws:iot:region:AWS-account-ID:Resource-type/Resource-name
```

下表列出了要为每种类型的操作指定的操作：

操作	资源类型	资源名称	ARN 示例
iot:Connect	client	客户端的客户端 ID	arn:aws:iot:us-east-1:123456789012:client/myClientId
iot:DeleteThingShadow	thing	事物名称	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:DescribeJobExecution	thing	事物名称	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetPendingJobExecutions	thing	事物名称	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetRetainedMessage	topic	保留的消息主题。	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:GetThingShadow	thing	事物名称	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:ListNamedShadowsForThing	全部	全部	*
iot:ListRetainedMessages	全部	全部	*
iot:Publish	topic	主题字符串	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:Receive	topic	主题字符串	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:RetainPublish	topic	用 REATIN 标志集发布的主题。	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:StartNextPendingJobExecution	thing	事物名称	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:Subscribe	topicfilter	主题筛选条件字符串	arn:aws:iot:us-east-1:123456789012:topicfilt myTopicFilter

操作	资源类型	资源名称	ARN 示例
iot:UpdateJobExecution	thing	事物名称	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:UpdateThingShadow	thing	事物的名称和影子的名称（如适用）	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:AssumeRoleWithCertificate	rolealias	指向角色 ARN 的角色别名	arn:aws:iot:us-east-1:123456789012:rolealias/CredentialProviderRole_alias

Amazon IoT Core 策略变量

Amazon IoT Core 在 Resource 或 Condition 块中定义可用于 Amazon IoT Core 策略的策略变量。评估策略时，将使用实际值替换策略变量。例如，如果设备使用客户端 ID 100-234-3456 连接到 Amazon IoT Core 消息代理，则 iot:ClientId 策略变量将在策略文档中替换为 100-234-3456。

Amazon IoT Core 策略可以使用通配符，并遵循与 IAM policy 类似的约定。在字符串中插入 *（星号）可以被视为通配符，匹配任何字符。例如，可以使用 * 在策略的 Resource 属性中描述多个 MQTT 主题名称。字符 + 和 # 在策略中被视为文字字符串。有关显示如何使用通配符的示例策略，请参阅在 MQTT 和 Amazon IoT Core 策略中使用通配符 (p. 317)。

还可以使用具有固定值的预定义策略变量来表示字符（这些字符本身有特殊的含义）。这些特殊字符包括 ${*}、${?} 和 ${$}。有关策略变量和特殊字符的更多信息，请参阅 IAM policy 元素：变量和标签和创建具有多个键或值的条件。

主题

- 基本 Amazon IoT Core 策略变量 (p. 303)
- 事物策略变量 (p. 304)
- X.509 证书 Amazon IoT Core 策略变量 (p. 305)

基本 Amazon IoT Core 策略变量

Amazon IoT Core 定义了以下基本策略变量：

- iot:ClientId：用于连接到 Amazon IoT Core 消息代理的客户端 ID。
- aws:SourceIp：连接到 Amazon IoT Core 消息代理的客户端的 IP 地址。

下面的 Amazon IoT Core 策略显示一个使用策略变量的策略。可以在策略的条件元素中使用 aws:SourceIp，以允许主体仅在特定地址范围内发出 API 请求。有关示例，请参阅 向用户和云服务授权以使用 Amazon IoT Jobs (p. 721)。

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
```

```
        "Resource": [
          "arn:aws:iot:us-east-1:123451234510:client/${iot:ClientId}"
        ]
      },
      {
        "Effect": "Allow",
        "Action": ["iot:Publish"],
        "Resource": [
          "arn:aws:iot:us-east-1:123451234510:topic/my/topic/${iot:ClientId}"
        ]
      }
    ]
}
```

在这些示例中，${iot:ClientId} 将替换为评估策略时连接到 Amazon IoT Core 消息代理的客户端的 ID。使用 ${iot:ClientId} 等策略变量时，您可能会无意中开放对意外主题的访问权限。例如，如果您使用借助 ${iot:ClientId} 来指定主题筛选条件的策略：

```
{
    "Effect": "Allow",
    "Action": ["iot:Subscribe"],
    "Resource": [
      "arn:aws:iot:us-east-1:123456789012:topicfilter/my/${iot:ClientId}/topic"
    ]
}
```

客户端可使用 + 作为客户端 ID 来进行连接。这样，用户便可以订阅与主题筛选条件 my/+/topic 匹配的任何主题。要防范此类安全漏洞，请使用 iot:Connect 策略操作来控制哪些客户端 ID 可以连接。例如，此策略仅允许那些客户端 ID 为 clientid1 的客户端建立连接：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": [
          "arn:aws:iot:us-east-1:123456789012:client/clientid1"
        ]
      }
    ]
}
```

事物策略变量

事物策略变量允许您根据事物名称、事物类型和事物属性值等事物属性来编写授予或拒绝权限的 Amazon IoT Core 策略。可以使用事物策略变量应用相同的策略来控制多个 Amazon IoT Core 设备。有关设备预调配的更多信息，请参阅设备预调配。事物名称从事物连接至 Amazon IoT Core 时发送的 MQTT Connect 消息中的客户端 ID 获取。

在 Amazon IoT Core 策略中使用事物策略变量时，请记住以下事项。

- 使用 AttachThingPrincipalAPI 将证书或委托人（经过身份验证的 Amazon Cognito 身份）附加到事物。
- 当您将事物名称替换为事物策略变量时，MQTT 连接消息或 TLS 连接中 clientId 的值必须与事物名称完全匹配。

可用的事物策略变量如下：

- iot:Connection.Thing.ThingName

它解析为 Amazon IoT Core 注册表中要评估策略的事物的名称。Amazon IoT Core 使用设备在进行身份验证时提供的证书，以确定用于验证连接的事物。此策略变量仅在设备通过 MQTT 或 MQTT 通过 WebSocket 协议连接时可用。

- `iot:Connection.Thing.ThingTypeName`

  它解析为与要评估策略的事物关联的事物类型。MQTT/WebSocket 连接的客户端 ID 必须与事物名称相同。此策略变量仅在通过 MQTT 或 MQTT 通过 WebSocket协议进行连接时可用。

- `iot:Connection.Thing.Attributes[`*`attributeName`*`]`

  它解析为与要评估策略的事物关联的指定属性的值。事物最多可以具有 50 个属性。每个属性都作为策略变量提供：`iot:Connection.Thing.Attributes[`*`attributeName`*`]`，其中 *`attributeName`* 是属性的名称。MQTT/WebSocket 连接的客户端 ID 必须与事物名称相同。此策略变量仅在通过 MQTT 或 MQTT 通过 WebSocket 协议进行连接时可用。

- `iot:Connection.Thing.IsAttached`

  `iot:Connection.Thing.IsAttached: ["true"]` 强制执行仅同时在 Amazon IoT 中注册并附加到可以访问策略内权限的委托人的设备。Amazon IoT Core如果设备提供的证书未附加到Amazon IoT Core注册表中的 IoT 事物，则您可以使用此变量阻止设备连接到。此变量具有值，`true`或者`false`表示连接的事物已使用 AttachThingPrincipalAPI 附加到证书或 Amazon Cognito 身份。事物名称被视为客户端 ID。

X.509 证书 Amazon IoT Core 策略变量

X.509 证书策略变量允许您根据 X.509 证书属性编写授予权限的 Amazon IoT Core 策略。以下部分介绍如何使用这些证书策略变量。

CertificateId

在 RegisterCertificateAPI 中，`certificateId`出现在响应正文中。要获取有关您的证书的信息，您可以使用`certificateId`中DescribeCertificate。

颁发者属性

以下 Amazon IoT Core 策略变量允许您根据由证书颁发者设置的证书属性来授予或拒绝权限。

- `iot:Certificate.Issuer.DistinguishedNameQualifier`
- `iot:Certificate.Issuer.Country`
- `iot:Certificate.Issuer.Organization`
- `iot:Certificate.Issuer.OrganizationalUnit`
- `iot:Certificate.Issuer.State`
- `iot:Certificate.Issuer.CommonName`
- `iot:Certificate.Issuer.SerialNumber`
- `iot:Certificate.Issuer.Title`
- `iot:Certificate.Issuer.Surname`
- `iot:Certificate.Issuer.GivenName`
- `iot:Certificate.Issuer.Initials`
- `iot:Certificate.Issuer.Pseudonym`
- `iot:Certificate.Issuer.GenerationQualifier`

使用者属性

以下 Amazon IoT Core 策略变量允许您根据由证书颁发者设置的证书使用者属性来授予或拒绝权限。

- iot:Certificate.Subject.DistinguishedNameQualifier
- iot:Certificate.Subject.Country
- iot:Certificate.Subject.Organization
- iot:Certificate.Subject.OrganizationalUnit
- iot:Certificate.Subject.State
- iot:Certificate.Subject.CommonName
- iot:Certificate.Subject.SerialNumber
- iot:Certificate.Subject.Title
- iot:Certificate.Subject.Surname
- iot:Certificate.Subject.GivenName
- iot:Certificate.Subject.Initials
- iot:Certificate.Subject.Pseudonym
- iot:Certificate.Subject.GenerationQualifier

X.509 证书允许这些属性包含一个或多个值。预设情况下，每个多值属性的策略变量会返回第一个值。例如，Certificate.Subject.Country 属性可能包含国家/地区名称列表，但在策略中进行评估时，iot:Certificate.Subject.Country 会替换为第一个国家/地区名称。您可以使用从 1 开始的索引请求第一个值以外的特定属性值。例如，iot:Certificate.Subject.Country.1 由 Certificate.Subject.Country 属性中第二个国家/地区名称替换。如果您指定不存在的索引值（例如，如果您在只有两个值分配到属性时请求第三个值），则不会执行替换功能，并且授权将失败。您可以在策略变量名称中使用 .List 后缀指定属性的所有值。

Registered devices (2)

对于在 Amazon IoT Core 注册表中注册为事物的设备，以下策略允许在 Amazon IoT Core 注册表中注册了事物名称的客户端进行连接，但仅限将证书的 Certificate.Subject.Organization 属性设置为 "Example Corp" 或 "AnyCompany" 的客户端有权发布到事物名称特定的主题。通过使用 "Condition" 字段来完成此限制，该字段指定要允许前面的操作而必须满足的条件。在此情况下，条件是与证书关联的 Certificate.Subject.Organization 属性必须包括列出的值之一：

```
{
    "Version":"2012-10-17",
    "Statement":[
      {
        "Effect":"Allow",
        "Action":[
          "iot:Connect"
        ],
        "Resource":[
          "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
        ]
      },
      {
        "Effect":"Allow",
        "Action":[
          "iot:Publish"
        ],
        "Resource":[
          "arn:aws:iot:us-east-1:123456789012:topic/my/topic/
${iot:Connection.Thing.ThingName}"
        ],
        "Condition":{
          "ForAllValues:StringEquals":{
            "iot:Certificate.Subject.Organization.List":[
              "Example Corp",
              "AnyCompany"
```

```
            ]
          }
        }
      }
    ]
}
```

Unregistered devices (2)

对于未在 Amazon IoT Core 注册表中注册为事物的设备，以下策略授予权限以使用客户端 ID `client1`、`client2` 和 `client3` 连接到 Amazon IoT Core，但仅限将证书的 `Certificate.Subject.Organization` 属性设置为 `"Example Corp"` 或 `"AnyCompany"` 的客户端有权发布到客户端 ID 特定的主题。通过使用 `"Condition"` 字段来完成此限制，该字段指定要允许前面的操作而必须满足的条件。在此情况下，条件是与证书关联的 `Certificate.Subject.Organization` 属性必须包括列出的值之一：

```
{
    "Version":"2012-10-17",
    "Statement":[
      {
        "Effect":"Allow",
        "Action":[
          "iot:Connect"
        ],
        "Resource":[
          "arn:aws:iot:us-east-1:123456789012:client/client1",
          "arn:aws:iot:us-east-1:123456789012:client/client2",
          "arn:aws:iot:us-east-1:123456789012:client/client3"
        ]
      },
      {
        "Effect":"Allow",
        "Action":[
          "iot:Publish"
        ],
        "Resource":[
          "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
        ],
        "Condition":{
          "ForAllValues:StringEquals":{
            "iot:Certificate.Subject.Organization.List":[
              "Example Corp",
              "AnyCompany"
            ]
          }
        }
      }
    ]
}
```

颁发者备用名称属性

以下 Amazon IoT Core 策略变量允许您根据由证书颁发者设置的颁发者备用名称属性来授予或拒绝权限。

- `iot:Certificate.Issuer.AlternativeName.RFC822Name`
- `iot:Certificate.Issuer.AlternativeName.DNSName`
- `iot:Certificate.Issuer.AlternativeName.DirectoryName`
- `iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Issuer.AlternativeName.IPAddress`

使用者备用名称属性

以下 Amazon IoT Core 策略变量允许您根据由证书颁发者设置的使用者备用名称属性来授予或拒绝权限。

- `iot:Certificate.Subject.AlternativeName.RFC822Name`
- `iot:Certificate.Subject.AlternativeName.DNSName`
- `iot:Certificate.Subject.AlternativeName.DirectoryName`
- `iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier`
- `iot:Certificate.Subject.AlternativeName.IPAddress`

其它属性

您可以使用 `iot:Certificate.SerialNumber` 来根据证书的序列号允许或拒绝访问 Amazon IoT Core 资源。`iot:Certificate.AvailableKeys` 策略变量包含具有值的所有证书策略变量的名称。

X.509 证书策略变量限制

以下限制适用于 X.509 证书策略变量：

通配符

如果证书属性中有通配符，则策略变量不会由证书属性值替换，并在策略文档中保留 `${policy-variable}` 文本。这可能会导致授权失败。可以使用以下通配符：`*`、`$`、`+`、`?` 和 `#`。

数组字段

包含数组的证书属性限制为五项。其它的项将被忽略。

字符串长度

所有字符串值的长度限制为 1024 个字符。如果证书属性包含超过 1024 个字符的字符串，则策略变量不会由证书属性值替换，并在策略文档中保留 `${policy-variable}`。这可能会导致授权失败。

特殊字符

在策略变量中使用时，任何特殊字符（例如 `,`、`"`、`\`、`+`、`=`、`<`、`>` 和 `;`）都必须使用反斜杠 (`\`) 作为前缀。例如，`Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US` 改为 `Amazon Web Service O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US`。

防止跨服务混淆代理

混淆代理问题是一个安全问题，即没有执行操作权限的实体可能会迫使更具权限的实体执行该操作。在 Amazon 中，跨服务模拟可能会导致混淆代理问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况，Amazon 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

要限制 Amazon IoT 为另一项服务提供的资源访问权限，我们建议使用资源策略中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键。如果使用两个全局条件上下文键，在同一策略语句中使用时，`aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户必须使用相同的账户 ID。

防止混淆代理问题最有效的方法是使用具有资源完整 Amazon Resource Name (ARN) 的 `aws:SourceArn` 全局条件上下文键。对于 Amazon IoT，您的 `aws:SourceArn` 必须符合格式：`arn:aws:iot:`*`region`*`:`*`account-id`*`:*`。确保*`##`*与您的 Amazon IoT 区域匹配，*`account-id`* 与您的客户账户 ID 相匹配。

以下示例说明了如何通过使用 Amazon IoT 角色信任策略中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键防止混淆代理问题。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:iot:us-east-1:123456789012:*"
        }
      }
    }
  ]
}
```

Amazon IoT Core 策略示例

本节中的策略示例说明了用于完成 Amazon IoT Core 中常见任务的策略文档。为解决方案创建策略时，您可以将其作为启动示例。

本节中的示例使用以下策略元素：

- the section called “Amazon IoT Core 策略操作” (p. 300)
- the section called “Amazon IoT Core 操作资源” (p. 302)
- the section called “基于身份的策略示例” (p. 370)
- the section called “基本 Amazon IoT Core 策略变量” (p. 303)
- the section called “X.509 证书 Amazon IoT Core 策略变量” (p. 305)

此节中的策略示例：

- 连接策略示例 (p. 309)
- 发布/订阅策略示例 (p. 316)
- 连接和发布策略示例 (p. 329)
- 保留的消息策略示例 (p. 330)
- 证书策略示例 (p. 331)
- 事物策略示例 (p. 335)
- 基本任务策略示例 (p. 336)

连接策略示例

以下策略授予权限以使用客户端 ID `client1` 连接到 Amazon IoT Core：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "iot:Connect"
```

```
        ],
        "Resource": [
          "arn:aws:iot:us-east-1:123456789012:client/client1"
        ]
      }
    ]
}
```

以下策略拒绝使用客户端 ID `client1` 和 `client2` 连接到 Amazon IoT Core 的权限，同时允许设备使用与已在 Amazon IoT Core 注册表中注册的事物名称匹配的客户端 ID 进行连接：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Deny",
        "Action": [
          "iot:Connect"
        ],
        "Resource": [
          "arn:aws:iot:us-east-1:123456789012:client/client1",
          "arn:aws:iot:us-east-1:123456789012:client/client2"
        ]
      },
      {
        "Effect": "Allow",
        "Action": [
          "iot:Connect"
        ],
        "Resource": [
          "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
        ]
      }
    ]
}
```

MQTT 持久性会话策略示例

`connectAttributes` 允许您在 IAM policy 中指定要在连接消息中使用的属性，如 `PersistentConnect` 和 `LastWill`。有关更多信息，请参阅 使用 ConnectAttributes (p. 85)

以下策略允许连接 `PersistentConnect` 功能：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        }
    ]
}
```

以下策略不允许 PersistentConnect，但允许使用其它功能：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringNotEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        }
    ]
```

上述策略也可以使用 StringEquals 表达，但允许使用包括新功能在内的任何其它功能：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        }
    ]
}
```

以下策略允许通过 PersistentConnect 和 LastWill 连接，但不允许使用任何其它新功能：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
```

```
                    "iot:ConnectAttributes": [
                        "PersistentConnect",
                        "LastWill"
                    ]
                }
            }
        }
    ]
}
```

以下策略允许客户端进行干净连接，无论是否具有 LastWill，但不允许使用其它功能：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "LastWill"
                    ]
                }
            }
        }
    ]
}
```

以下策略仅允许使用默认功能进行连接：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                    ]
                }
            }
        }
    ]
}
```

以下策略仅允许使用 PersistentConnect 连接，但只要连接使用 PersistentConnect，则允许使用任何新功能：

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

```
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        }
    ]
}
```

以下策略规定连接必须同时使用 PersistentConnect 和 LastWill，而不允许使用新功能：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect",
                        "LastWill"
                    ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "LastWill"
                    ]
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
```

```
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                    ]
                }
            }
        }
    ]
}
```

以下策略不能拥有 PersistentConnect，但可以有 LastWill，不允许使用任何其它新功能：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "iot:ConnectAttributes": [
                        "LastWill"
                    ]
                }
            }
        }
    ]
}
```

以下策略仅允许包括带有主题 "my/lastwill/topicName" 的 LastWill 客户端连接，同时允许任何使用 LastWill 主题的功能:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
```

```
                    "iot:LastWillTopic": "arn:aws:iot:*region*:*account-id*:topic/*my/
lastwill/topicName*"
                }
            }
        }
    ]
}
```

以下策略仅允许使用特定 LastWillTopic 的干净连接，同时允许任何使用 LastWillTopic 的功能：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "iot:LastWillTopic": "arn:aws:iot:*region*:*account-id*:topic/*my/
lastwill/topicName*"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:ConnectAttributes": [
                        "PersistentConnect"
                    ]
                }
            }
        }
    ]
}
```

Registered devices (3)

以下策略授予设备使用其事物名称作为客户端 ID 进行连接和订阅主题筛选条件 my/topic/filter 的权限。设备必须注册到 Amazon IoT Core。当设备连接到 Amazon IoT Core 时，它必须提供与 Amazon IoT Core 注册表中的 IoT 事物关联的证书：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "iot:Connect"
        ],
        "Resource": [
          "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
        ]
      },
      {
        "Effect": "Allow",
```

```
        "Action": [
          "iot:Subscribe"
        ],
        "Resource": [
          "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic/filter"
        ]
      }
    ]
}
```

Unregistered devices (3)

对于未在 Amazon IoT Core 注册表中注册为事物的设备，以下策略授予权限以使用客户端 ID `client1` 进行连接以及订阅主题筛选条件 `my/topic`：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/my/topic"
            ]
        }
    ]
}
```

发布/订阅策略示例

您使用的策略取决于您连接到 Amazon IoT Core 的方式。您可以使用 MQTT 客户端、HTTP 或连接到 WebSocket。Amazon IoT Core通过 MQTT 客户端连接时，将使用 X.509 证书进行身份验证。通过 HTTP 或 WebSocket 协议连接时，将使用 Signature Version 4 和 Amazon Cognito 进行身份验证。

本节内容：

- 在 MQTT 和 Amazon IoT Core 策略中使用通配符 (p. 317)
- 向特定主题发布、订阅消息/从特定主题接收消息的策略 (p. 317)
- 向具有特定前缀的主题发布、订阅消息/从具有特定前缀的主题接收消息的策略 (p. 319)
- 向特定于每台设备的主题发布、订阅消息/从特定于每台设备的主题接收消息的策略 (p. 320)
- 向主题名称中包含事物属性的主题发布、订阅消息/从主题名称中包含事物属性的主题接收消息的策略 (p. 322)
- 拒绝向主题名称的子主题发布消息的策略 (p. 323)
- 拒绝接收来自主题名称的子主题的消息的策略 (p. 325)
- 使用 MQTT 通配符订阅主题的策略 (p. 326)
- 适用于 HTTP 和 WebSocket客户端的策略 (p. 327)

在 MQTT 和 Amazon IoT Core 策略中使用通配符

MQTT 和 Amazon IoT Core 策略具有不同的通配符，因此选择时应仔细斟酌。在 MQTT 中，可在 MQTT 主题筛选条件中使用通配符 + 和 # 来订阅多个主题名称。Amazon IoT Core 策略使用 * 和 ? 作为通配符，并遵循 IAM policy 的约定。在策略文档中，* 表示字符的任意组合，问号 ? 表示任何单个字符。在策略文档中，MQTT 通配符 + 和 # 被视为没有特殊含义的字符。要在策略的 `resource` 属性中描述多个主题名称和主题筛选条件，请使用 * 和 ? 通配符代替 MQTT 通配符。

当选择要在策略文档中使用的通配符时，请考虑 * 字符不限于单个主题级别，因为 + 字符位于 MQTT 主题筛选条件中。为了帮助将通配符规范限制为单个 MQTT 主题筛选条件级别，请考虑使用多个 ? 字符。有关在策略资源中使用通配符的更多信息以及它们所匹配内容的更多示例，请参阅在资源 ARN 中使用通配符。

下表显示了 MQTT 和 MQTT 客户端的 Amazon IoT Core 策略中使用的不同通配符。

通配符	是 MQTT 通配符	MQTT 中的示例	是 Amazon IoT Core 策略通配符	适用于 MQTT 客户端 Amazon IoT Core 策略的示例
#	是	`some/#`	否	不适用
+	是	`some/+/topic`	否	不适用

*	否	不适用	是	`topicfilter/some/*/topic`
				
				`topicfilter/some/sensor*/topic`

?	否	不适用	是	`topic/some/?????/topic`
				
				`topicfilter/some/sensor???/topic`

向特定主题发布、订阅消息/从特定主题接收消息的策略

以下显示的是已注册和未注册设备向名为“some_specific_topic”的主题发布、订阅消息/从名为“some_specific_topic”的主题接收消息的示例。这些示例还强调了 `Publish` 和 `Receive` 使用“主题”作为资源，`Subscribe` 使用“主题筛选器”作为资源。

Registered devices

对于在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备连接到与 Amazon IoT Core 的注册表中的事物名称相匹配的 clientId。它还为名为“some_specific_topic”的主题提供 `Publish`、`Subscribe` 和 `Receive` 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        }
    ]
}
```

Unregistered devices

对于未在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备使用 clientId1、clientId2 或 clientId3 进行连接。它还为名为“some_specific_topic”的主题提供 Publish、Subscribe 和 Receive 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:topicfilter/some_specific_topic"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/some_specific_topic"
            ]
        }
    ]
}
```

向具有特定前缀的主题发布、订阅消息/从具有特定前缀的主题接收消息的策略

以下显示的是已注册和未注册的设备向具有前缀“topic_prefix”的主题发布、订阅消息/从具有前缀“topic_prefix”的主题接收消息的示例。

> **Note**
>
> 请注意，在此示例中使用了通配符 *。尽管通配符 * 对于在单个语句中为多个主题名称提供权限很有用，但它可能会给设备提供比所需更多的权限，从而导致意想不到的后果。因此，我们建议您仅在仔细考虑后才使用通配符 *。

Registered devices

对于在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备连接到与 Amazon IoT Core 的注册表中的事物名称相匹配的 clientId。它还会为具有前缀“topic_prefix”的主题提供 `Publish`、`Subscribe` 和 `Receive` 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
            ]
```

```
        }
    ]
}
```

Unregistered devices

对于未在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备使用 clientId1、clientId2 或 clientId3 进行连接。它还会为具有前缀“topic_prefix”的主题提供 `Publish`、`Subscribe` 和 `Receive` 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix*"
            ]
        }
    ]
}
```

向特定于每台设备的主题发布、订阅消息/从特定于每台设备的主题接收消息的策略

以下显示的是已注册和未注册的设备向特定于给定设备的主题发布、订阅消息/从特定于给定设备的主题接收消息的示例。

Registered devices

对于在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备连接到与 Amazon IoT Core 的注册表中的事物名称相匹配的 clientId。它提供向特定事物主题 (`sensor/device/${iot:Connection.Thing.ThingName}`) 发布内容的权限，还提供订阅特定事物主题 (`command/device/${iot:Connection.Thing.ThingName}`) 和从特定事物主题接收内容的权限。如果注册表中的事物名称为“thing1”，将允许设备发布到主题“sensor/device/thing1”，也可以订阅主题“command/device/thing1”和从此主题接收内容。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

Unregistered devices

对于未在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备使用 clientId1、clientId2 或 clientId3 进行连接。它提供发布到客户特定主题 (`sensor/device/${iot:ClientId}`) 的权限，还提供订阅客户特定主题 (`command/device/${iot:ClientId}`) 和从此客户特定主题接收内容的权限。如果设备以 clientId1 的身份连接 clientId，将允许它发布到主题“sensor/device/clientId1”，还可以订阅主题 `device/clientId1/command` 和从该主题接收内容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
```

```
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/sensor/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/command/device/
${iot:Connection.Thing.ThingName}"
            ]
        }
    ]
}
```

向主题名称中包含事物属性的主题发布、订阅消息/从主题名称中包含事物属性的主题接收消息的策略

以下显示的是已注册的设备向名称中包含事物属性的主题发布、订阅消息/从名称中包含事物属性的主题接收消息的示例。

> **Note**
>
> 仅在 Amazon IoT Core 的注册表中注册的设备存在事物属性。对于未注册的设备，没有相应的示例。

Registered devices

对于在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备连接到与 Amazon IoT Core 的注册表中的事物名称相匹配的 clientId。它提供发布到主题 (`sensor/${iot:Connection.Thing.Attributes[version]}`)，以及订阅主题名称中包含事物属性的主题 (`command/${iot:Connection.Thing.Attributes[location]}`) 和从该主题接收内容的权限。如果注册表中的事物名称中包含 `version=v1` 和 `location=Seattle`，将允许设备发布到主题“sensor/v1”，以及订阅主题“command/Seattle”和从该主题接收内容。

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

```
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/sensor/
${iot:Connection.Thing.Attributes[version]}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topicfilter/command/
${iot:Connection.Thing.Attributes[location]}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/command/
${iot:Connection.Thing.Attributes[location]}"
            ]
        }
    ]
}
```

Unregistered devices

由于仅在 Amazon IoT Core 的注册表中注册的设备存在事物属性，因此未注册的事物没有相应的示例。

拒绝向主题名称的子主题发布消息的策略

以下显示的是已注册和未注册的设备向除某些子主题之外的所有主题发布消息的示例。

Registered devices

对于在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备连接到与 Amazon IoT Core 的注册表中的事物名称相匹配的 clientId。它允许发布到所有以“department/”为前缀的主题，但不允许发布到“department/admins”子主题。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
            ]
        }
    ]
}
```

Unregistered devices

对于未在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备使用 clientId1、clientId2 或 clientId3 进行连接。它允许发布到所有以“department/”为前缀的主题，但不允许发布到“department/admins”子主题。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/department/admins"
            ]
        }
    ]
```

```
}
```

拒绝接收来自主题名称的子主题的消息的策略

以下显示的是已注册和未注册的设备订阅除某些子主题之外具有特定前缀的主题，以及从此类主题接收消息的示例。

Registered devices

对于在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备连接到与 Amazon IoT Core 的注册表中的事物名称相匹配的 clientId。此策略允许设备订阅任何具有前缀“topic_prefix”的主题。通过在 `iot:Receive` 的语句中使用 `NotResource`，我们允许设备接收来自设备订阅的所有主题（前缀为“topic_prefix/restricted”的主题除外）的消息。例如，通过此策略，设备可以订阅“topic_prefix/topic1”，甚至“topic_prefix/restricted”，但是，它们只能接收来自主题“topic_prefix/topic1”的消息，而不能接收来自主题“topic_prefix/restricted”的消息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix/*"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix/
restricted/*"
        }
    ]
}
```

Unregistered devices

对于未在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备使用 clientId1、clientId2 或 clientId3 进行连接。此策略允许设备订阅任何具有前缀“topic_prefix”的主题。通过在 `iot:Receive` 的语句中使用 `NotResource`，我们允许设备接收来自设备订阅的所有主题（前缀为“topic_prefix/restricted”的主题除外）的消息。例如，通过此策略，设备可以订阅“topic_prefix/topic1”，甚至“topic_prefix/restricted”，但是，它们只能接收来自主题“topic_prefix/topic1”的消息，而不能接收来自主题“topic_prefix/restricted”的消息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
```

```
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/topic_prefix/*"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "NotResource": "arn:aws:iot:us-east-1:123456789012:topic/topic_prefix/
restricted/*"
        }
    ]
}
```

使用 MQTT 通配符订阅主题的策略

MQTT 通配符 + 和 # 被视为文字字符串，但在 Amazon IoT Core 策略中使用时不被视为通配符。在 MQTT 中，+ 和 # 仅在订阅主题筛选器时被视为通配符，但在所有其他上下文中均被视为文字字符串。我们建议您仅在经过仔细考虑之后再将这些 MQTT 通配符用作 Amazon IoT Core 策略的一部分。

以下显示的是在 Amazon IoT Core 策略中使用 MQTT 通配符的已注册和未注册事物的示例。这些通配符被视为文字字符串。

Registered devices

对于在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备连接到与 Amazon IoT Core 的注册表中的事物名称相匹配的 clientId。此策略允许设备订阅主题“部门/+/员工”和“位置/#”。请注意，由于 + 和 # 在 Amazon IoT Core 策略中被视为文字字符串，因此设备可以订阅主题“部门/+/员工”，但不能订阅主题“部门/工程/员工”。同样，设备可以订阅主题“位置/#”，但不能订阅主题“位置/西雅图”。但是，设备订阅了主题“部门/+/员工”后，该策略将允许它们接收来自主题“部门/工程/员工”的消息。同样，设备订阅主题“位置/#”后，它们也将收到来自主题“位置/西雅图”的消息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/department/+/
employees"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/location/#"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
        }
    ]
}
```

Unregistered devices

对于未在 Amazon IoT Core 的注册表中注册的设备，以下策略允许设备使用 clientId1、clientId2 或 clientId3 进行连接。此策略允许设备订阅主题“部门/+/员工”和“位置/#”。请注意，由于 + 和 # 在 Amazon IoT Core 策略中被视为文字字符串，因此设备可以订阅主题“部门/+/员工”，但不能订阅主题“部门/工程/员工”。同样，设备可以订阅主题“位置/#”，但不能订阅主题“位置/西雅图”。但是，设备订阅了主题“部门/+/员工”后，该策略将允许它们接收来自主题“部门/工程/员工”的消息。同样，设备订阅主题“位置/#”后，它们也将收到来自主题“位置/西雅图”的消息。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/clientId1",
                "arn:aws:iot:us-east-1:123456789012:client/clientId2",
                "arn:aws:iot:us-east-1:123456789012:client/clientId3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/department/+/
employees"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topicfilter/location/#"
        },
        {
            "Effect": "Allow",
            "Action": "iot:Receive",
            "Resource": "arn:aws:iot:us-east-1:123456789012:topic/*"
        }
    ]
}
```

适用于 HTTP 和 WebSocket客户端的策略

通过 HTTP 或 WebSocket 协议连接时，将使用 Signature Version 4 和 Amazon Cognito 进行身份验证。Amazon Cognito 身份可以是经过身份验证的，也可以是未经身份验证的。经过身份验证的身份属于已通过任何受支持的身份提供商进行身份验证的用户。未经身份验证的身份通常属于未使用身份提供商进行身份验证的来宾用户。Amazon Cognito 提供了一个唯一的标识符和 Amazon 凭证以支持未经身份验证的身份。有关更多信息，请参阅the section called “使用 Amazon Cognito 身份的授权” (p. 337)：

对于以下操作，Amazon IoT Core 使用附加到 Amazon Cognito 身份（通过 `AttachPolicy` API）的 Amazon IoT Core 策略缩小附加到 Amazon Cognito Identity 池（由经过身份验证的身份组成）的权限范围。

- `iot:Connect`

- `iot:Publish`
- `iot:Subscribe`
- `iot:Receive`
- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

这意味着，Amazon Cognito 身份需要从附加到池的 IAM 角色策略以及通过 Amazon IoT Core `AttachPolicy` API附加到 Amazon Cognito Identity 附加到 Amazon IoT Core 策略处获得权限。

经过身份验证和未经身份验证的用户是不同的身份类型。如果您没有将 Amazon IoT 策略附加到 Amazon Cognito Identity，则经过身份验证的用户无法在 Amazon IoT 中获得授权并无权访问 Amazon IoT 资源和操作。

> **Note**
>
> 对于其它 Amazon IoT Core 操作或未经身份验证的身份，Amazon IoT Core 不会缩小附加到 Amazon Cognito 身份池角色的权限范围。无论是对于经过身份验证的身份还是未经过身份验证的身份，这都是我们建议附加到 Amazon Cognito 池角色的最宽松的策略。

HTTP

要允许未经过身份验证的 Amazon Cognito 身份通过 HTTP 向特定于 Amazon Cognito Identity 的主题发布消息，请将以下 IAM policy 附加到 Amazon Cognito Identity 池角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish",
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"]
        }
    ]
}
```

要允许经过身份验证的用户执行此操作，请使用Amazon IoT Core AttachPolicyAPI 将之前的策略附加到 Amazon Cognito Identity 池角色和 Amazon Cognito Identity。

> **Note**
>
> 在向 Amazon Cognito 身份授权时，Amazon IoT Core 会考虑这两个策略并授予指定的最小权限。仅当两个策略都允许请求的操作时，才允许操作。如果任一策略不允许某项操作，则该操作未经授权。

MQTT

要允许未经过身份验证的 Amazon Cognito 身份向特定 WebSocket 于账户中的 Amazon Cognito Identity 的主题发布 MQTT 消息，请将以下 IAM policy 附加到 Amazon Cognito Identity 池角色：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
```

```
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-
identity.amazonaws.com:sub}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/${cognito-
identity.amazonaws.com:sub}"]
        }
    ]
}
```

要允许经过身份验证的用户执行此操作，请使用Amazon IoT Core AttachPolicyAPI 将之前的策略附加到 Amazon Cognito Identity 池角色和 Amazon Cognito Identity。

> Note
>
> 在向 Amazon Cognito 身份授权时，Amazon IoT Core 会考虑这两个策略并授予指定的最小权限。仅当两个策略都允许请求的操作时，才允许操作。如果任一策略不允许某项操作，则该操作未经授权。

连接和发布策略示例

对于在 Amazon IoT Core 注册表中注册为事物的设备，以下策略授予权限以使用与事物名称匹配的客户端 ID 连接到 Amazon IoT Core，并将设备限制为在客户端 ID 或事物名称特定的 MQTT 主题上发布。要成功建立连接，必须在 Amazon IoT Core 注册表中注册事物名称，并使用附加到事物的身份或委托人对事物名称进行身份验证：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action":["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}"]
      },
      {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
      }
    ]
}
```

对于未在 Amazon IoT Core 注册表中注册为事物的设备，以下策略授予权限以使用客户端 ID `client1` 连接到 Amazon IoT Core，并将设备限制为在客户端 ID 特定的 MQTT 主题上发布：

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action":["iot:Publish"],
        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:ClientId}"]
      },
      {
        "Effect": "Allow",
        "Action": ["iot:Connect"],
```

```
        "Resource": ["arn:aws:iot:us-east-1:123456789012:client/client1"]
      }
    ]
}
```

保留的消息策略示例

使用保留的消息 (p. 81)需要具体策略。保留消息是使用设置的 RETAIN 标志发布并由 Amazon IoT Core 存储的 MQTT 消息。本节介绍了允许常见使用保留消息的策略示例。

本节内容：

- 连接和发布保留消息的策略 (p. 330)
- 连接和发布保留的 Will 消息的策略 (p. 330)
- 列出和获取保留消息的策略 (p. 331)

连接和发布保留消息的策略

对于要发布保留消息的设备，设备必须能够连接、发布（任何 MQTT 消息）和发布 MQTT 保留的消息。以下策略为主题授予了这些权限：`device/sample/configuration` 到客户端 **`device1`**。有关授予连接权限的另一个示例，请参阅 the section called “连接和发布策略示例” (p. 329)。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/device1"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/device/sample/configuration"
   ]
  }
 ]
}
```

连接和发布保留的 Will 消息的策略

客户端可以配置一条 Amazon IoT Core 将在客户端意外断开连接时发布的消息。MQTT 称这样的消息为 Will消息。客户端必须在连接权限中添加附加条件才能包含这些条件。

以下策略文档授予所有客户端连接和发布由主题、`will` 标识的 Will 消息的权限，Amazon IoT Core 也将保留。

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

```
    "iot:Connect"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/*"
   ],
   "Condition": {
    "ForAllValues:StringEquals": {
     "iot:ConnectAttributes": [
      "LastWill"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:Publish",
    "iot:RetainPublish"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/will"
   ]
  }
 ]
}
```

列出和获取保留消息的策略

服务和应用程序可以通过调用 ListRetainedMessages和GetRetainedMessage 来访问保留的消息，而无需支持 MQTT 客户端。调用这些操作的服务和应用程序必须使用如下示例的策略进行授权。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "iot:ListRetainedMessages"
   ],
   "Resource": [
    "*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "iot:GetRetainedMessage"
   ],
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:topic/foo"
   ]
  }
 ]
}
```

证书策略示例

对于在 Amazon IoT Core 注册表中注册的设备，以下策略授予权限以使用与事物名称匹配的客户端 ID 连接到 Amazon IoT Core，以及发布到名称等于设备用于对自身进行身份验证的证书的 `certificateId` 的主题：

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:CertificateId}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

对于未在 Amazon IoT Core 注册表中注册的设备，以下策略授予权限以使用客户端 ID `client1`、`client2` 和 `client3` 连接到 Amazon IoT Core，以及发布到名称等于设备用于对自身进行身份验证的证书的 `certificateId` 的主题：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${iot:CertificateId}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

对于在 Amazon IoT Core 注册表中注册的设备，以下策略授予权限以使用与事物名称匹配的客户端 ID 连接到 Amazon IoT Core，以及发布到名称等于设备用于对自身进行身份验证的证书的使用者 `CommonName` 字段的主题：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        }
    ]
}
```

Note

在此示例中，证书的使用者公用名用作主题标识符，并假设使用者公用名对于每个已注册的证书都是唯一的。如果证书在多个设备之间共享，则共享此证书的所有设备的使用者公用名将相同，因而允许从多个设备向同一主题发布权限（不推荐）。

对于未在 Amazon IoT Core 注册表中注册的设备，以下策略授予权限以使用客户端 ID `client1`、`client2` 和 `client3` 连接到 Amazon IoT Core，以及发布到名称等于设备用于对自身进行身份验证的证书的使用者 `CommonName` 字段的主题：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/
${iot:Certificate.Subject.CommonName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        }
    ]
}
```

Note

在此示例中，证书的使用者公用名用作主题标识符，并假设使用者公用名对于每个已注册的证书都是唯一的。如果证书在多个设备之间共享，则共享此证书的所有设备的使用者公用名将相同，因而允许从多个设备向同一主题发布权限（不推荐）。

对于在 Amazon IoT Core 注册表中注册的设备，以下策略授予权限以使用与事物名称匹配的客户端 ID 连接到 Amazon IoT Core，以及在用于对设备进行身份验证的证书将其 `Subject.CommonName.2` 字段设置为 `Administrator` 时发布到名称前缀为 `admin/` 的主题：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
```

```
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
                }
            }
        }
    ]
}
```

对于未在 Amazon IoT Core 注册表中注册的设备，以下策略授予权限以使用客户端 ID `client1`、`client2` 和 `client3` 连接到 Amazon IoT Core，以及在用于对设备进行身份验证的证书将其 `Subject.CommonName.2` 字段设置为 `Administrator` 时发布到名称前缀为 `admin/` 的主题：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/*"],
            "Condition": {
                "StringEquals": {
                    "iot:Certificate.Subject.CommonName.2": "Administrator"
                }
            }
        }
    ]
}
```

对于在 Amazon IoT Core 注册表中注册的设备，在用于对设备进行身份验证的证书将其任一 `Subject.CommonName` 字段设置为 `Administrator` 时，以下策略允许设备使用其事物名称在包含 `admin/` 并且后跟 `ThingName` 的特定主题上发布：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
```

```
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin/
${iot:Connection.Thing.ThingName}"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
                }
            }
        }
    ]
}
```

对于未在 Amazon IoT Core 注册表中注册的设备，以下策略授予权限以使用客户端 ID `client1`、`client2` 和 `client3` 连接到 Amazon IoT Core，以及在用于对设备进行身份验证的证书将其任一 `Subject.CommonName` 字段设置为 `Administrator` 时发布到主题 `admin`：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1",
                "arn:aws:iot:us-east-1:123456789012:client/client2",
                "arn:aws:iot:us-east-1:123456789012:client/client3"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/admin"],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iot:Certificate.Subject.CommonName.List": "Administrator"
                }
            }
        }
    ]
}
```

事物策略示例

如果用于在 Amazon IoT Core 中进行身份验证的证书附加到要评估策略的事物，则以下策略允许设备进行连接：

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

```
            "Action":["iot:Connect"],
            "Resource":[ "*" ],
            "Condition": {
                "Bool": {
                    "iot:Connection.Thing.IsAttached": ["true"]
                }
            }
        }
    ]
}
```

如果证书附加到具有特定事物类型的事物，并且该事物具有值为 attributeValue 的 attributeName 属性，则以下策略允许设备进行发布。有关事物策略变量的更多信息，请参阅事物策略变量 (p. 304)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/device/stats",
      "Condition": {
        "StringEquals": {
          "iot:Connection.Thing.Attributes[attributeName]": "attributeValue",
          "iot:Connection.Thing.ThingTypeName": "Thing_Type_Name"
        },
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
        }
      }
    }
  ]
}
```

以下策略允许设备发布到以事物的属性开头的主题。如果设备证书与事物不关联，则将无法解析此变量，并将导致访问被拒绝错误。有关事物策略变量的更多信息，请参阅事物策略变量 (p. 304)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.Attributes[attributeName]}/*"
    }
  ]
}
```

基本任务策略示例

此示例说明了任务目标所需的策略状态，任务目标是接收任务请求并与 Amazon IoT 通信任务执行状态的单个设备。

将 *us-west-2:57 EXAMPLE833* 替换为您的、冒号字符 (:) 和 12 位Amazon Web Services 账户数字，然后*uniqueThingName*替换为表示中设备的事物资源名称Amazon IoT。Amazon Web Services 区域

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:client/uniqueThingName"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/job/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/events/jobExecution/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/events/jobExecution/*",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/$aws/things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic",
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName/jobs/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeJobExecution",
        "iot:GetPendingJobExecutions",
        "iot:StartNextPendingJobExecution",
        "iot:UpdateJobExecution"
      ],
      "Resource": [
        "arn:aws:iot:us-west-2:57EXAMPLE833:topic/$aws/things/uniqueThingName"
      ]
    }
  ]
}
```

使用 Amazon Cognito 身份的授权

存在两种类型的 Amazon Cognito 身份：经过身份验证的身份和未经身份验证的身份。当您的应用程序支持未经身份验证的 Amazon Cognito 身份时，将不会执行身份验证，因此您不知道用户的身份。

未经身份验证的身份：对于未经身份验证的 Amazon Cognito 身份，您可以通过将 IAM 角色附加到未经身份验证的身份池来授予权限。我们建议您仅授予对希望可供未知用户使用的那些资源的访问权限。

> **Important**
>
> 对于连接到 Amazon IoT Core 的未经身份验证的 Amazon Cognito 用户，我们建议您在 IAM 策略中授予对非常有限的资源的访问权限。

经过身份验证的身份：对于经过身份验证的 Amazon Cognito 身份，您需要在两个位置指定权限：

- 将 IAM 策略附加到经过身份验证的 Amazon Cognito 身份池和
- 将 Amazon IoT Core 策略附加到 Amazon Cognito 身份（经过身份验证的用户）。

将未经身份验证和经过身份验证的 Amazon Cognito 用户连接到 Amazon IoT Core 的策略示例

以下示例显示了 Amazon Cognito 身份的 IAM 策略和 IoT 策略中的权限。经过身份验证的用户希望发布到设备特定的主题（例如 device/DEVICE_ID/status）。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/Client_ID"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/device/Device_ID/status"
            ]
        }
    ]
}
```

以下示例显示了 Amazon Cognito 未经身份验证角色的 IAM 策略中的权限。未经身份验证的用户希望发布到不需要身份验证的非设备特定主题。

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

```
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/non_device_specific_topic"
            ]
        }
    ]
}
```

GitHub 例子

以下 Web 应用程序示例说明 GitHub 了如何将针对已验证用户的策略附件合并到用户注册和身份验证过程中。

- 使用 Amazon Amplify 和 Amazon IoT Device SDK for JavaScript，MQTT 发布/订阅反应 web 应用程序
- 使用 Amazon Amplify、Amazon IoT Device SDK for JavaScript 和 Lambda 函数，MQTT 发布/订阅反应 web 应用程序

Amplify 是一组工具和服务，可以帮助您使用 Amazon 服务构建 Web 应用程序和移动应用程序。有关 Amplify 的更多信息，请参阅 Amplify Framework Documentation。

这两个示例都执行了以下步骤。

1. 用户注册账户时，应用程序会创建 Amazon Cognito 用户池和身份。
2. 用户进行身份验证时，应用程序将创建策略并将其附加到身份。这授予用户发布和订阅的权限。
3. 用户可以使用应用程序发布和订阅 MQTT 主题。

第一个示例直接在身份验证操作中使用 `AttachPolicy` API 操作。以下示例演示如何在使用 Amplify 和 Amazon IoT Device SDK for JavaScript 的反应 Web 应用程序中实现此 API。

```
function attachPolicy(id, policyName) {
    var Iot = new AWS.Iot({region: AWSConfiguration.region, apiVersion:
 AWSConfiguration.apiVersion, endpoint: AWSConfiguration.endpoint});
    var params = {policyName: policyName, target: id};

    console.log("Attach IoT Policy: " + policyName + " with cognito identity id: " + id);
    Iot.attachPolicy(params, function(err, data) {
         if (err) {
               if (err.code !== 'ResourceAlreadyExistsException') {
                  console.log(err);
               }
          }
         else  {
            console.log("Successfully attached policy with the identity", data);
         }
     });
}
```

此代码出现在 AuthDisplay.js 文件中。

第二个示例在 Lambda 函数中实现 `AttachPolicy` API 操作。以下示例说明 Lambda 如何使用此 API 调用。

```
iot.attachPolicy(params, function(err, data) {
     if (err) {
           if (err.code !== 'ResourceAlreadyExistsException') {
              console.log(err);
              res.json({error: err, url: req.url, body: req.body});
           }
      }
     else  {
        console.log(data);
        res.json({success: 'Create and attach policy call succeed!', url: req.url, body:
 req.body});
     }
 });
```

此代码出现在app.js文件的 `iot.GetPolicy` 函数中。

Note

当您使用通过 Amazon Cognito Identity 池获得的 `context.cognito_identity_id` 凭证调用函数时，Lambda 函数中的上下文对象包含 Amazon 的值。有关更多信息，请参阅下列内容。

- Node.js 中的 Amazon Lambda 上下文对象
- Python 中的 Amazon Lambda 上下文对象
- Ruby 中的 Amazon Lambda 上下文对象
- Java 中的 Amazon Lambda 上下文对象
- Amazon Lambda Go 中的 上下文对象
- Amazon Lambda C# 中的 上下文对象
- Amazon Lambda中的上下文对象 PowerShell

使用 Amazon IoT Core 凭证提供程序授予直接调用Amazon服务的权限

设备可使用 X.509 证书通过 TLS 双向身份验证协议连接到 Amazon IoT Core。其它 Amazon 服务不支持基于证书的身份验证，但可使用 Amazon Signature Version 4 格式的 Amazon 凭证调用这些服务。Signature Version 4 算法通常需要调用方具有访问密钥 ID 和秘密访问密钥。Amazon IoT Core 的凭证提供程序允许您使用内置的 X.509 证书作为唯一设备身份来对Amazon请求进行身份验证。这样就不再需要将访问密钥 ID 和私有访问密钥存储在您的设备上。

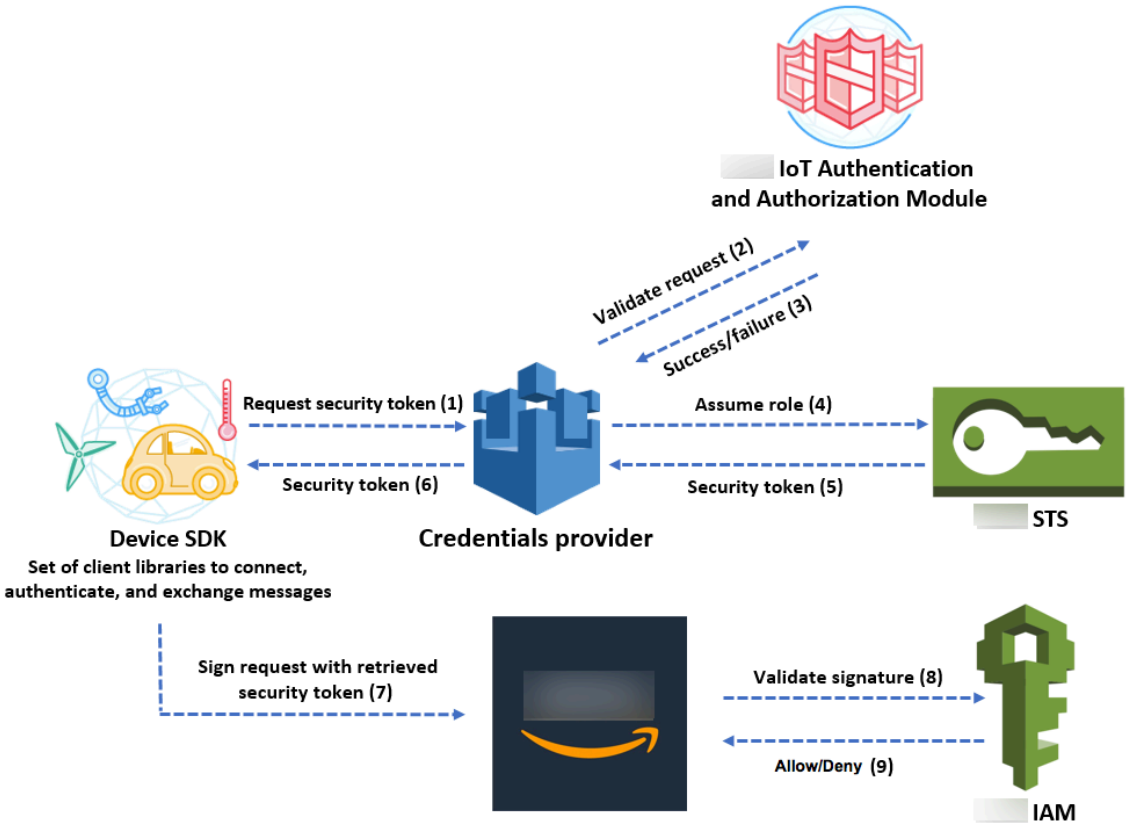
凭证提供程序使用 X.509 证书对调用方进行身份验证并颁发具有有限权限的临时安全令牌。令牌可用于对任何Amazon请求进行签名和身份验证。以这种方式对您的Amazon请求进行身份验证需要您创建并配置 Amazon Identity and Access Management (IAM) 角色并将适当的 IAM policy 附加到该角色，以便凭证提供程序可以代表您代入该角色。有关 Amazon IoT Core 和 IAM 的更多信息，请参阅 适用于 Amazon IoT 的 Identity and Access Management (p. 348)。

Amazon IoT 要求设备将服务器名称指示 (SNI) 扩展发送到传输层安全性 (TLS) 协议，并在 `host_name` 字段中提供完整的终端节点地址。`host_name` 字段必须包含您调用的终端节点，并且必须是：

- `aws iot describe-endpoint --endpoint-type iot:CredentialProvider` 返回的 `endpointAddress`。

没有正确 `host_name` 值的设备尝试的连接将失败。

下图说明了凭证提供程序工作流程。



1. Amazon IoT Core 设备向凭证提供程序发出 HTTPS 请求以获取安全令牌。该请求包括用于身份验证的设备 X.509 证书。
2. 凭证提供程序将该请求转发给 Amazon IoT Core 身份验证和授权模块来验证证书并验证设备是否有权请求安全令牌。
3. 如果证书有效并有权请求安全令牌，Amazon IoT Core 身份验证和授权模块将返回指示成功的消息。否则，它会向设备发送异常。
4. 成功验证证书之后，凭证提供程序将调用 Amazon Security Token Service (Amazon STS) 来代入您为它创建的 IAM 角色。
5. Amazon STS 将具有有限权限的临时安全令牌返回给凭证提供程序。
6. 凭证提供程序将该安全令牌返回给设备。
7. 设备使用该安全令牌通过Amazon Signature Version 4 对Amazon请求进行签名。
8. 请求的服务调用 IAM 来验证签名并根据附加到您为凭证提供程序创建的 IAM 角色的访问策略授权请求。
9. 如果 IAM 成功验证签名并授权请求，则请求成功。否则，IAM 将发送异常。

下一节介绍如何使用证书来获取安全令牌。编写此内容时，假定您已注册了设备并为它创建并激活了您自己的证书。

如何使用证书来获取安全令牌

1. 配置凭证提供程序代表您的设备代入的 IAM 角色。将以下信任策略附加到该角色。

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

```
        "Effect": "Allow",
        "Principal": {"Service": "credentials.iot.amazonaws.com"},
        "Action": "sts:AssumeRole"
    }
}
```

对于要调用的每个Amazon服务，将访问策略附加到该角色。凭证提供程序支持以下策略变量：

- `credentials-iot:ThingName`
- `credentials-iot:ThingTypeName`
- `credentials-iot:AwsCertificateId`

当设备在向Amazon服务发出的请求中提供了事物名称时，凭证提供程序会将 `credentials-iot:ThingName` 和 `credentials-iot:ThingTypeName` 作为上下文变量添加到安全令牌。如果设备没有在请求中提供事物名称，则凭证提供程序将提供 `credentials-iot:AwsCertificateId` 作为上下文变量。您将事物名称作为 `x-amzn-iot-thingname` HTTP 请求标头的值进行传递。

这三个变量仅适用于 IAM policy，而不适用于 Amazon IoT Core 策略。

2. 确保执行下一步（创建角色别名）的用户有权将新创建的角色传递给 Amazon IoT Core。以下策略将 `iam:GetRole` 和 `iam:PassRole` 权限都提供给了Amazon用户。`iam:GetRole` 权限可让用户获取有关您刚创建的角色的信息。`iam:PassRole` 权限可让用户将角色传递给其它 Amazon 服务。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::your Amazon Web Services ## id:role/your role name"
    }
}
```

3. 创建 Amazon IoT Core 角色别名。打算直接调用Amazon服务的设备必须知道在连接到 Amazon IoT Core 时要使用的角色 ARN。对角色 ARN 进行硬编码并不是一个很好的解决方案，因为它需要您在角色 ARN 发生更改时更新设备。更好的解决方案是使用 `CreateRoleAlias` API 创建一个指向角色 ARN 的角色别名。如果角色 ARN 发生更改，您只需更新角色别名。无需在设备上进行任何更改。此 API 接受以下参数：

   `roleAlias`

   必需。一个标识角色别名的任意字符串。它充当角色别名数据模型中的主键。它包含 1-128 个字符，并且必须仅包含字母数字字符以及 =、@ 和 - 符号。允许大写和小写字母字符。

   `roleArn`

   必需。角色别名所指向的角色的 ARN。

   `credentialDurationSeconds`

   可选。凭证有效的时间长度 (以秒为单位)。最小值为 900 秒（15 分钟）。最大值为 43200 秒（12 小时）。默认值为 3600 秒（1 小时）。

   Note

   Amazon IoT Core 凭证提供程序可以颁发凭证，其最大生命周期为 43200 秒（12 小时）。凭证的有效期长达 12 小时，有助于通过将凭证缓存更长时间来减少对凭证提供程序的调用次数。

`credentialDurationSeconds` 值必须小于或等于角色别名引用的 IAM 角色的最长会话持续时间。

相关此 API 的更多信息，请参阅 CreateRoleAlias。

4. 将策略附加到设备证书。附加到设备证书的策略必须向设备授予代入角色的权限。您可以通过授予对角色别名执行 `iot:AssumeRoleWithCertificate` 操作的权限来做到这一点，如以下示例所示。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"iot:AssumeRoleWithCertificate",
         "Resource":"arn:aws:iot:your region:your_aws_account_id:rolealias/your role
 alias"
      }
   ]
}
```

5. 向凭证提供程序发出 HTTPS 请求来获取安全令牌。提供以下信息：

   - Certificate：由于这是使用 TLS 双向身份验证的 HTTP 请求，因此，您在发出请求时必须向客户端提供证书和私有密钥。使用您向 Amazon IoT Core 注册您的证书时所使用的相同证书和私有密钥。

     要确保您的设备与 Amazon IoT Core（而不是模拟它的服务）进行通信，请参阅服务器身份验证，单击链接以下载相应的 CA 证书，然后将这些证书复制到您的设备中。
   - RoleAlias：您为证书提供者创建的角色别名的名称。
   - ThingName：您在注册事物时创建Amazon IoT Core的事物名称。这作为 `x-amzn-iot-thingname` HTTP 标头的值进行传递。仅当您使用事物属性作为 Amazon IoT Core 或 IAM policy 中的策略变量时，此值才是必需的。

     Note

     您提供的`x-amzn-iot-thingname`必须与分配给证书的Amazon IoT事物资源相匹配。ThingName如果不匹配，则返回 403 错误。

   在 Amazon CLI 中运行以下命令以获取您Amazon Web Services 账户的凭证提供程序终端节点。相关此 API 的更多信息，请参阅 DescribeEndpoint。

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

以下 JSON 对象是 describe-endpoint 命令的示例输出。它包含您用于请求安全令牌的 `endpointAddress`。

```
{
    "endpointAddress": "your_aws_account_specific_prefix.credentials.iot.your
 region.amazonaws.com"
}
```

使用终端节点向凭证提供程序发出 HTTPS 请求以返回安全令牌。以下示例命令使用 `curl`，但您可以使用任何 HTTP 客户端。

```
curl --cert your certificate --key your device certificate key pair -H "x-amzn-iot-
thingname: your thing name" --cacert AmazonRootCA1.pem https://your endpoint /role-
aliases/your role alias/credentials
```

此命令返回包含 accessKeyId、secretAccessKey、sessionToken 和过期的安全令牌对象。以下 JSON 对象是 curl 命令的示例输出。

```
   {"credentials":{"accessKeyId":"access key","secretAccessKey":"secret access
key","sessionToken":"session token","expiration":"2018-01-18T09:18:06Z"}}
```

然后，您可以使用 accessKeyId、secretAccessKey 和 sessionToken 值来对向Amazon服务发出的请求进行签名。有关演 end-to-end 示，请参阅Amazon安全博客中的如何通过使用Amazon凭证提供程序使设备中不再需要硬编码Amazon IoT凭证。

通过 IAM 跨账户访问

Amazon IoT Core 允许您支持委托人订阅并非委托人拥有的Amazon Web Services 账户中定义的主题或向该主题发布消息。您可以通过创建 IAM policy 和 IAM 角色并将策略附加到角色来配置跨账户访问。

首先，创建一个客户托管 IAM policy（如创建 IAM policy 中所述），就像您在 Amazon Web Services 账户中为其他用户和证书创建策略一样。

对于在 Amazon IoT Core 注册表中注册的设备，以下策略为设备授予权限以使用与设备的事物名称匹配的客户端 ID 连接到 Amazon IoT Core 以及发布到 my/topic/*thing-name* ，其中 *thing-name* 是设备的事物名称：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/my/topic/
${iot:Connection.Thing.ThingName}"],
        }
    ]
}
```

对于未在 Amazon IoT Core 注册表中注册的设备，以下策略为设备授予权限以使用在您的账户 (123456789012) 的 Amazon IoT Core 注册表中注册的事物名称 client1 连接到 Amazon IoT Core，以及发布到名称前缀为 my/topic/ 的客户端 ID 特定的主题：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
```

```
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/client1"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
            ]
        }
    ]
}
```

接下来，按照创建向 IAM 用户委派权限的角色中的步骤操作。输入要与之共享访问权限的Amazon Web Services 账户的账户 ID。接下来是最后一步，请将您刚刚创建的策略附加到角色。如果您稍后需要修改要向其授予权限的Amazon ID，可使用以下信任策略格式执行操作：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam:us-east-1:567890123456:user/MyUser"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Amazon IoT Core 中的数据保护

Amazon责任共担模式适用于 Amazon IoT Core 中的数据保护。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 Amazon Web Services 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon IAM Identity Center (successor to Amazon Single Sign-On) 或 Amazon Identity and Access Management (IAM) 设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用 Amazon 加密解决方案以及 Amazon Web Services 中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如 Name（名称）字段）。这包括使用控制台、API、Amazon CLI 或 Amazon SDK 处理 Amazon IoT 或其他 Amazon Web Services时。您在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

有关数据保护的更多信息，请参阅Amazon安全性博客上的Amazon责任共担模式和 GDPR 博客文章。

Amazon IoT 设备收集数据，对该数据进行一些处理，然后将数据发送到另一个 Web 服务。您可以选择在设备上将一些数据存储一段较短的时间。您有责任为静态数据提供任何数据保护。当您的设备向 Amazon IoT 发送数据时，它会通过 TLS 连接完成此操作，如本节后面所述。Amazon IoT 设备可以将数据发送到任何 Amazon 服务。有关每项服务的数据安全性的更多信息，请参阅该服务的文档。 Amazon IoT可以配置为将日志写入 CloudWatch 日志并将Amazon IoT API 调用记录到Amazon CloudTrail。有关这些服务的数据安全性的更多信息，请参阅适用于 Amazon 的身份验证和访问控制 CloudWatch和使用Amazon KMS 托管密钥加密 CloudTrail 日志文件。

Amazon IoT 中的数据加密

预设情况下，传输和静态中的 Amazon IoT 数据均加密。传输中的数据使用 TLS 进行加密 (p. 346)，静态数据使用 Amazon 拥有的密钥进行加密。Amazon IoT 目前不支持来自 Amazon Key Management Service (Amazon KMS) 客户托管的 Amazon KMS keys 客户主密钥 (CMK)；但是，Device Advisor 和 Amazon IoT Wireless 仅使用 Amazon 拥有的密钥 来加密客户数据。

Amazon IoT 中的传输安全

Amazon IoT 消息代理和 Device Shadow 服务可以使用 TLS 版本 1.2 对传输中的所有通信进行加密。TLS 用于确保受支持的应用程序协议（MQTT、HTTP 和 WebSocket）的保密性Amazon IoT。TLS 支持适用于许多编程语言和操作系统。Amazon内的数据是由特定的Amazon服务加密的。有关其它 Amazon 服务上的数据加密的更多信息，请参阅该服务的安全文档。

对于 MQTT，TLS 可将设备与代理之间的连接加密。Amazon IoT 使用 TLS 客户端身份验证来识别设备。对于 HTTP，TLS 可将设备与代理之间的连接加密。身份验证交由给Amazon Signature Version 4 执行。

Amazon IoT 要求设备将服务器名称指示 (SNI) 扩展发送到传输层安全性 (TLS) 协议，并在 `host_name` 字段中提供完整的终端节点地址。`host_name` 字段必须包含您调用的终端节点，并且必须是：

- `aws iot describe-endpoint --endpoint-type iot:Data-ATS` 返回的 `endpointAddress`

  or
- `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"` 返回的 `domainName`

CloudWatch 如果身份验证类型为自定义身份验证，没有正确`host_name`值的设备发起的连接Amazon IoT将失败，会将失败录入。

Amazon IoT不支持 SessionTicket TLS 扩展。

LoRaWAN 无线设备的传输安全

LoRaWAN 设备遵循 WA LoRaN™ 安全：Gemalto、Actility 和 Semtech 为 Al LoRa liance 准备的白皮书™ 中所述的安全实践。

有关 LoRa WAN 设备传输安全性的更多信息，请参阅适用于 LoRaWAN 的 Amazon IoT Core 数据安全性 (p. 1201)。

TLS 密码包支持

Amazon IoT 支持以下密码包：

- ECDHE-ECDSA-AES128-GCM-SHA256（推荐）
- ECDHE-RSA-AES128-GCM-SHA256（推荐）
- ECDHE-ECDSA-AES128-SHA256

- ECDHE-RSA-AES128-SHA256
- ECDHE-ECDSA-AES128-SHA
- ECDHE-RSA-AES128-SHA
- ECDHE-ECDSA-AES256- GCM-SHA384
- ECDHE-RSA-AES256- GCM-SHA384
- ECDHE-ECDSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA
- ECDHE-ECDSA-AES256-SHA
- AES128-GCM-SHA256
- AES128-SHA256
- AES128-SHA
- AES256-GCM-SHA384
- AES256-SHA256
- AES256-SHA

连接安全性

Amazon IoT 消息代理和 Device Shadow 服务依赖于使用加密和 TLS 1.2 进行通信。Amazon IoT 支持以下 TLS 密码包：

- ECDHE-ECDSA-AES128-GCM-SHA256（推荐）
- ECDHE-RSA-AES128-GCM-SHA256（推荐）
- ECDHE-ECDSA-AES128-SHA256
- ECDHE-RSA-AES128-SHA256
- ECDHE-ECDSA-AES128-SHA
- ECDHE-RSA-AES128-SHA
- ECDHE-ECDSA-AES256- GCM-SHA384
- ECDHE-RSA-AES256- GCM-SHA384
- ECDHE-ECDSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA384
- ECDHE-RSA-AES256-SHA
- ECDHE-ECDSA-AES256-SHA
- AES128-GCM-SHA256
- AES128-SHA256
- AES128-SHA
- AES256-GCM-SHA384
- AES256-SHA256
- AES256-SHA

Amazon IoT 中的数据加密

数据保护指在数据传输（发往和离开 Amazon IoT 时）和处于静态（存储在设备上或由其它 Amazon 服务存储）期间保护数据。发送到的所有数据都使用 MQTT、HTTPS 和 WebSocket 协议通过 TLS 连接发送，因此，预设情况下，它在传输过程中Amazon IoT是安全的。 Amazon IoT设备收集数据，然后将其发送到其他Amazon服务进行进一步处理。有关其它 Amazon 服务上的数据加密的更多信息，请参阅该服务的安全文档。

FreeRTOS 提供了一个 PKCS#11 库，用于提取密钥存储、访问加密对象和管理会话。您有责任使用此库对设备上的数据进行静态加密。有关更多信息，请参阅 FreeRTOS 公有密钥加密标准 (PKCS) #11 库。

Device Advisor

传输中加密

通过 Device Advisor 收发的所有数据都会在传输过程中加密。使用 Device Advisor API 时发送到服务和从服务发出的所有数据都使用 Signature Version 4 进行加密。有关 Amazon API 请求如何签署的更多信息，请参阅签署 Amazon API 请求。从测试设备发送到 Device Advisor 测试终端节点的所有数据都通过 TLS 连接发送，因此预设情况下其在传输过程中是安全的。

Amazon IoT 中的密钥管理

由于与 Amazon IoT 的所有连接都是使用 TLS 建立的，因此，初始 TLS 连接不需要客户端加密密钥。

设备必须使用 X.509 证书或 Amazon Cognito Identity 进行身份验证。您可以让 Amazon IoT 为您生成一个证书，在此情况下，它将生成一个公有/私有密钥对。如果您使用的是 Amazon IoT 控制台，则系统将提示您下载证书和密钥。如果您使用的是 `create-keys-and-certificate` CLI 命令，则该 CLI 命令将返回证书和密钥。您有责任将证书和私有密钥复制到您的设备上并确保其安全。

Amazon IoT 目前不支持来自 Amazon Key Management Service (Amazon KMS) 的客户托管的 Amazon KMS keys (KMS 密钥)；但是，Device Advisor 和 Amazon IoT Wireless 仅使用 Amazon 拥有的密钥 来加密客户数据。

Device Advisor

所有发送到 Device Advisor 的数据在使用 Amazon API 时会进行静态加密。Device Advisor 使用在 Amazon Key Management Service 中存储和管理的 KMS密钥对您的素有静态数据进行加密。Device Advisor 使用 Amazon 拥有的密钥 加密您的数据。有关 Amazon 拥有的密钥 的更多信息，请参阅 Amazon 拥有的密钥。

适用于 Amazon IoT 的 Identity and Access Management

Amazon Identity and Access Management (IAM) 是一项 Amazon Web Service，可以帮助管理员安全地控制对 Amazon 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）来使用 Amazon IoT 资源。IAM 是一项无需额外费用即可使用的 Amazon Web Service。

主题

- 受众 (p. 348)
- 使用 IAM 身份进行身份验证 (p. 349)
- 使用策略管理访问 (p. 350)
- Amazon IoT 如何与 IAM 协同工作 (p. 351)
- Amazon IoT 基于身份的策略示例 (p. 370)
- Amazon适用于 Amazon IoT 的托管策略 (p. 372)
- 对 Amazon IoT 身份和访问进行故障排除 (p. 382)

受众

使用 Amazon Identity and Access Management (IAM) 的方式因您可以在 Amazon IoT 中执行的操作而异。

服务用户 – 如果使用 Amazon IoT 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。当您使用更多 Amazon IoT 功能来完成工作时，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Amazon IoT 中的功能，请参阅 对 Amazon IoT 身份和访问进行故障排除 (p. 382)。

服务管理员 – 如果您在公司负责管理 Amazon IoT 资源，则您可能具有 Amazon IoT 的完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon IoT 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon IoT 搭配使用的更多信息，请参阅Amazon IoT 如何与 IAM 协同工作 (p. 351)。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon IoT 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon IoT 基于身份的策略示例，请参阅 Amazon IoT 基于身份的策略示例 (p. 370)。

使用 IAM 身份进行身份验证

在 Amazon IoT 中，身份可以是设备 (X.509) 证书、Amazon Cognito 身份或 IAM 用户或组。本主题仅讨论 IAM 身份。有关 Amazon IoT 支持的其他身份的更多信息，请参阅客户端身份验证 (p. 267)。

身份验证是您使用身份凭证登录 Amazon 的方法。您必须作为 Amazon Web Services 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 Amazon）。

如果您以编程方式访问 Amazon，则 Amazon 将提供软件开发工具包 (SDK) 和命令行界面（CLI），以便使用您的凭证对您的请求进行加密签名。如果您不使用 Amazon 工具，则必须自行对请求签名。有关使用推荐的方法自行对请求签名的更多信息，请参阅 Amazon 一般参考中的签名版本 4 签名流程。

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

Amazon Web Services 账户根用户

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

IAM 用户和组

IAM 用户是Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 对于需要长期凭证的使用场景定期轮换访问密钥。

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

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

IAM 角色

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

具有临时凭证的 IAM 角色在以下情况下很有用：

- Federated user access（联合用户访问）– 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 Amazon Web Services，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 Amazon Web Services使用其它 Amazon Web Services中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service（Amazon S3）中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 主体权限 – 当您使用 IAM 用户或角色在 Amazon 中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其它相关操作，请参阅服务授权参考中的 Amazon IoT 的操作、资源和条件键。
  - 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 Amazon Web Service 委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 Amazon Web Service关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 Amazon Web Services 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 Amazon 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

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

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

默认情况下，用户和角色没有权限。要向用户授予在其需要的资源上执行操作的权限，IAM 管理员可以创建 IAM policy。然后管理员可以向角色添加 IAM policy，并且用户可以代入角色。

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

基于身份的策略

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

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

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 Amazon Web Services。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 Amazon 托管式策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Simple Storage Service（Amazon S3）、Amazon WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其它策略类型

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

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

多个策略类型

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

Amazon IoT 如何与 IAM 协同工作

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

主题

- Amazon IoT 基于身份的策略 (p. 352)

- Amazon IoT 基于资源的策略 (p. 369)
- 基于 Amazon IoT 标签的授权 (p. 369)
- Amazon IoT IAM 角色 (p. 369)

Amazon IoT 基于身份的策略

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

操作

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

下表列出了 IAM IoT 操作、关联的 Amazon IoT API 以及操作处理的资源。

策略操作	Amazon IoT API	资源
物联网：AcceptCertificateTransfer	AcceptCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id* Note ARN 中指定的Amazon Web Services 账户必须是证书将传输到的账户。
物联网：AddThingToThingGroup	AddThingToThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：AssociateTargetsWithJob	AssociateTargetsWithJob	*
物联网：AttachPolicy	AttachPolicy	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* or arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：AttachPrincipalPolicy	AttachPrincipalPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：AttachSecurityProfile	AttachSecurityProfile	arn:aws:iot:*region*:*account-id*:securityprofile/*security-profile-name* arn:aws:iot:*region*:*account-id*:dimension/*dimension-name*
物联网：AttachThingPrincipal	AttachThingPrincipal	arn:aws:iot:*region*:*account-id*:cert/*cert-id*

策略操作	Amazon IoT API	资源
物联网：CancelCertificateTransfer	CancelCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id* Note ARN 中指定的Amazon Web Services 账户必须是证书将传输到的账户。
物联网：CancelJob	CancelJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：CancelJobExecution	CancelJobExecution	arn:aws:iot:*region*:*account-id*:job/*job-id* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：ClearDefaultAuthorizer	ClearDefaultAuthorizer	无
物联网：CreateAuthorizer	CreateAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name*
物联网：CreateCertificateFromCsr	CreateCertificateFromCsr	*
物联网：CreateDimension	CreateDimension	arn:aws:iot:*region*:*account-id*:dimension/*dimension-name*
物联网：CreateJob	CreateJob	arn:aws:iot:*region*:*account-id*:job/*job-id* arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* arn:aws:iot:*region*:*account-id*:thing/*thing-name* arn:aws:iot:*region*:*account-id*:jobtemplate/*job-template-id*
物联网：CreateJobTemplate	CreateJobTemplate	arn:aws:iot:*region*:*account-id*:job/*job-id* arn:aws:iot:*region*:*account-id*:jobtemplate/*job-template-id*
物联网：CreateKeysAndCertificate	CreateKeysAndCertificate	*
物联网：CreatePolicy	CreatePolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：CreatePolicyVersion	CreatePolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name* Note 这必须是 Amazon IoT 策略，而不是 IAM policy。
物联网：CreateRoleAlias	CreateRoleAlias	(参数：roleAlias) arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*

策略操作	Amazon IoT API	资源
物联网：CreateSecurityProfile	CreateSecurityProfile	arn:aws:iot:*region*:*account-id*:securityprofile/*security-profile-name* arn:aws:iot:*region*:*account-id*:dimension/*dimension-name*
物联网：CreateThing	CreateThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：CreateThingGroup	CreateThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* 针对要创建的组和父组 (如果使用)
物联网：CreateThingType	CreateThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
物联网：CreateTopicRule	CreateTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-name*
iot:DeleteCACertificate	DeleteCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*
物联网：DeleteCertificate	DeleteCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：DeleteDimension	DeleteDimension	arn:aws:iot:*region*:*account-id*:dimension/*dimension-name*
物联网：DeleteJob	DeleteJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：DeleteJobTemplate	DeleteJobTemplate	arn:aws:iot:*region*:*account-id*:job/*job-template-id*
物联网：DeleteJobExecution	DeleteJobExecution	arn:aws:iot:*region*:*account-id*:job/*job-id* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：DeletePolicy	DeletePolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：DeleteRegistrationCode	DeleteRegistrationCode	*
物联网：DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
物联网：DeleteSecurityProfile	DeleteSecurityProfile	arn:aws:iot:*region*:*account-id*:securityprofile/*security-profile-name* arn:aws:iot:*region*:*account-id*:dimension/*dimension-name*

策略操作	Amazon IoT API	资源
物联网：DeleteThing	DeleteThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：DeleteThingGroup	DeleteThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：DeleteThingType	DeleteThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
物联网：DeleteTopicRule	DeleteTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网:Deletev2LoggingLevel	删除v2LoggingLevel	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：DeprecateThingType	DeprecateThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
物联网：DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name* (参数：authorizerName) 无
iot:DescribeCACertificate	DescribeCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*
物联网：DescribeCertificate	DescribeCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	无
物联网：DescribeEndpoint	DescribeEndpoint	*
物联网：DescribeEventConfigurations	DescribeEventConfigurations	无
物联网：DescribeIndex	DescribeIndex	arn:aws:iot:*region*:*account-id*:index/*index-name*
物联网：DescribeJob	DescribeJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：DescribeJobExecution	DescribeJobExecution	无
物联网：DescribeJobTemplate	DescribeJobTemplate	arn:aws:iot:*region*:*account-id*:job/*job-template-id*
物联网：DescribeRoleAlias	DescribeRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
物联网：DescribeThing	DescribeThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：DescribeThingGroup	DescribeThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*

策略操作	Amazon IoT API	资源
物联网：DescribeThingRegistrationTask	DescribeThingRegistrationTask	无
物联网：DescribeThingType	DescribeThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
物联网：DetachPolicy	DetachPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id* or arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：DetachPrincipalPolicy	DetachPrincipalPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：DetachSecurityProfile	DetachSecurityProfile	arn:aws:iot:*region*:*account-id*:securityprofile/*security-profile-name* arn:aws:iot:*region*:*account-id*:dimension/*dimension-name*
物联网：DetachThingPrincipal	DetachThingPrincipal	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：DisableTopicRule	DisableTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：EnableTopicRule	EnableTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：GetEffectivePolicies	GetEffectivePolicies	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：GetIndexingConfiguration	GetIndexingConfiguration	无
物联网：GetJobDocument	GetJobDocument	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：GetLoggingOptions	GetLoggingOptions	*
物联网：GetPolicy	GetPolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：GetPolicyVersion	GetPolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：GetRegistrationCode	GetRegistrationCode	*
物联网：GetTopicRule	GetTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*

策略操作	Amazon IoT API	资源
物联网：ListAttachedPolicies	ListAttachedPolicies	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* or arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：ListAuthorizers	ListAuthorizers	无
iot:ListCACertificates	ListCACertificates	*
物联网：ListCertificates	ListCertificates	*
物联网：ListCertificatesByCA	ListCertificatesBy加州	*
物联网：ListIndices	ListIndices	无
物联网：ListJobExecutionsForJob	ListJobExecutionsForJob	无
物联网：ListJobExecutionsForThing	ListJobExecutionsForThing	无
物联网：ListJobs	ListJobs	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* 如果使用了 thingGroupName 参数
物联网：ListJobTemplates	ListJobs	无
物联网：ListOutgoingCertificates	ListOutgoingCertificates	*
物联网：ListPolicies	ListPolicies	*
物联网：ListPolicyPrincipals	ListPolicyPrincipals	*
物联网：ListPolicyVersions	ListPolicyVersions	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：ListPrincipalPolicies	ListPrincipalPolicies	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：ListPrincipalThings	ListPrincipalThings	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：ListRoleAliases	ListRoleAliases	无
物联网：ListTargetsForPolicy	ListTargetsForPolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*

策略操作	Amazon IoT API	资源
物联网：ListThingGroups	ListThingGroups	无
物联网：ListThingGroupsForThing	ListThingGroupsForThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：ListThingPrincipals	ListThingPrincipals	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：ListThingRegistrationTaskReports	ListThingRegistrationTaskReports	无
物联网：ListThingRegistrationTasks	ListThingRegistrationTasks	无
物联网：ListThingTypes	ListThingTypes	*
物联网：ListThings	ListThings	*
物联网：ListThingsInThingGroup	ListThingsInThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：ListTopicRules	ListTopicRules	*
物联网:Listv2LoggingLevels	ListV2LoggingLevels	无
iot:RegisterCACertificate	RegisterCACertificate	*
物联网：RegisterCertificate	RegisterCertificate	*
物联网：RegisterThing	RegisterThing	无
物联网：RejectCertificateTransfer	RejectCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：RemoveThingFromThingGroup	RemoveThingFromThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：ReplaceTopicRule	ReplaceTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：SearchIndex	SearchIndex	arn:aws:iot:*region*:*account-id*:index/*index-id*
物联网：SetDefaultAuthorizer	SetDefaultAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name*
物联网：SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：SetLoggingOptions	SetLoggingOptions	arn:aws:iot:*region*:*account-id*:role/*role-name*

策略操作	Amazon IoT API	资源
物联网:setv2LoggingLevel	setV2LoggingLevel	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网:setv2LoggingOptions	setV2LoggingOptions	arn:aws:iot:*region*:*account-id*:role/*role-name*
物联网：StartThingRegistrationTask	StartThingRegistrationTask	无
物联网：StopThingRegistrationTask	StopThingRegistrationTask	无
物联网：TestAuthorization	TestAuthorization	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：TestInvokeAuthorizer	TestInvokeAuthorizer	无
物联网：TransferCertificate	TransferCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot:*region*:*account-id*:authorizerfunction/*authorizer-function-name*
iot:UpdateCACertificate	UpdateCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*
物联网：UpdateCertificate	UpdateCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：UpdateDimension	UpdateDimension	arn:aws:iot:*region*:*account-id*:dimension/*dimension-name*
物联网：UpdateEventConfigurations	UpdateEventConfigurations	无
物联网：UpdateIndexingConfiguration	UpdateIndexingConfiguration	无
物联网：UpdateRoleAlias	UpdateRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
物联网：UpdateSecurityProfile	UpdateSecurityProfile	arn:aws:iot:*region*:*account-id*:securityprofile/*security-profile-name* arn:aws:iot:*region*:*account-id*:dimension/*dimension-name*
物联网：UpdateThing	UpdateThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：UpdateThingGroup	UpdateThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：UpdateThingGroupsForThing	UpdateThingGroupsForThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name* arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*

Amazon IoT 中的策略操作在操作前使用以下前缀：`iot:`。例如，要授予某人权限以使用 `ListThings` API 列出其Amazon Web Services 账户中已注册的所有 IoT 事物，请将 `iot:ListThings` 操作纳入其策略中。策略语句必须包括 `Action` 或 `NotAction` 元素。Amazon IoT 定义了自己的一组操作，这些操作描述了可使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

```
"Action": [
      "ec2:action1",
      "ec2:action2"
```

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

```
"Action": "iot:Describe*"
```

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

Device Advisor 操作

下表列出了 IAM IoT Device Advisor 操作、关联的 Amazon IoT Device Advisor API 以及操作处理的资源。

策略操作	Amazon IoT API	资源
物联网设备顾问：CreateSuiteDefinition	CreateSuiteDefinition	无
物联网设备顾问：DeleteSuiteDefinition	DeleteSuiteDefinition	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-definition-id*
物联网设备顾问：GetSuiteDefinition	GetSuiteDefinition	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-definition-id*
物联网设备顾问：GetSuiteRun	GetSuiteRun	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-run-id*
物联网设备顾问：GetSuiteRunReport	GetSuiteRunReport	arn:aws:iotdeviceadvisor:*region*:*account-id*:suiterun/*suite-definition-id*/*suite-run-id*
物联网设备顾问：ListSuiteDefinitions	ListSuiteDefinitions	无
物联网设备顾问：ListSuiteRuns	ListSuiteRuns	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-definition-id*
物联网设备顾问：ListTagsForResource	ListTagsForResource	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-definition-id* arn:aws:iotdeviceadvisor:*region*:*account-id*:suiterun/suite-definition-id/*suite-run-id*
物联网设备顾问：StartSuiteRun	StartSuiteRun	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-definition-id*
物联网设备顾问：TagResource	TagResource	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-definition-id* arn:aws:iotdeviceadvisor:*region*:*account-id*:suiterun/suite-definition-id/*suite-run-id*

策略操作	Amazon IoT API	资源
物联网设备顾问：UntagResource	UntagResource	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-definition-id* arn:aws:iotdeviceadvisor:*region*:*account-id*:suiterun/suite-definition-id/*suite-run-id*
物联网设备顾问：UpdateSuiteDefinition	UpdateSuiteDefinition	arn:aws:iotdeviceadvisor:*region*:*account-id*:suitedefinition/*suite-definition-id*
物联网设备顾问：StopSuiteRun	StopSuiteRun	arn:aws:iotdeviceadvisor:*region*:*account-id*:suiterun/suite-definition-id/*suite-run-id*

Amazon IoT Device Advisor 中的策略操作在操作前使用以下前缀：`iotdeviceadvisor:`。例如，要授予某人权限以Amazon Web Services 账户使用 ListSuiteDefinitions API 列出其中已注册的所有套件定义，您应将`iotdeviceadvisor:ListSuiteDefinitions`操作纳入其策略中。

资源

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

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

Amazon IoT 资源

策略操作	Amazon IoT API	资源
物联网：AcceptCertificateTransfer	AcceptCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id* Note ARN 中指定的Amazon Web Services 账户必须是证书将传输到的账户。
物联网：AddThingToThingGroup	AddThingToThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：AssociateTargetsWithJob	AssociateTargetsWithJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：AttachPolicy	AttachPolicy	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* or arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：AttachPrincipalPolicy	AttachPrincipalPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id*

策略操作	Amazon IoT API	资源
物联网：AttachThingPrincipal	AttachThingPrincipal	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：CancelCertificateTransfer	CancelCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id* **Note** ARN 中指定的Amazon Web Services 账户必须是证书将传输到的账户。
物联网：CancelJob	CancelJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：CancelJobExecution	CancelJobExecution	arn:aws:iot:*region*:*account-id*:job/*job-id* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：ClearDefaultAuthorizer	ClearDefaultAuthorizer	无
物联网：CreateAuthorizer	CreateAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name*
物联网：CreateCertificateFromCsr	CreateCertificateFromCsr	*
物联网：CreateJob	CreateJob	arn:aws:iot:*region*:*account-id*:job/*job-id* arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* arn:aws:iot:*region*:*account-id*:thing/*thing-name* arn:aws:iot:*region*:*account-id*:jobtemplate/*job-template-id*
物联网：CreateJobTemplate	CreateJobTemplate	arn:aws:iot:*region*:*account-id*:job/*job-id* arn:aws:iot:*region*:*account-id*:jobtemplate/*job-template-id*
物联网：CreateKeysAndCertificate	CreateKeysAndCertificate	*
物联网：CreatePolicy	CreatePolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
CreatePolicyVersion	物联网：CreatePolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name* **Note** 这必须是 Amazon IoT 策略，而不是 IAM policy。
物联网：CreateRoleAlias	CreateRoleAlias	(参数：roleAlias) arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*

策略操作	Amazon IoT API	资源
物联网：CreateThing	CreateThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：CreateThingGroup	CreateThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* 针对要创建的组和父组 (如果使用)
物联网：CreateThingType	CreateThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
物联网：CreateTopicRule	CreateTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-name*
iot:DeleteCACertificate	DeleteCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*
物联网：DeleteCertificate	DeleteCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：DeleteJob	DeleteJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：DeleteJobExecution	DeleteJobExecution	arn:aws:iot:*region*:*account-id*:job/*job-id* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：DeleteJobTemplate	DeleteJobTemplate	arn:aws:iot:*region*:*account-id*:jobtemplate/*job-template-id*
物联网：DeletePolicy	DeletePolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：DeletePolicyVersion	DeletePolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：DeleteRegistrationCode	DeleteRegistrationCode	*
物联网：DeleteRoleAlias	DeleteRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
物联网：DeleteThing	DeleteThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：DeleteThingGroup	DeleteThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：DeleteThingType	DeleteThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
物联网：DeleteTopicRule	DeleteTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网:Deletev2LoggingLevel	删除 v2LoggingLevel	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*

策略操作	Amazon IoT API	资源
物联网：DeprecateThingType	DeprecateThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
物联网：DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name* (参数：authorizerName) 无
iot:DescribeCACertificate	DescribeCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*
物联网：DescribeCertificate	DescribeCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：DescribeDefaultAuthorizer	DescribeDefaultAuthorizer	无
物联网：DescribeEndpoint	DescribeEndpoint	*
物联网：DescribeEventConfigurations	DescribeEventConfigurations	无
物联网：DescribeIndex	DescribeIndex	arn:aws:iot:*region*:*account-id*:index/*index-name*
物联网：DescribeJob	DescribeJob	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：DescribeJobExecution	DescribeJobExecution	无
物联网：DescribeJobTemplate	DescribeJobTemplate	arn:aws:iot:*region*:*account-id*:jobtemplate/*job-template-id*
物联网：DescribeRoleAlias	DescribeRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
物联网：DescribeThing	DescribeThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：DescribeThingGroup	DescribeThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：DescribeThingRegistrationTask	DescribeThingRegistrationTask	无
物联网：DescribeThingType	DescribeThingType	arn:aws:iot:*region*:*account-id*:thingtype/*thing-type-name*
物联网：DetachPolicy	DetachPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id* or arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：DetachPrincipalPolicy	DetachPrincipalPolicy	arn:aws:iot:*region*:*account-id*:cert/*cert-id*

策略操作	Amazon IoT API	资源
物联网：DetachThingPrincipal	DetachThingPrincipal	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：DisableTopicRule	DisableTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：EnableTopicRule	EnableTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：GetEffectivePolicies	GetEffectivePolicies	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：GetIndexingConfiguration	GetIndexingConfiguration	无
物联网：GetJobDocument	GetJobDocument	arn:aws:iot:*region*:*account-id*:job/*job-id*
物联网：GetLoggingOptions	GetLoggingOptions	*
物联网：GetPolicy	GetPolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：GetPolicyVersion	GetPolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：GetRegistrationCode	GetRegistrationCode	*
物联网：GetTopicRule	GetTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：ListAttachedPolicies	ListAttachedPolicies	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* or arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：ListAuthorizers	ListAuthorizers	无
iot:ListCACertificates	ListCACertificates	*
物联网：ListCertificates	ListCertificates	*
物联网：ListCertificatesByCA	ListCertificatesByCA	*
物联网：ListIndices	ListIndices	无
物联网：ListJobExecutionsForJob	ListJobExecutionsForJob	无
物联网：ListJobExecutionsForThing	ListJobExecutionsForThing	无

策略操作	Amazon IoT API	资源
物联网：ListJobs	ListJobs	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* 如果使用了 thingGroupName 参数
物联网：ListJobTemplates	ListJobTemplates	无
物联网：ListOutgoingCertificates	ListOutgoingCertificates	*
物联网：ListPolicies	ListPolicies	*
物联网：ListPolicyPrincipals	ListPolicyPrincipals	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：ListPolicyVersions	ListPolicyVersions	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：ListPrincipalPolicies	ListPrincipalPolicies	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：ListPrincipalThings	ListPrincipalThings	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：ListRoleAliases	ListRoleAliases	无
物联网：ListTargetsForPolicy	ListTargetsForPolicy	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：ListThingGroups	ListThingGroups	无
物联网：ListThingGroupsForThing	ListThingGroupsForThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：ListThingPrincipals	ListThingPrincipals	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：ListThingRegistrationTaskReports	ListThingRegistrationTaskReports	无
物联网：ListThingRegistrationTasks	ListThingRegistrationTasks	无
物联网：ListThingTypes	ListThingTypes	*
物联网：ListThings	ListThings	*
物联网：ListThingsInThingGroup	ListThingsInThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：ListTopicRules	ListTopicRules	*

策略操作	Amazon IoT API	资源
物联网:Listv2LoggingLevels	ListV2LoggingLevels	无
iot:RegisterCACertificate	RegisterCACertificate	*
物联网：RegisterCertificate	RegisterCertificate	*
物联网：RegisterThing	RegisterThing	无
物联网：RejectCertificateTransfer	RejectCertificateTransfer	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：RemoveThingFromThingGroup	RemoveThingFromThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name* arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：ReplaceTopicRule	ReplaceTopicRule	arn:aws:iot:*region*:*account-id*:rule/*rule-name*
物联网：SearchIndex	SearchIndex	arn:aws:iot:*region*:*account-id*:index/*index-id*
物联网：SetDefaultAuthorizer	SetDefaultAuthorizer	arn:aws:iot:*region*:*account-id*:authorizer/*authorizer-function-name*
物联网：SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot:*region*:*account-id*:policy/*policy-name*
物联网：SetLoggingOptions	SetLoggingOptions	*
物联网:setv2LoggingLevel	setV2LoggingLevel	*
物联网:setv2LoggingOptions	setV2LoggingOptions	*
物联网：StartThingRegistrationTask	StartThingRegistrationTask	无
物联网：StopThingRegistrationTask	StopThingRegistrationTask	无
物联网：TestAuthorization	TestAuthorization	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：TestInvokeAuthorizer	TestInvokeAuthorizer	无
物联网：TransferCertificate	TransferCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot:*region*:*account-id*:authorizerfunction/*authorizer-function-name*
iot:UpdateCACertificate	UpdateCACertificate	arn:aws:iot:*region*:*account-id*:cacert/*cert-id*

策略操作	Amazon IoT API	资源
物联网：UpdateCertificate	UpdateCertificate	arn:aws:iot:*region*:*account-id*:cert/*cert-id*
物联网：UpdateEventConfigurations	UpdateEventConfigurations	无
物联网：UpdateIndexingConfiguration	UpdateIndexingConfiguration	无
物联网：UpdateRoleAlias	UpdateRoleAlias	arn:aws:iot:*region*:*account-id*:rolealias/*role-alias-name*
物联网：UpdateThing	UpdateThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*
物联网：UpdateThingGroup	UpdateThingGroup	arn:aws:iot:*region*:*account-id*:thinggroup/*thing-group-name*
物联网：UpdateThingGroupsForThing	UpdateThingGroupsForThing	arn:aws:iot:*region*:*account-id*:thing/*thing-name*

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

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

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

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

Device Advisor 资源

要为 Amazon IoT Device Advisor IAM policy 定义资源级别限制，请使用以下资源 ARN 格式进行套件定义和套件运行。

套件定义资源 ARN 格式

```
arn:aws:iotdeviceadvisor:region:account-id:suitedefinition/suite-definition-id
```

套件运行资源 ARN 格式

```
arn:aws:iotdeviceadvisor:region:account-id:suiterun/suite-definition-id/suite-run-id
```

条件键

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

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

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

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

Amazon支持全局条件键和特定于服务的条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

Amazon IoT 定义了自己的一组条件键，还支持使用一些全局条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

Amazon IoT 条件键

Amazon IoT 条件键	描述	类型
`aws:RequestTag/${tag-key}`	用户向 Amazon IoT 发出的请求中包含的标签键。	字符串
`aws:ResourceTag/${tag-key}`	附加到 Amazon IoT 资源的标签的标签键组件。	字符串
`aws:TagKeys`	与请求中的资源关联的所有标签键名称的列表。	字符串

要查看 Amazon IoT 条件键的列表，请参阅 IAM 用户指南中的 Amazon IoT 条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon IoT 定义的操作。

示例

要查看 Amazon IoT 基于身份的策略的示例，请参阅 Amazon IoT 基于身份的策略示例 (p. 370)。

Amazon IoT 基于资源的策略

基于资源的策略是 JSON 策略文档，它们指定了指定委托人可在 Amazon IoT 资源上执行的操作以及在什么条件下可执行。

Amazon IoT 不支持基于 IAM 资源的策略。不过，它支持 Amazon IoT 基于资源的策略。有关更多信息，请参阅Amazon IoT Core 策略 (p. 299)：

基于 Amazon IoT 标签的授权

您可以将标签附加到 Amazon IoT 资源或将请求中的标签传递到 Amazon IoT。要基于标签控制访问，您需要使用 `iot:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。有关更多信息，请参阅在 IAM 策略中使用标签 (p. 260)：有关标记 Amazon IoT 资源的更多信息，请参阅给您的 Amazon IoT 资源加标签 (p. 259)。

要查看基于身份的策略（用于根据资源上的标签来限制对该资源的访问）的示例，请参阅根据标签查看 Amazon IoT 资源 (p. 372)。

Amazon IoT IAM 角色

IAM 角色是 Amazon Web Services 账户中具有特定权限的实体。

将临时凭证用于 Amazon IoT

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用Amazon STS API 操作（如AssumeRole或）获得临时安全凭证GetFederationToken。

Amazon IoT 支持使用临时凭证。

服务相关角色

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

Amazon IoT 不支持服务相关角色。

服务角色

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

Amazon IoT 基于身份的策略示例

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

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

主题

- 策略最佳实践 (p. 370)
- 使用 Amazon IoT 控制台 (p. 371)
- 允许用户查看他们自己的权限 (p. 371)
- 根据标签查看 Amazon IoT 资源 (p. 372)
- 根据标签查看 Amazon IoT Device Advisor 资源 (p. 372)

策略最佳实践

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

- Amazon 托管策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 Amazon 托管策略来为许多常见使用场景授予权限。您可以在 Amazon Web Services 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 Amazon 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管策略或工作职能的 Amazon 托管策略。
- 应用最低权限 – 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM policy 中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 Amazon Web Service（例如 Amazon CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。

- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- Require multi-factor authentication（MFA）[需要多重身份验证（MFA）] – 如果您所处的场景要求您的 Amazon Web Services 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Amazon IoT 控制台

要访问 Amazon IoT 控制台，您必须拥有一组最低的权限。这些权限必须允许您列出和查看有关您的Amazon Web Services 账户中的 Amazon IoT 资源的详细信息。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

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

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

允许用户查看他们自己的权限

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

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

根据标签查看 Amazon IoT 资源

您可以在基于身份的策略中使用条件，以便基于标签控制对 Amazon IoT 资源的访问。该示例说明了如何创建策略以允许查看事物。不过，只有在事物标签 Owner 具有该用户的用户名值时，才会授予权限。此策略还授予在控制台上完成此操作的必要权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListBillingGroupsInConsole",
            "Effect": "Allow",
            "Action": "iot:ListBillingGroups",
            "Resource": "*"
        },
        {
            "Sid": "ViewBillingGroupsIfOwner",
            "Effect": "Allow",
           "Action": "iot:DescribeBillingGroup",
            "Resource": "arn:aws:iot:*:*:billinggroup/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

您可以将该策略附加到您账户中的 IAM 用户。如果名为 `richard-roe` 的用户尝试查看 Amazon IoT 账单组，则必须将账单组标记为 `Owner=richard-roe` 或 `owner=richard-roe`。否则，他将被拒绝访问。条件标签键 `Owner` 匹配 `Owner` 和 `owner`，因为条件键名称不区分大小写。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。

根据标签查看 Amazon IoT Device Advisor 资源

您可以在基于身份的策略中使用条件，以便基于标签控制对 Amazon IoT Device Advisor 资源的访问。以下示例说明如何创建策略以允许查看特定套件定义。但是，仅当套件定义标签具有值设置为 `MQTT` 的 `SuiteType` 时，才会授予权限。此策略还授予在控制台上完成此操作的必要权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewSuiteDefinition",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:GetSuiteDefinition",
            "Resource": "arn:aws:iotdeviceadvisor:*:*:suitedefinition/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/SuiteType": "MQTT"}
            }
        }
    ]
}
```

Amazon适用于 Amazon IoT 的托管策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 Amazon 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户托管策略需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管式策略。这

些策略涵盖常见使用案例，可在您的 Amazon Web Services 账户 中使用。有关 Amazon 托管策略的更多信息，请参阅 IAM 用户指南中的Amazon 托管策略。

Amazon 服务负责维护和更新 Amazon 托管策略。您无法更改 Amazon 托管策略中的权限。服务偶尔会向 Amazon 托管式策略添加额外权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新功能或新操作可用时，服务最有可能会更新 Amazon 托管式策略。服务不会从 Amazon 托管式策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，Amazon 还支持跨多种服务的工作职能的托管式策略。例如，ReadOnlyAccessAmazon托管策略提供对所有Amazon服务和资源的只读访问权限。当服务启动新功能时，Amazon 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的 Amazon 托管策略。

> Note
>
> Amazon IoT 可同时与 Amazon IoT 和 IAM policy 结合使用。本主题仅讨论 IAM policy，它为控制面板和数据层面 API 操作定义策略操作。另请参阅 Amazon IoT Core 策略 (p. 299)。

Amazon托管策略： AWSIoTConfigAccess

您可以将 `AWSIoTConfigAccess` 策略附加得到 IAM 身份。

此策略向相关身份授予权限，以允许访问所有 Amazon IoT 配置操作。此策略可能会影响数据处理和存储。要在 Amazon Web Services Management Console中查看该策略，请参阅 AWSIoTConfigAccess。

权限详细信息

此策略包含以下权限。

- `iot` – 检索 Amazon IoT 数据和执行 IoT 配置操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AcceptCertificateTransfer",
                "iot:AddThingToThingGroup",
                "iot:AssociateTargetsWithJob",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CancelCertificateTransfer",
                "iot:CancelJob",
                "iot:CancelJobExecution",
                "iot:ClearDefaultAuthorizer",
                "iot:CreateAuthorizer",
                "iot:CreateCertificateFromCsr",
                "iot:CreateJob",
                "iot:CreateKeysAndCertificate",
                "iot:CreateOTAUpdate",
```

```
                "iot:CreatePolicy",
                "iot:CreatePolicyVersion",
                "iot:CreateRoleAlias",
                "iot:CreateStream",
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:CreateThingType",
                "iot:CreateTopicRule",
                "iot:DeleteAuthorizer",
                "iot:DeleteCACertificate",
                "iot:DeleteCertificate",
                "iot:DeleteJob",
                "iot:DeleteJobExecution",
                "iot:DeleteOTAUpdate",
                "iot:DeletePolicy",
                "iot:DeletePolicyVersion",
                "iot:DeleteRegistrationCode",
                "iot:DeleteRoleAlias",
                "iot:DeleteStream",
                "iot:DeleteThing",
                "iot:DeleteThingGroup",
                "iot:DeleteThingType",
                "iot:DeleteTopicRule",
                "iot:DeleteV2LoggingLevel",
                "iot:DeprecateThingType",
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachPrincipalPolicy",
                "iot:DetachThingPrincipal",
                "iot:DisableTopicRule",
                "iot:EnableTopicRule",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
```

```
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:RegisterCACertificate",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RejectCertificateTransfer",
                "iot:RemoveThingFromThingGroup",
                "iot:ReplaceTopicRule",
                "iot:SearchIndex",
                "iot:SetDefaultAuthorizer",
                "iot:SetDefaultPolicyVersion",
                "iot:SetLoggingOptions",
                "iot:SetV2LoggingLevel",
                "iot:SetV2LoggingOptions",
                "iot:StartThingRegistrationTask",
                "iot:StopThingRegistrationTask",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:TransferCertificate",
                "iot:UpdateAuthorizer",
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:UpdateEventConfigurations",
                "iot:UpdateIndexingConfiguration",
                "iot:UpdateRoleAlias",
                "iot:UpdateStream",
                "iot:UpdateThing",
                "iot:UpdateThingGroup",
                "iot:UpdateThingGroupsForThing",
                "iot:UpdateAccountAuditConfiguration",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DeleteAccountAuditConfiguration",
                "iot:StartOnDemandAuditTask",
                "iot:CancelAuditTask",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:CreateScheduledAudit",
                "iot:UpdateScheduledAudit",
                "iot:DeleteScheduledAudit",
                "iot:DescribeScheduledAudit",
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:CreateSecurityProfile",
                "iot:DescribeSecurityProfile",
                "iot:UpdateSecurityProfile",
                "iot:DeleteSecurityProfile",
                "iot:AttachSecurityProfile",
                "iot:DetachSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
```

```
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon托管策略： AWSIoTConfigReadOnlyAccess

您可以将 AWSIoTConfigReadOnlyAccess 策略附加得到 IAM 身份。

此策略向相关身份授予权限，以允许只读访问所有 Amazon IoT 配置操作。要在 Amazon Web Services Management Console中查看该策略，请参阅 AWSIoTConfigReadOnlyAccess。

权限详细信息

此策略包含以下权限。

- iot – 对 IoT 配置操作执行只读操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:DescribeAuthorizer",
                "iot:DescribeCACertificate",
                "iot:DescribeCertificate",
                "iot:DescribeDefaultAuthorizer",
                "iot:DescribeEndpoint",
                "iot:DescribeEventConfigurations",
                "iot:DescribeIndex",
                "iot:DescribeJob",
                "iot:DescribeJobExecution",
                "iot:DescribeRoleAlias",
                "iot:DescribeStream",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingRegistrationTask",
                "iot:DescribeThingType",
                "iot:GetEffectivePolicies",
                "iot:GetIndexingConfiguration",
                "iot:GetJobDocument",
                "iot:GetLoggingOptions",
                "iot:GetOTAUpdate",
                "iot:GetPolicy",
                "iot:GetPolicyVersion",
                "iot:GetRegistrationCode",
                "iot:GetTopicRule",
                "iot:GetV2LoggingOptions",
                "iot:ListAttachedPolicies",
```

```
                "iot:ListAuthorizers",
                "iot:ListCACertificates",
                "iot:ListCertificates",
                "iot:ListCertificatesByCA",
                "iot:ListIndices",
                "iot:ListJobExecutionsForJob",
                "iot:ListJobExecutionsForThing",
                "iot:ListJobs",
                "iot:ListOTAUpdates",
                "iot:ListOutgoingCertificates",
                "iot:ListPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPolicyVersions",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListRoleAliases",
                "iot:ListStreams",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroups",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:ListThingRegistrationTaskReports",
                "iot:ListThingRegistrationTasks",
                "iot:ListThings",
                "iot:ListThingsInThingGroup",
                "iot:ListThingTypes",
                "iot:ListTopicRules",
                "iot:ListV2LoggingLevels",
                "iot:SearchIndex",
                "iot:TestAuthorization",
                "iot:TestInvokeAuthorizer",
                "iot:DescribeAccountAuditConfiguration",
                "iot:DescribeAuditTask",
                "iot:ListAuditTasks",
                "iot:DescribeScheduledAudit",
                "iot:ListScheduledAudits",
                "iot:ListAuditFindings",
                "iot:DescribeSecurityProfile",
                "iot:ListSecurityProfiles",
                "iot:ListSecurityProfilesForTarget",
                "iot:ListTargetsForSecurityProfile",
                "iot:ListActiveViolations",
                "iot:ListViolationEvents",
                "iot:ValidateSecurityProfileBehaviors"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon托管策略： AWSIoTDataAccess

您可以将 AWSIoTDataAccess 策略附加得到 IAM 身份。

此策略向相关身份授予权限，以允许访问所有 Amazon IoT 数据操作。数据操作通过 MQTT 或 HTTP 协议发送数据。要在 Amazon Web Services Management Console中查看该策略，请参阅 AWSIoTDataAccess。

权限详细信息

此策略包含以下权限。

- `iot` – 检索 Amazon IoT 数据并允许对 Amazon IoT 消息收发操作进行完全访问。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect",
                "iot:Publish",
                "iot:Subscribe",
                "iot:Receive",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow",
                "iot:DeleteThingShadow",
                "iot:ListNamedShadowsForThing"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon托管策略： AWSIoTFullAccess

您可以将 `AWSIoTFullAccess` 策略附加得到 IAM 身份。

此策略向相关身份授予权限，以允许访问所有 Amazon IoT 配置和消息收发操作。要在 Amazon Web Services Management Console中查看该策略，请参阅 `AWSIoTFullAccess`。

权限详细信息

此策略包含以下权限。

- `iot` – 检索 Amazon IoT 数据并允许对 Amazon IoT 配置和消息收发操作进行完全访问。
- `iotjobsdata` – 检索 Amazon IoT Jobs 数据并允许完全访问 Amazon IoT Jobs 数据层面 API 操作。

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

Amazon托管策略： AWSIoTLogging

您可以将 AWSIoTLogging 策略附加得到 IAM 身份。

此策略向相关身份授予权限，以允许创建 Amazon CloudWatch Logs 组和将日志流式传输到这些组。本策略附加到您的 CloudWatch 日志记录角色。要在 Amazon Web Services Management Console中查看该策略，请参阅 AWSIoTLogging。

权限详细信息

此策略包含以下权限。

- logs— 检索 CloudWatch 日志。此外，允许创建 CloudWatch 日志组和将日志流式传输到这些组。

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

Amazon托管策略： AWSIoTOTAUpdate

您可以将 AWSIoTOTAUpdate 策略附加得到 IAM 身份。

此策略向相关身份授予权限，以允许创建 Amazon IoT 任务、Amazon IoT 代码签署任务，并描述 Amazon 代码签署者任务。要在 Amazon Web Services Management Console中查看该策略，请参阅 AWSIoTOTAUpdate。

权限详细信息

此策略包含以下权限。

- iot – 创建 Amazon IoT 任务和代码签署任务。

- `signer` – 执行创建 Amazon 代码签署者任务。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateJob",
            "signer:DescribeSigningJob"
        ],
        "Resource": "*"
    }
}
```

Amazon托管策略： AWSIoTRuleActions

您可以将 `AWSIoTRuleActions` 策略附加得到 IAM 身份。

此策略向相关身份授予权限，以允许访问所有 Amazon IoT 规则操作中支持的所有 Amazon Web Service。要在 Amazon Web Services Management Console中查看该策略，请参阅 AWSIoTRuleActions。

权限详细信息

此策略包含以下权限。

- `iot` - 执行用于发布规则操作消息的操作。
- `dynamodb` - 将消息插入到 DynamoDB 表或将消息拆分为 DynamoDB 表的多列。
- `s3` - 将对象存储在 Amazon S3 存储桶中。
- `kinesis` - 将消息发送到 Amazon Kinesis 流对象。
- `firehose` - 在 Kinesis Data Firehose 流对象中插入记录。
- `cloudwatch`-更改 CloudWatch 告警状态或将消息数据发送到 CloudWatch 指标。
- `sns` - 执行使用 Amazon SNS 发布通知的操作。此操作的范围为 Amazon IoT SNS 主题。
- `sqs` - 插入要添加到 SQS 队列的消息。
- `es`-向 OpenSearch 服务服务发送消息。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "dynamodb:PutItem",
            "kinesis:PutRecord",
            "iot:Publish",
            "s3:PutObject",
            "sns:Publish",
            "sqs:SendMessage*",
            "cloudwatch:SetAlarmState",
            "cloudwatch:PutMetricData",
```

```
            "es:ESHttpPut",
            "firehose:PutRecord"
        ],
        "Resource": "*"
    }
}
```

Amazon托管策略： AWSIoTThingsRegistration

您可以将 AWSIoTThingsRegistration 策略附加得到 IAM 身份。

此策略向相关身份授予权限，以允许使用 StartThingRegistrationTask API 批量注册事物。此策略可能会影响数据处理和存储。要在 Amazon Web Services Management Console中查看该策略，请参阅 AWSIoTThingsRegistration。

权限详细信息

此策略包含以下权限。

- iot - 批量注册时，执行用于创建内容以及附加策略和证书的操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachPrincipalPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateCertificateFromCsr",
                "iot:CreatePolicy",
                "iot:CreateThing",
                "iot:DescribeCertificate",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:DescribeThingType",
                "iot:DetachPolicy",
                "iot:DetachThingPrincipal",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListPolicyPrincipals",
                "iot:ListPrincipalPolicies",
                "iot:ListPrincipalThings",
                "iot:ListTargetsForPolicy",
                "iot:ListThingGroupsForThing",
                "iot:ListThingPrincipals",
                "iot:RegisterCertificate",
                "iot:RegisterThing",
                "iot:RemoveThingFromThingGroup",
                "iot:UpdateCertificate",
                "iot:UpdateThing",
                "iot:UpdateThingGroupsForThing",
                "iot:AddThingToBillingGroup",
```

```
                "iot:DescribeBillingGroup",
                "iot:RemoveThingFromBillingGroup"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

对 Amazon 托管式策略的 Amazon IoT 更新

查看有关 Amazon IoT 的 Amazon 托管式策略更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的自动提示，请订阅 Amazon IoT 文档历史记录页面上的 RSS 源。

更改	说明	日期
AWSIoTFullAccess (p. 378) – 对现有策略的更新	Amazon IoT 添加了新的权限，以允许用户使用 HTTP 协议访问 Amazon IoT Jobs 数据层面 API 操作。 一个新的 IAM policy 前缀 `iotjobsdata:` 为您提供了更精细的访问控制机制，以访问 Amazon IoT Jobs 数据层面终端节点。对于控制面板 API 操作，您仍然使用 `iot:` 前缀。有关更多信息，请参阅HTTPS 协议的 Amazon IoT Core 策略 (p. 729)：	2022 年 5 月 11 日
Amazon IoT 已开启跟踪更改	Amazon IoT 为其 Amazon 托管式策略开启了跟踪更改。	2022 年 5 月 11 日

对 Amazon IoT 身份和访问进行故障排除

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

主题

- 我无权在 Amazon IoT 中执行操作 (p. 382)
- 我无权执行 iam：PassRole (p. 383)
- 我希望允许我的 Amazon Web Services 账户以外的人访问我的 Amazon IoT 资源 (p. 383)

我无权在 Amazon IoT 中执行操作

如果您收到一个错误，指明您无权执行某个操作，则必须更新策略以允许您执行该操作。

当 IAM 用户尝试使用控制台查看有关事物资源的详细信息，但不具有`iot:DescribeThing`权限时，会发生以下示例错误。`mateojackson`

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iot:DescribeThing on resource: MyIoTThing
```

在此情况下，必须更新mateojackson用户的策略，以允许使用iot:*DescribeThing*操作访问事物资源。

如果您需要帮助，请联系您的 Amazon 管理员。管理员是向您提供登录凭证的人。

使用 Amazon IoT Device Advisor

如果您使用 DeAmazon IoT vice Advisor，则当用户mateojackson尝试使用控制台查看有关套件定义的详细信息，但不具有iotdeviceadvisor:*GetSuiteDefinition*权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotdeviceadvisor:GetSuiteDefinition on resource: MySuiteDefinition
```

在这种情况下，必须更新mateojackson用户的策略以允许使用iotdeviceadvisor:*GetSuiteDefinition*操作访问*MySuiteDefinition*资源。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 Amazon IoT。

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

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

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

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 Amazon 管理员。管理员是向您提供登录凭证的人。

我希望允许我的 Amazon Web Services 账户以外的人访问我的 Amazon IoT 资源

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

要了解更多信息，请参阅以下内容：

- 要了解 Amazon IoT 是否支持这些功能，请参阅 Amazon IoT 如何与 IAM 协同工作 (p. 351)。
- 要了解如何为您拥有的 Amazon Web Services 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 Amazon Web Services 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方Amazon Web Services 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的Amazon Web Services 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

日志记录和监控

监控是保持 Amazon IoT 和您的 Amazon 解决方案的可靠性、可用性和性能的重要方面。您应从 Amazon 解决方案的所有部分收集监控数据，以便更轻松地调试出现的多点故障。有关日志记录和监控流程的信息，请参阅监控 Amazon IoT (p. 394)

监控工具

Amazon 为您提供了各种可用于监控 Amazon IoT 的工具。您可以配置其中的一些工具以便进行监控。一些工具需要手动干预。建议您尽可能实现监控任务自动化。

自动监控工具

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

- Amazon CloudWatch 警报 — 按您指定的时间段观察单个指标，并根据相对于给定阈值的指标值在若干时间段内执行一项或多项操作。操作是一个发送到 Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon EC2 Auto Scaling 策略的通知。 CloudWatch 告警将不会调用操作，因为这些操作处于特定状态。该状态必须已改变并在指定的若干个时间段内保持不变。有关更多信息，请参阅使用 Amazon 监控 Amazon IoT告警和指标 CloudWatch (p. 400)：
- Amazon CloudWatch Logs — 监控、存储和访问来自Amazon CloudTrail或其他来源的日志文件。Ama CloudWatch zon Logs 还允许您查看 Device Advisor 测试用例采取的关键步骤Amazon IoT、生成的事件和Amazon IoT Core在测试执行过程中从您的设备或发送的 MQTT 消息。这些日志使您能够在设备上进行调试和采取纠正措施。有关更多信息，请参阅Amazon IoT使用 CloudWatch 日志进行监控 (p. 412)有关使用 Amazon 的更多信息 CloudWatch，请参阅亚马逊 CloudWatch 用户指南中的监控日志文件。
- Amazon CloudWatch Events — 匹配事件并将事件传送到一个或多个目标函数或流来进行更改、捕获状态信息和采取纠正措施。有关更多信息，请参阅《亚马逊 CloudWatch 用户指南》中的 “什么是亚马逊 CloudWatch 活动”。
- Amazon CloudTrail日志监控 — 在账户间共享日志文件，通过将 CloudTrail 日志文件发送到 Lo CloudWatch gs 对它们进行实时监控，在 Java 中编写日志处理应用程序，以及验证您的日志文件在交付后未发生更改 CloudTrail。有关更多信息使用 Amazon CloudTrail 记录 Amazon IoT API 调用 (p. 440)，请参阅《Amazon CloudTrail用户指南》中的 “使用 CloudTrail 日志文件”。

手动监控工具

监控的另一个重要环Amazon IoT节是手动监控 CloudWatch 警报未涵盖的项目。Amazon IoT CloudWatch、和其他Amazon服务控制台仪表板提供Amazon环境状态 at-a-glance 视图。建议您还要查看 Amazon IoT 上的日志文件。

- Amazon IoT 控制面板显示：
  - CA 证书
  - 证书
  - 策略
  - 规则
  - 事物
- CloudWatch 主页将显示以下内容：
  - 当前警报和状态。
  - 警报和资源的图表。
  - 服务运行状况。

  您可以使用 CloudWatch 来执行以下操作：
  - 创建自定义控制面板以监控您关心的服务。

- 绘制指标数据图，以排除问题并弄清楚趋势。
- 搜索并浏览您所有的 Amazon 资源指标。
- 创建和编辑警报以接收有关问题的通知。

Amazon IoT Core 的合规性验证

要了解某个 Amazon Web Service 是否在特定合规性计划范围内，请参阅 合规性计划范围内的 Amazon Web Services，然后选择您感兴趣的合规性计划。有关常规信息，请参阅Amazon Web Services合规性计划。

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

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

- 安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在 Amazon 上部署以安全性和合规性为重点的基准环境的步骤。
- Amazon合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- Amazon Config 开发人员指南中的使用规则评估资源 – 此 Amazon Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub— 这Amazon Web Service可以全面了解您的内部安全状态Amazon。Security Hub使用安全控制来评估您的Amazon资源，并检查您是否符合安全行业标准和最佳实践。有关支持的服务和控件的列表，请参阅 S ecurity Hub 控件参考资料。

Amazon IoT Core 中的故障恢复能力

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

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

Amazon IoT Core 将设备相关信息存储在设备注册表中。此外，它还存储 CA 证书、设备证书和设备影子数据。发生硬件或网络故障时，此数据会跨可用区自动复制，但不会跨区域复制。

在更新设备注册表时，Amazon IoT Core 将发布 MQTT 事件。您可以使用这些消息来备份注册表数据并将其保存到某个位置（如 DynamoDB 表）。您负责保存 Amazon IoT Core 为您创建的证书或您自己创建的证书。设备影子将存储有关您设备的状态数据，并可在设备重新联机时重新发送。Amazon IoTDevice Advisor 存储有关测试套件配置的信息。发生硬件或网络故障时，此数据将自动复制。

除非您特意选择这样做，否则 Amazon IoT Core 资源均特定于区域，并且不会跨Amazon Web Services 区域复制。

有关安全性最佳实践的更多信息，请参阅 Amazon IoT Core 中的安全最佳实践 (p. 388)。

将 Amazon IoT Core 和接口 VPC 终端节点一起使用

Amazon IoT Core 使您能够使用接口 VPC 端点在您的 VPC 中创建 IoT 数据端点。接口 VPC 终端节点由 Amazon PrivateLink此Amazon技术提供支持，技术使您能够Amazon通过使用私有 IP 地址访问在上运行的服务。有关更多信息，请参阅 Amazon Virtual Private Cloud。

为了将远程网络上的现场设备（如企业网络）连接到 Amazon VPC，请参阅网络到 Amazon VPC 的连接矩阵中列出的各种选项。

章节主题：

- 创建 Amazon IoT Core 的 VPC 终端节点 (p. 386)
- 使用 VPC 终端节点控制对 Amazon IoT Core 的访问 (p. 386)
- VPC 终端节点的限制 (p. 387)
- 使用 IoT Core 扩缩 VPC 终端节点 (p. 387)
- 将自定义域用于 VPC 终端节点 (p. 387)
- 适用于 Amazon IoT Core 的 VPC 终端节点的可用性 (p. 388)

创建 Amazon IoT Core 的 VPC 终端节点

要开始使用 VPC 终端节点，只需创建接口 VPC 终端节点，然后选择 Amazon IoT Core 作为 Amazon 服务。如果您使用 CLI，首先调用describe-vpc-endpoint-services以确保您选择了一个可用区域，其中存在 Amazon IoT Core于您的特定中Amazon Web Services 区域。例如，在 us-east-1 中，此命令将类似于：

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.data
```

> **Note**
>
> 用于自动创建 DNS 记录的 VPC 功能已被禁用。要连接到这些端点，您必须手动创建私有 DNS 记录。有关私有 VPC DNS 记录的更多信息，请参阅 接口终端节点的私有 DNS。有关 Amazon IoT Core VPC 限制的更多信息，请参阅 VPC 终端节点的限制 (p. 387)。

要将 MQTT 客户端连接到 VPC 端点接口，您必须在附加到 VPC 的私有托管区中手动创建 DNS 记录。要开始使用，请参阅创建私有托管区域。在您的私有托管区域中，为 VPC 终端节点的每个弹性网络接口 IP 创建别名记录。如果您有多个 VPC 终端节点的多个网络接口 IP，请在所有加权记录之间创建权重相等的加权 DNS 记录。通过描述字段中的 VPC 终端节点 ID 进行筛选时，可以从 DescribeNetworkInterfacesAPI 调用中获得这些 IP 地址。

使用 VPC 终端节点控制对 Amazon IoT Core 的访问

您可以将设备访问限制为 Amazon IoT Core 仅允许使用 VPC 条件上下文键通过 VPC 终端节点访问。Amazon IoT Core 支持以下 VPC 相关的上下文键：

- SourceVpc
- SourceVpce
- VPCSourceIp

> **Note**
>
> Amazon IoT Core 目前不支持 https://docs.amazonaws.cn/vpc/latest/privatelink/vpc-endpoints-access.html#vpc-endpoint-policies VPC 终端节点策略。

例如，以下策略授予权限以使用与事物名称匹配的客户端 ID 连接到 Amazon IoT Core，并发布到任何以事物名称为前缀的主题，具体条件是设备连接到具有特定 VPC 终端节点 ID 的 VPC 终端节点。此策略将拒绝连接到您的公有 IoT 数据终端节点的尝试。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }

        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

VPC 终端节点的限制

本部分介绍与公有终端节点相比，VPC 终端节点的限制。

- 当前，VPC 终端节点仅受 IoT 数据终端节点支持
- MQTT 保持活动状态的时间限制为 230 秒。保持活动状态超过该时段的时间将自动减少到 230 秒
- 每个 VPC 终端节点总共支持 100,000 台并发互联设备。如果您需要更多连接，请参阅 使用 IoT Core 扩缩 VPC 终端节点 (p. 387)。
- VPC 终端节点仅支持 IPv4 流量。
- VPC 终端节点将仅提供 ATS 证书，但自定义域除外。
- 目前不支持 VPC 终端节点策略。
- 对于为 Amazon IoT Core 数据面板创建的 VPC 端点，Amazon IoT Core 不支持使用地区或区域性公有 DNS 记录。

使用 IoT Core 扩缩 VPC 终端节点

Amazon IoT Core 接口 VPC 终端节点仅限于通过单个接口终端节点连接的 100,000 台互联设备。如果您的使用案例需要更多到代理的并发连接，那么我们建议您使用多个 VPC 终端节点并在接口终端节点之间手动路由您的设备。创建私有 DNS 记录将流量路由到 VPC 终端节点时，确保创建与 VPC 终端节点 数量相同的加权记录，以便在多个终端节点之间 分配流量。

将自定义域用于 VPC 终端节点

如果要将自定义域与 VPC 终端节点结合使用，必须在私有托管区域中创建自定义域名记录并在 Route53 中创建路由记录。有关更多信息，请参阅创建私有托管区域。

适用于 Amazon IoT Core 的 VPC 终端节点的可用性

Amazon IoT Core 接口 VPC 端点在所有 Amazon IoT Core 支持的区域中都可用。

Amazon IoT 中的基础设施安全性

作为一项托管式服务的集合，Amazon IoT 由亚马逊云科技：安全流程概览白皮书中所述的Amazon全球网络安全程序提供保护。

您可以使用 Amazon 发布的 API 调用通过网络访问 Amazon IoT。客户端必须支持传输层安全性 (TLS) 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。有关更多信息，请参阅Amazon IoT 中的传输安全 (p. 346)：

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

对生产实例集或带 Amazon IoT Core 的设备进行安全监控

IoT 实例集可能由大量具有不同功能、长期存在且地理位置分散的设备组成。这些特性导致实例集设置复杂且 容易出错 。由于设备的计算能力、内存和存储功能通常有限，因而限制了在设备本身上对加密和其它形式的安全功能的使用。此外，设备经常使用具有已知漏洞的软件。这些因素不仅令 IoT 实例集成为吸引黑客的目标，而且导致难以持续保护设备实例集安全。

Amazon IoT Device Defender 提供了识别安全问题和最佳实践偏离情况的工具，从而解决了这些难题。您可以使用 Amazon IoT Device Defender 分析、审计和监控已连接的设备，以检测异常行为并降低安全风险。Amazon IoT Device Defender 可以审计设备实例集，以确保其遵守安全最佳实践，并检测设备上的异常行为。这使您能够在整个 Amazon IoT 设备实例集中实施一致的安全策略，并能够在设备遭到破坏时快速响应。有关更多信息，请参阅Amazon IoT Device Defender (p. 843)：

Amazon IoT Device Advisor 根据需要向您的实例集推送更新和修补程序。Amazon IoTDevice Advisor 会自动更新测试用例。 您选择 的测试用例始终为 最新版本。有关更多信息，请参阅Device Advisor (p. 1016)：

Amazon IoT Core 中的安全最佳实践

本节包含有关 Amazon IoT Core 的安全最佳实践的信息。有关更多信息，请参阅 IoT 解决方案的十大安全黄金规则。

在 Amazon IoT 中保护 MQTT 连接

Amazon IoT Core是一种托管式云服务，使互联设备可以轻松安全地与云应用程序及其他设备交互。Amazon IoT Core支持 HTTP WebSocket、和 MQTT，这是一种专为容忍间歇性连接而设计的轻量级通信协议。如果您使用 MQTT 连接到 Amazon IoT，则您的每个连接都必须与一个称为客户端 ID 的标识符关联。MQTT 客户端 ID 唯一地标识 MQTT 连接。如果使用已为另一个连接声明的客户端 ID 建立新连接，则 Amazon IoT 消息代理将删除旧连接以允许新连接。客户端 ID 在每个Amazon Web Services 账户和每个 Amazon Web Services 区域中必须是唯一的。这意味着您不需要在Amazon Web Services 账户之外或在您的Amazon Web Services 账户内跨区域强制执行客户端 ID 的全局唯一性。

在设备实例集上删除 MQTT 连接的影响和严重程度取决于许多因素。其中包括：

- 您的使用案例（例如，您的设备向 Amazon IoT 发送的数据、数据量以及发送数据的频率）。
- 您的 MQTT 客户端配置（例如，自动重新连接设置、关联的退避计时以及使用 MQTT 持久性会话 (p. 79)）。
- 设备资源限制。
- 断开连接的根本原因、其主动性和持久性。

要避免客户端 ID 冲突及其潜在的负面影响，请确保每个设备或移动应用程序都有一个 Amazon IoT 或 IAM policy，以限制可以将哪些客户端 ID 用于与 Amazon IoT 消息代理的 MQTT 连接。例如，您可以使用 IAM policy 防止设备无意中通过使用已在使用的客户端 ID 关闭其它设备的连接。有关更多信息，请参阅Authorization (p. 298)：

您实例集中的所有设备必须具有相应的凭证，这些凭证仅具有授权执行预期操作的权限，包括（但不限于）Amazon IoT MQTT 操作，例如发布消息或订阅具有特定作用域和上下文的主题。具体的权限策略可能因您的使用案例而异。确定最能满足您的业务和安全要求的权限策略。

要简化权限策略的创建和管理，您可以使用 Amazon IoT Core 策略变量 (p. 303) 和 IAM policy 变量。策略变量可以放在策略中，并且在评估策略时，变量将由来自设备请求的值替换。使用策略变量，您可以创建单个策略以授予对多个设备的权限。您可以根据用于连接到 Amazon IoT 消息代理的 Amazon IoT 帐户配置、身份验证机制和网络协议，为您的用例确定相关的策略变量。但是，要编写最佳权限策略，应考虑使用案例和威胁模型的具体情况。

例如，如果您在 Amazon IoT 注册表中注册设备，您可以在 Amazon IoT 策略中使用事物策略变量 (p. 304)根据事物名称、事物类型和事物属性值等事物属性来授予或拒绝权限。事物名称从事物连接至 Amazon IoT 时发送的 MQTT Connect 消息中的客户端 ID 获取。当事物通过使用 TLS 双向Amazon IoT身份验证的 MQTT 或通过 WebSocket 协议（使用经过身份验证的 Amazon Cognito 身份）连接到时，事物策略变量将被替换。您可以使用 AttachThingPrincipalAPI 将证书和经过身份验证的 Amazon Cognito 身份附加到事物。 `iot:Connection.Thing.ThingName`是强制执行客户端 ID 限制的有用策略变量。以下示例 Amazon IoT 策略要求将已注册事物的名称用作与 Amazon IoT 消息代理的 MQTT 连接的客户端 ID：

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "iot:Connect",
   "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
   ]
  }
 ]
}
```

如果您要识别正在进行的客户端 ID 冲突，可以启用和使用 CloudWatch LogsAmazon IoT (p. 412)。对于 Amazon IoT 消息代理因客户端 ID 冲突而断开连接的每个 MQTT连接，将生成类似于以下内容的日志记录：

```
{
    "timestamp": "2019-04-28 22:05:30.105",
    "logLevel": "ERROR",
    "traceId": "02a04a93-0b3a-b608-a27c-1ae8ebdb032a",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "clientId01",
    "principalId": "1670fcf6de55adc1930169142405c4a2493d9eb5487127cd0091ca0193a3d3f6",
    "sourceIp": "203.0.113.1",
    "sourcePort": 21335,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID"
```

```
}
```

您可以使用CloudWatch 日志筛选条件（如）{$.reason= "DUPLICATE_CLIENT_ID" }来搜索客户端 ID 冲突的实例，或者设置CloudWatch 指标筛选条件和相应的 CloudWatch 告警以进行连续监控和报告。

您可以使用 Amazon IoT Device Defender 识别过度宽容的 Amazon IoT 和 IAM policy。Amazon IoTDevice Defender 还提供审核检查，如果实例集中的多个设备使用相同的客户端 ID 连接到 Amazon IoT 消息代理，则会通知您。

您可以使用 Amazon IoT Device Advisor，验证您的设备是否可以可靠地连接到 Amazon IoT Core 并遵守安全最佳实践。

另请参阅

- Amazon IoT Core
- Amazon IoT 的安全功能 (p. 264)
- Amazon IoT Core 策略变量 (p. 303)
- IAM policy 变量
- Amazon Cognito 身份
- Amazon IoT Device Defender
- CloudWatch 的日志Amazon IoT (p. 412)

使设备的时钟保持同步

请务必确保您的设备上有准确的时间。X.509 证书具有到期日期和时间。设备上的时钟用于验证服务器证书是否仍有效。如果您要构建商用 IoT 设备，请记住，您的产品在出售前可能会存放较长时间。在此期间，实时时钟可能会出现偏移误差，并且电池可能会放电，因此在工厂设置时间是不够的。

对于大多数系统，这意味着设备的软件必须包含网络时间协议 (NTP) 客户端。设备应等待，直到它与 NTP 服务器同步，然后再尝试连接到 Amazon IoT Core。如果无法做到这一点，系统将为用户提供一种设置设备时间的方法，以便后续连接成功。

设备与 NTP 服务器同步后，便能打开与 Amazon IoT Core 的连接。允许的时钟偏移量取决于您尝试通过此连接执行的操作。

验证服务器证书

为了与 Amazon IoT 进行交互，设备执行的第一项操作是打开安全连接。在将设备连接到 Amazon IoT 时，请确保正在与 Amazon IoT（而不是模拟 Amazon IoT 的其它服务器）进行通信。为每个 Amazon IoT 服务器提供了针对 `iot.amazonaws.com` 域颁发的证书。此证书已由受信任的证书颁发机构颁发给 Amazon IoT，该机构验证了我们的身份和域所有权。

在设备连接时，Amazon IoT Core 执行的第一项操作是向设备发送服务器证书。设备可以验证它们是否希望连接到 `iot.amazonaws.com`，并验证位于该连接一端的服务器是否拥有来自该域的受信任颁发机构的证书。

TLS 证书采用 X.509 格式，并包含各种信息，例如组织的名称、位置、域名和有效期。有效期被指定为一对时间值（分别名为 `notBefore` 和 `notAfter`）。像 Amazon IoT Core 这样的服务对其服务器证书使用有限的有效期（例如，一年），并在旧证书到期之前开始提供新证书。

每个设备使用一个身份

每个客户端使用一个身份。设备通常使用 X.509 客户端证书。Web 和移动应用程序使用 Amazon Cognito Identity。这使您能够对设备应用细化权限。

例如，您有一个由手机设备构成的应用程序，这些设备接收来自两个不同的智能家居对象（灯泡和恒温器）的状态更新。灯泡发送电池电量的状态，恒温器发送报告温度的消息。

Amazon IoT 单独对各个设备进行身份验证，并单独处理每个连接。您可以使用授权策略应用精细访问控制。您可以为恒温器定义一个策略，该策略允许恒温器发布到主题空间的策略。您可以为灯泡定义一个单独策略，该策略允许灯泡发布到不同的主题空间。最后，您可以为移动应用程序定义一个策略，该策略只允许它连接到和订阅恒温器和灯泡的主题以接收来自这些设备的消息。

应用最小权限原则，并尽可能缩小每个设备的权限范围。所有设备或用户都应在 Amazon IoT 中有一个 Amazon IoT 策略，该策略仅允许设备或用户使用已知的客户端 ID 进行连接，并发布和订阅一组已确定的固定主题。

使用次级Amazon Web Services 区域作为备份

考虑在次级Amazon Web Services 区域中存储数据的副本以作为备份。有关更多信息，请参阅 Amazon IoT 的灾难恢复。

使用即时预调配

手动创建和预调配每个设备是一项耗时的工作。Amazon IoT 可让您定义模板以便在设备首次连接到 Amazon IoT 时对其进行预调配。有关更多信息，请参阅即时预调配 (p. 780)：

运行 Amazon IoT Device Advisor 测试的权限

下面的策略模板显示了运行 Amazon IoT Device Advisor 测试用例所需的最低权限和 IAM 实体。您需要替换*your-device-role-arn*为您根据先决条件创建的设备角色 Amazon Resource Name (ARN)。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "your-device-role-arn",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iotdeviceadvisor.amazonaws.com"
                }
            }
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke*",
                "iam:ListRoles",   // Required to list device roles in the Device Advisor
 console
                "iot:Connect",
                "iot:CreateJob",
                "iot:DeleteJob",
                "iot:DescribeCertificate",
                "iot:DescribeEndpoint",
                "iot:DescribeJobExecution",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iot:GetPendingJobExecutions",
                "iot:GetPolicy",
                "iot:ListAttachedPolicies",
                "iot:ListCertificates",
```

```
                "iot:ListPrincipalPolicies",
                "iot:ListThingPrincipals",
                "iot:ListThings",
                "iot:Publish",
                "iot:StartNextPendingJobExecution",
                "iot:UpdateJobExecution",
                "iot:UpdateThingShadow",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:PutRetentionPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": "iotdeviceadvisor:*",
            "Resource": "*"
        }
    ]
}
```

针对 Device Advisor 防范跨服务混淆代理

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

我们建议在资源策略中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键，以限制 Device Advisor 为其它服务提供的资源访问权限。如果使用两个全局条件上下文键，在同一策略语句中使用时，`aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户必须使用相同的账户 ID。

`aws:SourceArn` 的值必须是套件定义资源的 ARN。套件定义资源是指您使用 Device Advisor 创建的测试套件。

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

以下示例演示如何使用 Device Advisor 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键来防范混淆代理问题。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "iotdeviceadvisor.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:iotdeviceadvisor:us-east-1:123456789012:suitedefinition/
ygp6rxa3tzvn"
```

```
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

监控 Amazon IoT

监控是保持 Amazon IoT 和您的 Amazon 解决方案的可靠性、可用性和性能的重要方面。

我们强烈建议您从 Amazon 解决方案的各个部分收集监控数据，以便更轻松地调试出现的多点故障。首先创建一个监控计划来回答以下问题。如果您不确定如何回答这些问题，您仍然可以继续启用日志记录 (p. 394)并建立性能基准。

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

下一步是启用日志记录 (p. 394)，并通过在不同时间和不同负载条件下测量性能，在您的环境中建立正常 Amazon IoT 性能的基准。监控 Amazon IoT 时，请保留历史监控数据，以便将其与当前性能数据进行比较。这将帮助您确定一般的性能模式和性能异常，并设计解决问题的方法。

要为 Amazon IoT 建立基准性能，您应一开始便监控这些指标。您可在以后随时监控更多指标。

- `PublishIn.Success` (p. 406)
- `PublishOut.Success` (p. 406)
- `Subscribe.Success` (p. 406)
- `Ping.Success` (p. 406)
- `Connect.Success` (p. 406)
- `GetThingShadow.Accepted` (p. 408)
- `UpdateThingShadow.Accepted` (p. 408)
- `DeleteThingShadow.Accepted` (p. 408)
- `RulesExecuted` (p. 405)

本部分中的主题可帮助您启动对 Amazon IoT 的日志记录和监控操作。

主题

- 配置 Amazon IoT 日志记录 (p. 394)
- 使用 Amazon 监控Amazon IoT告警和指标 CloudWatch (p. 400)
- Amazon IoT使用 CloudWatch 日志进行监控 (p. 412)
- 将设备端日志上载到 Amazon CloudWatch (p. 432)
- 使用 Amazon CloudTrail 记录 Amazon IoT API 调用 (p. 440)

配置 Amazon IoT 日志记录

您必须使用 Amazon IoT 控制台、CLI 或 API 启用日志记录，然后才能监控和记录 Amazon IoT 活动。

您可以为所有 Amazon IoT 或仅为特定事物组启用日志记录。您可以使用 Amazon IoT 控制台、CLI 或 API 配置 Amazon IoT 日志记录；但是，必须使用 CLI 或 API 为特定事物组配置日志记录。

在考虑如何配置 Amazon IoT 日志记录时，除非另有指定，否则默认日志记录配置将确定如何记录 Amazon IoT 活动。首先，您可能要获取默认日志级别 (p. 400)为 INFO 或 DEBUG 的详细日志。查看初始日志后，您可以将默认日志级别更改为较低的详细程度级别（如 WARN 或 ERROR），并对可能需要更多关注的资源设置更详细的资源特定日志级别。日志级别可随时更改。

配置日志记录角色和策略

在 Amazon IoT 中启用日志记录之前，您必须创建一个 IAM 角色和一个策略，用于向 Amazon 授予代表您监控 Amazon IoT 活动的权限。

Note

在启用Amazon IoT日志记录之前，请务必了解 CloudWatch Logs 访问权限。拥有 L CloudWatch ogs 访问权限的用户可以从您的设备查看调试信息。有关更多信息，请参阅 Amazon CloudWatch Logs 的身份验证和访问控制。
如果您预计负载测试会在 Amazon IoT Core 中导致高流量模式，请考虑关闭 IoT 日志记录以防止节流。如果检测到高流量，我们的服务可能会禁用您账户中的日志记录。

以下显示了如何为 Amazon IoT Core 资源创建日志记录角色和策略。有关如何为 WA LoRa N 创建 IAM 日志角色和策略的信息，请参阅为 Amazon IoT Wireless 创建日志记录角色和策略 (p. 1249)。Amazon IoT Core

创建日志记录角色

要创建日志记录角色，请打开 IAM 控制台的角色中心并选择 Create role（创建角色）。

1. 在 Select trusted entity（选择受信任的实体）下，选择 Amazon Service（Amazon 服务）。然后在 Use case（用例）下，选择 IoT。如果您看不到 IoT，请在 Use cases for other Amazon services:（其他 Amazon 服务的用例：）下拉菜单中输入并搜索 IoT。选择下一步。
2. 在 Add permissions（添加权限）页面上，您将看到自动附加到服务角色的策略。选择下一步。
3. 在 Name, review, and create（命名、检查和创建）页面上，为此角色输入 Role name（角色名称）和 Role description（角色描述），然后选择 Create role（创建角色）。
4. 在角色列表中，找到您创建的角色，将其打开，然后复制要在您创建时使用的角色 ARN (*logging-role-arn*)在 Amazon IoT（控制台）中配置默认日志记录 (p. 396)。

日志记录角色策略

以下策略文档提供了角色策略和信任策略，借助这些策略，可Amazon IoT CloudWatch 代表您向发出了什么请求。如果您还允许Amazon IoT Core适用于 LoRa WAN 的提交日志条目，则您将看到一个为您创建的策略文档，其中将记录这两个活动。有关如何为 WA LoRa N 创建 IAM 日志角色和策略的信息，请参阅为 Amazon IoT Wireless 创建日志记录角色和策略 (p. 1249)。Amazon IoT Core

Note

在您创建日志记录角色时已为您创建这些文档。这些文档具有变量 *${partition}#${region}*、*${log_stream}* 和 *${accountId}*，必须用您的值替换这些变量。

角色策略：

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogGroup",
    "logs:CreateLogStream",
```

```
    "logs:PutLogEvents",
    "logs:PutMetricFilter",
    "logs:PutRetentionPolicy",
    "iot:GetLoggingOptions",
    "iot:SetLoggingOptions",
    "iot:SetV2LoggingOptions",
    "iot:GetV2LoggingOptions",
    "iot:SetV2LoggingLevel",
    "iot:ListV2LoggingLevels",
    "iot:DeleteV2LoggingLevel"
   ],
   "Resource": [
    "arn:${partition}:logs:${region}:${accountId}:log-group::${log-stream}:",
    "arn:${partition}:logs:${region}:${accountId}:log-group::/aws/iot/"
   ]
  }
 ]
}
```

信任策略仅记录 Amazon IoT Core 活动：

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

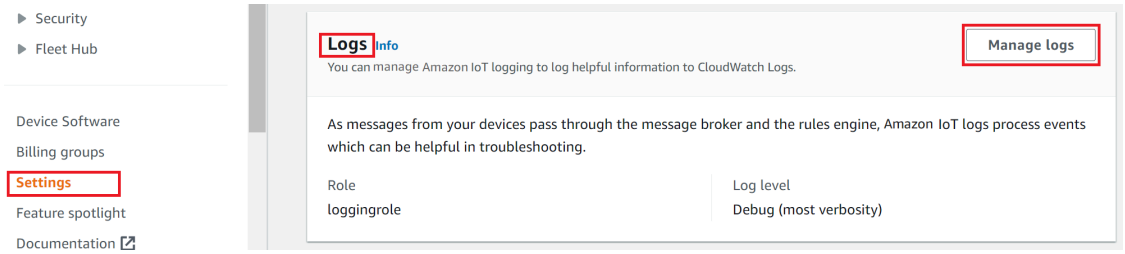
在 Amazon IoT（控制台）中配置默认日志记录

本部分介绍如何使用 Amazon IoT 控制台为所有 Amazon IoT 配置日志记录。要仅为特定事物组配置日志记录，您必须使用 CLI 或 API。有关为特定事物组配置日志记录的信息，请参阅在 Amazon IoT 中配置资源特定的日志记录 (CLI) (p. 398)。

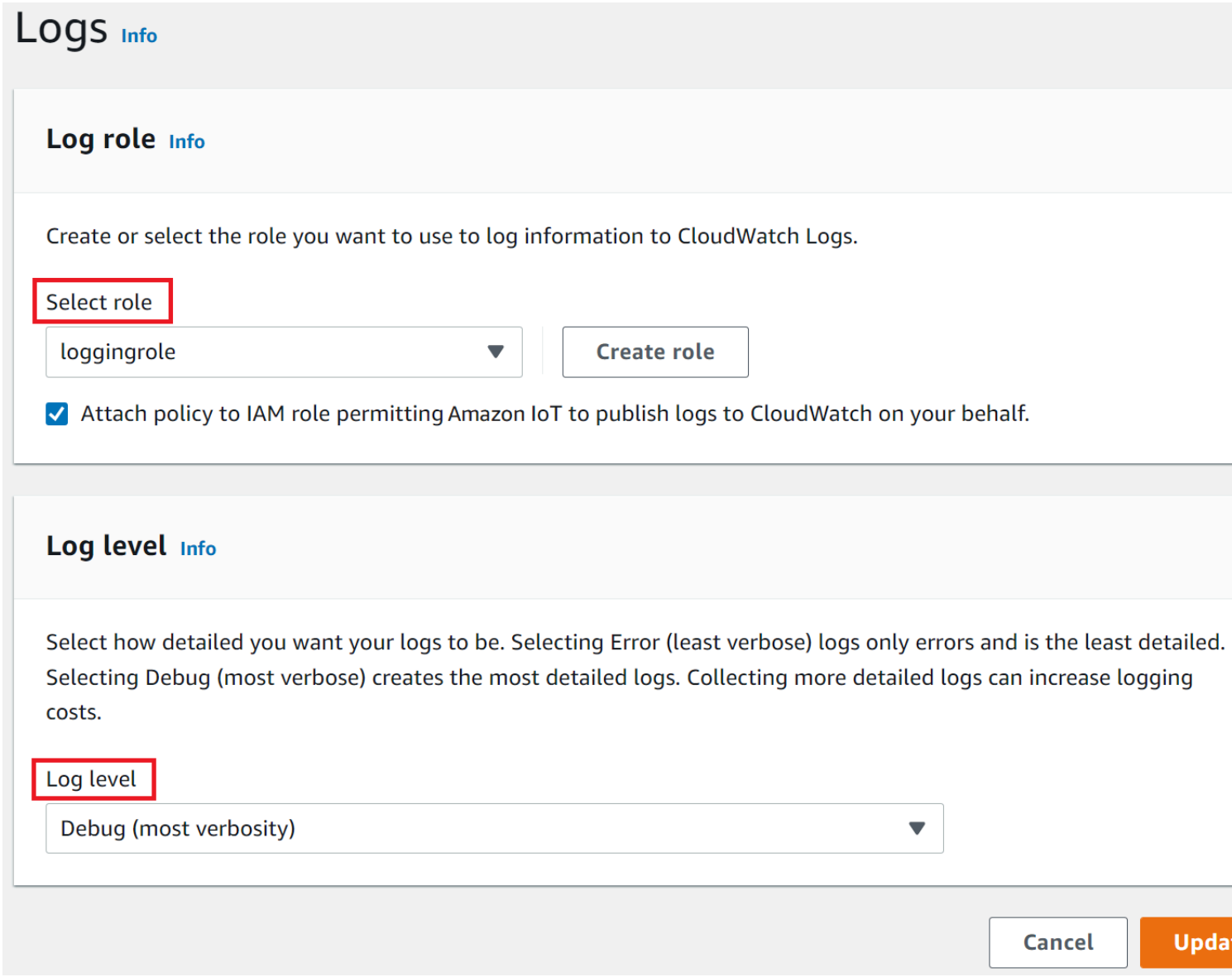
使用 Amazon IoT 控制台为所有 Amazon IoT 配置默认日志记录

1. 登录 Amazon IoT 控制台。有关更多信息，请参阅打开 Amazon IoT 控制台 (p. 17)：
2. 在左侧导航窗格中，选择 Settings (设置)。在 Settings（设置）页面的 Logs（日志）部分中，选择 Manage logs（管理日志）。

   Logs（日志）页面显示 Amazon IoT 使用的日志记录角色和详细程度级别。



3. 在 Logs（日志）页面上，选择 Select role（选择角色）以指定您在创建日志记录角色 (p. 395)中创建的角色，或选择 Create Role（创建角色）以创建用于日志记录的新角色。



4. 选择 Log le vel (详细程度级别 (p. 400)) ，该级别能够描述要显示在日志中的 CloudWatch 日志条目的详细信息级别。
5. 单击 Update (更新) 以保存您的更改。

启用日志记录后，请访问 在 CloudWatch 控制台中查看Amazon IoT日志 (p. 413) 以了解有关查看日志条目的更多信息。

在 Amazon IoT 中配置默认日志记录 (CLI)

本部分介绍如何使用 CLI 为 Amazon IoT 配置全局日志记录。

Note

您需要提供要使用角色的 Amazon Resource Name (ARN)。如果需要创建用于日志记录的角色，请参阅 创建日志记录角色 (p. 395)，然后继续操作。

用于调用 API 的委托人对于您的日志记录角色必须具有传递角色权限 (p. 444)。

您也可以使用 API 执行此流程，即使用 Amazon API 中与此处显示的 CLI 命令对应的方法。

使用 CLI 为 Amazon IoT 配置默认日志记录

1. 使用 set-v2-logging-options 命令为您的账户设置日志记录选项。

```
aws iot set-v2-logging-options \
    --role-arn logging-role-arn \
    --default-log-level log-level
```

其中：

--role-arn

向授予在 Logs 中写入日志的Amazon IoT权限的 CloudWatch 角色 ARN。

--default-log-level

要使用的日志级别 (p. 400)。有效值为：ERROR、WARN、INFO、DEBUG 或 DISABLED。

--no-disable-all-logs

启用所有 Amazon IoT 日志记录的可选参数。使用此参数可在当前禁用日志记录时启用日志记录。

--disable-all-logs

禁用所有 Amazon IoT 日志记录的可选参数。使用此参数可在当前启用日志记录时禁用日志记录。

2. 使用 get-v2-logging-options 命令获取当前日志记录选项。

```
aws iot get-v2-logging-options
```

启用日志记录后，请访问 在 CloudWatch 控制台中查看Amazon IoT日志 (p. 413) 以了解有关查看日志条目的更多信息。

Note

Amazon IoT 会继续支持使用较早的命令（set-logging-options 和 get-logging-options）在您的账户中设置和获取全局日志记录。请注意，使用这些命令时，生成的日志将包含纯文本而不是 JSON 负载，并且日志记录延迟通常会更高。将不会再对这些较早命令的实施进行更多改进。建议您使用“v2”版本来配置日志记录选项，如果可能，还请更改使用早期版本的传统应用程序。

在 Amazon IoT 中配置资源特定的日志记录 (CLI)

本部分介绍如何使用 CLI 为 Amazon IoT 配置资源特定的日志记录。可通过资源特定的日志记录为特定事物组 (p. 246)指定日志记录级别。

事物组可包含其它事物组以创建分层关系。此流程介绍如何配置单个事物组的日志记录。您可以将此流程应用于层次结构中的父事物组，以配置层次结构中所有事物组的日志记录。您也可以将此流程应用于子事情组，以覆盖其父级的日志记录配置。

除了事物组之外，您还可以录入目标，例如设备的客户端 ID、源 IP 和主体 ID。

Note

您需要提供要使用角色的 Amazon Resource Name (ARN)。如果需要创建用于日志记录的角色，请参阅 创建日志记录角色 (p. 395)，然后继续操作。
用于调用 API 的委托人对于您的日志记录角色必须具有传递角色权限 (p. 444)。

您也可以使用 API 执行此流程，即使用 Amazon API 中与此处显示的 CLI 命令对应的方法。

使用 CLI 为 Amazon IoT 配置资源特定的日志记录

1. 使用 set-v2-logging-options 命令为您的账户设置日志记录选项。

```
aws iot set-v2-logging-options \
    --role-arn logging-role-arn \
    --default-log-level log-level
```

其中：

--role-arn

向授予在 Logs 中写入日志的Amazon IoT权限的 CloudWatch 角色 ARN。

--default-log-level

要使用的日志级别 (p. 400)。有效值为：ERROR、WARN、INFO、DEBUG 或 DISABLED。

--no-disable-all-logs

启用所有 Amazon IoT 日志记录的可选参数。使用此参数可在当前禁用日志记录时启用日志记录。

--disable-all-logs

禁用所有 Amazon IoT 日志记录的可选参数。使用此参数可在当前启用日志记录时禁用日志记录。

2. 使用 set-v2-logging-level 命令为事物组配置资源特定的日志记录。

```
aws iot set-v2-logging-level \
             --log-target targetType=THING_GROUP,targetName=thing_group_name \
             --log-level log_level
```

--log-target

您要配置日志记录的资源的类型和名称。target_type 值必须是下列项之一：THING_GROUP | CLIENT_ID | SOURCE_IP | PRINCIPAL_ID。log-target 参数值可以是文本（如前面的命令示例所示），也可以是 JSON 字符串，如以下示例所示。

```
aws iot set-v2-logging-level \
             --log-target '{"targetType": "THING_GROUP","targetName":
 "thing_group_name"}' \
             --log-level log_level
```

--log-level

为指定资源生成日志时使用的日志记录级别。有效值为：DEBUG、INFO、ERROR、WARN 和 DISABLED

```
aws iot set-v2-logging-level \
             --log-target targetType=CLIENT_ID,targetName=ClientId1 \
             --log-level DEBUG
```

3. 使用 list-v2-logging-levels 命令列出当前配置的日志记录级别。

```
aws iot list-v2-logging-levels
```

4. 使用 delete-v2-logging-level 命令删除资源特定的日志记录级别，比如下面的例子。

```
aws iot delete-v2-logging-level \
```

```
            --target-type "THING_GROUP" \
            --target-name "thing_group_name"
```

```
aws iot delete-v2-logging-level \
            --target-type=CLIENT_ID
            --target-name=ClientId1
```

--targetType

`target_type` 值必须是下列项之一：`THING_GROUP` | `CLIENT_ID` | `SOURCE_IP` | `PRINCIPAL_ID`。

--targetName

要删除日志记录级别的事物组的名称。

启用日志记录后，请访问 在 CloudWatch 控制台中查看Amazon IoT日志 (p. 413) 以了解有关查看日志条目的更多信息。

日志级别

这些日志级别确定记录的事件，并应用于默认日志级别和资源特定的日志级别。

ERROR (错误)

导致操作失败的任何错误。

日志仅包含 ERROR 信息。

WARN

可能导致系统中出现不一致问题，但不会导致操作失败的所有情况。

日志包括 ERROR 和 WARN 信息。

INFO

有关事物流的高级别信息。

日志包括 INFO、ERROR 和 WARN 信息。

DEBUG

可能有助于调试问题的信息。

日志包括 DEBUG、INFO、ERROR 和 WARN 信息。

DISABLED

所有日志记录均处于禁用状态。

使用 Amazon 监控Amazon IoT告警和指标 CloudWatch

您可以Amazon IoT使用监控 CloudWatch，此工具可从收集原始数据，Amazon IoT并将数据处理为易读的近乎实时的指标。这些统计数据会保存两周，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。预设情况下，每隔一分钟都会自动 CloudWatch 将Amazon IoT指标数据发送到。有关更多信息，请参阅什么是亚马逊 CloudWatch、亚马逊 CloudWatch 事件和亚马逊 CloudWatch 日志？在亚马逊 CloudWatch 用户指南中。

使用 Amazon IoT 指标

Amazon IoT 报告的指标为您提供了可通过不同方式分析的信息。以下使用案例基于您每天将十个事物连接一次 Internet 的场景。每天：

- 十个事物几乎在同一时间连接到 Amazon IoT。
- 每个事物都会订阅主题筛选条件，接着在等待一个小时后断开连接。在此期间，事物之间相互通信并探索更多与环境状态有关的信息。
- 每个事物都会基于其新发现的数据，使用 `UpdateThingShadow` 发布一些看法，
- 每个事物都与 Amazon IoT 断开连接。

为了帮助您入门，这些主题探讨了您可能会遇到的一些问题。

- 我如何在事物每天没有成功建立连接时得到通知？ (p. 401)
- 我如何在事物每天没有发布数据时得到通知？ (p. 402)
- 我如何在事物的影子更新每天被拒绝时得到通知？ (p. 402)
- 如何为 Jobs 创建 CloudWatch 告警？ (p. 403)

关于 CloudWatch 告警和指标的更多信息

- 创建要监控的 CloudWatch 警报Amazon IoT (p. 401)
- Amazon IoT 指标和维度 (p. 404)

创建要监控的 CloudWatch 警报Amazon IoT

您可以创建 CloudWatch 警报，用于在警报改变状态时发送 Amazon SNS 消息。警报会在您规定的时间范围内监控某一项指标。当指标的值在多个时间段内超过给定阈值时，将执行一项或多项操作。操作是一个发送到 Amazon SNS 主题或 Auto Scaling 策略的通知。告警只会调用操作进行持续的状态变更。 CloudWatch 警报将不会简单地调用操作，因为这些操作均处于特定状态，该状态必须改变并在指定数量的时间段内一直保持。

以下主题介绍了使用 CloudWatch 警报的一些示例。

- 我如何知道事物每天是否成功建立连接？ (p. 401)
- 我如何在事物每天没有发布数据时得到通知？ (p. 402)
- 我如何在事物的影子更新每天被拒绝时得到通知？ (p. 402)
- 如何为 Jobs 创建 CloudWatch 告警？ (p. 403)

您可以在查看 CloudWatch 告警可以监控的所有指标Amazon IoT 指标和维度 (p. 404)。

我如何知道事物每天是否成功建立连接？

1. 创建名为 `things-not-connecting-successfully` 的 Amazon SNS 主题，并记录其 Amazon Resource Name (ARN)。此流程将您主题的 ARN 称为 *`sns-topic-arn`*。

   有关如何创建 Amazon SNS 通知的更多信息，请参阅 Amazon SNS 入门。
2. 创建告警。

```
aws cloudwatch put-metric-alarm \
    --alarm-name ConnectSuccessAlarm \
```

```
    --alarm-description "Alarm when my Things don't connect successfully" \
    --namespace AWS/IoT \
    --metric-name Connect.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. 测试告警。

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name ConnectSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. 验证警报是否显示在 CloudWatch 控制台中。

我如何在事物每天没有发布数据时得到通知？

1. 创建名为 `things-not-publishing-data` 的 Amazon SNS 主题，并记录其 Amazon Resource Name (ARN)。此流程将您主题的 ARN 称为 *sns-topic-arn*。

   有关如何创建 Amazon SNS 通知的更多信息，请参阅 Amazon SNS 入门。

2. 创建告警。

```
aws cloudwatch put-metric-alarm \
    --alarm-name PublishInSuccessAlarm\
    --alarm-description "Alarm when my Things don't publish their data \
    --namespace AWS/IoT \
    --metric-name PublishIn.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. 测试告警。

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name PublishInSuccessAlarm --state-reason
 "initializing" --state-value ALARM
```

4. 验证警报是否显示在 CloudWatch 控制台中。

我如何在事物的影子更新每天被拒绝时得到通知？

1. 创建名为 `things-shadow-updates-rejected` 的 Amazon SNS 主题，并记录其 Amazon Resource Name (ARN)。此流程将您主题的 ARN 称为 *sns-topic-arn*。

有关如何创建 Amazon SNS 通知的更多信息，请参阅 Amazon SNS 入门。

2. 创建告警。

```
aws cloudwatch put-metric-alarm \
    --alarm-name UpdateThingShadowSuccessAlarm \
    --alarm-description "Alarm when my Things Shadow updates are getting rejected" \
    --namespace AWS/IoT \
    --metric-name UpdateThingShadow.Success \
    --dimensions Name=Protocol,Value=MQTT \
    --statistic Sum \
    --threshold 10 \
    --comparison-operator LessThanThreshold \
    --period 86400 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions sns-topic-arn
```

3. 测试告警。

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value OK
```

```
aws cloudwatch set-alarm-state --alarm-name UpdateThingShadowSuccessAlarm --state-
reason "initializing" --state-value ALARM
```

4. 验证警报是否显示在 CloudWatch 控制台中。

如何为 Jobs 创建 CloudWatch 告警？

Jobs 服务为您提供了监控任务的 CloudWatch 指标。您可以创建 CloudWatch 警报来监控任意 任务指标 (p. 409)。

以下命令创建 CloudWatch 警报，以监控任务 *SampleOTAJob #####*执行总数，并在失败任务执行数超过 20 时通知您。警报通过每 300 秒检查报告值来监控 Jobs 指标 `FailedJobExecutionTotalCount`。它在单个报告值大于 20 时激活，这意味着自任务启动以来失败的任务执行数超过了 20。当告警关闭时，它会向提供的 Amazon SNS 主题发送通知。

```
aws cloudwatch put-metric-alarm \
    --alarm-name TotalFailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when total number of failed job execution exceeds the
 threshold for SampleOTAJob" \
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionTotalCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 20 \
    --comparison-operator GreaterThanThreshold \
    --period 300 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-many-
failed-job-ececutions
```

以下命令创建 CloudWatch 警报，以监控任务 *SampleOTAJob ##########*执行数。然后，在该时段内有超过 5 个任务执行失败时，它会通知您。警报通过每 3600 秒检查报告值来监控 Jobs 指标 `FailedJobExecutionCount`。它在单个报告值大于 5 时激活，这意味着过去 1 小时内失败的任务执行数超过了 5。当告警关闭时，它会向提供的 Amazon SNS 主题发送通知。

```
aws cloudwatch put-metric-alarm \
    --alarm-name FailedJobExecution-SampleOTAJob \
    --alarm-description "Alarm when number of failed job execution per hour exceeds the
 threshold for SampleOTAJob" \
    --namespace AWS/IoT \
    --metric-name FailedJobExecutionCount \
    --dimensions Name=JobId,Value=SampleOTAJob \
    --statistic Sum \
    --threshold 5 \
    --comparison-operator GreaterThanThreshold \
    --period 3600 \
    --unit Count \
    --evaluation-periods 1 \
    --alarm-actions arn:aws:sns:<AWS_REGION>:<AWS_ACCOUNT_ID>:SampleOTAJob-has-too-many-
failed-job-ececutions-per-hour
```

Amazon IoT 指标和维度

当您与 Amazon IoT 交互时，此服务每一分钟都会向 CloudWatch 发送以下指标和维度。您可以使用以下流程查看 的指标Amazon IoT

要查看指标（CloudWatch 控制台）

指标的分组首先依据服务命名空间，然后依据每个命名空间内的各种维度组合。

1. 打开 CloudWatch 控制台。
2. 在导航窗格中，选择 Metrics（指标）然后选择 All metrics（所有指标）。
3. 在浏览选项卡，搜索 Amazon IoT 查看指标列表。

查看指标 (CLI)

- 在命令提示符处输入下面的命令：

```
aws cloudwatch list-metrics --namespace "Amazon/IoT"
```

CloudWatch 显示的以下指标组Amazon IoT：

- Amazon IoT 指标 (p. 405)
- Amazon IoT Core 凭证提供程序指标 (p. 405)
- 规则指标 (p. 405)
- 规则操作指标 (p. 405)
- HTTP 操作特定指标 (p. 406)
- 消息代理指标 (p. 406)
- 设备影子指标 (p. 408)
- 任务指标 (p. 409)
- Device Defender audit 指标 (p. 410)
- Device Defender detect 指标 (p. 410)
- 设备预配置指标 (p. 410)
- 机群索引指标 (p. 412)
- 指标的维度 (p. 412)

Amazon IoT 指标

指标	描述
AddThingToDynamicThingGroupsFailed	与将事物添加到动态事物组相关联的失败事件数。DynamicThingGroupName 维度包含添加事物时失败的动态组的名称。
NumLogBatchesFailedToPublishThrottled	因限制错误而无法发布的日志事件的单个批次。
NumLogEventsFailedToPublishThrottled	批处理中的因限制错误而无法发布的日志事件的数量。

Amazon IoT Core 凭证提供程序指标

指标	描述
CredentialExchangeSuccess	向 Amazon IoT Core 凭证提供程序成功提交 AssumeRoleWithCertificate 请求的数量。

规则指标

指标	描述
ParseError	在规则所侦听的主题上发布的消息中出现的 JSON 分析错误的数量。RuleName 维度包含规则的名称。
RuleMessageThrottled	由于恶意行为或由于消息数超过了规则引擎的限制，规则引擎限制的消息数。RuleName 维度包含要触发的规则的名称。
RuleNotFound	找不到要触发的规则。RuleName 维度包含规则的名称。
RulesExecuted	已执行的 Amazon IoT 规则的数量。
TopicMatch	已在规则所侦听的主题上发布的传入消息的数量。RuleName 维度包含规则的名称。

规则操作指标

指标	描述
Failure	失败的规则操作调用的数量。RuleName 维度包含指定操作的规则的名称。ActionType 维度包含已调用的操作的类型。
Success	成功的规则操作调用的数量。RuleName 维度包含指定操作的规则的名称。ActionType 维度包含已调用的操作的类型。
ErrorActionFailure	失败的错误操作数量。RuleName 维度包含指定操作的规则的名称。ActionType 维度包含已调用的操作的类型。

指标	描述
ErrorActionSuccess	成功的错误操作数量。RuleName 维度包含指定操作的规则的名称。ActionType 维度包含已调用的操作的类型。

HTTP 操作特定指标

指标	描述
HttpCode_Other	如果来自下游 Web 服务/应用程序的响应的状态代码不是 2xx、4xx 或 5xx，则生成此指标。
HttpCode_4XX	如果来自下游 Web 服务/应用程序的响应的状态代码介于 400 和 499 之间，则生成此指标。
HttpCode_5XX	如果来自下游 Web 服务/应用程序的响应的状态代码介于 500 和 599 之间，则生成此指标。
HttpInvalidUrl	如果在替换模板后的终端节点 URL 不以 https:// 开头，则生成此指标。
HttpRequestTimeout	如果下游 Web 服务/应用程序未在请求超时限制内返回响应，则生成此指标。有关更多信息，请参阅 Service Quotas。
HttpUnknownHost	如果 URL 有效，但服务不存在或无法访问，则生成此指标。

消息代理指标

Note

消息代理指标显示在 CloudWatch 控制台中的 Protocol Metrics（协议指标）的下方。

指标	描述
Connect.AuthError	无法由消息代理授权的连接请求的数量。Protocol 维度包含用于发送 CONNECT 消息的协议。
Connect.ClientError	因 MQTT 消息未能满足Amazon IoT 配额 (p. 1301)中定义的要求而被拒绝的连接请求的数量。Protocol 维度包含用于发送 CONNECT 消息的协议。
Connect.ClientIDThrottle	因该客户端超出指定客户端 ID 所允许的连接请求速率而受限的连接请求的数量。Protocol 维度包含用于发送 CONNECT 消息的协议。
Connect.ServerError	因出现内部错误而导致失败的连接请求的数量。Protocol 维度包含用于发送 CONNECT 消息的协议。
Connect.Success	与消息代理的成功连接的数量。Protocol 维度包含用于发送 CONNECT 消息的协议。

指标	描述
Connect.Throttle	因账户超出允许的连接请求速率而受限的连接请求的数量。Protocol 维度包含用于发送 CONNECT 消息的协议。
Ping.Success	消息代理收到的 ping 消息的数量。Protocol 维度包含用于发送 ping 消息的协议。
PublishIn.AuthError	消息代理无法授权的发布请求的数量。Protocol 维度包含用于发布消息的协议。HTTP 发布不支持此指标。
PublishIn.ClientError	因消息未能满足Amazon IoT 配额 (p. 1301)中定义的要求而被消息代理拒绝的发布请求的数量。Protocol 维度包含用于发布消息的协议。HTTP 发布不支持此指标。
PublishIn.ServerError	因出现内部错误而导致消息代理无法处理的发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。HTTP 发布不支持此指标。
PublishIn.Success	消息代理已成功处理的发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。
PublishIn.Throttle	因客户端超出允许的入站消息速率而受限制的发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。HTTP 发布不支持此指标。
PublishOut.AuthError	Amazon IoT 无法授权的由消息代理发出的发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。
PublishOut.ClientError	因消息未能满足Amazon IoT 配额 (p. 1301)中定义的要求而被拒绝的由消息代理发出的发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。
PublishOut.Success	消息代理已成功发出的发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。
PublishOut.Throttle	因客户端超出允许的入站消息速率而节流的发布请求数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。
PublishRetained.AuthError	消息代理无法授权的 RETAIN 标签集发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。
PublishRetained.ServerError	因出现内部错误而导致消息代理无法处理的保留发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。
PublishRetained.Success	消息代理已成功处理的 RETAIN 标签集发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。
PublishRetained.Throttle	因客户端超出允许的入站消息速率而受限制的 RETAIN 标签集发布请求的数量。Protocol 维度包含用于发送 PUBLISH 消息的协议。

指标	描述
Subscribe.AuthError	无法授权的由客户端发出的订阅请求的数量。Protocol 维度包含用于发送 SUBSCRIBE 消息的协议。
Subscribe.ClientError	因 SUBSCRIBE 消息未能满足 Amazon IoT 配额 (p. 1301) 中定义的要求而被拒绝的订阅请求的数量。Protocol 维度包含用于发送 SUBSCRIBE 消息的协议。
Subscribe.ServerError	因出现内部错误而被拒绝的订阅请求的数量。Protocol 维度包含用于发送 SUBSCRIBE 消息的协议。
Subscribe.Success	消息代理已成功处理的订阅请求的数量。Protocol 维度包含用于发送 SUBSCRIBE 消息的协议。
Subscribe.Throttle	因客户端超出允许的订阅请求速率而受限的订阅请求的数量。Protocol 维度包含用于发送 SUBSCRIBE 消息的协议。
Unsubscribe.ClientError	因 UNSUBSCRIBE 消息未能满足 Amazon IoT 配额 (p. 1301) 中定义的要求而被拒绝的取消订阅请求的数量。Protocol 维度包含用于发送 UNSUBSCRIBE 消息的协议。
Unsubscribe.ServerError	因出现内部错误而被拒绝的取消订阅请求的数量。Protocol 维度包含用于发送 UNSUBSCRIBE 消息的协议。
Unsubscribe.Success	消息代理已成功处理的取消订阅请求的数量。Protocol 维度包含用于发送 UNSUBSCRIBE 消息的协议。
Unsubscribe.Throttle	因客户端超出允许的取消订阅请求速率而被拒绝的取消订阅请求的数量。Protocol 维度包含用于发送 UNSUBSCRIBE 消息的协议。

设备影子指标

Note

设备影子指标显示在 CloudWatch 控制台中的 Protocol Metrics（协议指标）的下方

指标	描述
DeleteThingShadow.Accepted	已成功处理的 DeleteThingShadow 请求的数量。Protocol 维度包含用于发出请求的协议。
GetThingShadow.Accepted	已成功处理的 GetThingShadow 请求的数量。Protocol 维度包含用于发出请求的协议。
ListThingShadow.Accepted	已成功处理的 ListThingShadow 请求的数量。Protocol 维度包含用于发出请求的协议。
UpdateThingShadow.Accepted	已成功处理的 UpdateThingShadow 请求的数量。Protocol 维度包含用于发出请求的协议。

任务指标

指标	描述
CanceledJobExecutionCount	在由确定的时间段CANCELED内，其状态已更改为的任务执行的数量 CloudWatch。（有关 CloudWatch 指标的更多信息，请参阅亚马逊 CloudWatch指标。）JobId 维度包含任务的 ID。
CanceledJobExecutionTotalCount	给定任务的状态为 CANCELED 的任务执行总数。JobId 维度包含任务的 ID。
ClientErrorCount	执行任务时生成的客户端错误数。JobId 维度包含任务的 ID。
FailedJobExecutionCount	在由确定的时间段FAILED内，其状态已更改为的任务执行的数量 CloudWatch。（有关 CloudWatch 指标的更多信息，请参阅亚马逊 CloudWatch 指标。）JobId 维度包含任务的 ID。
FailedJobExecutionTotalCount	给定任务的状态为 FAILED 的任务执行总数。JobId 维度包含任务的 ID。
InProgressJobExecutionCount	在由确定的时间段IN_PROGRESS内，其状态已更改为的任务执行的数量 CloudWatch。（有关 CloudWatch 指标的更多信息，请参阅亚马逊 CloudWatch指标。）JobId 维度包含任务的 ID。
InProgressJobExecutionTotalCount	给定任务的状态为 IN_PROGRESS 的任务执行总数。JobId 维度包含任务的 ID。
RejectedJobExecutionTotalCount	给定任务的状态为 REJECTED 的任务执行总数。JobId 维度包含任务的 ID。
RemovedJobExecutionTotalCount	给定任务的状态为 REMOVED 的任务执行总数。JobId 维度包含任务的 ID。
QueuedJobExecutionCount	在由确定的时间段QUEUED内，其状态已更改为的任务执行的数量 CloudWatch。（有关 CloudWatch 指标的更多信息，请参阅亚马逊 CloudWatch 指标。）JobId 维度包含任务的 ID。
QueuedJobExecutionTotalCount	给定任务的状态为 QUEUED 的任务执行总数。JobId 维度包含任务的 ID。
RejectedJobExecutionCount	在由确定的时间段REJECTED内，其状态已更改为的任务执行的数量 CloudWatch。（有关 CloudWatch 指标的更多信息，请参阅亚马逊 CloudWatch指标。）JobId 维度包含任务的 ID。
RemovedJobExecutionCount	在由确定的时间段REMOVED内，其状态已更改为的任务执行的数量 CloudWatch。（有关 CloudWatch 指标的更多信息，请参阅亚马逊 CloudWatch指标。）JobId 维度包含任务的 ID。
ServerErrorCount	执行任务时生成的服务器错误数。JobId 维度包含任务的 ID。

指标	描述
SuccededJobExecutionCount	在由确定的时间段SUCCESS内，其状态已更改为的任务执行的数量 CloudWatch。（有关 CloudWatch 指标的更多信息，请参阅亚马逊 CloudWatch指标。）JobId 维度包含任务的 ID。
SuccededJobExecutionTotalCount	给定任务的状态为 SUCCESS 的任务执行总数。JobId 维度包含任务的 ID。

Device Defender audit 指标

指标	描述
NonCompliantResources	在检查中发现的不合规资源数。系统报告执行的每次审核中，对于每次检查发现的不合规资源数。
ResourcesEvaluated	执行合规性评估的资源数。系统报告执行的每次审核中，对于每次检查所评估的资源数。

Device Defender detect 指标

指标	描述
Violations	自上次执行评估以来，所发现的安全配置文件行为的新违规数。系统针对特定安全配置文件，以及针对某个特定安全配置文件的特定行为，报告账户的新违规数。
ViolationsCleared	自上次执行评估以来，已解决的安全配置文件行为的违规数。系统针对特定安全配置文件，以及针对某个特定安全配置文件的特定行为，报告账户的已解决违规数。
ViolationsInvalidated	自上次执行评估以来，其信息不再可用（由于报告设备已停止报告，或者由于某个原因不再监控）的安全配置文件行为的违规数。系统针对特定安全配置文件，以及针对某个特定安全配置文件的特定行为，报告整个账户的已失效违规数。

设备预配置指标

Amazon IoT Fleet 预置指标

指标	描述
ApproximateNumberOfThingsRegistered	Fleet Provisioning 注册的事物的计数。 虽然计数通常是准确的，但 Amazon IoT Core 的分布式架构使得很难让已注册事物保持精确计数。 此指标将使用的统计数据是：

指标	描述
	• Max（最大值），用以报告已注册事物的总数。有关在 CloudWatch聚合窗口中注册的事物计数，请参阅RegisterThingFailed指标。 维度：ClaimCertificateId (p. 412)
CreateKeysAndCertificateFailed	调用 CreateKeysAndCertificate MQTT API 时故障的次数。 在“成功”（值 = 0）和“故障”（值 = 1）情况下均发出该指标。此指标可用于跟踪在 CloudWatch支持的聚合窗口期间（例如 5 分钟或 1 小时）创建和注册的证书数量。 此指标的可用统计数据包括： • Sum（总计），用以报告失败的调用数。 • SampleCount报告成功和失败的调用总数。
CreateCertificateFromCsrFailed	调用 CreateCertificateFromCsr MQTT API 时故障的次数。 在“成功”（值 = 0）和“故障”（值 = 1）情况下均发出该指标。此指标可用于跟踪在 CloudWatch支持的聚合窗口期间（例如 5 分钟或 1 小时）注册的事物数。 此指标的可用统计数据包括： • Sum（总计），用以报告失败的调用数。 • SampleCount报告成功和失败的调用总数。
RegisterThingFailed	调用 RegisterThing MQTT API 时故障的次数。 在“成功”（值 = 0）和“故障”（值 = 1）情况下均发出该指标。此指标可用于跟踪在 CloudWatch支持的聚合窗口期间（例如 5 分钟或 1 小时）注册的事物数。有关注册的事物总数，请参阅 ApproximateNumberOfThingsRegistered 指标。 此指标的可用统计数据包括： • Sum（总计），用以报告失败的调用数。 • SampleCount，拥以报告成功和失败的调用总数。 维度：TemplateName (p. 412)

Just-in-time 预置指标

指标	描述
ProvisionThing.ClientError	由于客户端错误无法预置设备的次数。例如，模板中指定的策略不存在。

指标	描述
ProvisionThing.ServerError	由于服务器错误而无法预置设备的次数。客户可以在等待后重新尝试预置设备，如果问题仍然存在，则可以尝试联系 Amazon IoT。
ProvisionThing.Success	成功预置设备的次数。

机群索引指标

Amazon IoT 机群索引指标

指标	描述
NamedShadowCountForDynamicGroupQueryLimit	对于不是动态事物组中特定于数据来源的查询术语，每件事物最多处理 25 个命名影子。由于事件而违反该限制时，将发出 NamedShadowCountForDynamicGroupQueryLimitExceeded 事件类型。

指标的维度

指标使用命名空间并为以下维度提供指标

维度	描述
ActionType	触发请求的规则所指定的操作类型 (p. 449)。
BehaviorName	正受监控的 Device Defender Detect 安全配置文件行为的名称。
ClaimCertificateId	用于预置设备的申请的 certificateId。
CheckName	正在监控其结果的 Device Defender Audit 检查的名称。
JobId	正在监控其进度或消息连接成功/失败的任务的 ID。
Protocol	用于提出请求的协议。有效值为：MQTT 或 HTTP
RuleName	由请求触发的规则的名称。
ScheduledAuditName	正在监控其检查结果的 Device Defender 计划审核的名称。如果报告的结果适用于按需执行的审计，此项具有值 OnDemand。
SecurityProfileName	正在监控其行为的 Device Defender Detect 安全配置文件的名称。
TemplateName	预配置模板的名称。

Amazon IoT使用 CloudWatch 日志进行监控

如果启用 Amazon IoT 日志记录 (p. 394)，则当消息从您的设备通过消息代理和规则引擎传递时，Amazon IoT 会发送关于每条消息的进度事件。在CloudWatch 控制台中， CloudWatch 日志出现在名为的日志组中AWSIotLogs。

有关 CloudWatch 日志的更多信息，请参阅CloudWatch 日志。有关支持Amazon IoT CloudWatch 的日志的信息，请参阅CloudWatch Amazon IoT日志条目 (p. 413)。

在 CloudWatch 控制台中查看Amazon IoT日志

Note

在以下情况之前，AWSIotLogsV2日志组不会在 CloudWatch 控制台中显示：

- 您已在 Amazon IoT 中启用日志记录。有关如何在 Amazon IoT 中启用日志记录的详细信息，请参阅 配置 Amazon IoT 日志记录 (p. 394)。
- 一些日志条目已由 Amazon IoT 操作编写。

在 CloudWatch 控制台中查看您的Amazon IoT日志

1. 浏览到 https://console.aws.amazon.com/cloudwatch/。在导航窗格中，选择 Log groups（日志组）。
2. 在筛选条件文本框中，输入 **AWSIotLogsV2**，然后按 Enter。
3. 双击 AWSIotLogsV2 日志组。
4. 选择 Search All（搜索全部）。显示为您的账户生成的 Amazon IoT 日志的完整列表。
5. 选择展开图标可查看单个流。

您也可以在筛选事件文本框中输入查询。可以尝试以下受到较多关注的查询：

- `{ $.logLevel = "INFO" }`

  查找所有日志级别为 INFO 的日志。
- `{ $.status = "Success" }`

  查找所有状态为 Success 的日志。
- `{ $.status = "Success" && $.eventType = "GetThingShadow" }`

  查找所有状态为 Success 且事件类型为 GetThingShadow 的日志。

有关创建筛选条件表达式的更多信息，请参阅CloudWatch 日志查询。

CloudWatch Amazon IoT日志条目

Amazon IoT 的每个组件均会生成自己的日志条目。每个日志条目都有一个 eventType 指定导致生成日志条目的操作。本部分介绍以下 Amazon IoT 组件所生成的日志条目。有关Amazon IoT Core LoRa WAN 监控的信息，请参见查看 CloudWatch Amazon IoT Wireless日志条目 (p. 1258)。

主题

- 消息代理日志条目 (p. 414)
- 设备影子日志条目 (p. 419)
- 规则引擎日志条目 (p. 420)
- 任务日志条目 (p. 425)
- 设备预配置日志条目 (p. 429)
- 动态事物组日志条目 (p. 430)
- 机群索引日志条目 (p. 431)
- 常 CloudWatch 日志属性 (p. 431)

消息代理日志条目

Amazon IoT 消息代理会生成以下事件的日志条目：

主题

- Connect 日志条目 (p. 414)
- Disconnect 日志条目 (p. 414)
- GetRetainedMessage 日志条目 (p. 415)
- ListRetainedMessage 日志条目 (p. 416)
- Publish-In 日志条目 (p. 416)
- Publish-Out 日志条目 (p. 417)
- Subscribe 日志条目 (p. 418)

Connect 日志条目

当 MQTT 客户端连接时，Amazon IoT 消息代理会生成一个 `eventType` 为 `Connect` 的日志条目。

Connect 日志条目示例

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Connect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，`Connect` 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

principalId

发出请求的委托人的 ID。

protocol

发出请求时使用的协议。有效值为 `MQTT` 或 `HTTP`。

sourceIp

请求的源 IP 地址。

sourcePort

请求的源端口。

Disconnect 日志条目

当 MQTT 客户端断开连接时，Amazon IoT 消息代理会生成一个 `eventType` 为 `Disconnect` 的日志条目。

Disconnect 日志条目示例

```
{
    "timestamp": "2017-08-10 15:37:23.476",
    "logLevel": "INFO",
    "traceId": "20b23f3f-d7f1-feae-169f-82263394fbdb",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Disconnect",
    "protocol": "MQTT",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490,
    "reason": "DUPLICATE_CLIENT_ID",
    "details": "A new connection was established with the same client ID",
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，Disconnect 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

principalId

发出请求的委托人的 ID。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

sourceIp

请求的源 IP 地址。

sourcePort

请求的源端口。

reason

客户端断开连接的原因。

details

错误的简要说明。

disconnectReason

客户端断开连接的原因。

GetRetainedMessage 日志条目

调用 GetRetainedMessage 时，Amazon IoT 消息代理会用 GetRetainedMessage 的 eventType 生成一个日志条目。

GetRetainedMessage 示例日志日志条目示例

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
```

```
    "status": "Success",
    "eventType": "GetRetainedMessage",
    "protocol": "HTTP",
    "topicName": "a/b/c",
    "qos": "1",
    "lastModifiedDate": "2017-08-07 18:47:56.664"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，GetRetainedMessage 日志条目还包含以下属性：

lastModifiedDate

Amazon IoT 储存保留消息时，Epoch 日期和时间（以毫秒为单位）存储保留的消息的时间。

protocol

发出请求时使用的协议。有效值：HTTP。

qos

发布请求中使用的服务质量 (QoS) 级别。有效值为 0 或 1。

topicName

已订阅主题的名称。

ListRetainedMessage 日志条目

调用 ListRetainedMessages 时，Amazon IoT 消息代理会用 ListRetainedMessage 的 eventType 生成一个日志条目。

ListRetainedMessage 示例日志日志条目示例

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ListRetainedMessage",
    "protocol": "HTTP"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，ListRetainedMessage 日志条目还包含以下属性：

protocol

发出请求时使用的协议。有效值：HTTP。

Publish-In 日志条目

当 Amazon IoT 消息代理收到 MQTT 消息时，它会生成一个 eventType 为 Publish-In 的日志条目。

Publish-In 日志条目示例

```
{
        "timestamp": "2017-08-10 15:39:30.961",
        "logLevel": "INFO",
        "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
        "accountId": "123456789012",
        "status": "Success",
```

```
        "eventType": "Publish-In",
        "protocol": "MQTT",
        "topicName": "$aws/things/MyThing/shadow/get",
        "clientId": "abf27092886e49a8a5c1922749736453",
        "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
        "sourceIp": "205.251.233.181",
        "sourcePort": 13490,
        "retain": "True"
    }
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，Publish-In 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

principalId

发出请求的委托人的 ID。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

保留

当消息的 REATIN 标志的值设置为 True 时使用的属性。如果消息未设置 REATIN 标志，则该属性不会显示在日志条目中。有关更多信息，请参阅MQTT 保留消息 (p. 81)：

sourceIp

请求的源 IP 地址。

sourcePort

请求的源端口。

topicName

已订阅主题的名称。

Publish-Out 日志条目

当消息代理发布 MQTT 消息时，它会生成一个 eventType 为 Publish-Out 的日志条目

Publish-Out 日志条目示例

```
{
    "timestamp": "2017-08-10 15:39:30.961",
    "logLevel": "INFO",
    "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Publish-Out",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/get",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，Publish-Out 日志条目还包含以下属性：

clientId

接收有关该 MQTT 主题的消息的订阅客户端的 ID。

principalId

发出请求的委托人的 ID。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

sourceIp

请求的源 IP 地址。

sourcePort

请求的源端口。

topicName

已订阅主题的名称。

Subscribe 日志条目

当 MQTT 客户端订阅主题时，Amazon IoT 消息代理会生成一个 eventType 为 Subscribe 的日志条目。

Subscribe 日志条目示例

```
{
    "timestamp": "2017-08-10 15:39:04.413",
    "logLevel": "INFO",
    "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "Subscribe",
    "protocol": "MQTT",
    "topicName": "$aws/things/MyThing/shadow/#",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "sourceIp": "205.251.233.181",
    "sourcePort": 13490
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，Subscribe 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

principalId

发出请求的委托人的 ID。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

sourceIp

请求的源 IP 地址。

sourcePort

请求的源端口。

topicName

已订阅主题的名称。

设备影子日志条目

Amazon IoT Device Shadow 服务会生成以下事件的日志条目：

主题

- DeleteThingShadow 日志条目 (p. 419)
- GetThingShadow 日志条目 (p. 419)
- UpdateThingShadow 日志条目 (p. 420)

DeleteThingShadow 日志条目

当收到删除设备影子的请求时，Device Shadow 服务会生成一个 `eventType` 为 `DeleteThingShadow` 的日志条目。

DeleteThingShadow 示例日志日志条目示例

```
{
    "timestamp": "2017-08-07 18:47:56.664",
    "logLevel": "INFO",
    "traceId": "1a60d02e-15b9-605b-7096-a9f584a6ad3f",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DeleteThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/delete"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，`DeleteThingShadow` 日志条目还包含以下属性：

deviceShadowName

要更新的影子的名称。

protocol

发出请求时使用的协议。有效值为 `MQTT` 或 `HTTP`。

topicName

发布请求时所基于的主题的名称。

GetThingShadow 日志条目

当收到影子的获取请求时，Device Shadow 服务会生成一个 `eventType` 为 `GetThingShadow` 的日志条目。

GetThingShadow 示例日志日志条目示例

```
{
    "timestamp": "2017-08-09 17:56:30.941",
    "logLevel": "INFO",
    "traceId": "b575f19a-97a2-cf72-0ed0-c64a783a2504",
```

```
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "MyThing",
    "topicName": "$aws/things/MyThing/shadow/get"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，GetThingShadow 日志条目还包含以下属性：

deviceShadowName

所请求的影子的名称。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

topicName

发布请求时所基于的主题的名称。

UpdateThingShadow 日志条目

当收到更新设备影子的请求时，Device Shadow 服务会生成一个 eventType 为 UpdateThingShadow 的日志条目。

UpdateThingShadow 示例日志日志条目示例

```
{
    "timestamp": "2017-08-07 18:43:59.436",
    "logLevel": "INFO",
    "traceId": "d0074ba8-0c4b-a400-69df-76326d414c28",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateThingShadow",
    "protocol": "MQTT",
    "deviceShadowName": "Jack",
    "topicName": "$aws/things/Jack/shadow/update"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，UpdateThingShadow 日志条目还包含以下属性：

deviceShadowName

要更新的影子的名称。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

topicName

发布请求时所基于的主题的名称。

规则引擎日志条目

Amazon IoT 规则引擎会生成以下事件的日志：

主题

- FunctionExecution 日志条目 (p. 421)

- RuleExecution 日志条目 (p. 421)
- RuleMatch 日志条目 (p. 422)
- RuleMessageThrottled 日志条目 (p. 423)
- RuleNotFound 日志条目 (p. 424)
- StartingRuleExecution 日志条目 (p. 424)

FunctionExecution 日志条目

当规则的 SQL 查询调用外部函数时，规则引擎会生成一个 `eventType` 为 `FunctionExecution` 的日志条目。在规则的操作向 Amazon IoT 或其它 Web 服务发出 HTTP 请求（例如，调用 `get_thing_shadow` 或 `machinelearning_predict`）时，将调用外部函数。

FunctionExecution 示例日志日志条目示例

```
{
    "timestamp": "2017-07-13 18:33:51.903",
    "logLevel": "DEBUG",
    "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
    "status": "Success",
    "eventType": "FunctionExecution",
    "clientId": "N/A",
    "topicName":"rules/test",
    "ruleName": "ruleTestPredict",
    "ruleAction": "MachinelearningPredict",
    "resources": {
        "ModelId": "predict-model"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，`FunctionExecution` 日志条目还包含以下属性：

clientId

`N/A` (针对 `FunctionExecution` 日志)。

principalId

发出请求的委托人的 ID。

resources

规则的操作所使用的资源的集合。

ruleName

匹配规则的名称。

topicName

已订阅主题的名称。

RuleExecution 日志条目

当 Amazon IoT 规则引擎触发规则的操作时，它会生成一个 `RuleExecution` 日志条目。

RuleExecution 示例日志日志条目示例

```
{
```

```
    "timestamp": "2017-08-10 16:32:46.070",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "resources": {
        "RepublishTopic": "rules/republish"
    },
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，RuleExecution 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

principalId

发出请求的委托人的 ID。

resources

规则的操作所使用的资源的集合。

ruleAction

所触发的操作的名称。

ruleName

匹配规则的名称。

topicName

已订阅主题的名称。

RuleMatch 日志条目

当消息代理收到与规则匹配的消息时，Amazon IoT 规则引擎会生成一个 eventType 为 RuleMatch 的日志条目。

RuleMatch 示例日志日志条目示例

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "INFO",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "RuleMatch",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，RuleMatch 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

principalId

发出请求的委托人的 ID。

ruleName

匹配规则的名称。

topicName

已订阅主题的名称。

RuleMessageThrottled 日志条目

当消息受到限制时，Amazon IoT 规则引擎会生成一个 `eventType` 为 `RuleMessageThrottled` 的日志条目。

RuleMessageThrottled 示例日志日志条目示例

```
{
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleMessageThrottled",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleExecutionThrottled",
    "details": "Message for Rule example_rule throttled"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，`RuleMessageThrottled` 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

details

错误的简要说明。

principalId

发出请求的委托人的 ID。

reason

字符串 “RuleMessageThrottled”。

ruleName

要触发的规则的名称。

topicName

已发布的主题的名称。

RuleNotFound 日志条目

当 Amazon IoT 规则引擎找不到具有给定名称的规则时，它会生成一个 `eventType` 为 `RuleNotFound` 的日志条目。

RuleNotFound 示例日志日志条目示例

```
{
    "timestamp": "2017-10-04 19:25:46.070",
    "logLevel": "ERROR",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleNotFound",
    "clientId": "abf27092886e49a8a5c1922749736453",
    "topicName": "$aws/rules/example_rule",
    "ruleName": "example_rule",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
    "reason": "RuleNotFound",
    "details": "Rule example_rule not found"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，`RuleNotFound` 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

details

错误的简要说明。

principalId

发出请求的委托人的 ID。

reason

字符串 “RuleNotFound”。

ruleName

找不到的规则的名称。

topicName

已发布的主题的名称。

StartingRuleExecution 日志条目

当 Amazon IoT 规则引擎开始触发规则的操作时，它会生成一个 `eventType` 为 `StartingRuleExecution` 的日志条目。

StartingRuleExecution 示例日志日志条目示例

```
{
    "timestamp": "2017-08-10 16:32:46.002",
    "logLevel": "DEBUG",
    "traceId": "30aa7ccc-1d23-0b97-aa7b-76196d83537e",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartingRuleExecution",
    "clientId": "abf27092886e49a8a5c1922749736453",
```

```
    "topicName": "rules/test",
    "ruleName": "JSONLogsRule",
    "ruleAction": "RepublishAction",
    "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，rule- 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

principalId

发出请求的委托人的 ID。

ruleAction

所触发的操作的名称。

ruleName

匹配规则的名称。

topicName

已订阅主题的名称。

任务日志条目

Amazon IoT 任务服务会生成以下事件的日志条目。当从设备收到 MQTT 或 HTTP 请求时，将生成日志条目。

主题

- DescribeJobExecution 日志条目 (p. 425)
- GetPendingJobExecution 日志条目 (p. 426)
- ReportFinalJobExecutionCount 日志条目 (p. 427)
- StartNextPendingJobExecution 日志条目 (p. 427)
- UpdateJobExecution 日志条目 (p. 428)

DescribeJobExecution 日志条目

当服务收到描述任务执行的请求时，Amazon IoT 任务服务会生成一个 eventType 为 DescribeJobExecution 的日志条目。

DescribeJobExecution 示例日志日志条目示例

```
{
    "timestamp": "2017-08-10 19:13:22.841",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "DescribeJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/get",
    "clientToken": "myToken",
    "details": "The request status is SUCCESS."
```

```
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，GetJobExecution 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

clientToken

用于确保请求幂等性的唯一、区分大小写的标识符。有关更多信息，请参阅如何确保幂等性。

details

来自 Jobs 服务的其它信息。

jobId

任务执行的任务 ID。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

topicName

发出请求所使用的主题。

GetPendingJobExecution 日志条目

当服务收到任务执行请求时，Amazon IoT 任务服务会生成一个 eventType 为 GetPendingJobExecution 的日志条目。

GetPendingJobExecution 示例日志日志条目示例

```
{
    "timestamp": "2018-06-13 17:45:17.197",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "GetPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "299966ad-54de-40b4-99d3-4fc8b52da0c5",
    "topicName": "$aws/things/299966ad-54de-40b4-99d3-4fc8b52da0c5/jobs/get",
    "clientToken": "24b9a741-15a7-44fc-bd3c-1ff2e34e5e82",
    "details": "The request status is SUCCESS."
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，GetPendingJobExecution 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

clientToken

用于确保请求幂等性的唯一、区分大小写的标识符。有关更多信息，请参阅如何确保幂等性。

details

来自 Jobs 服务的其它信息。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

topicName

已订阅主题的名称。

ReportFinalJobExecutionCount 日志条目

当任务完成时，Amazon IoT Jobs 服务会生成一个 `entryType` 为 `ReportFinalJobExecutionCount` 的日志条目。

ReportFinalJobExecutionCount 示例日志日志条目示例

```
{
    "timestamp": "2017-08-10 19:44:16.776",
    "logLevel": "INFO",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "ReportFinalJobExecutionCount",
    "jobId": "002",
    "details": "Job 002 completed. QUEUED job execution count: 0 IN_PROGRESS job execution
 count: 0 FAILED job execution count: 0 SUCCEEDED job execution count: 1 CANCELED job
 execution count: 0 REJECTED job execution count: 0 REMOVED job execution count: 0"
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，`ReportFinalJobExecutionCount` 日志条目还包含以下属性：

details

来自 Jobs 服务的其它信息。

jobId

任务执行的任务 ID。

StartNextPendingJobExecution 日志条目

当服务收到开始执行下一个待处理任务的请求时，Amazon IoT 任务服务会生成一个 `eventType` 为 `StartNextPendingJobExecution` 的日志条目。

StartNextPendingJobExecution 示例日志日志条目示例

```
{
    "timestamp": "2018-06-13 17:49:51.036",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "StartNextPendingJobExecution",
    "protocol": "MQTT",
    "clientId": "95c47808-b1ca-4794-bc68-a588d6d9216c",
    "topicName": "$aws/things/95c47808-b1ca-4794-bc68-a588d6d9216c/jobs/start-next",
    "clientToken": "bd7447c4-3a05-49f4-8517-dd89b2c68d94",
    "details": "The request status is SUCCESS."
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，`StartNextPendingJobExecution` 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

clientToken

用于确保请求幂等性的唯一、区分大小写的标识符。有关更多信息，请参阅如何确保幂等性。

details

来自 Jobs 服务的其它信息。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

topicName

发出请求所使用的主题。

UpdateJobExecution 日志条目

当服务收到更新任务执行的请求时，Amazon IoT 任务服务会生成一个 eventType 为 UpdateJobExecution 的日志条目。

UpdateJobExecution 示例日志日志条目示例

```
{
    "timestamp": "2017-08-10 19:25:14.758",
    "logLevel": "DEBUG",
    "accountId": "123456789012",
    "status": "Success",
    "eventType": "UpdateJobExecution",
    "protocol": "MQTT",
    "clientId": "thingOne",
    "jobId": "002",
    "topicName": "$aws/things/thingOne/jobs/002/update",
    "clientToken": "myClientToken",
    "versionNumber": "1",
    "details": "The destination status is IN_PROGRESS. The request status is SUCCESS."
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，UpdateJobExecution 日志条目还包含以下属性：

clientId

发出请求的客户端的 ID。

clientToken

用于确保请求幂等性的唯一、区分大小写的标识符。有关更多信息，请参阅如何确保幂等性。

details

来自 Jobs 服务的其它信息。

jobId

任务执行的任务 ID。

protocol

发出请求时使用的协议。有效值为 MQTT 或 HTTP。

topicName

发出请求所使用的主题。

versionNumber

任务执行的版本。

设备预配置日志条目

Amazon IoT 设备预配置服务会生成以下事件的日志。

主题

- GetDeviceCredentials 日志条目 (p. 429)
- ProvisionDevice 日志条目 (p. 429)

GetDeviceCredentials 日志条目

当客户端调用 `GetDeviceCredential` 时，Amazon IoT 设备预配置服务会生成一个 `eventType` 为 `GetDeviceCredential` 的日志条目。

GetDeviceCredentials示例日志日志条目示例

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "GetDeviceCredentials",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，`GetDeviceCredentials` 日志条目还包含以下属性：

details

错误的简要说明。

deviceCertificateId

设备证书的 ID。

ProvisionDevice 日志条目

当客户端调用 `ProvisionDevice` 时，Amazon IoT 设备预配置服务会生成一个 `eventType` 为 `ProvisionDevice` 的日志条目。

ProvisionDevice 示例日志日志条目示例

```
{
  "timestamp" : "2019-02-20 20:31:22.932",
  "logLevel" : "INFO",
  "traceId" : "8d9c016f-6cc7-441e-8909-7ee3d5563405",
  "accountId" : "123456789101",
  "status" : "Success",
  "eventType" : "ProvisionDevice",
  "provisioningTemplateName" : "myTemplate",
  "deviceCertificateId" :
 "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "details" : "Additional details about this log."
 }
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，ProvisionDevice 日志条目还包含以下属性：

details

错误的简要说明。

deviceCertificateId

设备证书的 ID。

provisioningTemplateName

预配置模板的名称。

动态事物组日志条目

Amazon IoT 动态事物组为以下事件生成日志。

主题

- AddThingToDynamicThingGroupsFailed 日志条目 (p. 430)

AddThingToDynamicThingGroupsFailed 日志条目

当 Amazon IoT 无法将事物添加到指定的动态组时，它会生成一个 eventType 为 AddThingToDynamicThingGroupsFailed 的日志条目。当事物满足要处于动态事物组中的条件时，就会发生此情况；但是，无法将其添加到动态组中，或者无法将其从动态组中删除。这可能是因为：

- 事物已经属于最大数量的组数。
- --override-dynamic-groups 选项用于将事物添加到静态事物组中。它已被从一个动态事物组中删除，以便可以再添加。

有关更多信息，请参阅动态事物组限制和冲突 (p. 257)。

AddThingToDynamicThingGroupsFailed 示例日志日志条目示例

此示例显示 AddThingToDynamicThingGroupsFailed 错误的日志条目。在本示例中，TestThing符合加入中列出的动态事物组的条件dynamicThingGroupNames，但无法添加到这些动态组中，如中所述reason。

```
{
 "timestamp": "2020-03-16 22:24:43.804",
 "logLevel": "ERROR",
 "traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
 "accountId": "57EXAMPLE833",
 "status": "Failure",
 "eventType": "AddThingToDynamicThingGroupsFailed",
 "thingName": "TestThing",
 "dynamicThingGroupNames": [
  "DynamicThingGroup11",
  "DynamicThingGroup12",
  "DynamicThingGroup13",
  "DynamicThingGroup14"
 ],
 "reason": "The thing failed to be added to the given dynamic thing group(s) because the
 thing already belongs to the maximum allowed number of groups."
}
```

除了 常 CloudWatch 日志属性 (p. 431) 之外，AddThingToDynamicThingGroupsFailed 日志条目还包含以下属性：

dynamicThingGroup名字

无法向其中添加该事物的动态事物组的数组。

reason

事物无法添加到动态事物组的原因。

thingName

无法添加到动态事物组的事物的名称。

机群索引日志条目

Amazon IoT 机群索引会生成以下事件的日志条目。

主题

- NamedShadowCountForDynamicGroupQueryLimitExceeded 日志条目 (p. 431)

NamedShadowCountForDynamicGroupQueryLimitExceeded 日志条目

对于动态组中不特定于数据来源的查询术语，每个对象最多处理 25 个命名影子。由于事件而违反该限制时，将发出 NamedShadowCountForDynamicGroupQueryLimitExceeded 事件类型。

NamedShadowCountForDynamicGroupQueryLimitExceeded 日志条目示例

此示例显示 NamedShadowCountForDynamicGroupQueryLimitExceeded 错误的日志条目。在此示例中，所有基于值的 DynamicGroup 结果都可能不准确，如 reason 字段中所述。

```
{
"timestamp": "2020-03-16 22:24:43.804",
"logLevel": "ERROR",
"traceId": "70b1f2f5-d95e-f897-9dcc-31e68c3e1a30",
"accountId": "571032923833",
"status": "Failure",
"eventType": "NamedShadowCountForDynamicGroupQueryLimitExceeded",
"thingName": "TestThing",
"reason": "A maximum of 25 named shadows per thing are processed for non-data source
 specific query terms in dynamic groups."
}
```

常 CloudWatch 日志属性

所有 Lo CloudWatch gs 日志条目都包含这些属性：

accountId

您的 Amazon Web Services 账户 ID。

eventType

已为其生成日志的事件类型。事件类型的值取决于生成日志条目的事件。每个日志条目描述都包括该日志条目的 eventType 的值。

logLevel

所使用的日志级别。有关更多信息，请参阅the section called “日志级别” (p. 400)：

状态

请求的状态。

timestamp

客户端连接到 Amazon IoT 消息代理的时间的 UNIX 时间戳。

traceId

一个随机生成的标识符，可用于将特定请求的所有日志关联起来。

将设备端日志上载到 Amazon CloudWatch

您可以将设备端的历史日志上载到 Amazon， CloudWatch 以监控和分析设备在现场的活动。设备端日志可以包括系统、应用程序和设备日志文件。此过程使用 CloudWatch Logs 规则操作参数将设备端日志发布到客户定义的日志组。

工作原理

当 Amazon IoT 设备向 Amazon IoT 主题发送包含格式化日志文件的 MQTT 消息时，该过程即开始。Amazon IoT规则监控消息主题并将日志文件发送到您定义的 Lo CloudWatch gs 组。然后，您可以查看和分析信息。

主题

- MQTT 主题 (p. 432)
- 规则操作 (p. 432)

MQTT 主题

选择要用于发布日志的 MQTT 主题命名空间。我们建议将此格式用于公共主题空间 `$aws/rules/things/thing_name/logs`，并将此格式用于错误主题 `$aws/rules/things/thing_name/logs/errors`。建议使用日志和错误主题的命名结构，但不是必需要求。有关更多信息，请参阅为 Amazon IoT Core 设计 MQTT 主题。

通过使用建议的公共主题空间，您可以利用 Amazon IoT 基本摄取保留主题。Amazon IoT基本摄取功能将设备数据安全地发送到 Amazon IoT 规则操作支持的 Amazon 服务。它从摄取路径中删除发布/订阅消息代理，因此更具成本效益。有关更多信息，请参阅借助基本摄取功能降低消息收发成本。

如果您使用 batchMode 上载日志文件，则您的消息必须遵循包括 UNIX 时间戳和消息的特定格式。有关更多信息，请参阅CloudWatch 日志规则操作中的 batchMode 的 MQTT 消息格式要求。

规则操作

当Amazon IoT收到来自客户端设备的 MQTT 消息时，Amazon IoT规则会监控客户定义的主题，并将内容发布到您定义的 CloudWatch 日志组。此过程使用 L CloudWatch ogs 规则操作为批量日志文件监控 MQTT。有关更多信息，请参阅CloudWatch 日志Amazon IoT规则操作。

批量模式

`batchMode`是 LoAmazon IoT CloudWatch gs 规则操作中的一个布尔参数。此参数是可选的，原定设置情况下处于关闭状态（`false`）。要批量上载设备端日志文件，创建 Amazon IoT 规则时必须启用此参数（`true`）。有关更多信息，请参阅Amazon IoT规则操作部分中的CloudWatch 日志。

使用 Amazon IoT 规则上载设备端日志

您可以使用Amazon IoT规则引擎将现有设备端日志文件（系统、应用程序和设备客户端日志）中的日志记录上载到 Amazon CloudWatch。当设备端日志发布到 MQTT 主题时，L CloudWatch ogs 规则操作会将消息传

输到 L CloudWatch ogs。此过程概述了如何使用开启（设置为 `true`）的规则操作 `batchMode` 参数批量上载设备日志。

要开始将设备端日志上载到 CloudWatch，请先完成以下先决条件。

先决条件

开始之前，请执行以下操作：

- 创建至少一台向 Amazon IoT Core 注册为 Amazon IoT 事物的目标 IoT 设备。有关更多信息，请参阅创建事物对象。
- 确定用于摄取和错误的 MQTT 主题空间。有关 MQTT 主题和建议的命名规则的更多信息，请参阅将设备端日志上载到 Amazon 中的MQTT 主题 (p. 432) MQTT 主题部分 CloudWatch。

有关这些先决条件的更多信息，请参阅将设备端日志上载到 CloudWatch。

创建 CloudWatch 日志组

要创建 CloudWatch 日志组，请完成以下步骤。根据您更喜欢通过Amazon Web Services Management Console还是 Amazon Command Line Interface（Amazon CLI）执行以下步骤，选择相应的选项卡。

Amazon Web Services Management Console

要创建 CloudWatch 日志组，请使用Amazon Web Services Management Console

1. 打开Amazon Web Services Management Console并导航到 CloudWatch。
2. 在导航栏上，选择 Logs（日志），然后选择 Log groups（日志组）。
3. 选择 Create log group (创建日志组)。
4. 更新 Log group name（日志组名称），并可选择更新 Retention setting（保留设置）字段。
5. 选择创建。

Amazon CLI

要创建 CloudWatch 日志组，请使用Amazon CLI

1. 要创建日志组，请运行以下命令。有关更多信息，请参阅《Amazon CLI v2 命令参考》中的 `create-log-group`。

   将示例中的日志组名称（`uploadLogsGroup`）替换为您首选的名称。

   ```
   aws logs create-log-group --log-group-name uploadLogsGroup
   ```

2. 要确认已正确创建日志组，请运行以下命令。

   ```
   aws logs describe-log-groups --log-group-name-prefix uploadLogsGroup
   ```

   示例输出：

   ```
   {
       "logGroups": [
           {
               "logGroupName": "uploadLogsGroup",
               "creationTime": 1674521804657,
               "metricFilterCount": 0,
   ```

```
            "arn": "arn:aws:logs:us-east-1:111122223333:log-
group:uploadLogsGroup:*",
            "storedBytes": 0
        }
    ]
}
```

创建主题规则

要创建 Amazon IoT 规则，请完成以下步骤。根据您更喜欢通过Amazon Web Services Management Console还是 Amazon Command Line Interface（Amazon CLI）执行以下步骤，选择相应的选项卡。

Amazon Web Services Management Console

使用 Amazon Web Services Management Console创建主题规则

1. 打开规则中心。
   a. 打开Amazon Web Services Management Console并导航到 Amazon IoT。
   b. 在导航栏上，选择 Message routing（消息路由），然后选择 Rules（规则）。
   c. 选择 Create rule（创建规则）。
2. 输入规则属性。
   a. 输入由字母数字组成的 Rule name（规则名称）。
   b. （可选）输入 Rule description（规则描述）和 Tags（标签）。
   c. 选择下一步。
3. 输入 SQL 语句。
   a. 使用您为摄取定义的 MQTT 主题输入 SQL 语句。

      例如，`SELECT * FROM '$aws/rules/things/thing_name/logs'`
   b. 选择下一步。
4. 输入规则操作。
   a. 在 “操作 1” 菜单上，选择CloudWatch日志。
   b. 选择 Log group name（日志组名称），然后选择您创建的日志组。
   c. 选择 Use batch mode（使用批量模式）。
   d. 为规则指定 IAM 角色。

      如果规则具有 IAM 角色，请执行以下操作。

      1. 在 IAM role（IAM 角色）菜单上，选择您的 IAM 角色。

      如果规则没有 IAM 角色，请执行以下操作。

      1. 选择 Create new role（创建新角色）。
      2. 对于 Role name（角色名称），输入唯一名称并选择 Create（创建）。
      3. 确认 IAM role（IAM 角色）字段中的 IAM 角色名称是否正确。
   e. 选择下一步。
5. 查看模板配置。
   a. 查看任务模板的设置以验证它们正确无误。
   b. 完成后，选择 Create (创建)。

Amazon CLI

使用 Amazon CLI 创建 IAM 角色和主题规则

1. 创建一个 IAM 角色，该角色授予对 Amazon IoT 规则的权限。

   a. 创建一个 IAM policy。

      要创建 IAM policy，请运行以下命令。确保更新 `policy-name` 参数值。有关更多信息，请参阅《Amazon CLI v2 命令参考》中的 `create-policy`。

      Note

      如果您使用的是 Microsoft Windows 操作系统，则可能需要将行尾标记 (\ ) 替换为勾号 (`) 或其他字符。

```
aws iam create-policy \
    --policy-name uploadLogsPolicy \
    --policy-document \
'{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iot:CreateTopicRule",
            "iot:Publish",
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:GetLogEvents"
        ],
        "Resource": "*"
    }
}'
```

   b. 将输出中的策略 ARN 复制到文本编辑器中。

      示例输出：

```
{
    "Policy": {
        "PolicyName": "uploadLogsPolicy",
        "PermissionsBoundaryUsageCount": 0,
        "CreateDate": "2023-01-23T18:30:10Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "AAABBBCCCDDDEEEFFFGGG",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::111122223333:policy/uploadLogsPolicy",
        "UpdateDate": "2023-01-23T18:30:10Z"
    }
}
```

   c. 创建 IAM 角色和信任策略。

      要创建 IAM policy，请运行以下命令。确保更新 `role-name` 参数值。有关更多信息，请参阅《Amazon CLI v2 命令参考》中的 `create-role`。

```
aws iam create-role \
--role-name uploadLogsRole \
--assume-role-policy-document \
'{
```

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

d. 向角色附加 IAM policy。

要创建 IAM policy，请运行以下命令。确保更新 `role-name` 和 `policy-arn` 参数值。有关更多信息，请参阅《Amazon CLI v2 命令参考》中的 attach-role-policy。

```
aws iam attach-role-policy \
--role-name uploadLogsRole \
--policy-arn arn:aws:iam::111122223333:policy/uploadLogsPolicy
```

e. 查看角色。

要确认已正确创建 IAM 角色，请运行以下命令。确保更新 `role-name` 参数值。有关更多信息，请参阅《Amazon CLI v2 命令参考》中的 get-role。

```
aws iam get-role --role-name uploadLogsRole
```

示例输出：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "uploadLogsRole",
        "RoleId": "AAABBBCCCDDDEEEFFFGGG",
        "Arn": "arn:aws:iam::111122223333:role/uploadLogsRole",
        "CreateDate": "2023-01-23T19:17:15+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "iot.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Description": "",
        "MaxSessionDuration": 3600,
        "RoleLastUsed": {}
    }
}
```

2. 在 Amazon CLI 中创建 Amazon IoT 主题规则。

a. 要创建 Amazon IoT 主题规则，请运行以下命令。确保更新 `--rule-name`、`sql` 语句、`description`、`roleARN` 和 `logGroupName` 参数值。有关更多信息，请参阅《Amazon CLI v2 命令参考》中的 create-topic-rule。

```
aws iot create-topic-rule \
--rule-name uploadLogsRule \
--topic-rule-payload \
 '{
  "sql":"SELECT * FROM 'rules/things/thing_name/logs'",
  "description":"Upload logs test rule",
  "ruleDisabled":false,
  "awsIotSqlVersion":"2016-03-23",
  "actions":[
   {"cloudwatchLogs":
    {"roleArn":"arn:aws:iam::111122223333:role/uploadLogsRole",
    "logGroupName":"uploadLogsGroup",
    "batchMode":true}
    }
   ]
}'
```

b. 要确认已正确创建规则，请运行以下命令。确保更新 `role-name` 参数值。有关更多信息，请参阅《Amazon CLI v2 命令参考》中的 get-topic-rule。

```
aws iot get-topic-rule --rule-name uploadLogsRule
```

示例输出：

```
{
    "ruleArn": "arn:aws:iot:us-east-1:111122223333:rule/uploadLogsRule",
    "rule": {
        "ruleName": "uploadLogsRule",
        "sql": "SELECT * FROM rules/things/thing_name/logs",
        "description": "Upload logs test rule",
        "createdAt": "2023-01-24T16:28:15+00:00",
        "actions": [
            {
                "cloudwatchLogs": {
                    "roleArn": "arn:aws:iam::111122223333:role/uploadLogsRole",
                    "logGroupName": "uploadLogsGroup",
                    "batchMode": true
                }
            }
        ],
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23"
    }
}
```

将设备端日志发送到 Amazon IoT

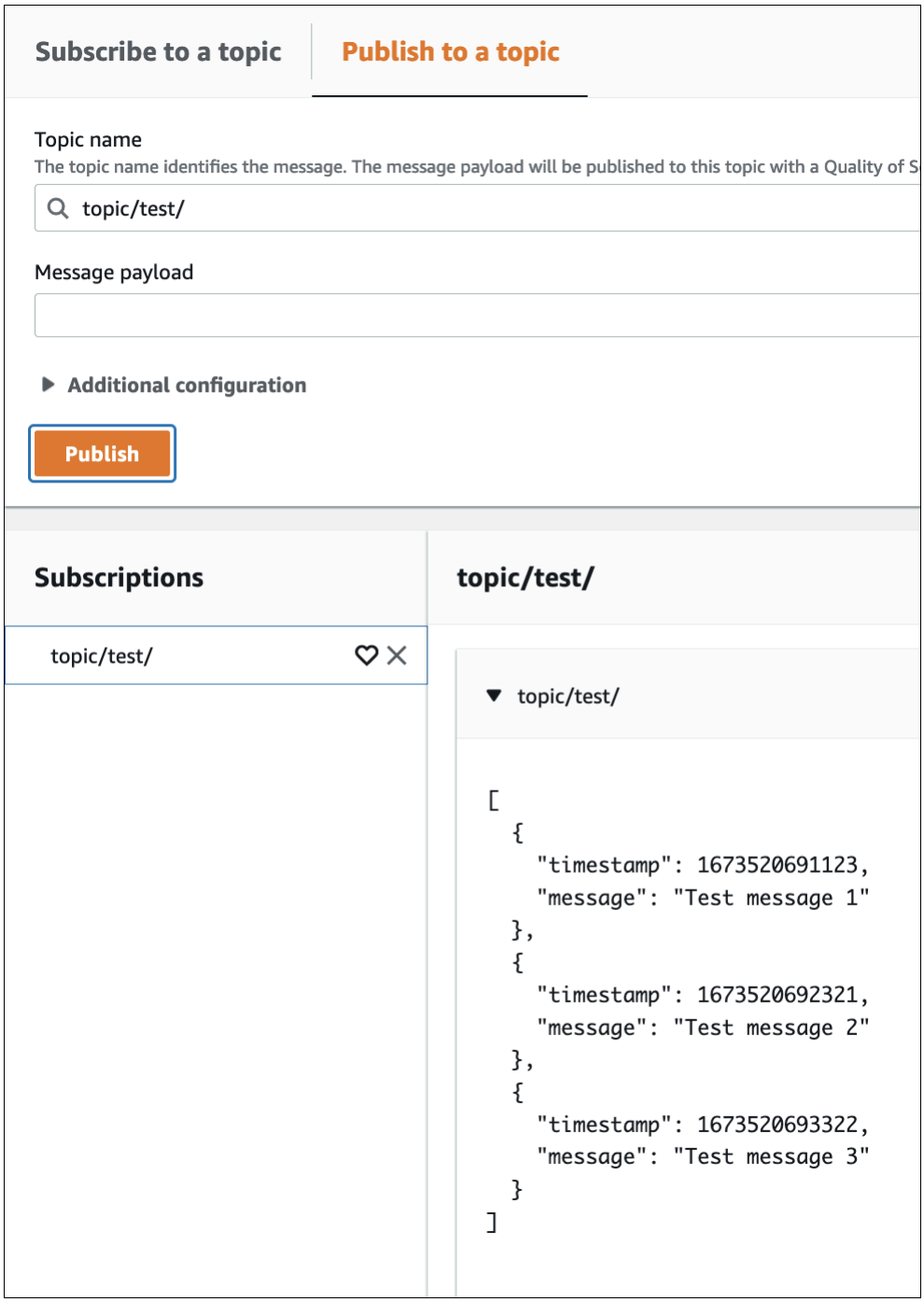
将设备端日志发送到 Amazon IoT

1. 要将历史日志发送到 Amazon IoT，请与您的设备通信，以确保满足以下条件。
   - 日志信息将发送到在本过程的先决条件部分中指定的正确主题命名空间。

     例如，`$aws/rules/things/thing_name/logs`
   - MQTT 消息有效负载的格式正确无误。有关 MQTT 主题和建议的命名规则的更多信息，请参阅将设备端日志上载到 Amazon CloudWatch (p. 432)中的 MQTT 主题 (p. 432) 部分。
2. 确认已在 Amazon IoT MQTT 客户端中接收 MQTT 消息。

a. 打开Amazon Web Services Management Console并导航到 Amazon IoT。
b. 要查看 MQTT test client（MQTT 测试客户端），请在导航栏上选择 Test（测试）、MQTT test client（MQTT 测试客户端）。
c. 对于 Subscribe to a topic（订阅主题）、Topic filter（主题筛选器），输入 topic namespace（主题命名空间）。
d. 选择 Subscribe。

   MQTT 消息显示在 Subscriptions（订阅）和 Topic（主题）表中，如下所示。这些消息可能需要长达五分钟才会显示。



查看日志数据

在 Logs 中查看您的 CloudWatch 日志记录

1. 打开Amazon Web Services Management Console，然后导航到CloudWatch。
2. 在导航栏上，依次选择 Logs（日志）和 Logs Insights（日志见解）。
3. 在 Select log group(s)（选择日志组）菜单上，选择您在 Amazon IoT 规则中指定的日志组。

4. 在 Logs insights（日志见解）页面上，选择 Run query（运行查询）。

使用 Amazon CloudTrail 记录 Amazon IoT API 调用

Amazon IoT与服务集成Amazon CloudTrail，后者提供了中的用户、角色或Amazon服务所采取操作的记录Amazon IoT。 CloudTrail 将对的所有 API 调用均Amazon IoT作为事件捕获，包括来自Amazon IoT控制台的调用和对Amazon IoT API 的代码调用。如果您创建跟踪，则可以使 CloudTrail事件持续传送到 Amazon S3 存储桶（包括的事件）Amazon IoT。如果您不配置跟踪记录，则仍可在 CloudTrail Convent history（事件历史记录）中查看最新事件。通过使用收集的信息 CloudTrail，您可以确定向发出了什么请求Amazon IoT、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其它详细信息。

要了解更多信息 CloudTrail，请参阅Amazon CloudTrail用户指南。

Amazon IoT中的信息 CloudTrail

CloudTrail 在您创建账户Amazon Web Services 账户时，将在该账户上启用。当活动发生时Amazon IoT，该活动将记录在事件历史记录中，并与其他Amazon服务 CloudTrail 事件一同保存在事件历史记录中。您可以在 Amazon Web Services 账户 中查看、搜索和下载最新事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

对于Amazon Web Services 账户中的事件的持续记录（包括 Amazon IoT 的事件），请创建跟踪记录。通过跟踪 CloudTrail ，可将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有Amazon Web Services 区域。此跟踪记录在Amazon分区中记录所有Amazon Web Services 区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。您可以配置其他Amazon服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取操作。有关更多信息，请参阅：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为配置 Amazon SNS 通知 CloudTrail
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail日志文件

> **Note**
>
> Amazon IoT不会记录数据层面的操作（设备端） CloudTrail。 CloudWatch 用于监控这些操作。

一般而言，由记录进行更改的Amazon IoT控制面板操作
CloudTrail。CreateThing、CreateKeysAndCertificate和之类的调用会UpdateCertificate留下 CloudTrail 条目，而诸如ListThings和之类的调用则ListTopicRules不留条目。

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

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

有关更多信息，请参阅 CloudTrail userIdentity 元素。

Amazon IoT 操作记录在 Amazon IoT API 参考中。Amazon IoTWireless 操作会记录在 Amazon IoT Wireless API 参考中。

了解 Amazon IoT 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时

间、请求参数等方面的信息。 CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了AttachPolicy操作。

```
{
    "timestamp":"1460159496",
    "AdditionalEventData":"",
    "Annotation":"",
    "ApiVersion":"",
    "ErrorCode":"",
    "ErrorMessage":"",
    "EventID":"8bff4fed-c229-4d2d-8264-4ab28a487505",
    "EventName":"AttachPolicy",
    "EventTime":"2016-04-08T23:51:36Z",
    "EventType":"AwsApiCall",
    "ReadOnly":"",
    "RecipientAccountList":"",
    "RequestID":"d4875df2-fde4-11e5-b829-23bf9b56cbcd",
    "RequestParamters":{
        "principal":"arn:aws:iot:us-
east-1:123456789012:cert/528ce36e8047f6a75ee51ab7beddb4eb268ad41d2ea881a10b67e8e76924d894",
        "policyName":"ExamplePolicyForIoT"
    },
    "Resources":"",
    "ResponseElements":"",
    "SourceIpAddress":"52.90.213.26",
    "UserAgent":"aws-internal/3",
    "UserIdentity":{
        "type":"AssumedRole",
        "principalId":"AKIAI44QH8DHBEXAMPLE",
        "arn":"arn:aws:sts::12345678912:assumed-role/iotmonitor-us-east-1-beta-
InstanceRole-1C5T1YCYMHPYT/i-35d0a4b6",
        "accountId":"222222222222",
        "accessKeyId":"access-key-id",
        "sessionContext":{
            "attributes":{
                "mfaAuthenticated":"false",
                "creationDate":"Fri Apr 08 23:51:10 UTC 2016"
            },
            "sessionIssuer":{
                "type":"Role",
                "principalId":"AKIAI44QH8DHBEXAMPLE",
                "arn":"arn:aws:iam::123456789012:role/executionServiceEC2Role/iotmonitor-
us-east-1-beta-InstanceRole-1C5T1YCYMHPYT",
                "accountId":"222222222222",
                "userName":"iotmonitor-us-east-1-InstanceRole-1C5T1YCYMHPYT"
            }
        },
        "invokedBy":{
            "serviceAccountId":"111111111111"
        }
    },
    "VpcEndpointId":""
}
```

Amazon IoT 规则

规则使您的设备能够与 Amazon 服务交互。基于 MQTT 主题流分析规则并执行操作。您可以使用规则来支持如下任务：

- 补充或筛选从设备接收的数据。
- 将从设备接收的数据写入 Amazon DynamoDB 数据库。
- 将文件保存到 Amazon S3。
- 向使用 Amazon SNS 的所有用户发送推送通知。
- 将数据发布到 Amazon SQS 队列。
- 调用 Lambda 函数来提取数据。
- 使用 Amazon Kinesis 处理来自大量设备的消息。
- 将数据发送到亚马逊 OpenSearch 服务。
- 捕获 CloudWatch 指标。
- 更改告 CloudWatch 警。
- 将 MQTT 消息中的数据发送 SageMaker 到 Amazon，以便根据机器学习 (ML) 模型进行预测。
- 向 Salesforce IoT 输入流发送消息。
- 将消息数据发送到 Amazon IoT Analytics 通道。
- 开始执行 Step Functions 状态机。
- 将消息数据发送到 Amazon IoT Events 输入。
- 将消息数据发送到 Amazon IoT SiteWise 中的资产属性。
- 将消息数据发送到 Web 应用程序或服务。

您的规则可以使用通过the section called “设备通信协议” (p. 75)支持的发布/订阅协议传递的 MQTT 消息。您还可以使用基本摄取 (p. 529)功能将设备数据安全地发送到前面列出的 Amazon 服务，而不会产生消息收发费用。基本摄取 (p. 529)功能通过从摄取路径中删除发布/订阅消息代理来优化数据流。这使其具有成本效益，同时仍保留 Amazon IoT 的安全和数据处理功能。

您必须先授予 Amazon IoT 代表您访问 Amazon 资源的权限，然后其才能够执行这些操作。执行这些操作时，即产生了您所用 Amazon 服务的标准费用。

目录

- 向 Amazon IoT 规则授予所需的访问权限 (p. 443)
- 传递角色权限 (p. 444)
- 创建 Amazon IoT 规则 (p. 445)
- 查看您的规则 (p. 448)
- 删除规则 (p. 448)
- Amazon IoT 规则操作 (p. 449)
- 排查规则问题 (p. 522)
- 使用 Amazon IoT 规则访问跨账户资源 (p. 522)
- 错误处理（错误操作）(p. 527)
- 借助基本摄取功能，降低消息收发成本 (p. 529)
- Amazon IoT SQL 参考 (p. 530)

向 Amazon IoT 规则授予所需的访问权限

您可以使用 IAM 角色来控制每个规则可以访问的Amazon资源。在创建规则之前，您必须创建一个 IAM 角色，并对其应用允许访问所需Amazon资源的策略。执行规则时，Amazon IoT 将代入该角色。

创建用于向 Amazon IoT 规则授予所需访问权限的 IAM 角色和 Amazon IoT 策略 (Amazon CLI)

1. 将以下信任策略文档（用于授予 Amazon IoT 代入此角色的权限）保存到名为 `iot-role-trust.json` 的文件。

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

使用 create-role 命令，指定 `iot-role-trust.json` 文件，创建 IAM 角色：

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document file://iot-role-trust.json
```

此命令的输出如下所示：

```
{
  "Role": {
    "AssumeRolePolicyDocument": "url-encoded-json",
    "RoleId": "AKIAIOSFODNN7EXAMPLE",
    "CreateDate": "2015-09-30T18:43:32.821Z",
    "RoleName": "my-iot-role",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:role/my-iot-role"
  }
}
```

2. 将以下 JSON 保存到名为 `my-iot-policy.json` 的文件中。

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

该 JSON 是授予 Amazon IoT 管理员对 DynamoDB 的访问权限的示例策略文档。

使用 create-policy 命令授予 Amazon IoT 在代入该角色后访问您的 Amazon 资源的权限，并传入 `my-iot-policy.json` 文件：

```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy.json
```

有关如何在 Amazon IoT 的策略中授予对 Amazon 服务的访问权限的更多信息，请参阅 创建 Amazon IoT 规则 (p. 445)。

create-policy 命令的输出中包含该策略的 ARN。您需要将该策略附加到角色。

```
{
  "Policy": {
    "PolicyName": "my-iot-policy",
    "CreateDate": "2015-09-30T19:31:18.620Z",
    "AttachmentCount": 0,
    "IsAttachable": true,
    "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
    "DefaultVersionId": "v1",
    "Path": "/",
    "Arn": "arn:aws:iam::123456789012:policy/my-iot-policy",
    "UpdateDate": "2015-09-30T19:31:18.620Z"
  }
}
```

3. 使用attach-role-policy命令将策略附加到您的角色：

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn
 "arn:aws:iam::123456789012:policy/my-iot-policy"
```

传递角色权限

规则定义的一部分是 IAM 角色，该角色授予对规则操作中指定的资源的访问权限。一旦触发规则操作，规则引擎会代入该角色。必须在与规则相同的Amazon Web Services 账户账户中定义该角色。

在创建或替换规则时，您实际上将角色提交到规则引擎中。执行这项操作的用户需要具有 iam:PassRole 权限。为确保拥有该权限，您需要创建一个策略以便授予 iam:PassRole 权限，并将其附加到您的 IAM 用户。以下策略介绍了如何向角色提供 iam:PassRole 权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::123456789012:role/myRole"
      ]
    }
  ]
}
```

在此策略示例中，将 iam:PassRole 权限授予了角色 myRole。该角色使用角色的 ARN 指定。您必须将此策略附加到您的 IAM 用户或用户所属的角色。有关更多信息，请参阅使用管理的策略。

Note

Lambda 函数使用基于资源的策略，该策略直接附加至 Lambda 函数本身。在您创建调用 Lambda 函数的规则时，您未传递角色，因此创建规则的用户无需 iam:PassRole 权限。有关 Lambda 函数授权的更多信息，请参阅使用资源策略授予权限。

创建 Amazon IoT 规则

您可以配置规则以从连接的设备路由数据。规则包括以下部分：

Rule name（规则名称）

规则的名称。

Note

我们建议不要在您的规则名称中使用个人身份信息。

可选描述

规则的文字描述。

Note

我们建议不要在您的规则描述中使用个人身份信息。

SQL 语句

一种简化的 SQL 语法，用于筛选接收的 MQTT 主题相关消息并向其它推送数据。有关更多信息，请参阅Amazon IoT SQL 参考 (p. 530)：

SQL 版本

评估规则时使用的 SQL 规则引擎的版本。尽管该属性是可选的，但我们强烈建议您指定 SQL 版本。预设情况下，Amazon IoT Core 控制台将此属性设置为 `2016-03-23`。如果未设置该属性，例如在 Amazon CLI 命令或 Amazon CloudFormation 模板，使用 `2015-10-08`。有关更多信息，请参阅SQL 版本 (p. 591)：

一个或多个操作

执行规则时 Amazon IoT 执行的操作。例如，您可以将数据插入 DynamoDB 表、将数据写入 Amazon S3 存储桶、发布至 Amazon SNS 主题或调用 Lambda 函数。

错误操作

Amazon IoT 在无法执行规则的操作时执行的操作。

当您创建规则时，请注意在主题上发布的数据量。如果您创建的规则包含通配符主题模式，它们可能与您的大部分消息匹配，并且您可能需要增加目标操作使用的 Amazon 资源容量。另外，如果您创建包含通配符主题模式的重新发布规则，最终可能获得一个造成无限循环的循环规则。

Note

创建和更新规则是管理员级操作。有权创建或更新规则的所有用户都能够访问规则处理的数据。

创建规则 (Amazon CLI)

使用create-topic-rule命令创建规则：

```
aws iot create-topic-rule --rule-name myrule --topic-rule-payload file://myrule.json
```

下面是一个负载文件示例，其中包含将发送至 `iot/test` 主题的所有消息插入指定 DynamoDB 表的规则。SQL 语句筛选信息，角色 ARN 授予 Amazon IoT 写入 DynamoDB 表的权限。

```
{
```

```
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "dynamoDB": {
          "tableName": "my-dynamodb-table",
          "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
          "hashKeyField": "topic",
          "hashKeyValue": "${topic(2)}",
          "rangeKeyField": "timestamp",
          "rangeKeyValue": "${timestamp()}"
      }
  }]
}
```

下面是一个负载文件示例，其中包含将发送至 `iot/test` 主题的所有消息插入指定 S3 存储桶的规则。SQL 语句筛选消息，角色 ARN 授予 Amazon IoT 写入 Amazon S3 存储桶的权限。

```
{
  "awsIotSqlVersion": "2016-03-23",
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "actions": [
    {
      "s3": {
        "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
        "bucketName": "my-bucket",
        "key": "myS3Key"
      }
    }
  ]
}
```

下面的示例有效负载文件包含将数据推送至 Amazon Serv OpenSearch ice 的规则：

```
{
  "sql":"SELECT *, timestamp() as timestamp FROM 'iot/test'",
  "ruleDisabled":false,
  "awsIotSqlVersion": "2016-03-23",
  "actions":[
    {
      "elasticsearch":{
        "roleArn":"arn:aws:iam::123456789012:role/aws_iot_es",
        "endpoint":"https://my-endpoint",
        "index":"my-index",
        "type":"my-type",
        "id":"${newuuid()}"
      }
    }
  ]
}
```

下面的负载文件示例包含调用 Lambda 函数的规则：

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "lambda": {
          "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-
function"
```

```
      }
  }]
}
```

下面的负载文件示例包含发布至 Amazon SNS 主题的规则：

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "sns": {
          "targetArn": "arn:aws:sns:us-west-2:123456789012:my-sns-topic",
          "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
      }
  }]
}
```

下面的负载文件示例包含在不同 MQTT 主题上重新发布的规则：

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "republish": {
          "topic": "my-mqtt-topic",
          "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
      }
  }]
}
```

下面的负载文件示例包含将数据推送至 Amazon Kinesis Data Firehose stream 流的规则：

```
{
  "sql": "SELECT * FROM 'my-topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "firehose": {
          "roleArn": ""arn:aws:iam::123456789012:role/my-iot-role",
          "deliveryStreamName": "my-stream-name"
      }
  }]
}
```

在下面的有效负载文件示例中，具有采用 Amazon SageMaker`machinelearning_predict` 函数重新发布至某个主题（如果 MQTT 负载中的数据分类为 1）的规则。

```
{
  "sql": "SELECT * FROM 'iot/test' where machinelearning_predict('my-model',
 'arn:aws:iam::123456789012:role/my-iot-aml-role', *).predictedLabel=1",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
    "republish": {
        "roleArn": "arn:aws:iam::123456789012:role/my-iot-role",
        "topic": "my-mqtt-topic"
    }
  }]
```

```
}
```

下面是一个示例负载文件，该文件包含将消息发布到 Salesforce IoT Cloud 输入流的规则。

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "salesforce": {
          "token": "ABCDEFGHI123456789abcdefghi123456789",
          "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-id/
connection-id/my-event"
      }
  }]
}
```

下面是一个负载文件示例，其中包含一个可以开始执行 Step Functions 状态机的规则。

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [{
      "stepFunctions": {
          "stateMachineName": "myCoolStateMachine",
          "executionNamePrefix": "coolRunning",
          "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
      }
  }]
}
```

查看您的规则

使用list-topic-rules命令列出您的规则：

```
aws iot list-topic-rules
```

使用get-topic-rule命令来获取规则信息：

```
aws iot get-topic-rule --rule-name myrule
```

删除规则

用完规则后可以将其删除。

删除规则 (Amazon CLI)

使用delete-topic-rule命令删除规则：

```
aws iot delete-topic-rule --rule-name myrule
```

Amazon IoT 规则操作

Amazon IoT 规则操作将指定规则触发后应执行的操作。您可以定义操作以将数据发送到 Amazon DynamoDB 数据库、将数据发送到 Amazon Kinesis Data Streams、调用 Amazon Lambda 函数，依此类推。Amazon IoT 支持在以下Amazon Web Services 区域中操作服务可用的操作。

规则操作	描述	API 中的名称
Apache Kafka (p. 450)	将消息发送到 Apache Kafka 集群。	`kafka`
CloudWatch 警报 (p. 457)	更改 Amazon CloudWatch 告警的状态。	`cloudwatchAlarm`
CloudWatch 日志 (p. 458)	将消息发送到 Amazon CloudWatch Logs。	`cloudwatchLogs`
CloudWatch 指标 (p. 460)	将消息发送到 CloudWatch 指标。	`cloudwatchMetric`
DynamoDB (p. 461)	将消息发送到 DynamoDB 表。	`dynamoDB`
DynamoDBv2 (p. 463)	将消息数据发送到 DynamoDB 表中的多列。	`dynamoDBv2`
Elasticsearch (p. 465)	将消息发送到 Elasticsearch 端点。	`elasticsearch`
HTTP (p. 467)	将消息发布到 HTTPS 端点。	`http`
IoT Analytics (p. 493)	将消息发送到 Amazon IoT Analytics 通道。	`iotAnalytics`
IoT Events (p. 495)	将消息发送到 Amazon IoT Events 输入。	`iotEvents`
IoT SiteWise (p. 496)	将消息数据发送到 Amazon IoT SiteWise 资产属性。	`iotSiteWise`
Kinesis Data Firehose (p. 500)	将消息发送到 Kinesis Data Firehose 传输流。	`firehose`
Kinesis Data Streams (p. 502)	将消息发送到 Kinesis 数据流。	`kinesis`
Lambda (p. 503)	使用消息数据作为输入调用 Lambda 函数。	`lambda`
位置 (p. 505)	向 Amazon Location Service 发送位置数据。	`location`
OpenSearch (p. 508)	将消息发送到 Amazon Serv OpenSearch ice 终端节点。	`OpenSearch`
Republish (p. 509)	在另一个 MQTT 主题上重新发布消息。	`republish`
S3 (p. 511)	将消息存储在 Amazon Simple Storage Service (Amazon S3) 存储桶中。	`s3`

规则操作	描述	API 中的名称
Salesforce IoT (p. 513)	将消息发送到 Salesforce IoT 输入流。	`salesforce`
SNS (p. 514)	将消息发布为 Amazon Simple Notification Service (Amazon SNS) 推送通知。	`sns`
SQS (p. 515)	将消息发送到 Amazon Simple Queue Service (Amazon SQS) 队列	`sqs`
Step Functions (p. 517)	启动 Amazon Step Functions 状态机。	`stepFunctions`
the section called “Timestream” (p. 518)	将消息发送到 Amazon Timestream 数据库表。	`timestream`

注意

- 您必须在与另一个服务资源相同的 Amazon Web Services 区域 中定义规则，以便规则操作可以与该资源进行交互。
- Amazon IoT 规则引擎可多次尝试执行一个操作，以防出现间歇性错误。如果所有尝试都失败，将丢弃消息，并且 CloudWatch 日志中将记录此错误。您可以为在发生故障后调用的每条规则指定一个错误操作。有关更多信息，请参阅错误处理（错误操作）(p. 527)：
- 某些规则操作会触发与 Amazon Key Management Service (Amazon KMS) 集成的服务中的操作，以支持静态数据加密。如果您使用客户托管的 Amazon KMS key （KMS 密钥）对数据进行静态加密，服务必须具有代表调用方使用 KMS 的权限。请参阅相应服务指南中的数据加密主题，以了解如何管理客户托管 KMS 密钥的权限。有关客户托管式 KMS 密钥的更多信息，请参阅 Amazon Key Management Service 开发人员指南中的 Amazon Key Management Service 概念。

Apache Kafka

Apache Kafka (Kafka) 操作将消息直接发送到您的 Amazon Managed Streaming for Apache Kafka (Amazon MSK) 或自行托管的 Apache Kafka 集群以进行数据分析和可视化。

Note

本主题假设您熟悉 Apache Kafka 平台及相关概念。有关 Apache Kafka 的更多信息，请参阅 Apache Kafka。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担的 IAM 角色，以执行 `ec2:CreateNetworkInterface`、`ec2:DescribeNetworkInterfaces`、`ec2:CreateNetworkInterfacePer` 和 `ec2:DescribeSecurityGroups` 操作。此角色创建并管理您的 Amazon Virtual Private Cloud 弹性网络接口，以便联系您的 Kafka 代理。有关更多信息，请参阅 向 Amazon IoT 规则授予所需的访问权限 (p. 443)。

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT Core 执行此规则操作。

  有关网络接口的更多信息，请参阅 Amazon EC2 用户指南中的弹性网络接口。

  附加到您指定角色的策略应如下所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateNetworkInterface",
            "ec2:DescribeNetworkInterfaces",
            "ec2:CreateNetworkInterfacePermission",
            "ec2:DeleteNetworkInterface",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeVpcAttribute",
            "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

- 如果您使用 Amazon Secrets Manager 来存储所需的凭证以连接到 Kafka 代理，则必须创建一个 IAM 角色，以便 Amazon IoT Core 可以承担该角色来执行 `secretsmanager:GetSecretValue` 和 `secretsmanager:DescribeSecret` 操作。

  附加到您指定角色的策略应如下所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetSecretValue",
            "secretsmanager:DescribeSecret"
            ],
            "Resource": [

 "arn:aws:secretsmanager:region:123456789012:secret:kafka_client_truststore-*",
                "arn:aws:secretsmanager:region:123456789012:secret:kafka_keytab-*"
            ]
        }
    ]
}
```

- 您可以在 Amazon Virtual Private Cloud 中运行您的 Apache Kafka 集群。您必须创建 Virtual Private Cloud (VPC) 目标，然后在子网中使用 NAT 网关将来自 Amazon IoT 的消息转发到公有 Kafka 集群。Amazon IoT 规则引擎会在 VPC 目标中列出的每个子网中创建一个网络接口，以将流量直接路由到 VPC。当您创建 VPC 目标时，Amazon IoT 规则引擎会自动创建 VPC 规则操作。有关 VPC 规则操作的更多信息，请参阅 Virtual Private Cloud (VPC) 目标 (p. 455)。
- 如果您使用客户托管的 Amazon KMS key （KMS 密钥）对数据进行静态加密，服务必须具有代表调用方使用 KMS 的权限。有关更多信息，请参阅 Amazon Managed Streaming for Apache Kafka 开发人员指南中的 Amazon MSK 加密。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

destinationArn

VPC 目标的 Amazon Resource Name (ARN)。有关如何创建 VPC 目标的更多信息，请参阅 Virtual Private Cloud (VPC) 目标 (p. 455)。

topic

要发送到 Kafka 代理的消息的 Kafka 主题。

您可以使用替代模板替换此字段。有关更多信息，请参阅the section called “替换模板” (p. 585)：

键（可选）

Kafka 消息键。

您可以使用替代模板替换此字段。有关更多信息，请参阅the section called “替换模板” (p. 585)：

分区（可选）

Kafka 消息分区。

您可以使用替代模板替换此字段。有关更多信息，请参阅the section called “替换模板” (p. 585)：

clientProperties

定义 Apache Kafka 生成器客户端属性的对象。

acks（可选）

生成器在考虑请求完成之前要求服务器收到的确认数。

如果指定 0 作为值，则生成器将不会等待来自服务器的任何确认。如果服务器没有收到该消息，则生成器将不会重试发送该消息。

有效值：0、1 默认值为 1。

bootstrap.servers

主机和端口对列表(`host1:port1`、`host2:port2` 等）用于建立到您的 Kafka 集群的初始连接。

compression.type（可选）

生成器生成的所有数据的压缩类型。

有效值：`none`、`gzip`、`snappy`、`lz4`、`zstd`。默认值为 `none`。

security.protocol

用于连接到您的 Kafka 代理的安全协议。

有效值：`SSL`、`SASL_SSL`。默认值为 `SSL`。

key.serializer

指定如何将您使用 `ProducerRecord` 提供的键对象转换为字节。

有效值：`StringSerializer`。

value.serializer

指定如何将您使用 `ProducerRecord` 提供的值对象转换为字节。

有效值：`ByteBufferSerializer`。

ssl.truststore

base64 格式的信任库文件或位于 Amazon Secrets Manager 的信任库文件位置。如果您的信任库受到 Amazon 证书授权机构 (CA) 的认证，则不需要此值。

此字段支持替换模板。如果使用 Secrets Manager 存储连接到 Kafka 代理所需的凭证，则可以使用 get_secret SQL 函数检索此字段的值。有关替换模板的更多信息，请参阅 the section called “替换模板” (p. 585)。有关 get_secret SQL 函数的更多信息，请参阅 the section called

“get_secret(secretId, secretType, key, roleArn)” (p. 557)。如果信任库采用文件的形式，请使用 SecretBinary 参数。如果信任库采用字符串的形式，请使用 SecretString 参数。

此值最大为 65 KB。

ssl.truststore.password

信任库存储的密码。仅当您为信任库创建了密码时，才需要此值。

ssl.keystore

密钥库文件。当您指定 SSL 作为 security.protocol 的值时，才需要此值。

此字段支持替换模板。您必须使用 Secrets Manager 来存储连接到您的 Kafka 代理所需的凭证。使用 get_secret SQL 函数检索此字段的值。有关替换模板的更多信息，请参阅 the section called “替换模板” (p. 585)。有关 get_secret SQL 函数的更多信息，请参阅 the section called “get_secret(secretId, secretType, key, roleArn)” (p. 557)。使用 SecretBinary 参数。

ssl.keystore.password

密钥库文件存储的密码。如果为 ssl.keystore 指定了值，则需要此值。

此字段的值可以是纯文本。此字段还支持替代模板。您必须使用 Secrets Manager 来存储连接到您的 Kafka 代理所需的凭证。使用 get_secret SQL 函数检索此字段的值。有关替换模板的更多信息，请参阅 the section called “替换模板” (p. 585)。有关 get_secret SQL 函数的更多信息，请参阅 the section called “get_secret(secretId, secretType, key, roleArn)” (p. 557)。使用 SecretString 参数。

ssl.key.password

密钥库文件中私有密钥的密码。

此字段支持替换模板。您必须使用 Secrets Manager 来存储连接到您的 Kafka 代理所需的凭证。使用 get_secret SQL 函数检索此字段的值。有关替换模板的更多信息，请参阅 the section called “替换模板” (p. 585)。有关 get_secret SQL 函数的更多信息，请参阅 the section called “get_secret(secretId, secretType, key, roleArn)” (p. 557)。使用 SecretString 参数。

sasl.mechanism

用于连接到您的 Kafka 代理的安全机制。当您为 security.protocol 指定 SASL_SSL 时，则需要此值。

有效值：PLAIN、SCRAM-SHA-512、GSSAPI。

Note

SCRAM-SHA-512是 cnorthorth-1、cnorth-1、cnorth-1、cnorth-1、cnorth us-gov-east -1、cnorthest us-gov-west -1

sasl.plain.username

用于从 Secrets Manager 中检索密钥字符串的用户名。当您为 security.protocol 指定 SASL_SSL、为 sasl.mechanism 指定 PLAIN 时，则需要此值。

sasl.plain.password

用于从 Secrets Manager 中检索密钥字符串的密码。当您为 security.protocol 指定 SASL_SSL、为 sasl.mechanism 指定 PLAIN 时，则需要此值。

sasl.scram.username

用于从 Secrets Manager 中检索密钥字符串的用户名。当您为 security.protocol 指定 SASL_SSL、为 sasl.mechanism 指定 SCRAM-SHA-512 时，则需要此值。

sasl.scram.password

用于从 Secrets Manager 中检索密钥字符串的密码。当您为 security.protocol 指定 SASL_SSL、为 sasl.mechanism 指定 SCRAM-SHA-512 时，则需要此值。

sasl.kerberos.keytab

Secrets Manager 中用于 Kerberos 身份验证的 keytab 文件。当您为 `security.protocol` 指定 `SASL_SSL`、为 `sasl.mechanism` 指定 `GSSAPI` 时，则需要此值。

此字段支持替换模板。您必须使用 Secrets Manager 来存储连接到您的 Kafka 代理所需的凭证。使用 get_secret SQL 函数检索此字段的值。有关替换模板的更多信息，请参阅 the section called “替换模板” (p. 585)。有关 get_secret SQL 函数的更多信息，请参阅 the section called “get_secret(secretId, secretType, key, roleArn)” (p. 557)。使用 `SecretBinary` 参数。

sasl.kerberos.service.name

Apache Kafka 运行的 Kerberos 主要名称。当您为 `security.protocol` 指定 `SASL_SSL`、为 `sasl.mechanism` 指定 `GSSAPI` 时，则需要此值。

sasl.kerberos.krb5.kdc

您的 Apache Kafka 生成器客户端连接到的密钥分配中心 (KDC) 的主机名。当您为 `security.protocol` 指定 `SASL_SSL`、为 `sasl.mechanism` 指定 `GSSAPI` 时，则需要此值。

sasl.kerberos.krb5.realm

您的 Apache Kafka 生成器客户端连接到的领域。当您为 `security.protocol` 指定 `SASL_SSL`、为 `sasl.mechanism` 指定 `GSSAPI` 时，则需要此值。

sasl.kerberos.principal

Kerberos 可以为其分配票证以访问 Kerberos 感知服务的唯一 Kerberos 身份。当您为 `security.protocol` 指定 `SASL_SSL`、为 `sasl.mechanism` 指定 `GSSAPI` 时，则需要此值。

示例

下面的 JSON 示例介绍了如何在 Amazon IoT 规则中定义 Apache Kafka 操作：

```
{
"topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
    {
        "kafka": {
            "destinationArn": "arn:aws:iot:region:123456789012:ruledestination/
vpc/VPCDestinationARN",
            "topic": "TopicName",
            "clientProperties": {
                "bootstrap.servers": "kafka.com:9092",
                "security.protocol": "SASL_SSL",
                "ssl.truststore": "${get_secret('kafka_client_truststore', 'SecretBinary',
'arn:aws:iam::123456789012:role/kafka-get-secret-role-name')}",
                "ssl.truststore.password": "kafka password",
               "sasl.mechanism": "GSSAPI",
               "sasl.kerberos.service.name": "kafka",
               "sasl.kerberos.krb5.kdc": "kerberosdns.com",
               "sasl.kerberos.keytab": "${get_secret('kafka_keytab',
               'SecretBinary', 'arn:aws:iam::123456789012:role/kafka-get-secret-role-
name')}",
               "sasl.kerberos.krb5.realm": "KERBEROSREALM",
               "sasl.kerberos.principal": "kafka-keytab/kafka-keytab.com"
            }
        }
    }
]
```

```
}
```

有关 Kerberos 设置的重要注意事项

- 您的密钥分配中心 (KDC) 必须通过目标 VPC 内的私有域名系统 (DNS) 进行解析。一种可能的方法是将 KDC DNS 条目添加到私有托管区域。有关此方法的更多信息，请参阅如何使用私有托管区域。
- 每个 VPC 都必须启用 DNS 解析。有关更多信息，请参阅将 DNS 与您的 VPC 一起使用。
- VPC 目标中的网络接口安全组和实例级安全组必须允许来自 VPC 内部以下端口的流量。
  - 引导代理侦听器端口上的 TCP 流量（通常为 9092，但必须在 9000-9100 范围内）
  - KDC 端口 88 上的 TCP 和 UDP 流量
- SCRAM-SHA-512是 cnorthorth-1、cnorth-1、cnorth-1、cnorth-1、cnorth us-gov-east -1、cnorthest us-gov-west -1

Virtual Private Cloud (VPC) 目标

Apache Kafka 规则操作将数据路由到 Amazon Virtual Private Cloud (Amazon VPC) 中的 Apache Kafka 集群。当您为规则操作指定 VPC 目标时，将自动启用 Apache Kafka 规则操作使用的 VPC 配置。

VPC 目标包含 VPC 内部的子网列表。规则引擎将在您在此列表中指定的每个子网中创建一个弹性网络接口。有关网络接口的更多信息，请参阅 Amazon EC2 用户指南中的弹性网络接口。

要求和注意事项

- 如果您使用的是自我管理的 Apache Kafka 集群，则将在互联网上使用公有终端节点访问此集群：
  - 您必须为子网中的实例创建 NAT 网关。NAT 网关具有可以连接到互联网的公有 IP 地址，这允许规则引擎将您的消息转发到公有 Kafka 集群。
  - 作为 NAT 网关的廉价替代方案，您可以使用由 VPC 目标创建的弹性网络接口 (ENI) 分配弹性 IP 地址。必须将您使用的安全组配置为阻止传入流量。

    Note

    如果 VPC 目标被禁用，然后重新启用，则必须将弹性 IP 与新的 ENI 重新关联。
- 如果 VPC 主题规则目标连续30天未收到任何流量，该目标将被禁用。
- 如果 VPC 目标使用的任何资源发生变化，目标将被禁用且无法使用。
- 可以禁用 VPC 目标的一些更改包括：删除 VPC、子网、安全组或使用的角色；将角色修改为不再拥有必要的权限；以及禁用目标。

定价

出于定价目的，除了在资源位于您的 VPC 中时向资源发送消息的操作之外，还会计量 VPC 规则操作。有关定价信息，请参阅 Amazon IoT Core 定价。

创建 Virtual Private Cloud (VPC) 主题规则目标

使用 CreateTopicRuleDestinationAPI 或Amazon IoT Core控制台，创建 Virtual Private Cloud (VPC) 目标。

当您创建 VPC 目标时，必须指定以下信息。

vpcId

VPC 目标的唯一 ID。

subnetIds

规则引擎在其中创建弹性网络接口的子网列表。规则引擎为列表中的每个子网分配一个网络接口。

securityGroups（可选）

要应用到网络接口的安全组列表。

roleArn

有权代表您创建网络接口的角色的 Amazon Resource Name (ARN)。

此 ARN 应附加一个类似于以下示例的策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/VPCDestinationENI": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": "CreateNetworkInterface",
                    "aws:RequestTag/VPCDestinationENI": "true"
                }
            }
        }
    ]
}
```

使用 Amazon CLI 创建 VPC 目标

以下示例显示如何使用 Amazon CLI 创建 VPC 目标。

```
aws --region regions iot create-topic-rule-destination --destination-configuration
 'vpcConfiguration={subnetIds=["subnet-123456789101230456"],securityGroups=[],vpcId="vpc-
123456789101230456",roleArn="arn:aws:iam::123456789012:role/role-name"}'
```

运行此命令后，VPC 目标状态将为 IN_PROGRESS。几分钟后，其状态将更改为 ERROR（如果命令不成功）或 ENABLED。当目标状态为 ENABLED 时，目标即可使用。

您可以使用以下命令获取 VPC 目标的状态。

```
aws --region region iot get-topic-rule-destination --arn "VPCDestinationARN"
```

使用 Amazon IoT Core 创建 VPC 目标控制台

以下步骤介绍如何使用 Amazon IoT Core 控制台创建 VPC 目标。

1. 导航到 Amazon IoT Core 控制台。在左侧的窗格中，Act（操作）选项卡下，选择 Destination（目标）。
2. 输入以下字段的值。
   - VPC ID
   - 子网 ID
   - 安全组
3. 选择具有创建网络接口所需权限的角色。前面的示例策略包含这些权限。

当 VPC 目标状态为 ENABLED（已启用）时，目标即可使用。

CloudWatch 警报

告 CloudWatch 警 (`cloudWatchAlarm`) 操作会更改 Amazon CloudWatch 告警的状态。您可以在此调用中指定状态更改原因和状态值。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `cloudwatch:SetAlarmState` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`alarmName`

CloudWatch 警报名称。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`stateReason`

警报更改的原因。

支持替换模板 (p. 585)：是

`stateValue`

警报状态的值。有效值：`OK`、`ALARM`、`INSUFFICIENT_DATA`。

支持替换模板 (p. 585)：是

`roleArn`

允许访问 CloudWatch 告警的 IM 角色。有关更多信息，请参阅要求 (p. 457)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例定义了Amazon IoT规则中的 CloudWatch 告警操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchAlarm": {
                    "alarmName": "IotAlarm",
                    "stateReason": "Temperature stabilized.",
                    "stateValue": "OK",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

另请参阅

- 什么是亚马逊 CloudWatch？ 在亚马逊 CloudWatch 用户指南中
- 在《亚马逊 CloudWatch 用户指南》中使用亚马逊 CloudWatch警报

CloudWatch 日志

CloudWatch Logs (`cloudwatchLogs`) 操作将数据发送到亚马逊 CloudWatch 日志。您可以使用 `batchMode` 在一条消息中上载多条设备日志记录并为其加上时间戳。您还可以指定操作要向其发送数据的日志组。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `logs:CreateLogStream`、`logs:DescribeLogStreams`、和 `logs:PutLogEvents` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用客户托管式Amazon KMS key（KMS 密钥）对 Log CloudWatch s 中的日志数据进行静态加密，服务必须具有代表调用方使用 KMS 的权限。有关更多信息，请参阅 Amazon Lo CloudWatch gs 用户指南Amazon KMS中的使用加密 CloudWatch 日志数据。

batchMode 的 MQTT 消息格式要求

如果您在batchMode关闭的情况下使用 CloudWatch Logs 规则操作，则没有 MQTT 消息格式要求。注意：这是参数的原定设置状态，其值为 false。但是，如果您在（参数值为）batchMode处于开启状态（参数值为true）时使用 Logs 规则操作，则包含设备端日志的 MQTT 消息必须格式化为包含时间戳和消息有效负载。 CloudWatch 注意：timestamp表示事件发生的时间，以自 1970 年 1 月 1 日 00:00:00 UTC 之后的毫秒数。

以下为发布格式的示例：

```
[
  {"timestamp": 1673520691093, "message": "Test message 1"},
  {"timestamp": 1673520692879, "message": "Test message 2"},
  {"timestamp": 1673520693442, "message": "Test message 3"}
]
```

根据生成设备端日志的方式，在发送这些日志之前，可能需要对其进行筛选和重新格式化，以符合此要求。有关更多信息，请参阅 MQTT 消息有效负载。

无论 batchMode 参数如何，message内容都必须符合 Amazon IoT 消息大小限制。有关更多信息，请参阅 Amazon IoT Core 终端节点和配额。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

logGroupName

操作要向其发送数据 CloudWatch 的日志组。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

roleArn

允许访问 CloudWatch 日志组的 IM 角色。有关更多信息，请参阅要求 (p. 458)：

支持替换模板 (p. 585)：否

（可选）batchMode

表示是否将提取批量日志记录并将其上载到批量 CloudWatch。值包括 true 或 false（原定设置）。有关更多信息，请参阅要求 (p. 458)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例定义 CloudWatch 了Amazon IoT规则中的 Logs。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchLogs": {
                    "logGroupName": "IotLogs",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw",
                    "batchMode": false
                }
```

```
            }
        ]
    }
}
```

另请参阅

- 什么是亚马逊 CloudWatch 日志？ 在《亚马逊 CloudWatch 日志用户指南》中

CloudWatch 指标

CloudWatch 指标 (`cloudwatchMetric`) 操作捕获亚马逊 CloudWatch指标。您可以指定指标命名空间、名称、值、单位和时间戳。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `cloudwatch:PutMetricData` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`metricName`

CloudWatch 指标名称。

支持替换模板 (p. 585)：是

`metricNamespace`

CloudWatch 指标命名空间名称。

支持替换模板 (p. 585)：是

`metricUnit`

支持的公制单位 CloudWatch。

支持替换模板 (p. 585)：是

`metricValue`

包含 CloudWatch 指标值的字符串。

支持替换模板 (p. 585)：是

`metricTimestamp`

（可选）包含用 Unix 纪元时间表示的 timestamp（以秒为单位）的字符串。默认为当前 Unix 纪元时间。

支持替换模板 (p. 585)：是

`roleArn`

允许访问 CloudWatch 指标的 IM 角色。有关更多信息，请参阅要求 (p. 460)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例定义了Amazon IoT规则中的 CloudWatch 指标操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "IotMetric",
                    "metricNamespace": "IotNamespace",
                    "metricUnit": "Count",
                    "metricValue": "1",
                    "metricTimestamp": "1456821314",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

下面的 JSON 示例定义了一个在Amazon IoT规则中带有替代模板的 CloudWatch 指标操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "cloudwatchMetric": {
                    "metricName": "${topic()}",
                    "metricNamespace": "${namespace}",
                    "metricUnit": "${unit}",
                    "metricValue": "${value}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_cw"
                }
            }
        ]
    }
}
```

另请参阅

- 什么是亚马逊 CloudWatch？ 在亚马逊 CloudWatch 用户指南中
- 使用《亚马逊 CloudWatch 用户指南》中的亚马逊 CloudWatch指标

DynamoDB

DynamoDB (dynamoDB) 操作可将所有或部分 MQTT 消息写入 Amazon DynamoDB 表。

您可以按照教程执行操作，该教程向您说明如何使用 DynamoDB 操作创建并测试规则。有关更多信息，请参阅教程：将设备数据存储在 DynamoDB 表中 (p. 188)：

Note

此规则将非 JSON 数据作为二进制数据写入到 DynamoDB 中。DynamoDB 控制台以 Base64 编码文本格式显示数据。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `dynamodb:PutItem` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用客户托管的 Amazon KMS key （KMS 密钥）对 DynamoDB 中的数据进行静态加密，服务必须具有代表调用方使用 KMS 的权限。有关更多信息，请参阅 Amazon DynamoDB 入门指南中的客户托管式 KMS。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`tableName`

DynamoDB 表的名称。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`hashKeyField`

哈希键（也称为分区键）的名称。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`hashKeyType`

（可选）哈希键（也称为分区键）的数据类型。有效值：`STRING`、`NUMBER`。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`hashKeyValue`

哈希键的值。考虑使用替代模板，例如 `${topic()}` 或者 `${timestamp()}`。

支持替换模板 (p. 585)：是

`rangeKeyField`

（可选）范围键（也称为排序键）的名称。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`rangeKeyType`

（可选）范围键（也称为排序键）的数据类型。有效值：`STRING`、`NUMBER`。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`rangeKeyValue`

（可选）范围键的值。考虑使用替代模板，例如 `${topic()}` 或者 `${timestamp()}`。

支持替换模板 (p. 585)：是

`payloadField`

（可选）负载将写入的列的名称。如果省略此值，负载将写入 `payload` 列。

支持替换模板 (p. 585)：是

`operation`

（可选）要执行的操作类型。有效值：`INSERT`、`UPDATE`、`DELETE`。

支持替换模板 (p. 585)：是

`roleARN`

允许访问 DynamoDB 表的 IAM 角色。有关更多信息，请参阅要求 (p. 462)：

支持替换模板 (p. 585)：否

写入 DynamoDB 表的数据是规则的 SQL 语句的结果。

示例

下面的 JSON 示例定义了 Amazon IoT 规则中的 DynamoDB 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDB": {
                    "tableName": "my_ddb_table",
                    "hashKeyField": "key",
                    "hashKeyValue": "${topic()}",
                    "rangeKeyField": "timestamp",
                    "rangeKeyValue": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDB"
                }
            }
        ]
    }
}
```

另请参阅

- Amazon DynamoDB 开发人员指南中的什么是 DynamoDB？
- Amazon DynamoDB 开发人员指南中的 DynamoDB 入门
- 教程：将设备数据存储在 DynamoDB 表中 (p. 188)

DynamoDBv2

DynamoDBv2 (`dynamoDBv2`) 操作可将所有或部分 MQTT 消息写入 Amazon DynamoDB 表。负载中的每个属性将写入 DynamoDB 数据库中的单独一列。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `dynamodb:PutItem` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。

- 如果要定义 MQTT 消息负载，则它必须包含一个与表的主分区键相匹配的根级键，以及一个与表的主排序键相匹配的根级键。
- 如果您使用客户托管的 Amazon KMS key （KMS 密钥）对 DynamoDB 中的数据进行静态加密，服务必须具有代表调用方使用 KMS 的权限。有关更多信息，请参阅 Amazon DynamoDB 入门指南中的客户托管式 KMS。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`putItem`

用于指定消息数据将写入的 DynamoDB 表的对象。该对象必须具备以下信息：

`tableName`

DynamoDB 表的名称。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`roleARN`

允许访问 DynamoDB 表的 IAM 角色。有关更多信息，请参阅要求 (p. 463)：

支持替换模板 (p. 585)：否

写入 DynamoDB 表的数据是规则的 SQL 语句的结果。

示例

以下 JSON 示例定义了在 Amazon IoT 规则中的 DynamoDBv2 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * AS message FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "dynamoDBv2": {
                    "putItem": {
                        "tableName": "my_ddb_table"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2",
                }
            }
        ]
    }
}
```

以下 JSON 示例定义了在 Amazon IoT 规则中带有替换模板的 DynamoDB 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2015-10-08",
        "actions": [
            {
                "dynamoDBv2": {
```

```
                    "putItem": {
                        "tableName": "${topic()}"
                    },
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_dynamoDBv2"
                }
            }
        ]
    }
}
```

另请参阅

- Amazon DynamoDB 开发人员指南中的什么是 DynamoDB？
- Amazon DynamoDB 开发人员指南中的 DynamoDB 入门

Elasticsearch

Elasticsearch (`elasticsearch`) 操作可将 MQTT 消息中的数据写入 Amazon Ser OpenSearch vice 域。然后，您可以使用像 Dashboards OpenSearch 这样的工具来查询和可视化 Serv OpenSearch ice 中的数据。

> Warning
>
> `Elasticsearch` 操作只能由现有规则操作使用。要创建新的规则操作或更新现有规则操作，请使用 `OpenSearch` 规则而不是操作。有关更多信息，请参阅OpenSearch (p. 508)：

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `es:ESHttpPut` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用客户托管式Amazon KMS key（KMS 密钥）对中的 OpenSearch数据进行静态加密，服务必须具有代表调用方使用 KMS 的权限。有关更多信息，请参阅 Amazon S ervice 开发人员指南中的为 Amazon S OpenSearch erv OpenSearch ice 静态加密数据。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`endpoint`

您的服务域端点。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`index`

您要在其中存储数据的索引。

支持替换模板 (p. 585)：是

`type`

您存储的文档类型。

支持替换模板 (p. 585)：是

`id`

每个文档的唯一标识符。

支持替换模板 (p. 585)：是

`roleARN`

允许访问 Service 域的 OpenSearch IM 角色。有关更多信息，请参阅要求 (p. 465)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例定义了 Amazon IoT 规则中的 Elasticsearch 操作，以及如何为 `elasticsearch` 操作指定域。有关更多信息，请参阅ElasticsearchAction。

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "my-type",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

下面的 JSON 示例定义了一个 Amazon IoT 规则中带有替代模板的 Elasticsearch 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "elasticsearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es"
                }
            }
        ]
    }
}
```

另请参阅

- OpenSearch (p. 508)
- 什么是亚马逊 OpenSearch 服务？

HTTP

HTTPS (`http`) 操作可将 MQTT 消息中的数据发送至 Web 应用程序或服务。

要求

此规则操作具有以下要求：

- 您必须先确认并启用 HTTPS 端点，然后规则引擎才能使用它们。有关更多信息，请参阅使用 HTTP 主题规则目标 (p. 469)：

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`url`

使用 HTTP POST 方法发送消息的 HTTPS 端点。如果使用 IP 地址代替主机名，则该地址必须是 IPv4 地址。不支持 IPv6 地址。

支持替换模板 (p. 585)：是

`confirmationUrl`

（可选）如果指定，则 Amazon IoT 使用确认 URL 创建匹配的主题规则目标。您必须先启用主题规则目标，然后才能在 HTTP 操作中使用它。有关更多信息，请参阅使用 HTTP 主题规则目标 (p. 469)：如果您使用替换模板，则必须先手动创建主题规则目标，然后才能使用 `http` 操作。`confirmationUrl` 必须是 `url` 的前缀。

`url` 和 `confirmationUrl` 之间的关系由以下内容描述：

- 如果 `url` 为硬编码且未提供 `confirmationUrl`，则我们隐式地将 `url` 字段视为 `confirmationUrl`。Amazon IoT 为 `url` 创建主题规则目标。
- 如果 `url` 和 `confirmationUrl` 为硬编码，则 `url` 必须以 `confirmationUrl` 开头。Amazon IoT 为 `confirmationUrl` 创建主题规则目标。
- 如果 `url` 包含替换模板，则必须指定 `confirmationUrl` 并且 `url` 必须以 `confirmationUrl` 开头。如果 `confirmationUrl` 包含替换模板，则必须先手动创建主题规则目标，然后才能使用 `http` 操作。如果 `confirmationUrl` 不包含替换模板，则 Amazon IoT 为 `confirmationUrl` 创建主题规则目标。

支持替换模板 (p. 585)：是

`headers`

（可选）要包含在对端点的 HTTP 请求中的标头列表。每个标头都必须包含以下信息：

`key`

标头的键。

支持替换模板 (p. 585)：否

`value`

标头的值。

支持替换模板 (p. 585)：是

Note

当负载为 JSON 格式时，默认内容类型是 application/json。否则，它是 application/octet-stream。您可以在标头中，使用键内容类型（不区分大小写）指定确切的内容类型来覆盖它。

`auth`

（可选）规则引擎在连接到 `url` 参数中指定的端点 URL 时使用的身份验证。目前，Signature Version 4 是唯一支持的身份验证类型。有关更多信息，请参阅 HTTP 授权。

支持替换模板 (p. 585)：否

示例

以下 JSON 示例定义了一个带有 HTTP 操作的 Amazon IoT 规则。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "http": {
                    "url": "https://www.example.com/subpath",
                    "confirmationUrl": "https://www.example.com",
                    "headers": [
                        {
                            "key": "static_header_key",
                            "value": "static_header_value"
                        },
                        {
                            "key": "substitutable_header_key",
                            "value": "${value_from_payload}"
                        }
                    ]
                }
            }
        ]
    }
}
```

HTTP 操作重试逻辑

Amazon IoT 规则引擎根据以下规则重试 HTTP 操作：

- 规则引擎尝试至少发送一次消息。
- 规则引擎最多重试两次。最大尝试次数为三次。
- 在下列情况中，规则引擎不会尝试重试：
  - 之前的尝试提供了大于 16384 字节的响应。
  - 下游 Web 服务或应用程序在尝试之后关闭了 TCP 连接。
  - 完成包括重试在内的请求总用时超过了请求超时限制。
  - 该请求返回除 429、500-599 之外的 HTTP 状态代码。

  Note

  标准数据传输费用适用于重试操作。

另请参阅

- 使用 HTTP 主题规则目标 (p. 469)
- Amazon物联网博客中的直接从 Amazon IoT Core 中路由数据到 Web 服务

使用 HTTP 主题规则目标

HTTP 主题规则目标是规则引擎可以将数据从主题规则路由到的 Web 服务。Amazon IoT Core 资源描述了 Amazon IoT 的网络服务。主题规则目标资源可以通过不同的规则共享。

Amazon IoT Core 可以将数据发送到另一个 Web 服务之前，必须确认可以访问该服务的端点。

HTTP 主题规则目标概述

HTTP 主题规则目标指的是支持确认 URL 和一个或多个数据收集 URL 的网络服务。HTTP 主题规则目标资源包含 Web 服务的确认 URL。配置 HTTP 主题规则操作时，您可以指定应该接收数据的端点的实际 URL 以及网络服务的确认 URL。确认目标后，主题规则会将 SQL 语句的结果发送到 HTTPS 端点（而不是发送到确认 URL）。

HTTP 主体规则目标可以处于以下状态之一：

已启用

已经确认目标，可以由规则操作使用。目标必须处于 ENABLED 状态才能在规则中使用。您只能启用处于 DISABLED 状态的目标。

DISABLED

已经确认目标，但规则操作无法使用。如果您希望暂时阻止流入终端节点的流量而无需再次完成确认流程，则此功能非常有用。您只能禁用处于 ENABLED 状态的目标。

IN_PROGRESS

正在确认目标。

ERROR (错误)

目标确认超时。

确认并启用 HTTP 主题规则目标后，可以与帐户中的任何规则一起使用。

以下各节介绍了对 HTTP 主题规则目标的常见操作。

创建和确认 HTTP 主题规则目标

您可以通过调用 `CreateTopicRuleDestination` 操作或使用 Amazon IoT 控制台来创建 HTTP 主题规则目标。

创建目标后，Amazon IoT 向确认 URL 发送确认请求。确认请求的格式如下：

```
HTTP POST {confirmationUrl}/?confirmationToken={confirmationToken}
Headers:
x-amz-rules-engine-message-type: DestinationConfirmation
x-amz-rules-engine-destination-arn:"arn:aws:iot:us-east-1:123456789012:ruledestination/
http/7a280e37-b9c6-47a2-a751-0703693f46e4"
Content-Type: application/json
Body:
{
    "arn":"arn:aws:iot:us-east-1:123456789012:ruledestination/http/7a280e37-b9c6-47a2-
a751-0703693f46e4",
    "confirmationToken": "AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "enableUrl": "https://iot.us-east-1.amazonaws.com/confirmdestination/
AYADeMXLrPrNY2wqJAKsFNn-…NBJndA",
    "messageType": "DestinationConfirmation"
}
```

确认请求的内容包含以下信息：

arn

要确认的主题规则目标的 Amazon Resource Name (ARN)。

confirmationToken

Amazon IoT Core 发送的确认令牌 示例中是已截断的令牌。您的令牌会更长。您需要此令牌用 Amazon IoT Core 来确认您的目的地。

enableUrl

您需要浏览以确认主题规则目标的 URL。

messageType

消息类型。

要完成端点确认过程，必须在确认 URL 收到确认请求后执行以下操作之一。

- 在确认请求中调用 `enableUrl`，然后调用 `UpdateTopicRuleDestination` 将主题规则的状态设置为 `ENABLED`。
- 调用 `ConfirmTopicRuleDestination` 操作并从确认请求中传递 `confirmationToken`。
- 复制 `confirmationToken` 然后粘贴到 Amazon IoT 控制台的目标确认对话框。

发送新确认请求

要为目标触发新确认消息，请调用 `UpdateTopicRuleDestination` 并将主题规则目标的状态设置为 `IN_PROGRESS`。

发送新的确认请求后，您需要重复确认过程

禁用并删除主题规则目标

要禁用目标，请调用 `UpdateTopicRuleDestination` 并将主题规则目标的状态设置为 `DISABLED`。可以再次启用处于 DISABLED 状态的主题规则，而无需发送新的确认请求。

要删除主题规则目标，请调用 `DeleteTopicRuleDestination`。

主题规则目标中的 HTTPS 端点支持的证书颁发机构

主题规则目标中的 HTTPS 端点支持以下证书颁发机构。您可以选择以下支持的证书颁发机构之一。签名仅供参考。请注意，你不能使用自签名证书，因为它们不起作用。

帮助我们改进此主题

让我们知道您的想法。

```
Alias name: swisssignplatinumg2ca
Certificate fingerprints:
  MD5:  C9:98:27:77:28:1E:3D:0E:15:3C:84:00:B8:85:03:E6
  SHA1: 56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
  SHA256:
 3B:22:2E:56:67:11:E9:92:30:0D:C0:B1:5A:B9:47:3D:AF:DE:F8:C8:4D:0C:EF:7D:33:17:B4:C1:82:1D:14:36

Alias name: hellenicacademicandresearchinstitutionsrootca2011
Certificate fingerprints:
  MD5:  73:9F:4C:4B:73:5B:79:E9:FA:BA:1C:EF:6E:CB:D5:C9
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:71

Alias name: teliasonerarootcav1
```

```
Certificate fingerprints:
  MD5:  37:41:49:1B:18:56:9A:26:F5:AD:C2:66:FB:40:A5:4C
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:89

Alias name: geotrustprimarycertificationauthority
Certificate fingerprints:
  MD5:  02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6C

Alias name: trustisfpsrootca
Certificate fingerprints:
  MD5:  30:C9:E7:1E:6B:E6:14:EB:65:B2:16:69:20:31:67:4D
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7D

Alias name: quovadisrootca3g3
Certificate fingerprints:
  MD5:  DF:7D:B9:AD:54:6F:68:A1:DF:89:57:03:97:43:B0:D7
  SHA1: 48:12:BD:92:3C:A8:C4:39:06:E7:30:6D:27:96:E6:A4:CF:22:2E:7D
  SHA256:
 88:EF:81:DE:20:2E:B0:18:45:2E:43:F8:64:72:5C:EA:5F:BD:1F:C2:D9:D2:05:73:07:09:C5:D8:B8:69:0F:46

Alias name: buypassclass2ca
Certificate fingerprints:
  MD5:  46:A7:D2:FE:45:FB:64:5A:A8:59:90:9B:78:44:9B:29
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:48

Alias name: secureglobalca
Certificate fingerprints:
  MD5:  CF:F4:27:0D:D4:ED:DC:65:16:49:6D:3D:DA:BF:6E:DE
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:69

Alias name: chunghwaepkirootca
Certificate fingerprints:
  MD5:  1B:2E:00:CA:26:06:90:3D:AD:FE:6F:15:68:D3:6B:B3
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D5

Alias name: verisignclass2g2ca
Certificate fingerprints:
  MD5:  2D:BB:E5:25:D3:D1:65:82:3A:B7:0E:FA:E6:EB:E2:E1
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A1

Alias name: szafirrootca2
Certificate fingerprints:
  MD5:  11:64:C1:89:B0:24:B1:8C:B1:07:7E:89:9E:51:9E:99
  SHA1: E2:52:FA:95:3F:ED:DB:24:60:BD:6E:28:F3:9C:CC:CF:5E:B3:3F:DE
  SHA256:
 A1:33:9D:33:28:1A:0B:56:E5:57:D3:D3:2B:1C:E7:F9:36:7E:B0:94:BD:5F:A7:2A:7E:50:04:C8:DE:D7:CA:FE

Alias name: quovadisrootca1g3
Certificate fingerprints:
  MD5:  A4:BC:5B:3F:FE:37:9A:FA:64:F0:E2:FA:05:3D:0B:AB
  SHA1: 1B:8E:EA:57:96:29:1A:C9:39:EA:B8:0A:81:1A:73:73:C0:93:79:67
```

```
  SHA256:
 8A:86:6F:D1:B2:76:B5:7E:57:8E:92:1C:65:82:8A:2B:ED:58:E9:F2:F2:88:05:41:34:B7:F1:F4:BF:C9:CC:74

Alias name: utndatacorpsgcca
Certificate fingerprints:
  MD5:  B3:A5:3E:77:21:6D:AC:4A:C0:C9:FB:D5:41:3D:CA:06
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:48

Alias name: autoridaddecertificacionfirmaprofesionalcifa62634068
Certificate fingerprints:
  MD5:  73:3A:74:7A:EC:BB:A3:96:A6:C2:E4:E2:C8:9B:C0:C3
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:EF

Alias name: securesignrootca11
Certificate fingerprints:
  MD5:  B7:52:74:E2:92:B4:80:93:F2:75:E4:CC:D7:F2:EA:26
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:12

Alias name: amazon-ca-g4-acm2
Certificate fingerprints:
  MD5:  B2:F1:03:2B:93:64:05:80:B8:A8:17:36:B9:1B:52:3C
  SHA1: A7:E6:45:32:1F:7A:B7:AD:C0:70:EA:73:5F:AB:ED:C3:DA:B4:D0:C8
  SHA256:
 D7:A8:7C:69:95:D0:E2:04:2A:32:70:A7:E2:87:FE:A7:E8:F4:C1:70:62:F7:90:C3:EB:BB:53:F2:AC:39:26:BE

Alias name: isrgrootx1
Certificate fingerprints:
  MD5:  0C:D2:F9:E0:DA:17:73:E9:ED:86:4D:A5:E3:70:E7:4E
  SHA1: CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
  SHA256:
 96:BC:EC:06:26:49:76:F3:74:60:77:9A:CF:28:C5:A7:CF:E8:A3:C0:AA:E1:1A:8F:FC:EE:05:C0:BD:DF:08:C6

Alias name: amazon-ca-g4-acm1
Certificate fingerprints:
  MD5:  E2:F1:18:19:61:5C:43:E0:D4:A8:5D:0B:FA:7C:89:1B
  SHA1: F2:0D:28:B6:29:C2:2C:5E:84:05:E6:02:4D:97:FE:8F:A0:84:93:A0
  SHA256:
 B0:11:A4:F7:29:6C:74:D8:2B:F5:62:DF:87:D7:28:C7:1F:B5:8C:F4:E6:73:F2:78:FC:DA:F3:FF:83:A6:8C:87

Alias name: etugracertificationauthority
Certificate fingerprints:
  MD5:  B8:A1:03:63:B0:BD:21:71:70:8A:6F:13:3A:BB:79:49
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3C

Alias name: geotrustuniversalca2
Certificate fingerprints:
  MD5:  34:FC:B8:D0:36:DB:9E:14:B3:C2:F2:DB:8F:E4:94:C7
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0B

Alias name: digicertglobalrootca
Certificate fingerprints:
  MD5:  79:E4:A9:84:0D:7D:3A:96:D7:C0:4F:E2:43:4C:89:2E
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:61
```

```
Alias name: staatdernederlandenevrootca
Certificate fingerprints:
  MD5:  FC:06:AF:7B:E8:1A:F1:9A:B4:E8:D2:70:1F:C0:F5:BA
  SHA1: 76:E2:7E:C1:4F:DB:82:C1:C0:A6:75:B5:05:BE:3D:29:B4:ED:DB:BB
  SHA256:
 4D:24:91:41:4C:FE:95:67:46:EC:4C:EF:A6:CF:6F:72:E2:8A:13:29:43:2F:9D:8A:90:7A:C4:CB:5D:AD:C1:5A

Alias name: utnuserfirstclientauthemailca
Certificate fingerprints:
  MD5:  D7:34:3D:EF:1D:27:09:28:E1:31:02:5B:13:2B:DD:F7
  SHA1: B1:72:B1:A5:6D:95:F9:1F:E5:02:87:E1:4D:37:EA:6A:44:63:76:8A
  SHA256:
 43:F2:57:41:2D:44:0D:62:74:76:97:4F:87:7D:A8:F1:FC:24:44:56:5A:36:7A:E6:0E:DD:C2:7A:41:25:31:AE

Alias name: actalisauthenticationrootca
Certificate fingerprints:
  MD5:  69:C1:0D:4F:07:A3:1B:C3:FE:56:3D:04:BC:11:F6:A6
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:66

Alias name: amazonrootca4
Certificate fingerprints:
  MD5:  89:BC:27:D5:EB:17:8D:06:6A:69:D5:FD:89:47:B4:CD
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:92

Alias name: amazonrootca3
Certificate fingerprints:
  MD5:  A0:D4:EF:0B:F7:B5:D8:49:95:2A:EC:F5:C4:FC:81:87
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A4

Alias name: amazonrootca2
Certificate fingerprints:
  MD5:  C8:E5:8D:CE:A8:42:E2:7A:C0:2A:5C:7C:9E:26:BF:66
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B4

Alias name: amazonrootca1
Certificate fingerprints:
  MD5:  43:C6:BF:AE:EC:FE:AD:2F:18:C6:88:68:30:FC:C8:E6
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6E

Alias name: affirmtrustpremium
Certificate fingerprints:
  MD5:  C4:5D:0E:48:B6:AC:28:30:4E:0A:BC:F9:38:16:87:57
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9A

Alias name: keynectisrootca
Certificate fingerprints:
  MD5:  CC:4D:AE:FB:30:6B:D8:38:FE:50:EB:86:61:4B:D2:26
  SHA1: 9C:61:5C:4D:4D:85:10:3A:53:26:C2:4D:BA:EA:E4:A2:D2:D5:CC:97
  SHA256:
 42:10:F1:99:49:9A:9A:C3:3C:8D:E0:2B:A6:DB:AA:14:40:8B:DD:8A:6E:32:46:89:C1:92:2D:06:97:15:A3:32

Alias name: equifaxsecureglobalebusinessca1
Certificate fingerprints:
  MD5:  51:F0:2A:33:F1:F5:55:39:07:F2:16:7A:47:C7:5D:63
```

```
  SHA1: 3A:74:CB:7A:47:DB:70:DE:89:1F:24:35:98:64:B8:2D:82:BD:1A:36
  SHA256:
 86:AB:5A:65:71:D3:32:9A:BC:D2:E4:E6:37:66:8B:A8:9C:73:1E:C2:93:B6:CB:A6:0F:71:63:40:A0:91:CE:AE

Alias name: affirmtrustpremiumca
Certificate fingerprints:
  MD5:  C4:5D:0E:48:B6:AC:28:30:4E:0A:BC:F9:38:16:87:57
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9A

Alias name: baltimorecodesigningca
Certificate fingerprints:
  MD5:  90:F5:28:49:56:D1:5D:2C:B0:53:D4:4B:EF:6F:90:22
  SHA1: 30:46:D8:C8:88:FF:69:30:C3:4A:FC:CD:49:27:08:7C:60:56:7B:0D
  SHA256:
 A9:15:45:DB:D2:E1:9C:4C:CD:F9:09:AA:71:90:0D:18:C7:35:1C:89:B3:15:F0:F1:3D:05:C1:3A:8F:FB:46:87

Alias name: gdcatrustauthr5root
Certificate fingerprints:
  MD5:  63:CC:D9:3D:34:35:5C:6F:53:A3:E2:08:70:48:1F:B4
  SHA1: 0F:36:38:5B:81:1A:25:C3:9B:31:4E:83:CA:E9:34:66:70:CC:74:B4
  SHA256:
 BF:FF:8F:D0:44:33:48:7D:6A:8A:A6:0C:1A:29:76:7A:9F:C2:BB:B0:5E:42:0F:71:3A:13:B9:92:89:1D:38:93

Alias name: certinomisrootca
Certificate fingerprints:
  MD5:  14:0A:FD:8D:A8:28:B5:38:69:DB:56:7E:61:22:03:3F
  SHA1: 9D:70:BB:01:A5:A4:A0:18:11:2E:F7:1C:01:B9:32:C5:34:E7:88:A8
  SHA256:
 2A:99:F5:BC:11:74:B7:3C:BB:1D:62:08:84:E0:1C:34:E5:1C:CB:39:78:DA:12:5F:0E:33:26:88:83:BF:41:58

Alias name: verisignclass3publicprimarycertificationauthorityg5
Certificate fingerprints:
  MD5:  CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:DF

Alias name: verisignclass3publicprimarycertificationauthorityg4
Certificate fingerprints:
  MD5:  3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:79

Alias name: verisignclass3publicprimarycertificationauthorityg3
Certificate fingerprints:
  MD5:  CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:44

Alias name: swisssignsilverg2ca
Certificate fingerprints:
  MD5:  E0:06:A1:C9:7D:CF:C9:FC:0D:C0:56:75:96:D8:62:13
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D5

Alias name: swisssignsilvercag2
Certificate fingerprints:
  MD5:  E0:06:A1:C9:7D:CF:C9:FC:0D:C0:56:75:96:D8:62:13
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D5
```

```
Alias name: atostrustedroot2011
Certificate fingerprints:
  MD5:  AE:B9:C4:32:4B:AC:7F:5D:66:CC:77:94:BB:2A:77:56
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:74

Alias name: comodoecccertificationauthority
Certificate fingerprints:
  MD5:  7C:62:FF:74:9D:31:53:5E:68:4A:D5:78:AA:1E:BF:23
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C7

Alias name: securetrustca
Certificate fingerprints:
  MD5:  DC:32:C3:A7:6D:25:57:C7:68:09:9D:EA:2D:A9:A2:D1
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:73

Alias name: soneraclass1ca
Certificate fingerprints:
  MD5:  33:B7:84:F5:5F:27:D7:68:27:DE:14:DE:12:2A:ED:6F
  SHA1: 07:47:22:01:99:CE:74:B9:7C:B0:3D:79:B2:64:A2:C8:55:E9:33:FF
  SHA256:
 CD:80:82:84:CF:74:6F:F2:FD:6E:B5:8A:A1:D5:9C:4A:D4:B3:CA:56:FD:C6:27:4A:89:26:A7:83:5F:32:31:3D

Alias name: cadisigrootr2
Certificate fingerprints:
  MD5:  26:01:FB:D8:27:A7:17:9A:45:54:38:1A:43:01:3B:03
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:03

Alias name: cadisigrootr1
Certificate fingerprints:
  MD5:  BE:EC:11:93:9A:F5:69:21:BC:D7:C1:C0:67:89:CC:2A
  SHA1: 8E:1C:74:F8:A6:20:B9:E5:8A:F4:61:FA:EC:2B:47:56:51:1A:52:C6
  SHA256:
 F9:6F:23:F4:C3:E7:9C:07:7A:46:98:8D:5A:F5:90:06:76:A0:F0:39:CB:64:5D:D1:75:49:B2:16:C8:24:40:CE

Alias name: verisignclass3g5ca
Certificate fingerprints:
  MD5:  CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:DF

Alias name: utnuserfirsthardwareca
Certificate fingerprints:
  MD5:  4C:56:41:E5:0D:BB:2B:E8:CA:A3:ED:18:08:AD:43:39
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:37

Alias name: addtrustqualifiedca
Certificate fingerprints:
  MD5:  27:EC:39:47:CD:DA:5A:AF:E2:9A:01:65:21:A9:4C:BB
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:16

Alias name: verisignclass3g3ca
Certificate fingerprints:
```

```
  MD5:  CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:44

Alias name: thawtepersonalfreemailca
Certificate fingerprints:
  MD5:  53:4B:1D:17:58:58:1A:30:A1:90:F8:6E:5C:F2:CF:65
  SHA1: E6:18:83:AE:84:CA:C1:C1:CD:52:AD:E8:E9:25:2B:45:A6:4F:B7:E2
  SHA256:
 5B:38:BD:12:9E:83:D5:A0:CA:D2:39:21:08:94:90:D5:0D:4A:AE:37:04:28:F8:DD:FF:FF:FA:4C:15:64:E1:84

Alias name: certplusclass3pprimaryca
Certificate fingerprints:
  MD5:  E1:4B:52:73:D7:1B:DB:93:30:E5:BD:E4:09:6E:BE:FB
  SHA1: 21:6B:2A:29:E6:2A:00:CE:82:01:46:D8:24:41:41:B9:25:11:B2:79
  SHA256:
 CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:85

Alias name: swisssigngoldg2ca
Certificate fingerprints:
  MD5:  24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:95

Alias name: swisssigngoldcag2
Certificate fingerprints:
  MD5:  24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:95

Alias name: dtrustrootclass3ca22009
Certificate fingerprints:
  MD5:  CD:E0:25:69:8D:47:AC:9C:89:35:90:F7:FD:51:3D:2F
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C1

Alias name: acraizfnmtrcm
Certificate fingerprints:
  MD5:  E2:09:04:B4:D3:BD:D1:A0:14:FD:1A:D2:47:C4:57:1D
  SHA1: EC:50:35:07:B2:15:C4:95:62:19:E2:A8:9A:5B:42:99:2C:4C:2C:20
  SHA256:
 EB:C5:57:0C:29:01:8C:4D:67:B1:AA:12:7B:AF:12:F7:03:B4:61:1E:BC:17:B7:DA:B5:57:38:94:17:9B:93:FA

Alias name: securitycommunicationevrootca1
Certificate fingerprints:
  MD5:  22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:37

Alias name: starfieldclass2ca
Certificate fingerprints:
  MD5:  32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:58

Alias name: opentrustrootcag3
Certificate fingerprints:
  MD5:  21:37:B4:17:16:92:7B:67:46:70:A9:96:D7:A8:13:24
  SHA1: 6E:26:64:F3:56:BF:34:55:BF:D1:93:3F:7C:01:DE:D8:13:DA:8A:A6
```

```
  SHA256:
 B7:C3:62:31:70:6E:81:07:8C:36:7C:B8:96:19:8F:1E:32:08:DD:92:69:49:DD:8F:57:09:A4:10:F7:5B:62:92

Alias name: opentrustrootcag2
Certificate fingerprints:
  MD5:  57:24:B6:59:24:6B:AE:C8:FE:1C:0C:20:F2:C0:4E:EB
  SHA1: 79:5F:88:60:C5:AB:7C:3D:92:E6:CB:F4:8D:E1:45:CD:11:EF:60:0B
  SHA256:
 27:99:58:29:FE:6A:75:15:C1:BF:E8:48:F9:C4:76:1D:B1:6C:22:59:29:25:7B:F4:0D:08:94:F2:9E:A8:BA:F2

Alias name: buypassclass2rootca
Certificate fingerprints:
  MD5:  46:A7:D2:FE:45:FB:64:5A:A8:59:90:9B:78:44:9B:29
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:48

Alias name: opentrustrootcag1
Certificate fingerprints:
  MD5:  76:00:CC:81:29:CD:55:5E:88:6A:7A:2E:F7:4D:39:DA
  SHA1: 79:91:E8:34:F7:E2:EE:DD:08:95:01:52:E9:55:2D:14:E9:58:D5:7E
  SHA256:
 56:C7:71:28:D9:8C:18:D9:1B:4C:FD:FF:BC:25:EE:91:03:D4:75:8E:A2:AB:AD:82:6A:90:F3:45:7D:46:0E:B4

Alias name: globalsignr2ca
Certificate fingerprints:
  MD5:  94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9E

Alias name: buypassclass3rootca
Certificate fingerprints:
  MD5:  3D:3B:18:9E:2C:64:5A:E8:D5:88:CE:0E:F9:37:C2:EC
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4D

Alias name: ecacc
Certificate fingerprints:
  MD5:  EB:F5:9D:29:0D:61:F9:42:1F:7C:C2:BA:6D:E3:15:09
  SHA1: 28:90:3A:63:5B:52:80:FA:E6:77:4C:0B:6D:A7:D6:BA:A6:4A:F2:E8
  SHA256:
 88:49:7F:01:60:2F:31:54:24:6A:E2:8C:4D:5A:EF:10:F1:D8:7E:BB:76:62:6F:4A:E0:B7:F9:5B:A7:96:87:99

Alias name: epkirootcertificationauthority
Certificate fingerprints:
  MD5:  1B:2E:00:CA:26:06:90:3D:AD:FE:6F:15:68:D3:6B:B3
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D5

Alias name: verisignclass1g2ca
Certificate fingerprints:
  MD5:  DB:23:3D:F9:69:FA:4B:B9:95:80:44:73:5E:7D:41:83
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7F

Alias name: certigna
Certificate fingerprints:
  MD5:  AB:57:A6:5B:7D:42:82:19:B5:D8:58:26:28:5E:FD:FF
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2D
```

```
Alias name: camerfirmaglobalchambersignroot
Certificate fingerprints:
  MD5:  C5:E6:7B:BF:06:D0:4F:43:ED:C4:7A:65:8A:FB:6B:19
  SHA1: 33:9B:6B:14:50:24:9B:55:7A:01:87:72:84:D9:E0:2F:C3:D2:D8:E9
  SHA256:
 EF:3C:B4:17:FC:8E:BF:6F:97:87:6C:9E:4E:CE:39:DE:1E:A5:FE:64:91:41:D1:02:8B:7D:11:C0:B2:29:8C:ED

Alias name: cfcaevroot
Certificate fingerprints:
  MD5:  74:E1:B6:ED:26:7A:7A:44:30:33:94:AB:7B:27:81:30
  SHA1: E2:B8:29:4B:55:84:AB:6B:58:C2:90:46:6C:AC:3F:B8:39:8F:84:83
  SHA256:
 5C:C3:D7:8E:4E:1D:5E:45:54:7A:04:E6:87:3E:64:F9:0C:F9:53:6D:1C:CC:2E:F8:00:F3:55:C4:C5:FD:70:FD

Alias name: soneraclass2rootca
Certificate fingerprints:
  MD5:  A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:27

Alias name: certumtrustednetworkca
Certificate fingerprints:
  MD5:  D5:E9:81:40:C5:18:69:FC:46:2C:89:75:62:0F:AA:78
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8E

Alias name: securitycommunicationrootca2
Certificate fingerprints:
  MD5:  6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F6

Alias name: globalsigneccrootcar5
Certificate fingerprints:
  MD5:  9F:AD:3B:1C:02:1E:8A:BA:17:74:38:81:0C:A2:BC:08
  SHA1: 1F:24:C6:30:CD:A4:18:EF:20:69:FF:AD:4F:DD:5F:46:3A:1B:69:AA
  SHA256:
 17:9F:BC:14:8A:3D:D0:0F:D2:4E:A1:34:58:CC:43:BF:A7:F5:9C:81:82:D7:83:A5:13:F6:EB:EC:10:0C:89:24

Alias name: globalsigneccrootcar4
Certificate fingerprints:
  MD5:  20:F0:27:68:D1:7E:A0:9D:0E:E6:2A:CA:DF:5C:89:8E
  SHA1: 69:69:56:2E:40:80:F4:24:A1:E7:19:9F:14:BA:F3:EE:58:AB:6A:BB
  SHA256:
 BE:C9:49:11:C2:95:56:76:DB:6C:0A:55:09:86:D7:6E:3B:A0:05:66:7C:44:2C:97:62:B4:FB:B7:73:DE:22:8C

Alias name: chambersofcommerceroot2008
Certificate fingerprints:
  MD5:  5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C0

Alias name: pscprocert
Certificate fingerprints:
  MD5:  E6:24:E9:12:01:AE:0C:DE:8E:85:C4:CE:A3:12:DD:EC
  SHA1: 70:C1:8D:74:B4:28:81:0A:E4:FD:A5:75:D7:01:9F:99:B0:3D:50:74
  SHA256:
 3C:FC:3C:14:D1:F6:84:FF:17:E3:8C:43:CA:44:0C:00:B9:67:EC:93:3E:8B:FE:06:4C:A1:D7:2C:90:F2:AD:B0

Alias name: thawteprimaryrootcag3
Certificate fingerprints:
  MD5:  FB:1B:5D:43:8A:94:CD:44:C6:76:F2:43:4B:47:E7:31
```

```
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4C

Alias name: quovadisrootca
Certificate fingerprints:
  MD5:  27:DE:36:FE:72:B7:00:03:00:9D:F4:F0:1E:6C:04:24
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:73

Alias name: thawteprimaryrootcag2
Certificate fingerprints:
  MD5:  74:9D:EA:60:24:C4:FD:22:53:3E:CC:3A:72:D9:29:4F
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:57

Alias name: deprecateditsecca
Certificate fingerprints:
  MD5:  A5:96:0C:F6:B5:AB:27:E5:01:C6:00:88:9E:60:33:E5
  SHA1: 12:12:0B:03:0E:15:14:54:F4:DD:B3:F5:DE:13:6E:83:5A:29:72:9D
  SHA256:
 9A:59:DA:86:24:1A:FD:BA:A3:39:FA:9C:FD:21:6A:0B:06:69:4D:E3:7E:37:52:6B:BE:63:C8:BC:83:74:2E:CB

Alias name: usertrustrsacertificationauthority
Certificate fingerprints:
  MD5:  1B:FE:69:D1:91:B7:19:33:A3:72:A8:0F:E1:55:E5:B5
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D2

Alias name: entrustrootcag2
Certificate fingerprints:
  MD5:  4B:E2:C9:91:96:65:0C:F4:0E:5A:93:92:A0:0A:FE:B2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:39

Alias name: networksolutionscertificateauthority
Certificate fingerprints:
  MD5:  D3:F3:A6:16:C0:FA:6B:1D:59:B1:2D:96:4D:0E:11:2E
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0C

Alias name: trustcenterclass4caii
Certificate fingerprints:
  MD5:  9D:FB:F9:AC:ED:89:33:22:F4:28:48:83:25:23:5B:E0
  SHA1: A6:9A:91:FD:05:7F:13:6A:42:63:0B:B1:76:0D:2D:51:12:0C:16:50
  SHA256:
 32:66:96:7E:59:CD:68:00:8D:9D:D3:20:81:11:85:C7:04:20:5E:8D:95:FD:D8:4F:1C:7B:31:1E:67:04:FC:32

Alias name: oistewisekeyglobalrootgaca
Certificate fingerprints:
  MD5:  BC:6C:51:33:A7:E9:D3:66:63:54:15:72:1B:21:92:93
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F5

Alias name: verisignuniversalrootcertificationauthority
Certificate fingerprints:
  MD5:  8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3C
```

```
Alias name: ttelesecglobalrootclass3ca
Certificate fingerprints:
  MD5:  CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:BD

Alias name: starfieldservicesrootg2ca
Certificate fingerprints:
  MD5:  17:35:74:AF:7B:61:1C:EB:F4:F9:3C:E2:EE:40:F9:A2
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B5

Alias name: addtrustexternalroot
Certificate fingerprints:
  MD5:  1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F2

Alias name: turktrustelektroniksertifikahizmetsaglayicisih5
Certificate fingerprints:
  MD5:  DA:70:8E:F0:22:DF:93:26:F6:5F:9F:D3:15:06:52:4E
  SHA1: C4:18:F6:4D:46:D1:DF:00:3D:27:30:13:72:43:A9:12:11:C6:75:FB
  SHA256:
 49:35:1B:90:34:44:C1:85:CC:DC:5C:69:3D:24:D8:55:5C:B2:08:D6:A8:14:13:07:69:9F:4A:F0:63:19:9D:78

Alias name: camerfirmachambersca
Certificate fingerprints:
  MD5:  5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C0

Alias name: certsignrootca
Certificate fingerprints:
  MD5:  18:98:C0:D6:E9:3A:FC:F9:B0:F5:0C:F7:4B:01:44:17
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:BB

Alias name: verisignuniversalrootca
Certificate fingerprints:
  MD5:  8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3C

Alias name: geotrustuniversalca
Certificate fingerprints:
  MD5:  92:65:58:8B:A2:1A:31:72:73:68:5C:B4:A5:7A:07:48
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:12

Alias name: luxtrustglobalroot2
Certificate fingerprints:
  MD5:  B2:E1:09:00:61:AF:F7:F1:91:6F:C4:AD:8D:5E:3B:7C
  SHA1: 1E:0E:56:19:0A:D1:8B:25:98:B2:04:44:FF:66:8A:04:17:99:5F:3F
  SHA256:
 54:45:5F:71:29:C2:0B:14:47:C4:18:F9:97:16:8F:24:C5:8F:C5:02:3B:F5:DA:5B:E2:EB:6E:1D:D8:90:2E:D5

Alias name: twcaglobalrootca
Certificate fingerprints:
```

```
  MD5:  F9:03:7E:CF:E6:9E:3C:73:7A:2A:90:07:69:FF:2B:96
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1B

Alias name: tubitakkamusmsslkoksertifikasisurum1
Certificate fingerprints:
  MD5:  DC:00:81:DC:69:2F:3E:2F:B0:3B:F6:3D:5A:91:8E:49
  SHA1: 31:43:64:9B:EC:CE:27:EC:ED:3A:3F:0B:8F:0D:E4:E8:91:DD:EE:CA
  SHA256:
 46:ED:C3:68:90:46:D5:3A:45:3F:B3:10:4A:B8:0D:CA:EC:65:8B:26:60:EA:16:29:DD:7E:86:79:90:64:87:16

Alias name: affirmtrustnetworkingca
Certificate fingerprints:
  MD5:  42:65:CA:BE:01:9A:9A:4C:A9:8C:41:49:CD:C0:D5:7F
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1B

Alias name: affirmtrustcommercialca
Certificate fingerprints:
  MD5:  82:92:BA:5B:EF:CD:8A:6F:A6:3D:55:F9:84:F6:D6:B7
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A7

Alias name: godaddyrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:DA

Alias name: starfieldrootg2ca
Certificate fingerprints:
  MD5:  D6:39:81:C6:52:7E:96:69:FC:FC:CA:66:ED:05:F2:96
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F5

Alias name: dtrustrootclass3ca2ev2009
Certificate fingerprints:
  MD5:  AA:C6:43:2C:5E:2D:CD:C4:34:C0:50:4F:11:02:4F:B6
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:81

Alias name: buypassclass3ca
Certificate fingerprints:
  MD5:  3D:3B:18:9E:2C:64:5A:E8:D5:88:CE:0E:F9:37:C2:EC
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4D

Alias name: verisignclass2g3ca
Certificate fingerprints:
  MD5:  F8:BE:C4:63:22:C9:A8:46:74:8B:B8:1D:1E:4A:2B:F6
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:BB

Alias name: digicerttrustedrootg4
Certificate fingerprints:
  MD5:  78:F2:FC:AA:60:1F:2F:B4:EB:C9:37:BA:53:2E:75:49
  SHA1: DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
```

```
  SHA256:
 55:2F:7B:DC:F1:A7:AF:9E:6C:E6:72:01:7F:4F:12:AB:F7:72:40:C7:8E:76:1A:C2:03:D1:D9:D2:0A:C8:99:88

Alias name: quovadisrootca2g3
Certificate fingerprints:
  MD5:  AF:0C:86:6E:BF:40:2D:7F:0B:3E:12:50:BA:12:3D:06
  SHA1: 09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
  SHA256:
 8F:E4:FB:0A:F9:3A:4D:0D:67:DB:0B:EB:B2:3E:37:C7:1B:F3:25:DC:BC:DD:24:0E:A0:4D:AF:58:B4:7E:18:40

Alias name: geotrustprimarycertificationauthorityg3
Certificate fingerprints:
  MD5:  B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D4

Alias name: geotrustprimarycertificationauthorityg2
Certificate fingerprints:
  MD5:  01:5E:D8:6B:BD:6F:3D:8E:A1:31:F8:12:E0:98:73:6A
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:66

Alias name: godaddyclass2ca
Certificate fingerprints:
  MD5:  91:DE:06:25:AB:DA:FD:32:17:0C:BB:25:17:2A:84:67
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E4

Alias name: trustcoreca1
Certificate fingerprints:
  MD5:  27:92:23:1D:0A:F5:40:7C:E9:E6:6B:9D:D8:F5:E7:6C
  SHA1: 58:D1:DF:95:95:67:6B:63:C0:F0:5B:1C:17:4D:8B:84:0B:C8:78:BD
  SHA256:
 5A:88:5D:B1:9C:01:D9:12:C5:75:93:88:93:8C:AF:BB:DF:03:1A:B2:D4:8E:91:EE:15:58:9B:42:97:1D:03:9C

Alias name: hellenicacademicandresearchinstitutionseccrootca2015
Certificate fingerprints:
  MD5:  81:E5:B4:17:EB:C2:F5:E1:4B:0D:41:7B:49:92:FE:EF
  SHA1: 9F:F1:71:8D:92:D5:9A:F3:7D:74:97:B4:BC:6F:84:68:0B:BA:B6:66
  SHA256:
 44:B5:45:AA:8A:25:E6:5A:73:CA:15:DC:27:FC:36:D2:4C:1C:B9:95:3A:06:65:39:B1:15:82:DC:48:7B:48:33

Alias name: utnuserfirstobjectca
Certificate fingerprints:
  MD5:  A7:F2:E4:16:06:41:11:50:30:6B:9C:E3:B4:9C:B0:C9
  SHA1: E1:2D:FB:4B:41:D7:D9:C3:2B:30:51:4B:AC:1D:81:D8:38:5E:2D:46
  SHA256:
 6F:FF:78:E4:00:A7:0C:11:01:1C:D8:59:77:C4:59:FB:5A:F9:6A:3D:F0:54:08:20:D0:F4:B8:60:78:75:E5:8F

Alias name: ttelesecglobalrootclass3
Certificate fingerprints:
  MD5:  CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:BD

Alias name: ttelesecglobalrootclass2
Certificate fingerprints:
  MD5:  2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:52
```

```
Alias name: addtrustclass1ca
Certificate fingerprints:
  MD5:  1E:42:95:02:33:92:6B:B9:5F:C0:7F:DA:D6:B2:4B:FC
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A7

Alias name: amzninternalrootca
Certificate fingerprints:
  MD5:  08:09:73:AC:E0:78:41:7C:0A:26:33:51:E8:CF:E6:60
  SHA1: A7:B7:F6:15:8A:FF:1E:C8:85:13:38:BC:93:EB:A2:AB:A4:09:EF:06
  SHA256:
 0E:DE:63:C1:DC:7A:8E:11:F1:AB:BC:05:4F:59:EE:49:9D:62:9A:2F:DE:9C:A7:16:32:A2:64:29:3E:8B:66:AA

Alias name: starfieldrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  D6:39:81:C6:52:7E:96:69:FC:FC:CA:66:ED:05:F2:96
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F5

Alias name: camerfirmachambersignca
Certificate fingerprints:
  MD5:  9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:CA

Alias name: secomscrootca2
Certificate fingerprints:
  MD5:  6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F6

Alias name: entrustevca
Certificate fingerprints:
  MD5:  D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4C

Alias name: secomscrootca1
Certificate fingerprints:
  MD5:  F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6C

Alias name: affirmtrustcommercial
Certificate fingerprints:
  MD5:  82:92:BA:5B:EF:CD:8A:6F:A6:3D:55:F9:84:F6:D6:B7
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A7

Alias name: digicertassuredidrootg3
Certificate fingerprints:
  MD5:  7C:7F:65:31:0C:81:DF:8D:BA:3E:99:E2:5C:AD:6E:FB
  SHA1: F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
  SHA256:
 7E:37:CB:8B:4C:47:09:0C:AB:36:55:1B:A6:F4:5D:B8:40:68:0F:BA:16:6A:95:2D:B1:00:71:7F:43:05:3F:C2

Alias name: affirmtrustnetworking
Certificate fingerprints:
  MD5:  42:65:CA:BE:01:9A:9A:4C:A9:8C:41:49:CD:C0:D5:7F
```

```
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1B

Alias name: izenpecom
Certificate fingerprints:
  MD5:  A6:B0:CD:85:80:DA:5C:50:34:A3:39:90:2F:55:67:73
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1F

Alias name: amazon-ca-g4-legacy
Certificate fingerprints:
  MD5:  6C:E5:BD:67:A4:4F:E3:FD:C2:4C:46:E6:06:5B:6D:55
  SHA1: EA:E7:DE:F9:0A:BE:9F:0B:68:CE:B7:24:0D:80:74:03:BF:6E:B1:6E
  SHA256:
 CD:72:C4:7F:B4:AD:28:A4:67:2B:E1:86:47:D4:40:E9:3B:16:2D:95:DB:3C:2F:94:BB:81:D9:09:F7:91:24:5E

Alias name: digicertassuredidrootg2
Certificate fingerprints:
  MD5:  92:38:B9:F8:63:24:82:65:2C:57:33:E6:FE:81:8F:9D
  SHA1: A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
  SHA256:
 7D:05:EB:B6:82:33:9F:8C:94:51:EE:09:4E:EB:FE:FA:79:53:A1:14:ED:B2:F4:49:49:45:2F:AB:7D:2F:C1:85

Alias name: comodoaaaservicesroot
Certificate fingerprints:
  MD5:  49:79:04:B0:EB:87:19:AC:47:B0:BC:11:51:9B:74:D0
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F4

Alias name: entrustnetpremium2048secureserverca
Certificate fingerprints:
  MD5:  EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:77

Alias name: trustcorrootcertca2
Certificate fingerprints:
  MD5:  A2:E1:F8:18:0B:BA:45:D5:C7:41:2A:BB:37:52:45:64
  SHA1: B8:BE:6D:CB:56:F1:55:B9:63:D4:12:CA:4E:06:34:C7:94:B2:1C:C0
  SHA256:
 07:53:E9:40:37:8C:1B:D5:E3:83:6E:39:5D:AE:A5:CB:83:9E:50:46:F1:BD:0E:AE:19:51:CF:10:FE:C7:C9:65

Alias name: entrust2048ca
Certificate fingerprints:
  MD5:  EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:77

Alias name: trustcorrootcertca1
Certificate fingerprints:
  MD5:  6E:85:F1:DC:1A:00:D3:22:D5:B2:B2:AC:6B:37:05:45
  SHA1: FF:BD:CD:E7:82:C8:43:5E:3C:6F:26:86:5C:CA:A8:3A:45:5B:C3:0A
  SHA256:
 D4:0E:9C:86:CD:8F:E4:68:C1:77:69:59:F4:9E:A7:74:FA:54:86:84:B6:C4:06:F3:90:92:61:F4:DC:E2:57:5C

Alias name: baltimorecybertrustroot
Certificate fingerprints:
  MD5:  AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:EB
```

```
Alias name: eecertificationcentrerootca
Certificate fingerprints:
  MD5:  43:5E:88:D4:7D:1A:4A:7E:FD:84:2E:52:EB:01:D4:6F
  SHA1: C9:A8:B9:E7:55:80:5E:58:E3:53:77:A7:25:EB:AF:C3:7B:27:CC:D7
  SHA256:
 3E:84:BA:43:42:90:85:16:E7:75:73:C0:99:2F:09:79:CA:08:4E:46:85:68:1F:F1:95:CC:BA:8A:22:9B:8A:76

Alias name: dstacescax6
Certificate fingerprints:
  MD5:  21:D8:4C:82:2B:99:09:33:A2:EB:14:24:8D:8E:5F:E8
  SHA1: 40:54:DA:6F:1C:3F:40:74:AC:ED:0F:EC:CD:DB:79:D1:53:FB:90:1D
  SHA256:
 76:7C:95:5A:76:41:2C:89:AF:68:8E:90:A1:C7:0F:55:6C:FD:6B:60:25:DB:EA:10:41:6D:7E:B6:83:1F:8C:40

Alias name: comodocertificationauthority
Certificate fingerprints:
  MD5:  5C:48:DC:F7:42:72:EC:56:94:6D:1C:CC:71:35:80:75
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:66

Alias name: thawteserverca
Certificate fingerprints:
  MD5:  EE:FE:61:69:65:6E:F8:9C:C6:2A:F4:D7:2B:63:EF:A2
  SHA1: 9F:AD:91:A6:CE:6A:C6:C5:00:47:C4:4E:C9:D4:A5:0D:92:D8:49:79
  SHA256:
 87:C6:78:BF:B8:B2:5F:38:F7:E9:7B:33:69:56:BB:CF:14:4B:BA:CA:A5:36:47:E6:1A:23:25:BC:10:55:31:6B

Alias name: secomvalicertclass1ca
Certificate fingerprints:
  MD5:  65:58:AB:15:AD:57:6C:1E:A8:A7:B5:69:AC:BF:FF:EB
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:04

Alias name: godaddyrootg2ca
Certificate fingerprints:
  MD5:  80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:DA

Alias name: globalchambersignroot2008
Certificate fingerprints:
  MD5:  9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:CA

Alias name: equifaxsecureebusinessca1
Certificate fingerprints:
  MD5:  14:C0:08:E5:A3:85:03:A3:BE:78:E9:67:4F:27:CA:EE
  SHA1: AE:E6:3D:70:E3:76:FB:C7:3A:EB:B0:A1:C1:D4:C4:7A:A7:40:B3:F4
  SHA256:
 2E:3A:2B:B5:11:25:05:83:6C:A8:96:8B:E2:CB:37:27:CE:9B:56:84:5C:6E:E9:8E:91:85:10:4A:FB:9A:F5:96

Alias name: quovadisrootca3
Certificate fingerprints:
  MD5:  31:85:3C:62:94:97:63:B9:AA:FD:89:4E:AF:6F:E0:CF
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:35

Alias name: usertrustecccertificationauthority
Certificate fingerprints:
```

```
  MD5:  FA:68:BC:D9:B5:7F:AD:FD:C9:1D:06:83:28:CC:24:C1
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7A

Alias name: quovadisrootca2
Certificate fingerprints:
  MD5:  5E:39:7B:DD:F8:BA:EC:82:E9:AC:62:BA:0C:54:00:2B
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:86

Alias name: soneraclass2ca
Certificate fingerprints:
  MD5:  A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:27

Alias name: twcarootcertificationauthority
Certificate fingerprints:
  MD5:  AA:08:8F:F6:F9:7B:B7:F2:B1:A7:1E:9B:EA:EA:BD:79
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:44

Alias name: baltimorecybertrustca
Certificate fingerprints:
  MD5:  AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:EB

Alias name: cia-crt-g3-01-ca
Certificate fingerprints:
  MD5:  E3:66:DD:D6:A0:D5:40:8F:FF:29:E2:C0:CB:6E:62:1A
  SHA1: 2B:EE:2C:BA:A3:1D:B5:FE:60:40:41:95:08:ED:46:82:39:4D:ED:E2
  SHA256:
 20:48:AD:4C:EC:90:7F:FA:4A:15:D4:CE:45:E3:C8:E4:2C:EA:78:33:DC:C7:D3:40:48:FC:60:47:27:42:99:EC

Alias name: entrustrootcertificationauthorityg2
Certificate fingerprints:
  MD5:  4B:E2:C9:91:96:65:0C:F4:0E:5A:93:92:A0:0A:FE:B2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:39

Alias name: verisignclass3g4ca
Certificate fingerprints:
  MD5:  3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:79

Alias name: xrampglobalcaroot
Certificate fingerprints:
  MD5:  A1:0B:44:B3:CA:10:D8:00:6E:9D:0F:D8:0F:92:0A:D1
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A2

Alias name: identrustcommercialrootca1
Certificate fingerprints:
  MD5:  B3:3E:77:73:75:EE:A0:D3:E3:7E:49:63:49:59:BB:C7
  SHA1: DF:71:7E:AA:4A:D9:4E:C9:55:84:99:60:2D:48:DE:5F:BC:F0:3A:25
```

```
  SHA256:
 5D:56:49:9B:E4:D2:E0:8B:CF:CA:D0:8A:3E:38:72:3D:50:50:3B:DE:70:69:48:E4:2F:55:60:30:19:E5:28:AE

Alias name: camerfirmachamberscommerceca
Certificate fingerprints:
  MD5:  B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C3

Alias name: verisignclass3g2ca
Certificate fingerprints:
  MD5:  A2:33:9B:4C:74:78:73:D4:6C:E7:C1:F3:8D:CB:5C:E9
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8B

Alias name: deutschetelekomrootca2
Certificate fingerprints:
  MD5:  74:01:4A:91:B1:08:C4:58:CE:47:CD:F0:DD:11:53:08
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D3

Alias name: certumca
Certificate fingerprints:
  MD5:  2C:8F:9F:66:1D:18:90:B1:47:26:9D:8E:86:82:8C:A9
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:24

Alias name: cybertrustglobalroot
Certificate fingerprints:
  MD5:  72:E4:4A:87:E3:69:40:80:77:EA:BC:E3:F4:FF:F0:E1
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A3

Alias name: globalsignrootca
Certificate fingerprints:
  MD5:  3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:99

Alias name: secomevrootca1
Certificate fingerprints:
  MD5:  22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:37

Alias name: globalsignr3ca
Certificate fingerprints:
  MD5:  C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3B

Alias name: staatdernederlandenrootcag3
Certificate fingerprints:
  MD5:  0B:46:67:07:DB:10:2F:19:8C:35:50:60:D1:0B:F4:37
  SHA1: D8:EB:6B:41:51:92:59:E0:F3:E7:85:00:C0:3D:B6:88:97:C9:EE:FC
  SHA256:
 3C:4F:B0:B9:5A:B8:B3:00:32:F4:32:B8:6F:53:5F:E1:72:C1:85:D0:FD:39:86:58:37:CF:36:18:7F:A6:F4:28
```

```
Alias name: staatdernederlandenrootcag2
Certificate fingerprints:
  MD5:  7C:A5:0F:F8:5B:9A:7D:6D:30:AE:54:5A:E3:42:A2:8A
  SHA1: 59:AF:82:79:91:86:C7:B4:75:07:CB:CF:03:57:46:EB:04:DD:B7:16
  SHA256:
 66:8C:83:94:7D:A6:3B:72:4B:EC:E1:74:3C:31:A0:E6:AE:D0:DB:8E:C5:B3:1B:E3:77:BB:78:4F:91:B6:71:6F

Alias name: aolrootca2
Certificate fingerprints:
  MD5:  D6:ED:3C:CA:E2:66:0F:AF:10:43:0D:77:9B:04:09:BF
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:BD

Alias name: dstrootcax3
Certificate fingerprints:
  MD5:  41:03:52:DC:0F:F7:50:1B:16:F0:02:8E:BA:6F:45:C5
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:39

Alias name: trustcenteruniversalcai
Certificate fingerprints:
  MD5:  45:E1:A5:72:C5:A9:36:64:40:9E:F5:E4:58:84:67:8C
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E7

Alias name: aolrootca1
Certificate fingerprints:
  MD5:  14:F1:08:AD:9D:FA:64:E2:89:E7:1C:CF:A8:AD:7D:5E
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E3

Alias name: affirmtrustpremiumecc
Certificate fingerprints:
  MD5:  64:B0:09:55:CF:B1:D5:99:E2:BE:13:AB:A6:5D:EA:4D
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:23

Alias name: microseceszignorootca2009
Certificate fingerprints:
  MD5:  F8:49:F4:03:BC:44:2D:83:BE:48:69:7D:29:64:FC:B1
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:78

Alias name: verisignclass1g3ca
Certificate fingerprints:
  MD5:  B1:47:BC:18:57:D1:18:A0:78:2D:EC:71:E8:2A:95:73
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:65

Alias name: certplusrootcag2
Certificate fingerprints:
  MD5:  A7:EE:C4:78:2D:1B:EE:2D:B9:29:CE:D6:A7:96:32:31
  SHA1: 4F:65:8E:1F:E9:06:D8:28:02:E9:54:47:41:C9:54:25:5D:69:CC:1A
  SHA256:
 6C:C0:50:41:E6:44:5E:74:69:6C:4C:FB:C9:F8:0F:54:3B:7E:AB:BB:44:B4:CE:6F:78:7C:6A:99:71:C4:2F:17

Alias name: certplusrootcag1
Certificate fingerprints:
  MD5:  7F:09:9C:F7:D9:B9:5C:69:69:56:D5:37:3E:14:0D:42
```

```
  SHA1: 22:FD:D0:B7:FD:A2:4E:0D:AC:49:2C:A0:AC:A6:7B:6A:1F:E3:F7:66
  SHA256:
 15:2A:40:2B:FC:DF:2C:D5:48:05:4D:22:75:B3:9C:7F:CA:3E:C0:97:80:78:B0:F0:EA:76:E5:61:A6:C7:43:3E

Alias name: addtrustexternalca
Certificate fingerprints:
  MD5:  1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F2

Alias name: entrustrootcertificationauthority
Certificate fingerprints:
  MD5:  D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4C

Alias name: verisignclass3ca
Certificate fingerprints:
  MD5:  EF:5A:F1:33:EF:F1:CD:BB:51:02:EE:12:14:4B:96:C4
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:05

Alias name: digicertassuredidrootca
Certificate fingerprints:
  MD5:  87:CE:0B:7B:2A:0E:49:00:E1:58:71:9B:37:A8:93:72
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5C

Alias name: globalsignrootcar3
Certificate fingerprints:
  MD5:  C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3B

Alias name: globalsignrootcar2
Certificate fingerprints:
  MD5:  94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9E

Alias name: verisignclass1ca
Certificate fingerprints:
  MD5:  86:AC:DE:2B:C5:6D:C3:D9:8C:28:88:D3:8D:16:13:1E
  SHA1: CE:6A:64:A3:09:E4:2F:BB:D9:85:1C:45:3E:64:09:EA:E8:7D:60:F1
  SHA256:
 51:84:7C:8C:BD:2E:9A:72:C9:1E:29:2D:2A:E2:47:D7:DE:1E:3F:D2:70:54:7A:20:EF:7D:61:0F:38:B8:84:2C

Alias name: thawtepremiumserverca
Certificate fingerprints:
  MD5:  A6:6B:60:90:23:9B:3F:2D:BB:98:6F:D6:A7:19:0D:46
  SHA1: E0:AB:05:94:20:72:54:93:05:60:62:02:36:70:F7:CD:2E:FC:66:66
  SHA256:
 3F:9F:27:D5:83:20:4B:9E:09:C8:A3:D2:06:6C:4B:57:D3:A2:47:9C:36:93:65:08:80:50:56:98:10:5D:BC:E9

Alias name: verisigntsaca
Certificate fingerprints:
  MD5:  F2:89:95:6E:4D:05:F0:F1:A7:21:55:7D:46:11:BA:47
  SHA1: 20:CE:B1:F0:F5:1C:0E:19:A9:F3:8D:B1:AA:8E:03:8C:AA:7A:C7:01
  SHA256:
 CB:6B:05:D9:E8:E5:7C:D8:82:B1:0B:4D:B7:0D:E4:BB:1D:E4:2B:A4:8A:7B:D0:31:8B:63:5B:F6:E7:78:1A:9D
```

```
Alias name: thawteprimaryrootca
Certificate fingerprints:
  MD5:  8C:CA:DC:0B:22:CE:F5:BE:72:AC:41:1A:11:A8:D8:12
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9F

Alias name: visaecommerceroot
Certificate fingerprints:
  MD5:  FC:11:B8:D8:08:93:30:00:6D:23:F9:7E:EB:52:1E:02
  SHA1: 70:17:9B:86:8C:00:A4:FA:60:91:52:22:3F:9F:3E:32:BD:E0:05:62
  SHA256:
 69:FA:C9:BD:55:FB:0A:C7:8D:53:BB:EE:5C:F1:D5:97:98:9F:D0:AA:AB:20:A2:51:51:BD:F1:73:3E:E7:D1:22

Alias name: digicertglobalrootg3
Certificate fingerprints:
  MD5:  F5:5D:A4:50:A5:FB:28:7E:1E:0F:0D:CC:96:57:56:CA
  SHA1: 7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
  SHA256:
 31:AD:66:48:F8:10:41:38:C7:38:F3:9E:A4:32:01:33:39:3E:3A:18:CC:02:29:6E:F9:7C:2A:C9:EF:67:31:D0

Alias name: xrampglobalca
Certificate fingerprints:
  MD5:  A1:0B:44:B3:CA:10:D8:00:6E:9D:0F:D8:0F:92:0A:D1
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A2

Alias name: digicertglobalrootg2
Certificate fingerprints:
  MD5:  E4:A6:8A:C8:54:AC:52:42:46:0A:FD:72:48:1B:2A:44
  SHA1: DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
  SHA256:
 CB:3C:CB:B7:60:31:E5:E0:13:8F:8D:D3:9A:23:F9:DE:47:FF:C3:5E:43:C1:14:4C:EA:27:D4:6A:5A:B1:CB:5F

Alias name: valicertclass2ca
Certificate fingerprints:
  MD5:  A9:23:75:9B:BA:49:36:6E:31:C2:DB:F2:E7:66:BA:87
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6B

Alias name: geotrustprimaryca
Certificate fingerprints:
  MD5:  02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6C

Alias name: netlockaranyclassgoldfotanusitvany
Certificate fingerprints:
  MD5:  C5:A1:B7:FF:73:DD:D6:D7:34:32:18:DF:FC:3C:AD:88
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:98

Alias name: geotrustglobalca
Certificate fingerprints:
  MD5:  F7:75:AB:29:FB:51:4E:B7:77:5E:FF:05:3C:99:8E:F5
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3A

Alias name: oistewisekeyglobalrootgbca
Certificate fingerprints:
```

```
  MD5:  A4:EB:B9:61:28:2E:B7:2F:98:B0:35:26:90:99:51:1D
  SHA1: 0F:F9:40:76:18:D3:D7:6A:4B:98:F0:A8:35:9E:0C:FD:27:AC:CC:ED
  SHA256:
 6B:9C:08:E8:6E:B0:F7:67:CF:AD:65:CD:98:B6:21:49:E5:49:4A:67:F5:84:5E:7B:D1:ED:01:9F:27:B8:6B:D6

Alias name: certumtrustednetworkca2
Certificate fingerprints:
  MD5:  6D:46:9E:D9:25:6D:08:23:5B:5E:74:7D:1E:27:DB:F2
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:04

Alias name: starfieldservicesrootcertificateauthorityg2
Certificate fingerprints:
  MD5:  17:35:74:AF:7B:61:1C:EB:F4:F9:3C:E2:EE:40:F9:A2
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B5

Alias name: comodorsacertificationauthority
Certificate fingerprints:
  MD5:  1B:31:B0:71:40:36:CC:14:36:91:AD:C4:3E:FD:EC:18
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:34

Alias name: comodoaaaca
Certificate fingerprints:
  MD5:  49:79:04:B0:EB:87:19:AC:47:B0:BC:11:51:9B:74:D0
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F4

Alias name: identrustpublicsectorrootca1
Certificate fingerprints:
  MD5:  37:06:A5:B0:FC:89:9D:BA:F4:6B:8C:1A:64:CD:D5:BA
  SHA1: BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
  SHA256:
 30:D0:89:5A:9A:44:8A:26:20:91:63:55:22:D1:F5:20:10:B5:86:7A:CA:E1:2C:78:EF:95:8F:D4:F4:38:9F:2F

Alias name: certplusclass2primaryca
Certificate fingerprints:
  MD5:  88:2C:8C:52:B8:A2:3C:F3:F7:BB:03:EA:AE:AC:42:0B
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:CB

Alias name: ttelesecglobalrootclass2ca
Certificate fingerprints:
  MD5:  2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:52

Alias name: accvraiz1
Certificate fingerprints:
  MD5:  D0:A0:5A:EE:05:B6:09:94:21:A1:7D:F1:B2:29:82:02
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:13

Alias name: digicerthighassuranceevrootca
Certificate fingerprints:
  MD5:  D4:74:DE:57:5C:39:B2:D3:9C:85:83:C5:C0:65:49:8A
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
```

```
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:CF

Alias name: amzninternalinfoseccag3
Certificate fingerprints:
  MD5:  E9:34:94:02:BA:BB:31:6B:22:E6:2B:A9:C4:F0:26:04
  SHA1: B9:B1:CA:38:F7:BF:9C:D2:D4:95:E7:B6:5E:75:32:9B:A8:78:2E:F6
  SHA256:
 81:03:0B:C7:E2:54:DA:7B:F8:B7:45:DB:DD:41:15:89:B5:A3:81:86:FB:4B:29:77:1F:84:0A:18:D9:67:6D:68

Alias name: cia-crt-g3-02-ca
Certificate fingerprints:
  MD5:  FD:B9:23:FD:D3:EB:2D:3E:57:EF:56:FF:DB:D3:E4:B9
  SHA1: 96:4A:BB:A7:BD:DA:FC:97:34:C0:0A:2D:F0:05:98:F7:E6:C6:6F:09
  SHA256:
 93:F1:72:FB:BA:43:31:5C:06:EE:0F:9F:04:89:B8:F6:88:BC:75:15:3C:BE:B4:80:AC:A7:14:3A:F6:FC:4A:C1

Alias name: entrustrootcertificationauthorityec1
Certificate fingerprints:
  MD5:  B6:7E:1D:F0:58:C5:49:6C:24:3B:3D:ED:98:18:ED:BC
  SHA1: 20:D8:06:40:DF:9B:25:F5:12:25:3A:11:EA:F7:59:8A:EB:14:B5:47
  SHA256:
 02:ED:0E:B2:8C:14:DA:45:16:5C:56:67:91:70:0D:64:51:D7:FB:56:F0:B2:AB:1D:3B:8E:B0:70:E5:6E:DF:F5

Alias name: securitycommunicationrootca
Certificate fingerprints:
  MD5:  F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6C

Alias name: globalsignca
Certificate fingerprints:
  MD5:  3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:99

Alias name: trustcenterclass2caii
Certificate fingerprints:
  MD5:  CE:78:33:5C:59:78:01:6E:18:EA:B9:36:A0:B9:2E:23
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B4

Alias name: camerfirmachambersofcommerceroot
Certificate fingerprints:
  MD5:  B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C3

Alias name: geotrustprimarycag3
Certificate fingerprints:
  MD5:  B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D4

Alias name: geotrustprimarycag2
Certificate fingerprints:
  MD5:  01:5E:D8:6B:BD:6F:3D:8E:A1:31:F8:12:E0:98:73:6A
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:66
```

```
Alias name: hongkongpostrootca1
Certificate fingerprints:
  MD5:  A8:0D:6F:39:78:B9:43:6D:77:42:6D:98:5A:CC:23:CA
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B2

Alias name: affirmtrustpremiumeccca
Certificate fingerprints:
  MD5:  64:B0:09:55:CF:B1:D5:99:E2:BE:13:AB:A6:5D:EA:4D
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:23

Alias name: hellenicacademicandresearchinstitutionsrootca2015
Certificate fingerprints:
  MD5:  CA:FF:E2:DB:03:D9:CB:4B:E9:0F:AD:84:FD:7B:18:CE
  SHA1: 01:0C:06:95:A6:98:19:14:FF:BF:5F:C6:B0:B6:95:EA:29:E9:12:A6
  SHA256:
 A0:40:92:9A:02:CE:53:B4:AC:F4:F2:FF:C6:98:1C:E4:49:6F:75:5E:6D:45:FE:0B:2A:69:2B:CD:52:52:3F:36
```

IoT Analytics

IoT Analytics (`iotAnalytics`) 操作可将 MQTT 消息中的数据发送至 Amazon IoT Analytics 通道。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `iotanalytics:BatchPutMessage` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。

  附加到您指定角色的策略应如下所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotanalytics:BatchPutMessage",
            "Resource": [
                "arn:aws:iotanalytics:us-west-2:account-id:channel/mychannel"
            ]
        }
    ]
}
```

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`batchMode`

（可选）是否批处理操作。默认值为 `false`。

如果 batchMode 为 true 且规则 SQL 语句的求值为一个 Array 时，每个 Array 元素在由 BatchPutMessage 传递到 Amazon IoT Analytics 通道时，都将作为单独的消息进行传递。生成的数组，其消息不得超过 100 条。

支持替换模板 (p. 585)：否

channelName

数据写入的 Amazon IoT Analytics 通道的名称。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

roleArn

允许访问 Amazon IoT Analytics 通道的 IAM 角色。有关更多信息，请参阅要求 (p. 493)：

支持替换模板 (p. 585)：否

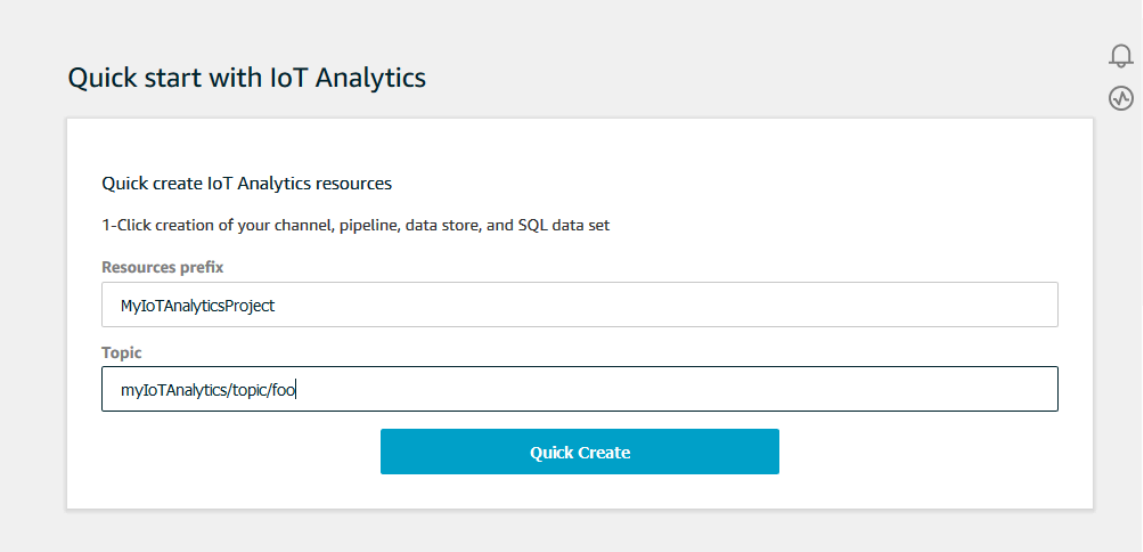
示例

以下 JSON 示例定义了 Amazon IoT 规则中的 IoT Analytics 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotAnalytics": {
                    "channelName": "mychannel",
                    "roleArn": "arn:aws:iam::123456789012:role/analyticsRole",
                }
            }
        ]
    }
}
```

另请参阅

- Amazon IoT Analytics 用户指南 中的什么是 Amazon IoT Analytics？
- Amazon IoT Analytics 控制台还具有快速入门功能，让您可以通过一次单击创建通道、数据存储和管道。有关更多信息，请参阅 Amazon IoT Analytics用户指南中的 Amazon IoT Analytics 控制台快速入门指南。



IoT Events

IoT Events (`iotEvents`) 操作可将触发规则的 MQTT 消息中的数据发送至 Amazon IoT Events 输入。

> **Important**
>
> 如果将有效负载发送到 Amazon IoT Core 时未使用 `Input attribute Key`，或者密钥不在密钥中指定的 JSON 路径中，则会导致 IoT 规则失败并显示错误 `Failed to send message to Iot Events`。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `iotevents:BatchPutMessage` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`batchMode`

（可选）是否批处理事件操作。默认值为 `false`。

如果 `batchMode` 为 `true` 且规则 SQL 语句的求值为一个 Array 时，通过调用 `BatchPutMessage` 将 Array 元素发送到 Amazon IoT Events 时，会将每个 Array 元素都视为单独的消息。生成的数组，其消息不得超过 10 条。

`batchMode` 为 `true` 时，不能指定 `messageId`。

支持替换模板 (p. 585)：否

`inputName`

Amazon IoT Events 输入的名称。

支持[替换模板 (p. 585)]：仅限 API 和 Amazon CLI

`messageId`

（可选）使用它可确保 Amazon IoT Events 探测器只处理一个具有给定 `messageId` 的输入（消息）。您可以使用 `${newuuid()}` 替代模板，以便为每个请求生成一个唯一的 ID。

当 `batchMode` 为 `true` 时，您无法指定 `messageId` -- 将分配一个新 UUID 值。

支持[替换模板 (p. 585)]：是

`roleArn`

允许 Amazon IoT 将输入发送到 Amazon IoT Events 探测器的 IAM 角色。有关更多信息，请参阅[要求 (p. 495)]：

支持[替换模板 (p. 585)]：否

示例

下面的 JSON 示例介绍了如何在 Amazon IoT 规则中定义 IoT Events 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotEvents": {
                    "inputName": "MyIoTEventsInput",
                    "messageId": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_events"
                }
            }
        ]
    }
}
```

另请参阅

- Amazon IoT Events 开发人员指南中的[什么是 Amazon IoT Events？]

IoT SiteWise

IoT SiteWise (`iotSiteWise`) 操作可将 MQTT 消息中的数据发送到中的资产属性Amazon IoT SiteWise。

您可以按照教程执行操作，该教程向您说明如何从 Amazon IoT 事物中提取数据。有关更多信息，请参阅 Amazon IoT SiteWise 用户指南 中的[将数据从 Amazon IoT 事物采集到 Amazon IoT SiteWise] 教程或[使用 Amazon IoT Core 规则采集数据]部分。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `iotsitewise:BatchPutAssetPropertyValue` 操作的 IAM 角色。有关更多信息，请参阅[向 Amazon IoT 规则授予所需的访问权限 (p. 443)]：

您可以将下列示例信任策略附加到要代入的角色。

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

为了提高安全性，您可以在 Condition 属性中指定 Amazon IoT SiteWise 资产层次结构路径。以下示例是指定资产层次结构路径的信任策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "iotsitewise:assetHierarchyPath": [
                        "/root node asset ID",
                        "/root node asset ID/*"
                    ]
                }
            }
        }
    ]
}
```

- 在使用此操作将数据发送到 Amazon IoT SiteWise 时，数据必须满足 BatchPutAssetPropertyValue 操作的要求。有关更多信息，请参阅《Amazon IoT SiteWise API 参考》中的 BatchPutAssetPropertyValue。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`putAssetPropertyValueEntries`

包含以下信息的资产属性值条目的列表：

`propertyAlias`

（可选）与您的资产属性关联的属性别名。您必须指定 propertyAlias 或同时指定 assetId 和 propertyId。有关更多信息，请参阅 Amazon IoT SiteWise 用户指南中的将工业数据流映射到资产属性。

支持替换模板 (p. 585)：是

`assetId`

（可选）Amazon IoT SiteWise 资产的 ID。您必须指定 propertyAlias 或同时指定 assetId 和 propertyId。

支持替换模板 (p. 585)：是

`propertyId`

（可选）资产属性的 ID。您必须指定 `propertyAlias` 或同时指定 `assetId` 和 `propertyId`。

支持替换模板 (p. 585)：是

`entryId`

（可选）此条目的唯一标识符。您可以定义 `entryId`，以便在出现故障时更好地跟踪哪条消息导致错误。默认为新 UUID。

支持替换模板 (p. 585)：是

`propertyValues`

要插入的属性值的列表，均包含以下格式的时间戳、质量和值 (TQV)：

`timestamp`

包含以下信息的时间戳结构：

`timeInSeconds`

包含用 Unix 纪元时间表示的时间（以秒为单位）的字符串。如果您的消息负载没有时间戳，则可使用 timestamp() (p. 577)，它将返回当前时间（以毫秒为单位）。要将该时间转换为秒，可以使用以下替换模板：**`${floor(timestamp() / 1E3)}`**。

支持替换模板 (p. 585)：是

`offsetInNanos`

（可选）包含与以秒为单位的时间的纳秒时间偏移量的字符串。如果您的消息负载没有时间戳，则可使用 timestamp() (p. 577)，它将返回当前时间（以毫秒为单位）。要计算与该时间的纳秒偏移量，可以使用以下替换模板：**`${(timestamp() % 1E3) * 1E6}`**。

支持替换模板 (p. 585)：是

对于 Unix 纪元时间，Amazon IoT SiteWise 仅接受时间戳在过去 7 天内和将来 5 分钟内的条目。

`quality`

（可选）描述值的质量的字符串。有效值：`GOOD`、`BAD`、`UNCERTAIN`。

支持替换模板 (p. 585)：是

`value`

包含以下值字段之一的值结构，具体取决于资产属性的数据类型：

`booleanValue`

（可选）包含值条目布尔值的字符串。

支持替换模板 (p. 585)：是

`doubleValue`

（可选）包含值条目的双值的字符串。

支持替换模板 (p. 585)：是

`integerValue`

（可选）包含值条目整数值的字符串。

支持替换模板 (p. 585)：是

`stringValue`

（可选）值条目的字符串值。

支持替换模板 (p. 585)：是

`roleArn`

授予 Amazon IoT 权限以将资产属性值发送到 Amazon IoT SiteWise 的 IAM 角色的 ARN。有关更多信息，请参阅要求 (p. 496)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例定义了Amazon IoT规则中的基础 IoT SiteWise 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "/some/property/alias",
                            "propertyValues": [
                                {
                                    "timestamp": {
                                        "timeInSeconds": "${my.payload.timeInSeconds}"
                                    },
                                    "value": {
                                        "integerValue": "${my.payload.value}"
                                    }
                                }
                            ]
                        }
                    ],
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
                }
            }
        ]
    }
}
```

下面的 JSON 示例定义了Amazon IoT规则 SiteWise 中的 IoT 操作。此示例将主题用作属性别名和 `timestamp()` 函数。例如，如果您将数据发布到 `/company/windfarm/3/turbine/7/rpm`，则此操作会将数据发送到具有与您指定的主题相同的属性别名的资产属性。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM '/company/windfarm/+/turbine/+/+'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "iotSiteWise": {
                    "putAssetPropertyValueEntries": [
                        {
                            "propertyAlias": "${topic()}",
```

```
                        "propertyValues": [
                            {
                                "timestamp": {
                                    "timeInSeconds": "${floor(timestamp() / 1E3)}",
                                    "offsetInNanos": "${(timestamp() % 1E3) * 1E6}"
                                },
                                "value": {
                                    "doubleValue": "${my.payload.value}"
                                }
                            }
                        ]
                    }
                ],
                "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sitewise"
            }
        }
    ]
}
}
```

另请参阅

- Amazon IoT SiteWise 用户指南 中的什么是 Amazon IoT SiteWise？
- Amazon IoT SiteWise 用户指南中的使用 Amazon IoT Core 规则摄取数据
- Amazon IoT SiteWise 用户指南中的将数据从 Amazon IoT 摄取到 Amazon IoT SiteWise
- Amazon IoT SiteWise 用户指南中的为 Amazon IoT SiteWise 规则操作进行故障排除

Kinesis Data Firehose

Kinesis Data Firehose (`firehose`) 操作可将 MQTT 消息中的数据发送至 Amazon Kinesis Data Firehose 流。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `firehose:PutRecord` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用 Kinesis Data Firehose 将数据发送到 Amazon S3 存储桶，并且您使用 Amazon Key Management Service (Amazon KMS) 客户托管的 Amazon KMS key （KMS 密钥）对 Amazon S3 中的数据进行静态加密，Kinesis Data Firehose 必须具有对存储桶的访问权限，并且有权代表调用方使用 Amazon KMS key。有关更多信息，请参阅 Amazon Kinesis Data Firehose 开发人员指南中的向 Kinesis Data Firehose 授予对 Amazon S3 目标的访问权限。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`batchMode`

（可选）通过使用 `PutRecordBatch` 批量交付 Kinesis Data Firehose 流。默认值为 `false`。

当 `batchMode` 为 `true` 且规则的 SQL 语句求值为一个数组时，每个数组元素在 `PutRecordBatch` 请求中形成一条记录。生成的数组，其记录不得超过 500 条。

支持替换模板 (p. 585)：否

`deliveryStreamName`

消息数据写入的 Kinesis Data Firehose 流。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`separator`

将用于分隔写入 Kinesis Data Firehose 流的记录的分隔符。如果省略此参数，则流不使用分隔符。有效值：`,`（逗号），`\t`（选项卡），`\n`（换行符），`\r\n`（窗口换行符）。

支持替换模板 (p. 585)：否

`roleArn`

授予对 Kinesis Data Firehose 流的访问权限的 IAM 角色。有关更多信息，请参阅要求 (p. 500)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例定义了 Amazon IoT 规则中的 Kinesis Data Firehose 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "firehose": {
                    "deliveryStreamName": "my_firehose_stream",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

以下 JSON 示例定义了 Amazon IoT 规则中一个带有替代模板的 Kinesis Data Firehose 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "firehose": {
                    "deliveryStreamName": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_firehose"
                }
            }
        ]
    }
}
```

另请参阅

- Amazon Kinesis Data Firehose 开发人员指南中的什么是 Amazon Kinesis Data Firehose？

Kinesis Data Streams

Kinesis Data Streams (`kinesis`) 操作将 MQTT 消息中的数据写入 Amazon Kinesis Data Streams。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `kinesis:PutRecord` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用 Amazon Key Management Service (Amazon KMS) 客户托管 Amazon KMS key （KMS 密钥）对 Kinesis Data Streams 中的数据进行静态加密，服务必须具有代表调用方使用 Amazon KMS key 的权限。有关更多信息，请参阅 Amazon Kinesis Data Streams 开发人员指南中的使用用户生成的 Amazon KMS keys 的权限

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`stream`

数据写入的 Kinesis 数据流。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`partitionKey`

用于确定将数据写入哪个分区的分区键。分区键通常由表达式（例如，`${topic()}` 或 `${timestamp()}`）组成。

支持替换模板 (p. 585)：是

`roleArn`

授予对 Kinesis 数据流的 Amazon IoT 访问权限的 IAM 角色 ARN。有关更多信息，请参阅要求 (p. 502)：

支持替换模板 (p. 585)：否

示例

以下 JSON 示例定义了 Amazon IoT 规则中的 Kinesis Data Streams 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "my_kinesis_stream",
                    "partitionKey": "${topic()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
```

```
    }
}
```

以下 JSON 示例定义了 Amazon IoT 规则中的一个带有替代模板的 Kinesis 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "kinesis": {
                    "streamName": "${topic()}",
                    "partitionKey": "${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_kinesis"
                }
            }
        ]
    }
}
```

另请参阅

- Amazon Kinesis Data Streams 开发人员指南中的什么是 Amazon Kinesis Data Streams？

Lambda

Lambda (`lambda`) 操作调用 Amazon Lambda 函数，传入 MQTT 消息。Amazon IoT 将异步调用 Lambda 函数。

您可以按照教程执行操作，该教程向您说明如何使用 Lambda 操作创建并测试规则。有关更多信息，请参阅教程：使用 Amazon Lambda函数格式化通知 (p. 194)：

要求

此规则操作具有以下要求：

- 如需使用 Amazon IoT 调用 Lambda 函数，则您必须配置一个策略，以便向 Amazon IoT 授予 `lambda:InvokeFunction`的权限。您只能调用在与您的 Lambda 策略所在同一 Amazon Web Services 区域 中定义的 Lambda 函数。Lambda 函数使用基于资源的策略，因此您必须将该策略附加到 Lambda 函数本身。

  使用以下 Amazon CLI 命令附加授予 `lambda:InvokeFunction` 权限的策略：

  ```
  aws lambda add-permission --function-name function_name --region region --principal
   iot.amazonaws.com --source-arn arn:aws:iot:region:account-id:rule/rule_name --source-
  account account-id --statement-id unique_id --action "lambda:InvokeFunction"
  ```

  `add-permission` 命令需要以下参数：

  `--function-name`

  Lambda 函数的名称 添加新的权限来更新函数的资源策略。

  `--region`

  函数的Amazon Web Services 区域。

`--principal`

获取权限的委托人。这应该是 `iot.amazonaws.com`，以便授予 Amazon IoT 调用 Lambda 函数的权限。

`--source-arn`

规则的 ARN。您可以使用 `get-topic-rule` Amazon CLI 命令来获取规则的 ARN。

`--source-account`

在其中定义规则的Amazon Web Services 账户。

`--statement-id`

唯一的语句标识符。

`--action`

要在此语句中允许的 Lambda 操作。要允许 Amazon IoT 调用 Lambda 函数，请指定 `lambda:InvokeFunction`。

Important

如果您在不提供源 Amazon IoT 或 `source-arn`的情况下为 `source-account` 委托人添加权限，则通过 Lambda 操作创建规则的所有 Amazon Web Services 账户 都可以触发规则，以便从 Amazon IoT 调用 Lambda 函数。

有关更多信息，请参阅 Amazon Lambda 权限。

- 如果您使用 Amazon Key Management Service (Amazon KMS) 客户托管的 Amazon KMS key （KMS 密钥）对 Lambda 中的数据进行静态加密，服务必须具有代表调用方使用 Amazon KMS key 的权限。有关更多信息，请参阅 Amazon Lambda开发人员指南中的静态加密。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`functionArn`

要调用的 Lambda 函数的 ARN。Amazon IoT 必须具有调用函数的权限。有关更多信息，请参阅要求 (p. 503)：

如果您未指定 Lambda 函数的版本或别名，则将执行该函数的最新版本。如果要执行 Lambda 函数的特定版本，则可执行其版本或别名。要指定一个版本或别名，请将该版本或别名附加到 Lambda 函数的 ARN。

```
arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction:someAlias
```

有关版本控制和别名的更多信息，请参阅 Amazon Lambda 函数版本控制和别名。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

示例

下面的 JSON 示例定义了 Amazon IoT 规则中的 Lambda 函数。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
```

```
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-
east-2:123456789012:function:myLambdaFunction"
                 }
            }
        ]
    }
}
```

下面的 JSON 示例定义了 Amazon IoT 规则中一个带有替代模板的 Lambda 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "lambda": {
                    "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:
${topic()}"
                }
            }
        ]
    }
}
```

另请参阅

- Amazon Lambda 开发人员指南中的什么是 Amazon Lambda？
- 教程：使用 Amazon Lambda函数格式化通知 (p. 194)

位置

Location (`location`) 操作会将您的地理位置数据路由到 Amazon Location Service。

> **Note**
>
> 此操作在中国不可用。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `geo:BatchUpdateDevicePosition` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`deviceId`

提供位置数据的设备的唯一 ID。有关更多信息，请参阅 Amazon Location Service API 参考 中的 `DeviceId`。

支持替换模板 (p. 585)：是

`latitude`

计算结果为表示设备位置纬度的双精度值的字符串。

支持替换模板 (p. 585)：是

`longitude`

计算结果为表示设备位置经度的双精度值的字符串。

支持替换模板 (p. 585)：是

`roleArn`

允许访问 Amazon Location Service 域的 IAM 角色。有关更多信息，请参阅要求 (p. 505)：

`timestamp`

对位置数据进行采样的时间。默认值是处理 MQTT 消息的时间。

`timestamp` 值由以下两个值组成：

- `value`：返回长纪元时间值的表达式。您可以使用 the section called “time_to_epoch(String, String)” (p. 577) 函数从消息负载中传递的日期或时间值创建有效的时间戳。支持替换模板 (p. 585)：是。
- `unit`：（可选）`value` 中描述的表达式生成的时间戳值的精度。有效值：`SECONDS` | `MILLISECONDS` |`MICROSECONDS` | `NANOSECONDS`。默认为 `MILLISECONDS`。支持替换模板 (p. 585)：仅限 API 和 Amazon CLI。

`trackerName`

Amazon Location 中更新位置的跟踪器资源的名称。有关更多信息，请参阅 Amazon Location Service 开发人员指南 中的跟踪器。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

示例

以下 JSON 示例定义了 Amazon IoT 规则中的定位操作。

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123454962127:role/service-role/ExampleRole",
     "trackerName": "MyTracker",
     "deviceId": "001",
     "sampleTime": {
      "value": "${timestamp()}",
      "unit": "SECONDS"
     },
```

```
     "latitude": "-12.3456",
     "longitude": "65.4321"
    }
   }
  ]
 }
}
```

以下 JSON 示例定义了 Amazon IoT 规则中带有替换模板的定位操作。

```
{
 "topicRulePayload": {
  "sql": "SELECT * FROM 'some/topic'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
   {
    "location": {
     "roleArn": "arn:aws:iam::123456789012:role/service-role/ExampleRole",
     "trackerName": "${TrackerName}",
     "deviceId": "${DeviceID}",
     "timestamp": {
      "value": "${timestamp()}",
      "unit": "SECONDS"
     },
     "latitude": "${get(position, 0)}",
     "longitude": "${get(position, 1)}"
    }
   }
  ]
 }
}
```

以下 MQTT 有效负载示例显示了上面示例中的替换模板如何访问数据。您可以使用 get-device-position-history CLI 命令来验证 MQTT 有效负载数据是否已在位置跟踪器中传递。

```
{
 "TrackerName": "mytracker",
 "DeviceID": "001",
 "position": [
  "-12.3456",
  "65.4321"
 ]
}
```

```
aws location get-device-position-history --device-id 001 --tracker-name mytracker
```

```
{
 "DevicePositions": [
  {
   "DeviceId": "001",
   "Position": [
    -12.3456,
    65.4321
   ],
   "ReceivedTime": "2022-11-11T01:31:54.464000+00:00",
   "SampleTime": "2022-11-11T01:31:54.308000+00:00"
  }
 ]
}
```

另请参阅

- Amazon Location Service 开发人员指南 中的什么是 Amazon Location Service？。

OpenSearch

OpenSearch (openSearch) 操作可将 MQTT 消息中的数据写入 Amazon Serv OpenSearch ice 域。然后，您可以使用像 Dashboards OpenSearch 这样的工具来查询和可视化 Serv OpenSearch ice 中的数据。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 es:ESHttpPut 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用客户托管式Amazon KMS key（KMS 密钥）对 Service 中的 OpenSearch 数据进行静态加密，服务必须具有代表调用方使用 KMS 的权限。有关更多信息，请参阅 Amazon S ervice 开发人员指南中的为 Amazon S OpenSearch erv OpenSearch ice 静态加密数据。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`endpoint`

Amazon Serv OpenSearch ice 域端点。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`index`

您要在其中存储数据的 OpenSearch 索引。

支持替换模板 (p. 585)：是

`type`

您存储的文档类型。

支持替换模板 (p. 585)：是

`id`

每个文档的唯一标识符。

支持替换模板 (p. 585)：是

`roleARN`

允许访问 Service 域的 OpenSearch IM 角色。有关更多信息，请参阅要求 (p. 508)：

支持替换模板 (p. 585)：否

限制

OpenSearch (openSearch) 操作不能用于向 VPC Elasticsearch 集群传送数据。

示例

下面的 JSON 示例定义了Amazon IoT规则中的 OpenSearch 操作，以及如何为`OpenSearch`操作指定域。有关更多信息，请参阅OpenSearchAction。

```
{
    "topicRulePayload": {
        "sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "my-index",
                    "type": "my-type",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

下面的 JSON 示例定义了一个在Amazon IoT规则中带有替换模板的 OpenSearch 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "openSearch": {
                    "endpoint": "https://my-endpoint",
                    "index": "${topic()}",
                    "type": "${type}",
                    "id": "${newuuid()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
                }
            }
        ]
    }
}
```

另请参阅

什么是亚马逊 OpenSearch 服务？ 在《亚马逊 OpenSearch 服务开发者指南》中

Republish

重新发布 (`republish`) 操作可将 MQTT 消息重新发布到其它 MQTT 主题。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `iot:Publish` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`headers`

MQTT 版本 5.0 标头信息。

有关更多信息，请参阅《Amazon API 参考》中的 RepublishAction 和 MqttHeaders。

`topic`

消息重新发布到的 MQTT 主题。

若要重新发布到以 $ 开头的预留主题，请使用 $$ 代替。例如，如果要重新发布到设备影子主题 `$aws/things/MyThing/shadow/update`，请将主题指定为 `$$aws/things/MyThing/shadow/update`。

> **Note**
>
> 不支持重新发布到保留的任务主题 (p. 101)。
> Amazon IoT Device Defender 保留主题不支持 HTTP 发布。

支持替换模板 (p. 585)：是

`qos`

（可选）重新发布消息时要使用的服务质量 (QoS) 级别。有效值：`0`、`1`。默认值为 `0`。有关 MQTT QoS 的更多信息，请参阅 MQTT (p. 77)。

支持替换模板 (p. 585)：否

`roleArn`

允许 Amazon IoT 发布至 MQTT 主题的 IAM 角色。有关更多信息，请参阅要求 (p. 509)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例定义了 Amazon IoT 规则中的重新发布操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "another/topic",
                    "qos": 1,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
        ]
    }
```

```
}
```

以下 JSON 示例定义了 Amazon IoT 规则中带有替代模板的重新发布操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish"
                }
            }
        ]
    }
}
```

以下 JSON 示例在 Amazon IoT 规则中使用 `headers` 定义了重新发布操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "republish": {
                    "topic": "${topic()}/republish",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
                    "headers": {
                        "payloadFormatIndicator": "UTF8_DATA",
                        "contentType": "rule/contentType",
                        "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
                        "userProperties": [
                            {
                                "key": "ruleKey1",
                                "value": "ruleValue1"
                            },
                            {
                                "key": "ruleKey2",
                                "value": "ruleValue2"
                            }
                        ]
                    }
                }
            }
        ]
    }
}
```

S3

S3 (`s3`) 操作将 MQTT 消息中的数据写入 Amazon Simple Storage Service (Amazon S3) 存储桶。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `s3:PutObject` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用 Amazon Key Management Service (Amazon KMS) 客户托管的 Amazon KMS key （KMS 密钥）对 Amazon S3 中的数据进行静态加密，服务必须具有代表调用方使用 Amazon KMS key 的权限。有关更多信息，请参阅 Amazon简单存储服务开发人员指南中的 Amazon 托管 Amazon KMS keys 和客户托管Amazon KMS keys。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`bucket`

数据写入的 Amazon S3 存储桶。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`cannedacl`

（可选）Amazon S3 封装的 ACL，用于控制对由对象键标识的对象的访问权限。有关更多信息，包括允许的值，请参阅 封装 ACL。

支持替换模板 (p. 585)：否

`key`

数据写入的文件路径。

考虑一个例子，其中此参数是 `${topic()}/${timestamp()}`，并且规则会收到一条消息，其中主题为 `some/topic`。如果当前时间戳为 `1460685389`，那么这个操作会将数据写入一个 S3 存储桶中名为 `some/topic` 的文件夹中的文件 `1460685389`。

> **Note**
>
> 如果您使用静态密钥，Amazon IoT 每次调用规则时都会覆盖单个文件。我们建议您使用消息时间戳或其它唯一的消息标识符，这样可以在 Amazon S3 中针对接收的每个消息保存一个新文件。

支持替换模板 (p. 585)：是

`roleArn`

允许访问 Amazon S3 存储桶的 IAM 角色。有关更多信息，请参阅要求 (p. 511)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例介绍了如何在 Amazon IoT 规则中定义 S3 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
```

```
            {
                "s3": {
                    "bucketName": "my-bucket",
                    "cannedacl": "public-read",
                    "key": "${topic()}/${timestamp()}",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3"
                }
            }
        ]
    }
}
```

另请参阅

- Amazon Simple Storage Service 用户指南中的什么是 Amazon S3？

Salesforce IoT

Salesforce IoT (`salesforce`) 操作将来自触发了规则的 MQTT 消息数据发送到 Salesforce IoT 输入流。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`url`

由 Salesforce IoT 输入流公开的 URL。在创建输入流时，可从 Salesforce IoT 平台获得该 URL。有关更多信息，请参阅 Salesforce IoT 文档。

支持替换模板 (p. 585)：否

`token`

用于验证对指定的 Salesforce IoT 输入流的访问的令牌。在创建输入流时，可从 Salesforce IoT 平台获得该令牌。有关更多信息，请参阅 Salesforce IoT 文档。

支持替换模板 (p. 585)：否

示例

以下 JSON 示例定义了 Amazon IoT 规则中的 Salesforce IoT 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "salesforce": {
                    "token": "ABCDEFGHI123456789abcdefghi123456789",
                    "url": "https://ingestion-cluster-id.my-env.sfdcnow.com/streams/stream-
id/connection-id/my-event"
                }
            }
        ]
    }
```

```
}
```

SNS

SNS (sns) 操作将 MQTT 消息中的数据作为 Amazon Simple Notification Service (Amazon SNS) 推送通知发送。

您可以按照教程执行操作，该教程向您说明如何使用 SNS 操作创建和测试规则。有关更多信息，请参阅教程：发送 Amazon SNS 通知 (p. 182)：

> Note
>
> SNS 操作不支持Amazon SNS FIFO（先进先出）主题。由于规则引擎是一个完全分布式服务，因此无法保证触发 SNS 操作时的消息顺序。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `sns:Publish` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用 Amazon Key Management Service (Amazon KMS) 客户托管的 Amazon KMS key（KMS 密钥）对 Amazon SNS 中的数据进行静态加密，服务必须具有代表调用方使用 Amazon KMS key 的权限。有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的密钥管理。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`targetArn`

推送通知将发送到的 SNS 主题或单个设备。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`messageFormat`

（可选）消息格式。Amazon SNS 使用此设置来确定是否应解析负载，以及是否应提取负载的特定于平台的相关部分。有效值：`JSON`、`RAW`。默认值为 `RAW`。

支持替换模板 (p. 585)：否

`roleArn`

允许访问 SNS 的 IAM 角色。有关更多信息，请参阅要求 (p. 514)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例介绍了如何在 Amazon IoT 规则中定义 SNS 操作。

```
{
    "topicRulePayload": {
```

```
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-2:123456789012:my_sns_topic",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

以下 JSON 示例定义了 Amazon IoT 规则中带有替代模板的 SNS 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sns": {
                    "targetArn": "arn:aws:sns:us-east-1:123456789012:${topic()}",
                    "messageFormat": "JSON",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sns"
                }
            }
        ]
    }
}
```

另请参阅

- Amazon Simple Notification Service 开发人员指南中的什么是 Amazon Simple Notification Service？
- 教程：发送 Amazon SNS 通知 (p. 182)

SQS

SQS (`sqs`) 操作将 MQTT 消息中的数据发送到 Amazon Simple Queue Service (Amazon SQS) 队列。

> **Note**
>
> SQS 操作不支持 Amazon SQS FIFO（先进先出）队列。由于规则引擎是一个完全分布式服务，因此无法保证触发 SQS 操作时的消息顺序。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `sqs:SendMessage` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。
- 如果您使用 Amazon Key Management Service (Amazon KMS) 客户托管的 Amazon KMS key（KMS 密钥）对 Amazon SQS 中的数据进行静态加密，服务必须具有代表调用方使用 Amazon KMS key 的权限。有关更多信息，请参阅 Amazon Simple Queue Service 开发人员指南中的密钥管理。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`queueUrl`

数据写入的 Amazon SQS 队列的 URL。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`useBase64`

将该参数设置为 `true` 以配置规则操作，以便在将数据写入 Amazon SQS 队列之前对消息数据进行 base64 编码。默认值为 `false`。

支持替换模板 (p. 585)：否

`roleArn`

允许访问 Amazon SQS 队列的 IAM 角色。有关更多信息，请参阅要求 (p. 515)：

支持替换模板 (p. 585)：否

示例

下面的 JSON 示例介绍了如何在 Amazon IoT 规则中定义 SQS 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
my_sqs_queue",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
}
```

以下 JSON 示例定义了 Amazon IoT 规则中带有替代模板的 SQS 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "sqs": {
                    "queueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/
${topic()}",
                    "useBase64": true,
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_sqs"
                }
            }
        ]
    }
```

```
}
```

另请参阅

- Amazon Simple Queue Service 开发人员指南中的什么是 Amazon Simple Queue Service？

Step Functions

Step Functions (`stepFunctions`) 操作启动 Amazon Step Functions 状态机。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `states:StartExecution` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择或创建一个角色以允许 Amazon IoT 执行此规则操作。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`stateMachineName`

开始执行 Step Functions 状态机的名称。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

`executionNamePrefix`

（可选）将为状态机执行操作指定名称，由此前缀后加 UUID 组成。如果未提供名称，Step Functions 会自动为每次状态机执行创建一个唯一的名称。

支持替换模板 (p. 585)：是

`roleArn`

授予 Amazon IoT 权限以启动状态机的角色 ARN。有关更多信息，请参阅要求 (p. 517)：

支持替换模板 (p. 585)：否

示例

以下 JSON 示例定义了 Amazon IoT 规则中的 Step Functions 操作。

```
{
    "topicRulePayload": {
        "sql": "SELECT * FROM 'some/topic'",
        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "stepFunctions": {
                    "stateMachineName": "myStateMachine",
                    "executionNamePrefix": "myExecution",
                    "roleArn": "arn:aws:iam::123456789012:role/aws_iot_step_functions"
                }
            }
```

```
        ]
    }
}
```

另请参阅

- Amazon Step Functions 开发人员指南中的什么是 Amazon Step Functions？

Timestream

Timestream 规则操作将 MQTT 消息中的属性（度量）写入 Amazon Timestream 表。有关 Amazon Timestream 的更多信息，请参阅什么是 Amazon Timestream？。

> Note
>
> Amazon Timestream 并未在所有Amazon Web Services 区域中推出。如果 Amazon Timestream 在您所在区域中不可用，则它将不会显示在规则操作列表中。

此规则存储在 Timestream 数据库中的属性是由规则的查询语句生成的属性。对查询语句结果中每个属性的值进行解析以推断其数据类型（如 the section called “DynamoDBv2” (p. 463) 操作）。每个属性的值都写入到 Timestream 表中自己的记录中。要指定或更改属性的数据类型，请在查询语句中使用 `cast()` (p. 547) 函数。有关每条 Timestream 记录内容的更多信息，请参阅 the section called “Amazon Timestream 记录内容” (p. 519)。

> Note
>
> 对于 SQL V2 (2016-03-23)，整数数字值，例如 `10.0`，将转换为整数表示形式 (`10`)。显式将它们转换为 `Decimal` 值，例如通过使用 cast() (p. 547) 函数，不会阻止此行为——结果仍然是 `Integer` 值。这可能会导致类型不匹配错误，从而阻止在 Timestream 数据库中记录数据。若要可靠地将整数数值处理为 `Decimal` 值，请使用 SQL V1 (2015-10-08) 作为规则查询语句。

> Note
>
> Timestream 规则操作可以写入 Amazon Timestream 表的值的最大数目为 100。有关更多信息，请参阅 Amazon Timestream 配额参考。

要求

此规则操作具有以下要求：

- Amazon IoT 可以承担以执行 `timestream:DescribeEndpoints` 和 `timestream:WriteRecords` 操作的 IAM 角色。有关更多信息，请参阅向 Amazon IoT 规则授予所需的访问权限 (p. 443)：

  在 Amazon IoT 控制台中，您可以选择、更新或创建一个角色以允许 Amazon IoT 以执行此规则操作。
- 如果使用客户托管的 Amazon Key Management Service (Amazon KMS) 加密 Timestream 中的静态数据，服务必须具有代表调用方使用 Amazon KMS key 的权限。有关更多信息，请参阅 Amazon 服务如何使用 Amazon KMS。

参数

使用此操作创建 Amazon IoT 规则时，您必须指定以下信息：

`databaseName`

Amazon Timestream 数据库的名称，该数据库中具有用于接收此操作创建的记录的表。另请参阅 **`tableName`**。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

dimensions

写入每个度量记录的时间序列的元数据属性。例如，EC2 实例的名称和可用区或风力涡轮机制造商的名称都是维度。

name

元数据维度名称。这是数据库表记录中列的名称。

维度不能命名为：measure_name、measure_value 或者 time。这些是预留的名称。维度名称不能以 ts_ 或者 measure_value 开头，并且它们不能包含冒号 (:) 字符。

支持替换模板 (p. 585)：否

value

在数据库记录的此列中写入的值。

支持替换模板 (p. 585)：是

roleArn

角色的 Amazon Resource Name (ARN)，该角色授予 Amazon IoT 写入 Timestream 数据库表的权限。有关更多信息，请参阅要求 (p. 518)：

支持替换模板 (p. 585)：否

tableName

要将度量记录写入的数据库表的名称。另请参阅 **databaseName**。

支持替换模板 (p. 585)：仅限 API 和 Amazon CLI

timestamp

用于条目的时间戳的值。如果为空，则使用处理条目的时间。

unit

value 中描述的表达式生成的时间戳值的精度。

有效值：SECONDS | MILLISECONDS |MICROSECONDS | NANOSECONDS。默认为 MILLISECONDS。

value

返回长纪元时间值的表达式。

您可以使用 the section called “time_to_epoch(String, String)” (p. 577) 函数从消息负载中传递的日期或时间值创建有效的时间戳。

Amazon Timestream 记录内容

通过此操作写入 Amazon Timestream 表的数据包括时间戳、Timestream 规则操作中的元数据以及规则查询语句的结果。

对于查询语句结果中的每个属性（度量），此规则操作将记录写入具有这些列的指定 Timestream 表。

列名称	属性类型	值	注释
####	维度	在 Timestream 规则操作条目中指定的值。	每个维度都会在规则操作条目中指定的 Timestream 数据库中创建一个具有维度名称的列。

列名称	属性类型	值	注释
measure_name	MEASURE_NAME	属性的名称	查询语句结果中的属性名称，其值已在 measure_value::*data-type* 列。
measure_value::*data-type*	MEASURE_VALUE	查询语句结果中属性的值。属性的名称位于 measure_name 列。	该值被解释* 并转换为最佳匹配：bigint、boolean、double 或者 varchar。Amazon Timestream 为每种数据类型创建一个单独的列。消息中的值可以转换为另一个数据类型，方法是在规则的查询语句中使用 cast() (p. 547) 函数。
time	TIMESTAMP	数据库中记录的日期和时间。	此值由规则引擎或 timestamp 属性分配（如果已定义）。

* 从消息负载读取的属性值解释如下。请参阅 the section called “示例” (p. 520)，了解各个案例的说明。

- 一个无引号的 true 或者 false 值将被解释为 boolean 类型。
- 十进制数字被解释为 double 类型。
- 没有小数点的数值被解释为 bigint 类型。
- 带引号的字符串被解释为 varchar 类型。
- 对象和数组值将转换为 JSON 字符串并存储为 varchar 类型。

示例

以下 JSON 示例定义了 Amazon IoT 规则内包含替换模板的 Timestream 规则操作。

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'iot/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "timestream": {
          "roleArn": "arn:aws:iam::123456789012:role/aws_iot_timestream",
          "tableName": "devices_metrics",
          "dimensions": [
            {
              "name": "device_id",
              "value": "${clientId()}"
            },
            {
              "name": "device_firmware_sku",
              "value": "My Static Metadata"
            }
```

```
          ],
          "databaseName": "record_devices"
        }
      }
    ]
  }
}
```

将上一示例中定义的 Timestream 主题规则操作与以下消息负载结合使用会使得 Amazon Timestream 记录写入下表中。

```
{
  "boolean_value": true,
  "integer_value": 123456789012,
  "double_value": 123.456789012,
  "string_value": "String value",
  "boolean_value_as_string": "true",
  "integer_value_as_string": "123456789012",
  "double_value_as_string": "123.456789012",
  "array_of_integers": [23,36,56,72],
  "array of strings": ["red", "green","blue"],
  "complex_value": {
    "simple_element": 42,
    "array_of_integers": [23,36,56,72],
    "array of strings": ["red", "green","blue"]
  }
}
```

下表显示了使用指定主题规则操作处理先前创建消息负载的数据库列和记录。`device_firmware_sku` 和 `device_id` 列是主题规则操作中定义的维度。Timestream 主题规则操作将创建 `time` 列以及 `measure_name` 和 `measure_value::*` 列，它将使用主题规则操作的查询语句结果中的值来填充这些值。

device_firmw	device_id	measure_na	measure_va	measure_va	measure_va	measure_va	time
我的静态元数据	iotconsole-150	complex_value	EXAMPLE738-0	{"simple_element":42,"array_of_integers":[23,36,56,72],"array of strings":["red","green","blue"]}	-	-	2020-08-26 22:42:16.423
我的静态元数据	iotconsole-150	integer_value_as_string	EXAMPLE738-0	123456789012	-	-	2020-08-26 22:42:16.423
我的静态元数据	iotconsole-150	boolean_value	EXAMPLE738-0	-	-	TRUE	2020-08-26 22:42:16.423
我的静态元数据	iotconsole-150	integer_value	EXAMPLE738-0	123456789012	-	-	2020-08-26 22:42:16.423
我的静态元数据	iotconsole-150	string_value	EXAMPLE738-0	字符串值	-	-	2020-08-26 22:42:16.423
我的静态元数据	iotconsole-150	array_of_integers	EXAMPLE738-0	[23,36,56,72]	-	-	2020-08-26 22:42:16.423
我的静态元数据	iotconsole-150	字符串数组	EXAMPLE738-0	["red","green","blue"]	-	-	2020-08-26 22:42:16.423
我的静态元数据	iotconsole-150	boolean_value_as_string	EXAMPLE738-0	TRUE	-	-	2020-08-26 22:42:16.423

device_firmw	device_id	measure_na	measure_va	measure_va	measure_va	measure_va	time
我的静态元数据	iotconsole-15	EXAMPLE	38-0	-	123.456789	012	2020-08-26 22:42:16.423
我的静态元数据	iotconsole-15	EXAMPLE	38-0	string	123.45679	-	-	2020-08-26 22:42:16.423

排查规则问题

如果遇到规则问题，应启用 CloudWatch Logs。通过分析您的日志，您可以确定问题是否与授权相关，或者是否为诸如 WHERE 子句状态不匹配的问题。有关更多信息，请参阅设置 CloudWatch 日志。

使用 Amazon IoT 规则访问跨账户资源

您可以配置 Amazon IoT 规则，以便在一个账户的 MQTT 主题上摄取的数据可以路由到另一个账户的 Amazon 服务（如 Amazon SQS 和 Lambda）。下面介绍了如何设置 Amazon IoT 规则，以在从一个账户中的 MQTT 主题到另一个账户中的目标完成跨账户数据摄取。

跨账户规则可以在目标资源上使用基于资源的权限。因此，只有支持基于资源的权限的目标才能使用 Amazon IoT 规则进行跨账户访问。支持的目的地包括 Amazon SQS、Amazon SNS、Amazon S3 和 Amazon Lambda。

> Note
>
> 您必须在与另一个服务资源相同的 Amazon Web Services 区域 中定义规则，以便规则操作可以与该资源进行交互。有关 Amazon IoT 规则操作的更多信息，请参阅 Amazon IoT 规则操作 (p. 449)。

先决条件

- 熟悉 Amazon IoT 规则
- 了解 IAM 用户、角色和基于资源的权限
- 安装 Amazon CLI

Amazon SQS 的跨账户设置

场景：账户 A 将 MQTT 消息中的数据发送到账户 B 的 Amazon SQS 队列。

Amazon Web Services 账户	账户名称	描述
1111-1111-1111	账户 A	规则操作：`sqs:SendMessage`
2222-2222-2222	账户 B	Amazon SQS 队列 • ARN：*arn:aws:sqs:region:2222-2222-2222:ExampleQueu* • URL：*https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue*

执行账户 A 任务

注意

要运行以下命令，您的 IAM 用户应具有 `iot:CreateTopicRule` 使用规则的 Amazon Resource Name (ARN) 作为资源的权限，并且具有 `iam:PassRole` 操作使用资源作为角色的 ARN 的权限。

1. 使用账户 A 的 IAM 用户配置 Amazon CLI。
2. 创建信任 Amazon IoT 规则引擎的 IAM 角色，并附加允许访问账户 B 的 Amazon SQS 队列的策略。请参阅授予 Amazon IoT 所需访问权限中的示例命令和策略文档。
3. 要创建附加到主题的规则，请运行create-topic-rule 命令。

```
aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
```

下面是一个负载文件示例，其中包含的规则会将发送至 `iot/test` 主题的所有消息插入指定 Amazon SQS 队列。SQL 语句筛选信息，角色 ARN 授予 Amazon IoT 写入 Amazon SQS 队列的权限。

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "sqs": {
    "queueUrl": "https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role",
    "useBase64": false
   }
  }
 ]
}
```

有关如何在 Amazon IoT 规则中定义 Amazon SQS 操作的更多信息，请参阅 Amazon IoT 规则操作——Amazon SQS。

执行账户 B 任务

1. 使用账户 B 的 IAM 用户配置 Amazon CLI。
2. 要向账户 A 授予 Amazon SQS 队列资源的权限，请运行 add-permission 命令。

```
aws sqs add-permission --queue-url https://sqs.region.amazonaws.com/2222-2222-2222/
ExampleQueue --label SendMessagesToMyQueue --aws-account-ids 1111-1111-1111 --actions
 SendMessage
```

Amazon SNS 的跨账户设置

场景：账户 A 将 MQTT 消息中的数据发送到账户 B 的 Amazon SNS 主题。

Amazon Web Services 账户	账户名称	描述
1111-1111-1111	账户 A	规则操作：`sns:Publish`
2222-2222-2222	账户 B	Amazon SNS 主题 ARN：*arn:aws:sns:region:2222-2222-2222:ExampleTopic*

执行账户 A 任务

注意

要运行以下命令，您的 IAM 用户应具有 `iot:CreateTopicRule` 的权限，拥有能够作为资源的规则 ARN，并具有 `iam:PassRole` 操作的权限，拥有作为角色 ARN 的资源。

1. 使用账户 A 的 IAM 用户配置 Amazon CLI。
2. 创建信任 Amazon IoT 规则引擎的 IAM 角色，并附加允许访问账户 B 的 Amazon SNS 主题的策略。有关命令和策略文档的示例，请参阅授予 Amazon IoT 所需的访问权限。
3. 要创建附加到主题的规则，请运行create-topic-rule 命令。

```
aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
```

下面是一个负载文件示例，其中包含的规则会将发送至 `iot/test` 主题的所有消息插入指定 Amazon SNS 主题。SQL 语句筛选消息，角色 ARN 授予 Amazon IoT 写入 Amazon SNS 主题的权限。

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "sns": {
    "targetArn": "arn:aws:sns:region:2222-2222-2222:ExampleTopic",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
   }
  }
 ]
}
```

有关如何在 Amazon IoT 规则中定义 Amazon SNS 操作的更多信息，请参 阅Amazon IoT 规则操作——Amazon SNS。

执行账户 B 任务

1. 使用账户 B 的 IAM 用户配置 Amazon CLI。
2. 要向账户 A 授予 Amazon SNS 主题资源的权限，请运行 add-permission 命令。

```
aws sns add-permission --topic-arn arn:aws:sns:region:2222-2222-2222:ExampleTopic --
label Publish-Permission --aws-account-id 1111-1111-1111 --action-name Publish
```

Amazon S3 的跨账户设置

场景：账户 A 将 MQTT 消息中的数据发送到账户 B 的 Amazon S3 存储桶。

Amazon Web Services 账户	账户名称	描述
1111-1111-1111	账户 A	规则操作：`s3:PutObject`
2222-2222-2222	账户 B	Amazon S3 存储桶 ARN：*arn:aws:s3:::ExampleBucket*

执行账户 A 任务

注意

要运行以下命令，您的 IAM 用户应具有 `iot:CreateTopicRule` 的权限，能够将规则 ARN 作为资源，并具有 `iam:PassRole` 操作的权限，将资源作为角色 ARN。

1. 使用账户 A 的 IAM 用户配置 Amazon CLI。
2. 创建信任 Amazon IoT 规则引擎的 IAM 角色，然后附加允许访问账户 B 的 Amazon S3 存储桶的策略。有关命令和策略文档的示例，请参阅授予 Amazon IoT 所需的访问权限。
3. 要创建附加到目标 S3 存储桶的规则，请运行create-topic-rule 命令。

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
```

下面是一个负载文件示例，其中包含的规则会将发送至 `iot/test` 主题的所有消息插入指定 Amazon S3 存储桶。SQL 语句筛选消息，角色 ARN 授予 Amazon IoT 将消息写入 Amazon S3 存储桶的权限。

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "s3": {
    "bucketName": "ExampleBucket",
    "key": "${topic()}/${timestamp()}",
    "roleArn": "arn:aws:iam::1111-1111-1111:role/my-iot-role"
   }
  }
 ]
}
```

有关如何在 Amazon IoT 规则中定义 Amazon S3 操作的更多信息，请参阅 Amazon IoT 规则操作——Amazon S3。

执行账户 B 任务

1. 使用账户 B 的 IAM 用户配置 Amazon CLI。
2. 创建信任账户 A 的委托人的存储桶策略。

下面的示例负载文件定义信任另一账户委托人的存储桶策略。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AddCannedAcl",
   "Effect": "Allow",
   "Principal": {
    "AWS": [
     "arn:aws:iam::1111-1111-1111:root"
    ]
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::ExampleBucket/*"
  }
 ]
```

```
}
```

有关更多信息，请参阅存储桶策略示例。

3. 要将存储桶策略附加到指定的存储桶，请运行put-bucket-policy 命令。

```
aws s3api put-bucket-policy --bucket ExampleBucket --policy file://./my-bucket-policy.json
```

4. 要跨账户访问工作，请确保您的阻止所有公有访问设置正确。有关更多信息，请参阅 Amazon S3 安全性最佳实践。

Amazon Lambda 跨账户设置

场景：账户 A 调用账户 B 的 Amazon Lambda 函数，传入 MQTT 消息。

Amazon Web Services 账户	账户名称	描述
1111-1111-1111	账户 A	规则操作：`lambda:InvokeFunction`
2222-2222-2222	账户 B	Lambda 函数 ARN：*arn:aws:lambda:region:2222-2222-2222:function:example-function*

执行账户 A 任务

注意

要运行以下命令，您的 IAM 用户应具有 `iot:CreateTopicRule` 的权限，能够将规则 ARN 作为资源，以及对于 `iam:PassRole` 的权限，能够将资源作为角色 ARN。

1. 使用账户 A 的 IAM 用户配置 Amazon CLI。
2. 运行create-topic-rule 命令以创建一个规则，用于定义对账户 B 的 Lambda 函数的跨账户访问。

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
```

下面是一个负载文件示例，其中包含的规则会将发送至 `iot/test` 主题的所有消息插入指定 Lambda 函数。SQL 语句筛选信息，角色 ARN 授予 Amazon IoT 将数据传入 Lambda 函数的权限。

```
{
 "sql": "SELECT * FROM 'iot/test'",
 "ruleDisabled": false,
 "awsIotSqlVersion": "2016-03-23",
 "actions": [
  {
   "lambda": {
    "functionArn": "arn:aws:lambda:region:2222-2222-2222:function:example-function"
   }
  }
 ]
}
```

有关如何在 Amazon IoT 规则中定义 Amazon Lambda 操作的更多信息，请参阅 Amazon IoT 规则操作——Lambda。

执行账户 B 任务

1. 使用账户 B 的 IAM 用户配置 Amazon CLI。
2. 运行 Lambda 的 add-permission 命令以给予 Amazon IoT 规则权限来触发 Lambda 函数。要运行以下命令，您的 IAM 用户应具有 `lambda:AddPermission` 操作的权限。

```
aws lambda add-permission --function-name example-function --region us-east-1 --
principal iot.amazonaws.com --source-arn arn:aws:iot:region:1111-1111-1111:rule/
example-rule --source-account 1111-1111-1111 --statement-id "unique_id" --action
 "lambda:InvokeFunction"
```

选项：

--principal

此字段授予 Amazon IoT（由 `iot.amazonaws.com` 展示）调用 Lambda 函数的权限。

--source-arn

此字段确保仅 Amazon IoT 中的 `arn:aws:iot:region:1111-1111-1111:rule/example-rule` 触发此 Lambda 函数，并且相同或不同账户中的任何其它规则都不能触发此 Lambda 函数。

--source-account

此字段确保 Amazon IoT 仅代表 `1111-1111-1111` 账户触发此 Lambda 函数。

注意

如果您看到一条错误消息在您的 Amazon Lambda 函数的控制台中 Configuration（配置）下显示：“The rule could not be found”（找不到规则），请忽略错误消息并继续测试连接。

错误处理（错误操作）

当 Amazon IoT 从设备接收消息时，规则引擎会检查该消息是否匹配规则。如果匹配，将评估规则的查询语句，然后触发规则的操作并向操作传递查询语句的结果。

如果在触发操作时出现问题，则在为规则指定了错误操作的情况下，规则引擎将触发错误操作。这可能在以下情况下发生：

- 规则没有权限访问 Amazon S3 存储桶。
- 用户错误导致超过 DynamoDB 预置吞吐量。

Note

本主题中介绍的错误处理适用于规则操作 (p. 449)。要调试 SQL 问题，包括外部函数，可以设置 Amazon IoT日志。有关更多信息，请参阅??? (p. 394)：

错误操作消息格式

对于每个规则和每条消息，会生成一条消息。例如，如果同一个规则中的两个规则操作失败，则错误操作将收到包含这两项错误的一条消息。

错误操作消息示例如下所示。

```
{
  "ruleName": "TestAction",
```

```
  "topic": "testme/action",
  "cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
  "clientId": "iotconsole-1511213971966-0",
  "base64OriginalPayload": "ewogICJtZXNzYWdlIjogIkhlbGxvIHZyb20gQVdTIElvVCBjb25zb2xlIgp9",
  "failures": [
    {
      "failedAction": "S3Action",
      "failedResource": "us-east-1-s3-verify-user",
      "errorMessage": "Failed to put S3 object. The error received was The
 specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error
 Code: NoSuchBucket; Request ID: 9DF5416B9B47B9AF; S3 Extended Request ID:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=).
 Message arrived on: error/action, Action: s3, Bucket: us-
east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2:
 yMah1cwPhqTH267QLPhTKeVPKJB8BO5ndBHzOmWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
    }
  ]
}
```

ruleName

触发错误操作的规则的名称。

topic

收到原始消息的主题。

cloudwatchTraceId

登录时使用的唯一标识 CloudWatch。

clientId

消息发布程序的客户端 ID。

base64OriginalPayload

Base64 编码的原始消息负载。

失败

failedAction

无法完成的操作的名称（例如，“S3Action”）。

failedResource

资源的名称（例如 S3 存储桶的名称）。

errorMessage

错误的描述和说明。

错误操作示例

下面是一个具有添加的错误操作的规则示例。以下规则具有将消息数据写入 DynamoDB 表的操作，以及将数据写入 Amazon S3 存储桶的错误操作：

```
{
    "sql" : "SELECT * FROM ..."
    "actions" : [{
        "dynamoDB" : {
            "table" : "PoorlyConfiguredTable",
            "hashKeyField" : "AConstantString",
            "hashKeyValue" : "AHashKey"}}
    ],
    "errorAction" : {
```

```
        "s3" : {
            "roleArn": "arn:aws:iam::123456789012:role/aws_iot_s3",
            "bucketName" : "message-processing-errors",
            "key" : "${replace(topic(), '/', '-') + '-' + timestamp() + '-' + newuuid()}"
        }
    }
}
```

您可以在错误操作的 SQL 语句中使用任意函数或替换，但外部函数除外 (例如，`get_thing_shadow`、`aws_lambda` 和 `machinelearning_predict`)。

有关规则以及如何指定错误操作的更多信息，请参阅创建 Amazon IoT 规则。

有关使用 CloudWatch 监控规则成功还是失败的更多信息，请参阅Amazon IoT 指标和维度 (p. 404)。

借助基本摄取功能，降低消息收发成本

借助基本摄取功能，您能够将设备数据安全发送到 Amazon IoT 规则操作 (p. 449) 所支持的 Amazon 服务，而不会产生消息收发成本。基本摄取功能通过从摄取路径中删除发布/订阅消息代理来优化数据流，因此更具成本效益。

您可以使用基本摄取功能从设备或应用程序发送消息。这些消息的主题名称以 `$aws/rules/`*`rule_name`* 开头（表示其前三个级别），其中 *`rule_name`* 是要调用的 Amazon IoT 规则的名称。

您可以借助基本摄取功能来使用现有规则，方法为将基本摄取前缀 (`$aws/rules/`*`rule_name`*) 添加到通常用于调用规则的消息主题。例如，如果有名为 `BuildingManager` 的规则且其由主题类似 `Buildings/Building5/Floor2/Room201/Lights` (`"sql": "SELECT * FROM 'Buildings/#'"`) 的消息调用，那么通过发送主题为 `$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights` 的消息，您就可以使用基本摄取功能调用相同规则。

注意：

- 您的设备和规则无法订阅基本摄取预留主题。有关更多信息，请参阅 保留的主题 (p. 96)。
- 如果您需要发布/订阅代理来将消息分配给多个订阅者（例如，将消息传送到其它设备以及规则引擎），则应继续使用 Amazon IoT 消息代理来处理消息分配。但是，请确保在基本摄取主题以外的主题上发布您的消息。

使用基本摄取功能

使用基本摄取功能之前，请确保您的设备或应用程序使用具有 `$aws/rules/*` 发布权限的策略 (p. 299)。您也可以为策略中具有 `$aws/rules/`*`rule_name`*`/*` 的各个规则指定权限。否则，您的设备和应用程序可以继续使用与 Amazon IoT Core 的现有连接。

当消息到达规则引擎时，从基本摄取调用的规则与通过消息代理订阅调用的规则之间的执行或错误处理没有任何区别。

您可以创建与基本摄取搭配使用的规则。记住以下内容：

- 基本摄取主题的初始前缀 (`$aws/rules/`*`rule_name`*) 不可用于 topic(Decimal) (p. 577) 函数。
- 如果您定义的规则仅通过基本摄取来调用，则 `FROM` 子句在 `rule` 定义的 `sql` 字段中是可选项目。如果规则还由必须通过消息代理发送的其他消息来调用（例如，因为这些其他消息必须分配给多个订阅者），则其仍是必要项目。有关更多信息，请参阅 Amazon IoT SQL 参考 (p. 530)。
- 基本摄取主题的前三个级别 (`$aws/rules/`*`rule_name`*) 不计入 8 个分段长度限制或针对主题的 256 个总字符限制。否则，应用 Amazon IoT 限制中记录的相同限制。
- 如果所收到消息的 Basic Ingest 主题指定了非活动或不存在的规则，则会在 Amazon CloudWatch 日志中创建错误日志来帮助您进行调试。有关更多信息，请参阅 规则引擎日志条目 (p. 420)。将指示

RuleNotFound 指标，您可以在该指标上创建警报。有关更多信息，请参阅规则指标 (p. 405)中的“规则指标”。

- 您仍可以使用 QoS 1 在基本摄取主题上进行发布。在消息成功传送到规则引擎后，您会收到 PUBACK。收到 PUBACK 并不意味着您的规则操作成功完成。您可以配置错误操作以在操作运行期间处理错误。有关信息，请参阅错误处理（错误操作）(p. 527)。

Amazon IoT SQL 参考

在 Amazon IoT 中，规则通过一种类似 SQL 的语法来定义。SQL 语句由三类子句组成：

SELECT

必需。从传入消息负载提取信息并执行信息转换。要使用的消息由主题筛选条件 (p. 95)在 FROM 子句中指定。

SELECT 子句支持 数据类型 (p. 533)、运算符 (p. 536)、函数 (p. 541)、文本 (p. 582)、Case 语句 (p. 582)、JSON 扩展 (p. 583)、替换模板 (p. 585)、嵌套对象查询 (p. 586) 和 二进制负载 (p. 587)。

FROM

MQTT 消息主题筛选条件 (p. 95)，用于标识要从中提取数据的消息。对于发送到符合此处指定的主题筛选条件的 MQTT 主题的每条消息，将触发该规则。对于通过消息代理传递的消息所触发的规则是必需的。对于仅使用基本提取 (p. 529)功能触发的规则是可选的。

WHERE

（可选）添加用于确定是否执行规则指定的操作的条件逻辑。

WHERE 子句支持 数据类型 (p. 533)、运算符 (p. 536)、函数 (p. 541)、文本 (p. 582)、Case 语句 (p. 582)、JSON 扩展 (p. 583)、替换模板 (p. 585) 和 嵌套对象查询 (p. 586)。

SQL 语句的示例如下所示：

```
SELECT color AS rgb FROM 'topic/subtopic' WHERE temperature > 50
```

MQTT 消息 (也称为传入负载) 的示例如下所示：

```
{
    "color":"red",
    "temperature":100
}
```

如果此消息在 'topic/subtopic' 主题上发布，则触发规则并评估 SQL 语句。如果 color 属性大于 50，SQL 语句将提取 "temperature" 属性的值。WHERE 子句指定条件 temperature > 50。AS 关键字将 "color" 属性重命名为 "rgb"。结果（也称为传出负载）如下所示：

```
{
    "rgb":"red"
}
```

此数据随后将转发至规则的操作，在其中发送数据供后续处理。有关规则操作的更多信息，请参阅 Amazon IoT 规则操作 (p. 449)。

Note

评论目前不支持 Amazon IoT SQL 语法。

包含空格的属性名称不能用作 SQL 语句中的字段名称。虽然传入负载可以在属性名称中包含空格，但这些名称不能在 SQL 语句中使用。但是，如果您使用通配符 (*) 字段名称规范，它们将传递到传出负载。

SELECT 子句

Amazon IoT SELECT 子句基本与 ANSI SQL SELECT 子句相同，仅存在一些细微区别。

SELECT 子句支持 数据类型 (p. 533)、运算符 (p. 536)、函数 (p. 541)、文本 (p. 582)、Case 语句 (p. 582)、JSON 扩展 (p. 583)、替换模板 (p. 585)、嵌套对象查询 (p. 586) 和 二进制负载 (p. 587)。

您可以使用 SELECT 子句从传入 MQTT 消息中提取信息。也可以使用 SELECT * 检索整个传入消息负载。例如：

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT * FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50}
```

如果负载是 JSON 对象，您可以参考对象中的键。您的传出负载将包含键值对。例如：

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL statement: SELECT color FROM 'topic/subtopic'
Outgoing payload: {"color":"red"}
```

您可以使用 AS 关键字重命名键。例如：

```
Incoming payload published on topic 'topic/subtopic':{"color":"red", "temperature":50}
SQL:SELECT color AS my_color FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red"}
```

您可以通过用逗号分隔来选择多个项目。例如：

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT color as my_color, temperature as fahrenheit FROM 'topic/subtopic'
Outgoing payload: {"my_color":"red","fahrenheit":50}
```

您可以通过在向传入负载添加项目时包括“*”来选择多个项目。例如：

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT *, 15 as speed FROM 'topic/subtopic'
Outgoing payload: {"color":"red", "temperature":50, "speed":15}
```

您可以使用 "VALUE" 关键字来生成不属于 JSON 对象的传出负载。使用 SQL 版本 2015-10-08 时，您只能选择一个项目。使用 SQL 版本 2016-03-23 或更高版本时，您还可以选择作为顶级对象输出的数组。

Example

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT VALUE color FROM 'topic/subtopic'
Outgoing payload: "red"
```

您可以使用 '.' 语法深入剖析传入负载中的嵌套 JSON 对象。例如：

```
Incoming payload published on topic 'topic/subtopic': {"color":
{"red":255,"green":0,"blue":0}, "temperature":50}
```

```
SQL: SELECT color.red as red_value FROM 'topic/subtopic'
Outgoing payload: {"red_value":255}
```

有关如何使用包含预留字符（如数字或连字符（减号）的 JSON 对象和属性名称的信息，请参阅 JSON 扩展 (p. 583)

您可以使用函数 (参阅 函数 (p. 541)) 来转换传入负载。您可以使用括号进行分组。例如：

```
Incoming payload published on topic 'topic/subtopic': {"color":"red", "temperature":50}
SQL: SELECT (temperature - 32) * 5 / 9 AS celsius, upper(color) as my_color FROM 'topic/
subtopic'
Outgoing payload: {"celsius":10,"my_color":"RED"}
```

FROM 子句

FROM 子句在主题 (p. 94)或主题筛选条件 (p. 95)中订阅您的规则。您必须将主题或主题筛选条件用单引号 (') 引起来。对于发送到符合此处指定的主题筛选条件的 MQTT 主题的每条消息，将触发该规则。主题筛选条件可用于订阅一组类似的主题。

示例：

传入负载已发布至主题 `'topic/subtopic'`：`{temperature: 50}`

传入负载已发布至主题 `'topic/subtopic-2'`：`{temperature: 50}`

SQL：`"SELECT temperature AS t FROM 'topic/subtopic'"`.

该规则已订阅到 `'topic/subtopic'`，因此传入的负载将传递给该规则。传递给规则操作的传出负载为：`{t: 50}`。规则未订阅 `'topic/subtopic-2'`，因此在 `'topic/subtopic-2'` 上发布的消息不会触发规则。

通配符示例：

您可以使用“#”（多级）通配符来匹配一个或多个特定路径元素：

传入负载已发布至主题 `'topic/subtopic'`：`{temperature: 50}`。

传入负载已发布至主题 `'topic/subtopic-2'`：`{temperature: 60}`。

传入负载已发布至主题 `'topic/subtopic-3/details'`：`{temperature: 70}`。

传入负载已发布至主题 `'topic-2/subtopic-x'`：`{temperature: 80}`。

SQL：`"SELECT temperature AS t FROM 'topic/#'"`.

规则已订阅以 `'topic'` 开头的所有主题，因此它将执行三次，将 `{t: 50}`（适用于 topic/subtopic）、`{t: 60}`（适用于 topic/subtopic-2）和 `{t: 70}`（适用于 topic/subtopic-3/details）的传出负载发送至其操作。它未订阅 `'topic-2/subtopic-x'`，因此，`{temperature: 80}` 消息不会触发规则。

+ 通配符示例：

您可以使用“+”（单级）通配符来匹配任一特定路径元素：

传入负载已发布至主题 `'topic/subtopic'`：`{temperature: 50}`。

传入负载已发布至主题 `'topic/subtopic-2'`：`{temperature: 60}`。

传入负载已发布至主题 `'topic/subtopic-3/details'`：`{temperature: 70}`。

传入负载已发布至主题 `'topic-2/subtopic-x'`：`{temperature: 80}`。

SQL：`"SELECT temperature AS t FROM 'topic/+'"`.

规则已订阅包含两个路径元素的所有主题，其中第一个元素为 'topic'。规则将对发送至 'topic/subtopic' 和 'topic/subtopic-2' 的消息执行，但不对 'topic/subtopic-3/details'（它的级别比主题筛选条件更多）或 'topic-2/subtopic-x'（它不以 topic 开头）执行。

WHERE 子句

WHERE 子句可确定是否执行规则指定的操作。如果 WHERE 子句的计算结果为 true，则执行规则操作。否则，不执行规则操作。

WHERE 子句支持 数据类型 (p. 533)、运算符 (p. 536)、函数 (p. 541)、文本 (p. 582)、Case 语句 (p. 582)、JSON 扩展 (p. 583)、替换模板 (p. 585) 和 嵌套对象查询 (p. 586)。

示例：

传入负载已发布至 topic/subtopic：{"color":"red", "temperature":40}。

SQL：SELECT color AS my_color FROM 'topic/subtopic' WHERE temperature > 50 AND color <> 'red'.

在这种情况下，规则将被触发，但不会执行规则所指定的操作。没有传出的负载。

您可以在 WHERE 子句中使用函数和运算符。但是，您无法引用在 SELECT 中通过 AS 关键字创建的任何别名。首先评估 WHERE 语句，以确定是否评估 SELECT。

带非 JSON 负载的示例：

传入的非 JSON 负载，发布位置：“topic/subtopic”：“80”

SQL：`SELECT decode(encode(*, 'base64'), 'base64') AS value FROM 'topic/subtopic' WHERE decode(encode(*, 'base64'), 'base64') > 50

在这种情况下，规则将被触发，并且将执行规则所指定的操作。传出的负载将由 SELECT 子句转换为 JSON 负载 {"value":80}。

数据类型

Amazon IoT 规则引擎支持所有 JSON 数据类型。

支持的数据类型

类型	意义
Int	离散的 Int。最大 34 位。
Decimal	精度为 34 位的 Decimal，最小非零数量级为 1E-999，最大数量级为 9.999...E999。 Note 部分函数返回双精度 Decimal 值，而不是 34 位数字精度。 对于 SQL V2 (2016-03-23)，整数数字值，例如 10.0，将被处理为 Int 值 (10)，而不是预期的 Decimal 值 (10.0)。若要可靠地将整数数值处理为 Decimal 值，请使用 SQL V1 (2015-10-08) 作为规则查询语句。
Boolean	True 或 False。

类型	意义
`String`	UTF-8 字符串。
`Array`	不必为相同类型的一系列值。
`Object`	包含一个键和一个值的 JSON 值。键必须是字符串。值可以是任意类型。
`Null`	`Null` 由 JSON 定义。它是表示缺少某个值的实际值。您必须通过使用 SQL 语句中 `Null` 关键字明确创建一个 `Null` 值。例如：`"SELECT NULL AS n FROM 'topic/subtopic'"`

`Undefined`	非值。在 JSON 中无法明确表示，只能忽略该值。例如，在对象 `{"foo": null}` 中，键“foo”返回 NULL，但键“bar”返回 `Undefined`。在内部，SQL 语言将 `Undefined` 作为值处理，但它在 JSON 中无法表示，因此，在序列化为 JSON 时，结果为 `Undefined`。
	`{"foo":null, "bar":undefined}`
	序列化为 JSON 如下：
	`{"foo":null}`
	同样，`Undefined` 在被自身序列化时会转化为空字符串。使用无效参数（例如，错误的类型、错误的参数号等）调用的函数将返回 `Undefined`。

转换

下表列出当一个类型的值转换为另一个类型时 (为函数提供错误类型的值时) 返回的结果。例如，如果绝对值函数“abs”(期待的值类型为 `Int` 或 `Decimal`) 被赋予 `String` 时，它会尝试遵循以下规则将 `String` 转换为 `Decimal`。在这种情况下，“abs("-5.123")”将被视为“abs(-5.123)”。

> **Note**
>
> 不会尝试转换 `Array`、`Object`、`Null` 或 `Undefined`。

To decimal

参数类型	结果
`Int`	没有小数点的 `Decimal`。
`Decimal`	源值。
`Boolean`	`Undefined`。（您可以明确使用 cast 函数使 true = 1.0，false = 0.0。）
`String`	SQL 引擎会尝试将字符串解析为 `Decimal`。Amazon IoT 会尝试解析与正则表达式相匹配的字符串：`^-?\d+(\.\d+)?((?i)E-?\d+)?$`。可自动转换为 `Decimal` 的字符串示例包括 "0"、"-1.2"、"5E-12"。

参数类型	结果
数组	Undefined.
对象	Undefined.
Null	Null.
未定义	Undefined.

To int

参数类型	结果
Int	源值。
Decimal	源值都舍入到最接近的 Int。
Boolean	Undefined。（您可以明确使用 cast 函数使 true = 1.0，false = 0.0。）
String	SQL 引擎会尝试将字符串解析为 Decimal。Amazon IoT 会尝试解析与正则表达式相匹配的字符串：^-?\d+(\.\d+)?((?i)E-?\d+)?$。可自动转换为 Decimal 的字符串示例包括 "0"、"-1.2"、"5E-12"。Amazon IoT 尝试将 String 转换为 Decimal，然后截断该 Decimal 的小数位，以得到一个 Int。
数组	Undefined.
对象	Undefined.
Null	Null.
未定义	Undefined.

To Boolean

参数类型	结果
Int	Undefined。（您可以明确使用 cast 函数使 0 = False，any_nonzero_value = True。）
Decimal	Undefined。（您可以明确使用 cast 函数使 0 = False，any_nonzero_value = True。）
Boolean	原始值。
String	"true"=True 和 "false"=False（不区分大小写）。其它字符串值为 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

To string

参数类型	结果
Int	标准表示法中 Int 的字符串表示形式。
Decimal	科学表示法中 Decimal 值的字符串表示。
Boolean	"true" 或 "false"。均为小写。
String	原始值。
数组	Array 序列化为 JSON。结果字符串为逗号分隔的列表，括在方括号中。String 带引号。Decimal、Int、Boolean 和 Null 不带引号。
对象	序列化为 JSON 的对象。结果字符串为键值对的逗号分隔列表，以大括号开头和结束。String 带引号。Decimal、Int、Boolean 和 Null 不带引号。
Null	Undefined.
未定义	Undefined。

运算符

可以在 SELECT 和 WHERE 子句中使用以下运算符。

AND 运算符

返回 Boolean 结果。执行逻辑与运算。如果左右操作数为 true，则返回 true。否则，返回 false。需要提供 Boolean 操作数或不区分大小写的“true”或“false”字符串操作数。

语法： *expression* AND *expression*。

AND 运算符

左侧操作数	右侧操作数	Output
Boolean	Boolean	Boolean。如果两个操作数均为 true，则为 true。否则为 false。
String/Boolean	String/Boolean	如果所有字符串均为“true”或“false”（不区分大小写），则它们将被转换为 Boolean 并作为 *boolean* AND *boolean* 正常处理。
其它值	其它值	Undefined.

OR 运算符

返回 Boolean 结果。执行逻辑或运算。如果左右操作数至少有一个为 true，则返回 true。否则，返回 false。需要提供 Boolean 操作数或不区分大小写的“true”或“false”字符串操作数。

语法： *expression* OR *expression*。

OR 运算符

左侧操作数	右侧操作数	Output
`Boolean`	`Boolean`	`Boolean`。如果任意一个操作数为 true，则为 true。否则为 false。
`String/Boolean`	`String/Boolean`	如果所有字符串均为 "true" 或 "false" (不区分大小写)，则将其转换为布尔值并作为 *`boolean`* `OR` *`boolean`* 正常处理。
其它值	其它值	`Undefined`.

NOT 运算符

返回 `Boolean` 结果。执行逻辑非运算。如果操作数为 false，则返回 true。否则返回 true。需要 `Boolean` 操作数或不区分大小写的“true”或“false”字符串操作数。

语法：`NOT` *`expression`*。

NOT 运算符

操作数	Output
`Boolean`	`Boolean`。如果操作数为 false，则为 true。否则为 true。
`String`	如果字符串为“true”或“false”（不区分大小写），它将被转换为对应的布尔值，并返回相反的值。
其它值	`Undefined`.

> 运算符

返回 `Boolean` 结果。如果左侧操作数大于右侧操作数，则返回 true。两个操作数将转换为 `Decimal`，然后进行比较。

语法： *`expression`* `>` *`expression`*。

> 运算符

左侧操作数	右侧操作数	Output
`Int/Decimal`	`Int/Decimal`	`Boolean`。如果左侧操作数大于右侧操作数，则为 true。否则为 false。
`String/Int/Decimal`	`String/Int/Decimal`	如果所有字符串可以转换为 `Decimal`，则为 `Boolean`。如果左侧操作数大于右侧操作数，则返回 true。否则为 false。
其它值	`Undefined`.	`Undefined`.

>= operator

返回 `Boolean` 结果。如果左侧操作数大于等于右侧操作数，则返回 true。两个操作数将转换为 `Decimal`，然后进行比较。

语法：*expression* >= *expression*。

>= operator

左侧操作数	右侧操作数	Output
Int/Decimal	Int/Decimal	Boolean。如果左侧操作数大于等于右侧操作数，则为 true。否则为 false。
String/Int/Decimal	String/Int/Decimal	如果所有字符串可以转换为 Decimal，则为 Boolean。如果左侧操作数大于等于右侧操作数，则返回 true。否则为 false。
其它值	Undefined.	Undefined.

< 运算符

返回 Boolean 结果。如果左侧操作数小于右侧操作数，则返回 true。两个操作数将转换为 Decimal，然后进行比较。

语法：*expression* < *expression*。

< 运算符

左侧操作数	右侧操作数	Output
Int/Decimal	Int/Decimal	Boolean。如果左侧操作数小于右侧操作数，则为 true。否则为 false。
String/Int/Decimal	String/Int/Decimal	如果所有字符串可以转换为 Decimal，则为 Boolean。如果左侧操作数小于右侧操作数，则返回 true。否则为 false。
其它值	Undefined	Undefined

<= 运算符

返回 Boolean 结果。如果左侧操作数小于等于右侧操作数，则返回 true。两个操作数将转换为 Decimal，然后进行比较。

语法：*expression* <= *expression*。

<= 运算符

左侧操作数	右侧操作数	Output
Int/Decimal	Int/Decimal	Boolean。如果左侧操作数小于等于右侧操作数，则为 true。否则为 false。
String/Int/Decimal	String/Int/Decimal	如果所有字符串可以转换为 Decimal，则为 Boolean。如果左侧操作数小于等于右侧操作数，则返回 true。否则为 false。
其它值	Undefined	Undefined

<> 运算符

返回 Boolean 结果。如果左右操作数不相等，则返回 true。否则返回 false。

语法： *expression* <> *expression*。

<> 运算符

左侧操作数	右侧操作数	Output
Int	Int	如果左侧操作数不等于右侧操作数，则为 true。否则为 false。
Decimal	Decimal	如果左侧操作数不等于右侧操作数，则为 true。否则为 false。在比较之前，Int 会被转换为 Decimal。
String	String	如果左侧操作数不等于右侧操作数，则为 true。否则为 false。
数组	数组	如果各个操作数中的项不相等且顺序不同，则为 true。否则为 false
对象	对象	如果各个操作数的键和值均不相等，则为 true。否则为 false。键/值的顺序不重要。
Null	Null	False.
任意值	Undefined	Undefined。
Undefined	任意值	Undefined。
不匹配的类型	不匹配的类型	True.

= 运算符

返回 Boolean 结果。如果左右操作数相等，则返回 true。否则返回 false。

语法： *expression* = *expression*。

= 运算符

左侧操作数	右侧操作数	Output
Int	Int	如果左侧操作数等于右侧操作数，则为 true。否则为 false。
Decimal	Decimal	如果左侧操作数等于右侧操作数，则为 true。否则为 false。在比较之前，Int 会被转换为 Decimal。
String	String	如果左侧操作数等于右侧操作数，则为 true。否则为 false。
数组	数组	如果各个操作数中的项相等且顺序相同，则为 true。否则为 false。
对象	对象	如果各个操作数的键和值相等，则为 true。否则为 false。键/值的顺序不重要。
任意值	Undefined	Undefined.
Undefined	任意值	Undefined.
不匹配的类型	不匹配的类型	False.

+ 运算符

“+”是一个重载运算符。它可用于字符串联接或相加。

语法： *expression* + *expression*。

+ 运算符

左侧操作数	右侧操作数	Output
String	任意值	将右侧操作数转换为一个字符串，并联接到左侧操作数的末尾。
任意值	String	将左侧操作数转换为一个字符串，并将右侧操作数联接到转换后的左侧操作数的末尾。
Int	Int	Int 值。将操作数相加。
Int/Decimal	Int/Decimal	Decimal 值。将操作数相加。
其它值	其它值	Undefined.

- 运算符

从左侧操作数中减去右侧操作数。

语法： *expression* - *expression*。

- 运算符

左侧操作数	右侧操作数	Output
Int	Int	Int 值。从左侧操作数中减去右侧操作数。
Int/Decimal	Int/Decimal	Decimal 值。从左侧操作数中减去右侧操作数。
String/Int/Decimal	String/Int/Decimal	如果所有字符串都正确地转换为小数，则返回 Decimal 值。从左侧操作数中减去右侧操作数。否则返回 Undefined。
其它值	其它值	Undefined.
其它值	其它值	Undefined.

* 运算符

左侧操作数乘以右侧操作数。

语法： *expression* * *expression*。

* 运算符

左侧操作数	右侧操作数	Output
Int	Int	Int 值。左侧操作数乘以右侧操作数。
Int/Decimal	Int/Decimal	Decimal 值。左侧操作数乘以右侧操作数。
String/Int/Decimal	String/Int/Decimal	如果所有字符串都正确地转换为小数，则返回 Decimal 值。左侧操作数乘以右侧操作数。否则返回 Undefined。
其它值	其它值	Undefined.

/ 运算符

左侧操作数除以右侧操作数。

语法： *expression* / *expression*。

/ 运算符

左侧操作数	右侧操作数	Output
Int	Int	Int 值。左侧操作数除以右侧操作数。
Int/Decimal	Int/Decimal	Decimal 值。左侧操作数除以右侧操作数。
String/Int/Decimal	String/Int/Decimal	如果所有字符串都正确地转换为小数，则返回 Decimal 值。左侧操作数除以右侧操作数。否则返回 Undefined。
其它值	其它值	Undefined.

% 运算符

返回左侧操作数除以右侧操作数得到的余数。

语法： *expression* % *expression*。

% 运算符

左侧操作数	右侧操作数	Output
Int	Int	Int 值。返回左侧操作数除以右侧操作数得到的余数。
String/Int/Decimal	String/Int/Decimal	如果所有字符串都正确地转换为小数，则返回 Decimal 值。返回左侧操作数除以右侧操作数得到的余数。否则为 Undefined。
其它值	其它值	Undefined.

函数

您可以使用 SQL 表达式的 SELECT 或 WHERE 子句中的以下内置函数。

abs(Decimal)

返回数字的绝对值。SQL 版本 2015-10-08 及更高版本支持。

示例：`abs(-5)` 返回 5。

参数类型	结果
Int	Int，参数的绝对值。
Decimal	Decimal，参数的绝对值。
Boolean	Undefined.

参数类型	结果
String	Decimal。结果是参数的绝对值。如果字符串无法转换，则结果为 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

accountid()

以 String 形式返回拥有该规则的账户的 ID。SQL 版本 2015-10-08 及更高版本支持。

示例：

```
accountid()  = "123456789012"
```

acos(Decimal)

以弧度为单位返回数字的反余弦值。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：`acos(0)` = 1.5707963267948966

参数类型	结果
Int	Decimal (双精度)，参数的反余弦值。虚数结果返回 Undefined。
Decimal	Decimal (双精度)，参数的反余弦值。虚数结果返回 Undefined。
Boolean	Undefined.
String	Decimal，参数的反余弦值。如果字符串无法转换，则结果为 Undefined。虚数结果返回 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

asin(Decimal)

以弧度为单位返回数字的反正弦值。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：`asin(0)` = 0.0

参数类型	结果
Int	Decimal (双精度)，参数的反正弦值。虚数结果返回 Undefined。
Decimal	Decimal (双精度)，参数的反正弦值。虚数结果返回 Undefined。
Boolean	Undefined.
String	Decimal (双精度)，参数的反正弦值。如果字符串无法转换，则结果为 Undefined。虚数结果返回 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

atan(Decimal)

以弧度为单位返回数字的反正切值。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：`atan(0)` = 0.0

参数类型	结果
Int	Decimal (双精度)，参数的反正切值。虚数结果返回 Undefined。
Decimal	Decimal (双精度)，参数的反正切值。虚数结果返回 Undefined。
Boolean	Undefined.
String	Decimal，参数的反正切值。如果字符串无法转换，则结果为 Undefined。虚数结果返回 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

atan2(Decimal, Decimal)

以弧度的形式返回 x 轴正方向与由两个参数定义的 (x,y) 点之间的角度。 逆时针的角，角度为正数（上半平面，y > 0），顺时针的角，角度为负数（下半平面，y < 0）。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：`atan2(1, 0)` = 1.5707963267948966

参数类型	参数类型	结果
Int/Decimal	Int/Decimal	Decimal (双精度)，x
Int/Decimal/String	Int/Decimal/String	Decimal，所描述点的结果为 Undefined。
其它值	其它值	Undefined.

aws_lambda(functionArn, inputJson)

调用指定的 Lambda 函数并将 inputJson 传递给 Lambda 函数，并且返回 Lambda 函数生成的 JSON。

Arguments

参数	描述
functionArn	要调用的 Lambda 函数的 ARN。Lambda 函数必须返回 JSON 数据。
inputJson	传递到 Lambda 函数的 JSON 输入。要传递嵌套对象查询和文本，您必须使用 SQL 版本 2016-03-23。

您必须授予 Amazon IoT lambda:InvokeFunction 权限，以便调用指定的 Lambda 函数。以下示例说明如何使用 Amazon CLI 授予 lambda:InvokeFunction 权限：

```
aws lambda add-permission --function-name "function_name"
--region "region"
--principal iot.amazonaws.com
--source-arn arn:aws:iot:us-east-1:account_id:rule/rule_name
--source-account "account_id"
--statement-id "unique_id"
--action "lambda:InvokeFunction"
```

add-permission 命令的参数如下：

--function-name

Lambda 函数的名称 添加新的权限来更新函数的资源策略。

--region

您账户的 Amazon Web Services 区域。

--principal

获取权限的委托人。这应该是 iot.amazonaws.com，以便授予 Amazon IoT 权限调用 Lambda 函数。

--source-arn

规则的 ARN。您可以使用 get-topic-rule Amazon CLI 命令来获取规则的 ARN。

--source-account

在其中定义规则的Amazon Web Services 账户。

--statement-id

唯一的语句标识符。

--action

要在此声明中允许的 Lambda 操作。要允许 Amazon IoT 调用 Lambda 函数，请指定 `lambda:InvokeFunction`。

Important

如果您在不提供源 `source-arn` 或 `source-account`的情况下为 Amazon IoT 委托人添加权限，则通过 Lambda 操作创建规则的所有 Amazon Web Services 账户 都可以触发规则，以便从 Amazon IoT 调用 Lambda 函数。有关更多信息，请参阅 Lambda 权限模型。

假设 JSON 消息负载如下所示：

```
{
    "attribute1": 21,
    "attribute2": "value"
}
```

`aws_lambda` 函数可用于调用 Lambda 函数，如下所示。

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function",
 {"payload":attribute1}) as output FROM 'topic-filter'
```

如果您想要传递完整的 MQTT 消息负载，您可以使用“*”指定 JSON 负载，如下示例所示。

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function", *) as output
 FROM 'topic-filter'
```

`payload.inner.element` 从在主题“topic/subtopic”上发布的消息选择数据。

`some.value` 从 Lambda 函数生成的输出中选择数据。

Note

规则引擎限制 Lambda 函数的执行持续时间。从规则执行的 Lambda 函数调用应在 2000 毫秒内完成。

bitand(Int, Int)

在两个 `Int` (转换成的) 参数的位表示之间逐位执行与运算。SQL 版本 2015-10-08 及更高版本支持。

示例：`bitand(13, 5) = 5`

参数类型	参数类型	结果
`Int`	`Int`	`Int`，对两个参数逐位
`Int/Decimal`	`Int/Decimal`	`Int`，对两个参数逐位入至最近的 `Int`。如果 `Undefined`。
`Int/Decimal/String`	`Int/Decimal/String`	`Int`，对两个参数逐位小数并向下舍入至最近 `Undefined`。
其它值	其它值	`Undefined`.

bitor(Int, Int)

在两个参数的位表示之间逐位执行或运算。SQL 版本 2015-10-08 及更高版本支持。

示例：`bitor(8, 5)` = 13

参数类型	参数类型	结果
Int	Int	Int，对两个参数逐位
Int/Decimal	Int/Decimal	Int，对两个参数逐位入至最近的 Int。如果
Int/Decimal/String	Int/Decimal/String	Int，对两个参数逐位小数并向下舍入至最近 Undefined。
其它值	其它值	Undefined.

bitxor(Int, Int)

在两个 Int (转换成的) 参数的位表示之间逐位执行异或运算。SQL 版本 2015-10-08 及更高版本支持。

示例：`bitor(13, 5)` = 8

参数类型	参数类型	结果
Int	Int	Int，对两个参数逐位
Int/Decimal	Int/Decimal	Int，对两个参数逐位至最近的 Int。
Int/Decimal/String	Int/Decimal/String	Int，对两个参数逐位并向下舍入到最近的 I Undefined。
其它值	其它值	Undefined.

bitnot(Int)

对 Int (转换成的) 参数的位表示逐位执行非运算。SQL 版本 2015-10-08 及更高版本支持。

示例：`bitnot(13)` = 2

参数类型	结果
Int	Int，对参数逐位执行非运算。
Decimal	Int，对参数逐位执行非运算。Decimal 值会向下舍入至最近的 Int。
String	Int，对参数逐位执行非运算。字符串转换为小数值并向下舍入至最近的 Int。如果任何转换失败，结果为 Undefined。
其它值	其它值。

cast()

将值从一个数据类型转换为另一个数据类型。强制转换与标准转换大体相同，但是增加了在数字与布尔值之间进行强制转换的能力。如果 Amazon IoT 无法确定如何将一个数据类型强制转换为另一个数据类型，则结果为 `Undefined`。SQL 版本 2015-10-08 及更高版本支持。格式：cast(# as ##)。

示例：

```
cast(true as Int) = 1
```

在调用 `cast` 时，“as”后面可能出现以下关键字：

对于 SQL 版本 2015-10-08 和 2016-03-23

Keyword	结果
`String`	将值强制转换为 `String`。
Nvarchar	将值强制转换为 `String`。
文本	将值强制转换为 `String`。
Ntext	将值强制转换为 `String`。
varchar	将值强制转换为 `String`。
`Int`	将值强制转换为 `Int`。
整数	将值强制转换为 `Int`。
Double	将值强制转换为 `Decimal`（双精度）。

此外，对于 SQL 版本 2016-03-23

Keyword	结果
`Decimal`	将值强制转换为 `Decimal`。
Bool	将值强制转换为 `Boolean`。
`Boolean`	将值强制转换为 `Boolean`。

强制转换规则：

强制转换为 decimal

参数类型	结果
`Int`	没有小数点的 `Decimal`。
`Decimal`	源值。 **Note** 对于 SQL V2 (2016-03-23)，整数数字值，例如 `10.0`，将返回一个 `Int` 值 (`10`)，而不是预期的 `Decimal` 值 (`10.0`)。若要可靠地将整数数值转换为 `Decimal` 值，请使用 SQL V1 (2015-10-08) 作为规则查询语句。

参数类型	结果
Boolean	true = 1.0，false = 0.0。
String	尝试将字符串解析为 Decimal。Amazon IoT 会尝试解析与正则表达式相匹配的字符串：^-?\d+(\.\d+)?((?i)E-?\d+)?$。可自动转换为小数的字符串示例包括 "0"、"-1.2"、"5E-12"。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

强制转换为 int

参数类型	结果
Int	源值。
Decimal	源值，向下舍入到最近的 Int。
Boolean	true = 1.0，false = 0.0。
String	尝试将字符串解析为 Decimal。Amazon IoT 会尝试解析与正则表达式相匹配的字符串：^-?\d+(\.\d+)?((?i)E-?\d+)?$。可自动转换为小数的字符串示例包括 "0"、"-1.2"、"5E-12"。Int 尝试将字符串转换为 Amazon IoT，然后向下舍入到最近的 Decimal。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

强制转换为 **Boolean**

参数类型	结果
Int	0 = False，任何非零值 = True。
Decimal	0 = False，任何非零值 = True。
Boolean	源值。
String	"true" = True 和 "false" = False（不区分大小写）。其它字符串值 = Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.

参数类型	结果
未定义	Undefined.

强制转换为 string

参数类型	结果
Int	标准表示法中 Int 的字符串表示形式。
Decimal	科学表示法中 Decimal 值的字符串表示。
Boolean	"true" 或 "false"，全小写。
String	"true"=True 和 "false"=False (不区分大小写)。其它字符串值 = Undefined。
数组	数组序列化为 JSON。结果字符串为逗号分隔的列表，括在方括号中。 String 带引号。 Decimal、Int 和 Boolean 不带引号。
对象	序列化为 JSON 的对象。JSON 字符串为键值对的逗号分隔列表，以大括号开头和结束。String 带引号。Decimal、Int、Boolean 和 Null 不带引号。
Null	Undefined.
未定义	Undefined.

ceil(Decimal)

将给定的 Decimal 向上舍入到最近的 Int。SQL 版本 2015-10-08 及更高版本支持。

示例：

ceil(1.2) = 2

ceil(-1.2) = -1

参数类型	结果
Int	Int，参数值。
Decimal	Int，Decimal 值向上舍入到最近的 Int。
String	Int。字符串将转换为 Decimal 并向上舍入到最近的 Int。如果字符串无法转换为 Decimal，则结果为 Undefined。
其它值	Undefined.

chr(String)

返回给定 Int 参数对应的 ASCII 字符。SQL 版本 2015-10-08 及更高版本支持。

示例：

chr(65) = "A".

chr(49) = "1".

参数类型	结果
Int	与指定的 ASCII 值对应的字符。如果参数不是有效的 ASCII 值，则结果为 Undefined。
Decimal	与指定的 ASCII 值对应的字符。Decimal 参数会向下舍入至最近的 Int。如果参数不是有效的 ASCII 值，则结果为 Undefined。
Boolean	Undefined.
String	如果 String 可以转换为 Decimal，则向下舍入到最近的 Int。如果参数不是有效的 ASCII 值，则结果为 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
其它值	Undefined.

clientid()

返回发送消息的 MQTT 客户端的 ID，如果未通过 MQTT 发送消息，则返回 n/a。SQL 版本 2015-10-08 及更高版本支持。

示例：

clientid() = "123456789012"

concat()

联接数组或字符串。该函数可接受任意数量的参数，并返回 String 或 Array。SQL 版本 2015-10-08 及更高版本支持。

示例：

concat() = Undefined.

concat(1) = "1".

concat([1, 2, 3], 4) = [1, 2, 3, 4]。

concat([1, 2, 3], "hello") = [1, 2, 3, "hello"]

concat("con", "cat") = "concat"

concat(1, "hello") = "1hello"

concat("he","is","man") = "heisman"

concat([1, 2, 3], "hello", [4, 5, 6]) = [1, 2, 3, "hello", 4, 5, 6]

参数数量	结果
0	Undefined.
1	不经修改返回参数。
2+	如果任意参数为 Array，那么结果为包含所有参数的一个数组。如果没有参数为数组，并且至少有一个参数为 String，则结果是所有参数的 String 表示的联接。参数将使用上文列出的标准转换被转换为字符串。 .

cos(Decimal)

以弧度为单位返回数字的余弦值。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：

cos(0) = 1.

参数类型	结果
Int	Decimal (双精度)，参数的余弦值。虚数结果返回 Undefined。
Decimal	Decimal (双精度)，参数的余弦值。虚数结果返回 Undefined。
Boolean	Undefined.
String	Decimal (双精度)，参数的余弦值。如果字符串无法转换为 Decimal，则结果为 Undefined。虚数结果返回 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

cosh(Decimal)

以弧度为单位返回数字的双曲余弦值。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：cosh(2.3) = 5.037220649268761。

参数类型	结果
Int	Decimal (双精度)，参数的双曲余弦值。虚数结果返回 Undefined。

参数类型	结果
Decimal	Decimal (双精度)，参数的双曲余弦值。虚数结果返回 Undefined。
Boolean	Undefined.
String	Decimal (双精度)，参数的双曲余弦值。如果字符串无法转换为 Decimal，则结果为 Undefined。虚数结果返回 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

解码（值，decodingScheme）

使用 decode 函数来解码一个编码的值。如果解码字符串为 JSON 文档，则返回可寻址对象。否则，解码字符串会返回为字符串。如果字符串无法解码，则函数返回 NULL。此函数支持解码 base64 编码的字符串和协议缓冲区 (protobuf) 消息格式。

SQL 版本 2016-03-23 及更高版本支持。

值

字符串值或任何有效的表达式，如 Amazon IoT SQL 参考 (p. 530) 中所定义，将返回一个字符串。

decodingScheme

表示用于解码值的方案的文字字符串。目前仅支持'base64''proto'和。

解码 base64 编码字符串

在此示例中，消息负载包含一个编码值。

```
{
    encoded_temp: "eyAidGVtcGVyYXR1cmUiOiAzMyB9Cg=="
}
```

SQL 语句中的decode函数解码消息负载中的值。

```
SELECT decode(encoded_temp,"base64").temperature AS temp from 'topic/subtopic'
```

解码 encoded_temp 值会生成以下有效的 JSON 文档，该文档允许 SELECT 语句读取温度值。

```
{ "temperature": 33 }
```

此处显示了此示例中 SELECT 语句的结果。

```
{ "temp": 33 }
```

如果解码的值不是有效的 JSON 文档，则解码的值将作为字符串返回。

解码 protobuf 消息有效负载

您可以使用解码 SQL 函数来配置可以解码 protobuf 消息有效负载的规则。有关更多信息，请参阅解码 protobuf 消息有效负载 (p. 588)。

此函数签名类似以下内容：

```
decode(<ENCODED DATA>, 'proto', '<S3 BUCKET NAME>', '<S3 OBJECT KEY>', '<PROTO NAME>',
 '<MESSAGE TYPE>')
```

`ENCODED DATA`

指定要解码的 protobuf 编码数据。如果发送到规则的整个消息都是 protobuf 编码的数据，则可以使用引用原始的二进制传入负载*。否则，此字段必须是 base-64 编码的 JSON 字符串，并且可以直接传递对该字符串的引用。

1) 要解码原始二进制 protobuf 传入的有效负载：

```
decode(*, 'proto', ...)
```

2) 要解码由 base64 编码的字符串 'a.b' 表示的 protobuf 编码的消息：

```
decode(a.b, 'proto', ...)
```

`proto`

以 protobuf 消息格式指定要解码的数据。如果您指定 `base64` 而不是 `proto`，此函数会将 base64 编码的字符串解码为 JSON。

`S3 BUCKET NAME`

您上传 `FileDescriptorSet` 文件的 Amazon S3 存储桶的名称。

`S3 OBJECT KEY`

指定 Amazon S3 存储桶中 `FileDescriptorSet` 文件的对象键。

`PROTO NAME`

生成 `FileDescriptorSet` 文件的 `.proto` 文件名（不包括扩展名）。

`MESSAGE TYPE`

`FileDescriptorSet` 文件中 protobuf 消息结构的名称，要解码的数据应符合该结构。

使用解码 SQL函数的示例 SQL 表达式可能如下所示：

```
SELECT VALUE decode(*, 'proto', 's3-bucket', 'messageformat.desc', 'myproto',
 'messagetype') FROM 'some/topic'
```

- *

  表示二进制传入负载，它符合名为 `mymessagetype` 的 protobuf 消息类型。

- `messageformat.desc`

FileDescriptorSet 文件存储在名为 s3-bucket 的 Amazon S3 存储桶中。

- myproto

  用于生成名为 myproto.proto 的 FileDescriptorSet 文件的原始 .proto 文件。

- messagetype

  myproto.proto 中定义的名为 messagetype 的消息类型（以及任何导入的依赖项）。

encode(value, encodingScheme)

根据编码方案，使用 encode 函数将负载（可能是非 JSON 数据）编码为字符串表示形式。SQL 版本 2016-03-23 及更高版本支持。

值

Amazon IoT SQL 参考 (p. 530) 中所定义的任何有效的表达式。您可以指定 * 以对整个负载进行编码，无论它是否为 JSON 格式。如果您提供了表达式，评估的结果将在编码之前转换为字符串。

encodingScheme

代表您要使用的编码方案的文字字符串。目前仅支持 'base64'。

endswith(String, String)

返回 Boolean 来表示第一个 String 参数是否以第二个 String 参数结尾。如果任一参数为 Null 或 Undefined，则结果为 Undefined。SQL 版本 2015-10-08 及更高版本支持。

示例：endswith("cat","at") = true。

参数类型 1	参数类型 2	结果
String	String	如果第一个参数以第二 false。
其它值	其它值	两个参数都使用标准转 数以第二个参数结尾， 参数为 Null 或 Unde

exp(Decimal)

返回 e 的 Decimal 参数次方。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：exp(1) = e.

参数类型	结果
Int	Decimal (双精度)，e ^ 参数。
Decimal	Decimal (双精度)，e ^ 参数。
String	Decimal (双精度)，e ^ 参数。如果 String 无法转换为 Decimal，则结果为 Undefined。

参数类型	结果
其它值	Undefined.

floor(Decimal)

将给定的 Decimal 向下舍入到最接近的 Int。SQL 版本 2015-10-08 及更高版本支持。

示例：

floor(1.2) = 1

floor(-1.2) = -2

参数类型	结果
Int	Int，参数值。
Decimal	Int，Decimal 值会向下舍入至最接近的 Int。
String	Int。字符串将转换为 Decimal 并向下舍入到最接近的 Int。如果字符串无法转换为 Decimal，则结果为 Undefined。
其它值	Undefined.

get

从一个集合数据类型 (数组、字符串、对象) 中提取值。第一个参数不会进行任何转换。根据表中的记载对第二个参数进行转换。SQL 版本 2015-10-08 及更高版本支持。

示例：

get(["a", "b", "c"], 1) = "b"

get({"a":"b"}, "a") = "b"

get("abc", 1) = "a"

参数类型 1	参数类型 2	结果
数组	任何类型 (转换为 Int)	在第二个参数 (已经转 零开始索引得到的项目 Undefined。如果索 于数组长度)，则结果
字符串	任何类型 (转换为 Int)	在第二个参数 (已经转 零开始索引得到的字符 Undefined。如果索 字符串长度)，则结果
对象	String (不进行转换)	第一个参数对象中存储 键相对应。
其它值	任意值	Undefined.

get_dynamodb（tableName partitionKeyName、partitionKeyValue、、 sortKeyName、 sortKeyValue、roleArn）

从 DynamoDB 表中检索数据。get_dynamodb() 可让您在计算规则时查询 DynamoDB 表。您可以使用从 DynamoDB 中检索到的数据筛选或增加消息负载。SQL 版本 2016-03-23 及更高版本支持。

get_dynamodb() 接受以下参数：

tableName

要查询的 DynamoDB 表的名称。

partitionKeyName

分区键的名称。有关更多信息，请参阅 DynamoDB Keys。

partitionKeyValue

用于标识记录的分区键的值。有关更多信息，请参阅 DynamoDB Keys。

sortKeyName

（可选）排序键的名称。仅当查询的 DynamoDB 表使用复合键时需要此参数。有关更多信息，请参阅 DynamoDB Keys。

sortKeyValue

（可选）排序键的值。仅当查询的 DynamoDB 表使用复合键时需要此参数。有关更多信息，请参阅 DynamoDB Keys。

roleArn

授予对 DynamoDB 表的访问权限的 IAM 角色的 ARN。规则引擎将承担此角色以代表您访问 DynamoDB 表。避免使用过于宽容的角色。仅向角色授予规则所需的那些权限。以下是授予对一个 DynamoDB 表的访问权限的示例策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dynamodb:GetItem",
            "Resource": "arn:aws:dynamodb:aws-region:account-id:table/table-name"
        }
    ]
}}
```

举例说明您如何使用 get_dynamodb()，例如，您有一个 DynamoDB 表，其中包含连接到 Amazon IoT 的所有设备的设备 ID 和位置信息。以下 SELECT 语句使用 get_dynamodb() 函数检索指定的设备 ID 的位置：

```
SELECT *, get_dynamodb("InServiceDevices", "deviceId", id,
"arn:aws:iam::12345678910:role/getdynamo").location AS location FROM 'some/
topic'
```

Note

- 每个 SQL 语句最多可以调用 get_dynamodb() 一次。在单个 SQL 语句中调用 get_dynamodb() 多次会导致规则在不调用任何操作的情况下终止。
- 如果 get_dynamodb() 返回 8 KB 以上的数据，则无法调用规则的操作。

get_mqtt_property(name)

引用以下任何 MQTT5 标头：contentType、payLoadFormatIndicator、responseTopic 和 correlationData。此函数将以下任何文字字符串作为参数：content_type、format_indicator、response_topic 和 correlation_data。有关更多信息，请参阅下面的函数参数表。

contentType

字符串：一个 UTF-8 编码的字符串，用于描述发布消息的内容。

payLoadFormat指标

字符串：一个枚举字符串值，指示有效负载是否格式化为 UTF-8。有效值为 UNSPECIFIED_BYTES 和 UTF8_DATA。

responseTopic

字符串：一个 UTF-8 编码的字符串，用作响应消息的主题名称。响应主题用于描述接收方应作为请求-响应流程的一部分发布到的主题。主题不得包含通配符。

correlationData

字符串：请求消息的发送方使用 base64 编码的二进制数据来识别收到响应消息时响应消息的请求。

下表显示了可接受的函数参数，及其与 get_mqtt_property 函数相关的返回类型：

函数参数

SQL	返回的数据类型（如果存在）	返回的数据类型（如果不存在）
get_mqtt_property("format_indicator")	字符串（UNSPECIFIED_BYTES 或 UTF8_DATA）	字符串 (UNSPECIFIED_BYTES)
get_mqtt_property("content_type")	字符串	未定义
get_mqtt_property("response_topic")	字符串	未定义
get_mqtt_property("correlation_data")	base64 编码的字符串	未定义
get_mqtt_property("some_invalid_name")	未定义	未定义

以下示例规则 SQL 引用下列任何 MQTT5 标头：contentType、payLoadFormatIndicator、responseTopic 和 correlationData。

```
SELECT *, get_mqtt_property('content_type') as contentType,
          get_mqtt_property('format_indicator') as payloadFormatIndicator,
          get_mqtt_property('response_topic') as responseTopic,
          get_mqtt_property('correlation_data') as correlationData
FROM 'some/topic'
```

get_secret(secretId, secretType, key, roleArn)

在 Amazon Secrets Manager 中检索密钥当前版本的加密 SecretString 或者 SecretBinary字段中的值。有关创建和维护密钥的更多信息，请参阅CreateSecretUpdateSecret、和PutSecretValue。

get_secret() 接受以下参数：

secretId

字符串：要检索的密钥的 Amazon Resource Name (ARN) 或友好名称。

secretType

字符串：密钥类型。有效值：`SecretString | SecretBinary`。

SecretString

- 对于使用 API、Amazon CLI 或 Amazon Secrets Manager 控制台创建为 JSON 对象的密钥：
  - 如果为 `key` 参数指定值，则函数返回指定键的值。
  - 如果您没有为 `key` 参数指定值，则函数将返回整个 JSON 对象。
- 对于通过使用 API 或 Amazon CLI 创建的非 JSON 对象：
  - 如果为 `key` 参数指定值，则函数失败并出现异常。
  - 如果您没有为 `key` 参数指定值，则函数返回密钥的内容。

SecretBinary

- 如果为 `key` 参数指定值，则函数失败并出现异常。
- 如果您没有为 `key` 参数指定值，则函数将密钥值作为 base64 编码 UTF-8 字符串返回。

key

（可选）字符串：存储在密钥的 `SecretString` 字段中的 JSON 对象里的键名称。如果您希望仅检索存储在密钥中而不是整个 JSON 对象中的键值，请使用此值。

如果您为此参数指定了一个值，而密钥的 `SecretString` 字段中不包含 JSON 对象，此函数将失败并出现异常。

roleArn

String：具有 `secretsmanager:GetSecretValue` 和 `secretsmanager:DescribeSecret` 权限的角色 ARN。

Note

此函数始终返回密钥的当前版本（带有 `AWSCURRENT` 标签的版本）。Amazon IoT 规则引擎会将每个密钥缓存长达 15 分钟。因此，规则引擎最多可能需要 15 分钟才能更新密钥。这意味着，如果您在使用 Amazon Secrets Manager 更新后 15 分钟内检索密钥，此函数可能会返回旧版本。
此功能不是计量的，可以免费使用，但需支付 Amazon Secrets Manager 费用。由于密钥缓存机制，规则引擎偶尔会调用 Amazon Secrets Manager。由于规则引擎是完全分布的服务，因此您可能会在 15 分钟缓存窗口期间看到规则引擎中的多个 Secrets Manager API 调用。

示例：

您可以在 HTTPS 规则操作中的身份验证标头里使用 `get_secret` 函数，如下面的 API 密钥身份验证示例所示。

```
"API_KEY": "${get_secret('API_KEY', 'SecretString', 'API_KEY_VALUE',
 'arn:aws:iam::12345678910:role/getsecret')}"
```

有关 HTTPS 规则操作的更多信息，请参阅 the section called “HTTP” (p. 467)。

get_thing_shadow(thingName, shadowName, roleARN)

返回指定事物的指定影子。SQL 版本 2016-03-23 及更高版本支持。

thingName

String：您要检索其影子的事物的名称。

shadowName

（可选）字符串：影子的名称。只有在引用命名的影子时，才需要使用该参数。

roleArn

String：具有 `iot:GetThingShadow` 的 ARN 角色。

示例：

与命名的影子一起使用时，请提供 `shadowName` 参数。

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","MyThingShadow","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

与未命名的影子一起使用时，请省略 `shadowName` 参数。

```
SELECT * from 'topic/subtopic'
WHERE
    get_thing_shadow("MyThing","arn:aws:iam::123456789012:role/AllowsThingShadowAccess")
    .state.reported.alarm = 'ON'
```

获取用户属性 (userPropertyKey)

引用用户属性，这是 MQTT5 中支持的一种类型的属性标头。

userProperty

字符串：用户属性是一个键-值对。此函数将键作为参数，并返回与关联的键匹配的所有值的数组。

函数参数

对于邮件标题中的以下用户属性：

键	值
某个键	某个值
其他键	其他值
某个键	带有重复键的值

下表显示了预期的 SQL 行为：

SQL	返回的数据类型	返回的数据值
get_user_properties('some key')	字符串数组	`['some value', 'value with duplicate key']`

SQL	返回的数据类型	返回的数据值
get_user_properties('other key')	字符串数组	`['a different value']`
get_user_properties( )	键值对对象的数组	`[{'"some key": "some value"'}, {"other key": "a different value"}, {"some key": "value with duplicate key"}]`
get_user_properties('non-existent key')	未定义	

以下示例规则 SQL 将用户属性（一种类型的 MQTT5 属性标头）引用到有效负载中：

```
SELECT *, get_user_properties('user defined property key') as userProperty
FROM 'some/topic'
```

哈希函数

Amazon IoT 可以提供以下哈希函数：

- md2
- md5
- sha1
- sha224
- sha256
- sha384
- sha512

所有哈希函数都可以输入一个字符串参数。结果为该字符串的哈希值。对非字符串参数进行标准字符串转换。所有哈希函数在 SQL 版本 2015-10-08 及更高版本中均受支持。

示例：

`md2("hello")` = "a9046c73e00331af68917d3804f70655"

`md5("hello")` = "5d41402abc4b2a76b9719d911017c592"

indexof(String, String)

返回第二个参数的第一个索引 (从零开始) 作为第一个参数的子字符串。两个参数均为字符串。如果参数数据类型不是字符串，则应用标准字符串转换规则进行转换。此函数只对字符串有效，不适用于数组。SQL 版本 2016-03-23 及更高版本支持。

示例：

`indexof("abcd", "bc")` = 1

isNull()

如果参数为 `Null` 值，则返回 true。SQL 版本 2015-10-08 及更高版本支持。

示例：

`isNull(5)` = false.

`isNull(Null)` = true.

参数类型	结果
`Int`	false
`Decimal`	false
`Boolean`	false
`String`	false
`Array`	false
`Object`	false
`Null`	真实
`Undefined`	false

isUndefined()

如果参数为 `Undefined`，则返回 true。SQL 版本 2016-03-23 及更高版本支持。

示例：

`isUndefined(5)` = false.

`isUndefined(floor([1,2,3]))` = true.

参数类型	结果
`Int`	false
`Decimal`	false
`Boolean`	false
`String`	false
`Array`	false
`Object`	false
`Null`	false
`Undefined`	真实

length(String)

返回输入字符数中的字符数。对非 `String` 参数应用标准转换规则。SQL 版本 2016-03-23 及更高版本支持。

示例：

`length("hi")` = 2

`length(false)` = 5

ln(Decimal)

返回参数的自然对数。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：ln(e) = 1。

参数类型	结果
Int	Decimal (双精度)，参数的自然对数。
Decimal	Decimal (双精度)，参数的自然对数。
Boolean	Undefined.
String	Decimal (双精度)，参数的自然对数。如果字符串无法转换为 Decimal，则结果为 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

log(Decimal)

返回参数的以 10 为底的对数。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：log(100) = 2.0。

参数类型	结果
Int	Decimal (双精度)，参数以 10 为底的对数。
Decimal	Decimal (双精度)，参数以 10 为底的对数。
Boolean	Undefined.
String	Decimal (双精度)，参数以 10 为底的对数。如果 String 无法转换为 Decimal，则结果为 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

lower(String)

返回给定 String 的小写版本。非字符串参数使用标准转换规则转换为字符串。SQL 版本 2015-10-08 及更高版本支持。

示例：

lower("HELLO") = "hello"。

lower(["HELLO"]) = "[\"hello\"]"。

lpad(String, Int)

返回 String 参数，在输入参数的左侧填充由第二个参数指定的数量的空格。Int 参数必须介于 0 到 1000 之间。如果提供的值在这一有效范围之外，则参数被设置为与其最近的值（0 或 1000）。SQL 版本 2015-10-08 及更高版本支持。

示例：

lpad("hello", 2) = " hello".

lpad(1, 3) = " 1"

参数类型 1	参数类型 2	结果
String	Int	String，在输入 Str 格。
String	Decimal	Decimal 参数向下舍 填充指定数量的空格。
String	String	第二个参数被转换为 D 并在 String 左侧填充 法转换为 Int，则结果
其它值	Int/Decimal/String	第一个值使用标准转换 String 应用 LPAD 函 Undefined。
任意值	其它值	Undefined.

ltrim(String)

从提供的 String 中删除所有前导空白（制表符和空格）。SQL 版本 2015-10-08 及更高版本支持。

示例：

Ltrim(" h i ") = "hi "。

参数类型	结果
Int	删除了所有前导空白的 Int 的 String 表示形式。
Decimal	删除了所有前导空白的 Decimal 的 String 表示形式。
Boolean	布尔值（“true”或“false”）在删除所有前导空白之后的 String 表示形式。
String	删除所有前导空白之后的参数。
数组	Array（使用标准转换规则）在删除所有前导空白之后的 String 表示形式。

参数类型	结果
对象	对象（使用标准转换规则）在删除所有前导空白之后的 `String` 表示形式。
Null	`Undefined`.
未定义	`Undefined`.

machinelearning_predict(modelId, roleArn, record)

利用`machinelearning_predict`函数，根据 Amazon SageMaker 模型使用 MQTT 消息中的数据进行预测。SQL 版本 2015-10-08 及更高版本支持。`machinelearning_predict` 函数的参数如下：

modelId

对其运行预测的模型的 ID。必须启用模型的实时终端节点。

roleArn

IAM 角色，拥有具备 `machinelearning:Predict` 和 `machinelearning:GetMLModel` 权限的策略并允许访问运行预测所针对的模型。

record

将传递到 Pred SageMaker ict API 的数据。该参数应表示为单层 JSON 对象。如果记录是多级 JSON 对象，该记录将通过序列化其值来进行平展。例如，以下 JSON：

```
{ "key1": {"innerKey1": "value1"}, "key2": 0}
```

会变为：

```
{ "key1": "{\"innerKey1\": \"value1\"}", "key2": 0}
```

该函数返回具有以下字段的 JSON 对象：

predictedLabel

基于模型的输入分类。

details

包含以下属性：

PredictiveModelType

模型类型。有效值为 REGRESSION、BINARY、MULTICLASS。

算法

用于预测 SageMaker 的算法。该值必须为 SGD。

predictedScores

包含与每个标签对应的原始分类分数。

predictedValue

预测的值 SageMaker。

mod(Decimal, Decimal)

返回第一个参数除以第二个参数的余数。等效于 remainder(Decimal, Decimal) (p. 570)。您还可以使用“%”作为相同取模功能的插入运算符。SQL 版本 2015-10-08 及更高版本支持。

示例：`mod(8, 3)` = 2。

左侧操作数	右侧操作数	Output
`Int`	`Int`	`Int`，第一个参数对第
`Int/Decimal`	`Int/Decimal`	`Decimal`，第一个参数
`String/Int/Decimal`	`String/Int/Decimal`	如果所有字符串转换为参数取模的值。否则为
其它值	其它值	`Undefined`.

nanvl (AnyValue, AnyValue)

如果第一个参数为有效的 `Decimal`，则返回第一个参数。否则，返回第二个参数。SQL 版本 2015-10-08 及更高版本支持。

示例：`Nanvl(8, 3)` = 8。

参数类型 1	参数类型 2	Output
未定义	任意值	第二个参数。
Null	任意值	第二个参数。
`Decimal` (NaN)	任意值	第二个参数。
`Decimal` (非 NaN)	任意值	第一个参数。
其它值	任意值	第一个参数。

newuuid()

返回随机的 16 字节 UUID。SQL 版本 2015-10-08 及更高版本支持。

示例：`newuuid()` = `123a4567-b89c-12d3-e456-789012345000`

numbytes(String)

返回输入字符串 UTF-8 编码中的字节数。对非 `String` 参数应用标准转换规则。SQL 版本 2016-03-23 及更高版本支持。

示例：

`numbytes("hi")` = 2

`numbytes("€")` = 3

parse_time(String, Long[, String])

使用 parse_time 函数可将时间戳的格式设置为人类可读的日期/时间格式。SQL 版本 2016-03-23 及更高版本支持。要将时间戳字符串转换为毫秒，请参阅 time_to_epoch(String, String) (p. 577)。

parse_time 函数采用下列参数：

pattern

（字符串）遵循 Joda 时间格式的日期/时间模式。

timestamp

(Long) 要采用自 Unix 纪元时间以来的毫秒数格式表示的时间。请参阅函数 timestamp() (p. 577)。

timezone

（字符串）采用日期/时间格式的时区。默认值为“UTC”。此函数支持 Joda-Time 时区。此参数是可选的。

示例：

在将此消息发布到主题“A/B”时，负载 {"ts": "1970.01.01 AD at 21:46:40 CST"} 发送到 S3 存储桶：

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", 100000000, 'America/
Belize' ) as ts FROM 'A/B'",

        "ruleDisabled": false,
        "awsIotSqlVersion": "2016-03-23",
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

在将此消息发布到主题“A/B”时，与 {"ts": "2017.06.09 AD at 17:19:46 UTC"} 类似（但具有当前日期/时间）的负载发送到 S3 存储桶：

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT parse_time(\"yyyy.MM.dd G 'at' HH:mm:ss z\", timestamp() ) as ts
 FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "ruleDisabled": false,
        "actions": [
            {
                "s3": {
                    "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                    "bucketName": "BUCKET_NAME",
                    "key": "KEY_NAME"
                }
```

```
            }
        ],
        "ruleName": "RULE_NAME"
    }
}
```

parse_time() 也可用作替换模板。例如，在将此消息发布到主题“A/B”时，负载发送到密钥为“2017”的 S3 存储桶：

```
{
    "ruleArn": "arn:aws:iot:us-east-2:ACCOUNT_ID:rule/RULE_NAME",
    "topicRulePayload": {
        "sql": "SELECT * FROM 'A/B'",
        "awsIotSqlVersion": "2016-03-23",
        "ruleDisabled": false,
        "actions": [{
            "s3": {
                "roleArn": "arn:aws:iam::ACCOUNT_ID:rule:role/ROLE_NAME",
                "bucketName": "BUCKET_NAME",
                "key": "${parse_time('yyyy', timestamp(), 'UTC')}"
            }
        }],
        "ruleName": "RULE_NAME"
    }
}
```

power(Decimal, Decimal)

返回第一个参数的第二个参数次幂的值。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。SQL 版本 2015-10-08 及更高版本支持。

示例：power(2, 5) = 32.0。

参数类型 1	参数类型 2	Output
Int/Decimal	Int/Decimal	Decimal (双精度)，返值。
Int/Decimal/String	Int/Decimal/String	Decimal (双精度)，返值。所有字符串均转换为 Decimal，则结果
其它值	其它值	Undefined.

principal()

返回设备用于身份验证的委托人，取决于触发消息的发布方式。下表介绍了为每个发布方法和协议返回的委托人。

消息的发布方式	协议	凭证类型
MQTT 客户端	MQTT	X.509 设备证书
Amazon IoT 控制台 MQTT 客户端	MQTT	IAM 用户或角色
Amazon CLI	HTTP	IAM 用户或角色
Amazon IoT Device SDK	MQTT	X.509 设备证书

消息的发布方式	协议	凭证类型
Amazon IoT Device SDK	MQTT 结束了 WebSocket	IAM 用户或角色

下面的示例展示了 `principal()` 能够返回的不同类型的值：

- X.509 证书指纹：`ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373`
- IAM 角色 ID 和会话名称：`ABCD1EFG3HIJK2LMNOP5:my-session-name`
- 返回用户 ID：`ABCD1EFG3HIJK2LMNOP5`

rand()

返回在 0.0 到 1.0 之间均匀分布的伪随机双精度值。SQL 版本 2015-10-08 及更高版本支持。

示例：

`rand()` = 0.8231909191640703

regexp_matches(String, String)

如果字符串（第一个参数）包含正则表达式（第二个参数）的匹配项，则返回 true。

示例：

`regexp_matches("aaaa", "a{2,}")` = true.

`regexp_matches("aaaa", "b")` = false.

第一个参数：

参数类型	结果
`Int`	`String` 的 `Int` 表示形式。
`Decimal`	`String` 的 `Decimal` 表示形式。
`Boolean`	布尔值 ("true" 或 "false") 的 `String` 表示形式。
`String`	该 `String`。
数组	`Array` (使用标准转换规则) 的 `String` 表示形式。
对象	对象 (使用标准转换规则) 的 `String` 表示形式。
Null	`Undefined`.
未定义	`Undefined`.

第二个参数：

必须是有效的正则表达式。非字符串类型使用标准转换规则转换为 `String`。根据类型，生成的字符串可能不是有效的正则表达式。如果（转换后的）参数不是有效的正则表达式，则结果为 `Undefined`。

regexp_replace(String, String, String)

用第三个参数替换在第一个参数中出现的所有第二个参数 (正则表达式)。用“$”引用捕获组。SQL 版本 2015-10-08 及更高版本支持。

示例：

`regexp_replace("abcd", "bc", "x")` = "axd"。

`regexp_replace("abcd", "b(.*)d", "$1")` = "ac"。

第一个参数：

参数类型	结果
Int	String 的 Int 表示形式。
Decimal	String 的 Decimal 表示形式。
Boolean	布尔值 ("true" 或 "false") 的 String 表示形式。
String	源值。
数组	Array (使用标准转换规则) 的 String 表示形式。
对象	对象 (使用标准转换规则) 的 String 表示形式。
Null	Undefined.
未定义	Undefined.

第二个参数：

必须是有效的正则表达式。非字符串类型使用标准转换规则转换为 String。根据类型，生成的字符串可能不是有效的正则表达式。如果（转换后的）参数不是有效的正则表达式，则结果为 Undefined。

第三个参数：

必须是有效的正则表达式替代字符串。（可以引用捕获组。）非字符串类型使用标准转换规则转换为 String。如果（转换后的）参数不是有效的正则表达式替代字符串，则结果为 Undefined。

regexp_substr(String, String)

在第一个参数中查找第二个参数（正则表达式）的第一个匹配项。用“$”引用捕获组。SQL 版本 2015-10-08 及更高版本支持。

示例：

`regexp_substr("hihihello", "hi")` = "hi"

`regexp_substr("hihihello", "(hi)*")` = "hihi"

第一个参数：

参数类型	结果
Int	String 的 Int 表示形式。
Decimal	String 的 Decimal 表示形式。
Boolean	布尔值 ("true" 或 "false") 的 String 表示形式。
String	String 参数。
数组	Array (使用标准转换规则) 的 String 表示形式。

参数类型	结果
对象	对象 (使用标准转换规则) 的 String 表示形式。
Null	Undefined.
未定义	Undefined.

第二个参数：

必须是有效的正则表达式。非字符串类型使用标准转换规则转换为 String。根据类型，生成的字符串可能不是有效的正则表达式。如果（转换后的）参数不是有效的正则表达式，则结果为 Undefined。

remainder(Decimal, Decimal)

返回第一个参数除以第二个参数的余数。等效于 mod(Decimal, Decimal) (p. 565)。您还可以使用“%”作为相同取模功能的插入运算符。SQL 版本 2015-10-08 及更高版本支持。

示例：remainder(8, 3) = 2。

左侧操作数	右侧操作数	Output
Int	Int	Int，第一个参数对第
Int/Decimal	Int/Decimal	Decimal，第一个参数
String/Int/Decimal	String/Int/Decimal	如果所有字符串转换为参数取模的值。否则为
其它值	其它值	Undefined.

replace(String, String, String)

用第三个参数替换在第一个参数中出现的所有第二个参数。SQL 版本 2015-10-08 及更高版本支持。

示例：

replace("abcd", "bc", "x") = "axd".

replace("abcdabcd", "b", "x") = "axcdaxcd".

所有参数

参数类型	结果
Int	String 的 Int 表示形式。
Decimal	String 的 Decimal 表示形式。
Boolean	布尔值 ("true" 或 "false") 的 String 表示形式。
String	源值。
数组	Array (使用标准转换规则) 的 String 表示形式。
对象	对象 (使用标准转换规则) 的 String 表示形式。
Null	Undefined.

参数类型	结果
未定义	Undefined.

rpad(String, Int)

返回字符串参数，在输入参数的右侧填充在第二个参数中指定的数量的空格。Int 参数必须介于 0 到 1000 之间。如果提供的值在这一有效范围之外，则参数被设置为与其最近的值（0 或 1000）。SQL 版本 2015-10-08 及更高版本支持。

示例：

`rpad("hello", 2) = "hello  ".`

`rpad(1, 3) = "1  ".`

参数类型 1	参数类型 2	结果
String	Int	在 String 的右侧填充由 Int 指定数量的空格。
String	Decimal	Decimal 参数向下舍入到最近的 Int，并且在字符串的右侧填充由 Int 指定数量的空格。
String	String	第二个参数转换为 Decimal，并向下舍入到最近的 Int。在 String 的右侧填充由 Int 值指定数

参数类型 1	参数类型 2	结果
		量的空格。
其它值	Int/Decimal/String	第一个值将使用标准转换规则转换为 String，然后对该 String 应用 rpad 函数。如果它无法转换，则结果为 Undefined。
任意值	其它值	Undefined.

round(Decimal)

将给定的 Decimal 舍入到最近的 Int。如果 Decimal 与上下两个 Int 值距离相同 (例如 0.5)，Decimal 将向上进位。SQL 版本 2015-10-08 及更高版本支持。

示例：Round(1.2) = 1。

Round(1.5) = 2.

Round(1.7) = 2.

Round(-1.1) = -1.

Round(-1.5) = -2.

参数类型	结果
Int	参数。
Decimal	Decimal 会向下舍入至最近的 Int。
String	Decimal 会向下舍入至最近的 Int。如果字符串无法转换为 Decimal，则结果为 Undefined。
其它值	Undefined.

rtrim(String)

从提供的 String 中删除所有尾随空白（制表符和空格）。SQL 版本 2015-10-08 及更高版本支持。

示例：

`rtrim(" h i ") = "h i"`

参数类型	结果
Int	String 的 Int 表示形式。
Decimal	String 的 Decimal 表示形式。
Boolean	布尔值 ("true" 或 "false") 的 String 表示形式。
数组	Array (使用标准转换规则) 的 String 表示形式。
对象	对象 (使用标准转换规则) 的 String 表示形式。
Null	Undefined.
未定义	Undefined

sign(Decimal)

返回给定数字的符号。当参数的符号为正时，将返回 1。当参数的符号为负时，将返回 -1。如果参数为 0，则返回 0。SQL 版本 2015-10-08 及更高版本支持。

示例：

`sign(-7) = -1.`

`sign(0) = 0。`

`sign(13) = 1.`

参数类型	结果
Int	Int，Int 值的符号。
Decimal	Int，Decimal 值的符号。
String	Int，Decimal 值的符号。字符串将转换为 Decimal 值，并返回 Decimal 值的符号。如果 String 无法转换为 Decimal，则结果为 Undefined。SQL 版本 2015-10-08 及更高版本支持。
其它值	Undefined.

sin(Decimal)

以弧度为单位返回数字的正弦值。在代入函数之前，Decimal 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：`sin(0) = 0.0`

参数类型	结果
Int	Decimal (双精度)，参数的正弦值。
Decimal	Decimal (双精度)，参数的正弦值。

参数类型	结果
Boolean	Undefined.
String	Decimal (双精度)，参数的正弦值。如果字符串无法转换为 Decimal，则结果为 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
Undefined	Undefined.

sinh(Decimal)

以弧度为单位返回数字的双曲正弦值。在代入函数之前，Decimal 值舍入到双精度。结果是双精度的 Decimal 值。SQL 版本 2015-10-08 及更高版本支持。

示例：sinh(2.3) = 4.936961805545957

参数类型	结果
Int	Decimal (双精度)，参数的双曲正弦值。
Decimal	Decimal (双精度)，参数的双曲正弦值。
Boolean	Undefined.
String	Decimal (双精度)，参数的双曲正弦值。如果字符串无法转换为 Decimal，则结果为 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

substring(String, Int[, Int])

输入值为 String 后跟一个或两个 Int 值。对于 String 和单个 Int 参数，此函数在输入的 String 中从指定的 Int 索引 (从零开始，包括零) 到 String 结束提取子字符串并返回。对于 String 和两个 Int 参数，此函数在输入的 String 中从第一个 Int 索引参数 (从零开始，包括零) 到第二个 Int 索引参数 (从零开始，包括零) 提取子字符串并返回。索引小于零时将设置为零。大于 String 长度的索引设置为 String 长度。在三个参数的版本中，如果第一个索引大于等于第二个索引，那么结果为空 String。

如果提供的参数不是 (*String*、*Int*) 或 (*String*、*Int*、*Int*)，则系统会对参数进行标准转换，尽量将其转换为正确的类型。如果无法转换类型，函数的结果为 Undefined。SQL 版本 2015-10-08 及更高版本支持。

示例：

substring("012345", 0) = "012345"。

`substring("012345", 2)` = "2345"。

`substring("012345", 2.745)` = "2345"。

`substring(123, 2)` = "3".

`substring("012345", -1)` = "012345"。

`substring(true, 1.2)` = "rue"。

`substring(false, -2.411E247)` = "false".

`substring("012345", 1, 3)` = "12".

`substring("012345", -50, 50)` = "012345"。

`substring("012345", 3, 1)` = "".

sql_version()

返回此规则中指定的 SQL 版本。SQL 版本 2015-10-08 及更高版本支持。

示例：

`sql_version()` = "2016-03-23"

sqrt(Decimal)

返回数字的平方根。在代入函数之前，`Decimal` 参数舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：`sqrt(9)` = 3.0。

参数类型	结果
`Int`	参数的平方根。
`Decimal`	参数的平方根。
`Boolean`	`Undefined`.
`String`	参数的平方根。如果字符串无法转换为 `Decimal`，则结果为 `Undefined`。
数组	`Undefined`.
对象	`Undefined`.
Null	`Undefined`.
未定义	`Undefined`.

startswith(String, String)

返回显示第一个字符串参数是否以第二个字符串参数开头的 `Boolean`。如果任一参数为 `Null` 或 `Undefined`，则结果为 `Undefined`。SQL 版本 2015-10-08 及更高版本支持。

示例：

`startswith("ranger","ran")` = true

参数类型 1	参数类型 2	结果
String	String	第一个字符串是否以第
其它值	其它值	两个参数都使用标准转 符串以第二个字符串开 Null 或 Undefined

tan(Decimal)

以弧度为单位返回数字的正切值。在代入函数之前，Decimal 值舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：tan(3) = -0.1425465430742778

参数类型	结果
Int	Decimal (双精度)，参数的正切值。
Decimal	Decimal (双精度)，参数的正切值。
Boolean	Undefined.
String	Decimal (双精度)，参数的正切值。如果字符串无法转换为 Decimal，则结果为 Undefined。
数组	Undefined.
对象	Undefined.
Null	Undefined.
未定义	Undefined.

tanh(Decimal)

以弧度为单位返回数字的双曲正切值。在代入函数之前，Decimal 值舍入到双精度。SQL 版本 2015-10-08 及更高版本支持。

示例：tanh(2.3) = 0.9800963962661914

参数类型	结果
Int	Decimal (双精度)，参数的双曲正切值。
Decimal	Decimal (双精度)，参数的双曲正切值。
Boolean	Undefined.
String	Decimal (双精度)，参数的双曲正切值。如果字符串无法转换为 Decimal，则结果为 Undefined。
数组	Undefined.
对象	Undefined.

参数类型	结果
Null	Undefined.
未定义	Undefined.

time_to_epoch(String, String)

使用 time_to_epoch 函数将时间戳字符串转换为 Unix 纪元时间的毫秒数。SQL 版本 2016-03-23 及更高版本支持。要将毫秒转换为格式化的时间戳字符串，请参阅 parse_time(String, Long[, String]) (p. 566)。

time_to_epoch 函数采用下列参数：

timestamp

（字符串）要转换为自 Unix 纪元以来毫秒数的时间戳字符串。如果时间戳字符串未指定时区，则函数将使用 UTC 时区。

pattern

（字符串）遵循 JDK11 时间格式的日期/时间模式。

示例：

```
time_to_epoch("2020-04-03 09:45:18 UTC+01:00", "yyyy-MM-dd HH:mm:ss VV") = 1585903518000
```

```
time_to_epoch("18 December 2015", "dd MMMM yyyy") = 1450396800000
```

```
time_to_epoch("2007-12-03 10:15:30.592 America/Los_Angeles", "yyyy-MM-dd HH:mm:ss.SSS z") = 1196705730592
```

timestamp()

返回 Amazon IoT 规则引擎观察到的当前时间戳，用距离 1970 年 1 月 1 日（星期四）协调世界时 (UTC) 00:00:00 的毫秒数来表示。SQL 版本 2015-10-08 及更高版本支持。

示例：`timestamp() = 1481825251155`

topic(Decimal)

返回已向其发送触发规则的消息的主题。如果未指定参数，则返回整个主题。Decimal 参数用于指定特定主题段，使用 1 指定第一个段。对于主题 foo/bar/baz，主题 (1) 将返回 foo，主题 (2) 将返回 bar，以此类推。SQL 版本 2015-10-08 及更高版本支持。

示例：

```
topic() = "things/myThings/thingOne"
```

```
topic(1) = "things"
```

在使用基本提取 (p. 529)功能时，主题的初始前缀 ($aws/rules/*rule-name*) 对 topic() 函数不可用。例如，给定以下主题：

```
$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights
```

```
topic() = "Buildings/Building5/Floor2/Room201/Lights"
```

```
topic(3) = "Floor2"
```

traceid()

返回 MQTT 消息的跟踪 ID (UUID)，如果未通过 MQTT 发送消息，则返回 Undefined。SQL 版本 2015-10-08 及更高版本支持。

示例：

`traceid()` = "12345678-1234-1234-1234-123456789012"

transform(String, Object, Array)

返回对象数组，其中包含 Object 参数对 Array 参数的指定转换结果。

SQL 版本 2016-03-23 及更高版本支持。

字符串

要使用的转换模式。有关支持的转换模式以及这些模式如何利用 Object 和 Array 参数创建 Result 的信息，请参阅下表。

对象

一个对象，其中包含要应用于 Array 的各个元素的属性。

数组

对象的数组，其中将应用 Object 的属性。

此数组中的每个对象都对应于函数响应中的一个对象。函数响应中的每个对象都包含原始对象中存在的属性以及 Object 提供的属性，这些属性由 String 中指定的转换模式决定。

String 参数	**Object** 参数	**Array** 参数	结果
enrichArray	对象	对象数组	一个对象的数组，其中每个对象都包含Array参数中元素的属性和 Object 参数的属性。
其它任何值	任意值	任意值	未定义

Note

此函数返回的数组限制为 128 KiB。

变换函数示例 1

此示例演示如何使用 transform() 函数从数据对象和数组中生成单个对象数组。

在此示例中，将以下消息将发布到 MQTT 主题 A/B。

```
{
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
```

```
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

此用于主题规则操作的 SQL 语句使用 `String` 值为 `enrichArray` 的 transform() 函数。在本例中，`Object` 是来自消息负载的 `attributes` 属性，而 `Array` 是 `values` 数组，其中包含三个对象。

```
select value transform("enrichArray", attributes, values) from 'A/B'
```

在收到消息负载后，SQL 语句将计算以下响应。

```
[
  {
    "a": 3,
    "data1": 1,
    "data2": 2
  },
  {
    "b": 4,
    "data1": 1,
    "data2": 2
  },
  {
    "c": 5,
    "data1": 1,
    "data2": 2
  }
]
```

变换函数示例 2

此示例演示 transform() 函数如何使用文本值来包含和重命名消息负载中的单个属性。

在此示例中，将以下消息将发布到 MQTT 主题 `A/B`。此消息与 the section called “变换函数示例 1” (p. 578) 中使用的消息相同。

```
{
    "attributes": {
        "data1": 1,
        "data2": 2
    },
    "values": [
        {
            "a": 3
        },
        {
            "b": 4
        },
        {
            "c": 5
        }
    ]
}
```

此用于主题规则操作的 SQL 语句使用 `String` 值为 `enrichArray` 的 transform() 函数。transform() 函数中的 `Object` 在消息负载中具有一个名为 `key` 的单一属性，该属性的值为 `attributes.data1`，而 `Array` 是 `values` 数组，其中包含与上一个示例中使用的三个相同对象。

```
select value transform("enrichArray", {"key": attributes.data1}, values) from 'A/B'
```

收到消息负载后，此 SQL 语句将计算为以下响应。请注意 data1 属性在 key 响应中是如何命名的。

```
[
  {
    "a": 3,
    "key": 1
  },
  {
    "b": 4,
    "key": 1
  },
  {
    "c": 5,
    "key": 1
  }
]
```

变换函数示例 3

此示例演示如何使用 transform() 函数可在嵌套 SELECT 子句中选择多个属性并创建新对象以供后续处理。

在此示例中，将以下消息将发布到 MQTT 主题 A/B。

```
{
  "data1": "example",
  "data2": {
    "a": "first attribute",
    "b": "second attribute",
    "c": [
      {
        "x": {
          "someInt": 5,
          "someString": "hello"
        },
        "y": true
      },
      {
        "x": {
          "someInt": 10,
          "someString": "world"
        },
        "y": false
      }
    ]
  }
}
```

此转换函数的 Object 是 SELECT 语句返回的对象，其中包含消息的 data2 对象的 a 和 b 元素。Array 参数包括两个来自原始消息中 data2.c 数组的对象。

```
select value transform('enrichArray', (select a, b from data2), (select value c from
 data2)) from 'A/B'
```

借助前面的消息，SQL 语句将计算以下响应。

```
[
  {
    "x": {
```

```
      "someInt": 5,
      "someString": "hello"
    },
    "y": true,
    "a": "first attribute",
    "b": "second attribute"
  },
  {
    "x": {
      "someInt": 10,
      "someString": "world"
    },
    "y": false,
    "a": "first attribute",
    "b": "second attribute"
  }
]
```

此响应中返回的数组可以用于支持 batchMode 的主题规则操作。

trim(String)

从提供的 String 中删除所有前导空白和尾随空白。SQL 版本 2015-10-08 及更高版本支持。

示例：

Trim(" hi ") = "hi"

参数类型	结果
Int	Int 在删除所有前导和尾随空白之后的 String 表示形式。
Decimal	Decimal 在删除所有前导和尾随空白之后的 String 表示形式。
Boolean	Boolean（“true”或“false”）在删除所有前导和尾随空白之后的 String 表示形式。
String	删除所有前导和尾随空白之后的 String。
数组	Array 使用标准转换规则进行转换后的 String 表示形式。
对象	对象使用标准转换规则进行转换后的 String 表示形式。
Null	Undefined.
未定义	Undefined.

trunc(Decimal, Int)

按照第二个参数指定的 Decimal 位数截断第一个参数。如果第二个参数小于零，则它会设置为零。如果第二个参数大于 34，则它会设置为 34。将从结果中删除结尾的零。SQL 版本 2015-10-08 及更高版本支持。

示例：

trunc(2.3, 0) = 2.

trunc(2.3123, 2) = 2.31。

trunc(2.888, 2) = 2.88。

trunc(2.00, 5) = 2.

参数类型 1	参数类型 2	结果
Int	Int	源值。
Int/Decimal	Int/Decimal	第一个参数被截断到由数如果不是 Int，将向
Int/Decimal/String	Int/Decimal	第一个参数被截断到由参数如果不是 Int，将转换为 Decimal 值。Undefined。
其它值		Undefined.

upper(String)

返回给定 String 的大写版本。非 String 参数将使用标准转换规则转换为 String。SQL 版本 2015-10-08 及更高版本支持。

示例：

upper("hello") = "HELLO"

upper(["hello"]) = "[\"HELLO\"]"

文本

您可以直接在规则 SQL 的 SELECT 和 WHERE 子句中指定文本对象，该对象可用于传递信息。

> Note
>
> 文本只能在使用 SQL 版本 2016-03-23 或更高版本时使用。

使用 JSON 对象语法 (键值对、逗号分隔、键为字符串/值为 JSON 值、括在大括号 {} 中)。例如：

传入负载已发布至主题 topic/subtopic：{"lat_long": [47.606,-122.332]}

SQL 语句：SELECT {'latitude': get(lat_long, 0),'longitude':get(lat_long, 1)} as lat_long FROM 'topic/subtopic'

生成的传出负载为：{"lat_long":{"latitude":47.606,"longitude":-122.332}}。

您也可以在规则 SQL 的 SELECT 和 WHERE 子句中直接指定数组，用于分组信息。使用 JSON 语法 (在方括号 [] 中用逗号分隔项目来创建数组文本)。例如：

传入负载已发布至主题 topic/subtopic：{"lat": 47.696, "long": -122.332}

SQL 语句：SELECT [lat,long] as lat_long FROM 'topic/subtopic'

生成的传出负载为：{"lat_long": [47.606,-122.332]}。

Case 语句

Case 语句可以用于执行分支，就像switch 语句一样。

语法：

```
CASE v WHEN t[1] THEN r[1]
  WHEN t[2] THEN r[2] ...
  WHEN t[n] THEN r[n]
  ELSE r[e] END
```

评估表达式 *v*，并与每个 WHEN 子句的 *t[i]* 值进行相等匹配。如果找到匹配，相应的 *r[i]* 表达式会成为 CASE 语句的结果。评估 WHEN 子句，这样，如果有多个匹配子句，第一个匹配子句的结果成为 CASE 语句的结果。如果没有匹配，ELSE 子句的 *r[e]* 就是结果。如果没有匹配项且没有 ELSE 子句，结果是 Undefined。

CASE 语句至少需要一个 WHEN 子句。ELSE 子句是可选的。

例如：

传入负载已发布至主题 topic/subtopic：

```
{
    "color":"yellow"
}
```

SQL 语句

```
SELECT CASE color
        WHEN 'green' THEN 'go'
        WHEN 'yellow' THEN 'caution'
        WHEN 'red' THEN 'stop'
        ELSE 'you are not at a stop light' END as instructions
    FROM 'topic/subtopic'
```

生成的传出负载为：

```
{
    "instructions":"caution"
}
```

> **Note**
>
> 如果 *v* 是 Undefined，则 Case 语句的结果为 Undefined。

JSON 扩展

您可以使用以下 ANSI SQL 语法扩展，更加轻松地使用嵌套 JSON 对象。

"." 运算符

此运算符访问嵌入式 JSON 对象和函数中的成员与 ANSI SQL 和相同 JavaScript。例如：

```
SELECT foo.bar AS bar.baz FROM 'topic/subtopic'
```

从以下发送至 topic/subtopic 主题的消息负载中选择 foo 对象中的 bar 属性。

```
{
  "foo": {
    "bar": "RED",
    "bar1": "GREEN",
    "bar2": "BLUE"
  }
```

```
}
```

如果 JSON 属性名称包含连字符或数字字符，则简单的“点”符号将不起作用。相反，您必须使用 get 函数 (p. 555)来提取属性的值。

在此示例中，以下消息将发送至 iot/rules 主题。

```
{
  "mydata": {
    "item2": {
      "0": {
        "my-key": "myValue"
      }
    }
  }
}
```

通常情况下，my-key 的值将被标识为在此查询中。

```
SELECT * from iot/rules WHERE mydata.item2.0.my-key= "myValue"
```

但是，由于属性名称 my-key 包含连字符，而 item2 包含一个数字字符，因此 get 函数 (p. 555)必须按以下查询所示的方法使用。

```
SELECT * from 'iot/rules' WHERE get(get(get(mydata,"item2"),"0"),"my-key") = "myValue"
```

* 运算符

该运算符与 ANSI SQL 中的 * 通配符的运作方式相同。该运算符仅用于 SELECT 子句，并会创建包含消息数据的全新 JSON 对象。如果消息负载不是 JSON 格式，* 将以原始字节形式返回整个消息负载。例如：

```
SELECT * FROM 'topic/subtopic'
```

将函数应用到属性值

下面显示了一个可能由设备发布的 JSON 负载示例：

```
{
    "deviceid" : "iot123",
    "temp" : 54.98,
    "humidity" : 32.43,
    "coords" : {
        "latitude" : 47.615694,
        "longitude" : -122.3359976
    }
}
```

下面的示例将函数应用到 JSON 负载中的一个属性值：

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

此查询的结果为以下 JSON 对象：

```
{
   "temp": 54.98,
   "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1"
}
```

替换模板

您可以使用替换模板来补充规则被触发并且 Amazon IoT 执行操作后返回的 JSON 数据。替换模板的语法是 ${expression}，其中 expression 可以是 SELECT 子句、WHERE 子句中的 Amazon IoT 以及 Amazon IoT 规则操作 (p. 449) 支持的任意表达式。此表达式可插入到规则的操作字段中，使您可以动态配置操作。在生效时，此功能替换操作中的一段信息。这包括函数、运算符和原始消息负载中存在的信息。

> **Important**
>
> 因为替换模板中的表达式与“SELECT ...”语句分开评估，所以，您无法引用使用 AS 子句创建的别名。您只能引用原始负载中存在的信息和函数 (p. 541)以及操作员 (p. 536)。

有关支持的表达式的更多信息，请参阅Amazon IoT SQL 参考 (p. 530)。

以下规则操作支持替换模板。每个操作都支持不同的可替换字段。

- Apache Kafka (p. 450)
- CloudWatch 警报 (p. 457)
- CloudWatch 日志 (p. 458)
- CloudWatch 指标 (p. 460)
- DynamoDB (p. 461)
- DynamoDBv2 (p. 463)
- Elasticsearch (p. 465)
- HTTP (p. 467)
- Kinesis Data Streams (p. 502)
- Kinesis Data Firehose (p. 500)
- Lambda (p. 503)
- 位置 (p. 505)
- OpenSearch (p. 508)
- Republish (p. 509)
- S3 (p. 511)
- SNS (p. 514)
- SQS (p. 515)
- Step Functions (p. 517)
- Timestream (p. 518)

替换模板显示在规则内的操作参数中：

```
{
    "sql": "SELECT *, timestamp() AS timestamp FROM 'my/iot/topic'",
    "ruleDisabled": false,
    "actions": [{
        "republish": {
            "topic": "${topic()}/republish",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

如果该规则由发布到 `my/iot/topic` 的以下 JSON 触发：

```
{
    "deviceid": "iot123",
```

```
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    }
}
```

然后，此规则将以下 JSON 发布到 `my/iot/topic/republish`，Amazon IoT 使用它来替代 `${topic()}/republish`：

```
{
    "deviceid": "iot123",
    "temp": 54.98,
    "humidity": 32.43,
    "coords": {
        "latitude": 47.615694,
        "longitude": -122.3359976
    },
    "timestamp": 1579637878451
}
```

嵌套对象查询

您可以使用嵌套 SELECT 子句来查询数组和内部 JSON 对象中的属性。SQL 版本 2016-03-23 及更高版本支持。

请考虑以下 MQTT 消息：

```
{
    "e": [
        { "n": "temperature", "u": "Cel", "t": 1234, "v": 22.5 },
        { "n": "light", "u": "lm", "t": 1235, "v": 135 },
        { "n": "acidity", "u": "pH", "t": 1235, "v": 7 }
    ]
}
```

Example

您可以使用以下规则将值转换为新数组。

```
SELECT (SELECT VALUE n FROM e) as sensors FROM 'my/topic'
```

该规则生成以下输出。

```
{
    "sensors": [
        "temperature",
        "light",
        "acidity"
    ]
}
```

Example

通过使用相同的 MQTT 消息，您还可以使用以下规则查询嵌套对象中的特定值。

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

该规则生成以下输出。

```
{
    "temperature": [
        {
            "v": 22.5
        }
    ]
}
```

Example

您也可以使用更复杂的规则来平展输出。

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'), 0).v as temperature FROM 'topic'
```

该规则生成以下输出。

```
{
    "temperature": 22.5
}
```

使用二进制负载

如果您发送原始二进制负载，Amazon IoT Core 通过 S3 操作将其向下游路由到 Amazon S3 存储桶。然后，原始二进制负载编码为 base64 并附加到 JSON。当将消息负载作为原始二进制数据（而不是 JSON 对象）进行处理时，可以使用 * 运算符在 SELECT 子句中对其进行引用。

本主题内容：

- 二进制负载示例 (p. 587)
- 解码 protobuf 消息有效负载 (p. 588)

二进制负载示例

当您使用 * 将消息负载作为原始二进制数据进行引用时，您可以向规则添加数据。如果您有空的或 JSON 有效负载，生成的有效负载可以使用规则添加数据。下面显示了支持 SELECT 子句的示例。

- 对于二进制有效负载，您可以将以下 SELECT 子句仅与 * 一起使用。
  - ```
    SELECT * FROM 'topic/subtopic'
    ```
  - ```
    SELECT * FROM 'topic/subtopic' WHERE timestamp() % 12 = 0
    ```
- 您还可以添加数据并使用以下 SELECT 子句。
  - ```
    SELECT *, principal() as principal, timestamp() as time FROM 'topic/subtopic'
    ```
  - ```
    SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'topic/subtopic'
    ```
- 您还可以使用带二进制负载的 SELECT 子句。
  - 以下是指在 WHERE 子句中的 device_type。

    ```
    SELECT * FROM 'topic/subtopic' WHERE device_type = 'thermostat'
    ```

- 还支持以下内容。

```
{
 "sql": "SELECT * FROM 'topic/subtopic'",
 "actions": [
  {
   "republish": {
    "topic": "device/${device_id}"
   }
  }
 ]
}
```

以下规则操作不支持二进制负载，因此您必须对它们进行解码。

- 一些规则操作不支持二进制负载输入（例如，Lambda 操作），您必须解码二进制负载。如果 Lambda 规则操作是 base64 编码并在 JSON 负载中，则可以接收二进制数据。为此，您可以将规则更改如下：

```
SELECT encode(*, 'base64') AS data FROM 'my_topic'
```

- SQL 语句不支持字符串作为输入。要将字符串输入转换为 JSON，您可以运行以下命令。

```
SELECT decode(encode(*, 'base64'), 'base64') AS payload FROM 'topic'
```

解码 protobuf 消息有效负载

协议缓冲区 (protobuf) 是一种开源数据格式，用于以紧凑的二进制形式序列化结构化数据。它用于通过网络传输数据或将数据存储在文件中。Protobuf 允许您以较小的数据包大小发送数据，并且速度比其他消息传递格式更快。Amazon IoT Core规则通过提供 decode(value, decodingScheme) (p. 552) SQL 函数来支持 protobuf，该函数允许您将 protobuf 编码的消息有效负载解码为 JSON 格式，并将它们路由到下游服务。本节详细介绍在Amazon IoT Core规则中配置 protobuf 解码的 step-by-step 过程。

本节内容：

- 先决条件 (p. 588)
- 创建描述符文件 (p. 588)
- 将描述符文件上传到 S3 存储桶 (p. 589)
- 在规则中配置 protobuf 解码 (p. 590)
- 限制 (p. 590)
- 最佳实践 (p. 591)

先决条件

- 对协议缓冲区 (protobuf) 有基本的了解
- 定义消息类型和相关依赖项的 `.proto` 文件
- 在您的系统上安装 Protobuf 编译器 (protoc)

创建描述符文件

如果您已有描述符文件，可以跳过此步骤。描述符文件 (`.desc`) 是 `.proto` 文件的编译版本，它是一个文本文件，用于定义 protobuf 序列化中使用的数据结构和消息类型。要生成描述符文件，必须定义一个 `.proto` 文件，并使用 protoc 编译器对其进行编译。

1. 创建定义消息类型的 .proto 文件。示例 .proto 文件可能如下所示：

```
syntax = "proto3";

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;
}
```

在此示例 .proto 文件中，使用 proto3 语法并定义消息类型 Person。Person 消息定义指定三个字段（名称、ID 和电子邮件）。有关 .proto 文件消息格式的更多信息，请参阅语言指南 (proto3)。

2. 使用 protoc 编译器编译 .proto 文件并生成一个描述符文件。创建描述符 (.desc) 文件的示例命令如下所示：

```
protoc --descriptor_set_out=<FILENAME>.desc \
    --proto_path=<PATH_TO_IMPORTS_DIRECTORY> \
    --include_imports \
    <PROTO_FILENAME>.proto
```

此示例命令会生成一个描述符文件 <FILENAME>.desc，Amazon IoT Core 规则可以使用该文件来解码符合 <PROTO_FILENAME>.proto 中定义的数据结构的 protobuf 有效负载。

- --descriptor_set_out

  指定应生成的描述符文件 (<FILENAME>.desc) 的名称。
- --proto_path

  指定正在编译的文件所引用的任何导入的 .proto 文件的位置。如果您有多个位于不同位置的已导入 .proto 文件，可以多次指定此标志。
- --include_imports

  指定还应编译任何已导入的 .proto 文件，并将其包含在 <FILENAME>.desc 描述符文件中。
- <PROTO_FILENAME>.proto

  指定要编译的 .proto 文件的名称。

有关 protoc 参考的更多信息，请参阅 API 参考。

将描述符文件上传到 S3 存储桶

创建描述符文件 <FILENAME>.desc 后，使用 Amazon API、Amazon SDK 或 Amazon Web Services Management Console 将描述符文件 <FILENAME>.desc 上传到 Amazon S3 存储桶。

重要注意事项

- 确保将描述符文件上传到您打算配置规则的同一 Amazon 区域中您的 Amazon Web Services 账户中的 Amazon S3 存储桶。
- 确保您授予从 S3 读取 FileDescriptorSet 的 Amazon IoT Core 访问权限。如果您的 S3 存储桶已禁用服务器端加密 (SSE)，或者 S3 存储桶已使用 Amazon S3 托管密钥 (SSE-S3) 加密，则不需要进行额外的策略配置。这可以通过示例存储桶策略来实现：

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

```
   "Sid": "Statement1",
   "Effect": "Allow",
   "Principal": {
    "Service": "iot.amazonaws.com"
   },
   "Action": "s3:Get*",
                      "Resource": "arn:aws:s3:::<BUCKET NAME>/<FILENAME>.desc"
  }
 ]
}
```

- 如果您的 S3 存储桶是使用 Amazon Key Management Service 密钥 (SSE-KMS) 加密的，请确保在访问 S3 存储桶时授予使用该密钥的 Amazon IoT Core 权限，方法是将以下语句添加到您的密钥策略中：

```
{
 "Sid": "Statement1",
 "Effect": "Allow",
 "Principal": {
  "Service": "iot.amazonaws.com"
 },
 "Action": [
  "kms:Decrypt",
  "kms:GenerateDataKey*",
  "kms:DescribeKey"
 ],
        "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

}
```

在规则中配置 protobuf 解码

将描述符文件上传到 Amazon S3 存储桶后，配置一条规则，该规则可以使用 decode (value, decodingScheme) SQL 函数解码 (p. 552)您的 protobuf 消息有效负载格式。详细的函数签名和示例可以在 SQL 参考的 decode (value, decodingScheme) (p. 552) SAmazon IoT QL 函数中找到。

以下是使用 decode (value, decodingScheme) (p. 552) 函数的 SQL 表达式示例：

```
SELECT VALUE decode(*, 'proto', '<BUCKET NAME>, '<FILENAME>.desc', '<PROTO_FILENAME>',
 '<PROTO_MESSAGE_TYPE>') FROM '<MY_TOPIC>'
```

在此示例表达式中：

- 您可以使用 decode (value, decodingScheme) (p. 552) SQL 函数来解码引用的二进制消息有效负载*。这可以是二进制 protobuf 编码的有效负载，也可以是表示 base64 编码的 protobuf 有效载荷的 JSON 字符串。
- 提供的消息有效负载使用中 `PROTO_FILENAME.proto` 定义的 `Person` 消息类型进行编码。
- 名为的 Amazon S3 存储桶`BUCKET NAME`包含`FILENAME.desc`生成的自`PROTO_FILENAME.proto`。

完成配置后，向 Amazon IoT Core 发布一条关于规则订阅的主题的消息。

限制

Amazon IoT Core 规则支持 protobuf，但存在以下限制：

- 不支持在替换模板中解码 protobuf 消息有效负载。
- 解码 protobuf 消息有效负载时，可以在单个 SQL 表达式中使用解码 SQL 函数 (p. 552)最多 2 次。

- 最大入站有效负载大小为 128 KiB（1KiB = 1024 字节），最大出站有效负载大小为 128 KiB，存储在 Amazon S3 存储桶中的 `FileDescriptorSet` 对象的最大大小为 32 KiB。
- 不支持使用 SSE-C 加密进行加密的 Amazon S3 存储桶。

最佳实践

以下是一些最佳实践和故障排查提示。

- 在 Amazon S3 存储桶中备份您的原型文件。

  备份您的原型文件是一种很好的做法，以防出现问题。例如，如果您在运行 protoc 时错误地修改了没有备份的原型文件，这可能会导致您的生产堆栈出现问题。有多种方法可以在 Amazon S3 存储桶中备份您的文件。例如，您可以在 S3 存储桶中使用版本控制。有关如何备份 Amazon S3 存储桶中的文件的更多信息，请参阅 Amazon S3 开发人员指南。
- 配置 Amazon IoT 日志记录以查看日志条目。

  配置Amazon IoT日志记录是一种很好的做法，这样您就可以查看您的账户的Amazon IoT日志 CloudWatch。当规则的 SQL 查询调用外部函数时，Amazon IoT Core 规则会生成一个带有 `FunctionExecution` 的 `eventType` 的日志条目，其中包含可帮助您排查故障的原因字段。可能的错误包括找不到 Amazon S3 对象，或者 protobuf 文件描述符无效。有关如何配置 Amazon IoT 日志记录和查看日志条目的更多信息，请参阅配置 Amazon IoT 日志记录和规则引擎日志条目。
- 使用新的对象键更新 `FileDescriptorSet`，并更新规则中的对象键。

  您可以`FileDescriptorSet`通过将更新后的描述符文件上传到 Amazon S3 存储桶来更新。您对 `FileDescriptorSet` 的更新最多需要 15 分钟能够反映出来。为了避免这种延迟，`FileDescriptorSet`使用新的对象键上传更新后更新后更新后更新规则中的此对象密钥是一种很好的做法。

SQL 版本

Amazon IoT 规则引擎使用一种类似 SQL 的语法从 MQTT 消息选择数据。SQL 语句基于 SQL 版本进行解释，该版本由描述此规则的 JSON 文档中的 `awsIotSqlVersion` 属性指定。有关 JSON 规则文档结构的更多信息，请参阅创建规则 (p. 445)。借助 `awsIotSqlVersion` 属性，您可以指定想要使用的 Amazon IoT SQL 规则引擎版本。当部署新版本时，您可以继续使用早期版本或更改规则以使用新版本。您当前的规则将继续使用创建时所用的版本。

以下 JSON 示例介绍了如何使用 `awsIotSqlVersion` 属性指定 SQL 版本：

```
{
    "sql": "expression",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [{
        "republish": {
            "topic": "my-mqtt-topic",
            "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
        }
    }]
}
```

Amazon IoT 目前支持以下 SQL 版本：

- `2016-03-23` – 2016 年 3 月 23 日构建的 SQL 版本（建议）。
- `2015-10-08` – 2015 年 10 月 8 日构建的 SQL 原始版本。
- `beta` – 最新的 SQL 测试版本。此版本可能会给您的规则带来突破性变化。

2016-03-23 SQL 规则引擎版本中的新增功能

- 针对选择嵌套 JSON 对象的修复程序。
- 针对阵列查询的修复程序。
- 对象内查询支持。有关更多信息，请参阅嵌套对象查询 (p. 586)：
- 支持将阵列作为顶级对象输出。
- 添加可应用于 JSON 和非 JSON 格式数据的 `encode(value, encodingScheme)` 函数。有关更多信息，请参阅编码函数 (p. 554)。

将 Array 作为顶级对象输出

此功能允许规则将阵列作为顶级对象返回。例如，给定了以下 MQTT 消息：

```
{
    "a": {"b":"c"},
    "arr":[1,2,3,4]
}
```

以及以下规则：

```
SELECT VALUE arr FROM 'topic'
```

该规则生成以下输出。

```
[1,2,3,4]
```

Amazon IoT Device Shadow 服务

Amazon IoT Device Shadow 服务在 Amazon IoT 事物对象中添加影子。无论设备是否连接到 Amazon IoT，影子都可以向应用程序和其他服务提供设备的状态。Amazon IoT 事物对象可以具有多个命名的影子，以便 IoT 解决方案具有更多选项以将设备连接到其它应用程序和服务。

在明确创建之前，Amazon IoT 事物对象没有任何命名的影子；不过，在创建事物时，将会为其创建未命名的经典影子。可以使用 Amazon IoT 控制台创建、更新和删除影子。设备、其它 Web 客户端和服务可以使用 MQTT 和预留的 MQTT 主题 (p. 105)、使用 Device Shadow REST API (p. 616) 的 HTTP 以及适用于 Amazon IoT 的 Amazon CLI 创建、更新和删除影子。由于 Amazon 将影子存储在云中，无论是否连接了设备，影子都可以从应用程序和其它云服务中收集和报告设备状态数据。

使用影子

影子为设备、应用程序和其它云服务提供可靠的数据存储以共享数据。它们允许设备、应用程序和其它云服务连接和断开连接，而不会丢失设备的状态。

在设备、应用程序和其它云服务连接到 Amazon IoT 后，它们可以通过设备影子访问和控制设备的当前状态。例如，应用程序可以更新影子以请求更改设备的状态。Amazon IoT 发布一条消息以指示对设备所做的更改。设备接收该消息，更新其状态以保持匹配，然后发布一条消息以指示其更新状态。Device Shadow 服务在相应的影子中反映该更新状态。应用程序可以订阅影子的更新，也可以查询影子以获取其当前状态。

在设备脱机时，应用程序仍然可以与 Amazon IoT 和设备的影子进行通信。在设备重新连接时，它接收其影子的当前状态，以便它可以更新其状态以与其影子的状态匹配，然后发布一条消息以指示其更新状态。同样，如果应用程序脱机并且设备状态在其脱机时发生变化，则设备将影子保持更新状态，以便应用程序可以在设备重新连接时查询影子以获取其当前状态。

如果设备经常处于离线状态，但您希望将其配置为在重新连接后接收增量消息，则可以使用持久性会话功能。有关持久性会话过期期限的更多信息，请参阅持久性会话过期期限。

选择使用命名或未命名的影子

Device Shadow 服务支持命名的影子和未命名的经典影子，与过去使用的一样。事物对象可以具有多个命名的影子，并且最多可以具有一个未命名的经典影子。事物对象可以同时具有命名和未命名的影子；不过，用于访问每种影子的 API 略有不同，因此，决定哪种类型的影子最适合您的解决方案并仅使用该类型可能更高效。有关用于访问影子的 API 的更多信息，请参阅影子主题 (p. 105)。

通过使用命名的影子，您可以为事物对象的状态创建不同的视图。例如，您可以将一个具有很多属性的事物对象划分为具有逻辑属性组的影子，每个影子由其影子名称标识。您也可以将属性分组到不同的影子，并使用策略控制访问以限制对属性的访问。有关与设备影子配合使用的策略的更多信息，请参阅 Amazon IoT 的操作、资源和条件键。

经典的未命名影子比命名的影子更简单，但在某些程度上比命名的影子更受限制。每个 Amazon IoT 事物对象只能具有一个未命名的影子。如果预计您的 IoT 解决方案对影子数据的需求有限，您可能希望以这种方式开始使用影子。不过，如果您认为将来可能要添加其它影子，请考虑从一开始就使用命名的影子。

机群索引支持不同的未命名影子和命名影子。有关更多信息，请参阅管理实例集索引 (p. 803)。

访问影子

每个影子具有预留的 MQTT 主题 (p. 105)和 HTTP URL (p. 616)，该 URL 支持对影子执行 `get`、`update` 和 `delete` 操作。

影子使用 JSON 影子文档 (p. 627)以存储和检索数据。影子的文档包含一个状态属性，以描述设备状态的以下方面：

- `desired`

  应用程序更新 `desired` 对象以指定设备属性的所需状态。
- `reported`

  设备在 `reported` 对象中报告其当前状态。
- `delta`

  Amazon IoT 在 `delta` 对象中报告所需的状态和报告的状态之间的差异。

存储在影子中的数据是由更新操作的消息正文的状态属性确定的。后续的更新操作可以修改现有数据对象的值，也可以在影子的状态对象中添加和删除键和其它元素。有关访问影子的更多信息，请参阅 在设备中使用影子 (p. 596) 和 在应用程序和服务中使用影子 (p. 598)。

> **Important**
>
> 发出更新请求的权限应限制为受信任的应用程序和设备。这可防止意外更改影子的状态属性；否则，使用影子的设备和应用程序应设计为需要更改状态属性中的键。

在设备、应用程序和其它云服务中使用影子

要在设备、应用程序和其它云服务中使用影子，需要在它们之间保持一致性和协调性。Amazon IoT Device Shadow 服务存储影子状态，在影子状态发生变化时发送消息，以及响应指示更改了其状态的消息。您的 IoT 解决方案中的设备、应用程序和其它云服务必须管理其状态，并使其与设备影子的状态保持一致。

影子状态数据是动态的，有权访问影子的设备、应用程序和其它云服务可以对其进行更改。因此，请务必考虑每个设备、应用程序和其它云服务如何与影子进行交互。例如：

- 在将状态数据传送到影子时，设备 应仅写入到影子状态的 `reported` 属性中。
- 在通过影子将状态更改请求传送到设备时，应用程序和其它云服务 应仅写入到 `desired` 属性中。

> **Important**
>
> 影子数据对象中包含的数据与其它影子和其他事物对象属性的数据（例如事物的属性以及事物对象的设备可能发布的 MQTT 消息内容）无关。不过，如果需要，设备可以在不同的 MQTT 主题和影子中报告相同的数据。
> 支持多个影子的设备必须将它在不同影子中报告的数据保持一致。

消息顺序

不保证 Amazon IoT 服务发送的消息按任何特定顺序到达设备。以下方案显示了此时发生的情况。

初始状态文档：

```
{
  "state": {
    "reported": {
      "color": "blue"
    }
  },
  "version": 9,
  "timestamp": 123456776
}
```

更新 1：

```
{
  "state": {
```

```
    "desired": {
      "color": "RED"
    }
  },
  "version": 10,
  "timestamp": 123456777
}
```

更新 2：

```
{
  "state": {
    "desired": {
      "color": "GREEN"
    }
  },
  "version": 11,
  "timestamp": 123456778
}
```

最终状态文档：

```
{
  "state": {
    "reported": {
      "color": "GREEN"
    }
  },
  "version": 12,
  "timestamp": 123456779
}
```

这将产生两个增量消息：

```
{
  "state": {
    "color": "RED"
  },
  "version": 11,
  "timestamp": 123456778
}
```

```
{
  "state": {
    "color": "GREEN"
  },
  "version": 12,
  "timestamp": 123456779
}
```

设备可能不会按次序收到这些消息。由于这些消息中的状态是累计的，设备可以安全地弃用版本号比正在追踪的版本号更早的所有消息。如果设备在接收版本 11 的增量之前收到版本 12 的增量，则可以安全地弃用版本 11 的消息。

修剪影子消息

要降低发送到您的设备的影子消息的大小，请定义一项规则，以仅选择设备所需的字段然后将消息重新发布到设备正在侦听的 MQTT 主题。

规则应在 JSON 中指定，且应与以下内容类似：

```
{
  "sql": "SELECT state, version FROM '$aws/things/+/shadow/update/delta'",
  "ruleDisabled": false,
  "actions": [
    {
      "republish": {
        "topic": "${topic(3)}/delta",
        "roleArn": "arn:aws:iam:123456789012:role/my-iot-role"
      }
    }
  ]
}
```

SELECT 语句用于确定将消息中的哪些字段重新发布到指定的主题。“+”通配符用于匹配所有影子名称。该规则指定，应将所有匹配的消息重新发布到指定的主题。在此情况下，可以使用 "topic()" 函数指定将消息重新发布到哪个主题。topic(3) 用于评估原始主题中的事物名称。有关创建规则的更多信息，请参阅Amazon IoT 规则 (p. 442)。

在设备中使用影子

本部分介绍了设备使用 MQTT 消息与影子进行通信，这是设备与 Amazon IoT Device Shadow 服务进行通信的首选方法。

影子通信使用 MQTT 的发布/订阅通信模型以模拟请求/响应模型。每个影子操作包含请求主题、成功的响应主题 (accepted) 和错误响应主题 (rejected)。

如果您希望应用程序和服务能够确定是否连接了设备，请参阅检测是否连接了设备 (p. 599)。

> **Important**
>
> 由于 MQTT 使用发布/订阅通信模型，您必须先订阅响应主题，然后再发布请求主题。否则，您不会收到您发布请求的响应。
> 如果您使用 Amazon IoT Device SDK (p. 1283) 来调用 Device Shadow 服务 API，这将为您进行处理。

本部分中的示例使用主题的缩写形式，其中 *ShadowTopicPrefix* 可以表示命名或未命名的影子，如该表中所述。

影子可以是命名或未命名（经典）的。每个影子使用的主题仅在主题前缀上有所不同。下表显示每种影子类型使用的主题前缀。

ShadowTopicPrefix 值	影子类型
$aws/things/*thingName*/shadow	未命名的（经典）影子
$aws/things/*thingName*/shadow/name/*shadowName*	命名的影子

> **Important**
>
> 确保应用程序或服务使用的影子与设备中的相应实施保持一致，并且该实施支持使用这些影子。例如，考虑如何创建、更新和删除影子。还要考虑如何在设备以及通过影子访问设备的应用程序或服务中处理更新。您的设计应明确指定如何更新和报告设备的状态，以及应用程序和服务如何与设备及其影子进行交互。

要创建完整的主题，请为要表示的影子类型选择 *ShadowTopicPrefix*，将 *thingName* 和 *shadowName*（如果适用）替换为相应的值，然后在其后面附加主题存根，如下表中所示。请记住，主题区分大小写。

有关影子的保留主题的更多信息，请参阅 影子主题 (p. 105)。

首次连接到 Amazon IoT 时初始化设备

在 Amazon IoT 中注册设备后，它应该为支持的影子订阅这些 MQTT 消息。

主题	意义	在收到该主题时设备应执行的操作
ShadowTopicPrefix/delete/accepted	已接受 delete 请求并且 Amazon IoT 删除了影子。	为满足删除的影子要求而需要执行的操作，例如，停止发布更新。
ShadowTopicPrefix/delete/rejected	Amazon IoT 已拒绝 delete 请求，并且未删除影子。消息正文包含错误信息。	响应消息正文中的错误消息。
ShadowTopicPrefix/get/accepted	Amazon IoT 已接受 get 请求，并且消息正文包含当前影子文档。	处理消息正文中的状态文档所需的操作。
ShadowTopicPrefix/get/rejected	Amazon IoT 已拒绝 get 请求，并且消息正文包含错误信息。	响应消息正文中的错误消息。
ShadowTopicPrefix/update/accepted	Amazon IoT 已接受 update 请求，并且消息正文包含当前影子文档。	确认消息正文中的更新数据与设备状态匹配。
ShadowTopicPrefix/update/rejected	Amazon IoT 已拒绝 update 请求，并且消息正文包含错误信息。	响应消息正文中的错误消息。
ShadowTopicPrefix/update/delta	发送到 Amazon IoT 的请求更新了影子文档，并且消息正文包含请求的更改。	更新设备的状态以与消息正文中的所需状态匹配。
ShadowTopicPrefix/update/documents	最近完成了影子更新，并且消息正文包含当前影子文档。	确认消息正文中的更新状态与设备的状态匹配。

在为每个影子订阅上表中的消息后，设备应进行测试，以确定是否已将 /get 主题发布到它支持的每个影子以创建这些影子。如果收到 /get/accepted 消息，则消息正文包含影子文档，设备可以使用该文档初始化其状态。如果收到 /get/rejected 消息，应发布具有当前设备状态的 /update 消息以创建影子。

例如，假设您有事物 My_IoT_Thing，但其没有任何经典或命名影子。如果现在发布关于预留主题 $aws/things/My_IoT_Thing/shadow/get 的 /get 请求，其会在 $aws/things/My_IoT_Thing/shadow/get/rejected 主题返回一条错误，因为该事物没有任何影子。要想解决此错误，请先使用含当前设备状态（例如以下有效负载）的 $aws/things/My_IoT_Thing/shadow/update 主题发布一条 /update 消息。

```
 "state": {
    "reported": {
      "welcome": "aws-iot",
      "color": "yellow"
    }
  }
}
```

现在为该事物创建了经典影子，消息将发布到 $aws/things/My_IoT_Thing/shadow/update/accepted 主题。如果发布到 $aws/things/My_IoT_Thing/shadow/get 主题，其会将含设备状态的响应返回到 $aws/things/My_IoT_Thing/shadow/get/accepted 主题。

对于命名影子，必须先创建命名影子或发布含影子名称的更新，才能使用 get 请求。例如，要创建命名影子 namedShadow1，首先将设备状态信息发布到 $aws/things/My_IoT_Thing/shadow/namedShadow1/update 主题。若要检索状态信息，请将 /get 请求用于命名影子 $aws/things/My_IoT_Thing/shadow/namedShadow1/get。

在设备连接到 Amazon IoT 后处理消息

在设备连接到 Amazon IoT 后，它可以接收 /update/delta 消息，并且应按以下方式将设备状态与其影子更改保持匹配：

1. 读取所有收到的 /update/delta 消息并同步设备状态以保持匹配。
2. 以 reported 消息正文发布 /update 消息，该消息正文包含设备的当前状态，无论设备状态何时发生变化。

在连接设备后，它应在出现指示时发布这些消息。

指示	主题	有效负载
设备的状态已发生变化。	*ShadowTopicPrefix*/update	具有 reported 属性的影子文档。
设备可能与影子不同步。	*ShadowTopicPrefix*/get	（空）
对设备执行的操作指示设备不再支持影子，例如在移除或更换设备时。	*ShadowTopicPrefix*/delete	（空）

在设备重新连接到 Amazon IoT 时处理消息

在具有一个或多个影子的设备连接到 Amazon IoT 时，它应按以下方式将其状态与它支持的所有影子的状态进行同步：

1. 读取所有收到的 /update/delta 消息并同步设备状态以保持匹配。
2. 使用 reported 消息正文发布 /update 消息，该消息正文包含设备的当前状态。

在应用程序和服务中使用影子

本部分介绍了应用程序或服务如何与 Amazon IoT Device Shadow 服务进行交互。该示例假定应用程序或服务仅与影子进行交互，并通过影子与设备进行交互。该示例不包括任何管理操作，例如创建或删除影子。

该示例使用 Amazon IoT Device Shadow 服务的 REST API 与影子进行交互。与 在设备中使用影子 (p. 596) 中使用的示例（它使用发布/订阅通信模型）不同，该示例使用 REST API 的请求/响应通信模型。这意味着，应用程序或服务必须先发出请求，然后才能从 Amazon IoT 收到响应。但该模型的一个缺点是，它不支持通知。如果应用程序或服务需要及时通知设备状态变化，请考虑使用 MQTT 或基于 WSS 的 MQTT 协议，这些协议支持发布/订阅通信模型，如在设备中使用影子 (p. 596)中所述。

> **Important**
>
> 确保应用程序或服务使用的影子与设备中的相应实施保持一致，并且该实施支持使用这些影子。例如，考虑如何创建、更新和删除影子，以及如何在设备以及访问影子的应用程序或服务中处理更

新。您的设计应明确指定如何更新和报告设备的状态，以及应用程序和服务如何与设备及其影子进行交互。

命名的影子的 REST API URL 为：

```
https://endpoint/things/thingName/shadow?name=shadowName
```

未命名的影子的 REST API URL 为：

```
https://endpoint/things/thingName/shadow
```

其中：

endpoint

CLI 命令返回的终端节点：

```
aws iot describe-endpoint --endpoint-type IOT:Data-ATS
```

thingName

影子所属的事物对象的名称

shadowName

命名的影子的名称。该参数不用于未命名的影子。

在连接到 Amazon IoT 时初始化应用程序或服务

在应用程序首次连接到 Amazon IoT 时，它应将 HTTP GET 请求发送到它使用的影子的 URL，以获取它使用的影子的当前状态。这样，它就可以将应用程序或服务同步到影子。

在应用程序或服务连接到 Amazon IoT 后处理状态更改

在应用程序或服务连接到 Amazon IoT 后，它可以在使用的影子的 URL 上发送 HTTP GET 请求以定期查询当前状态。

在最终用户与应用程序或服务交互以更改设备状态时，应用程序或服务可以向它使用的影子的 URL 发送 HTTP POST 请求以更新影子的 `desired` 状态。该请求返回已接受的更改，但您可能需要发出 HTTP GET 请求以轮询影子，直到设备使用新状态更新了影子为止。

检测是否连接了设备

要确定当前是否连接了设备，请在影子文档中包含一个 `connected` 属性；如果设备由于错误而断开连接，则使用 MQTT Last Will and Testament (LWT) 消息将 `connected` 属性设置为 `false`。

Note

Amazon IoT Device Shadow 服务忽略发送到 Amazon IoT 保留主题（以 $ 开头的主题）的 MQTT LWT 消息。不过，它们由订阅的客户端和 Amazon IoT 规则引擎处理，因此，您需要创建发送到非保留主题的 LWT 消息，并创建一个规则以将 MQTT LWT 消息作为影子更新消息重新发布到影子的保留更新主题 *ShadowTopicPrefix*/update。

向 Device Shadow 服务发送 LWT 消息

1. 创建一个规则以在保留的主题上重新发布 MQTT LWT 消息。以下示例规则会侦听关于 `my/things/myLightBulb/update` 主题的消息并其重新发布到 `$aws/things/myLightBulb/shadow/update`。

```
{
    "rule": {
    "ruleDisabled": false,
    "sql": "SELECT * FROM 'my/things/myLightBulb/update'",
    "description": "Turn my/things/ into $aws/things/",
    "actions": [
        {
        "republish": {
            "topic": "$$aws/things/myLightBulb/shadow/update",
            "roleArn": "arn:aws:iam:123456789012:role/aws_iot_republish"
            }
        }
     ]
   }
}
```

2. 在设备连接到 Amazon IoT 时，它在非保留主题中注册 LWT 消息，以使重新发布规则能够识别该消息。在此示例中，该主题为 `my/things/myLightBulb/update`，它将 connected 属性设置为 `false`。

```
{
    "state": {
        "reported": {
            "connected":"false"
        }
    }
}
```

3. 在连接后，设备在其影子更新主题 `$aws/things/myLightBulb/shadow/update` 上发布消息以报告其当前状态，这包括将其 `connected` 属性设置为 `true`。

```
{
     "state": {
        "reported": {
            "connected":"true"
        }
    }
}
```

4. 在设备正常断开连接之前，它在其影子更新主题 `$aws/things/myLightBulb/shadow/update` 上发布消息以报告其最新状态，这包括将其 `connected` 属性设置为 `false`。

```
{
    "state": {
        "reported": {
            "connected":"false"
        }
    }
}
```

5. 如果设备由于错误而断开连接，Amazon IoT 消息代理将代表设备发布设备的 LWT 消息。重新发布规则检测该消息，并发布影子更新消息以更新设备影子的 `connected` 属性。

模拟 Device Shadow 服务通信

本主题说明了 Device Shadow 服务如何充当中介，并允许设备和应用程序使用影子以更新、存储和检索设备的状态。

要说明本主题中所述的交互并进一步对其进行阐述，您需要具有一个 Amazon Web Services 账户 账户以及可以在其中运行 Amazon CLI 的系统。如果没有该账户和系统，您仍然可以在代码示例中看到交互。

在该示例中，Amazon IoT 控制台表示设备。Amazon CLI 表示通过影子访问设备的应用程序或服务。Amazon CLI 界面与应用程序可用于与 Amazon IoT 通信的 API 非常相似。该示例中的设备是一个智能灯泡，应用程序显示灯泡的状态，并且可以更改灯泡的状态。

设置模拟

这些流程打开模拟设备的 Amazon IoT 控制台和模拟应用程序的命令行窗口以初始化模拟。

设置模拟环境

1. 您需要具有一个 Amazon Web Services 账户 以自行运行本主题中的示例。如果您没有 Amazon Web Services 账户，请按照 设置您的Amazon Web Services 账户 (p. 16) 中所述操作创建一个账户。
2. 打开 Amazon IoT 控制台，然后在左侧菜单中选择 Test (测试) 以打开 MQTT client (MQTT 客户端)。
3. 在另一个窗口中，在已安装 Amazon CLI 的系统上打开一个终端窗口。

您应打开了两个窗口：一个窗口显示 Amazon IoT 控制台上的 Test (测试) 页面，另一个窗口显示命令行提示符。

初始化设备

在该模拟中，我们将使用一个名为 mySimulatedThing 的事物对象及其名为 simShadow1 的影子。

创建事物对象及其 IoT 策略

要创建事物对象，请在 Amazon IoT 控制台：

1. 选择 Manage（管理），然后选择 Things（事物）。
2. 如果事物均已列出，请单击 Create（创建）按钮，否则单击 Register a single thing（注册单个事物）以创建单个 Amazon IoT 事物。
3. 输入名称 `mySimulatedThing`，将其它设置保留为默认值，然后单击 Next（下一步）。
4. 使用单击证书创建可生成证书，以验证设备与 Amazon IoT 的连接。单击 Activate（激活）以激活证书。
5. 您可以附上策略 `My_IoT_Policy`，这将授予设备发布和订阅 MQTT 预留主题的权限。有关如何创建 Amazon IoT 事物以及如何创建此策略的更多详情，请参阅 创建一个事物对象 (p. 36)。

创建事物对象的命名影子

按如下方式向 `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` 主题发布更新请，您可以为物件创建命名影子。

也可以按以下方式创建命名影子：

1. 在 Amazon IoT 控制台中，在显示的事物列表中选择您的事物对象，然后选择 Shadows（影子）。
2. 选择 Add a shadow（添加影子），输入名称 `simShadow1`，然后选择 Create（创建）添加命名的影子。

订阅并发布预留的 MQTT 主题

在控制台中，订阅预留的 MQTT 影子主题。这些主题是 `get`、`update` 和 `delete` 操作的响应，以便设备在发布操作后准备好接收响应。

在 MQTT client (MQTT 客户端) 中订阅 MQTT 主题

1. 在 MQTT 客户端中，选择 Subscribe to a topic（订阅主题）。

2. 输入 get、update 和 delete 要订阅的主题。每次从以下列表中复制一个主题，然后将其粘贴到 Topic filter（主题筛选条件）字段，然后单击 Subscribe（订阅）。您应看到主题显示在 Subscriptions（订阅）项下。

   - $aws/things/mySimulatedThing/shadow/name/simShadow1/delete/accepted
   - $aws/things/mySimulatedThing/shadow/name/simShadow1/delete/rejected
   - $aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted
   - $aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected
   - $aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted
   - $aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected
   - $aws/things/mySimulatedThing/shadow/name/simShadow1/update/delta
   - $aws/things/mySimulatedThing/shadow/name/simShadow1/update/documents

   此时，在 Amazon IoT 发布主题时，模拟的设备已准备好接收这些主题。

在 MQTT client（MQTT 客户端）中发布 MQTT 主题

在设备初始化自身并订阅响应主题后，它应查询支持的影子。此模拟只支持一个影子，即支持命名为 mySimulatedThing 和命名为 simShadow1 的事物对象的影子。

从 MQTT client (MQTT 客户端) 中获取当前影子状态

1. 在 MQTT client (MQTT 客户端) 中，选择 Publish to a topic (发布到主题)。
2. 在 Publish（发布）中，输入以下主题，并从输入主题下方的消息正文窗口中删除任何内容。然后，您可以选择 Publish to topic（发布主题）以发布请求。$aws/things/mySimulatedThing/shadow/name/simShadow1/get。

   如果您未创建命名影子 simShadow1，您将在 $aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected 主题中收到一条消息，并且 code 为 404（例如，在该示例中，由于尚未创建影子，因此我们接下来将进行创建）。

```
{
  "code": 404,
  "message": "No shadow exists with name: 'simShadow1'"
}
```

使用设备的当前状态创建影子

1. 在 MQTT client（MQTT 客户端）中，选择 Publish to a topic（发布主题），然后输入该主题。

```
$aws/things/mySimulatedThing/shadow/name/simShadow1/update
```

2. 在消息正文窗口中，在您输入主题的位置下面输入该影子文档，以显示设备正在报告其 ID 并以 RGB 值形式报告其当前颜色。选择 Publish（发布）以发布请求。

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
```

```
    }
  },
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

如果您在主题中收到一条消息：

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`：表示创建了影子，并且消息正文包含当前影子文档。
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`：查看消息正文中的错误。
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`：影子已存在，并且消息正文包含当前影子状态，如示例中所示。通过使用该消息，您可以设置设备或确认它与影子状态匹配。

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
    }
  },
  "version": 3,
  "timestamp": 1591140517,
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

从应用程序中发送更新

本部分使用 Amazon CLI 说明应用程序如何与影子进行交互。

使用 Amazon CLI 获取影子的当前状态

从命令行中，输入以下命令。

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 /dev/
stdout
```

在 Windows 平台上，您可以使用 con 而不是 /dev/stdout。

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 con
```

由于影子存在并且设备已将其初始化以反映其当前状态，因此，它应返回以下影子文档。

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        },
        {
          "timestamp": 1591140517
        }
      ]
    }
  },
  "version": 3,
  "timestamp": 1591141111
}
```

应用程序可以使用该响应以初始化设备状态的表示形式。

如果应用程序更新状态（例如，在最终用户将智能灯泡的颜色更改为黄色时），应用程序将发送 update-thing-shadow 命令。该命令对应于 UpdateThingShadow REST API。

从应用程序中更新影子

从命令行中，输入以下命令。

Amazon CLI v2.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1
 \
    --cli-binary-format raw-in-base64-out \
    --payload '{"state":{"desired":{"ColorRGB":[255,255,0]}},"clientToken":"21b21b21-
bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

Amazon CLI v1.x

```
aws iot-data update-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1
 \
    --payload '{"state":{"desired":{"ColorRGB":[255,255,0]}},"clientToken":"21b21b21-
bfd2-4279-8c65-e2f697ff4fab"}' /dev/stdout
```

如果成功，该命令应返回以下影子文档。

```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
      ]
    }
  },
  "version": 4,
  "timestamp": 1591141596,
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

响应设备中的更新

返回到 MQTT client（MQTT 客户端）中的 Amazon 控制台中，您应看到以下消息：Amazon IoT 已发布，以反映前一部分中发布的更新命令。

在 MQTT client (MQTT 客户端) 中查看更新消息

在 MQTT client（客户端）中，在 Subscriptions（订阅）列中选择 $aws/things/mySimulatedThing/shadow/name/simShadow1/updae/delta。如果主题名称被截断，您可以在其中暂停以查看完整主题。在本主题的主题日志中，您应该看到与此类似的 `/delta` 消息。

```
{
  "version": 4,
  "timestamp": 1591141596,
  "state": {
    "ColorRGB": [
      255,
      255,
      0
    ]
  },
```

```
  "metadata": {
    "ColorRGB": [
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      }
    ]
  },
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

设备将处理该消息的内容，以将设备状态设置为与消息中的 `desired` 状态匹配。

在设备更新状态以与消息中的 `desired` 状态匹配后，它必须发布更新消息以将新报告的状态发回到 Amazon IoT。该流程在 MQTT client (MQTT 客户端) 中模拟这种情况。

从设备中更新影子

1. 在 MQTT client (MQTT 客户端) 中，选择 Publish to a topic (发布到主题)。
2. 在消息正文窗口中，在消息正文窗口上方的主题字段中，输入影子的主题，然后输入 `/update` 操作：`$aws/things/mySimulatedThing/shadow/name/simShadow1/update`，然后在消息正文中输入此更新的影子文档，该文档描述了设备的当前状态。选择 Publish（发布）以发布更新的设备状态。

   ```
   {
     "state": {
       "reported": {
         "ColorRGB": [255,255,0]
         }
     },
     "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
   }
   ```

   如果消息已由 Amazon IoT 成功接收，您应在 MQTT client（MQTT 客户端）中的 $aws/things/mySimulatedThing/shadow/name/simShadow1/updae/accepted 消息日志中看到新的响应，其中包含影子的当前状态，如示例所示。

   ```
   {
     "state": {
       "reported": {
         "ColorRGB": [
           255,
           255,
           0
         ]
       }
     },
     "metadata": {
       "reported": {
         "ColorRGB": [
           {
             "timestamp": 1591142747
           },
           {
             "timestamp": 1591142747
           },
   ```

```
        {
          "timestamp": 1591142747
        }
      ]
    }
  },
  "version": 5,
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

如果成功更新为报告的设备状态，还会导致 Amazon IoT 在消息中将影子状态详细描述发送到 主题，例如，设备在前面的流程中执行的影子更新导致的以下消息正文。

```
{
  "previous": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          128,
          128,
          128
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          },
          {
            "timestamp": 1591140517
          }
        ]
      }
```

```
    },
    "version": 4
  },
  "current": {
    "state": {
      "desired": {
        "ColorRGB": [
          255,
          255,
          0
        ]
      },
      "reported": {
        "ID": "SmartLamp21",
        "ColorRGB": [
          255,
          255,
          0
        ]
      }
    },
    "metadata": {
      "desired": {
        "ColorRGB": [
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          },
          {
            "timestamp": 1591141596
          }
        ]
      },
      "reported": {
        "ID": {
          "timestamp": 1591140517
        },
        "ColorRGB": [
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          },
          {
            "timestamp": 1591142747
          }
        ]
      }
    },
    "version": 5
  },
  "timestamp": 1591142747,
  "clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```


观察应用程序中的更新

应用程序现在可以查询影子以获取设备报告的当前状态。

使用 Amazon CLI 获取影子的当前状态

1. 从命令行中，输入以下命令。

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 /
dev/stdout
```

在 Windows 平台上，您可以使用 con 而不是 /dev/stdout。

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1
 con
```

2. 由于设备刚刚更新了影子以反映其当前状态，因此，它应返回以下影子文档。

```
{
  "state": {
    "desired": {
      "ColorRGB": [
        255,
        255,
        0
      ]
    },
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        255,
        255,
        0
      ]
    }
  },
  "metadata": {
    "desired": {
      "ColorRGB": [
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        },
        {
          "timestamp": 1591141596
        }
      ]
    },
    "reported": {
      "ID": {
        "timestamp": 1591140517
      },
      "ColorRGB": [
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        },
        {
          "timestamp": 1591142747
        }
      ]
    }
  },
```

```
  "version": 5,
  "timestamp": 1591143269
}
```

模拟补充内容

试验 Amazon CLI（表示应用程序）和控制台（表示设备）之间的交互以模拟您的 IoT 解决方案。

与影子交互

本主题介绍了 Amazon IoT 为处理影子而提供的三种方法的关联消息。这些方法包括：

`UPDATE`

创建影子（如果不存在），或使用消息正文中提供的状态信息更新现有影子的内容。Amazon IoT 记录每次更新的时间戳，以指示上次更新状态的时间。在影子的状态发生变化时，Amazon IoT 将 `/delta` 消息发送到所有 MQTT 订阅者，其中包含 `desired` 和 `reported` 状态之间的差异。收到 `/delta` 消息的设备或应用程序可以根据该差异执行操作。例如，设备可以将其状态更新为所需的状态，或者应用程序可以更新其 UI 以反映设备的状态变化。

`GET`

检索包含影子的完整状态的当前影子文档，包括元数据。

`DELETE`

删除设备影子及其内容。

您无法还原已删除的设备影子文档，但可以创建具有已删除设备影子文档的名称的新设备影子。如果您创建的设备影子文档与过去48小时内删除的文档同名，新设备影子文档的版本号和已删除文档的版本号相同。如果设备影子文档已删除超过 48 小时，具有相同名称的新设备影子文档的版本号为 0。

协议支持

Amazon IoT 支持 MQTT 和基于 HTTPS 的 REST API 协议以与影子进行交互。Amazon IoT 为 MQTT 发布和订阅操作提供一组保留的请求和响应主题。设备和应用程序应在发布请求主题之前订阅响应主题，以获取有关 Amazon IoT 如何处理请求的信息。有关更多信息，请参阅 Device Shadow MQTT 主题 (p. 620) 和 Device Shadow REST API (p. 616)。

请求和报告状态

在使用 Amazon IoT 和影子设计 IoT 解决方案时，您应该确定请求更改的应用程序或设备以及实施更改的应用程序或设备。通常，设备实施更改并向影子报告更改，而应用程序和服务响应并请求影子中的更改。您的解决方案可能有所不同，但本主题中的示例假定客户端应用程序或服务请求影子中的更改，而设备执行更改并向影子报告更改。

更新影子

应用程序或服务可以使用 UpdateThingShadow (p. 617) API 或发布到 /update (p. 622) 主题以更新影子的状态。更新仅影响请求中指定的字段。

在客户端请求状态更改时更新影子

在客户端使用 MQTT 协议请求影子中的状态更改时

1. 客户端应具有当前影子文档，以便它可以确定要更改的属性。有关如何获取当前影子文档的信息，请参阅 /get 操作。
2. 客户端订阅以下 MQTT 主题：
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/accepted
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/rejected
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/delta
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/documents
3. 客户端使用包含影子的所需状态的状态文档发布 $aws/things/*thingName*/shadow/name/*shadowName*/update 请求主题。只需在文档中包含要更改的属性。以下是具有所需状态的文档的示例。

   ```
   {
     "state": {
       "desired": {
         "color": {
           "r": 10
         },
         "engine": "ON"
       }
     }
   }
   ```
4. 如果更新请求有效，Amazon IoT 将在影子中更新所需的状态并在以下主题上发布消息：
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/accepted
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/delta

   /update/accepted 消息包含一个 /accepted 响应状态文档 (p. 629) 影子文档，而 /update/delta 消息包含一个 /delta 响应状态文档 (p. 629) 影子文档。
5. 如果更新请求无效，Amazon IoT 将在 $aws/things/*thingName*/shadow/name/*shadowName*/update/rejected 主题中发布一条消息，其中包含描述错误的 错误响应文档 (p. 631) 影子文档。

在客户端使用 API 请求影子中的状态更改时

1. 客户端调用 UpdateThingShadow （p. 617） API 并将 请求状态文档 (p. 628) 状态文档作为其消息正文。
2. 如果请求有效，Amazon IoT 将返回 HTTP 成功响应代码和 /accepted 响应状态文档 (p. 629) 影子文档以作为其响应消息正文。

   Amazon IoT 还会向 $aws/things/*thingName*/shadow/name/*shadowName*/update/delta 主题发布一条 MQTT 消息，其中包含订阅它的任何设备或客户端的 /delta 响应状态文档 (p. 629) 影子文档。
3. 如果请求无效，Amazon IoT 将返回 HTTP 错误响应代码 错误响应文档 (p. 631) 和以作为其响应消息正文。

当设备在 /update/delta 主题上收到 /desired 状态时，它将在设备中进行所需的更改。然后，它向 /update 主题发送一条消息，以向影子报告其当前状态。

在设备报告其当前状态时更新影子

在设备使用 MQTT 协议向影子报告其当前状态时

1. 在更新影子之前，设备应订阅以下 MQTT 主题：
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/accepted
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/rejected
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/delta
   - $aws/things/*thingName*/shadow/name/*shadowName*/update/documents
2. 设备向 $aws/things/*thingName*/shadow/name/*shadowName*/update 主题发布一条消息以报告其当前状态，例如，在该示例中。

```
{
    "state": {
        "reported" : {
            "color" : { "r" : 10 },
            "engine" : "ON"
        }
    }
}
```

3. 如果 Amazon IoT 接受更新，它将向 $aws/things/*thingName*/shadow/name/*shadowName*/update/accepted 主题发布一条消息，其中包含 /accepted 响应状态文档 (p. 629) 影子文档。
4. 如果更新请求无效，Amazon IoT 将在 $aws/things/*thingName*/shadow/name/*shadowName*/update/rejected 主题中发布一条消息，其中包含描述错误的 错误响应文档 (p. 631) 影子文档。

在设备使用 API 向影子报告其当前状态时

1. 设备调用 UpdateThingShadow （p. 617） API 并将 请求状态文档 (p. 628) 状态文档作为其消息正文。
2. 如果请求有效，Amazon IoT 将更新影子并返回 HTTP 成功响应代码以及 /accepted 响应状态文档 (p. 629) 影子文档以作为其响应消息正文。

   Amazon IoT 还会向 $aws/things/*thingName*/shadow/name/*shadowName*/update/delta 主题发布一条 MQTT 消息，其中包含订阅它的任何设备或客户端的 /delta 响应状态文档 (p. 629) 影子文档。
3. 如果请求无效，Amazon IoT 将返回 HTTP 错误响应代码 错误响应文档 (p. 631) 和以作为其响应消息正文。

乐观锁

您可以使用状态文档版本来确保正在更新的设备的影子文档为最新版本。当您为更新请求提供版本时，如果状态文档的当前版本与提供的版本不符，该服务将显示 HTTP 409 冲突响应代码并拒绝请求。

例如：

初始文档：

```
{
  "state": {
    "desired": {
      "colors": [
        "RED",
        "GREEN",
```

```
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

更新：（版本不匹配；该请求将被拒绝）

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 9
}
```

结果：

```
{
  "code": 409,
  "message": "Version conflict",
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

更新：（版本匹配；请求将被接受）

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 10
}
```

最终状态：

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 11
}
```

检索影子文档

您可以使用 GetThingShadow (p. 616) API 或订阅并发布到 /get (p. 620) 主题以检索影子文档。这会检索完整的影子文档，包括 `desired` 和 `reported` 状态之间的任何增量。无论设备还是客户端发出请求，该任务的流程都是相同的。

使用 MQTT 协议检索影子文档

1. 在更新影子之前，设备或客户端应订阅以下 MQTT 主题：
   - `$aws/things/thingName/shadow/name/shadowName/get/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/get/rejected`
2. 设备或客户端使用空消息正文向 `$aws/things/thingName/shadow/name/shadowName/get` 主题发布一条消息。
3. 如果请求成功，Amazon IoT 将向 `$aws/things/thingName/shadow/name/shadowName/get/accepted` 主题发布一条消息，并在消息正文中包含 /accepted 响应状态文档 (p. 629)。
4. 如果请求无效，Amazon IoT 将向 `$aws/things/thingName/shadow/name/shadowName/get/rejected` 主题发布一条消息，并在消息正文中包含 错误响应文档 (p. 631)。

使用 REST API 检索影子文档

1. 设备或客户端使用空消息正文调用 GetThingShadow (p. 616) API。
2. 如果请求有效，Amazon IoT 将返回 HTTP 成功响应代码以及 /accepted 响应状态文档 (p. 629) 影子文档以作为其响应消息正文。
3. 如果请求无效，Amazon IoT 将返回 HTTP 错误响应代码和 错误响应文档 (p. 631) 以作为其响应消息正文。

删除影子数据

可以使用两种方法删除影子数据：您可以在影子文档中删除特定的属性，也可以完全删除影子。

- 要从影子中删除特定的属性，请更新影子，但将您要删除的属性的值设置为 `null`。将从影子文档中删除值为 `null` 的字段。
- 要删除整个影子，请使用 DeleteThingShadow (p. 618) API 或发布到 /delete (p. 626) 主题。

> **Note**
>
> 删除影子不会立即将其版本号重置为 0。将在 48 小时后重置为零。

从影子文档中删除属性

使用 MQTT 协议从影子中删除属性

1. 设备或客户端应具有当前影子文档，以便它可以确定要更改的属性。有关如何获取当前影子文档的信息，请参阅 检索影子文档 (p. 613)。
2. 设备或客户端订阅以下 MQTT 主题：
   - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
   - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
3. 设备或客户端使用将 `null` 值分配给要删除的影子属性的状态文档以发布 `$aws/things/thingName/shadow/name/shadowName/update` 请求主题。只需在文档中包含要更改的属性。以下是删除 `engine` 属性的文档的示例。

```
{
  "state": {
    "desired": {
      "engine": null
    }
```

```
  }
}
```

4. 如果更新请求有效，Amazon IoT 将在影子中删除指定的属性，在 $aws/things/*thingName*/shadow/name/*shadowName*/update/accepted 主题中发布一条消息，并在消息正文中包含 /accepted 响应状态文档 (p. 629) 影子文档。
5. 如果更新请求无效，Amazon IoT 将在 $aws/things/*thingName*/shadow/name/*shadowName*/update/rejected 主题中发布一条消息，其中包含描述错误的 错误响应文档 (p. 631) 影子文档。

使用 REST API 从影子中删除属性

1. 设备或客户端使用 请求状态文档 (p. 628) 将 null 值分配给要删除的影子属性的以调用 UpdateThingShadow (p. 617) API。仅在文档中包含要删除的属性。以下是删除 engine 属性的文档的示例。

```
{
  "state": {
    "desired": {
      "engine": null
    }
  }
}
```

2. 如果请求有效，Amazon IoT 将返回 HTTP 成功响应代码和 /accepted 响应状态文档 (p. 629) 影子文档以作为其响应消息正文。
3. 如果请求无效，Amazon IoT 将返回 HTTP 错误响应代码 错误响应文档 (p. 631) 和以作为其响应消息正文。

删除影子

以下是删除设备影子时的一些注意事项。

- 将设备的影子状态设置为 null 并不会删除影子。在下次更新时，影子版本将会增加。
- 删除设备的影子并不会删除事物对象。删除事物对象并不会删除相应设备的影子。
- 删除影子不会立即将其版本号重置为 0。将在 48 小时后重置为零。

使用 MQTT 协议删除影子

1. 设备或客户端订阅以下 MQTT 主题：
   - $aws/things/*thingName*/shadow/name/*shadowName*/delete/accepted
   - $aws/things/*thingName*/shadow/name/*shadowName*/delete/rejected
2. 设备或客户端使用空消息缓冲区发布 $aws/things/*thingName*/shadow/name/*shadowName*/delete。
3. 如果删除请求有效，Amazon IoT 将删除影子，在 $aws/things/*thingName*/shadow/name/*shadowName*/delete/accepted 主题中发布一条消息，并在消息正文中包含简短的 /accepted 响应状态文档 (p. 629) 影子文档。以下是接受的删除消息的示例：

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

4. 如果更新请求无效，Amazon IoT 将在 $aws/things/*thingName*/shadow/name/*shadowName*/delete/rejected 主题中发布一条消息，其中包含描述错误的 错误响应文档 (p. 631) 影子文档。

使用 REST API 删除影子

1. 设备或客户端使用空消息缓冲区调用 DeleteThingShadow (p. 618) API。
2. 如果请求有效，Amazon IoT 将在消息正文中返回 HTTP 成功响应代码以及 /accepted 响应状态文档 (p. 629) 和简短 /accepted 响应状态文档 (p. 629) 影子文档。以下是接受的删除消息的示例：

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

3. 如果请求无效，Amazon IoT 将返回 HTTP 错误响应代码 错误响应文档 (p. 631) 和以作为其响应消息正文。

Device Shadow REST API

影子将显示用于更新状态信息的以下 URI：

```
https://account-specific-prefix-ats.iot.region.amazonaws.com/things/thingName/shadow
```

您的Amazon Web Services 账户账户具有特定的端点。要查找端点，您可以：

- 从 Amazon CLI 使用 describe-endpoint 命令。
- 使用 Amazon IoT 控制台设置。在 Settings（设置），端点将列在 Custom endpoint（自定义端点）下。
- 使用 Amazon IoT 控制台事物详细信息页面。在控制台中，执行以下操作：
  1. 打开 Manage（管理），并在 Manage（管理）中，选择 Things（事物）。
  2. 在事物列表中，选择要获取端点 URI 的事物。
  3. 选择 Device Shadows 选项卡，然后选择影子。您可以在 Device Shadow details（Device Shadow 详细信息）页面的 Device Shadow URL 部分查看端点 URI。

终端节点的格式如下：

```
identifier.iot.region.amazonaws.com
```

shadow REST API 遵循设备通信协议 (p. 75)中所述的相同 HTTPS 协议/端口映射。

> **Note**
>
> 您必须将 `iotdevicegateway` 用作身份验证的服务名称，才能使用 API。有关更多信息，请参阅 IotDataPlane。

API 操作

- GetThingShadow (p. 616)
- UpdateThingShadow (p. 617)
- DeleteThingShadow (p. 618)
- ListNamedShadowsForThing (p. 619)

您还可以通过提供 `name=shadowName` 用 API 创建命名影子，作为 API 的查询参数的一部分。

GetThingShadow

获取指定事物的影子。

响应状态文档包括 desired 状态与 reported 状态之间的增量。

请求

该请求包括标准的 HTTP 标头以及以下 URI：

```
HTTP GET https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

未命名的（经典）影子不需要使用 name 查询参数。

响应

请求成功后，响应将包括标准的 HTTP 标头以及以下代码和正文：

```
HTTP 200
Response Body: response state document
```

有关更多信息，请参阅响应状态文档示例 (p. 628)。

Authorization

要检索影子，需要一项允许调用方执行 iot:GetThingShadow 操作的策略。Device Shadow 服务接受两种形式的身份验证：使用 IAM 凭证的 Signature Version 4 或使用客户端证书的 TLS 双向身份验证。

以下是允许调用方检索设备的影子的示例策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:GetThingShadow",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

UpdateThingShadow

更新指定事物的影子。

更新仅影响请求状态文档中指定的字段。值为 null 的所有字段都会从设备的影子中删除。

请求

该请求包括标准的 HTTP 标头以及以下 URI 和正文：

```
HTTP POST https://endpoint/things/thingName/shadow?name=shadowName
Request body: request state document
```

未命名的（经典）影子不需要使用 name 查询参数。

有关更多信息，请参阅请求状态文档示例 (p. 628)。

响应

请求成功后，响应将包括标准的 HTTP 标头以及以下代码和正文：

```
HTTP 200
Response body: response state document
```

有关更多信息，请参阅响应状态文档示例 (p. 628)。

Authorization

要更新影子，需要一项允许调用方执行 iot:UpdateThingShadow 操作的策略。Device Shadow 服务接受两种形式的身份验证：使用 IAM 凭证的 Signature Version 4 或使用客户端证书的 TLS 双向身份验证。

以下是允许调用方更新设备的影子的示例策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:UpdateThingShadow",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

DeleteThingShadow

删除指定事物的影子。

请求

该请求包括标准的 HTTP 标头以及以下 URI：

```
HTTP DELETE https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

未命名的（经典）影子不需要使用 name 查询参数。

响应

请求成功后，响应将包括标准的 HTTP 标头以及以下代码和正文：

```
HTTP 200
Response body: Empty response state document
```

请注意，删除影子不会将其版本号重置为 0。

Authorization

要删除设备的影子，需要一项允许调用方执行 iot:DeleteThingShadow 操作的策略。Device Shadow 服务接受两种形式的身份验证：使用 IAM 凭证的 Signature Version 4 或使用客户端证书的 TLS 双向身份验证。

以下是允许调用方删除设备的影子的示例策略：

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

```
      "Action": "iot:DeleteThingShadow",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

ListNamedShadowsForThing

列出指定事物的影子。

请求

该请求包括标准的 HTTP 标头以及以下 URI：

```
HTTP GET /api/things/shadow/ListNamedShadowsForThing/thingName?
nextToken=nextToken&pageSize=pageSize
Request body: (none)
```

nextToken

用于检索下一组结果的令牌。

该值在分页结果中返回，并在返回下一页的调用中使用。

pageSize

在每个调用中返回的影子名称的数量。另请参阅 nextToken。

thingName

要列出命名的影子的事物的名称。

响应

在成功后，响应将包括标准 HTTP 标头以及以下响应代码和 影子名称列表响应文档 (p. 632)。

Note

未命名的（经典）影子不会显示在该列表中。如果只有经典影子或 thingName 您指定的影子不存在，响应为空列表。

```
HTTP 200
Response body: Shadow name list document
```

Authorization

要列示设备的影子，需要一项允许调用方执行 iot:ListNamedShadowsForThing 操作的策略。Device Shadow 服务接受两种形式的身份验证：使用 IAM 凭证的 Signature Version 4 或使用客户端证书的 TLS 双向身份验证。

以下是一个示例策略，它允许调用方列出事物的命名影子：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:ListNamedShadowsForThing",
      "Resource": [
```

```
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

Device Shadow MQTT 主题

Device Shadow 服务使用保留的 MQTT 主题，以使设备和应用程序能够获取、更新或删除设备的状态信息（影子）。

要订阅影子主题并向该主题发布消息，需要获得基于主题的授权。Amazon IoT 保留向现有主题结构添加新主题的权利。为此，建议您订阅影子主题时勿使用通配符。例如，请勿在订阅时使用诸如 `$aws/things/thingName/shadow/#` 的主题筛选条件，因为与该主题筛选条件匹配的主题数量可能会随着 Amazon IoT 引入新的影子主题而增加。要了解针对这些主题发布的消息示例，请查看 与影子交互 (p. 610)。

影子可以是命名或未命名（经典）的。每个影子使用的主题仅在主题前缀上有所不同。下表显示每种影子类型使用的主题前缀。

ShadowTopicPrefix 值	影子类型
`$aws/things/thingName/shadow`	未命名的（经典）影子
`$aws/things/thingName/shadow/name/shadowName`	命名的影子

要创建完整的主题，请为要表示的影子类型选择 *ShadowTopicPrefix*，将 *thingName* 和 *shadowName*（如果适用）替换为相应的值，然后在其后面附加主题存根，如以下几部分中所示。

以下是用于与影子交互的 MQTT 主题。

主题

- /get (p. 620)
- /get/accepted (p. 621)
- /get/rejected (p. 621)
- /update (p. 622)
- /update/delta (p. 623)
- /update/accepted (p. 624)
- /update/documents (p. 624)
- /update/rejected (p. 625)
- /delete (p. 626)
- /delete/accepted (p. 626)
- /delete/rejected (p. 627)

/get

要获得设备的影子，请在此主题下发布一条空消息：

```
ShadowTopicPrefix/get
```

Amazon IoT 通过向 /get/accepted (p. 621) 或 /get/rejected (p. 621) 发布消息来进行响应。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get"
      ]
    }
  ]
}
```

/get/accepted

在返回设备的影子时，Amazon IoT 将响应影子文档发布到该主题：

```
ShadowTopicPrefix/get/accepted
```

有关更多信息，请参阅响应状态文档 (p. 628)。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/accepted"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get/accepted"
      ]
    }
  ]
}
```

/get/rejected

Amazon IoT 在无法返回设备的影子时向此主题发布错误响应文档：

```
ShadowTopicPrefix/get/rejected
```

有关更多信息，请参阅错误响应文档 (p. 631)。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/get/rejected"
      ]
    },
    {
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get/rejected"
      ]
    }
  ]
}
```

/update

向此主题发布请求状态文档来更新设备的影子：

```
ShadowTopicPrefix/update
```

消息正文包含部分请求状态文档 (p. 628)。

尝试更新设备状态的客户端将发送具有 `desired` 属性的 JSON 请求状态文档，例如：

```
{
  "state": {
    "desired": {
      "color": "red",
      "power": "on"
    }
  }
}
```

更新其影子的设备将发送具有 `reported` 属性的 JSON 请求状态文档，例如：

```
{
  "state": {
    "reported": {
      "color": "red",
      "power": "on"
    }
```

```
  }
}
```

Amazon IoT 通过向 /update/accepted (p. 624) 或 /update/rejected (p. 625) 发布消息来进行响应。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update"
      ]
    }
  ]
}
```

/update/delta

在接受设备的影子更改时，Amazon IoT 将响应状态文档发布到该主题，并且请求状态文档包含不同的 `desired` 和 `reported` 状态值：

```
ShadowTopicPrefix/update/delta
```

消息缓冲区包含一个 /delta 响应状态文档 (p. 629)。

消息正文详细信息

- 发布到 `update/delta` 的消息仅包括 `desired` 部分和 `reported` 部分之间有所不同的预期属性。无论这些属性包含在当前更新消息中还是已存储在 Amazon IoT 中，它将包含所有此类属性。`desired` 部分和 `reported` 部分之间相同的属性则不包含在内。
- 如果某个属性位于 `reported` 部分，但在 `desired` 部分没有等效值，则不会包含在内。
- 如果某个属性位于 `desired` 部分，但在 `reported` 部分没有等效值，则将包含在内。
- 如果某个属性已从 `reported` 部分删除，但仍存在于 `desired` 部分，则将包含在内。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/delta"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/delta"
      ]
    }
  ]
}
```

/update/accepted

Amazon IoT 在接受设备影子的更改时向此主题发布响应状态文档：

```
ShadowTopicPrefix/update/accepted
```

消息缓冲区包含一个 /accepted 响应状态文档 (p. 629)。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/
accepted"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/accepted"
      ]
    }
  ]
}
```

/update/documents

每次影子更新成功执行时，Amazon IoT 都会向该主题发布状态文档：

```
ShadowTopicPrefix/update/documents
```

消息正文包含一个 /documents 响应状态文档 (p. 629)。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/
documents"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/documents"
      ]
    }
  ]
}
```

/update/rejected

Amazon IoT 在拒绝设备影子的更改时向此主题发布错误响应文档：

```
ShadowTopicPrefix/update/rejected
```

消息正文包含一个错误响应文档 (p. 631)。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/update/
rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
```

```
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update/rejected"
      ]
    }
  ]
}
```

/delete

要删除设备的影子，请将一条空消息发布到删除主题：

```
ShadowTopicPrefix/delete
```

消息内容将被忽略。

请注意，删除影子不会将其版本号重置为 0。

Amazon IoT 通过向 /delete/accepted (p. 626) 或 /delete/rejected (p. 627) 发布消息来进行响应。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete"
      ]
    }
  ]
}
```

/delete/accepted

Amazon IoT 在删除设备的影子时向此主题发布消息：

```
ShadowTopicPrefix/delete/accepted
```

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/delete/
accepted"
```

```
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete/accepted"
      ]
    }
  ]
}
```

/delete/rejected

Amazon IoT 在无法删除设备的影子时向此主题发布错误响应文档：

```
ShadowTopicPrefix/delete/rejected
```

消息正文包含一个错误响应文档 (p. 631)。

示例 policy

以下是所需策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/things/thingName/shadow/delete/
rejected"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/delete/rejected"
      ]
    }
  ]
}
```

Device Shadow 服务文档

Device Shadow 服务遵守 JSON 规范下的所有规则。值、对象和数组均存储在设备的影子文档中。

目录

- 影子文档示例 (p. 628)

- 文档属性 (p. 632)
- 增量状态 (p. 633)
- 对影子文档进行版本控制 (p. 634)
- 影子文档中的客户端令牌 (p. 634)
- 空影子文档属性 (p. 634)
- 影子文档中的数组值 (p. 635)

影子文档示例

在使用 REST API (p. 616) 或 MQTT 发布/订阅消息 (p. 620)的 UPDATE、GET 和 DELETE 操作中，Device Shadow 服务使用以下文档。

示例

- 请求状态文档 (p. 628)
- 响应状态文档 (p. 628)
- 错误响应文档 (p. 631)
- 影子名称列表响应文档 (p. 632)

请求状态文档

请求状态文档具有以下格式：

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "clientToken": "token",
    "version": version
}
```

- `state` — 更新仅影响指定字段。通常，您将使用 `desired` 或 `reported` 属性，但不能在同一请求中同时使用这两个属性。
  - `desired` — 请求在设备中更新的状态属性和值。
  - `reported` — 设备报告的状态属性和值。
- `clientToken` — 如果使用，您可以通过客户端令牌匹配请求和相应的响应。
- `version` - 如果使用，仅当指定的版本与 Device Shadow 服务拥有的最新版本相符时，该服务才会处理更新。

响应状态文档

响应状态文档具有以下格式，具体取决于响应类型。

/accepted 响应状态文档

```
{
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

/delta 响应状态文档

```
{
    "state": {
        "attribute1": integer2,
        "attribute2": "string2",
        ...
        "attributeN": boolean2
        }
    },
    "metadata": {
        "attribute1": {
            "timestamp": timestamp
        },
        "attribute2": {
            "timestamp": timestamp
        },
        ...
        "attributeN": {
            "timestamp": timestamp
        }
    },
    "timestamp": timestamp,
    "clientToken": "token",
    "version": version
}
```

/documents 响应状态文档

```
{
  "previous" : {
    "state": {
```

```
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer1,
            "attribute2": "string1",
            ...
            "attributeN": boolean1
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version-1
},
"current": {
    "state": {
        "desired": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        },
        "reported": {
            "attribute1": integer2,
            "attribute2": "string2",
            ...
            "attributeN": boolean2
        }
    },
    "metadata": {
        "desired": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
```

```
                "timestamp": timestamp
            }
        },
        "reported": {
            "attribute1": {
                "timestamp": timestamp
            },
            "attribute2": {
                "timestamp": timestamp
            },
            ...
            "attributeN": {
                "timestamp": timestamp
            }
        }
    },
    "version": version
  },
  "timestamp": timestamp,
  "clientToken": "token"
}
```

响应状态文档属性

- previous — 成功更新后，包含对象在更新之前的 state。
- current — 成功更新后，包含对象在更新之后的 state。
- state
  - reported — 只有在事物报告了 reported 部分中的任何数据时才存在，并且仅包含请求状态文档中的字段。
  - desired — 只有在设备报告了 desired 部分中的任何数据时才存在，并且仅包含请求状态文档中的字段。
  - delta — 只有在 desired 数据与影子的当前 reported 数据不同时才存在。
- metadata — 包含 desired 和 reported 部分中每个属性的时间戳，因此，您可以确定状态的更新时间。
- timestamp – Amazon IoT 生成响应时的新纪元日期和时间。
- clientToken — 只有在向 /update 主题发布有效 JSON 时使用了客户端令牌时才存在。
- version – Amazon IoT 中共享的设备影子文档的当前版本。它将在文档先前版本号的基础上增加一个数。

错误响应文档

错误响应文档具有以下格式：

```
{
    "code": error-code,
    "message": "error-message",
    "timestamp": timestamp,
    "clientToken": "token"
}
```

- code — 用于表明错误类型的 HTTP 响应代码。
- message — 用于提供额外信息的文本消息。
- timestamp – Amazon IoT 生成响应时的日期和时间。此属性并非在所有错误响应文档中都存在。
- clientToken — 只有在发布的消息中使用了客户端令牌时才存在。

有关更多信息，请参阅Device Shadow 错误消息 (p. 636)。

影子名称列表响应文档

影子名称列表响应文档具有以下格式：

```
{
    "results": [
        "shadowName-1",
        "shadowName-2",
        "shadowName-3",
        "shadowName-n"
    ],
    "nextToken": "nextToken",
    "timestamp": timestamp
}
```

- `results` — 影子名称数组。
- `nextToken` — 在获取序列中的下一页的分页请求中使用的令牌值。在没有其它要返回的影子名称时，该属性不存在。
- `timestamp` – Amazon IoT 生成响应时的日期和时间。

文档属性

设备的影子文档具有以下属性：

`state`

`desired`

设备的所需状态。应用程序可以写入到文档的该部分以直接更新设备的状态，而无需连接到设备。

`reported`

设备的报告状态。设备写入到文档的该部分以报告其新状态。应用程序读取文档的该部分以确定设备的上次报告状态。

`metadata`

有关存储在文档 `state` 部分的数据的信息，其中包括 `state` 部分中每个属性的时间戳（以 Epoch 时间表示），让您能够确定它们的更新时间。

Note

元数据不会影响服务限制或定价的文档大小。有关更多信息，请参阅 Amazon IoT Service Limits。

`timestamp`

指示 Amazon IoT 发送消息的时间。通过使用消息中的时间戳以及 `desired` 或 `reported` 部分中各个属性的时间戳，设备可以确定属性的存在时间，即使设备没有内部时钟。

`clientToken`

特定于设备的字符串，让您能在 MQTT 环境中将响应与请求关联起来。

`version`

文档版本。文档每次更新时，此版本号都会递增。用于确保正在更新的文档为最新版本。

有关更多信息，请参阅影子文档示例 (p. 628)。

增量状态

增量状态是一种虚拟类型的状态，包含 desired 状态和 reported 状态之间的差异。对于 desired 部分中的字段，如果 reported 部分没有这些字段，则它们将包含在增量中。对于 reported 部分中的字段，如果 desired 部分没有这些字段，则它们不会包含在增量中。增量包含元数据，且其值与 desired 字段的元数据相同。例如：

```
{
  "state": {
    "desired": {
      "color": "RED",
      "state": "STOP"
    },
    "reported": {
      "color": "GREEN",
      "engine": "ON"
    },
    "delta": {
      "color": "RED",
      "state": "STOP"
    }
  },
  "metadata": {
    "desired": {
      "color": {
        "timestamp": 12345
      },
      "state": {
        "timestamp": 12345
      }
      },
      "reported": {
        "color": {
          "timestamp": 12345
        },
        "engine": {
          "timestamp": 12345
        }
      },
      "delta": {
        "color": {
          "timestamp": 12345
        },
        "state": {
          "timestamp": 12345
        }
      }
    },
    "version": 17,
    "timestamp": 123456789
  }
}
```

当嵌套对象不同时，增量将包含根路径。

```
{
  "state": {
    "desired": {
      "lights": {
        "color": {
          "r": 255,
          "g": 255,
```

```
          "b": 255
        }
      }
    },
    "reported": {
      "lights": {
        "color": {
          "r": 255,
          "g": 0,
          "b": 255
        }
      }
    },
    "delta": {
      "lights": {
        "color": {
          "g": 255
        }
      }
    }
  },
  "version": 18,
  "timestamp": 123456789
}
```

Device Shadow 服务通过循环访问 `desired` 状态中的每个字段并将其与 `reported` 状态进行比较来计算增量。

数组的处理方式与值类似。如果 `desired` 部分中的数组与 `reported` 部分中的数组不匹配，则整个预期数组将被复制到增量中。

对影子文档进行版本控制

Device Shadow 服务支持在每个更新消息（请求和响应）中进行版本控制。这意味着，每次更新影子时，JSON 文档的版本将会增加。这可以确保两件事情：

- 如果客户端尝试使用旧版本号覆盖影子，它将收到错误消息。客户端将被告知必须先进行重新同步，然后才能更新设备的影子。
- 如果消息的版本比客户端存储的版本低，客户端则可决定不对该消息执行操作。

客户端可以在影子文档中不包含版本以绕过版本匹配。

影子文档中的客户端令牌

收发基于 MQTT 的消息时，您可以使用客户端令牌，以验证请求和请求响应是否包含相同的客户端令牌。这可以确保响应与请求相互关联。

> **Note**
>
> 客户端令牌不能超过 64 字节。超过 64 字节的客户端令牌将导致 400（错误请求）响应和 Invalid clientToken (无效的客户端令牌) 错误消息。

空影子文档属性

如果影子文档中的 `reported` 和 `desired` 属性不适用于当前影子状态，它们可能为空或省略。例如，只有在影子文档具有所需状态时，它才包含 `desired` 属性。以下是没有 `desired` 属性的状态文档的有效示例：

```
{
```

```
    "reported" : { "temp": 55 }
}
```

reported 属性也可能为空，例如，设备尚未更新影子时：

```
{
    "desired" : { "color" : "RED" }
}
```

如果更新导致 desired 或 reported 属性变为 null，则会将其从文档中删除。下面说明了如何将 desired 属性设置为 null 以删除该属性。例如，在设备更新其状态时，您可以执行该操作。

```
{
    "state": {
        "reported": {
            "color": "red"
        },
        "desired": null
    }
}
```

影子文档也可能没有 desired 或 reported 属性，从而使影子文档为空。这是一个空影子（但有效）文档的示例。

```
{
}
```

影子文档中的数组值

影子支持数组，但将其视为正常值进行处理，因为对数组的更新将替换整个数组。无法更新数组的某个部分。

初始状态：

```
{
    "desired" : { "colors" : ["RED", "GREEN", "BLUE" ] }
}
```

更新：

```
{
    "desired" : { "colors" : ["RED"] }
}
```

最终状态：

```
{
    "desired" : { "colors" : ["RED"] }
}
```

数组不能包含空值。例如，以下数组无效并将被拒绝。

```
{
    "desired" : {
        "colors" : [ null, "RED", "GREEN" ]
```

```
    }
}
```

Device Shadow 错误消息

Device Shadow 服务在尝试更改状态文档失败时向错误主题发布消息 (通过 MQTT)。此消息仅将作为对发布到其中一个保留的 $aws 主题的请求的响应。如果客户端使用 REST API 来更新文档，则客户端将收到作为响应一部分的 HTTP 错误代码，且不会发送任何 MQTT 错误消息。

HTTP 错误代码	错误消息
400（错误请求）	• JSON 无效 • 必需节点缺失：状态 • 状态节点必须是对象 • 预期节点必须是对象 • 报告节点必须是对象 • 版本无效 • clientToken 无效 Note 超过 64 字节的客户端令牌将引发此响应。 • JSON 包含的嵌套层级过多；最多嵌套 6 个层级 • 状态包含无效节点
401（未授权）	• Unauthorized
403（禁止访问）	• 禁止
404（未找到）	• 事物未找到 • 不存在名为 *shadowName* 的影子
409（冲突）	• 版本冲突
413（负载过大）	• 负载超出允许的最大值
415（媒体类型不受支持）	• 文档编码不受支持；受支持的编码是 UTF-8
429（请求过多）	• 如果单一连接中正在传输的请求超过 10 个，则 Device Shadow 服务将生成此条错误消息。动态请求是已启动但尚未完成的进行中的请求。
500（内部服务器错误）	• 内部服务故障

任务

使用 Amazon IoT Jobs 定义一组远程操作，这些操作被发送到一个或多个连接到 Amazon IoT 的设备并在这些设备上运行。例如，您可以定义一个任务，该任务指示一组设备下载并安装应用程序或固件更新、重启、轮换证书或执行远程故障排除操作。

访问 Amazon IoT 任务

您可以使用控制台或 Amazon IoT Core API 来开始使用 Amazon IoT Jobs。

使用控制台

登录 Amazon Web Services Management Console，然后转至 Amazon IoT 控制台。在导航窗格中，选择 Manage（管理），然后选择 Jobs（任务）。您可以从此部分创建和管理任务。如果要创建和管理任务模板，请在导航窗格中，选择 Job templates（任务模板）。有关更多信息，请参阅使用 Amazon Web Services Management Console 创建和管理任务。 (p. 645)：

使用 API 或 CLI

您可以通过 Amazon IoT Core API 操作来开始此过程。有关更多信息，请参阅 Amazon IoT API 参考。用于构建 Amazon IoT 任务的 Amazon IoT Core API 由 Amazon SDK 提供支持。有关更多信息，请参阅 Amazon SDK 和工具包。

您可以使用 Amazon CLI 运行用于创建和管理任务和任务模板的命令。有关更多信息，请参阅 Amazon IoT CLI 参考。

Amazon IoT Jobs 区域和终端节点

Amazon IoT Jobs 支持特定于您的 Amazon Web Services 区域的控制面板和数据层面 API 终端节点。数据层面 API 终端节点特定于您的 Amazon Web Services 账户 和 Amazon Web Services 区域。有关 Amazon IoT Jobs 终端节点的更多信息，请参阅《Amazon 一般参考》中的 Amazon IoT Device Management - 任务数据终端节点。

Amazon IoT Jobs 是什么？

使用 Amazon IoT Jobs 定义一组远程操作，这些操作被发送到一个或多个连接到 Amazon IoT 的设备并在这些设备上运行。

要创建任务，首先定义任务文档，其中包含描述设备必须远程执行的操作的说明列表。要执行这些操作，请指定目标列表，这些都是单独事物、事物组 (p. 246)，或同时选择两者。任务文件和目标共同构成了部署。

每个部署都可以有额外的配置：

- Rollout（推出）：此配置定义了每分钟有多少设备接收任务文档。
- Abort（中止）：如果一定数量的设备没有收到任务通知，则可使用此配置取消任务，避免向整个机群发送错误的更新。
- Timeout（超时）：如果在一定时间内没有从您的工作目标收到响应，任务可能会失败。您可以跟踪在这些设备上运行的任务。
- Retry（重试）：如果设备报告失败或任务超时，则可使用 Amazon IoT Jobs 自动将任务文档重新发送到设备。

- S@@ ch eduling（计划）：此配置使您能够为future 日期和时间安排任务，并创建周期性维护窗口，使您能够为未来的日期和时间安排任务。

Amazon IoT Jobs 会发送消息，通知目标任务可用。目标通过以下方式开始执行 任务：下载任务文档、执行它指定的操作并向 Amazon IoT 报告其进度。通过运行 Amazon IoT Jobs 提供的命令，可以跟踪特定目标和所有目标的任务的进度。任务开始时，状态为正在进行中。然后，设备在显示此状态的同时报告增量更新，直到任务成功、失败或超时。

以下主题描述任务的一些关键概念以及任务和任务执行的生命周期。

主题

- 任务关键概念 (p. 638)
- 任务和任务执行状态 (p. 640)

任务关键概念

以下概念可以帮助您更多地了解 Amazon IoT Jobs，以及如何创建和部署任务以在设备上运行远程操作。

基本概念

以下是使用 Amazon IoT Jobs 时需要了解的基本概念。

任务

任务是被发送到一台或多台连接到 Amazon IoT 的设备并在这些设备上运行的远程操作。例如，您可以定义一个任务，该任务指示一组设备下载并安装应用程序或运行固件更新、重启、轮换证书或执行远程故障排除操作。

任务文档

要创建任务，您必须先创建一个任务文档，该文档是要由设备执行的远程操作的描述。

任务文档是 UTF-8 编码的 JSON 文档，其中包含您的设备执行任务所需的信息。任务文档包含一个或多个 URL，设备可从这些 URL 下载更新或其它数据。任务文档可存储在 Amazon S3 存储桶中，或者随用于创建任务的命令内联包含。

Tip

有关任务文档示例，请参阅Amazon IoT SK k p 示例 JavaScript。 jobs-agent.js

目标

在创建任务时，您需要指定一系列目标，它们是应执行操作的设备。目标可以是事物和/或事物组 (p. 246)。Amazon IoT Jobs 会向每个目标发送一条消息以向该目标告知任务可用。

部署

通过提供任务文档并指定目标列表创建任务之后，系统会将任务文档部署到要为其执行更新的远程目标设备上。对于快照任务，任务会在部署到目标设备后完成。对于持续任务，任务会在设备添加到组时部署到该组设备。

任务执行

任务执行是目标设备上的任务的实例。目标通过下载任务文档来开始执行任务。目标随后会执行在文档中指定的操作并将其进度报告给 Amazon IoT。执行编号是特定目标上的任务执行的唯一标识符。Amazon IoT Jobs 服务提供的命令可跟踪目标上任务执行的进度以及某个任务跨所有目标的进度。

任务类型概念

以下概念可帮助您更多地了解可以使用 Amazon IoT Jobs 创建的不同类型的任务。

快照任务

预设情况下，任务将发送到您在创建该任务时指定的所有目标。在这些目标完成任务 (或报告它们无法执行此操作) 后，任务即完成。

持续任务

持续的任务将发送到您在创建该任务时指定的所有目标。它继续运行并发送到添加到目标组的任何新设备（事物）。例如，在将设备添加到组时，可使用持续任务加入或升级设备。您可以通过在创建作业时设置可选参数，来使作业成为持续作业。

Note

当使用动态事物组确定目标 IoT 机群时，我们建议您使用连续任务而不是快照任务。通过使用连续任务，加入组的设备即使在任务创建之后也会收到任务执行。

预签名 URL

为了安全、有时间限制地访问任务文档中未包含的数据，可以使用预签名的Amazon S3 URL。将数据放置在 Amazon S3 存储桶中，并添加一个指向任务文档中的数据的占位符链接。当 Amazon IoT Jobs 收到任务文档的请求时，它会解析任务文档以查找占位符链接并将其替换为预签名 Amazon S3 URL。

占位符链接采用以下格式：

```
${aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket/key}
```

其中 *bucket* 是您的存储桶名称，*key* 是您链接到的存储桶中的对象。

在北京和宁夏区域中，仅当资源拥有者具有 ICP（互联网内容提供商）许可证时，预签名 URL 才起作用。有关更多信息，请参阅在中国的Amazon服务入门文档中的 Amazon Simple Storage Service。

任务配置概念

以下概念可以帮助您了解如何配置任务。

推出

您可以指定向目标发送待处理任务执行通知的速度。这样，您就可以创建一个分段推出以更好地管理更新、重启和其它操作。您可以使用静态推出率或指数级推出率来创建部署配置。您可以使用静态推出速率来指定每分钟可通知的最大任务目标数。

有关设置推出速率的示例，以及有关配置任务推出的更多信息，请参阅任务推出、计划和中止配置 (p. 672)。

计划

通过任务计划，您可以计划向目标组中的所有设备推出连续任务和快照任务的任务文档的时间。此外，您可以您可以您可以

超时

只要任务部署卡在 IN_PROGRESS 状态的时间超出预期，您就会收到任务超时通知。有两种类型的计时器：进行中计时器和步骤计时器。当任务为 IN_PROGRESS 时，您可以监控和跟踪任务部署的进度。

推出和中止配置特定于您的任务，而超时配置特定于任务部署。有关更多信息，请参阅任务执行超时和重试配置 (p. 677)：

重试

通过任务重试，可以在任务失败和/或超时时重试任务执行。您最多可以重试 10 次任务执行。您可以监控和跟踪重试的进度以及任务执行是否成功。

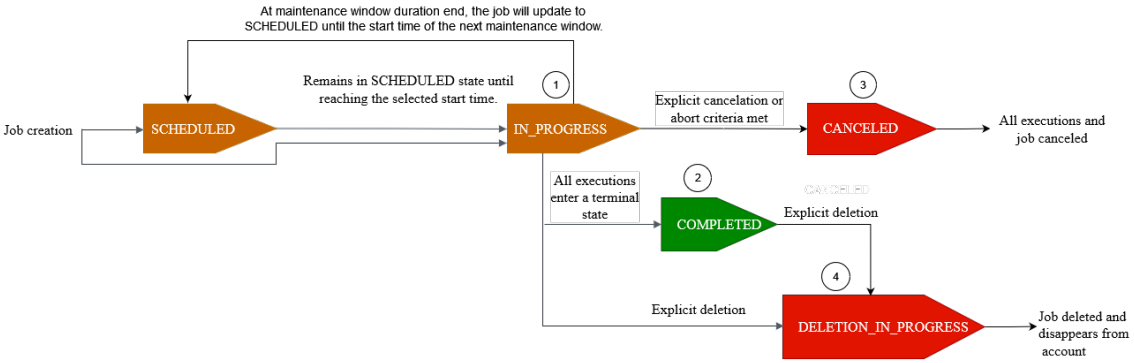
推出和中止配置特定于您的任务，而超时和重试配置特定于任务执行。有关更多信息，请参阅任务执行超时和重试配置 (p. 677)：

任务和任务执行状态

以下各个部分介绍 Amazon IoT 任务的生命周期和任务执行的生命周期。

任务状态

下图演示了 Amazon IoT 任务的不同状态。



使用 Amazon IoT Jobs 创建的任务可以处于以下状态之一：

- 已计划

  在使用Amazon IoT控制台、CreateJobAPI 或 API 初始创建任务或任务模板期间，您可以在Amazon IoT控制台或 CreateJobTemplateAPI 或 CreateJobCreateJobTemplateAPI 中选择可选的调度配置。SchedulingConfig当您启动包含特定 startTime、endTime 和 endBehavoir 的计划任务时，任务状态将更新为 SCHEDULED。当任务达到您选择的startTime，startTime或者如果您选择了任务推出时，状态将从更新为，IN_PROGRESS并开始SCHEDULED向目标组中的所有设备推出任务文档。
- IN_PROGRESS

  当您使用Amazon IoT控制台或 CreateJobAPI 创建任务时，任务状态更新为IN_PROGRESS。在任务创建期间，Amazon IoT Jobs 开始向目标组中的设备推出任务执行。一旦所有任务执行都已推出，Amazon IoT Jobs 就会等待设备完成远程操作。

  有关适用于进行中任务的并发和限制的信息，请参阅 任务限制 (p. 730)。

  Note

  当IN_PROGRESS任务达到当前维护时段的结束时，任务文档的发布将停止。该作业将更新到，SCHEDULEDstartTime直到下一个维护时段结束。
- COMPLETED

连续任务通过方式方式：

- 对于未选择可选计划配置的连续任务，该任务始终处于进行中，并继续针对添加到目标组的任何新设备运行。它永远不会达到状态为COMPLETED。
- 对于选择了可选调度配置的连续作业，以下情况成立：
  - 如果提供endTime了，则连续作业将在结束且所有作业执行都达到终端COMPLETED状态时endTime达到状态。
  - 如果在可选的调度配置中未提供，则连续作业将继续执行任务文档的发布。endTime

对于快照任务，当其所有任务执行都进入最终状态（如 SUCCEEDED、FAILED、TIMED_OUT、REMOVED 或 CANCELED）时，任务状态更改为 COMPLETED。

- CANCELED

  当您使用Amazon IoT控制台、CancelJobAPI 或取消任务时 任务中止配置 (p. 677)，任务状态将更改为CANCELED。在取消任务期间，Amazon IoT Jobs 开始取消之前创建的任务执行。

  有关适用于正在取消的任务的并发和限制的信息，请参阅 任务限制 (p. 730)。

- DELETION_IN_PROGRESS

  当您使用Amazon IoT控制台或 DeleteJobAPI 删除任务时，任务状态更改为DELETION_IN_PROGRESS。在任务删除期间，Amazon IoT Jobs 开始删除之前创建的任务执行。删除所有任务执行后，该任务将从您的 Amazon 账户中消失。

任务执行状态

下表显示了 Amazon IoT 任务执行的不同状态，以及状态更改是由设备发起还是由 Amazon IoT Jobs 发起。

任务执行状态和源

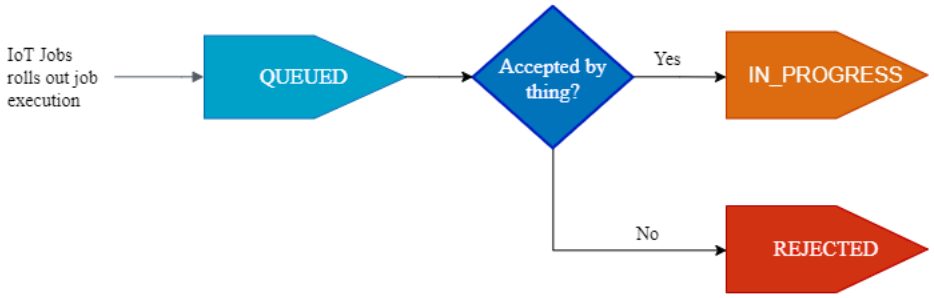
任务执行状态	由设备发起？	由 Amazon IoT Jobs 发起？	终端状态？	是否可以重试？
QUEUED	否	是	否	不适用
IN_PROGRESS	是	否	否	不适用
SUCCEEDED	是	否	是	不适用
FAILED	是	否	是	是
TIMED_OUT	否	是	是	是
REJECTED	是	否	是	否
REMOVED	否	是	是	否
CANCELED	否	是	是	否

以下部分详细介绍了当您使用 Amazon IoT Jobs 创建任务时推出的任务执行的状态。

- QUEUED

  当 Amazon IoT Jobs 为目标设备推出任务执行时，任务执行状态设置为 QUEUED。任务执行保留为 QUEUED 状态，直到：
  - 您的设备收到任务执行，调用任务 API 操作并将状态报告为 IN_PROGRESS。
  - 您取消任务或任务执行，或者当满足您使用中止配置指定的中止条件且状态更改为 CANCELED 时。

- 您的设备已从目标组中移除且状态更改为 REMOVED。



- IN_PROGRESS

  如果您的 IoT 设备订阅了保留的任务主题 (p. 101) $notify 和 $notify-next，并且您的设备调用 StartPendingJobExecution API 或 UpdateJobExecution API 且状态为 IN_PROGRESS，则 Amazon IoT Jobs 会将任务执行状态设置为 IN_PROGRESS。

  UpdateJobExecution API 可以多次调用，状态为 IN_PROGRESS。您可以使用 statusDetails 对象指定有关执行步骤的其他详细信息。

  Note

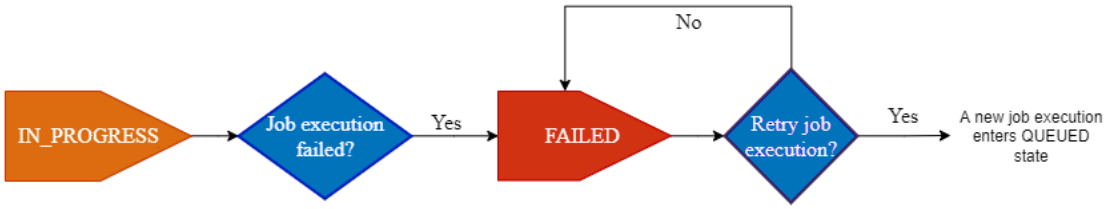
  如果您为每个设备创建多个任务，Amazon IoT Jobs 和 MQTT 协议不能保证交付顺序。

- SUCCEEDED

  当您的设备成功完成远程操作时，设备必须调用 UpdateJobExecution API 且状态为 SUCCEEDED，以表示任务执行获得成功。然后，Amazon IoTJobs 将任务执行状态更新并返回为 SUCCEEDED。

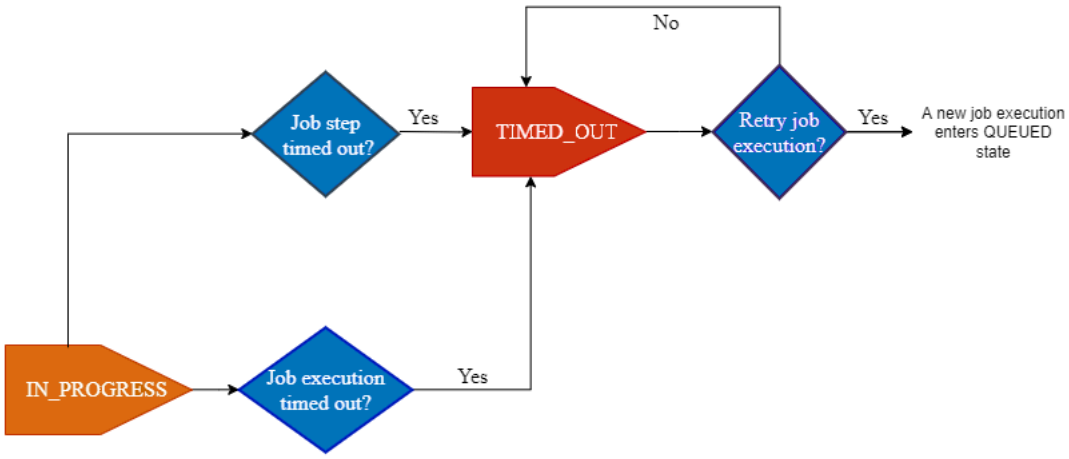
- FAILED

  当您的设备未能完成远程操作时，设备必须调用 UpdateJobExecution API 且状态为 Failed，以表示任务执行失败。然后，Amazon IoTJobs 将任务执行状态更新并返回为 Failed。您可以使用任务执行重试配置 (p. 678)重试设备的这一任务执行。



- TIMED_OUT

  如果您的设备在状态为 IN_PROGRESS 时无法完成任务步骤，或在进行中计时器的超时持续时间内无法完成远程操作，Amazon IoT Jobs 会将任务执行状态设置为 TIMED_OUT。您还有一个用于正在进行的任务的每个任务步骤的步骤计时器，并且仅适用于任务执行。正在进行的计时器持续时间是使用 任务执行超时配置 (p. 677) 的 inProgressTimeoutInMinutes 属性指定的。您可以使用任务执行重试配置 (p. 678)重试设备的这一任务执行。



- REJECTED

  当您的设备收到无效或不兼容的请求时，设备必须调用 `UpdateJobExecution` API 且状态为 `REJECTED`。然后，Amazon IoT然后，Jobs 将任务执行状态更新并返回为 `REJECTED`。

- REMOVED

  当设备不再是任务执行的有效目标时，例如当它脱离动态事物组时，Amazon IoT Jobs 将任务执行状态设置为 `REMOVED`。您可以将事物重新附加到目标组，然后重新启动设备的任务执行。

- CANCELED

  当您使用控制台、`CancelJob` 或 `CancelJobExecution` API 取消任务或取消任务执行时，或当满足使用 任务中止配置 (p. 677) 指定的中止条件时，Amazon IoT Jobs 将取消任务并将任务执行状态设置为 `CANCELED`。

管理任务

使用任务通知设备软件或固件更新。您可以使用 Amazon IoT 控制台、任务管理和控制 API 操作 (p. 701)Amazon Command Line Interface 或 Amazon SDK 来创建和管理任务。

任务的代码签名

向设备发送代码时，为了让设备检测代码是否在传输过程中被修改，建议您使用 Amazon CLI 对代码文件进行签名。有关说明，请参阅使用 Amazon CLI 创建和管理任务 (p. 647)。

有关更多信息，请参阅什么是 Amazon IoT 的代码签名？

任务文档

在创建任务之前，您必须创建任务文档。如果您使用 Amazon IoT 的代码签名，则必须将任务文档上载到受版本控制的 Amazon S3 存储桶。有关创建 Amazon S3 存储桶并将向其上载文件的信息，请参阅 Amazon S3 入门指南中的 Amazon Simple Storage Service 入门。

> **Tip**
>
> 有关任务文档示Amazon IoT例： JavaScriptjobs-agent.js

预签名 URL

您的任务文档可以包含指向您的代码文件（或其它文件）的预签名 Amazon S3 URL。预签名 Amazon S3 URL 的有效期有限，因此在设备请求任务文档时才生成。因为在您创建任务文档时尚未创建预签名 URL，所以在您的任务文档中使用占位符 URL。占位符 URL 类似如下所示：

```
${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/<bucket>/<code
file>}
```

其中：

- *###*是包含代码文件的 Amazon S3 存储桶。
- *####*是代码文件的 Amazon S3 密钥。

当设备请求任务文档时，Amazon IoT 会生成预签名 URL 并使用预签名 URL 替换占位符 URL。然后将您的任务文档发送到设备。

IAM 角色，用于授予从 S3 下载文件的权限

在创建使用预签名 Amazon S3 URL 的任务时，您必须提供一个 IAM 角色，该角色可授予从存储数据或更新的 Amazon S3 存储桶中下载文件的权限。该角色还必须向 Amazon IoT 授予代入角色的权限。

您可以为预签名 URL 指定可选的超时。有关更多信息CreateJob。

向 Amazon IoT Jobs 授予代入您的角色的权限

1. 转到 IAM 控制台的角色中心，然后选择您的角色。
2. 在信任关系选项卡上，选择 Edit Trust Relationship（编辑信任关系），用以下 JSON 替换当前策略文档。选择 Update Trust Policy（更新信任策略）。

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

3. 为防止出现混淆代理人问题，请在策略中添加全局条件键 aws:SourceArn 和 aws:SourceAccount。

   Important

   您的 aws:SourceArn 必须符合此格式：arn:aws:iot:*region*:*account-id*:*。确保*#*#与您的 Amazon IoT 区域匹配，*account-id*与您的客户账户 ID 相匹配。有关更多信息，请参阅防止跨服务混淆代理 (p. 308)。

```
{
  "Effect": "Allow",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Principal": {
        "Service":
          "iot.amazonaws.com"
       },
      "Action": "sts:AssumeRole",
      "Condition": {
         "StringEquals": {
            "aws:SourceAccount": "123456789012"
          },
          "ArnLike": {
               "aws:SourceArn": "arn:aws:iot:*:123456789012:job/*"
          }
       }
     }
   ]
}
```

4. 如果您的任务使用的任务文档是一个 Amazon S3 对象，请选择 Permissions（权限），然后使用以下 JSON 添加策略，授予从您的 Amazon S3 存储桶下载文件的权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::your_S3_bucket/*"
        }
    ]
}
```

主题

- 使用 Amazon Web Services Management Console 创建和管理任务。 (p. 645)
- 使用 Amazon CLI 创建和管理任务。 (p. 647)

使用 Amazon Web Services Management Console 创建和管理任务。

创建 任务

1. 登录 Amazon Web Services Management Console，然后登录 Amazon IoT 控制台。
2. 在左侧导航窗格中的“Manage”（管理）部分下，选择 Remote Actions（远程操作），然后选择 Jobs（任务）。
3. 在 Jobs（任务）对话框的 Jobs（任务）页面上，选择 Create job（创建任务）。
4. 根据您使用的设备，您可以创建自定义任务、FreeRTOS OTA 更新任务或 Amazon IoT Greengrass 任务。在此示例中，选择 Create a custom job（创建自定义任务）。选择下一步。
5. 在 Job properties（任务属性）对话框的 Custom job properties（自定义任务属性）页面上，为以下字段输入您的信息：
   - Name（名称）：输入唯一的字母数字任务名称。
   - Description - optional（描述 - 可选）：输入有关您的任务的可选描述。
   - Tags - optional（标签 - 可选）：

Note

我们建议不要在您的任务 ID 或描述中使用个人身份信息。

选择下一步。

6. 在 Job targets（任务目标）对话框的 File configuration（文件配置）页面上，选择要运行此任务的 Things（事务）或 Thing groups（事务组）。

   在 Job document（任务文档）对话框中，选择以下选项之一：

   - From file（源文件）：您之前上传到 Amazon S3 存储桶的 JSON 任务文件
     - 代码签名

       在 Amazon S3 URL 中的任务文档中`${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}`，必须用作占位符，直到使用您的代码签名配置文件将其替换为已签名的代码文件路径。新的签名代码文件最初将出现在您的 Amazon S3 源存储桶的`SignedImages`文件夹中。将创建包含`Codesigned_`前缀的新任务文档，使用签名的代码文件路径替换代码签名占位符，并将其放置在您的 Amazon S3 URL 中以创建新任务。
     - 预签名资源 URL

       在预签名角色下拉列表中，选择您在预签名 URL 中创建的 IAM 角色。对于从 Amazon S3 下载对象的设备来说，使用`${aws:iot:s3-presigned-url:`预先签名位于 Amazon S3 中的对象的 URL 是一种最佳安全实践。

       如果您想使用预签名 URL 作为代码签名占位符，请使用以下示例模板：

       ```
       ${aws:iot:s3-presigned-url:${aws:iot:code-sign-signature:<S3 URL>}
       ```

   - From template（源模板）：包含任务文档和任务配置的任务模板。任务模板可以是您创建的自定义任务模板或 Amazon 管理的模板。

     如果您正在创建任务执行常用的远程操作（例如重启设备），可以使用 Amazon 托管模板。这些模板已经进行了预置以供使用。有关更多信息，请参阅 创建自定义任务模板 (p. 667) 和 从托管模板创建自定义任务模板 (p. 663)。

7. 在 Job configuration（任务配置）对话框的 Job configuration（任务配置）页面上，选择以下任务类型之一：

   - Snapshot job（快照任务）：快照任务在目标设备和组上完成运行后即完成。
   - Continuous job（连续任务）：连续任务适用于事物组，并会在随后添加到指定目标组的任何设备上运行。

8. 在 Additional configurations - optional（其他配置 - 可选）对话框中，查看以下可选任务配置并做出相应的选择：

   - 推出配置
   - 计划配置
   - 任务执行超时配置
   - 任务执行重试配置 - 新增
   - 中止配置

   有关任务配置的更多信息，请参阅以下部分：

   - 任务推出、计划和中止配置 (p. 672)
   - 任务执行超时和重试配置 (p. 677)

查看您的所有任务选择，然后选择 Submit（提交）以创建任务。

在您创建任务后，控制台会生成一个 JSON 签名并将其放在您的任务文档中。您可以使用 Amazon IoT 控制台查看状态、取消或删除任务。要管理任务，请转到控制台的 Job 中心。

使用 Amazon CLI 创建和管理任务。

本部分介绍如何创建和管理任务。

创建任务

要创建 Amazon IoT 任务，使用 CreateJob 命令。任务将进行排队以便在您指定的目标 (事物或事物组) 上执行。要创建 Amazon IoT 任务，您需要一个任务文档，该文档可包含在请求的正文中或作为指向 Amazon S3 文档的链接。如果任务包含使用预签名的 Amazon S3 URL 下载文件，则您需要一个 IAM 角色 Amazon Resource Name (ARN)，它具有下载文件的权限并会向 Amazon IoT Jobs 服务授予代入角色的权限。

使用任务进行代码签名

如果您使用 Amazon IoT 的代码签名，必须启动代码签名任务并将输出包括在您的任务文档中。这将替换作业文档中的代码签名签名占位符，在使用您的代码签名配置文件将其替换为已签名的代码文件路径之前，该占位符是必需的。代码签名占位符：

```
    ${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}
```

使用start-signing-job命令创建代码签名作业。 `start-signing-job`返回任务 ID。要获取存储签名的 Amazon S3 位置，使用 describe-signing-job 命令。然后，您可以从 Amazon S3 中下载签名。有关代码签名任务的更多信息，请参阅 Amazon IoT 的代码签名。

您的任务文档必须包含代码文件的预签名 URL 占位符和使用以下内容放置在 Amazon S3 存储桶中的 JSON 签名输start-signing-job出：

```
{
    "presign": "${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/bucket/image}",
}
```

使用任务文档创建任务

以下命令说明了如何使用存储在 Amazon S3 存储桶 (j *obBucket) ###### (j ob-document.js* on) 和具有从 Amazon S3 下载文件的权限的角色 (S *3DownloadRole*) 创建任务。

```
aws iot create-job  \
      --job-id 010  \
      --targets arn:aws:iot:us-east-1:123456789012:thing/thingOne  \
      --document-source https://s3.amazonaws.com/my-s3-bucket/job-document.json  \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\": 50,
 \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000,
 \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType\":
 \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20}, { \"action
\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings\": 200,
 \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/
S3DownloadRole\", \"expiresInSec\":3600}"
```

任务在 *thingOne* 上运行。

可选的 `timeout-config` 参数指定每个设备完成其任务执行所具有的时间。计时器在任务执行状态设置为 `IN_PROGRESS` 时启动。如果任务执行状态未在时间到期之前设置为其它最终状态，则会设置为 `TIMED_OUT`。

进行中计时器无法更新，将应用到该任务的全部任务执行。只要任务执行保持在 `IN_PROGRESS` 状态的时间长度超过了此间隔，会失败，并切换为最终 `TIMED_OUT` 状态。Amazon IoT 还将发布 MQTT 通知。

有关创建任务推出和中止相关配置的更多信息，请参阅任务推出和中止配置。

> **Note**
>
> 在您创建任务时，系统会检索指定为 Amazon S3 文件的任务文档。如果在创建任务文档任务后，您更改用作任务文档源的 Amazon S3 文件的内容不会更改发送到任务目标的内容。

更新任务

要更新任务，使用 UpdateJob 命令。您可以更新任务的 `description`、`presignedUrlConfig`、`jobExecutionsRolloutConfig`、`abortConfig` 和 `timeoutConfig` 字段。

```
aws iot update-job  \
  --job-id 010  \
  --description "updated description" \
  --timeout-config inProgressTimeoutInMinutes=100 \
  --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\": 50,
 \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000,
 \"numberOfSucceededThings\": 1000}, \"maximumPerMinute\": 1000}}" \
  --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType\":
 \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20}, { \"action
\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings\": 200,
 \"thresholdPercentage\": 50}]}" \
  --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/S3DownloadRole\",
 \"expiresInSec\":3600}"
```

有关更多信息，请参阅任务推出和中止配置。

取消任务

要取消任务，使用 CancelJob 命令。取消任务将使 Amazon IoT 停止为任务推出任何新的任务执行。这还将取消处于 `QUEUED` 状态的任何任务执行。Amazon IoT 会将任何任务执行的最终状态保持不变，因为设备已完成任务。如果任务执行的状态为 `IN_PROGRESS`，它也将保持不变，除非您使用可选的 `--force` 参数。

以下命令说明如何取消 ID 为 010 的任务。

```
aws iot cancel-job --job-id 010
```

该命令将显示以下输出：

```
{
    "jobArn": "string",
    "jobId": "string",
    "description": "string"
}
```

当您取消任务时，将取消处于 QUEUED 状态的任务执行。如果您指定了可选的 --force 参数，将取消处于 IN_PROGRESS 状态的任务执行。不会取消处于最终状态的任务执行。

> Warning
>
> 取消处于 IN_PROGRESS 状态的任务（通过设置 --force 参数）将取消正在进行的任何任务执行，并且会导致运行该任务的设备无法更新任务执行状态。请谨慎使用，并且 确保执行已取消的任务的每个设备能够恢复到有效状态。

已取消任务或其某个任务执行的状态最终是一致的。Amazon IoT 将尽快停止将该任务的新任务执行和 QUEUED 任务执行安排给设备。将任务执行的状态更改为 CANCELED 可能需要一些时间，具体取决于设备数和其它因素。

如果任务因满足 AbortConfig 对象所定义的条件而被取消，服务会为 comment 和 reasonCode 字段添加自动填充值。当任务取消由用户驱动时，您可以为 reasonCode 创建自己的值。

取消任务执行

要取消设备上的任务执行，使用 CancelJobExecution 命令。它将取消处于 QUEUED 状态的任务执行。如果要取消正在进行的任务执行，您必须使用 --force 参数。

以下命令说明如何取消在 myThing 上运行的任务 010 的任务执行。

```
aws iot cancel-job-execution --job-id 010 --thing-name myThing
```

该命令不显示任何输出。

将取消处于 QUEUED 状态的任务执行。如果您指定了可选的 --force 参数，则将取消处于 IN_PROGRESS 状态的任务执行。无法取消处于最终状态的任务执行。

> Warning
>
> 取消处于 IN_PROGRESS 状态的任务执行后，设备无法更新任务执行状态。请谨慎使用，并且 确保设备能够恢复到有效状态。

如果任务执行处于最终状态，或者任务执行处于 IN_PROGRESS 状态并且 --force 参数未设置为 true，则此命令将引发 InvalidStateTransitionException。

已取消任务执行的状态最终是一致的。将任务执行的状态更改为 CANCELED 可能需要一些时间，具体取决于各种因素。

删除任务

要删除任务及其任务执行，使用 DeleteJob 命令。预设情况下，您只能删除处于最终状态（SUCCEEDED 或 CANCELED）的任务。否则，会出现异常。如果 IN_PROGRESS 参数设置为 force，可以删除处于 true 状态的任务。

要删除任务，运行以下命令：

```
aws iot delete-job --job-id 010 --force|--no-force
```

该命令不显示任何输出。

> Warning
>
> 删除处于 IN_PROGRESS 状态的任务后，部署该任务的设备将无法访问任务信息或更新任务执行状态。请谨慎使用，并且确保部署已删除的任务的每个设备能够恢复到有效状态。

删除任务可能需要一些时间，具体取决于为该任务创建的任务执行的数量以及其它因素。在删除任务过程中，任务的状态将显示为 DELETION_IN_PROGRESS。如果您尝试删除或取消其状态已为 DELETION_IN_PROGRESS 的任务，将导致错误。

只能有 10 个任务的状态可以同时为 DELETION_IN_PROGRESS。否则，会出现 LimitExceededException。

获取任务文档

要检索任务的任务文档，使用 GetJobDocument 命令。任务文档是对设备将执行的远程操作的描述。

要获取任务文档，运行以下命令：

```
aws iot get-job-document --job-id 010
```

该命令返回指定任务的任务文档：

```
{
    "document": "{\n\t\"operation\":\"install\",\n\t\"url\":\"http://amazon.com/
firmWareUpate-01\",\n\t\"data\":\"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/job-
test-bucket/datafile}\"\n}"
}
```

> **Note**
>
> 在使用此命令检索任务文档时，占位符 URL 不会替换为预签名 Amazon S3 URL。在设备调用 GetPendingJobExecutionsAPI 时：

列出任务

要获取 Amazon Web Services 账户 中的所有任务的列表，使用 ListJobs 命令。任务数据和任务执行数据将在 有限的时间内保留。运行以下命令来列出您的Amazon Web Services 账户中的所有任务：

```
aws iot list-jobs
```

该命令将返回您账户中按任务状态排序的所有任务：

```
{
    "jobs": [
        {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486687079.743,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/013",
            "createdAt": 1486687079.743,
            "targetSelection": "SNAPSHOT",
            "jobId": "013"
        },
        {
            "status": "SUCCEEDED",
            "lastUpdatedAt": 1486685868.444,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/012",
            "createdAt": 1486685868.444,
            "completedAt": 148668789.690,
            "targetSelection": "SNAPSHOT",
            "jobId": "012"
        },
        {
```

```
            "status": "CANCELED",
            "lastUpdatedAt": 1486678850.575,
            "jobArn": "arn:aws:iot:us-east-1:123456789012:job/011",
            "createdAt": 1486678850.575,
            "targetSelection": "SNAPSHOT",
            "jobId": "011"
        }
    ]
}
```

描述任务

要获取任务的状态，运行 DescribeJob 命令。以下命令说明如何描述任务：

```
$ aws iot describe-job --job-id 010
```

该命令返回指定任务的状态。例如：

```
{
    "documentSource": "https://s3.amazonaws.com/job-test-bucket/job-document.json",
    "job": {
        "status": "IN_PROGRESS",
        "jobArn": "arn:aws:iot:us-east-1:123456789012:job/010",
        "targets": [
            "arn:aws:iot:us-east-1:123456789012:thing/myThing"
        ],
        "jobProcessDetails": {
            "numberOfCanceledThings": 0,
            "numberOfFailedThings": 0,
            "numberOfInProgressThings": 0,
            "numberOfQueuedThings": 0,
            "numberOfRejectedThings": 0,
            "numberOfRemovedThings": 0,
            "numberOfSucceededThings": 0,
            "numberOfTimedOutThings": 0,
            "processingTargets": [
                arn:aws:iot:us-east-1:123456789012:thing/thingOne,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupOne,
                arn:aws:iot:us-east-1:123456789012:thing/thingTwo,
                arn:aws:iot:us-east-1:123456789012:thinggroup/thinggroupTwo
            ]
        },
        "presignedUrlConfig": {
            "expiresInSec": 60,
            "roleArn": "arn:aws:iam::123456789012:role/S3DownloadRole"
        },
        "jobId": "010",
        "lastUpdatedAt": 1486593195.006,
        "createdAt": 1486593195.006,
        "targetSelection": "SNAPSHOT",
        "jobExecutionsRolloutConfig": {
            "exponentialRate": {
                "baseRatePerMinute": integer,
                "incrementFactor": integer,
                "rateIncreaseCriteria": {
                    "numberOfNotifiedThings": integer, // Set one or the other
                    "numberOfSucceededThings": integer // of these two values.
                },
            "maximumPerMinute": integer
         }
        },
        "abortConfig": {
```

```
            "criteriaList": [
                {
                    "action": "string",
                    "failureType": "string",
                    "minNumberOfExecutedThings": integer,
                    "thresholdPercentage": integer
                }
            ]
        },
        "timeoutConfig": {
           "inProgressTimeoutInMinutes": number
         }
    }
}
```

列出任务的执行

在特定设备上运行的任务由任务执行对象表示。运行 ListJobExecutionsForJob 命令来列出任务的所有任务执行。以下命令说明如何列出任务的执行：

```
aws iot list-job-executions-for-job --job-id 010
```

该命令将返回任务执行的列表：

```
{
    "executionSummaries": [
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 1234567890
        }
    },
    {
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingTwo",
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "lastUpdatedAt": 1486593345.659,
            "queuedAt": 1486593196.378,
            "startedAt": 1486593345.659,
            "executionNumber": 4567890123
        }
    }
    ]
}
```

列出事物的任务执行

运行 ListJobExecutionsForThing 命令来列出在事物上运行的所有任务执行。以下命令说明如何列出事物的任务执行：

```
aws iot list-job-executions-for-thing --thing-name thingOne
```

该命令将返回正在指定事物上运行的或在指定事物上运行过的任务执行的列表：

```
{
```

```
    "executionSummaries": [
    {
        "jobExecutionSummary": {
            "status": "QUEUED",
            "lastUpdatedAt": 1486687082.071,
            "queuedAt": 1486687082.071,
            "executionNumber": 9876543210
        },
        "jobId": "013"
    },
    {
        "jobExecutionSummary": {
            "status": "IN_PROGRESS",
            "startAt": 1486685870.729,
            "lastUpdatedAt": 1486685870.729,
            "queuedAt": 1486685870.729,
            "executionNumber": 1357924680
        },
        "jobId": "012"
    },
    {
        "jobExecutionSummary": {
            "status": "SUCCEEDED",
            "startAt": 1486678853.415,
            "lastUpdatedAt": 1486678853.415,
            "queuedAt": 1486678853.415,
            "executionNumber": 4357680912
        },
        "jobId": "011"
    },
    {
        "jobExecutionSummary": {
            "status": "CANCELED",
            "startAt": 1486593196.378,
            "lastUpdatedAt": 1486593196.378,
            "queuedAt": 1486593196.378,
            "executionNumber": 2143174250
        },
        "jobId": "010"
    }
    ]
}
```


描述任务执行

运行 DescribeJobExecution 命令来获取任务执行的状态。您必须指定任务 ID 和事物名称（也可以选择指定执行编号）来标识任务执行。以下命令说明如何描述任务执行：

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

该命令将返回 JobExecution。例如：


```
{
    "execution": {
        "jobId": "017",
        "executionNumber": 4516820379,
        "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/thingOne",
        "versionNumber": 123,
        "createdAt": 1489084805.285,
        "lastUpdatedAt": 1489086279.937,
        "startedAt": 1489086279.937,
        "status": "IN_PROGRESS",
```

```
        "approximateSecondsBeforeTimedOut": 100,
        "statusDetails": {
            "status": "IN_PROGRESS",
            "detailsMap": {
                "percentComplete": "10"
            }
        }
    }
}
```

删除任务执行

运行 DeleteJobExecution 命令来删除任务执行。您必须指定任务 ID、事物名称和执行编号来标识任务执行。以下命令说明如何删除任务执行：

```
aws iot delete-job-execution --job-id 017 --thing-name thingOne --execution-number
 1234567890 --force|--no-force
```

该命令不显示任何输出。

预设情况下，任务执行的状态必须为 QUEUED 或处于最终状态（SUCCEEDED、FAILED、REJECTED、TIMED_OUT、REMOVED 或 CANCELED）。否则将出错。要删除状态为 IN_PROGRESS 的任务执行，您可以将 `force` 参数设置为 `true`。

> **Warning**
>
> 删除状态为 IN_PROGRESS 的任务执行后，执行该任务的设备将无法访问任务信息或更新任务执行状态。请谨慎使用，并且 确保 设备能够恢复到有效状态。

任务模板

使用任务模板预置可以部署到多组目标设备的任务。对于要部署到设备的频繁执行的远程操作（例如重启或安装应用程序），可以使用模板定义标准配置。想执行诸如部署安全补丁和错误修复等操作时，还可以从现有任务中创建模板。

创建任务模板时，您可以指定以下配置和资源。

- 任务属性
- 任务文件和目标
- 推出、计划和取消标准
- 超时和重试标准

自定义和 Amazon 托管模板

根据要执行的远程操作，您可以创建自定义任务模板或使用 Amazon 托管模板。使用自定义任务模板提供自己的自定义任务文档，并创建可重复使用的任务以部署到设备。Amazon 托管模板是由 Amazon IoT Jobs提供的模板常用操作的任务。这些模板具有用于某些远程操作的预定义任务文档，因此您无需创建自己的任务文档。托管模板可帮助您创建可重复使用的任务，来更快地执行设备。

主题

- 使用 Amazon 托管式模板部署一般远程操作 (p. 655)
- 创建自定义任务模板 (p. 667)

使用 Amazon 托管式模板部署一般远程操作

Amazon 托管模板是由 Amazon 提供的任务模板用于频繁执行的远程操作，例如重启、下载文件或在设备上安装应用程序。这些模板具有用于每个远程操作的预定义任务文档，因此您无需创建自己的任务文档。

您可以从一组预定义的配置中进行选择，并使用这些模板创建任务，而无需编写任何其他代码 使用托管模板，您可以查看部署到机群的任务文档。您可以使用这些模板创建任务，然后进一步创建可以重复用于远程操作的自定义任务模板。

托管模板包含哪些内容？

每个 Amazon 托管模板包含：

- 指定操作名称及其参数的任务文档。例如，如果您使用下载文件模板，操作名称是下载文件，参数可以是：
  - 要下载到设备的文件的 URL，可以是互联网资源，也可以是公共或预签名 S3 URL。
  - 设备上用于存储下载文件的本地文件路径。

  有任务文档及其参数的更多信息，请参阅 托管模板远程操作和任务文档 (p. 656)。
- 在任务文档中运行命令的环境。

  Note

  作业文档中的替换模式总数应小于或等于 10。

先决条件

要使设备运行托管模板任务文档指定的远程操作，您必须：

- 在设备上安装特定软件

  必须使用自己的设备软件和设备软件或Amazon IoT设备上的 Device Client。根据您的业务案例，您也可以选择同时运行它们，以使它们执行不同的功能。
  - 使用自己的设备软件和任务处理程序

    您可以使用支持远程操作的处理程序库为设备编写自己的代码。Amazon IoT Device SDK要部署和运行任务，请确保设备代理库已正确安装并在设备上运行。

    您也可以选择使用自己的处理程序，支持远程操作。有关更多信息，请参阅Amazon IoT设备客户端GitHub存储库中的示例作业处理程序。
  - 使用 Amazon IoT Device Client

    您安装并运行Amazon IoT设备上的 Device Client 是因为默认情况下它支持直接从控制台使用所有托管模板。

    Device Client 是用 C++ 编写的开源软件，您可以在基于 Linux 的嵌入式 IoT 设备上编译和安装。Device Client 有基本客户端和离散的客户端功能。基本客户端通过 MQTT 协议建立连接 Amazon IoT，可以连接不同的客户端功能。

    要在设备上执行远程操作，请使用 Device Client 的客户端任务功能。此功能包含一个用于接收任务文档的解析器以及实现任务文档中指定的远程操作的任务处理程序。有关Device Client 及其功能的更多信息，请参阅 Amazon IoT Device Client。

    在设备上运行时，Device Client 会收到任务文档，并具有用于在文档中运行命令的特定于平台的实现。有关设置 Device Client 和使用任务功能的详细信息，请参阅Amazon IoT教程。
- 使用支持的环境

对于每个托管模板，您将找到有关可用于运行远程操作的环境信息。我们建议您将模板与模板中指定的受支持的 Linux 环境结合使用。您可以使用 Amazon IoT Device Client. 运行管理模板远程操作，因为它支持常见的微处理器和 Linux 环境，例如 Debian 和 Ubuntu。

托管模板远程操作和任务文档

以下部分列出了 Amazon IoT Jobs 的不同 Amazon 托管式模板，并描述了可在设备上执行的远程操作。在下面的部分中，您将找到有关任务文档的信息以及每个远程操作的任务文档参数的描述。设备端软件使用模板名称和参数来执行远程操作。

Amazon 托管式模板接受输入参数，您可以在使用模板创建任务时为这些参数指定值。所有托管式模板都有以下两个共同的可选输入参数：`runAsUser` 和 `pathToHandler`。除了 `AWS-Reboot` 模板时，模板需要额外的输入参数，在使用模板创建任务时必须为这些参数指定值。这些所需的输入参数因您所选的模板而异。例如，如果您选择 `AWS-Download-File` 模板，则必须指定要安装的软件包列表以及从中下载文件的 URL。

使用 Amazon IoT 控制台或 Amazon CLI 创建使用托管式模板的任务时，可以为输入参数指定一个值。使用 CLI 时，您可以使用 `document-parameters` 对象提供这些值。有关更多信息，请参阅文档参数。

> **Note**
>
> 仅当从 Amazon 托管式模板创建任务时，才能使用 `document-parameters`。此参数不能与自定义任务模板一起使用，也不能从中创建作业。

下面显示了常见可选输入参数的说明。您将在下面的部分中看到每个托管式模板所需的其他输入参数的说明。

`runAsUser`

此参数指定是否以其他用户身份运行任务处理程序。如果在任务创建过程中未指定，任务处理程序将以与 Device Client 相同的用户身份运行。当您以其他用户身份运行任务处理程序时，请指定不超过 256 个字符的字符串值。

`pathToHandler`

设备上运行的任务处理程序的路径。如果在创建任务时未指定，Device Client 将使用当前工作目录。

下面显示了不同的远程操作、任务文档以及接受的参数。所有这些模板都支持在设备上运行远程操作的 Linux 环境。

Amazon - 下载- 文件

模板名称

`Amazon-Download-File`

模板描述

Amazon 提供的托管模板用于下载文件。

输入参数

此模板具有以下必需参数。您还可以指定可选参数 `runAsUser` 和 `pathToHandler`

`downloadUrl`

要从中下载文件的 URL，可以是互联网资源、可公开访问的 Amazon S3 中的对象，也可以是 Amazon S3 中的只能由设备使用预签名 URL 访问的对象。有关使用预签名 URL 和授予权限的详细信息，请参阅预签名 URL (p. 644)。

`filePath`

本地文件路径，显示设备中存储下载文件的位置。

设备的行为

设备从指定位置下载文件，验证下载完成并将其存储在本地。

任务文档

下面显示了任务文档及其最新版本。模板显示了任务处理程序和 shell 脚本的路径，`download-file.sh`，必须运行任务处理程序才能下载文件。它还显示了必需参数 `downloadUrl` 和 `filePath`。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Download-File",
        "type": "runHandler",
        "input": {
          "handler": "download-file.sh",
          "args": [
            "${aws:iot:parameter:downloadUrl}",
            "${aws:iot:parameter:filePath}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Amazon–安装 - 应用程序。

模板名称

`Amazon-Install-Application`

模板描述

Amazon 提供的托管模板用于安装一个或多个应用程序。

输入参数

此模板具有以下必需参数，`packages`。您还可以指定可选参数 `runAsUser` 和 `pathToHandler`

`packages`

要安装的一个或多个应用程序列表，以空格分隔。

设备的行为

设备将按照任务文档中的指定安装应用程序。

任务文档

下面显示了任务文档及其最新版本。模板显示了任务处理程序和 shell 脚本的路径，`install-packages.sh`，必须运行任务处理程序才能下载文件。它还显示必需参数 `packages`。

```
{
  "version": "1.0",
```

```
  "steps": [
    {
      "action": {
        "name": "Install-Application",
        "type": "runHandler",
        "input": {
          "handler": "install-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Amazon - 重启

模板名称

`Amazon-Reboot`

模板描述

Amazon 提供的托管模板用于重新启动设备。

输入参数

此模板没有必需参数。但是，您可以指定可选参数 `runAsUser` 和 `pathToHandler`。

设备的行为

设备已成功重新启动。

任务文档

下面显示了任务文档及其最新版本。模板显示了任务处理程序和 shell 脚本的路径，`reboot.sh`，必须运行任务处理程序才能重新启动设备。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Amazon–删除–应用程序

模板名称

`Amazon-Remove-Application`

模板描述

Amazon 提供的托管模板用于卸载一个或多个应用程序。

输入参数

此模板具有以下必需参数，packages。您还可以指定可选参数 runAsUser 和 pathToHandler

packages

要卸载的一个或多个应用程序列表，以空格分隔。

设备的行为

设备将按照任务文档中的指定卸载应用程序。

任务文档

下面显示了任务文档及其最新版本。模板显示了任务处理程序和 shell 脚本的路径，remove-packages.sh，必须运行任务处理程序才能下载文件。它还显示必需参数 packages。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Remove-Application",
        "type": "runHandler",
        "input": {
          "handler": "remove-packages.sh",
          "args": [
            "${aws:iot:parameter:packages}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Amazon— 重新启动 — 应用

模板名称

Amazon-Restart-Application

模板描述

Amazon 提供的托管模板用于停止和重新启动一个或多个服务。

输入参数

此模板具有以下必需参数，services。您还可以指定可选参数 runAsUser 和 pathToHandler

services

将重新启动的一个或多个应用程序列表，以空格分隔。

设备的行为

指定的应用程序停止然后在设备上重新启动。

任务文档

下面显示了任务文档及其最新版本。模板显示了任务处理程序和 shell 脚本的路径，`restart-services.sh`，必须运行任务处理程序才能重新启动系统服务。它还显示必需参数 `services`。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Restart-Application",
        "type": "runHandler",
        "input": {
          "handler": "restart-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Amazon–开始–复制

模板名称

`Amazon-Start-Application`

模板描述

Amazon提供的托管模板启动一个或多个服务。

输入参数

此模板具有以下必需参数，`services`。您还可以指定可选参数 `runAsUser` 和 `pathToHandler`

`services`

将启动的一个或多个应用程序列表，以空格分隔。

设备的行为

指定的应用程序开始在设备上运行。

任务文档

下面显示了任务文档及其最新版本。模板显示了任务处理程序和 shell 脚本的路径，`start-services.sh`，必须运行任务处理程序才能启动系统服务。它还显示必需参数 `services`。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Start-Application",
        "type": "runHandler",
        "input": {
          "handler": "start-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
```

```
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Amazon–停止–应用程序

模板名称

`Amazon-Stop-Application`

模板描述

Amazon 提供的托管模板停止一个或多个服务。

输入参数

此模板具有以下必需参数，`services`。您还可以指定可选参数 `runAsUser` 和 `pathToHandler`

`services`

将停止的一个或多个应用程序列表，以空格分隔。

设备的行为

指定的应用程序停止在设备上运行。

任务文档

下面显示了任务文档及其最新版本。模板显示了任务处理程序和 shell 脚本的路径，`stop-services.sh`，必须运行任务处理程序才能停止系统服务。它还显示必需参数 `services`。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Stop-Application",
        "type": "runHandler",
        "input": {
          "handler": "stop-services.sh",
          "args": [
            "${aws:iot:parameter:services}"
          ],
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

Amazon–Run–Command

模板名称

`Amazon-Run-Command`

模板描述

Amazon 提供的用于运行 shell 命令的托管模板。

输入参数

此模板具有以下必需参数，command。您还可以指定可选参数 runAsUser。

command

逗号分隔的命令字符串，命令本身中包含的任何逗号都需要转义。

设备的行为

设备按照任务文档中的指定运行 shell 命令。

任务文档

下面显示了任务文档及其最新版本。此模板显示任务命令的路径，以及您提供的设备将运行的命令。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Run-Command",
        "type": "runCommand",
        "input": {
          "command": "${aws:iot:parameter:command}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

主题

- 通过使用 Amazon Web Services Management Console 从 Amazon 托管式模板中创建任务 (p. 662)
- 通过使用 Amazon CLI 从 Amazon 托管式模板中创建任务 (p. 664)

通过使用 Amazon Web Services Management Console 从 Amazon 托管式模板中创建任务

使用 Amazon Web Services Management Console 获取 Amazon 托管模板有关的信息并使用这些模板创建任务。然后，您可以将创建的任务保存为自己的自定义模板。

获取有关托管模板的详细信息

您可以从 Amazon IoT 控制台中获取有关可供使用的不同托管模板信息。

1. 要查看可用的托管模板，请转到 Amazon IoT控制台的任务模板中心然后选择托管模板选项卡。
2. 选择托管模板查看其详细信息。

详细信息页面包括以下信息：

- 托管模板的名称、描述和 Amazon Resource Name (ARN)。
- 可以执行远程操作的环境，例如 Linux。
- JSON 任务文档，用于指定任务处理程序的路径以及要在设备上运行的命令。例如，下面显示了一个示例任务文档 AWS-Reboot 模板。模板显示了任务处理程序和 shell 脚本的路径，reboot.sh，必须运行任务处理程序才能重新启动设备。

```
{
```

```
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

有关任务文档及其各种远程操作参数的详细信息，请参阅 托管模板远程操作和任务文档 (p. 656)。

- 任务文档的最新版本。

使用托管模板创建任务

您可以使用 Amazon 管理控制台选择用于创建任务的 Amazon 托管模板。本节向您演示了应如何进行操作。

您也可以启动就业创造工作流程，然后选择 Amazon 创建任务时要使用的托管模板。有关工作流的更多信息，请参阅 使用 Amazon Web Services Management Console 创建和管理任务。 (p. 645)。

1. 选择您的 Amazon 托管模板

   转至Amazon IoT控制台的任务模板中心，选择托管模板选项卡，然后选择模板。
2. 使用托管模板创建任务

   1. 在模板详细信息页面上，选择 Create job(创建任务)。

      控制台切换到添加模板配置的工作流的创建任务的自定义任务属性的步骤。
   2. 输入唯一的字母数字任务名称以及可选的描述和标签，然后选择下一步。
   3. 选择要在此任务中运行的事物或事物组作为任务目标。
   4. 在 Job document（任务文档）部分中，您的模板随其配置设置和输入参数一起显示。输入所选模板的输入参数的值。例如，如果您选择了 Amazon-Download-File 模板：
      - 对于 downloadUrl，输入要下载的文件的 URL，例如：`https://example.com/index.html`。
      - 对于 filePath，输入设备上存储下载文件的路径，例如：*path/to/file*。

      还可以选择输入 `runAsUser` 和 `pathToHandler` 参数的值。有关每个模板的输入参数的更多信息，请参阅托管模板远程操作和任务文档 (p. 656)。
   5. 在 Job configuration（作业配置）页面上，将作业类型选择为连续或快照作业。快照任务在目标设备和组上完成运行后即算完成任务。连续任务适用于事物组，并会添加到指定目标组的任何设备上运行。
   6. 继续为您的任务添加任何其他配置，然后查看并创建任务。有关其他配置的更多信息，请参阅：
      - 任务推出、计划和中止配置 (p. 672)
      - 任务执行超时和重试配置 (p. 677)

从托管模板创建自定义任务模板

您可以使用 Amazon 托管模板和自定义任务作为创建自己的自定义任务模板的起点。要创建自定义任务模板，请先从 Amazon 托管模板，如上一部分中所述。

然后，您可以将自定义任务另存为模板，创建自己的自定义任务模板。要另存为模板：

1. 转至Amazon IoT控制台的任务中心然后选择包含托管模板的任务。
2. 选择另存为任务模板，然后创建自定义任务模板。有关创建自定义模板的更多信息，请参阅 从现有任务创建任务模板 (p. 668)。

通过使用 Amazon CLI 从 Amazon 托管式模板中创建任务

使用 Amazon CLI 获取 Amazon 托管模板有关的信息并使用这些模板创建任务。然后，您可以将任务另存为模板，并创建自己的自定义模板。

列出托管模板

`list-managed-job-templates` Amazon CLI 命令将列出您 Amazon Web Services 账户中的所有任务模板。

```
aws iot list-managed-job-templates
```

原定设置情况下，运行此命令将显示所有可用 Amazon 托管式模板及其详细信息。

```
{
    "managedJobTemplates": [
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Reboot:1.0",
            "templateName": "AWS-Reboot",
            "description": "A managed job template for rebooting the device.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Remove-Application:1.0",
            "templateName": "AWS-Remove-Application",
            "description": "A managed job template for uninstalling one or more
 applications.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },
        {
            "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Stop-Application:1.0",
            "templateName": "AWS-Stop-Application",
            "description": "A managed job template for stopping one or more system
 services.",
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        },

        ...

        {
            "templateArn": "arn:aws:iot:us-east-1::jobtemplate/AWS-Restart-
Application:1.0",
            "templateName": "AWS-Restart-Application",
            "description": "A managed job template for restarting one or more system
 services.",
```

```
            "environments": [
                "LINUX"
            ],
            "templateVersion": "1.0"
        }
    ]
}
```

有关更多信息，请参阅ListManagedJobTemplates。

获取有关托管式模板的详细信息

以下 describe-managed-job-template Amazon CLI 命令获取有关指定任务模板的详细信息。指定任务模板名称和可选的模板版本。如果未指定模板版本，则返回预定义的原定设置版本。以下示例显示运行命令以获取有关 Amazon-Download-File 模板的详细信息。

```
aws iot describe-managed-job-template \
    --template-name Amazon-Download-File
```

该命令显示模板详细信息和 ARN、任务文档以及 documentParameters 参数，此参数是模板输入参数的键值对的列表。有关不同模板和输入参数的信息，请参阅托管模板远程操作和任务文档 (p. 656)。

> **Note**
>
> 使用此 API 时返回的 documentParameters 对象只能在从 Amazon 托管式模板中创建任务时使用。此对象不得用于自定义任务模板。有关演示如何使用此参数的示例，请参阅使用托管模板创建任务 (p. 666)。

```
{
    "templateName": "Amazon-Download-File",
    "templateArn": "arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0",
    "description": "A managed job template for downloading a file.",
    "templateVersion": "1.0",
    "environments": [
        "LINUX"
    ],
    "documentParameters": [
        {
            "key": "downloadUrl",
            "description": "URL of file to download.",
            "regex": "(.*?)",
            "example": "http://www.example.com/index.html",
            "optional": false
        },
        {
            "key": "filePath",
            "description": "Path on the device where downloaded file is written.",
            "regex": "(.*?)",
            "example": "/path/to/file",
            "optional": false
        },
        {
            "key": "runAsUser",
            "description": "Execute handler as another user. If not specified, then handler
 is executed as the same user as device client.",
            "regex": "(.){0,256}",
            "example": "user1",
            "optional": true
        },
        {
            "key": "pathToHandler",
```

```
            "description": "Path to handler on the device. If not specified, then device
 client will use the current working directory.",
            "regex": "(.){0,4096}",
            "example": "/path/to/handler/script",
            "optional": true
        }
    ],
    "document": "{\"version\":\"1.0\",\"steps\":[{\"action\":{\"name
\":\"Download-File\",\"type\":\"runHandler\",\"input\":{\"handler\":
\"download-file.sh\",\"args\":[\"${aws:iot:parameter:downloadUrl}\",
\"${aws:iot:parameter:filePath}\"],\"path\":\"${aws:iot:parameter:pathToHandler}\"},
\"runAsUser\":\"${aws:iot:parameter:runAsUser}\"}}]}"
}
```

有关更多信息，请参阅DescribeManagedJobTemplate。

使用托管模板创建任务

create-job Amazon CLI 命令可用于从任务模板创建任务。它的目标是名为 thingOne 的设备，并将指定托管模板的 Amazon Resource Name (ARN)，用作该任务的基础。您可以覆盖高级配置（如超时和取消配置），方法是传递 create-job 命令的关联参数。

该示例显示如何创建使用 AWS-Download-File 模板的任务。它还展示了如何使用 document-parameters 参数指定模板的输入参数。

> **Note**
>
> 只能将 document-parameters 对象与 Amazon 托管式模板一起使用。此对象不得与自定义任务模板一起使用。

```
aws iot create-job \
   --targets arn:aws:iot:region:account-id:thing/thingOne \
   --job-id "new-managed-template-job" \
   --job-template-arn arn:aws:iot:region::jobtemplate/AWS-Download-File:1.0 \
   --document-parameters downloadUrl=https://example.com/index.html,filePath=path/to/file
```

其中：

- *region* 是 Amazon Web Services 区域。
- *account-id* 唯一的 Amazon Web Services 账户 数字。
- *thingOne* 是任务指向的 IoT 事物的名称。
- *Amazon-Download-File:1.0* 是托管式模板的名称。
- https://*example.com/index*.html 是从中下载文件的 URL。
- https://*pathto/file/index* 是设备上用于存储下载文件的路径。

运行此命令会为模板 *Amazon-Download-File* 创建任务，如下所示。

```
{
    "jobArn": "arn:aws:iot:region:account-id:job/new-managed-template-job",
    "jobId": "new-managed-template-job",
    "description": "A managed job template for downloading a file."
}
```

从托管模板创建自定义任务模板

1. 使用上一节中介绍的托管模板创建任务。

2. 使用您创建的任务的 ARN 创建自定义任务模板。有关更多信息，请参阅从现有任务创建任务模板 (p. 670)：

创建自定义任务模板

您可以使用 Amazon CLI 和 Amazon IoT 控制台创建任务模板。您还可以使用 Amazon CLI、Amazon IoT 控制台和 Fleet Hub for Amazon IoT Device Management web 应用程序从任务模板创建任务。有关在 Fleet Hub 应用程序中使用任务模板的更多信息，请参阅在 Fleet Hub for Amazon IoT Device Management 中使用任务模板。

> **Note**
>
> 作业文档中的替换模式总数应小于或等于 10。

主题

- 通过使用 Amazon Web Services Management Console 创建自定义任务模板 (p. 667)
- 通过使用 Amazon CLI 创建自定义任务模板 (p. 669)

通过使用 Amazon Web Services Management Console 创建自定义任务模板

本主题介绍如何使用 Amazon IoT 控制台创建、删除任务模板并查看有关详细信息。

创建自定义任务模板

您可以创建一个原始 自定义任务模板或从现有任务创建任务模板。您还可以从使用 Amazon 托管模板创建的现有作业创建自定义作业模板。有关更多信息，请参阅从托管模板创建自定义任务模板 (p. 663)：

创建原始任务模板

1. 开始创建您的任务模板

   1. 转至Amazon IoT控制台任务模板中心然后选择自定义模板选项卡。
   2. 选择 Create job template（创建任务模板）。

   > **Note**
   >
   > 您还可以从 Fleet Hub 下的 Related services（相关服务）页面导航到 Job templates（任务模板）页面。

2. 指定任务模板属性

   在创建任务模板页面中，输入任务名称的字母数字标识符和字母数字描述，提供有关该模板的其他详细信息。

   > **Note**
   >
   > 我们建议不要在您的任务 ID 或描述中使用个人身份信息。

3. 提供任务文档

   提供 JSON 任务文件，该文件存储在 S3 存储桶中或作为任务中指定的内联任务文档。当您使用此模板创建任务时，此任务文件将成为任务文档。

   如果任务文件存储在 S3 存储桶中，请输入 S3 URL 或选择浏览 S3，然后导航到您的工作文档并选择它。

Note

您可以选择当前区域中的 S3 存储桶。

4. 继续为您的任务添加任何其他配置，然后查看并创建任务。有关其他可选配置的信息，请参阅以下链接：

   - 任务推出、计划和中止配置 (p. 672)
   - 任务执行超时和重试配置 (p. 677)

从现有任务创建任务模板

1. 选择您的任务

   1. 转到Amazon IoT 控制台的任务中心，选择要用作任务模板基础的任务。
   2. 选择另存为任务模板。

   Note

   您可以选择不同的任务文档或编辑原始任务中的高级配置，然后选择 Create job template（创建任务模板）。您的新任务模板将显示在 Job template（任务模板）页面。

2. 指定任务模板属性

   在创建任务模板页面中，输入任务名称的字母数字标识符和字母数字描述，提供有关该模板的其他详细信息。

   Note

   任务文档是您在创建模板时指定的任务文件。如果是在任务中指定任务文档的，而不是 S3 位置，可以在此任务的详细信息页面中看到任务文档。

3. 继续为您的任务添加任何其他配置，然后查看并创建任务。有关其他配置的更多信息，请参阅：

   - 任务推出、计划和中止配置 (p. 672)
   - 任务执行超时和重试配置 (p. 677)

从自定义任务模板创建任务。

您可以通过转到任务模板的详细信息页面，从自定义任务模板创建任务，如本主题中所述。您还可以创建任务，或者通过选择运行任务创建工作流程时要使用的任务模板来创建任务。有关更多信息，请参阅使用 Amazon Web Services Management Console 创建和管理任务。 (p. 645)：

本主题介绍如何从自定义任务模板的详细信息页面创建任务。您还可以从 Amazon 托管模板创建任务。有关更多信息，请参阅使用托管模板创建任务 (p. 663)：

1. 选择自定义任务模板

   转至Amazon IoT控制台的任务模板中心，选择自定义模板选项卡，然后选择您的模板。

2. 使用自定义模板创建任务

   要创建任务：

   1. 在模板详细信息页面上，选择 Create job(创建任务)。

      控制台切换到添加模板配置的工作流的创建任务的自定义任务属性的步骤。
   2. 输入唯一的字母数字任务名称以及可选的描述和标签，然后选择下一步。
   3. 选择要在此任务中运行的事物或事物组作为任务目标。

在任务文档部分中，您的模板随其配置设置一起显示。如果要使用不同的任务文档，请选择 Browse（浏览），然后选择不同的存储桶和文档。选择下一步。

4. 在 Job configuration（作业配置）页面上，将作业类型选择为连续或快照作业。快照任务在目标设备和组上完成运行后即算完成任务。连续任务适用于事物组，并会添加到指定目标组的任何设备上运行。
5. 继续为您的任务添加任何其他配置，然后查看并创建任务。有关其他配置的更多信息，请参阅：
   - 任务推出、计划和中止配置 (p. 672)
   - 任务执行超时和重试配置 (p. 677)

Note

当根据作业模板创建的作业更新作业模板提供的现有参数时，这些更新的参数将覆盖作业模板为该作业提供的现有参数。

您还可以使用 Fleet Hub Web 应用程序从任务模板创建任务。有关在 Fleet Hub 中创建任务的信息，请参阅在 Fleet Hub for Amazon IoT Device Management 中使用任务模板。

删除任务模板

要删除任务模板，首先转至Amazon IoT控制台任务模板中心然后选择自定义模板选项卡。然后，选择要删除的任务模板，然后选择 Delete(删除)。

Note

删除是永久性的，任务模板不再显示在自定义模板选项卡。

通过使用 Amazon CLI 创建自定义任务模板

本主题介绍如何使用 Amazon CLI 创建、删除任务模板以及检索有关详细信息。

从头开始创建任务模板

以下Amazon CLI命令说明了如何使用存储在 S3 存储桶 (S3) 中的任务文档 (j *ob-document.js* on)) 和具有从 Amazon S3 下载文件的权限的角色 (S *3DownloadRole*) 创建任务。

```
aws iot create-job-template  \
      --job-template-id 010  \
      --document-source https://s3.amazonaws.com/my-s3-bucket/job-document.json  \
      --timeout-config inProgressTimeoutInMinutes=100 \
      --job-executions-rollout-config "{ \"exponentialRate\": { \"baseRatePerMinute\":
 50, \"incrementFactor\": 2, \"rateIncreaseCriteria\": { \"numberOfNotifiedThings\": 1000,
 \"numberOfSucceededThings\": 1000}}, \"maximumPerMinute\": 1000}" \
      --abort-config "{ \"criteriaList\": [ { \"action\": \"CANCEL\", \"failureType\":
 \"FAILED\", \"minNumberOfExecutedThings\": 100, \"thresholdPercentage\": 20}, { \"action
\": \"CANCEL\", \"failureType\": \"TIMED_OUT\", \"minNumberOfExecutedThings\": 200,
 \"thresholdPercentage\": 50}]}" \
      --presigned-url-config "{\"roleArn\":\"arn:aws:iam::123456789012:role/S3DownloadRole
\", \"expiresInSec\":3600}"
```

可选的 `timeout-config` 参数指定每个设备完成其任务执行所具有的时间。计时器在任务执行状态设置为 `IN_PROGRESS` 时启动。如果任务执行状态未在时间到期之前设置为其它最终状态，则会设置为 `TIMED_OUT`。

进行中计时器无法更新，将应用到该任务的全部任务执行。只要任务执行保持在 `IN_PROGRESS` 状态的时间长度超过了此间隔，任务执行将失败，并切换为最终 `TIMED_OUT` 状态。Amazon IoT 还将发布 MQTT 通知。

有关创建任务推出和中止相关配置的更多信息，请参阅任务推出和中止配置。

Note

在您创建任务时，系统会检索指定为 Amazon S3 文件的任务文档。如果在创建任务后更改任务文档源 Amazon S3文件的内容，发送到任务目标的内容不会更改。

从现有任务创建任务模板

以下 Amazon CLI 命令通过指定某个现有任务的 Amazon Resource Name (ARN) 来创建任务模板。新任务模板将使用任务中指定的所有配置。或者，您可以使用任何可选参数更改现有任务中的任何配置。

```
aws iot create-job-template  \
      --job-arn arn:aws:iot:region:123456789012:job/job-name  \
      --timeout-config inProgressTimeoutInMinutes=100
```

获取有关任务模板的详细信息

以下 Amazon CLI 命令将获取有关指定任务模板的详细信息。

```
aws iot describe-job-template \
      --job-template-id template-id
```

该命令将显示以下输出。

```
{
   "abortConfig": {
      "criteriaList": [
         {
            "action": "string",
            "failureType": "string",
            "minNumberOfExecutedThings": number,
            "thresholdPercentage": number
         }
      ]
   },
   "createdAt": number,
   "description": "string",
   "document": "string",
   "documentSource": "string",
   "jobExecutionsRolloutConfig": {
      "exponentialRate": {
         "baseRatePerMinute": number,
         "incrementFactor": number,
         "rateIncreaseCriteria": {
            "numberOfNotifiedThings": number,
            "numberOfSucceededThings": number
         }
      },
      "maximumPerMinute": number
   },
   "jobTemplateArn": "string",
   "jobTemplateId": "string",
   "presignedUrlConfig": {
      "expiresInSec": number,
      "roleArn": "string"
```

```
   },
   "timeoutConfig": {
      "inProgressTimeoutInMinutes": number
   }
}
```

列出任务模板

以下 Amazon CLI 命令将列出您Amazon Web Services 账户中的所有任务模板。

```
aws iot list-job-templates
```

该命令将显示以下输出。

```
{
   "jobTemplates": [
      {
         "createdAt": number,
         "description": "string",
         "jobTemplateArn": "string",
         "jobTemplateId": "string"
      }
   ],
   "nextToken": "string"
}
```

要检索其它结果页面，使用 nextToken 字段的值。

删除任务模板

以下 Amazon CLI 命令将删除指定的任务模板。

```
aws iot delete-job-template \
      --job-template-id template-id
```

该命令不显示任何输出。

从自定义任务模板创建任务。

以下 Amazon CLI 命令将从自定义任务模板创建任务。它的目标是名为 thingOne 的设备，并将指定任务模板的 Amazon Resource Name (ARN)，用作该任务的基础。您可以覆盖高级配置（如超时和取消配置），方法是传递 create-job 命令的关联参数。

> **Warning**
>
> 仅当从 Amazon 托管式模板创建任务时，document-parameters 对象才能与 create-job 命令一起使用。此对象不得与自定义任务模板一起使用。有关演示如何使用此参数创建任务的示例，请参阅使用托管模板创建任务 (p. 666)。

```
aws iot create-job \
      --targets arn:aws:iot:region:123456789012:thing/thingOne  \
```

```
    --job-template-arn arn:aws:iot:region:123456789012:jobtemplate/template-id
```

任务配置

对于部署到指定目标的每个任务，您可以拥有以下附加配置。

- Rollout（推出）：此配置定义了每分钟有多少设备接收任务文档。
- Scheduling（计划）：如果要为future 日期和时间安排任务，请使用此配置来安排任务。
- Abort（中止）：如果部分设备未收到任务通知或设备报告其任务执行失败，可使用此配置取消任务。
- Timeout（超时）：如果任务目标在其任务执行开始后的一段时间内没有响应，则任务可能会失败。
- Retry（重试）：如果设备在尝试完成任务执行时报告失败或任务执行超时，则可使用此配置为设备重试任务执行。

通过使用这些配置，您可以监控任务执行状态，避免向整个机群发送错误的更新。

主题

- 任务配置的工作原理 (p. 672)
- 指定其他配置 (p. 680)

任务配置的工作原理

您可以在部署任务时使用推出和中止配置，在执行任务时使用超时和重试配置。以下各部分显示了有关这些配置如何工作的更多信息。

主题

- 任务推出、计划和中止配置 (p. 672)
- 任务执行超时和重试配置 (p. 677)

任务推出、计划和中止配置

您可以使用任务推出、中止和计划配置来定义每分钟接收任务文档的设备数量，提前一年计划任务推出，以及确定特定数量的设备没有收到任务文档时取消任务的条件。

任务推出配置

您可以指定向目标发送待处理任务执行通知的速度。您可以创建分段推出来更好地管理更新、重启和其他操作。若要指定目标的通知方式，请使用任务推出速率。

任务推出速率

您可以使用恒定推出速率或指数推出速率来创建推出配置。您可以使用恒定推出速率来指定每分钟可通知的最大任务目标数。

在满足各种条件和阈值时，可使用指数推出速率部署 Amazon IoT 任务。如果失败任务数与指定的一组条件相匹配，则您可以取消任务推出。在创建任务时，使用 `JobExecutionsRolloutConfig` 对象设置任务推出速率条件。在创建任务时，使用 `AbortConfig` 对象设置任务中止条件。

下面的示例显示推出速率的工作原理。例如，基本速率为每分钟 50 次、增量因子为 2、通知和成功设备数量各为 1000 的任务推出，其工作方式如下：该任务将以每分钟执行 50 次任务的速率开始，并以该速率继续工作，直到 1000 个事物收到任务执行通知或执行了 1000 次成功的任务。

下表说明了推出将如何通过前四个增量继续。

每分钟的推出速率	50	100	200	400
满足速率提高的通知设备数或成功的任务执行	1000	2000	3000	4,000

Note

如果您的最大并发任务限制为 500 个 (`isConcurrent = True`)，则所有活动任务的状态将保持为 `IN-PROGRESS`，并且在并发作业数达到 499 个 (`isConcurrent = False`)) 之前，不会推出任何新任务执行。此规则适用于快照任务和连续任务。
如果 `isConcurrent = True`，表示该作业当前正在向目标组中的所有设备推出作业执行。如果 `isConcurrent = False`，表示该任务已完成向目标组中的所有设备推出全部任务执行，并且如果您选择了作业终止配置，则当目标组中的所有设备都达到终端状态或目标组的阈值百分比时，任务将更新其状态。`isConcurrent = True` 和 `isConcurrent = False` 的任务级别状态都为 `IN_PROGRESS`。
有关活动和并发任务限制的更多信息，请参阅活动和并发任务限制 (p. 730)。

使用动态事物组的连续任务的任务推出速率

当您使用连续任务在实例集中推出远程操作时，Amazon IoT Jobs 为目标事物组中的设备推出任务执行。对于添加到动态事务组中的任何新设备，这些任务执行会继续推出到新添加的设备，甚至在创建任务之后也是如此。

推出配置只能控制在创建任务之前添加到组中的设备的推出速率。创建任务后，对于任何新设备，一旦设备加入目标组，就会立即近乎实时地创建任务执行。

Job 调度配置

您可以使用预先确定的开始时间、结束时间和结束行为，最多提前一年安排连续任务或快照任务，以确定在到达结束时间后每个任务执行将发生的情况。此外，您可以为连续作业创建一个可选的定期维护窗口，该窗口具有灵活的频率、开始时间和持续时间，以便将任务文档发布到目标组内的所有设备。

任务计划配置

开始时间

计划任务的开始时间是该任务开始向目标组中的所有设备推出任务文档的未来日期和时间。计划任务的开始时间适用于连续任务和快照任务。计划任务在最初创建时，将保持 `SCHEDULED` 状态。达到您选择的 `startTime` 后，它将更新为 `IN_PROGRESS` 并开始推出任务文档。`startTime` 必须小于或等于自您创建计划任务的初始日期和时间起一年。

对于在定期维护窗口期间在遵守夏令时 (DST) 的位置进行可选调度配置的作业，从 DST 切换到标准时间，从标准时间切换到 DST 时，时间将更改一小时。

Note

Amazon 管理控制台中显示的时区是您当前的系统时区。但是，这些时区将在系统中转换为 UTC。

结束时间

计划任务的结束时间是该任务停止向目标组中的任何剩余设备推出任务文档的未来日期和时间。计划任务的结束时间适用于连续任务和快照任务。在计划任务达到所选的 `endTime`，并且所有任务执行都达到终止状态后，它会将其状态从 `IN_PROGRESS` 更新为 `COMPLETED`。`endTime` 必须小于或等于自您创建计划任务的初始日期和时间起两年。介于 `startTime` 和 `endTime` 之间的最短持续时间为三十分钟。将根据重试配置中所做的选择重试任务执行，直到任务达到 `endTime`，然后 `endBehavior` 将决定如何继续。

对于在定期维护窗口期间在遵守夏令时 (DST) 的位置进行可选调度配置的作业，从 DST 切换到标准时间，从标准时间切换到 DST 时，时间将更改一小时。

> **Note**
>
> Amazon 管理控制台中显示的时区是您当前的系统时区。但是，这些时区将在系统中转换为 UTC。

结束行为

计划任务的结束行为决定了当任务达到所选的 endTime 时，任务执行和所有未完成的任务执行会发生什么情况。

以下列出了创建任务或任务模板时可以选择的结束行为：

- STOP_ROLLOUT
  - STOP_ROLLOUT 将停止向任务目标组中的所有剩余设备推出任务文档。此外QUEUED，所有IN_PROGRESS任务执行都将继续，直到达到终止状态。除非选择 CANCEL 或 FORCE_CANCEL，否则这是默认结束行为。
- CANCEL
  - CANCEL 将停止向任务目标组中的所有剩余设备推出任务文档。此外，所有 QUEUED 任务执行都将被取消，而所有 IN_PROGRESS 任务执行都将继续，直到达到终止状态。
- FORCE_CANCEL
  - FORCE_CANCEL 将停止向任务目标组中的所有剩余设备推出任务文档。此外，所有 QUEUED 和 IN_PROGRESS 任务执行都将被取消。

> **Note**
>
> 如果不选择 endTime，那么将无法选择 endBehavior。

最长持续时间

无论 startTime 和 endTime 如何，计划任务的最长持续时间都必须小于或等于两年。

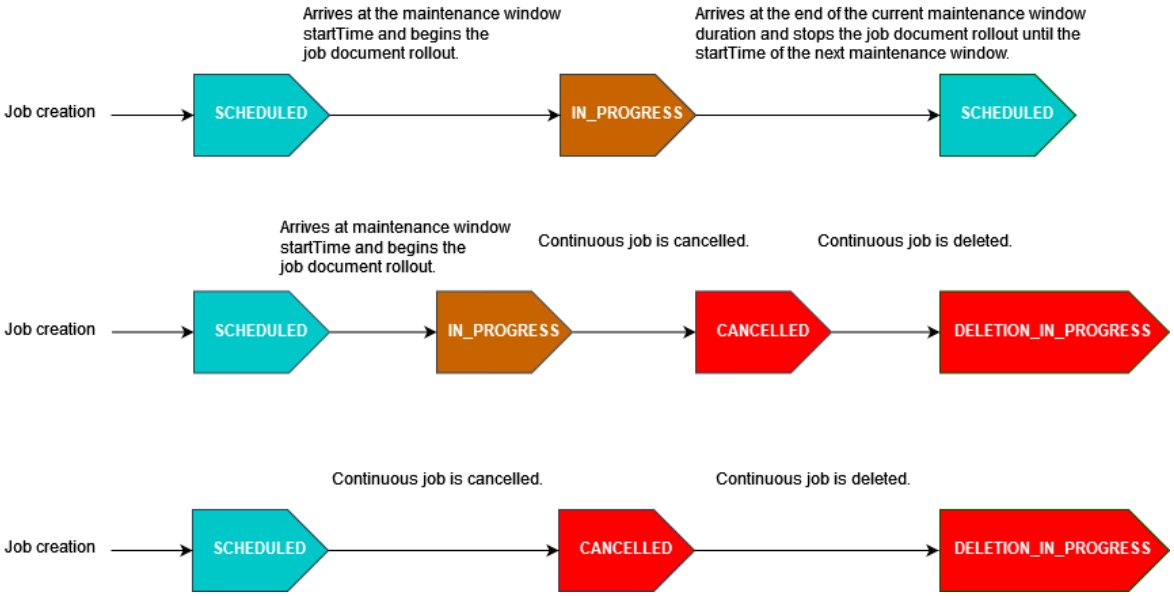
下表列出了计划任务的常见持续时间场景：

预定Job 示例编号	startTime	endTime	最长持续时间
1	在首次创造就业机会后立即生效。	首次创造就业机会一年后。	一年
2	首次创造就业机会一个月后。	首次创造就业机会后13个月。	一年
3	首次创造就业机会一年后。	首次创造就业机会两年后。	一年
4	在最初创造就业机会后立即生效。	首次创造就业机会两年后。	两年

周期性维护时段

维护窗口是Amazon管理控制台计划配置中以及CreateJob和CreateJobTemplate APISchedulingConfig 中的可选配置。您可以设置定期维护窗口，预先确定的开始时间、持续时间和维护窗口出现的频率（每天、每周或每月）。维护时段仅适用于连续作业。定期维护窗口的最大持续时间为 23 小时 50 分钟。

下图说明了具有可选维护时段的计划任务场景的任务状态：



有关作业状态的更多信息，请参阅任务和任务执行状态 (p. 640)。

> **Note**
>
> 如果任务在维护`endTime`时段内到达，则任务将从更新`IN_PROGRESS`为`COMPLETED`。此外，所有剩余的任务执行都将遵循该作业`endBehavior`的执行。

Cron 表达式

对于在维护时段内以自定义频率推出任务文档的计划作业，则使用 cron 表达式输入自定义频率。cron 表达式有六个必填字段，之间以空格分隔。

语法

```
cron(fields)
```

字段	值	通配符
分钟	0-59	, - * /
小时	0-23	, - * /
Day-of-month	1-31	, - * ? / L W
月份	1-12 或 JAN-DEC	, - * /
Day-of-week	1-7 或 SUN-SAT	, - * ? L #
年	1970-2199	, - * /

通配符

- , (逗号) 通配符包含其他值。在“月份”字段中，JAN、FEB 和 MAR 将包含 January、February 和 March。

- - (破折号) 通配符用于指定范围。在“日”字段中，1-15 将包含指定月份的 1 - 15 日。
- * (星号) 通配符包含该字段中的所有值。在“小时”字段中，* 将包含每个小时。不能在 Day-of-month 和 Day-of-week 字段中都使用 *。如果您在一个中使用它，则必须在另一个中使用 ？。
- / (正斜杠) 通配符用于指定增量。在“分钟”字段中，您可以输入 1/10 以指定从一个小时的第一分钟开始的每个第十分钟 (例如，第 11 分钟、第 21 分钟和第 31 分钟，依此类推)。
- ? (问号) 通配符用于指定一个或另一个。在 Day-of-month 字段中，你可以输入 7 如果你不在乎 7 号是一周中的哪一天，你可以输入？ 在 Day-of-week 字段中。
- Day-of-month -month (日期) ay-of-week 字段字段中的 L 通配符用于指定该月或该周最后一天。
- Day-of-month 字段中的**W**通配符指定工作日。在 Day-of-month 字段中，**3W**指定最接近该月第三天的工作日。
- Day-of-week 字段中的 # 通配符用于指定一个月内特定日期的特定实例。例如，3#2 指该月的第二个星期二：3 指的是星期二，因为它是每周的第三天，2 是指该月内该类型的第二天。

  Note

  如果使用 “#” 字符，则只能在 day-of-week 字段中定义一个表达式。例如，`"3#1,6#3"` 是无效的，因为它被解释为两个表达式。

限制

- 您无法在同一 cron 表达式中为 Day-of-month 和 Day-of-week 字段同时指定值。如果您在其中一个字段中指定了值 (或一个 *) ，则必须在另一个字段中使用 ? (问号) 。

示例

在使用 cron 表达式进行定期维护时段时，请参阅以下`startTime` cron 字符串示例。

分钟	小时	Day-month	月	星期几	年	意思
0	10	*	*	?	*	每天上午的 10:00 (UTC) 运行
15	12	*	*	?	*	每天在下午 12:15 (UTC) 运行
0	18	?	*	MON-FRI	*	每星期一到星期五的下午 6:00 (UTC) 运行
0	8	1	*	?	*	每月第 1 天的上午 8:00 (UTC) 运行

定期维护窗口持续时间结束逻辑

当维护时段期间的任务部署达到当前维护时段的持续时间结束时，将执行以下操作：

- 该Job 将停止向目标组中的所有剩余设备发布任务文档。它将在下一维护时`startTime`段恢复。
- 所有状态为的任务执行都`QUEUED`将保持状态，`QUEUED`直到下一个维护时段出现为止，在此时段内，`IN_PROGRESS`当设备准备好开始执行作业文档中指定的操作时，它们可以切换到该时段。`startTime`

- 所有状态为的任务执行IN_PROGRESS将继续执行作业文档中指定的操作，直到它们达到终止状态为止。中指定的任何重试都JobExecutionsRetryConfig将在下一个维护时startTime段内进行。

任务中止配置

使用此配置可创建条件，以便设备的阈值百分比符合该条件时取消任务。例如，在以下情况，您可以使用此配置取消任务：

- 设备的阈值百分比没有收到任务执行通知时，例如设备与 OTA 更新不兼容。在这种情况下，设备可以报告 REJECTED 状态。
- 设备的阈值百分比报告其任务执行失败时，例如设备尝试从 Amazon S3 URL 下载任务文档时发生连接中断。在这种情况下，设备必须进行编程，才能向 Amazon IoT 报告 FAILURE 状态。
- 任务执行开始后，任务执行超时的设备阈值百分比报告 TIMED_OUT 状态时。
- 出现多次重试失败时。添加重试配置时，每次重试都可能让您的 Amazon Web Services 账户 产生额外费用。在这种情况下，取消任务可以取消排队的任务执行，避免重试这些执行。有关重试配置以及其与中止配置搭配使用的更多信息，请参阅 任务执行超时和重试配置 (p. 677)。

您可以使用 Amazon IoT 控制台或 Amazon IoT Jobs API 设置任务中止条件。

任务执行超时和重试配置

当任务执行的时间超过设置的持续时间时，系统使用任务执行超时配置向您发送 任务通知 (p. 691)。如果任务失败或超时，系统会使用任务执行重试配置来重试执行。

任务执行超时配置

只要任务执行卡在 IN_PROGRESS 状态的时间超出预期，系统便会使用任务执行超时配置通知您。任务处于 IN_PROGRESS 状态时，您可以监控任务执行的进度。

任务超时计时器

有两种类型的计时器：进行中计时器和步骤计时器。

进行中计时器

创建任务或任务模板时，您可以为进行中计时器指定介于 1 分钟到 7 天之间的值。在任务执行开始之前，您可以更新此计时器的值。计时器启动后便无法更新，且计时器值应用于任务的所有任务执行。只要任务执行保持在 IN_PROGRESS 状态的时间长度超过了此间隔，任务执行将失败，并切换为最终 TIMED_OUT 状态。Amazon IoT 还将发布 MQTT 通知。

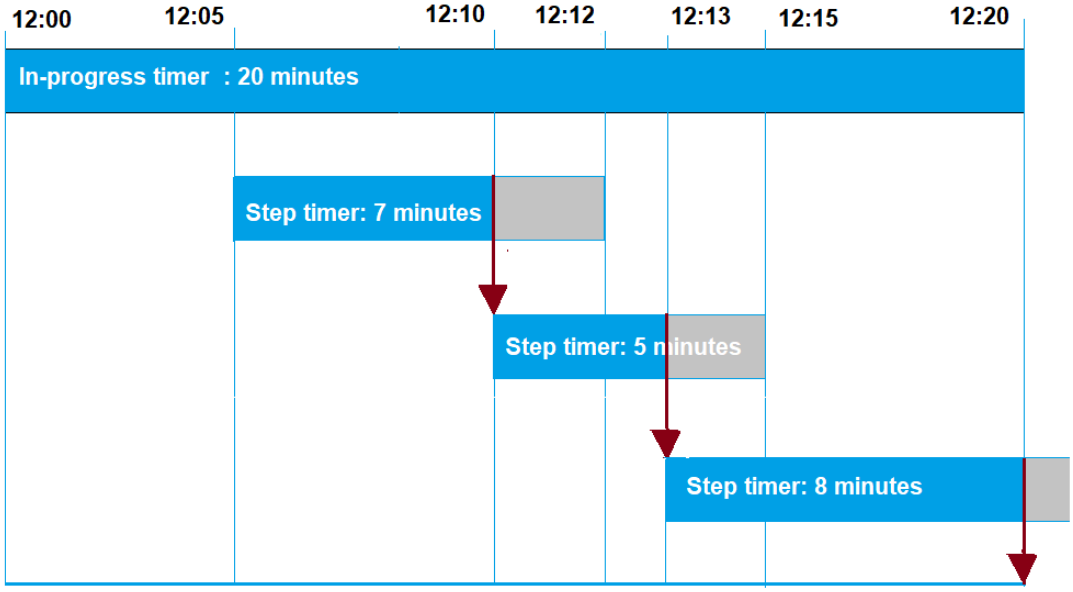
步骤计时器

您还可以设置仅适用于要更新的任务执行的步骤计时器。这种计时器不会对进行中计时器产生影响。您每次更新任务执行时，都可以为此步骤计时器设置新值。为事物启动下一个待处理任务执行时，您还可以创建新的步骤计时器。如果任务执行保持在 IN_PROGRESS 状态的时间长度超过了此步骤计时器间隔，它将失败，并切换为最终 TIMED_OUT 状态。

> **Note**
>
> 您可以使用 Amazon IoT 控制台或 Amazon IoT Jobs API 设置进行中计时器。若要指定步骤计时器，请使用 API。

任务超时计时器工作原理

下图说明了进行中超时计时器与步骤超时计时器在 20 分钟超时期限内相互作用的方式。



下面介绍了不同步骤：

1. 12:00

   此时会创建一个新任务，还会在创建任务时启动 20 分钟的进行中计时器。进行中计时器开始运行，任务执行切换到 IN_PROGRESS 状态。
2. 12:05 PM

   此时会创建一个值为 7 分钟的新步骤计时器。任务执行现在会在 12:12 PM 超时。
3. 12:10 PM

   此时会创建一个值为 5 分钟的新步骤计时器。创建新步骤计时器后，系统会丢弃旧步骤计时器，任务执行现在会在 12:15 PM 超时。
4. 12:13 PM

   此时会创建一个值为 9 分钟的新步骤计时器。系统会丢弃旧步骤计时器，任务执行现在会在 12:20 PM 超时，因为进行中计时器会在 12:20 PM 超时。步骤计时器不能超过进行中计时器的绝对界限。

任务执行重试配置

您可以在满足某组条件时使用重试配置来重试任务执行。如果任务超时或设备报告失败，就可以尝试重试。如果要因超时错误重试执行，则必须启用超时配置。

如何使用重试配置

下面介绍了如何使用重试配置：

1. 确定是否对 FAILED、TIMED_OUT 或这两种失败条件使用重试配置。对于 TIMED_OUT 状态，在报告状态之后，Amazon IoT Jobs 会自动重试设备的任务执行。
2. 对于 FAILED 状态，请检查是否可以重试任务执行失败。如果可以重试，请对设备进行编程，以便向 Amazon IoT 报告 FAILURE 状态。以下部分介绍了有关可重试和不可重试失败的更多信息。
3. 使用前述的信息指定每种失败类型的重试次数。最多可以为单台设备的两种失败类型组合指定 10 次重试。当执行成功或达到指定的尝试次数时，重试尝试会自动停止。
4. 添加中止配置，以便在重复出现重试失败时取消任务，以避免大量重试产生额外费用。

Note

当作业结束重复出现的维护时段时，所有IN_PROGRESS任务执行将继续执行作业文档中确定的操作，直到它们达到终止状态。如果任务执行达到维护时段FAILED或维护时段TIMED_OUT之外的终止状态，则在未用尽重试尝试的情况下，将在下一个维护窗口中进行重试。在下一startTime次维护窗口出现时，将创建新的任务执行并进入状态状态，QUEUED直到设备准备好开始为止。

重试和中止配置

每次重试都会让您的 Amazon Web Services 账户 产生额外费用。为避免重复出现的重试失败产生额外费用，我们建议添加中止配置。有关定价的更多信息，请参阅 Amazon IoT Device Management 定价。

当设备的高阈值百分比超时或报告失败时，您就可能遇到多次重试失败。在这种情况下，您可以使用中止配置取消任务，避免任何排队的任务执行或更多重试尝试。

Note

符合取消任务执行的中止条件仅会取消 QUEUED 任务执行。系统不会尝试设备的任何排队重试。不过，当前具有 IN_PROGRESS 状态的任务执行不会遭到取消。

在重试失败的任务执行之前，我们还建议您检查任务执行失败是否可以重试，如下面的部分所述。

FAILED 失败类型重试

若要重试 FAILED 失败类型，就必须对设备进行编程，以便向 FAILURE 报告失败任务执行 Amazon IoT 状态。您还必须使用重试 FAILED 任务的条件来设置重试配置，以及指定要执行的重试次数。Amazon IoT Jobs 检测到失败状态时，将自动重试该设备的任务执行。任务执行成功或达到最大重试次数，重试才会停止。

您可以跟踪每次重试以及在这些设备上运行的任务。通过跟踪执行状态，您可以在达到指定的重试次数后，使用设备报告失败以及发起另一次重试。

可重试和不可重试失败

任务执行失败可以是可重试或不可重试的失败。每次重试都会让您的 Amazon Web Services 账户 产生费用。为避免因多次重试而产生额外费用，请先检查任务执行失败是否可以重试。可重试失败示例包括设备尝试从 Amazon S3 URL 下载任务文档时发生的连接错误。如果任务执行失败可重试，则可对设备进行编程，以便在出现任务失败时报告 FAILURE 状态，以及将重试配置设置为重试 FAILED 执行。

如果执行不可重试，为避免重试任务执行以及可能让您的账户产生额外费用，我们建议您对设备进行编程，以便向 Amazon IoT 报告 REJECTED 状态。不可重试失败的示例包括：设备与接收任务更新不兼容，或执行任务时遇到内存错误。在这些情况下,Amazon IoT Jobs 不会重试任务执行，因为只有在检测到 FAILED 或 TIMED_OUT 状态时才会重试任务执行。

在确定任务执行失败可重试后，如果重试仍然失败，请考虑检查设备日志。

Note

如果在初始任务创建期间选择了具有可选计划配置的任务，则当该任务达到其 endTime 时，所选的 endBehavior 将停止向目标组中的所有剩余设备推出任务文档，并指示如何继续执行剩余的任务，如果通过重试配置进行了选择，还会指示其重试尝试。

TIMEOUT 失败类型重试

如果在创建任务时启用了超时配置，待状态从 IN_PROGRESS 更改为 TIMED_OUT 时，Amazon IoT Jobs 会重试设备的任务执行。如果进行中计时器超时，或者您指定的步骤计时器处于 IN_PROGRESS 中然后超时，此状态可能会发生更改。任务执行成功或达到此失败类型的最大重试次数后，重试才会停止。

持续任务和事物组成员资格更新

对于任务状态为 IN_PROGRESS 的持续任务，当事物组成员资格更新时，重试次数将重置为零。例如，假设您指定了五次重试，并且已经执行了三次重试。如果现已从事物组中删除事物，然后该事物重新加入该组

（例如动态事物组），则重试次数将重置为零。您现在可以对事物组执行五次重试，而非剩余的两次重试。此外，从事物组中删除事物将取消其他重试次数。

指定其他配置

您可以在创建任务或任务模板时指定其他配置。下文介绍了何时可以指定这些配置。

- 创建自定义任务模板时。从模板创建任务时，系统会保存您指定的其他配置设置。
- 使用任务文件创建自定义任务时。任务文件可以是在 S3 存储桶中上传的 JSON 文件。
- 使用自定义任务模板创建自定义任务时。如果模板已经指定了这些设置，则可以重复使用，也可以通过指定新的配置设置将其覆盖。
- 使用 Amazon 托管式模板创建自定义任务时。

主题

- 使用 Amazon Web Services Management Console 指定任务配置 (p. 680)
- 使用 Amazon IoT Jobs API 指定任务配置 (p. 682)

使用 Amazon Web Services Management Console 指定任务配置

您可以使用 Amazon IoT 控制台为任务添加不同配置。创建任务后，您可以在任务详细信息页面上查看任务配置的状态详细信息。有关不同配置及其工作方式的更多信息，请参阅 任务配置的工作原理 (p. 672)。

创建任务或任务模板时添加任务配置。

创建自定义任务模板

创建自定义任务模板时指定推出配置

1. 转至 Amazon IoT 控制台的任务模板中心，然后选择 Create job template（创建任务模板）。
2. 指定任务模板属性、提供任务文档、展开要添加的配置，然后指定配置参数。

创建自定义任务

创建自定义任务时指定推出配置

1. 转至 Amazon IoT 控制台的 Job 中心，然后选择 Create job（创建任务）。
2. 请选择 Create a custom job（创建自定义任务），指定任务属性、目标，然后指定任务文档使用任务文件还是模板。您可以使用自定义模板或 Amazon 托管式模板。
3. 请选择任务配置，展开 Rollout configuration（推出部署），再在其中指定使用 Constant rate（恒定速率）或 Exponential rate（指数速率）。然后，指定配置参数。

下一部分介绍可以为每个配置指定的参数。

推出配置

您可以指定使用恒定推出速率还是指数速率。

- 设置恒定推出速率

  要为任务执行设置恒定速率，请选择 Constant rate（恒定速率），然后为速率上限指定 Maximum per minute（每分钟的最大值）。此值为可选值，范围从 1 到 1000。如果不设置该值，则会使用 1000 作为默认值。
- 设置指数推出速率

若要设置指数速率，请选择 Exponential rate（指数速率），然后指定这些参数：

- 每分钟的基本速率

  指 Number of notified devices（通知设备数）或 Number of succeeded devices（成功设备数）阈值满足 Rate increase criteria（速率增加条件）之前，执行作业的速率。
- 增量因子

  指 Number of notified devices（通知设备数）或 Number of succeeded devices（成功设备数）阈值满足 Rate increase criteria（速率增加条件）之后，推出速率增加的指数因子。
- 速率增加条件

  Number of notified devices（通知设备数）或 Number of succeeded devices（成功设备数）的阈值。

中止配置

请选择 Add new configuration（添加新配置），然后为每个配置指定以下参数：

- 失败类型

  指定启动任务中止的失败类型。其中包括 FAILED（失败）、REJECTED（已被拒绝）、TIMED_OUT（超时）或 ALL（全部）。
- 增量因子

  指定在满足任务中止条件之前必须完成的任务执行数量。
- 阈值百分比

  指定启动任务中止的已执行事物的总数。

计划配置

每项任务都可以在初始创建后立即启动，计划在以后的某个日期和时间启动，也可以在定期维护时段内生效。

请选择 Add new configuration（添加新配置），然后为每个配置指定以下参数：

- 任务开始

  指定任务开始的日期和时间。
- 周期性维护时段

  定期维护窗口定义了任务可以将作业文档部署到作业中的目标设备的特定日期和时间。维护时段可以每日、每日、每月或每日。
- 任务结束

  指定任务结束的日期和时间。
- 任务结束行为

  为所有未完成的任务执行选择任务结束时的结束行为。

  Note

  当具有可选计划配置和所选结束时间的任务达到结束时间时，该任务将停止向目标组中的所有剩余设备推出任务文档，利用所选结束行为来确定如何继续执行剩余的任务，并根据重试配置确定重试。

超时配置

默认无超时，任务运行会遭到取消或删除。若要使用超时，请选择 Enable timeout（启用超时）,然后指定 1 分钟到 7 天之间的超时值。

重试配置

> Note
>
> 创建任务后，便无法更新重试次数。您只能删除所有失败类型的重试配置。创建任务时，请考虑为配置采用适当的重试次数。为避免因潜在的重试失败产生额外成本，请添加中止配置。

请选择 Add new configuration（添加新配置），然后为每个配置指定以下参数：

- 失败类型

  指定应触发任务执行重试的失败类型。其中包括 Failed（失败）、Timeout（超时）和 All（全部）。
- 重试次数

  请为所选择的 Failure type（失败类型）指定重试次数。最多可以为两种失败类型组合指定 10 次重试。

使用 Amazon IoT Jobs API 指定任务配置

您可以使用CreateJob或 CreateJobTemplateAPI 来指定不同的作业配置。下面介绍了何时可以添加这些配置。添加配置后，您可以使用JobExecutionSummary和JobExecutionSummaryForJob来查看其状态。

有关不同配置及其工作方式的更多信息，请参阅 任务配置的工作原理 (p. 672)。

推出配置

您可以为推出配置指定恒定推出速率或指数推出速率。

- 设置恒定推出速率

  若要设置恒定推出速率，请使用 `JobExecutionsRolloutConfig` 对象将 `maximumPerMinute` 参数添加到 `CreateJob` 请求。该参数会指定任务执行的速率上限。此值为可选值，范围从 1 到 1000。如果不设置该值，则会使用 1000 作为默认值。

```
   "jobExecutionsRolloutConfig": {
       "maximumPerMinute": 1000
   }
```

- 设置指数推出速率

  若要设置任务可变推出速率，请使用 `JobExecutionsRolloutConfig` 对象。您可以在运行 `ExponentialRolloutRate` API 操作时配置 `CreateJob` 属性。以下示例使用 `exponentialRate` 参数设置指数推出速率。有关这些参数的更多信息，请参阅 `ExponentialRolloutRate`

```
{
...
  "jobExecutionsRolloutConfig": {
    "exponentialRate": {
      "baseRatePerMinute": 50,
      "incrementFactor": 2,
      "rateIncreaseCriteria": {
        "numberOfNotifiedThings": 1000,
        "numberOfSucceededThings": 1000
```

```
      },
      "maximumPerMinute": 1000
    }
  }
...
}
```

其中，参数：

baseRatePer分钟

指定达到 numberOfNotifiedThings 或 numberOfSucceededThings 阈值之前任务的执行速率。

incrementFactor

指定达到 numberOfNotifiedThings 或 numberOfSucceededThings 阈值之后推出速率增加的指数因子。

rateIncreaseCriteria

指定 numberOfNotifiedThings 或 numberOfSucceededThings 阈值。

中止配置

若要使用 API 添加此配置，请在运行 CreateJob 或 CreateJobTemplate API 操作时指定 AbortConfig 参数。以下示例介绍了任务推出的中止配置，该任务经历了如 CreateJob API 操作指定的多次失败执行。

Note

删除任务执行会影响已完成执行总数的计算值。当任务中止时，服务会创建自动的 comment 和 reasonCode 以将用户驱动的取消与任务中止取消区分开来。

```
"abortConfig": {
   "criteriaList": [
      {
         "action": "CANCEL",
         "failureType": "FAILED",
         "minNumberOfExecutedThings": 100,
         "thresholdPercentage": 20
      },
      {
         "action": "CANCEL",
         "failureType": "TIMED_OUT",
         "minNumberOfExecutedThings": 200,
         "thresholdPercentage": 50
      }
   ]
}
```

其中，参数：

action

指定满足中止条件时要采取的操作。此参数是必需的，并且 CANCEL 是唯一的有效值。

failureType

指定哪种失败类型应启动任务中止。有效值为 FAILED、REJECTED、TIMED_OUT 和 ALL。

minNumberOfExecutedThings

指定在满足任务中止条件之前必须完成的任务执行数量。在此示例中，Amazon IoT 直到至少 100 个设备已完成任务执行时才检查是否应执行任务中止。

thresholdPercentage

指定可启动任务中止的已执行任务的事物总数。在此示例中，Amazon IoT 会按顺序检查；如果达到阈值百分比，则启动任务中止。如果完成 100 次执行后至少有 20% 的完整执行失败，则会取消任务推出。如果不符合该条件，在完成 200 次执行后，Amazon IoT 会检查是否至少有 50% 的已完成执行超时。如果是这样，则会取消任务推出。

计划配置

要使用此 API 添加此配置，请在运行 CreateJob 或 CreateJobTemplate API 操作时指定可选 SchedulingConfig。

```
   "SchedulingConfig": {
     "endBehavior": string
     "endTime": string
     "maintenanceWindows": string
     "startTime": string
   }
```

其中，参数：

startTime

指定任务开始的日期和时间。

endTime

指定任务结束的日期和时间。

维护窗口

指定是否为计划任务选择了可选维护时段，以向目标组中的所有设备推出任务文档。的字符串格式maintenanceWindow为 YYYY/MM/DD 表示日期，hh: mm 表示时间。

终结行为

指定计划任务到达 endTime 时的任务行为。

Note

任务的可选 SchedulingConfig 可在 DescribeJob 和 DescribeJobTemplate API 中查看。

超时配置

若要使用 API 添加此配置，请在运行 CreateJob 或 CreateJobTemplate API 操作时指定 TimeoutConfig 参数。

使用超时配置

1. 若要在创建任务或任务模板时设置进行中计时器，请为可选TimeoutConfig对象的inProgressTimeoutInMinutes属性设置一个值。

```
   "timeoutConfig": {
     "inProgressTimeoutInMinutes": number
```

```
    }
```

2. 若要为任务执行设置步骤计时器，请在调用stepTimeoutInMinutes时设置一个值UpdateJobExecution。步骤计时器仅应用于您更新的任务执行。您每次更新任务执行时，可以为此计时器设置新值。

   Note

   UpdateJobExecution 也可以通过创建值为 -1 的步骤计时器来丢弃已经创建的步骤计时器。

```
{
   ...
    "statusDetails": {
      "string" : "string"
   },
   "stepTimeoutInMinutes": number
}
```

3. 若要创建新的步骤计时器，您也可以调用 StartNextPendingJobExecutionAPI 操作。

重试配置

Note

创建任务时，请考虑为配置采用适当的重试次数。为避免因潜在的重试失败产生额外成本，请添加中止配置。创建任务后，便无法更新重试次数。您只能使用 UpdateJobAPI 操作将重试次数设置为 0。

若要使用 API 添加此配置，请在运行 `CreateJob` 或 `CreateJobTemplate` API 操作时指定 `jobExecutionsRetryConfig` 参数。

```
{
...
  "jobExecutionsRetryConfig": {
      "criteriaList": [
         {
            "failureType": "string",
            "numberOfRetries": number
         }
      ]
  }
...
}
```

其中 criteriaList 是指定条件列表的数组，其可确定任务的每种失败类型允许的重试次数。

设备和任务

设备可以使用 MQTT、HTTP 签名版本 4 或 HTTP TLS 的任务与 Amazon IoT Jobs 通信。要确定设备与 Amazon IoT Jobs 通信时使用的端点，请运行 DescribeEndpoint 命令。例如，如果您运行此命令：

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

您会获得与以下内容相似的结果：

```
{
    "endpointAddress": "a1b2c3d4e5f6g7-ats.iot.us-west-2.amazonaws.com"
```

```
}
```

使用 MQTT 协议

设备可以用 MQTT 协议的任务与 Amazon IoT Jobs 通信。设备将订阅 MQTT 主题，以便获得新任务通知和接收来自 Amazon IoT Jobs 服务的响应。设备在 MQTT 主题上发布以查询或更新任务执行的状态。每台设备都有自己的一般 MQTT 主题。有关发布和订阅 MQTT 主题的更多信息，请参阅 设备通信协议 (p. 75)。

利用此通信方法，您的设备可使用其特定于设备的证书和私有密钥向 Amazon IoT Jobs 服务进行身份验证。

您的设备可以订阅以下主题。`thing-name` 是与设备关联的事物名称。

- **`$aws/things/thing-name/jobs/notify`**

  设备订阅此主题，在某个事物的待处理执行列表中添加或删除任务执行时接收通知。
- **`$aws/things/thing-name/jobs/notify-next`**

  订阅此主题，以在下一个待处理任务执行发生更改时通知您。
- **`$aws/things/thing-name/request-name/accepted`**

  Amazon IoT Jobs 服务在 MQTT 主题上发布成功和失败消息。通过将 `accepted` 或 `rejected` 追加到用于发出请求的主题来构成该主题。此处，`request-name` 是请求的名称，例如 `Get` 和主题可以是：`$aws/things/myThing/jobs/get`.Amazon IoT 然后，Jobs 将在 `$aws/things/myThing/jobs/get/accepted` 主题上发布成功消息。
- **`$aws/things/thing-name/request-name/rejected`**

  此处，`request-name` 是请求的名称，例如 `Get`。如果请求失败，Amazon IoT Jobs 将在 `$aws/things/myThing/jobs/get/rejected` 主题上发布失败消息。

您还可以使用以下 HTTPS API 操作：

- 通过调用 `UpdateJobExecution` API 来更新任务执行的状态。
- 通过调用 `DescribeJobExecution` API 来查询任务执行的状态。
- 通过调用 `GetPendingJobExecutions` API 来检索待处理任务执行的列表。
- 通过使用 `jobId` 作为 `$next` 调用 `DescribeJobExecution` API 来检索下一个待定任务执行。
- 通过调用 `StartNextPendingJobExecution` API 来获取和开启下一个待处理任务执行。

使用 HTTP Signature 版本 4

设备可以使用端口443 上的 HTTP 签名版本 4 与 Amazon IoT Jobs 通信。这是由Amazon SDK 和 CLI 使用的方法。有关这些工具的更多信息，请参阅Amazon CLI命令引用： iot-jobs-data或 AmazonSDK 和工具，并参考您的首选语言的 IotJobsDataPlane 部分。

利用这种通信方法，您的设备将使用 IAM 凭证向 Amazon IoT Jobs 进行身份验证。

可以使用此方法执行以下命令：

- DescribeJobExecution

  ```
  aws iot-jobs-data describe-job-execution ...
  ```
- GetPendingJobExecutions

  ```
  aws iot-jobs-data get-pending-job-executions ...
  ```
- StartNextPendingJobExecution

```
aws iot-jobs-data start-next-pending-job-execution ...
```

- UpdateJobExecution

```
aws iot-jobs-data update-job-execution ...
```

使用 HTTP TLS

设备可以使用支持此协议的第三方软件客户端在端口8443上使用 HTTP TLS 与 Amazon IoT Jobs 通信。

利用此方法，您的设备可使用基于 X.509 证书的身份验证（例如，其特定于设备的证书和私有密钥）。

可以使用此方法执行以下命令：

- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

对设备进行编程以使用任务

本部分中的示例使用 MQTT 来演示设备如何使用 Amazon IoT Jobs 服务。或者，您也可以使用对应的 API 或 CLI 命令。对于这些示例，我们假定名为 `MyThing` 的设备订阅以下 MQTT 主题：

- `$aws/things/MyThing/jobs/notify` –、 或 `$aws/things/MyThing/jobs/notify-next`
- `$aws/things/MyThing/jobs/get/accepted`
- `$aws/things/MyThing/jobs/get/rejected`
- `$aws/things/MyThing/jobs/jobId/get/accepted`
- `$aws/things/MyThing/jobs/jobId/get/rejected`

如果您使用 Amazon IoT 的代码签名，则设备代码必须验证代码文件的签名。该签名在任务文档中的 `codesign` 属性内。有关验证代码文件签名的更多信息，请参阅设备代理示例。

主题

- 设备工作流程 (p. 687)
- 任务流 (p. 688)
- 任务通知 (p. 691)

设备工作流程

设备可以使用以下任一方式处理它运行的任务。

- 获取下一个任务

  1. 当设备首次联机时，它应订阅设备的 `notify-next` 主题。
  2. 使用 jobId `$next` 调用 DescribeJobExecution (p. 714) MQTT API 以获取下一个任务、其任务文档和其他详细信息，包括保存在 `statusDetails` 中的任何状态。如果任务文档具有代码文件签名，则您必须验证签名，然后才能继续处理任务请求。
  3. 调用 UpdateJobExecution (p. 715) MQTT API 以更新任务状态。或者，要在一个调用中将此步骤与上一步骤合并，设备可调用 StartNextPendingJobExecution (p. 713)。

4. （可选）您可在调用 UpdateJobExecution (p. 715) 或 StartNextPendingJobExecution (p. 713) 时为 `stepTimeoutInMinutes` 设置值来添加步骤计时器。
5. 使用 UpdateJobExecution (p. 715) MQTT API 执行任务文档所指定的操作以报告任务的进度。
6. 通过使用此 jobId 调用 DescribeJobExecution (p. 714) MQTT API 来继续监控任务执行。如果删除了任务执行，DescribeJobExecution (p. 714) 将返回 `ResourceNotFoundException`。

   如果在设备运行任务时取消或删除了任务执行，则设备应能够恢复到有效状态。
7. 在任务完成后调用 UpdateJobExecution (p. 715) MQTT API 以更新任务状态并报告成功或失败。
8. 由于此任务的执行状态已更改为最终状态，因此，可用于执行的下一个任务（如果有）将发生更改。设备将收到下一个待处理任务执行已更改的通知。此时，设备将按照步骤 2 中所述继续操作。

如果设备保持联机状态，则它在完成任务或添加新的待处理任务执行时，将继续收到下一个待处理任务执行的通知，包括其任务执行数据。在发生此情况时，设备将按照步骤 2 中所述继续操作。

- 从可用任务中选择

1. 当设备首次联机时，它应订阅事物的 `notify` 主题。
2. 调用 GetPendingJobExecutions (p. 713) MQTT API 以获取待处理任务执行的列表。
3. 如果列表中包含一个或多个任务执行，请选择一个任务执行。
4. 调用 DescribeJobExecution (p. 714) MQTT API 以获取任务文档和其它详细信息，包括保存在 `statusDetails` 中的任何状态。
5. 调用 UpdateJobExecution (p. 715) MQTT API 以更新任务状态。如果在此命令中将 `includeJobDocument` 字段设置为 `true`，则设备可跳过上一个步骤并在此时检索任务文档。
6. （可选）您可在调用 UpdateJobExecution (p. 715) 时为 `stepTimeoutInMinutes` 设置值来添加步骤计时器。
7. 使用 UpdateJobExecution (p. 715) MQTT API 执行任务文档所指定的操作以报告任务的进度。
8. 通过使用此 jobId 调用 DescribeJobExecution (p. 714) MQTT API 来继续监控任务执行。如果在设备运行任务时取消或删除了任务执行，则设备应能够恢复到有效状态。
9. 在任务完成后调用 UpdateJobExecution (p. 715) MQTT API 以更新任务状态并报告成功或失败。

如果设备保持联机状态，则在一个新的待处理任务执行变为可用时，它将收到所有待处理任务执行的通知。在发生此情况时，设备可按照步骤 2 中所述继续操作。

如果设备无法执行任务，它将调用 UpdateJobExecution (p. 715) MQTT API 来将任务状态更新为 `REJECTED`。

任务流

下面显示了任务工作流程中的不同步骤，从启动新任务到报告任务执行的完成状态。

开始新任务

在创建新任务时，Amazon IoT Jobs 将在 `$aws/things/`*`thing-name`*`/jobs/notify` 主题上为每个目标设备发布一条消息。

消息包含以下信息：

```
{
    "timestamp":1476214217017,
    "jobs":{
        "QUEUED":[{
            "jobId":"0001",
            "queuedAt":1476214216981,
```

```
            "lastUpdatedAt":1476214216981,
            "versionNumber" : 1
        }]
    }
}
```

在对任务执行进行排队时，设备将在 '$aws/things/*thingName*/jobs/notify' 主题上收到此消息。

Note

对于具有可选 SchedulingConfig 的任务，该任务将保持初始状态 SCHEDULED。当任务达到所选的 startTime 时，将发生以下情况：

- 任务状态将更新为 IN_PROGRESS。
- 任务将开始向目标组中的所有设备推出任务文档。

获取任务信息

要获取有关任务执行的更多信息，请调用 DescribeJobExecution (p. 714) MQTT API 并将 includeJobDocument 字段设置为 true（原定设置值）。

如果请求成功，Amazon IoT Jobs 服务将在 $aws/things/MyThing/jobs/0023/get/accepted 主题上发布消息：

```
{
    "clientToken" : "client-001",
    "timestamp" : 1489097434407,
    "execution" : {
        "approximateSecondsBeforeTimedOut": number,
        "jobId" : "023",
        "status" : "QUEUED",
        "queuedAt" : 1489097374841,
        "lastUpdatedAt" : 1489097374841,
        "versionNumber" : 1,
        "jobDocument" : {
            < contents of job document >
        }
    }
}
```

如果请求失败，Amazon IoT Jobs 服务将在 $aws/things/MyThing/jobs/0023/get/rejected 主题上发布消息。

设备现在已有任务文档，它可使用该文档执行任务的远程操作。如果任务文档包含一个 Amazon S3 预签名 URL，则设备可使用该 URL 下载任务的任何所需文件。

报告任务执行状态

当设备执行任务时，它可调用 UpdateJobExecution (p. 715) MQTT API 来更新任务执行的状态。

例如，设备可通过在 IN_PROGRESS 主题上发布以下消息来将任务执行状态更新为 $aws/things/MyThing/jobs/0023/update：

```
{
    "status":"IN_PROGRESS",
    "statusDetails": {
        "progress":"50%"
    },
```

```
    "expectedVersion":"1",
    "clientToken":"client001"
}
```

Jobs 通过将消息发布到 $aws/things/MyThing/jobs/0023/update/accepted 或 $aws/things/MyThing/jobs/0023/update/rejected 主题来做出响应：

```
{
    "clientToken":"client001",
    "timestamp":1476289222841
}
```

设备可通过调用 StartNextPendingJobExecution (p. 713) 来合并两个以前的请求。这将获取并开始下一个待处理任务执行，并允许设备更新任务执行状态。此请求还在有待处理任务执行时返回任务文档。

如果任务中包含 TimeoutConfig，则进行中计时器将开始运行。您还可以通过为调用时设置一个值，为任务执行设置步骤计stepTimeoutInMinutes时器UpdateJobExecution。步骤计时器仅应用于您更新的任务执行。您每次更新任务执行时，可以为此计时器设置新值。您还可以在呼叫时创建计步器StartNextPendingJobExecution。如果任务执行保持在 IN_PROGRESS 状态的时间长度超过了此步骤计时器间隔，它将失败，并切换为最终 TIMED_OUT 状态。步骤计时器对您在创建任务时设置的进行中计时器没有任何影响。

status 字段可设置为 IN_PROGRESS、SUCCEEDED 或 FAILED。您无法更新已处于最终状态的任务执行的状态。

报告执行已完成

在设备执行完任务后，它将调用 UpdateJobExecution (p. 715) MQTT API。如果任务已成功，则将 status 设置为 SUCCEEDED，并在消息负载的 statusDetails 中，将有关任务的其它信息作为名称-值对添加。任务执行完成时，进行中计时器和步骤计时器结束。

例如：

```
{
    "status":"SUCCEEDED",
    "statusDetails": {
        "progress":"100%"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

如果任务未成功，则将 status 设置为 FAILED，并在 statusDetails 中，添加有关出现的错误的信息：

```
{
    "status":"FAILED",
    "statusDetails": {
        "errorCode":"101",
        "errorMsg":"Unable to install update"
    },
    "expectedVersion":"2",
    "clientToken":"client-001"
}
```

Note

statusDetails 属性可包含任意数量的名称-值对。

在 Amazon IoT Jobs 服务收到此更新时，它会在 $aws/things/MyThing/jobs/notify 主题上发布消息以指示任务执行已完成：

```
{
    "timestamp":1476290692776,
    "jobs":{}
}
```

其它任务

如果设备有其它待处理的任务执行，这些任务执行将包含在发布到 $aws/things/MyThing/jobs/notify 的消息中。

例如：

```
{
    "timestamp":1476290692776,
    "jobs":{
        "QUEUED":[{
            "jobId":"0002",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }],
        "IN_PROGRESS":[{
            "jobId":"0003",
            "queuedAt":1476290646230,
            "lastUpdatedAt":1476290646230
        }]
    }
}
```

任务通知

当任务待处理或列表中的第一个任务执行更改时，Amazon IoT Jobs 服务将向保留主题发布 MQTT 消息。设备可通过订阅这些主题来跟踪待处理任务。

任务通知类型。

任务通知将作为 JSON 负载发布到 MQTT 主题。通知有两种：

ListNotification

AListNotification 包含一个列表，其中待处理的任务执行不超过 15 个。它们依次按照状态（IN_PROGRESS 任务执行在 QUEUED 任务执行之前）和排队的时间进行排序。

如果满足以下条件之一，则发布 ListNotification。

- 新的任务执行已排队或更改为非最终状态（IN_PROGRESS 或 QUEUED）。
- 旧的任务执行更改为最终状态（FAILED、SUCCEEDED、CANCELED、TIMED_OUT、REJECTED 或 REMOVED）。

列表通知（**QUEUED**或中最多有 15 个待处理的任务执行**IN_PROGRESS**）	
没有可选的计划配置和定期维护窗口 （最多 10 次任务执行）	具有可选的计划配置和定期维护窗口 （最多 5 个任务执行）

列表通知（**QUEUED**或中最多有 15 个待处理的任务执行**IN_PROGRESS**）	
总是出现在 ListNotification.	仅在维护 ListNotification 时段内出现。

NextNotification

- NextNotification 包含队列中下一个任务执行的摘要信息。

  每当列表中第一个任务执行更改时，则会发布 NextNotification。
  - 新的任务执行作为 QUEUED 被添加到列表，并且在列表中排在第一。
  - 不是列表第一的现有任务执行的状态从 QUEUED 更改为 IN_PROGRESS，然后成为列表中的第一个。（当列表中没有其它 IN_PROGRESS 任务执行时，或者当状态从 QUEUED 更改为 IN_PROGRESS 的任务执行排队的时间早于列表中其他所有 IN_PROGRESS 任务执行时，会出现这种情况。）
  - 列表第一的任务执行状态更改为最终状态，并且从列表中删除。

有关发布和订阅 MQTT 主题的更多信息，请参阅 the section called “设备通信协议” (p. 75)。

> **Note**
>
> 当您使用 HTTP Signature Version 4 或 HTTP TLS 与 Jobs 通信时，通知不可用。

任务待处理

当某个事物的待处理任务执行列表中添加或删除了任务时，或者列表中的第一个任务执行更改时，Amazon IoT Jobs 服务会在 MQTT 主题上发布消息：

- $aws/things/*thingName*/jobs/notify
- $aws/things/*thingName*/jobs/notify-next

消息包含以下示例负载：

$aws/things/*thingName*/jobs/notify:

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : [ {
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0
    } ]
  }
}
```

如果调用的任务执行this-job源自选定可选计划配置的作业，并且任务文档的发布计划在维护时段内进行，则该任务将仅在定期维护时段内出现。在维护窗口之外，调用的作业this-job将从待处理任务执行列表中排除，如下所示。

```
{
  "timestamp" : 10011,
  "jobs" : {
    "IN_PROGRESS" : [ {
      "jobId" : "other-job",
      "queuedAt" : 10003,
      "lastUpdatedAt" : 10009,
      "executionNumber" : 1,
      "versionNumber" : 1
    } ],
    "QUEUED" : []
  }
}
```

$aws/things/*thingName*/jobs/notify-next:

```
{
  "timestamp" : 10011,
  "execution" : {
    "jobId" : "other-job",
    "status" : "IN_PROGRESS",
    "queuedAt" : 10009,
    "lastUpdatedAt" : 10009,
    "versionNumber" : 1,
    "executionNumber" : 1,
    "jobDocument" : {"c":"d"}
  }
}
```

如果调用的任务执行other-job源自选定可选计划配置的作业，并且任务文档的发布计划在维护时段内进行，则该任务将仅在定期维护时段内出现。在维护窗口之外，调用的任务other-job不会被列为下一次执行的任务，如下所示。

```
{} //No other pending jobs
```

```
{
  "timestamp" : 10011,
  "execution" : {
      "jobId" : "this-job",
      "queuedAt" : 10011,
      "lastUpdatedAt" : 10011,
      "executionNumber" : 1,
      "versionNumber" : 0,
      "jobDocument" : {"a":"b"}
  }
} // "this-job" is pending next to "other-job"
```

可能的任务执行状态值为
QUEUED、IN_PROGRESS、FAILED、SUCCEEDED、CANCELED、TIMED_OUT、REJECTED 和 REMOVED。

以下一系列示例显示了，在创建任务执行和任务执行从一种状态更改为另一种状态时向每个主题发布的通知。

首先，创建了一个名为 job1 的任务。此通知发布到 jobs/notify 主题：

```
{
  "timestamp": 1517016948,
  "jobs": {
    "QUEUED": [
```

```
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

此通知发布到 jobs/notify-next 主题：

```
{
  "timestamp": 1517016948,
  "execution": {
    "jobId": "job1",
    "status": "QUEUED",
    "queuedAt": 1517016947,
    "lastUpdatedAt": 1517016947,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

当创建另一个任务 (job2) 时，此通知将发布到 jobs/notify 主题：

```
{
  "timestamp": 1517017192,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

通知未发布到 jobs/notify-next 主题，因为队列中的下一个任务 (job1) 尚未更改。当 job1 开始执行时，其状态更改为 IN_PROGRESS。没有发布任何通知，因为任务列表和队列中的下一个任务尚未更改。

当添加第三个任务 (job3) 时，此通知将发布到 jobs/notify 主题：

```
{
  "timestamp": 1517017906,
  "jobs": {
    "IN_PROGRESS": [
```

```
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517017472,
        "startedAt": 1517017472,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ],
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

通知未发布到 jobs/notify-next 主题，因为队列中的下一个任务仍为 job1。

当 job1 完成后，其状态更改为 SUCCEEDED，此通知将发布到 jobs/notify 主题：

```
{
  "timestamp": 1517186269,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517017905,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

此时，已从队列中删除 job1，要执行的下一个任务是 job2。此通知发布到 jobs/notify-next 主题：

```
{
  "timestamp": 1517186269,
  "execution": {
    "jobId": "job2",
    "status": "QUEUED",
    "queuedAt": 1517017191,
    "lastUpdatedAt": 1517017191,
```

```
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

如果 job3 必须在 job2 之前开始执行（不推荐），job3 的状态可以更改为 IN_PROGRESS。更改后，job2 则不再是队列中的下一个任务，此通知将发布到 jobs/notify-next 主题：

```
{
  "timestamp": 1517186779,
  "execution": {
    "jobId": "job3",
    "status": "IN_PROGRESS",
    "queuedAt": 1517017905,
    "startedAt": 1517186779,
    "lastUpdatedAt": 1517186779,
    "versionNumber": 2,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
```

没有向 jobs/notify 主题发布任何通知，因为没有添加或删除任何任务。

如果设备拒绝 job2，并且将其状态更新为 REJECTED，则此通知将发布到 jobs/notify 主题：

```
{
  "timestamp": 1517189392,
  "jobs": {
    "IN_PROGRESS": [
      {
        "jobId": "job3",
        "queuedAt": 1517017905,
        "lastUpdatedAt": 1517186779,
        "startedAt": 1517186779,
        "executionNumber": 1,
        "versionNumber": 2
      }
    ]
  }
}
```

如果 job3（仍在进行）被强制删除，则此通知将发布到 jobs/notify 主题：

```
{
  "timestamp": 1517189551,
  "jobs": {}
}
```

此时队列为空。此通知发布到 jobs/notify-next 主题：

```
{
  "timestamp": 1517189551
}
```

Amazon IoT 任务 API 操作

Amazon IoT 任务 API 可用于以下任一类别：

- 用于管理任务（例如管理和控制任务）的 API 操作。这是控制面板。
- 用于执行这些任务的设备的 API 操作。这是数据层面，允许您发送和接收数据。

任务的管理和控制操作将使用 HTTPS 协议 API。设备可以使用 MQTT 或 HTTPS 协议 API。控制面板 API 旨在用于在创建和跟踪任务时通常进行的少量调用。它通常会为一个请求打开一个连接，然后在收到响应后关闭此连接。数据层面 HTTPS 和 MQTT API 允许进行长时间轮询。这些 API 操作旨在用于可扩展至数百万台设备的大量流量。

每个 Amazon IoT Jobs HTTPS API 均有一个对应的命令，您可使用该命令从 Amazon CLI 调用该 API。命令采用小写形式，并且在构成 API 名称的单词之间使用连字符。例如，您可以通过键入以下内容，在 CLI 上调用 `CreateJob` API：

```
aws iot create-job ...
```

如果在操作过程中发生错误，您会收到包含有关错误的信息的错误响应。

ErrorResponse

包含有关 Amazon IoT Jobs 服务操作期间发生的错误的信息。

以下示例显示此操作的语法：

```
{
    "code": "ErrorCode",
    "message": "string",
    "clientToken": "string",
    "timestamp": timestamp,
    "executionState": JobExecutionState
}
```

以下是此 `ErrorResponse` 的描述：

`code`

可将 ErrorCode 设置为：

InvalidTopic

请求发送至 Amazon IoT 任务命名空间中未映射到任何 API 操作的主题。

InvalidJson

请求的内容无法解释为有效的 UTF-8 编码的 JSON。

InvalidRequest

请求的内容无效。例如，当 `UpdateJobExecution` 请求包含无效的状态详细信息时，将返回此代码。消息包含有关错误的详细信息。

InvalidStateTransition

更新尝试将任务执行更改为某个状态，但该状态由于任务执行的当前状态而无效（例如，尝试将处于 SUCCEEDED 状态的请求更改为 IN_PROGRESS 状态）。在这种情况下，错误消息的正文还包含 `executionState` 字段。

ResourceNotFound

请求主题所指定的 JobExecution 不存在。

VersionMismatch

请求中指定的预期版本与 Amazon IoT Jobs 服务中的任务执行版本不匹配。在这种情况下，错误消息的正文还包含 executionState 字段。

InternalError

处理请求期间出现内部错误。

RequestThrottled

请求已被阻止。

TerminalStateReached

在处于最终状态的任务上执行描述任务的命令时发生。

message

错误消息字符串。

clientToken

用于将请求与其答复关联起来的任意字符串。

timestamp

用自纪元以来的秒数表示的时间。

executionState

一个 JobExecutionState 对象。仅在 code 字段具有 InvalidStateTransition 或 VersionMismatch 值时包含此字段。在这些情况下，这使得不必执行单独的 DescribeJobExecution 请求以获取当前任务执行状态数据。

下面列出了任务 API 操作和数据类型。

- 任务管理和控制 API 以及数据类型 (p. 698)
- 任务设备 MQTT 和 HTTPS API 操作以及数据类型 (p. 711)

任务管理和控制 API 以及数据类型

以下命令可供 CLI 中的 Job 管理和控制和通过 HTTPS 协议使用。

- 任务管理和控制数据类型 (p. 699)
- 任务管理和控制 API 操作 (p. 701)

要确定 CLI 命令的 *endpoint-url* 参数，请运行此命令。

```
aws iot describe-endpoint --endpoint-type=iot:Jobs
```

此命令将返回以下输出。

```
{
"endpointAddress": "account-specific-prefix.jobs.iot.aws-region.amazonaws.com"
}
```

Note

Jobs 终端节点不支持 ALPN `z-amzn-http-ca`。

任务管理和控制数据类型

管理和控制应用程序使用以下数据类型与 Amazon IoT Jobs 进行通信。

任务

Job 对象包含有关任务的详细信息。下面显示语法：

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED",
    "forceCanceled": boolean,
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "comment": "string",
    "targets": ["string"],
    "description": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp,
    "jobProcessDetails": {
        "processingTargets": ["string"],
        "numberOfCanceledThings": long,
        "numberOfSucceededThings": long,
        "numberOfFailedThings": long,
        "numberOfRejectedThings": long,
        "numberOfQueuedThings": long,
        "numberOfInProgressThings": long,
        "numberOfRemovedThings": long,
        "numberOfTimedOutThings": long
    },
    "presignedUrlConfig": {
        "expiresInSec": number,
        "roleArn": "string"
    },
    "jobExecutionsRolloutConfig": {
        "exponentialRate": {
           "baseRatePerMinute": integer,
           "incrementFactor": integer,
           "rateIncreaseCriteria": {
              "numberOfNotifiedThings": integer, // Set one or the other
              "numberOfSucceededThings": integer // of these two values.
           },
           "maximumPerMinute": integer
      }
    },
    "abortConfig": {
       "criteriaList": [
          {
             "action": "string",
             "failureType": "string",
             "minNumberOfExecutedThings": integer,
             "thresholdPercentage": integer
          }
       ]
    },
    "SchedulingConfig": {
      "startTime": string
      "endTime": string
      "timeZone": string
```

```
      "endTimeBehavior": string

   },
    "timeoutConfig": {
        "inProgressTimeoutInMinutes": long
    }
}
```

有关更多信息，请参阅 Job 或 job。

JobSummary

JobSummary 对象包含任务摘要。下面显示语法：

```
{
    "jobArn": "string",
    "jobId": "string",
    "status": "IN_PROGRESS|CANCELED|SUCCEEDED|SCHEDULED",
    "targetSelection": "CONTINUOUS|SNAPSHOT",
    "thingGroupId": "string",
    "createdAt": timestamp,
    "lastUpdatedAt": timestamp,
    "completedAt": timestamp
}
```

有关更多信息，请参阅 JobSummary 或 job-summary。

JobExecution

JobExecution 对象表示设备上的任务执行。下面显示语法：

```
{
    "approximateSecondsBeforeTimedOut": 50,
    "executionNumber": 1234567890,
    "forceCanceled": true|false,
    "jobId": "string",
    "lastUpdatedAt": timestamp,
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED",
    "forceCanceled": boolean,
    "statusDetails": {
        "detailsMap": {
            "string": "string" ...
        },
        "status": "string"
    },
    "thingArn": "string",
    "versionNumber": 123
}
```

有关更多信息，请参阅 JobExecution 或 job-execution。

JobExecutionSummary

JobExecutionSummary 对象包含任务执行摘要信息。下面显示语法：

```
{
    "executionNumber": 1234567890,
    "queuedAt": timestamp,
```

```
    "lastUpdatedAt": timestamp,
    "startedAt": timestamp,
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED"
}
```

有关更多信息，请参阅 JobExecutionSummary 或 job-execution-summary。

JobExecutionSummaryForJob

JobExecutionSummaryForJob 对象包含有关特定任务的任务执行的信息摘要。下面显示语法：

```
{
    "executionSummaries": [
        {
            "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyThing",
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            }
        },
        ...
    ]
}
```

有关更多信息，请参阅 JobExecutionSummaryForJob 或 job-execution-summary-for-job。

JobExecutionSummaryForThing

JobExecutionSummaryForThing 对象包含有关特定事物上的任务执行的信息摘要。下面显示语法：

```
{
    "executionSummaries": [
        {
            "jobExecutionSummary": {
                "status": "IN_PROGRESS",
                "lastUpdatedAt": 1549395301.389,
                "queuedAt": 1541526002.609,
                "executionNumber": 1
            },
            "jobId": "MyThingJob"
        },
        ...
    ]
}
```

有关更多信息，请参阅 JobExecutionSummaryForThing 或 job-execution-summary-for-thing。

任务管理和控制 API 操作

使用以下 API 操作或 CLI 命令：

AssociateTargetsWithJob

将组与持续任务关联。必须满足以下标准：

- 必须已在 targetSelection 字段设置为 CONTINUOUS 的情况下创建任务。
- 任务状态当前必须为 IN_PROGRESS。

- 与任务关联的目标总数不能超过 100。

HTTPS request

```
POST /jobs/jobId/targets

{
"targets": [ "string" ],
"comment": "string"
}
```

有关更多信息，请参阅AssociateTargetsWithJob。

CLI syntax

```
aws iot  associate-targets-with-job \
--targets <value> \
--job-id <value> \
[--comment <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"targets": [
"string"
],
"jobId": "string",
"comment": "string"
}
```

有关更多信息，请参阅associate-targets-with-job。

CancelJob

取消任务。

HTTPS request

```
PUT /jobs/jobId/cancel

{
"force": boolean,
"comment": "string",
"reasonCode": "string"
}
```

有关更多信息，请参阅CancelJob。

CLI syntax

```
aws iot cancel-job \
    --job-id <value> \
    [--force <value>]  \
    [--comment <value>]  \
    [--reasonCode <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
    "jobId": "string",
    "force": boolean,
    "comment": "string"
}
```

有关更多信息，请参阅cancel-job。

CancelJobExecution

取消设备上的任务执行。

HTTPS request

```
PUT /things/thingName/jobs/jobId/cancel

{
"force": boolean,
"expectedVersion": "string",
"statusDetails": {
    "string": "string"
    ...
}
}
```

有关更多信息，请参阅CancelJobExecution。

CLI syntax

```
aws iot cancel-job-execution \
--job-id <value> \
--thing-name <value> \
[--force | --no-force] \
[--expected-version <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string",
"thingName": "string",
"force": boolean,
"expectedVersion": long,
"statusDetails": {
"string": "string"
}
}
```

有关更多信息，请参阅cancel-job-execution。

CreateJob

创建任务。您可以将任务文档作为 Amazon S3 存储桶（documentSource 参数）中或请求正文（document 参数）中的文档链接提供。

将可选的 targetSelection 参数设置为 CONTINUOUS，可使任务成为持续任务。（默认值为 SNAPSHOT。）持续任务可用于在将设备添加到组时加入或升级设备，因为它将继续运行并在新添加的事物上执行，甚至在组中的事物在创建任务期间完成任务后也是如此。

任务可以具有可选的 TimeoutConfig，这会设置进行中计时器的值。进行中计时器无法更新，应用到该任务的全部执行。

对 CreateJob API 的参数执行以下验证：

- targets 参数必须是有效事物或事物组 ARN 的列表。所有事物和事物组必须在您的Amazon Web Services 账户中。
- documentSource 参数必须是指向任务文档的有效 Amazon S3 URL。Amazon S3 URL 的格式为：https://s3.amazonaws.com/*bucketName*/*objectName*。
- 存储在由 documentSource 参数指定的 URL 中的文档必须是 UTF-8 编码的 JSON 文档。
- 由于 MQTT 消息大小限制 (128 KB) 和加密的原因，任务文档的大小限制为 32 KB。
- jobId 必须在您的Amazon Web Services 账户中是唯一的。

HTTPS request

```
PUT /jobs/jobId

{
"targets": [ "string" ],
"document": "string",
"documentSource": "string",
"description": "string",
"jobTemplateArn": "string",
"presignedUrlConfigData": {
    "roleArn": "string",
    "expiresInSec": "integer"
},
"targetSelection": "CONTINUOUS|SNAPSHOT",
"jobExecutionsRolloutConfig": {
    "exponentialRate": {
       "baseRatePerMinute": integer,
       "incrementFactor": integer,
       "rateIncreaseCriteria": {
          "numberOfNotifiedThings": integer, // Set one or the other
          "numberOfSucceededThings": integer // of these two values.
       },
       "maximumPerMinute": integer
  }
},
"abortConfig": {
   "criteriaList": [
      {
         "action": "string",
         "failureType": "string",
         "minNumberOfExecutedThings": integer,
         "thresholdPercentage": integer
      }
   ]
},
"SchedulingConfig": {
    "startTime": string
    "endTime": string
    "timeZone": string


    "endTimeBehavior": string
```

```
   }
"timeoutConfig": {
  "inProgressTimeoutInMinutes": long
}
}
```

有关更多信息，请参阅CreateJob。

CLI syntax

```
aws iot create-job \
    --job-id <value> \
    --targets <value> \
    [--document-source <value>] \
    [--document <value>] \
    [--description <value>] \
    [--job-template-arn <value>] \
    [--presigned-url-config <value>] \
    [--target-selection <value>] \
    [--job-executions-rollout-config <value>] \
    [--abort-config <value>] \
    [--timeout-config <value>] \
    [--document-parameters <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
    "jobId": "string",
    "targets": [ "string" ],
    "documentSource": "string",
    "document": "string",
    "description": "string",
    "jobTemplateArn": "string",
    "presignedUrlConfig": {
        "roleArn": "string",
        "expiresInSec": long
     },
    "targetSelection": "string",
    "jobExecutionsRolloutConfig": {
          "exponentialRate": {
              "baseRatePerMinute": integer,
              "incrementFactor": integer,
              "rateIncreaseCriteria": {
                 "numberOfNotifiedThings": integer, // Set one or the other
                 "numberOfSucceededThings": integer // of these two values.
              },
      "maximumPerMinute": integer
      }
    },
    "abortConfig": {
    "criteriaList": [
        {
           "action": "string",
           "failureType": "string",
           "minNumberOfExecutedThings": integer,
           "thresholdPercentage": integer
         }
      ]
    },
    "timeoutConfig": {
          "inProgressTimeoutInMinutes": long
    },
```

```
    "documentParameters": {
    "string": "string"
    }
}
```

有关更多信息，请参阅create-job。

DeleteJob

删除任务及其相关的任务执行。

删除任务可能需要一些时间，具体取决于为该任务创建的任务执行的数量以及其它各种因素。正在删除任务时，任务的状态显示为“DELETION_IN_PROGRESS”。尝试删除或取消已处于“DELETION_IN_PROGRESS”状态的任务将导致错误。

HTTPS request

```
DELETE /jobs/jobId?force=force
```

有关更多信息，请参阅DeleteJob。

CLI syntax

```
aws iot  delete-job \
--job-id <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string",
"force": boolean
}
```

有关更多信息，请参阅delete-job。

DeleteJobExecution

删除任务执行。

HTTPS request

```
DELETE /things/thingName/jobs/jobId/executionNumber/executionNumber?force=force
```

有关更多信息，请参阅DeleteJobExecution。

CLI syntax

```
aws iot  delete-job-execution \
--job-id <value> \
--thing-name <value> \
--execution-number <value> \
[--force | --no-force]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long,
"force": boolean
}
```

有关更多信息，请参阅delete-job-execution。

DescribeJob

获取任务执行的详细信息。

HTTPS request

```
GET /jobs/jobId
```

有关更多信息，请参阅DescribeJob。

CLI syntax

```
aws iot describe-job \
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string"
}
```

有关更多信息，请参阅describe-job。

DescribeJobExecution

获取任务执行的详细信息。任务的执行状态必须为 SUCCEEDED 或 FAILED。

HTTPS request

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

有关更多信息，请参阅DescribeJobExecution。

CLI syntax

```
aws iot  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string",
"thingName": "string",
"executionNumber": long
}
```

有关更多信息，请参阅describe-job-execution。

GetJobDocument

获取任务的任务文档。

Note

占位符 URL 不会替换为返回文档中的预签名 Amazon S3 URL。仅在 Amazon IoT Jobs 服务通过 MQTT 接收请求时生成预签名 URL。

HTTPS request

```
GET /jobs/jobId/job-document
```

有关更多信息，请参阅GetJobDocument。

CLI syntax

```
aws iot get-job-document \
--job-id <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string"
}
```

有关更多信息，请参阅get-job-document。

ListJobExecutionsForJob

获取任务的任务执行的列表。

HTTPS request

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

有关更多信息，请参阅ListJobExecutionsForJob。

CLI syntax

```
aws iot  list-job-executions-for-job \
--job-id <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

有关更多信息，请参阅list-job-executions-for-job。

ListJobExecutionsForThing

获取事物的任务执行的列表。

HTTPS request

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

有关更多信息，请参阅ListJobExecutionsForThing。

CLI syntax

```
aws iot list-job-executions-for-thing \
--thing-name <value> \
[--status <value>] \
[--max-results <value>] \
[--next-token <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"thingName": "string",
"status": "string",
"maxResults": "integer",
"nextToken": "string"
}
```

有关更多信息，请参阅list-job-executions-for-thing。

ListJobs

获取 Amazon Web Services 账户中的任务列表。

HTTPS request

```
GET /jobs?
status=status&targetSelection=targetSelection&thingGroupName=thingGroupName&thingGroupId=thingGroup
```

有关更多信息，请参阅ListJobs。

CLI syntax

```
aws iot list-jobs \
[--status <value>] \
```

```
[--target-selection <value>] \
[--max-results <value>] \
[--next-token <value>] \
[--thing-group-name <value>] \
[--thing-group-id <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"status": "string",
"targetSelection": "string",
"maxResults": "integer",
"nextToken": "string",
"thingGroupName": "string",
"thingGroupId": "string"
}
```

有关更多信息，请参阅list-jobs。

UpdateJob

更新指定任务的受支持字段。timeoutConfig 的更新值仅对新的正在进行的执行有效。当前正在进行的执行继续使用旧超时配置执行。

HTTPS request

```
PATCH /jobs/jobId
{
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
     },
  "maximumPerMinute": number
  },
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

有关更多信息，请参阅 UpdateJob。

CLI syntax

```
aws iot  update-job \
--job-id <value> \
[--description <value>] \
[--presigned-url-config <value>] \
[--job-executions-rollout-config <value>] \
[--abort-config <value>] \
[--timeout-config <value>] \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"description": "string",
"presignedUrlConfig": {
  "expiresInSec": number,
  "roleArn": "string"
},
"jobExecutionsRolloutConfig": {
  "exponentialRate": {
     "baseRatePerMinute": number,
     "incrementFactor": number,
     "rateIncreaseCriteria": {
        "numberOfNotifiedThings": number,
        "numberOfSucceededThings": number
     }
  },
  "maximumPerMinute": number
},
"abortConfig": {
  "criteriaList": [
     {
        "action": "string",
        "failureType": "string",
        "minNumberOfExecutedThings": number,
        "thresholdPercentage": number
     }
  ]
},
"timeoutConfig": {
  "inProgressTimeoutInMinutes": number
}
}
```

有关更多信息，请参阅update-job。

任务设备 MQTT 和 HTTPS API 操作以及数据类型

可通过 MQTT 和 HTTPS 协议使用以下命令。在数据层面上对执行任务的设备使用这些 API 操作。

任务设备 MQTT 和 HTTPS 数据类型

可使用以下数据类型通过 MQTT 和 HTTPS 协议与 Amazon IoT Jobs 服务进行通信。

JobExecution

包含有关任务执行的数据。以下示例显示该语法：

```
{
    "jobId" : "string",
    "thingName" : "string",
    "jobDocument" : "string",
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED",
    "statusDetails": {
        "string": "string"
    },
    "queuedAt" : "timestamp",
    "startedAt" : "timestamp",
    "lastUpdatedAt" : "timestamp",
    "versionNumber" : "number",
    "executionNumber": long
}
```

有关更多信息，请参阅 JobExecution 或 job-execution。

JobExecutionState

包含有关任务执行的状态的数据。以下示例显示该语法：

```
{
    "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED",
    "statusDetails": {
        "string": "string"
        ...
    }
    "versionNumber": "number"
}
```

有关更多信息，请参阅 JobExecutionState 或 job-execution-state。

JobExecutionSummary

包含有关任务执行的信息的子集。以下示例显示该语法：

```
{
    "jobId": "string",
    "queuedAt": timestamp,
    "startedAt": timestamp,
    "lastUpdatedAt": timestamp,
    "versionNumber": "number",
    "executionNumber": long
}
```

有关更多信息，请参阅 JobExecutionSummary 或 job-execution-summary。

请参阅以下部分了解有关 MQTT 和 HTTPS API 操作的更多信息：

- 任务设备 MQTT API (p. 712)
- 任务设备 HTTP API (p. 717)

任务设备 MQTT API

可以通过将 MQTT 消息发布到用于 Jobs 命令的保留主题 (p. 101)来发出任务设备命令。

您的设备端客户端必须订阅这些命令的响应消息主题。如果您使用 Amazon IoT 设备客户端，您的设备将自动订阅响应主题。这意味着消息代理将向发布命令消息的客户端发布响应消息主题，无论您的客户端是否订阅了该响应消息主题。这些响应消息不会通过消息代理，也无法由其它客户端或规则订阅。

订阅机群监控解决方案的任务和 jobExecution 事件主题时，首先启用任务和任务执行事件 (p. 1088)以接收云端的任何事件。Job 进度消息，这些消息通过消息代理处理并且可以由 Amazon IoT 规则使用，将发布为 任务事件 (p. 1099)。由于消息代理会发布响应消息，即使没有明确订阅响应消息也是如此，因此必须配置您的客户端以接收和标识其接收的消息。您的客户端还必须确认传入消息中的 *thingName* 在客户端对消息执行操作前应用于客户端的事物名称。

> **Note**
>
> Amazon IoT 发送的用于响应 MQTT Job API 命令消息的消息都会计入您的账户，无论您是否明确订阅了这些消息。

下面显示了 MQTT API 操作及其请求和响应语法。所有 MQTT API 操作都具有以下参数：

clientToken

可用于将请求和响应关联起来的可选客户端令牌。在此处输入任意值，它将反映在响应中。

timestamp

发送消息的时间（用从纪元开始的秒数表示）。

GetPendingJobExecutions

针对指定的事物，获取未处于最终状态的所有任务的列表。

要调用此 API，请在 $aws/things/*thingName*/jobs/get 上发布消息。

请求负载：

```
{ "clientToken": "string" }
```

消息代理将发布 $aws/things/*thingName*/jobs/get/accepted 和 $aws/things/*thingName*/jobs/get/rejected，即使您没有指定订阅它们。但是，为了让您的客户端接收消息，客户端必须侦听这些消息。有关更多信息，请参阅关于 Job API 消息的说明 (p. 712)。

响应负载：

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ],
"timestamp" : 1489096425069,
"clientToken" : "client-001"
}
```

其中，inProgressJobs 和 queuedJobs 返回状态为 IN_PROGRESS 或 QUEUED 的 JobExecutionSummary (p. 712) 对象的列表。

StartNextPendingJobExecution

获取并启动事物的下一个待处理任务执行（状态为 IN_PROGRESS 或 QUEUED）。

- 首先返回状态为 IN_PROGRESS 的所有任务执行。
- 按任务执行的创建顺序返回这些执行。
- 如果下一个待处理任务执行的状态为 QUEUED，则其状态将更改为 IN_PROGRESS，并且任务执行的状态详细信息将设定为指定内容。
- 如果下一个待处理任务执行已处于 IN_PROGRESS 状态，则不会更改其状态详细信息。
- 如果没有待处理的任务执行，则响应不包括 execution 字段。
- （可选）您可以通过为 stepTimeoutInMinutes 属性设置值来创建步骤计时器。如果您没有通过运行 UpdateJobExecution 更新此属性的值，任务执行将在步骤计时器到期时超时。

要调用此 API，请在 $aws/things/*thingName*/jobs/start-next 上发布消息。

请求负载：

```
{
"statusDetails": {
    "string": "job-execution-state"
    ...
},
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

statusDetails

描述任务执行状态的名称-值对的集合。如果未指定，则 statusDetails 保持不变。

stepTimeOutInMinutes

指定此设备完成该任务执行所具有的时间。如果任务执行状态未在此计时器过期或者重置计时器（通过调用 UpdateJobExecution，将状态设置为 IN_PROGRESS，并在字段 stepTimeoutInMinutes 中指定新超时值）之前设置为最终状态，则任务执行状态将设置为 TIMED_OUT。设置此超时不会影响在创建任务时（CreateJob，使用 timeoutConfig 字段）可能指定的任务执行超时。

消息代理将发布 $aws/things/*thingName*/jobs/start-next/accepted 和 $aws/things/*thingName*/jobs/start-next/rejected，即使您没有指定订阅它们。但是，为了让您的客户端接收消息，客户端必须侦听这些消息。有关更多信息，请参阅关于 Job API 消息的说明 (p. 712)。

响应负载：

```
{
"execution" : JobExecutionData,
"timestamp" : timestamp,
"clientToken" : "string"
}
```

其中，execution 是 JobExecution (p. 711) 对象。例如：

```
{
"execution" : {
    "jobId" : "022",
    "thingName" : "MyThing",
    "jobDocument" : "< contents of job document >",
    "status" : "IN_PROGRESS",
    "queuedAt" : 1489096123309,
    "lastUpdatedAt" : 1489096123309,
    "versionNumber" : 1,
    "executionNumber" : 1234567890
},
"clientToken" : "client-1",
"timestamp" : 1489088524284,
}
```

DescribeJobExecution

获取有关任务执行的详细信息。

您可以将 jobId 设置为 $next 以返回事物的下一个待处理任务执行（状态为 IN_PROGRESS 或 QUEUED）。

要调用此 API，请在 $aws/things/*thingName*/jobs/*jobId*/get 上发布消息。

请求负载：

```
{
"executionNumber": long,
"includeJobDocument": boolean,
"clientToken": "string"
}
```

thingName

与设备关联的事物的名称。

jobId

创建此任务时向其分配的唯一标识符。

或使用 $next 返回事物的下一个待处理任务执行（状态为 IN_PROGRESS 或 QUEUED）。在此情况下，首先返回状态为 IN_PROGRESS 的所有任务执行。按任务执行的创建顺序返回这些执行。

executionNumber

（可选）标识设备上的任务执行的数字。如果未指定，则返回最新的任务执行。

includeJobDocument

（可选）除非设置为 false，否则响应将包含任务文档。默认为 true。

消息代理将发布 $aws/things/*thingName*/jobs/*jobId*/get/accepted 和 $aws/things/*thingName*/jobs/*jobId*/get/rejected，即使您没有指定订阅它们。但是，为了让您的客户端接收消息，客户端必须侦听这些消息。有关更多信息，请参阅关于 Job API 消息的说明 (p. 712)。

响应负载：

```
{
"execution" : JobExecutionData,
"timestamp": "timestamp",
"clientToken": "string"
}
```

其中，execution 是 JobExecution (p. 711) 对象。

UpdateJobExecution

更新任务执行的状态。（可选）您可以通过为 stepTimeoutInMinutes 属性设置值创建步骤计时器。如果您没有通过再次运行 UpdateJobExecution 更新此属性的值，任务执行将在步骤计时器到期时超时。

要调用此 API，请在 $aws/things/*thingName*/jobs/*jobId*/update 上发布消息。

请求负载：

```
{
"status": "job-execution-state",
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"executionNumber": long,
"includeJobExecutionState": boolean,
```

```
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"clientToken": "string"
}
```

`status`

任务执行的新状态（IN_PROGRESS、FAILED、SUCCEEDED，或 REJECTED）。这必须在每次更新时指定。

`statusDetails`

描述任务执行状态的名称-值对的集合。如果未指定，则 statusDetails 保持不变。

`expectedVersion`

任务执行的预期当前版本。每次更新任务执行时，其版本将递增。如果存储在 Amazon IoT Jobs 服务中的任务执行版本不匹配，则更新将被拒绝并出现 VersionMismatch 错误，并且将返回包含当前任务执行状态数据的 ErrorResponse (p. 697)。（这样就不必执行单独的 DescribeJobExecution 请求以获取任务执行状态数据。）

`executionNumber`

（可选）标识设备上的任务执行的数字。如果未指定，则使用最新的任务执行。

`includeJobExecutionState`

（可选）在包含此参数且设置为 true 时，响应将包含 JobExecutionState 字段。默认为 false。

`includeJobDocument`

（可选）在包含此参数且设置为 true 时，响应将包含 JobDocument。默认为 false。

`stepTimeoutInMinutes`

指定此设备完成该任务执行所具有的时间。如果任务执行状态未在此计时器过期或者重置计时器（通过再次调用 UpdateJobExecution，将状态设置为 IN_PROGRESS，并在此字段中指定新超时值）之前设置为最终状态，则任务执行状态将设置为 TIMED_OUT。设置或重置此超时不会影响在创建任务时（使用 CreateJob 和 timeoutConfig）可能指定的任务执行超时。

消息代理将发布 $aws/things/*thingName*/jobs/*jobId*/update/accepted 和 $aws/things/*thingName*/jobs/*jobId*/update/rejected，即使您没有指定订阅它们。但是，为了让您的客户端接收消息，客户端必须侦听这些消息。有关更多信息，请参阅关于 Job API 消息的说明 (p. 712)。

响应负载：

```
{
"executionState": JobExecutionState,
"jobDocument": "string",
"timestamp": timestamp,
"clientToken": "string"
}
```

`executionState`

一个 JobExecutionState (p. 712) 对象。

`jobDocument`

一个任务文档 (p. 638)对象。

`timestamp`

发送消息的时间（用从纪元 开始的秒数表示）。

`clientToken`

用于将请求和响应关联起来的客户端令牌。

使用 MQTT 协议时，您还可以执行以下更新：

JobExecutionsChanged

在事物的待处理任务执行列表中添加或删除任务执行时发送。

使用 主题：

`$aws/things/thingName/jobs/notify`

消息负载：

```
{
"jobs" : {
    "JobExecutionState": [ JobExecutionSummary ... ]
         },
    "timestamp": timestamp
}
```

NextJobExecutionChanged

当存在对事物的待处理任务执行列表中的下一个任务执行的更改（使用 `jobId $next` 为 `DescribeJobExecution` 定义）时发送。当下一个任务的执行详细信息发生更改时不会发送此消息，而仅在将由 `DescribeJobExecution`（`jobId` 为 `$next`）返回的下一个任务发生更改时发送。考虑状态为 `QUEUED` 的任务执行 J1 和 J2。J1 是待处理任务执行列表中的下一个待处理任务执行。如果 J2 的状态更改为 `IN_PROGRESS`，而 J1 的状态保持不变，则将发送此通知，并且此通知包含 J2 的详细信息。

使用 主题：

`$aws/things/thingName/jobs/notify-next`

消息负载：

```
{
"execution" : JobExecution,
"timestamp": timestamp,
}
```

任务设备 HTTP API

设备可以使用端口443 上的 HTTP 签名版本 4 与 Amazon IoT Jobs 通信。这是由Amazon SDK 和 CLI 使用的方法。有关这些工具的更多信息，请参阅 Amazon CLI 命令引用：iot-jobs-data 或 Amazon SDK 和工具。

以下命令可用于执行任务的设备。有关将 API 操作与 MQTT 协议结合使用的信息，请参阅任务设备 MQTT API (p. 712)。

GetPendingJobExecutions

针对指定的事物，获取未处于最终状态的所有任务的列表。

HTTPS request

```
GET /things/thingName/jobs
```

响应:

```
{
"inProgressJobs" : [ JobExecutionSummary ... ],
"queuedJobs" : [ JobExecutionSummary ... ]
}
```

有关更多信息，请参阅GetPendingJobExecutions。

CLI syntax

```
aws iot-jobs-data get-pending-job-executions \
--thing-name <value>  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"thingName": "string"
}
```

有关更多信息，请参阅get-pending-job-executions。

StartNextPendingJobExecution

获取并启动事物的下一个待处理任务执行（状态为 IN_PROGRESS 或 QUEUED）。

- 首先返回状态为 IN_PROGRESS 的所有任务执行。
- 按任务执行的创建顺序返回这些执行。
- 如果下一个待处理任务执行的状态为 QUEUED，则其状态将更改为 IN_PROGRESS，并且任务执行的状态详细信息将设定为指定内容。
- 如果下一个待处理任务执行已处于 IN_PROGRESS 状态，其状态详细信息不会发生变化。
- 如果没有待处理的任务执行，则响应不包括 execution 字段。
- （可选）您可以通过为 stepTimeoutInMinutes 属性设置值来创建步骤计时器。如果您没有通过运行 UpdateJobExecution 更新此属性的值，任务执行将在步骤计时器到期时超时。

HTTPS request

以下示例显示请求语法：

```
PUT /things/thingName/jobs/$next
{
"statusDetails": {
    "string": "string"
    ...
},
"stepTimeoutInMinutes": long
}
```

有关更多信息，请参阅StartNextPendingJobExecution。

CLI syntax

摘要：

```
aws iot-jobs-data  start-next-pending-job-execution \
```

```
--thing-name <value> \
{--step-timeout-in-minutes <value>] \
[--status-details <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"thingName": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": long
}
```

有关更多信息，请参阅start-next-pending-job-execution。

DescribeJobExecution

获取有关任务执行的详细信息。

您可以将 jobId 设置为 $next 以返回事物的下一个待处理任务执行。任务的执行状态必须为 QUEUED 或 IN_PROGRESS。

HTTPS request

请求:

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

响应:

```
{
"execution" : JobExecution,
}
```

有关更多信息，请参阅DescribeJobExecution。

CLI syntax

摘要：

```
aws iot-jobs-data  describe-job-execution \
--job-id <value> \
--thing-name <value> \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string",
"thingName": "string",
"includeJobDocument": boolean,
"executionNumber": long
```

```
}
```

有关更多信息，请参阅describe-job-execution。

UpdateJobExecution

更新任务执行的状态。（可选）您可以通过为 stepTimeoutInMinutes 属性设置值来创建步骤计时器。如果您没有通过再次运行 UpdateJobExecution 更新此属性的值，任务执行将在步骤计时器到期时超时。

HTTPS request

请求:

```
POST /things/thingName/jobs/jobId
{
"status": "job-execution-state",
"statusDetails": {
    "string": "string"
    ...
},
"expectedVersion": "number",
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"stepTimeoutInMinutes": long,
"executionNumber": long
}
```

有关更多信息，请参阅UpdateJobExecution。

CLI syntax

摘要：

```
aws iot-jobs-data  update-job-execution \
--job-id <value> \
--thing-name <value> \
--status <value> \
[--status-details <value>] \
[--expected-version <value>] \
[--include-job-execution-state | --no-include-job-execution-state] \
[--include-job-document | --no-include-job-document] \
[--execution-number <value>]  \
[--cli-input-json <value>] \
[--step-timeout-in-minutes <value>] \
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{
"jobId": "string",
"thingName": "string",
"status": "string",
"statusDetails": {
"string": "string"
},
"stepTimeoutInMinutes": number,
"expectedVersion": long,
"includeJobExecutionState": boolean,
"includeJobDocument": boolean,
"executionNumber": long
}
```

有关更多信息，请参阅`update-job-execution`。

使用 Amazon IoT Jobs 保护用户和设备

要授权用户将 Amazon IoT Jobs 与其设备结合使用，必须使用 IAM 策略向他们授予权限。然后，必须使用 Amazon IoT Core 策略对设备进行授权，以安全地连接到 Amazon IoT，接收作业执行，并更新执行状态。

Amazon IoT Jobs 所需的策略类型

下表显示了进行授权所必须使用的不同类型的策略。有关要使用的所需策略的更多信息，请参阅 Authorization (p. 298)。

所需策略类型

使用案例	协议	身份验证	控制面板/数据层面	身份类型	所需策略类型
授权管理员、运营商或云服务安全地使用 Jobs	HTTPS	Amazon 签名版本 4 身份验证（端口 443）	控制面板和数据层面	Amazon Cognito 身份、IAM 或联合身份用户	IAM policy
授权您的 IoT 设备安全地使用 Jobs	MQTT/HTTPS	TCP 或 TLS 双向身份验证（端口 8883 或 443）	数据层面	X.509 证书	Amazon IoT Core 策略

要授权可以同时在控制面板和数据层面上执行的 Amazon IoT Jobs 操作，您必须使用 IAM 策略。身份必须已向 Amazon IoT 进行身份验证才能执行这些操作，必须为 Amazon Cognito 身份 (p. 287)或 IAM 用户、组和角色 (p. 286)。有关身份验证的更多信息，请参阅身份验证 (p. 264)。

现在，必须使用 Amazon IoT Core 策略在数据层面上对设备进行授权，才能安全地连接到设备网关。设备网关使设备能够安全地与 Amazon IoT 通信，接收任务执行，然后更新任务执行状态。通过使用安全的 MQTT (p. 77) 或 HTTPS (p. 91) 通信协议保护设备通信。这些协议使用 Amazon IoT 提供的 X.509 客户端证书 (p. 267)对设备连接进行身份验证。

下面显示如何对用户、云服务和设备授权，以便使用 Amazon IoT Jobs。有关控制面板和数据层面 API 操作的更多信息，请参阅Amazon IoT 任务 API 操作 (p. 697)。

主题

- 向用户和云服务授权以使用 Amazon IoT Jobs (p. 721)
- 授权设备在数据层面上安全地使用 Amazon IoT Jobs (p. 728)

向用户和云服务授权以使用 Amazon IoT Jobs

要向用户和云服务授权，您必须同时在控制面板和数据层面上使用 IAM 策略。这些策略必须与 HTTPS 协议一起使用，并且必须使用 Amazon 签名版本 4 身份验证（端口 443）对用户进行身份验证。

> Note
>
> 不得在控制面板上使用 Amazon IoT Core 策略。只有 IAM 策略用于向用户或云服务授权。有关使用所需策略类型的更多信息，请参阅 Amazon IoT Jobs 所需的策略类型 (p. 721)。

IAM 策略是包含策略声明的 JSON 文档。策略语句使用效果、操作 和资源 元素以指定资源、允许或拒绝的操作以及允许或拒绝操作的条件。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

Warning

我们建议您不要使用通配符权限，例如 IAM 策略或 Amazon IoT Core 策略中的 "Action": ["iot:*"]。使用通配符权限不是推荐的安全最佳实践。有关更多信息，请参阅Amazon IoT 策略过于宽容 (p. 904)。

控制面板上的 IAM 策略

在控制面板上，IAM 策略对操作使用 iot: 前缀，以便授权执行相应的任务 API 操作。例如，iot:CreateJob 策略操作授予用户使用 CreateJob API 的权限。

策略操作

下表显示了 IAM 策略操作和使用 API 操作的权限的列表。有关资源类型的信息，请参阅 Amazon IoT 定义的资源类型。有关 Amazon IoT 操作的更多信息，请参阅 Amazon IoT 定义的操作。

控制面板上的 IAM 策略操作

策略操作	API 操作	资源类型	描述
iot:AssociateTargetsWithJob	AssociateTargetsWithJob	• 任务 • thing • thinggroup	表示将组与连续任务关联的权限。每次请求关联目标时，都会检查 iot:AssociateTargetsWithJob 权限。
iot:CancelJob	CancelJob	任务	表示取消任务的权限。每次请求取消任务时，都会检查 iot:CancelJob 权限。
iot:CancelJobExecution	CancelJobExecution	• 任务 • thing	表示取消任务执行的权限。每次请求取消任务执行时，都会检查 iot: CancelJobExecution 权限。
iot:CreateJob	CreateJob	• 任务 • thing • thinggroup • jobtemplate	表示创建任务的权限。每次请求创建任务时，都会检查 iot: CreateJob 权限。
iot:CreateJobTemplate	CreateJobTemplate	• 任务 • jobtemplate	表示创建任务模板的权限。每次请求创建任务模板时，都会检查 iot: CreateJobTemplate 权限。
iot:DeleteJob	DeleteJob	任务	表示删除任务的权限。每次请求删除任务时，都会检查 iot: DeleteJob 权限。
iot:DeleteJobTemplate	DeleteJobTemplate	jobtemplate	表示删除任务模板的权限。每次请求删除任务模板时，都会检查 iot: CreateJobTemplate 权限。
iot:DeleteJobExecution	DeleteJobTemplate	• 任务 • thing	表示删除任务执行的权限。每次请求删除任务执行时，都会检查 iot: DeleteJobExecution 权限。
iot:DescribeJob	DescribeJob	任务	表示描述任务的权限。每次请求描述任务时，都会检查 iot: DescribeJob 权限。
iot:DescribeJobExecution	DescribeJobExecution	• 任务 • thing	表示描述任务执行的权限。每次请求描述任务执行时，都会检查 iot: DescribeJobExecution 权限。

策略操作	API 操作	资源类型	描述
iot:DescribeJobTemplate	DescribeJobTemplate	jobtemplate	表示描述任务模板的权限。每次请求描述任务模板时，都会检查 iot:DescribeJobTemplate 权限。
iot:DescribeManagedJobTemplate	DescribeManagedJobTemplate	jobtemplate	表示描述托管式任务模板的权限。每次请求描述托管式任务模板时，都会检查 iot:DescribeManagedJobTemplate 权限。
iot:GetJobDocument	GetJobDocument	任务	表示获取任务的任务文档的权限。每次请求获取任务文档时，都会检查 iot:GetJobDocument 权限。
iot:ListJobExecutionsForJob	ListJobExecutionsForJob	任务	表示列出任务的任务执行的权限。每次请求列出任务的任务执行时，都会检查 iot:ListJobExecutionsForJob 权限。
iot:ListJobExecutionsForThing	ListJobExecutionsForThing	thing	表示列出任务的任务执行的权限。每次请求列出事物的任务执行时，都会检查 iot:ListJobExecutionsForThing 权限。
iot:ListJobs	ListJobs	无	表示列出任务的权限。每次请求列出任务时，都会检查 iot:ListJobs 权限。
iot:ListJobTemplates	ListJobTemplates	无	表示列出任务模板的权限。每次请求列出任务模板时，都会检查 iot:ListJobTemplates 权限。
iot:ListManagedJobTemplates	ListManagedJobTemplates	无	表示列出托管式任务模板的权限。每次请求列出托管式任务模板时，都会检查 iot:ListManagedJobTemplates 权限。
iot:UpdateJob	UpdateJob	任务	表示更新任务的权限。每次请求更新任务时，都会检查 iot:UpdateJob 权限。
iot:TagResource	TagResource	• 任务 • jobtemplate • thing	授予标记特定资源的权限。
iot:UntagResource	UntagResource	• 任务 • jobtemplate • thing	授予取消标记特定资源的权限。

基本 IAM 策略示例

下面显示了 IAM 策略的示例，该策略授予用户对 IoT 事物和事物组执行以下操作的权限。

在此示例中：

- 将 *region* 替换为您的 Amazon Web Services 区域，例如 us-east-1。
- 将 *account-id* 替换为您的 Amazon Web Services 账户编号，例如 57EXAMPLE833。
- 将 *thing-group-name* 替换为目标任务所属 IoT 事物组的名称，例如 FirmwareUpdateGroup。
- 将 *thing-name* 替换为目标任务所属 IoT 事物的名称，例如 MyIoTThing。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob",
                ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:region:account-id:thinggroup/thing-group-name"
        },
        {
            "Action": [
                "iot:DescribeJob",
                "iot:CancelJob",
                "iot:DeleteJob",
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:region:account-id:job/*"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution",
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iot:region:account-id:thing/thing-name"
                "arn:aws:iot:region:account-id:job/*"
            ]
        }
    ]
}
```

适用于基于 IP 的授权的 IAM policy 示例

您可以限制主体 从特定 IP 地址对控制面板端点进行 API 调用。要指定可以允许的 IP 地址，请在 IAM policy 的条件元素中使用 aws:SourceIp 全局条件键。

使用此条件键还可以拒绝其它 Amazon Web Service代表您进行这些 API 调用，例如 Amazon CloudFormation。要允许访问这些服务，请将 aws:ViaAWSService 全局条件键与 aws:SourceIp 键一起使用。这样可以确保源 IP 地址访问限制仅适用于由主体直接发出的请求。有关更多信息，请参阅 Amazon：基于源 IP 拒绝访问 Amazon。

以下示例说明如何仅允许能够对控制面板端点进行 API 调用的特定 IP 地址。aws:ViaAWSService 键设置为 true，这允许其它服务代表您进行 API 调用。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
            "Effect": "Allow",
            "Action": [
                "iot:CreateJobTemplate",
                "iot:CreateJob"
             ],
            "Resource": ["*"],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "123.45.167.89"
                }
            },
            "Bool": {"aws:ViaAWSService": "true"}
```

```
        }
    ],
}
```

数据层面上的 IAM 策略

数据层面上的 IAM 策略使用 `iotjobsdata:` 前缀以授权用户可以执行的任务 API 操作。在数据层面上，可以通过使用 `iotjobsdata:DescribeJobExecution` 策略操作授予用户使用 DescribeJobExecution API 的权限。

> Warning
>
> 当设备的目标指向 Amazon IoT Jobs 时，不建议在数据层面上使用 IAM 策略。我们建议您在控制面板上使用 IAM 策略，供用户创建和管理任务。在数据层面上，要授权设备检索任务执行和更新执行状态，请使用 HTTPS 协议的 Amazon IoT Core 策略 (p. 729)。

基本 IAM 策略示例

必须授权的 API 操作通常由用户键入 CLI 命令来执行。下面显示了用户执行 `DescribeJobExecution` 操作的示例。

在此示例中：

- 将 *region* 替换为您的 Amazon Web Services 区域，例如 `us-east-1`。
- 将 *account-id* 替换为您的 Amazon Web Services 账户编号，例如 `57EXAMPLE833`。
- 将 *thing-name* 替换为目标任务所属 IoT 事物的名称，例如 `myRegisteredThing`。
- *job-id* 是使用 API 确定的目标任务的唯一标识符。

```
aws iot-jobs-data describe-job-execution \
    --endpoint-url "https://account-id.jobs.iot.region.amazonaws.com" \
    --job-id jobID --thing-name thing-name
```

下面显示了授权执行此操作的示例 IAM 策略：

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": ["iotjobsdata:DescribeJobExecution"],
        "Effect": "Allow",
        "Resource": "arn:aws:iot:region:account-id:thing/thing-name",
    }
   }
```

适用于基于 IP 的授权的 IAM policy 示例

您可以限制主体 从特定 IP 地址对数据面板端点进行 API 调用。要指定可以允许的 IP 地址，请在 IAM policy 的条件元素中使用 `aws:SourceIp` 全局条件键。

使用此条件键还可以拒绝其它 Amazon Web Service代表您进行这些 API 调用，例如 Amazon CloudFormation。要允许访问这些服务，请将 `aws:ViaAWSService` 全局条件键与 `aws:SourceIp` 条件键一起使用。这可确保 IP 地址访问限制仅适用于主体直接发出的请求。有关更多信息，请参阅 Amazon：基于源 IP 拒绝访问 Amazon。

以下示例说明如何仅允许能够对数据面板端点进行 API 调用的特定 IP 地址。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
            "Effect": "Allow",
            "Action": ["iotjobsdata:*"],
            "Resource": ["*"],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "123.45.167.89"
                }
            },
            "Bool": {"aws:ViaAWSService": "false"}
        }
    ],
}
```

以下示例说明如何限制特定 IP 地址或地址范围，使其无法对数据面板端点进行 API 调用。

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
            "Effect": "Deny",
            "Action": ["iotjobsdata:*"],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "123.45.167.89",
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            },
            "Resource": ["*"],
        }
    ],
}
```

控制面板和数据层面的 IAM 策略示例

如果用户同时在控制面板和数据层面上执行 API 操作，则控制面板策略操作必须使用 `iot:` 前缀，而数据层面策略操作必须使用 `iotjobsdata:` 前缀。

例如，`DescribeJobExecution` API 可以同时用于控制面板和数据层面中。在控制面板上，DescribeJobExecution API 用于描述任务执行。在数据层面上，DescribeJobExecution API 用于获取任务执行的详细信息。

以下 IAM 策略授予用户同时在控制面板和数据层面上使用 `DescribeJobExecution` API 的权限。

在此示例中：

- 将 *region* 替换为您的 Amazon Web Services 区域，例如 `us-east-1`。
- 将 *account-id* 替换为您的 Amazon Web Services 账户编号，例如 `57EXAMPLE833`。
- 将 *thing-name* 替换为目标任务所属 IoT 事物的名称，例如 `MyIoTThing`。

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

```
            "Action": ["iotjobsdata:DescribeJobExecution"],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:region:account-id:thing/thing-name"
        },
        {
            "Action": [
                "iot:DescribeJobExecution",
                "iot:CancelJobExecution",
                "iot:DeleteJobExecution",
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iot:region:account-id:thing/thing-name"
                "arn:aws:iot:region:account-id:job/*"
            ]
        }
    ]
}
```

授权对 IoT 资源进行标记

为了更好地控制用户可以创建、修改或使用的任务和任务模板，可以将标签附加到任务或任务模板。还可以通过将标签放在账单组中并向其附加标签，帮助您辨别所有权并分配和分摊成本。

当用户希望标记他们使用 Amazon Web Services Management Console或 Amazon CLI 创建的任务或任务模板时，您的 IAM 策略必须授予用户标记它们的权限。要授予权限，IAM 策略必须使用 `iot:TagResource` 操作。

有关标记实例的一般信息，请参阅给您的 Amazon IoT 资源加标签 (p. 259)。

IAM 策略示例

有关显示如何授予标记权限的示例，请考虑这样一位用户：该用户运行以下命令创建任务并将其标记到特定环境。

在此示例中：

- 将 *region* 替换为您的 Amazon Web Services 区域，例如 `us-east-1`。
- 将 *account-id* 替换为您的 Amazon Web Services 账户编号，例如 `57EXAMPLE833`。
- 将 *thing-name* 替换为目标任务所属 IoT 事物的名称，例如 `MyIoTThing`。

```
aws iot create-job
    --job-id test_job
    --targets "arn:aws:iot:region:account-id:thing/thingOne"
    --document-source "https://s3.amazonaws.com/my-s3-bucket/job-document.json"
    --description "test job description"
    --tags Key=environment,Value=beta
```

对于本示例，您必须使用以下 IAM 策略：

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Action": [ "iot:CreateJob", "iot:CreateJobTemplate", "iot:TagResource" ],
        "Effect": "Allow",
        "Resource": [
            "arn:aws:iot:aws-region:account-id:job/*",
            "arn:aws:iot:aws-region:account-id:jobtemplate/*"
```

```
        ]
    }
}
```

授权设备在数据层面上安全地使用 Amazon IoT Jobs

要授权设备在数据层面上与 Amazon IoT Jobs 安全交互，必须使用 Amazon IoT Core 策略。任务的 Amazon IoT Core 策略是包含策略语句的 JSON 文档。这些策略还使用效果、操作 和资源 元素，并遵循与 IAM 策略类似的约定。有关这些元素的更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

这些策略可以与 MQTT 和 HTTPS 协议一起使用，并且必须使用 TCP 或 TLS 双向身份验证来对设备进行身份验证。下面展示了如何在不同的通信协议中使用这些策略。

> Warning
>
> 我们建议您不要使用通配符权限，例如 IAM 策略或 Amazon IoT Core 策略中的 `"Action": ["iot:*"]`。使用通配符权限不是推荐的安全最佳实践。有关更多信息，请参阅Amazon IoT 策略过于宽容 (p. 904)。

MQTT 协议的 Amazon IoT Core 策略

MQTT 协议的 Amazon IoT Core 策略授予您使用任务设备 MQTT API 操作的权限。MQTT API 操作用于处理为任务命令保留的 MQTT 主题。有关这些 API 操作的更多信息，请参阅 任务设备 MQTT API (p. 712)。

MQTT 策略使用策略操作（如 `iot:Connect`、`iot:Publish`、`iot:Subscribe` 和 `iot:Receieve`）来处理任务主题。这些策略允许您连接到消息代理，订阅任务 MQTT 主题，以及在设备和云之间发送和接收 MQTT 消息。有关这些操作的更多信息，请参阅Amazon IoT Core 策略操作 (p. 300)。

有关 Amazon IoT Jobs 的主题的信息，请参阅任务主题 (p. 101)。

基本 MQTT 策略示例

下面的示例显示如何使用 `iot:Publish` 和 `iot:Subscribe` 发布和订阅任务和任务执行。

在此示例中：

- 将 *region* 替换为您的 Amazon Web Services 区域，例如 `us-east-1`。
- 将 *account-id* 替换为您的 Amazon Web Services 账户编号，例如 `57EXAMPLE833`。
- 将 *thing-name* 替换为目标任务所属 IoT 事物的名称，例如 `MyIoTThing`。

```
{
    "Statement": [
        {
           "Effect": "Allow",
           "Action": [
                "iot:Publish",
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/events/job/*",
                "arn:aws:iot:region:account-id:topic/$aws/events/jobExecution/*",
                "arn:aws:iot:region:account-id:topic/$aws/things/thing-name/jobs/*"
            ]
        }
    ],
    "Version": "2012-10-17"
}
```

HTTPS 协议的 Amazon IoT Core 策略

数据层面上的 Amazon IoT Core 策略还可以将 HTTPS 协议与 TLS 身份验证机制结合使用来向您的设备授权。在数据层面上，策略使用 `iotjobsdata:` 前缀以授权您的设备可以执行的任务 API 操作。例如，`iotjobsdata:DescribeJobExecution` 策略操作授予用户使用 DescribeJobExecution API 的权限。

> **Note**
>
> 数据层面策略操作必须使用 `iotjobsdata:` 前缀。在控制面板上，操作必须使用 `iot:` 前缀。有关使用控制面板和数据层面策略操作时的 IAM 策略示例，请参阅 控制面板和数据层面的 IAM 策略示例 (p. 726)。

策略操作

下表显示了授权设备使用 API 操作的 Amazon IoT Core 策略操作和权限的列表。有关可以在数据层面执行的 API 操作的列表，请参阅任务设备 HTTP API (p. 717)。

> **Note**
>
> 这些任务执行策略操作仅适用于 HTTP TLS 终端节点。如果您使用 MQTT 终端节点，则必须之前定义的 MQTT 策略操作。

数据层面上的 Amazon IoT Core 策略操作

策略操作	API 操作	资源类型	描述
`iotjobsdata:DescribeJobExecution`	DescribeJobExecution	• 任务 • thing	表示检索任务执行的权限。每次请求检索任务执行时，都会检查 `iotjobsdata:DescribeJobExecution` 权限。
`iotjobsdata:GetPendingJobExecutions`	GetPendingJobExecutions	thing	表示一个权限，用于为事物检索未处于最终状态的任务的列表。每次请求检索该列表时，都会检查 `iotjobsdata:GetPendingJobExecutions` 权限。
`iotjobsdata:StartNextPendingJobExecution`	StartNextPendingJobExecution	thing	表示一个权限，用于为事物获取和启动下一个待处理任务执行。也即，将状态为 QUEUED 的任务执行更新为状态 IN_PROGRESS。每次请求启动下一个待处理任务执行，都会检查 `iot:StartNextPendingJobExecution` 权限。
`iotjobsdata:UpdateJobExecution`	UpdateJobExecution	thing	表示更新任务执行的权限。每次请求更新任务执行的状态时，都会检查 `iot:UpdateJobExecution` 权限。

基本策略示例

下面显示了一个 Amazon IoT Core 策略示例，该策略授予用户针对任何资源对数据层面 API 操作执行操作的权限。您可以将策略范围限定为特定资源，例如 IoT 事物。在您的示例中：

- 将 *region* 替换为您的 Amazon Web Services 区域，例如 `us-east-1`。
- 将 *account-id* 替换为您的 Amazon Web Services 账户编号，例如 `57EXAMPLE833`。

- 将 *thing-name* 替换为 IoT 事物的名称，例如 MyIoTthing。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iotjobsdata:GetPendingJobExecutions",
                "iotjobsdata:StartNextPendingJobExecution",
                "iotjobsdata:DescribeJobExecution",
                "iotjobsdata:UpdateJobExecution"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:region:account-id:thing/thing-name"
        }
    ]
}
```

必须使用这些策略的一个示例情形是：IoT 设备使用 Amazon IoT Core 策略来访问其中一个 API 操作，例如下面的 DescribeJobExecution API 示例：

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument&namespaceId=namespaceId
 HTTP/1.1
```

任务限制

Amazon IoT Jobs 具有服务限额或限制，它们与您 Amazon Web Services 账户的服务资源或操作的最大数量相对应。

活动和并发任务限制

本节将帮助您了解有关活动和并发任务以及适用于这些任务的限制的更多信息。

活动任务和活动任务限制

当您使用 Amazon IoT 控制台或 CreateJob API 创建任务时，任务状态更改为 IN_PROGRESS。所有正在进行的任务都是活动任务 并计入活动任务限制。这包括正在推出新任务执行的任务，或者正在等待设备完成任务执行的任务。此限制同时适用于连续任务和快照任务。

并发任务和任务并发限制

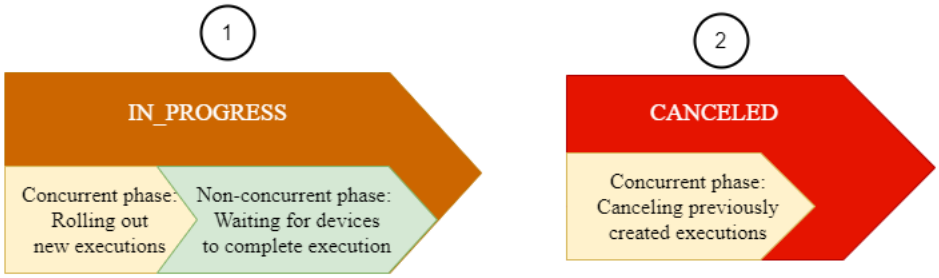
正在推出新的任务执行的正在进行的任务或正在取消之前创建的任务执行的任务为并发任务 并计入任务并发限制。由于 Amazon IoT Jobs 可以按每分钟 1000 台设备的速率快速推出和取消任务执行，因此每个任务都为 concurrent，并且仅在很短的时间内就会达到任务并发限制。在推出或取消任务执行之后，该任务将不再为并发的，而不计入任务并发限制。在等待设备完成任务执行的同时，您可以使用任务并发创建大量任务。

> **Note**
>
> 如果计划在维护时段内执行的具有可选计划配置和作业文档部署的任务达到选定值，startTime并且您达到了最大作业并发限制，则该计划作业将变为状态为CANCELED。

要确定任务是否为并发的，您可以通过 Amazon IoT 控制台或者通过使用 DescribeJob 或 ListJob API 来使用任务的 IsConcurrent 属性。此限制同时适用于连续任务和快照任务。

要查看您 Amazon Web Services 账户的活动任务和任务并发限制以及其它 Amazon IoT Jobs 配额并请求增加限制，请参阅《Amazon 一般参考》中的 Amazon IoT 设备管理终端节点和配额。

下图显示了任务并发如何适用于正在进行的任务和正在取消的任务。



Note

带有可选 SchedulingConfig 的新任务将保持初始状态 SCHEDULED，并在达到选定 startTime 时更新为 IN_PROGRESS。在带有可选 SchedulingConfig 的新任务达到选定的 startTime，并更新为 IN_PROGRESS 后，它将计入活动任务限制和任务并发限制。状态为 SCHEDULED 的任务将计入活动任务限制，但不会计入任务并发限制。

下表显示了适用于活跃和并发任务以及任务状态的并发和非并发阶段的限制。

活动和并发任务限制

作业状态	阶段	活动任务限制	任务并发限制
SCHEDULED	非并发阶段：Amazon IoT Jobs 会等到任务的计划 startTime，以开始向您的设备发送任务执行通知。此阶段的任务只计入活动任务限制，并且会将 IsConcurrent 属性设置为 false。	适用	不适用
IN_PROGRESS	并发阶段：Amazon IoT Jobs 接受用于创建任务的请求，并开始向您的设备推出任务执行通知。这个阶段的任务是并发的（由 IsConcurrent 属性设置为 true 来指示），并同时计入活动任务和任务并发限制。	适用	适用
	非并发阶段：Amazon IoT Jobs 等待设备报告其任务执行的结果。此阶段的任务只计入活动任务限制，并且会将 IsConcurrent 属性设置为 false。	适用	不适用
Canceled	并发阶段：Amazon IoT Jobs 接受用于取消任务的请求，并开始取消之前为您的设备创建的任务执行。此阶段的任务是并发的，并且会将 IsConcurrent 属性设置为 true。取消任务和任务执行后，该任务将不再为并发的，而不计入任务并发限制。	不适用	适用

Note

定期维护窗口的最大持续时间为 23 小时 50 分钟。

Amazon IoT 安全隧道

当设备部署在远程站点的受限防火墙后方时，您需要一种方法来获取对这些设备的访问权限，以便进行故障排除、配置更新和其它操作任务。使用安全隧道可通过由 Amazon IoT 管理的安全连接建立与远程设备的双向通信。安全隧道不要求更新现有的入站防火墙规则，因此，您可以在远程站点上保留防火墙规则提供的相同安全级别。

例如，位于几百英里之外的工厂的传感器设备在测量工厂温度时遇到问题。您可以使用安全隧道打开并快速启动到该传感器设备的会话。确定问题（例如，配置文件错误）后，可以重置文件并通过同一会话重新启动传感器设备。与更传统的故障排除（例如，向工厂派遣技术人员调查传感器设备）相比，安全隧道可减少事件响应和恢复时间以及运营成本。

什么是安全隧道？

使用安全隧道访问部署在远程站点端口限制防火墙后的设备。您可以使用 Amazon Web Services 云 从作为源设备的笔记本电脑或台式计算机连接到目标设备。源和目标通过使用在每台设备上运行的开源本地代理进行通信。本地代理使用防火墙允许的开放端口（通常为 443）与 Amazon Web Services 云 通信。通过隧道传输的数据已使用传输层安全性 (TLS) 进行加密。

主题

- 安全隧道概念 (p. 732)
- 安全隧道的工作原理 (p. 733)
- 安全隧道生命周期 (p. 733)

安全隧道概念

在与远程设备建立通信时，安全隧道会使用以下术语。有关安全隧道工作原理的信息，请参阅 安全隧道的工作原理 (p. 733)。

客户端访问令牌 (CAT)

创建新隧道时由安全隧道生成的一对令牌。源设备和目标设备使用 CAT 连接到安全隧道服务。在连接到隧道时只能使用 CAT 一次。要重新连接到隧道，请使用 RotateTunnelAccessTokenAPI 操作或 rotate-tunnel-access-tokenCLI 命令轮换客户端访问令牌。

客户端令牌

由客户端生成的唯一值，Amazon IoT 安全隧道可以使用它对同一隧道进行所有后续重试连接。此字段为可选项。如果未提供客户端令牌，则客户端访问令牌 (CAT) 对于同一隧道只能使用一次。使用同一 CAT 的后续连接尝试将遭到拒绝。有关使用客户端令牌的更多信息，请参阅中的本地代理引用实现 GitHub。

目标应用程序

在目标设备上运行的应用程序。例如，目标应用程序可以是使用安全隧道建立 SSH 会话的 SSH 守护程序。

目标设备

要访问的远程设备。

设备代理

连接到 Amazon IoT 设备网关并通过 MQTT 侦听新隧道通知的 IoT 应用程序。有关更多信息，请参阅IoT 代理代码段 (p. 762)：

本地代理

在源设备和目标设备上运行并在安全隧道和设备应用程序之间中继数据流的软件代理。本地代理可以在源模式或目标模式下运行。有关更多信息，请参阅本地代理 (p. 749)：

源设备

操作员用于启动与目标设备（通常是笔记本电脑或台式计算机）的会话的设备。

隧道

一个通过 Amazon IoT 的逻辑通道，可在源设备和目标设备之间实现双向通信。

安全隧道的工作原理

下面显示了安全隧道如何在源设备和目标设备之间建立连接。有关客户端访问令牌 (CAT) 等不同术语的信息，请参阅 安全隧道概念 (p. 732)。

1. 打开隧道

   要打开隧道来启动与远程目标设备的会话，可以使用、open-t Amazon CLIunnel 命令或 OpenTunnel API。Amazon Web Services Management Console
2. 下载客户端访问令牌对

   打开隧道后，您可以下载源和目标的客户端访问令牌 (CAT) 并将其保存在源设备上。在启动本地代理之前，必须先检索 CAT 并立即保存。
3. 在目标模式下启动本地代理

   已安装且正在目标设备上运行的 IoT 代理，会订阅 MQTT 预留主题 `$aws/things/thing-name/tunnels/notify` 并接收 CAT。此处的 *thing-name* 是您为目标创建的 Amazon IoT 事物的名称。有关更多信息，请参阅安全隧道主题 (p. 105)：

   然后，IoT 代理使用 CAT 在目标模式下启动本地代理，再在隧道的目标端设置连接。有关更多信息，请参阅IoT 代理代码段 (p. 762)：
4. 在源模式下启动本地代理

   隧道打开后，Amazon IoT Device Management 会提供可在源设备上下载的源的 CAT。您可以使用 CAT 在源模式下启动本地代理，然后连接隧道的源端。有关本地代理的更多信息，请参阅 本地代理 (p. 749)。
5. 打开 SSH 会话

   由于隧道的两端都已连线，您可以使用源端的本地代理来开启 SSH 会话。

有关如何使用 Amazon Web Services Management Console 打开隧道及开启 SSH 会话，请参阅 打开隧道并启动与远程设备的 SSH 会话 (p. 735)。

以下视频不仅会介绍安全隧道的工作原理，还会全程指导您设置与 Raspberry Pi 设备的 SSH 会话。

安全隧道生命周期

隧道的状态可能是 `OPEN` 或 `CLOSED`。与隧道的连接状态可能是 `CONNECTED` 或 `DISCONNECTED`。下面展示了不同隧道和连接状态的工作原理。

1. 打开隧道时，其状态为 OPEN。隧道的源和目标连接状态设置为 DISCONNECTED。
2. 当设备（源或目标）连接到隧道时，相应的连接状态会更改为 CONNECTED。
3. 当设备与隧道断开连接且隧道状态保持为 OPEN 时，相应的连接状态会更改回 DISCONNECTED。只要隧道状态保持为 OPEN，设备就可以重复连接隧道并断开与隧道的连接。

   Note

   在连接到隧道时只能使用客户端访问令牌 (CAT) 一次。要重新连接到隧道，请使用 RotateTunnelAccessTokenAPI 操作或 rotate-tunnel-access-tokenCLI 命令轮换客户端访问令牌。有关示例，请参阅 通过轮换客户端访问令牌解决 Amazon IoT 安全隧道连接问题 (p. 768)。
4. 在调用 CloseTunnel 时或者隧道状态保持 OPEN 的时间长于 MaxLifetimeTimeout 值时，隧道的状态会变为 CLOSED。您可以在调用 OpenTunnel 时配置 MaxLifetimeTimeout。如果您未指定值，则 MaxLifetimeTimeout 默认为 12 小时。

   Note

   隧道状态为 CLOSED 时，无法重新打开该隧道。
5. 当隧道可见时，您可以调用 DescribeTunnel 和 ListTunnels 来查看隧道元数据。隧道可能会在 Amazon IoT 控制台中显示至少三个小时，然后才会被删除。

Amazon IoT 安全隧道教程

Amazon IoT 安全隧道可帮助客户通过 Amazon IoT 管理的安全连接，与防火墙后面的远程设备建立双向通信。

要演示 Amazon IoT 安全隧道，请使用我们的 GitHub 上的 Amazon IoT 安全隧道演示。

以下教程会帮助您了解如何开始使用安全隧道。您将了解如何：

1. 使用快速设置和手动设置方法创建安全隧道以访问远程设备。
2. 使用手动设置方法时配置本地代理，并连接到隧道以访问目标设备。
3. 无需配置本地代理，即可从浏览器 SSH 到远程设备。
4. 将使用 Amazon CLI 或使用手动设置方法创建的隧道转换为使用快速设置方法。

本部分中的教程

本节中的教程重点介绍如何使用 Amazon Web Services Management Console 和 Amazon IoT API 参考创建隧道。在 Amazon IoT 控制台中，您可以从 Tunnels hub（隧道中心）页面或从您创建的事物的详细信息页面创建隧道。有关更多信息，请参阅Amazon IoT 控制台中的隧道创建方法 (p. 735)。

以下是本节中的教程：

- 打开隧道并使用基于浏览器的 SSH 访问远程设备 (p. 736)

  本教程介绍如何使用快速设置方法从 Tunnels hub（隧道中心）页面打开隧道。您还将学习如何使用基于浏览器的 SSH 和 Amazon IoT 控制台内的上下文命令行界面访问远程设备。

- 使用手动设置打开隧道并连接到远程设备 (p. 741)

  本教程介绍如何使用手动设置方法从 Tunnels hub（隧道中心）页面打开隧道。您还将学习如何从源设备中的终端配置和启动本地代理并连接到隧道。
- 为远程设备打开隧道并使用基于浏览器的 SSH (p. 746)

本教程介绍如何从您创建的事物的详细信息页面打开隧道。您将了解如何创建新隧道和如何使用现有隧道。现有隧道对应于为设备创建的最新开放隧道。您还可以使用基于浏览器的 SSH 来访问远程设备。

Amazon IoT 安全隧道教程

- 打开隧道并启动与远程设备的 SSH 会话 (p. 735)
- 为远程设备打开隧道并使用基于浏览器的 SSH (p. 746)

打开隧道并启动与远程设备的 SSH 会话

在这些教程中，您将学习如何远程访问防火墙后面的设备。您无法启动到设备的直接 SSH 会话，因为防火墙阻止所有入站流量。这些教程介绍如何打开隧道，然后使用该隧道启动到远程设备的 SSH 会话。

教程的先决条件

运行本教程的先决条件可能会有所不同，具体取决于您使用手动设置方法还是快速设置方法来打开隧道和访问远程设备。

Note

对于这两种设置方法，您必须都允许端口 443 上的出站流量。

- 有关快速设置方法教程的先决条件的信息，请参阅快速设置方法的先决条件 (p. 736)。
- 有关手动设置方法教程的先决条件的信息，请参阅手动设置方法的先决条件 (p. 741)。如果使用此设置方法，必须在源设备上配置本地代理。要下载本地代理源代码，请参阅 GitHub 上的本地代理参考实现。

隧道设置方法

在这些教程中，您将学习打开隧道和连接到远程设备的手动设置方法和快速设置方法。下表显示了这两种安装方法之间的差异。创建隧道后，您可以使用浏览器内命令行界面通过 SSH 连接到远程设备。如果放错了令牌或隧道断开连接，可以发送新的访问令牌以重新连接到隧道。

快速设置方法和手动设置方法

标准	快速设置	手动设置
创建隧道	使用默认的可编辑配置创建新隧道。要访问远程设备，只能使用 SSH 作为目标服务。	通过手动指定隧道配置来创建隧道。可以使用此方法通过 SSH 以外的服务连接到远程设备。
访问令牌	如果在创建隧道时指定了事物名称，则目标访问令牌将通过预留的 MQTT 主题自动传送到您的设备。您无需在源设备上下载或管理令牌。	您必须在源设备上手动下载或管理令牌。如果在创建隧道时指定了事物名称，则目标访问令牌将通过预留的 MQTT 主题自动传送到远程设备。
本地代理	系统会自动为您配置基于 Web 的本地代理，用于与设备进行交互。您不必手动配置本地代理。	您必须手动配置和启动本地代理。要配置本地代理，您可以使用 Amazon IoT 设备客户端或下载 GitHub 上的本地代理参考实现。

Amazon IoT 控制台中的隧道创建方法

本部分中的教程介绍如何使用 Amazon Web Services Management Console 和 OpenTunnel API 创建隧道。如果在创建隧道时配置目标，Amazon IoT 安全隧道会通过 MQTT 和预留的 MQTT 主题 ($aws/

things/RemoteDeviceA/tunnels/notify) 将目标客户端访问令牌传送到远程设备。收到 MQTT 消息后，远程设备上的 IoT 代理会以目标模式启动本地代理。有关更多信息，请参阅保留的主题 (p. 96)。

Note

如果要通过另一种方法将目标客户端访问令牌传送到远程设备，则可以省略目标配置。有关更多信息，请参阅配置远程设备和使用 IoT 代理 (p. 761)。

在 Amazon IoT 控制台中，您可以使用以下任一方法创建隧道。有关可帮助您了解如何使用这些方法创建隧道的教程的信息，请参阅本部分中的教程 (p. 734)。

- 隧道中心

  创建隧道时，您可以指定是使用快速设置方法还是手动设置方法来创建隧道，并提供可选的隧道配置详细信息。配置详细信息还包括目标设备的名称和要用于连接设备的服务。创建隧道后，可以在浏览器内进行 SSH，也可以在 Amazon IoT 控制台外部打开终端来访问您的远程设备。
- 事物详细信息页面

  创建隧道时，除了选择设置方法和提供任何可选的隧道配置详细信息外，还可以指定是使用最新的开放隧道，还是为设备创建新隧道。您无法编辑现有隧道的配置详细信息。您可以使用快速设置方法在浏览器中将访问令牌和 SSH 轮换到远程设备中。要使用此方法打开隧道，您必须已经在 Amazon IoT 注册表中创建了 IoT 事物（例如 RemoteDeviceA）。有关更多信息，请参阅在 Amazon IoT 注册表中注册设备。

本部分中的教程

- 打开隧道并使用基于浏览器的 SSH 访问远程设备 (p. 736)
- 使用手动设置打开隧道并连接到远程设备 (p. 741)

打开隧道并使用基于浏览器的 SSH 访问远程设备

您可以使用快速设置方法或手动设置方法来创建隧道。本教程介绍如何使用快速设置方法打开隧道，以及如何使用基于浏览器的 SSH 连接到远程设备。有关演示如何使用手动设置方法打开隧道的示例，请参阅使用手动设置打开隧道并连接到远程设备 (p. 741)。

使用快速设置方法时，您可以通过可编辑的默认配置创建新隧道。系统会为您配置基于 Web 的本地代理，并使用 MQTT 将访问令牌自动传递到远程目标设备。创建隧道后，您可以使用控制台中的命令行界面开始与远程设备进行交互。

使用快速设置方法，必须使用 SSH 作为目标服务才能访问远程设备。有关不同设置方法的信息，请参阅隧道设置方法 (p. 735)。

快速设置方法的先决条件

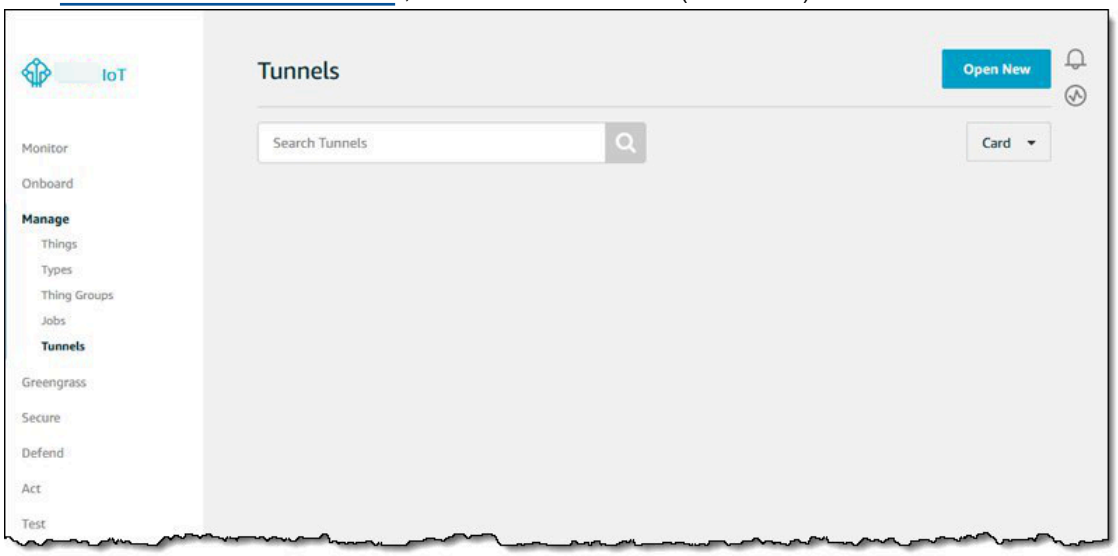
- 位于远程设备前面的防火墙必须允许端口 443 上的出站流量。您创建的隧道将使用此端口连接到远程设备。
- 您有一个 IoT 设备代理（请参阅 IoT 代理代码段 (p. 762)）在连接到 Amazon IoT 设备网关的远程设备上运行，并配置了 MQTT 主题订阅。有关更多信息，请参阅将设备连接到 Amazon IoT 设备网关。
- 您必须在远程设备上运行 SSH 守护进程。

打开隧道

您可以使用 Amazon Web Services Management Console、Amazon IoT API 参考或 Amazon CLI 打开安全隧道。您可以选择配置目标名称，但本教程不要求这样做。如果您配置了目标，安全隧道将使用 MQTT 自动将访问令牌传送到远程设备。有关更多信息，请参阅Amazon IoT 控制台中的隧道创建方法 (p. 735)。

使用控制台打开隧道

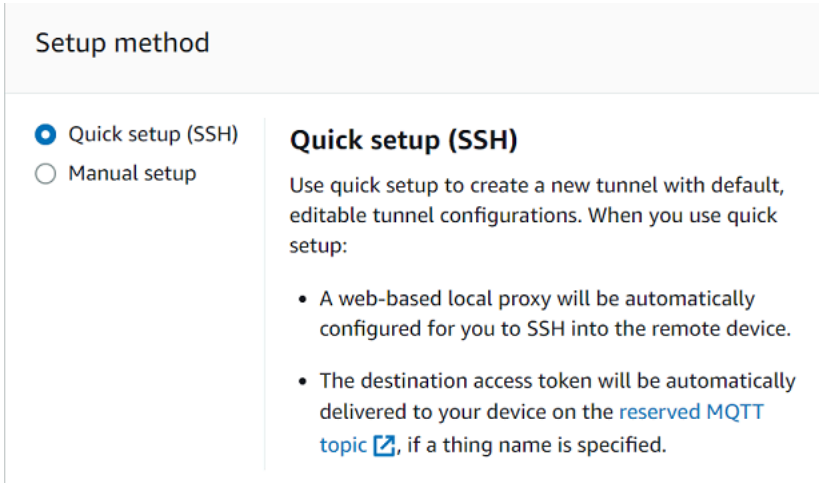
1. 转至 Amazon IoT 控制台的隧道中心，然后选择 Create tunnel（创建隧道）。



2. 对于本教程，请选择 Quick setup（快速设置）作为隧道创建方法，然后选择 Next（下一步）。

   Note

   如果您从所创建事物的详细信息页面创建安全隧道，则可以选择是创建新隧道还是使用现有隧道。有关更多信息，请参阅为远程设备打开隧道并使用基于浏览器的 SSH (p. 746)。



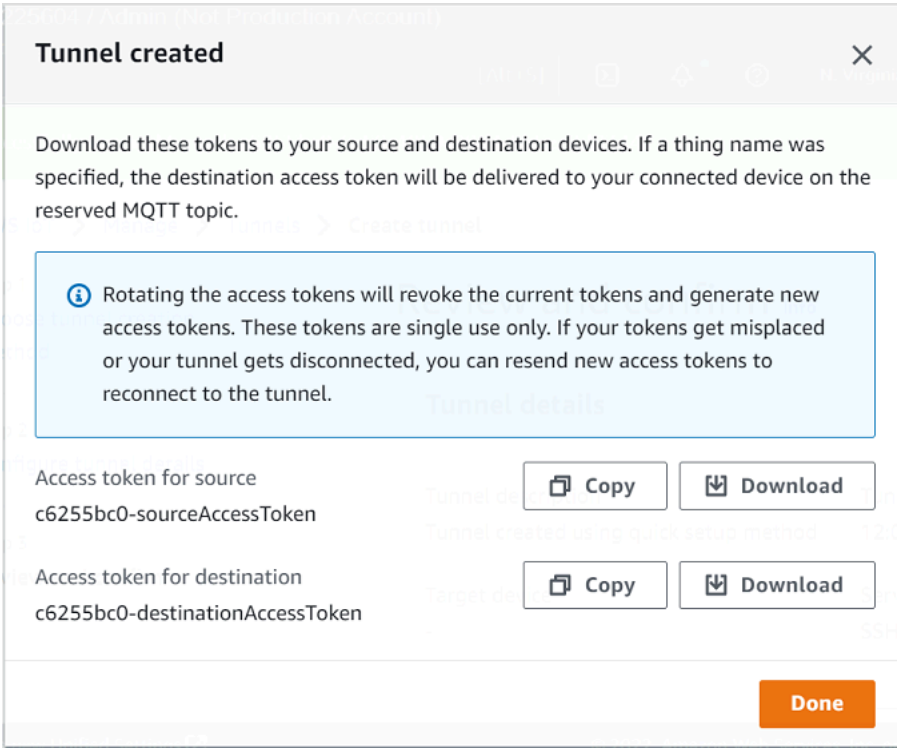
3. 查看并确认隧道配置详细信息。要创建隧道，请选择 Confirm and create（确认并创建）。如果要编辑这些详细信息，请选择 Previous（上一页）返回上一页，然后确认并创建隧道。

   Note

   使用快速设置时，无法编辑服务名称。您必须使用 SSH 作为 Service（服务）。

4. 要创建隧道，请选择 Done（完成）。

对于本教程，您不必下载源或目标访问令牌。这些令牌在连接到隧道时只能使用一次。如果您的隧道断开连接，您可以生成新令牌并将其发送到远程设备以重新连接到隧道。有关更多信息，请参阅重新发送隧道访问令牌 (p. 743)。



使用 API 打开隧道

要打开新隧道，您可以使用 OpenTunnel API 操作。

> **Note**
>
> 您只能从 Amazon IoT 控制台使用快速设置方法创建隧道。当您使用 Amazon IoT API 参考 API 或 Amazon CLI 时，将使用手动设置方法。您可以打开所创建的现有隧道，然后将隧道设置方法更改为使用快速设置。有关更多信息，请参阅打开现有隧道并使用基于浏览器的 SSH (p. 748)。

下面介绍一个有关如何运行此 API 操作的示例。或者，如果要指定事物名称和目标服务，请使用 `DestinationConfig` 参数。有关演示如何使用此参数的示例，请参阅为远程设备打开新隧道 (p. 747)。

```
aws iotsecuretunneling open-tunnel
```

运行此命令将创建新隧道，并为您提供源和目标访问令牌。

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

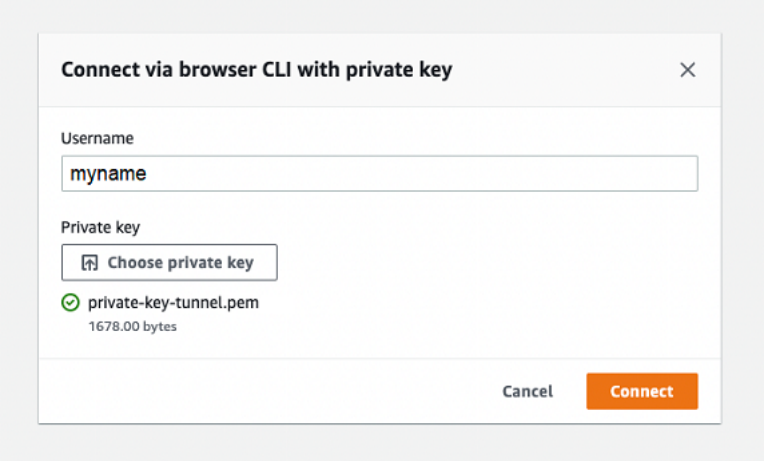
使用基于浏览器的 SSH

使用快速设置方法创建隧道，并且目标设备已连接到隧道后，您可以使用基于浏览器的 SSH 访问远程设备。使用基于浏览器的 SSH，您可以通过在控制台中的上下文命令行界面中输入命令来直接与远程设备通信。此功能使您可以更轻松地与远程设备进行交互，因为您不必在控制台外部打开终端，也不必配置本地代理。

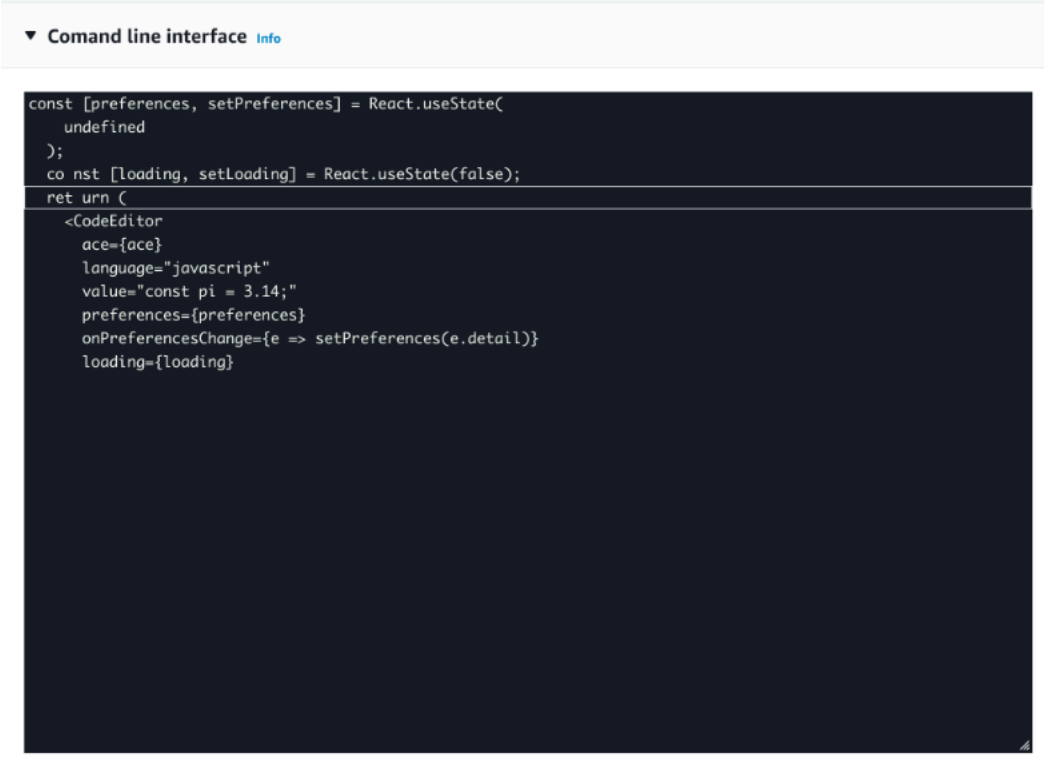
使用基于浏览器的 SSH

1. 转至 Amazon IoT 控制台的隧道中心，然后选择所创建的隧道以查看详细信息。
2. 展开 Secure Shell (SSH) [安全外壳 (SSH)] 部分，然后选择 Connect（连接）。
3. 选择是否要通过提供用户名和密码在 SSH 连接中进行身份验证，或者，为了进行更安全的身份验证，您可以使用设备的私钥。

   - 要使用您的用户名和密码进行连接，请选择 Use password（使用密码）。然后，您可以输入您的用户名和密码，并开始使用浏览器内 CLI。
   - 要使用目标设备的私钥进行连接，请选择 Use private key（使用私钥）。指定您的用户名并上传设备的私钥文件，然后选择 Connect（连接）以开始使用浏览器内 CLI。



通过 SSH 连接的身份验证后，您可以快速开始输入命令，并使用浏览器 CLI 与设备进行交互，因为已经为您配置了本地代理。



如果浏览器 CLI 在隧道持续时间后保持打开状态，则可能会超时，从而导致命令行界面断开连接。您可以复制隧道，并启动另一个会话以在控制台本身内与远程设备进行交互。

对使用基于浏览器的 SSH 时出现的问题进行故障排查

以下显示了如何对在使用基于浏览器的 SSH 时可能出现的一些问题进行故障排查。

- 您看到的是错误，而不是命令行界面

  您可能会看到错误，因为您的目标设备已断开连接。您可以选择 Generate new access tokens（生成新访问令牌）来生成新的访问令牌，并使用 MQTT 将令牌发送到您的远程设备。新令牌可用于重新连接到隧道。重新连接到隧道将清除历史记录，并刷新命令行会话。
- 使用私钥进行身份验证时，您会看到隧道已断开连接错误

  您可能会看到此错误是因为您的私钥可能未被目标设备接受。要解决此错误，请检查您上传的用于身份验证的私钥文件。如果您仍然看到错误，请检查您的设备日志。您也可以通过向远程设备发送新访问令牌来尝试重新连接到隧道。
- 使用会话时隧道已关闭

  如果您的隧道因保持打开超过指定的持续时间而关闭，您的命令行会话可能会断开连接。隧道一旦关闭就无法重新打开。要重新连接，必须打开另一条通往设备的隧道。

  您可以通过复制隧道来创建与已关闭隧道配置相同的新隧道。您可以从 Amazon IoT 控制台复制已关闭的隧道。要复制隧道，请选择已关闭的隧道以查看其详细信息，然后选择 Duplicate tunnel（复制隧道）。指定要使用的隧道持续时间，然后创建新隧道。

清理

- 关闭隧道

我们建议您在使用完隧道后将其关闭。如果隧道的打开时间超过指定的隧道持续时间，隧道也可能会关闭。隧道一旦关闭就无法重新打开。您仍然可以通过选择关闭的隧道，然后选择 Duplicate tunnel（复制隧道）来复制隧道。指定要使用的隧道持续时间，然后创建新隧道。

- 要从 Amazon IoT 控制台关闭单条隧道或多条隧道，请转到 Tunnels hub（隧道中心），选择要关闭的隧道，然后选择 Close tunnel（关闭隧道）。
- 要使用 Amazon IoT API 参考 API 关闭单条隧道或多条隧道，请使用 CloseTunnel API。

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- 删除隧道

  您可以从您的 Amazon Web Services 账户中永久删除隧道。

  Warning

  删除是永久性操作，无法撤消。

  - 要从 Amazon IoT 控制台删除单条隧道或多条隧道，请转到 Tunnels hub（隧道中心），选择要删除的隧道，然后选择 Delete tunnel（删除隧道）。
  - 要使用 Amazon IoT API 参考 API 删除单条隧道或多条隧道，请使用 CloseTunnel API。使用 API 时，将 `delete` 标志设置为 `true`。

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

使用手动设置打开隧道并连接到远程设备

当您打开隧道时，可以选择快速设置方法或手动设置方法来打开通往远程设备的隧道。本教程介绍如何使用手动设置方法打开隧道，以及如何配置和启动本地代理以连接到远程设备。

使用手动设置方法时，必须在创建隧道时手动指定隧道配置。创建隧道后，可以在浏览器内，也可以在 Amazon IoT 控制台外部打开终端来进行 SSH。本教程说明如何使用控制台外部的终端访问远程设备。您还将学习如何配置本地代理，然后连接到本地代理以与远程设备进行交互。要连接到本地代理，必须在创建隧道时下载源访问令牌。

通过此设置方法，可以使用 SSH 以外的服务（例如 FTP）连接到远程设备。有关不同设置方法的信息，请参阅隧道设置方法 (p. 735)。

手动设置方法的先决条件

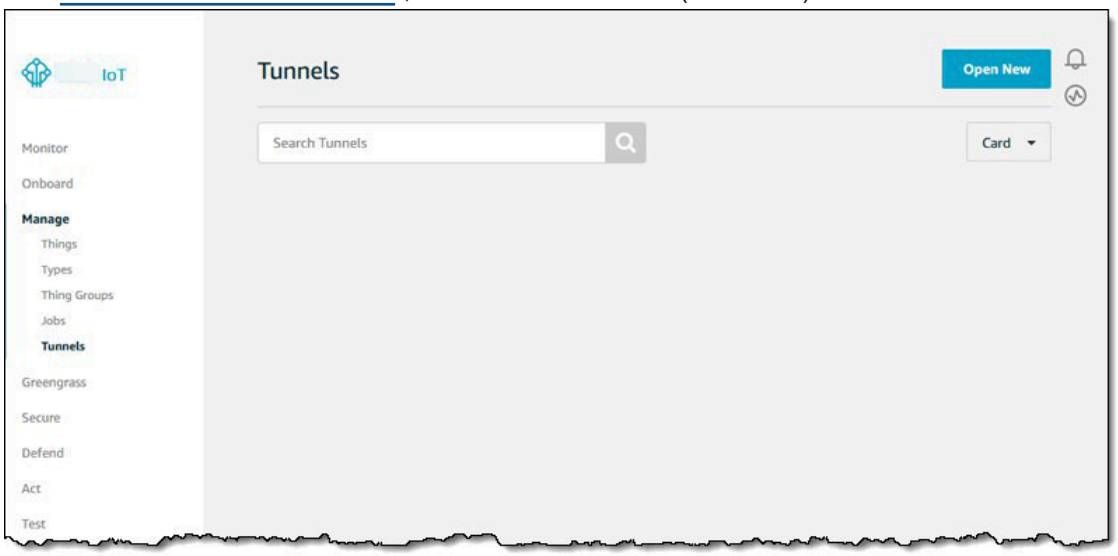
- 位于远程设备前面的防火墙必须允许端口 443 上的出站流量。您创建的隧道将使用此端口连接到远程设备。
- 您有一个 IoT 设备代理（请参阅 IoT 代理代码段 (p. 762)）在连接到 Amazon IoT 设备网关的远程设备上运行，并配置了 MQTT 主题订阅。有关更多信息，请参阅将设备连接到 Amazon IoT 设备网关。
- 您必须在远程设备上运行 SSH 守护进程。
- 您已经从 GitHub 下载了本地代理源代码，并为您选择的平台构建了它。在本教程中，我们将构建的本地代理可执行文件称为 `localproxy`。

打开隧道

您可以使用 Amazon Web Services Management Console、Amazon IoT API 参考或 Amazon CLI 打开安全隧道。您可以选择配置目标名称，但本教程不要求这样做。如果您配置了目标，安全隧道将使用 MQTT 自动将访问令牌传送到远程设备。有关更多信息，请参阅Amazon IoT 控制台中的隧道创建方法 (p. 735)。

在控制台中打开隧道

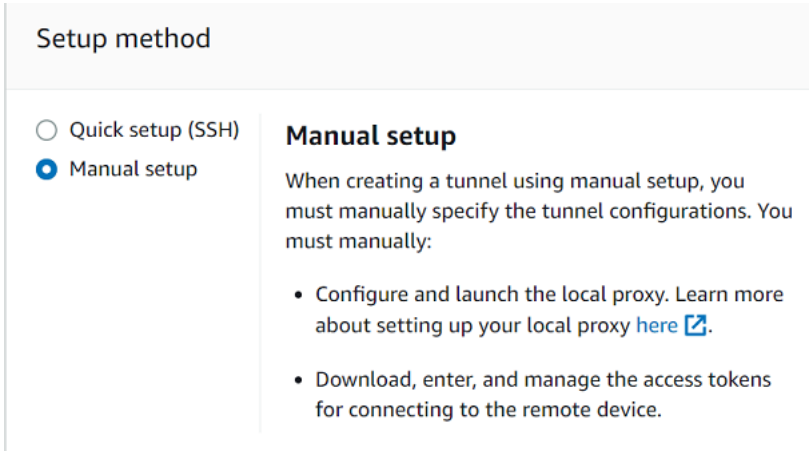
1. 转至 Amazon IoT 控制台的隧道中心，然后选择 Create tunnel（创建隧道）。



2. 在本教程中，选择 Manual setup（手动设置）作为隧道创建方法，然后选择 Next（下一步）。有关使用快速设置方法创建隧道的信息，请参阅打开隧道并使用基于浏览器的 SSH 访问远程设备 (p. 736)。

   Note

   如果您从事物详细信息页面创建安全隧道，则可以选择是创建新隧道还是使用现有隧道。有关更多信息，请参阅为远程设备打开隧道并使用基于浏览器的 SSH (p. 746)。



3. （可选）输入隧道的配置设置。您也可以跳过此步骤，继续下一步以创建隧道。

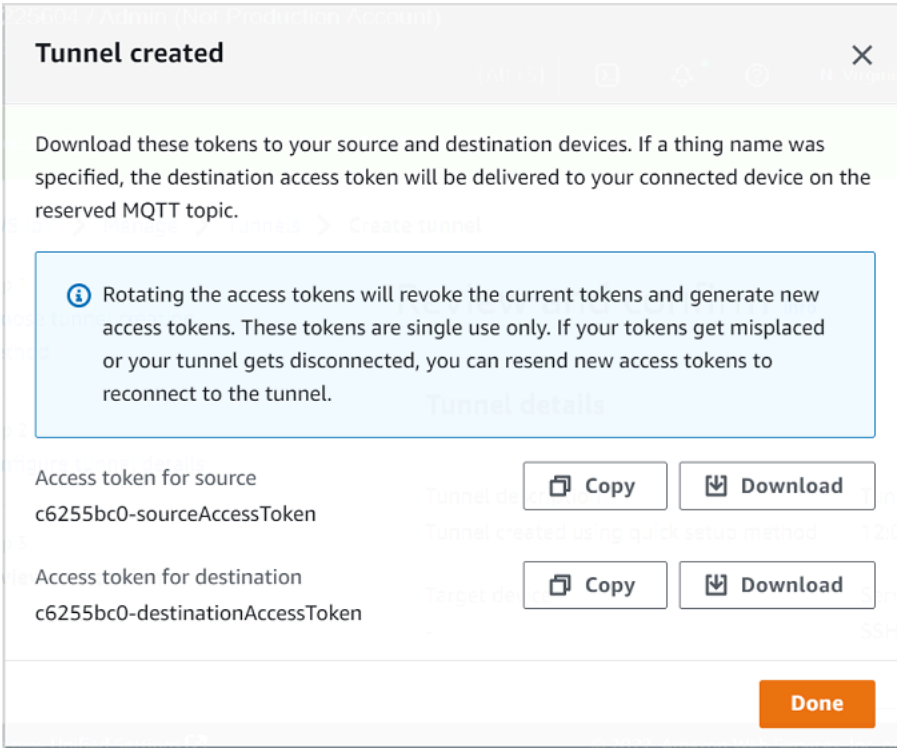
   请以键-值对的形式输入隧道描述、隧道超时持续时间和资源标记，以帮助您识别资源。对于本教程，您可以跳过目标配置。

   Note

   不会根据您保持隧道打开的持续时间向您收取费用。只有在创建新隧道时才会产生费用。有关定价的信息，请参阅 Amazon IoT Device Management 定价中的安全隧道。

4. 下载客户端访问令牌，然后选择 Done（完成）。选择 Done（完成）之后，令牌将无法下载。

   这些令牌在连接到隧道时只能使用一次。如果您放错了令牌或隧道断开连接，则可以生成新令牌并将其发送到远程设备以重新连接到隧道。



使用 API 打开隧道

要打开新隧道，您可以使用 OpenTunnel API 操作。您还可以使用 API 指定其他配置，例如隧道持续时间和目标配置。

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
```

运行此命令将创建新隧道，并为您提供源和目标访问令牌。

```
{
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

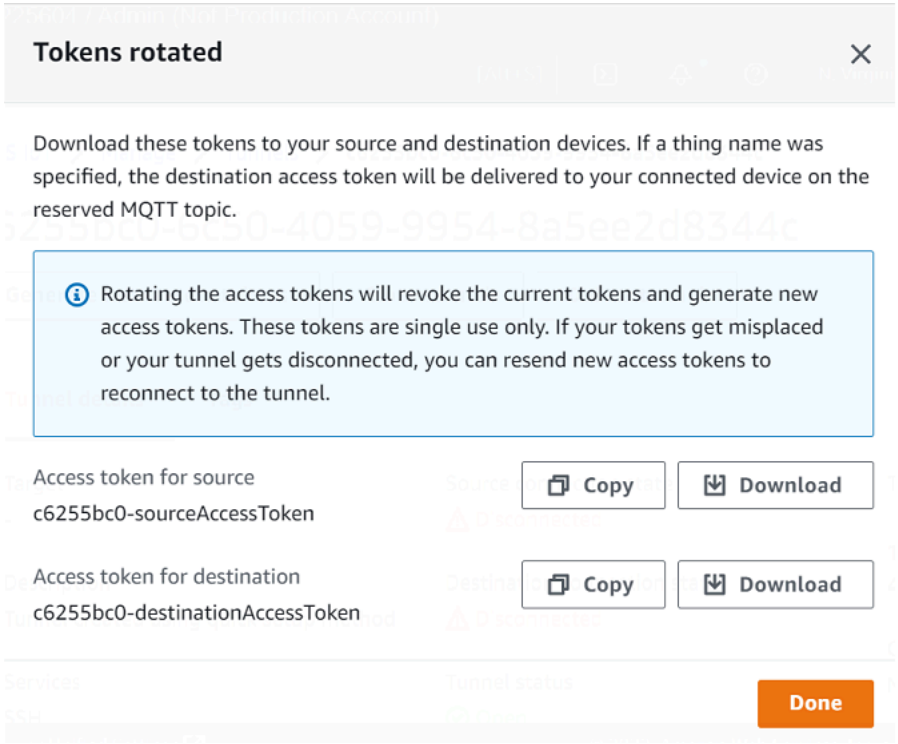
重新发送隧道访问令牌

创建隧道时获得的令牌只能用于连接到隧道一次。如果您放错了访问令牌或隧道断开连接，可以使用 MQTT 向远程设备重新发送新访问令牌，无需额外付费。Amazon IoT 安全隧道将撤消当前令牌，并返回新访问令牌以重新连接到隧道。

从控制台轮换令牌

1. 转至 Amazon IoT 控制台的隧道中心，然后选择所创建的隧道。
2. 在隧道详细信息页面中，选择 Generate new access tokens（生成新访问令牌），然后选择 Next（下一步）。

3. 为您的隧道下载新访问令牌，然后选择 Done（完成）。这些令牌只能使用一次。如果放错了令牌或隧道断开连接，可以发送新访问令牌。



使用 API 轮换访问令牌

要轮换隧道访问令牌，可以使用 RotateTunnelAccessToken API 操作撤消当前令牌，并返回新访问令牌以重新连接到隧道。例如，以下命令可轮换目标设备 *RemoteThing1* 的访问令牌。

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

运行此命令将生成以下示例中所示的新访问令牌。然后，如果设备代理设置正确，系统会使用 MQTT 将令牌传送到设备以连接到隧道。

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

有关说明如何以及何时轮换访问令牌的示例，请参阅通过轮换客户端访问令牌解决 Amazon IoT 安全隧道连接问题 (p. 768)。

配置和启动本地代理

要连接到远程设备，请在笔记本电脑上打开终端，然后配置并启动本地代理。本地代理借助 WebSocket 安全连接，通过使用安全隧道传输由源设备上运行的应用程序发送的数据。您可以从 GitHub 下载本地代理源。

配置本地代理后，复制源客户端访问令牌，然后在源模式下使用它来启动本地代理。以下显示了启动本地代理的示例命令。在以下命令中，本地代理配置用以侦听端口 5555 上的新连接。在此命令中：

- `-r` 指定 Amazon Web Services 区域，该区域必须与创建隧道的区域相同。
- `-s` 指定代理应连接到的端口。
- `-t` 指定客户端令牌文本。

```
./localproxy -r us-east-1 -s 5555 -t source-client-access-token
```

运行此命令将在源模式下启动本地代理。如果在运行此命令后收到以下错误，请设置 CA 路径。有关信息，请参阅 GitHub 上的安全隧道本地代理。

```
Could not perform SSL handshake with proxy server: certificate verify failed
```

以下显示了在 source 模式下运行本地代理的示例输出。

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://data.tunneling.iot.us-
east-1.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-east-1.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

启动 SSH 会话

打开另一个终端并使用以下命令通过连接到端口 5555 上的本地代理来启动新的 SSH 会话。

```
ssh username@localhost -p 5555
```

系统可能会提示您输入 SSH 会话的密码。完成 SSH 会话后，键入 **exit** 以关闭会话。

清理

- 关闭隧道

  我们建议您在使用完隧道后将其关闭。如果隧道的打开时间超过指定的隧道持续时间，隧道也可能会关闭。隧道一旦关闭就无法重新打开。您仍然可以通过打开关闭的隧道，然后选择 Duplicate tunnel（复制隧道）来复制隧道。指定要使用的隧道持续时间，然后创建新隧道。

  - 要从 Amazon IoT 控制台关闭单条隧道或多条隧道，请转到 Tunnels hub（隧道中心），选择要关闭的隧道，然后选择 Close tunnel（关闭隧道）。
  - 要使用 Amazon IoT API 参考 API 关闭单条隧道或多条隧道，请使用 CloseTunnel API 操作。

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

- 删除隧道

  您可以从您的 Amazon Web Services 账户中永久删除隧道。

  > Warning
  >
  > 删除是永久性操作，无法撤消。

  - 要从 Amazon IoT 控制台删除单条隧道或多条隧道，请转到 Tunnels hub（隧道中心），选择要删除的隧道，然后选择 Delete tunnel（删除隧道）。
  - 要使用 Amazon IoT API 参考 API 删除单条隧道或多条隧道，请使用 CloseTunnel API 操作。使用 API 时，将 `delete` 标志设置为 `true`。

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

为远程设备打开隧道并使用基于浏览器的 SSH

您可以从 Amazon IoT 控制台中的 Tunnels hub（隧道中心），或从您创建的 IoT 事物的详细信息页面创建隧道。从隧道中心创建隧道时，可以指定是使用快速设置还是手动设置来创建隧道。有关示例教程，请参阅打开隧道并启动与远程设备的 SSH 会话 (p. 735)。

从 Amazon IoT 控制台的事物详细信息页面创建隧道时，还可以指定是为该事物创建新隧道，还是打开现有隧道，如本教程中所示。如果选择现有隧道，可以访问为该设备创建的最新的已打开隧道。然后，可以使用终端中的命令行界面通过 SSH 连接到设备。

先决条件

- 位于远程设备前面的防火墙必须允许端口 443 上的出站流量。您创建的隧道将使用此端口连接到远程设备。
- 您已在 Amazon IoT 注册表中创建一个 IoT 事物（例如 `RemoteDevice1`）。此事物对应于云中您的远程设备表示。有关更多信息，请参阅在 Amazon IoT 注册表中注册设备。
- 您有一个 IoT 设备代理（请参阅 IoT 代理代码段 (p. 762)）在连接到 Amazon IoT 设备网关的远程设备上运行，并配置了 MQTT 主题订阅。有关更多信息，请参阅将设备连接到 Amazon IoT 设备网关。
- 您必须在远程设备上运行 SSH 守护进程。

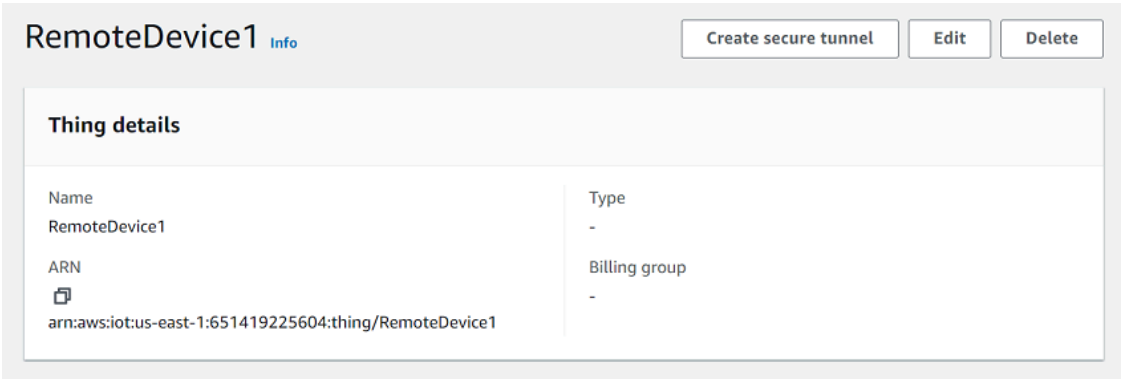
为远程设备打开新隧道

假设您要打开一条通往远程设备 RemoteDevice1 的隧道。首先，在 Amazon IoT 注册表中创建一个名为 RemoteDevice1 的 IoT 事物。然后，您可以使用 Amazon Web Services Management Console、Amazon IoT API 参考 API 或 Amazon CLI 创建一个隧道。

通过在创建隧道时配置目标，安全隧道服务可通过 MQTT 和保留的 MQTT 主题 (`$aws/things/RemoteDeviceA/tunnels/notify`) 将目标客户端访问令牌传递给远程设备。有关更多信息，请参阅Amazon IoT 控制台中的隧道创建方法 (p. 735)。

从控制台为远程设备创建隧道

1. 选择事物 RemoteDevice1 以查看其详细信息，然后选择 Create secure tunnel（创建安全隧道）。



2. 选择创建新隧道还是打开现有隧道。要创建新隧道，请选择 Create new tunnel（创建新隧道）。然后，您可以选择是使用手动设置方法还是快速设置方法来创建隧道。有关更多信息，请参阅 使用手动设置打开隧道并连接到远程设备 (p. 741) 和 打开隧道并使用基于浏览器的 SSH 访问远程设备 (p. 736)：

使用 API 为远程设备创建隧道

要打开新隧道，您可以使用 OpenTunnel API 操作。以下代码显示了运行此命令的示例。

```
aws iotsecuretunneling open-tunnel \
    --region us-east-1 \
    --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
    --cli-input-json file://input.json
```

以下显示了 input.json 文件的内容。您可以使用 destinationConfig 参数指定目标设备的名称（例如 *RemoteDevice1*）和要用来访问目标设备的服务（例如 *SSH*）。您还可以选择指定其他参数，例如隧道描述和标签。

input.json 的内容

```
{
   "description": "Tunnel to remote device1",
   "destinationConfig": {
      "services": [ "SSH" ],
      "thingName": "RemoteDevice1"
   }
}
```

运行此命令将创建新隧道，并为您提供源和目标访问令牌。

```
{
```

```
    "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
    "tunnelArn": "arn:aws:iot:us-
east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
    "sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
    "destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

打开现有隧道并使用基于浏览器的 SSH

假设您使用手动设置方法或使用 Amazon IoT API 参考 API 为远程设备 RemoteDevice1 创建了隧道。然后，您可以打开设备的现有隧道，并选择 Quick setup（快速设置）以使用基于浏览器的 SSH 功能。由于无法编辑现有隧道的配置，因此您无法使用手动设置方法。

要使用基于浏览器的 SSH 功能，您无需下载源访问令牌，也无需配置本地代理。系统将自动为您配置基于 Web 的本地代理，以便您可以开始与远程设备交互。

使用快速设置方法和基于浏览器的 SSH

1. 转到您创建的事物 RemoteDevice1 的详细信息页面，然后 Create secure tunnel（创建安全隧道）。
2. 选择 Use existing tunnel（使用现有隧道）打开您为远程设备创建的最新开放隧道。由于无法编辑隧道配置，因此您无法为隧道使用手动设置方法。要使用快速设置方法，请选择 Quick setup（快速设置）。
3. 继续查看并确认隧道配置详细信息并创建隧道。无法编辑隧道配置。

   创建隧道时，安全隧道将使用 RotateTunnelAccessToken API 操作来撤消原始访问令牌，并生成新访问令牌。如果您的远程设备使用 MQTT，这些令牌将通过其订阅的 MQTT 主题自动传送到远程设备。也可以选择将这些令牌手动下载到源设备。

创建隧道后，您可以使用基于浏览器的 SSH 和上下文命令行界面直接从控制台与远程设备进行交互。要使用此命令行界面，请为创建的事物选择隧道，然后在详细信息页面中展开 Command-line interface（命令行界面）部分。由于已经为您配置了本地代理，因此，您可以开始输入命令以快速开始访问远程设备 RemoteDevice1 并与之交互。

有关快速设置方法和使用基于浏览器的 SSH 的更多信息，请参阅打开隧道并使用基于浏览器的 SSH 访问远程设备 (p. 736)。

清理

- 关闭隧道

  我们建议您在使用完隧道后将其关闭。如果隧道的打开时间超过指定的隧道持续时间，隧道也可能会关闭。隧道一旦关闭就无法重新打开。您仍然可以通过打开关闭的隧道，然后选择 Duplicate tunnel（复制隧道）来复制隧道。指定要使用的隧道持续时间，然后创建新隧道。

  - 要从 Amazon IoT 控制台关闭单条隧道或多条隧道，请转到 Tunnels hub（隧道中心），选择要关闭的隧道，然后选择 Close tunnel（关闭隧道）。
  - 要使用 Amazon IoT API 参考 API 关闭单条隧道或多条隧道，请使用 CloseTunnel API 操作。

  ```
  aws iotsecuretunneling close-tunnel \
      --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
  ```

- 删除隧道

  您可以从您的 Amazon Web Services 账户中永久删除隧道。

  Warning

  删除是永久性操作，无法撤消。

- 要从 Amazon IoT 控制台删除单条隧道或多条隧道，请转到 Tunnels hub（隧道中心），选择要删除的隧道，然后选择 Delete tunnel（删除隧道）。
- 要使用 Amazon IoT API 参考 API 删除单条隧道或多条隧道，请使用 CloseTunnel API 操作。使用 API 时，将 `delete` 标志设置为 `true`。

```
aws iotsecuretunneling close-tunnel \
    --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
    --delete true
```

本地代理

本地代理借助安全连接，通过使用安全隧道传输由源设备上运行的 WebSocket 应用程序发送的数据。您可以从下载本地代理源代码GitHub。

本地代理可以在两种模式下运行：`source` 或 `destination`。在源模式下，本地代理与启动 TCP 连接的客户端应用程序在同一设备或网络上运行。在目标模式下，本地代理与目标应用程序一起在远程设备上运行。通过使用隧道多路复用功能，一条隧道一次最多可以支持三个数据流。对于每个数据流，安全隧道使用多个 TCP 连接，这将减小发生超时的几率。有关更多信息，请参阅多路复用数据流并在安全隧道中使用同步 TCP 连接 (p. 757)：

如何使用本地代理

您可以在源设备和目标设备上运行本地代理，将数据传输到安全隧道终端节点。如果您的设备位于使用 Web 代理的网络中，Web 代理可以在将连接转发到互联网之前拦截它们。在这种情况下，您需要配置本地代理，才能使用 Web 代理。有关更多信息，请参阅为使用 Web 代理的设备配置本地代理 (p. 752)：

本地代理工作流

以下步骤展示如何在源设备和目标设备上运行本地代理。

1. 将本地代理连接到安全隧道

   首先，本地代理必须与安全隧道建立连接。启动本地代理时，请使用以下参数：

   - `-r` 参数，用于指定打开隧道的 Amazon Web Services 区域。
   - `-t` 参数，用于传递从 `OpenTunnel` 返回的源或目标客户端访问令牌。

     Note

     不能同时连接两个使用相同客户端访问令牌值的本地代理。
2. 执行源操作或目标操作

   建立 WebSocket 连接后，本地代理会根据其配置执行源模式操作或目标模式操作。

   默认情况下，如果出现任何输入/输出 (I/O) 错误或 WebSocket 连接意外关闭，本地代理会尝试重新连接到安全隧道。这会导致 TCP 连接关闭。如果出现任何 TCP 套接字错误，本地代理将通过隧道发送一条消息，通知对方关闭其 TCP 连接。预设情况下，本地代理始终使用 SSL 通信。
3. 终止本地代理

   当您使用隧道后，可以安全地终止本地代理进程。我们建议您通过调用 `CloseTunnel` 显式关闭隧道。调用 `CloseTunnel` 后可能不会立即关闭活动隧道客户端。

有关如何使用 Amazon Web Services Management Console 打开隧道及开启 SSH 会话，请参阅 打开隧道并启动与远程设备的 SSH 会话 (p. 735)。

本地代理最佳实践

运行本地代理时，请遵循以下最佳实践：

- 避免使用本地代理参数 `-t` 传入访问令牌。我们建议您使用 `AWSIOT_TUNNEL_ACCESS_TOKEN` 环境变量设置本地代理的访问令牌。
- 在操作系统或环境中以最少权限运行本地代理可执行文件。
  - 避免以管理员身份在 Windows 上运行本地代理。
  - 避免以 root 身份在 Linux 和 macOS 上运行本地代理。
- 考虑在单独的主机、容器、沙盒、chroot jail 或虚拟化环境上运行本地代理。
- 使用相关安全标志构建本地代理，具体取决于您的工具链。
- 在具有多个网络接口的设备上，使用 `-b` 参数将 TCP 套接字绑定到用于与目标应用程序通信的网络接口。

命令和输出示例

下面显示了您运行的命令示例以及相应的输出。该示例说明如何在 `source` 和 `destination` 模式下配置本地代理。本地代理将 HTTPS 协议升级为 WebSockets以建立长期连接，然后开始通过连接将数据传输到安全隧道设备终端节点。

在运行这些命令之前：

您必须已打开隧道并获得源和目标的客户端访问令牌。您还必须已按照前面所述构建本地代理。要构建本地代理，请在 GitHub 存储库中打开本地代理源代码，然后按照构建和安装本地代理的说明进行操作。

> **Note**
>
> 示例中使用的以下命令使用 `verbosity` 标志来说明运行本地代理后先前描述的不同步骤概览。建议您仅将此标记用于测试。

在源模式下运行本地代理

以下命令显示如何在源模式下运行本地代理。

Linux/macOS

在 Linux 或 macOS 的终端中运行以下命令来配置和启动源上的本地代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -s 5555 -v 5 -r us-west-2
```

其中：

- `-s` 是源侦听端口，用于在源模式下启动本地代理。
- `-v` 表示输出的详细程度，可以是 0 到 6 之间的值。
- `-r` 是打开隧道的终端节点区域。

有关这些参数的更多信息，请参阅使用命令行参数设置的选项。

Windows

在 Windows 中，您可以配置本地代理，操作类似于 Linux 或 macOS 的配置，但是如何定义环境变量则与其它平台不同。在 cmd 窗口中运行以下命令来配置和启动源上的本地代理。

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
```

```
.\localproxy -s 5555 -v 5 -r us-west-2
```

其中：

- `-s` 是源侦听端口，用于在源模式下启动本地代理。
- `-v` 表示输出的详细程度，可以是 0 到 6 之间的值。
- `-r` 是打开隧道的终端节点区域。

有关这些参数的更多信息，请参阅使用命令行参数设置的选项。

以下显示了在 source 模式下运行本地代理的示例输出。

```
...
...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://data.tunneling.iot.us-
west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

在目标模式下运行本地代理

以下命令显示如何在目标模式下运行本地代理。

Linux/macOS

在 Linux 或 macOS 的终端中运行以下命令来配置和启动目标上的本地代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -d 22 -v 5 -r us-west-2
```

其中：

- `-d` 是在目标模式下启动本地代理的目标应用程序。

- `-v` 表示输出的详细程度，可以是 0 到 6 之间的值。
- `-r` 是打开隧道的终端节点区域。

有关这些参数的更多信息，请参阅使用命令行参数设置的选项。

Windows

在 Windows 中，您可以配置本地代理，操作类似于 Linux 或 macOS 的配置，但是如何定义环境变量则与其它平台不同。在 cmd 窗口中运行以下命令来配置和启动目标上的本地代理。

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -d 22 -v 5 -r us-west-2
```

其中：

- `-d` 是在目标模式下启动本地代理的目标应用程序。
- `-v` 表示输出的详细程度，可以是 0 到 6 之间的值。
- `-r` 是打开隧道的终端节点区域。

有关这些参数的更多信息，请参阅使用命令行参数设置的选项。

以下显示了在 `destination` 模式下运行本地代理的示例输出。

```
...
...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://data.tunneling.iot.us-
west-2.amazonaws.com:443
Resolved proxy  server IP: 10.10.0.11
Connected successfully with proxy server
Performing SSL handshake with proxy server
Successfully completed SSL handshake with proxy server
HTTP/1.1 101 Switching Protocols

...

Connection: upgrade
channel-id: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
upgrade: websocket

...

Web socket session ID: 01234567890abc23-00001234-0005678a-b1234c5de677a001-2bc3d456
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Setting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

为使用 Web 代理的设备配置本地代理

您可以在 Amazon IoT 设备上使用本地代理与 Amazon IoT 安全隧道 API 通信。本地代理借助安全连接，通过使用安全隧道传输设备应用程序发送的数据。 WebSocket 本地代理可以以 `source` 或者 `destination`

模式工作。在 source 模式中，它在启动 TCP 连接的同一设备或网络上运行。在 destination 模式下，本地代理与目标应用程序一起在远程设备上运行。有关更多信息，请参阅本地代理 (p. 749)：

本地代理需要直接连接到互联网才能使用 Amazon IoT 安全隧道。对于具有安全隧道的长期 TCP 连接，本地代理服务器会升级 HTTPS 请求，以建立 WebSockets 与一个安全隧道设备连接终端节点的连接。

如果您的设备位于使用 Web 代理的网络中，Web 代理可以在将连接转发到互联网之前将其拦截。要建立与安全隧道设备连接终端节点的长期连接，请将本地代理配置为使用 Web 代理，如 Websocket 规范中所述。

> Note
>
> Amazon IoT Device Client (p. 1285) 不支持使用 Web 代理的设备。要使用 Web 代理，您需要使用本地代理并将其配置为使用 Web 代理，如下所述。

以下步骤显示了本地代理如何与 Web 代理协同工作。

1. 本地代理发送一个 HTTP CONNECT 请求发送到包含安全隧道服务远程地址的 Web 代理服务，同时发布的还有 Web 代理身份验证信息。
2. 然后，Web 代理将创建到远程安全隧道终端节点的长期连接。
3. TCP 连接已建立，并且本地代理现在将在源模式和目标模式下工作以进行数据传输。

要完成本流程，请执行以下步骤。

- 构建本地代理 (p. 753)
- 配置 Web 代理 (p. 753)
- 配置和启动本地代理 (p. 754)

构建本地代理

在 GitHub 存储库中打开本地代理源代码，然后按照构建和安装本地代理的说明进行操作。

配置 Web 代理

本地代理依赖于 HTTP 隧道机制，如 HTTP/1.1 规范中所述。为了符合规范，您的 Web 代理必须允许设备使用 CONNECT 方法。

如何配置 Web 代理取决于您使用的 Web 代理和 Web 代理版本。要确保正确配置 Web 代理，请检查 Web 代理的文档。

要配置 Web 代理，请首先确定 Web 代理 URL 并确认 Web 代理是否支持 HTTP 隧道。稍后在配置和启动本地代理时，将使用 Web 代理 URL。

1. 识别您的 Web 代理 URL

   您的 Web 代理 URL 采用以下格式。

   ```
   protocol://web_proxy_host_domain:web_proxy_port
   ```

   Amazon IoT 安全隧道仅支持 Web 代理的基本身份验证。要使用基本身份验证，您必须指定 **username** 和 **password** 作为 Web 代理 URL 的一部分。Web 代理 URL 采用以下格式。

   ```
   protocol://username:password@web_proxy_host_domain:web_proxy_port
   ```

   - *##*可以是 http 或者 https。建议使用 https。
   - *web_proxy_host_domain* 是 Web 代理的 IP 地址或解析为 Web 代理 IP 地址的 DNS 名称。
   - *web_proxy_port* 是 Web 代理正在侦听的端口。

- Web 代理使用此 **username** 和 **password** 来验证请求。

2. 测试您的 Web 代理 URL

   要确认您的 Web 代理是否支持 TCP 隧道，请使用 curl 命令，并确保您获得 2xx 或 3xx 响应。

   例如，如果您的 Web 代理 URL 是 https://server.com:1235，请使用 proxy-insecure 标记与 curl 命令，因为 Web 代理可能依赖于自签名证书。

```
export HTTPS_PROXY=https://server.com:1235
curl -I https://aws.amazon.com --proxy-insecure
```

   如果您的 Web 代理 URL 具有 http 端口（例如，http://server.com:1234），您不一定要使用 proxy-insecure 标记。

```
export HTTPS_PROXY=http://server.com:1234
curl -I https://aws.amazon.com
```

配置和启动本地代理

要配置本地代理以使用 Web 代理，您必须以 DNS 域名或 Web 代理所用的 IP 地址和端口号码来配置 HTTPS_PROXY 环境变量。

配置了本地代理后，您可以使用本地代理，如本自述文件文档所述。

> Note
>
> 环境变量声明区分大小写。我们建议您使用全部大写或全部小写字母定义每个变量一次。以下示例显示了全部使用大写字母的环境变量名称。如果同时使用大写字母和小写字母指定相同的变量，则使用小写字母指定的变量优先。

以下命令说明如何将目标上运行的本地代理配置为使用 Web 代理并启动本地代理。

- AWSIOT_TUNNEL_ACCESS_TOKEN：此变量保存目标的客户端访问令牌 (CAT)。
- HTTPS_PROXY：此变量保存用于配置本地代理的 Web 代理 URL 或 IP 地址。

以下示例中显示的命令取决于您使用的操作系统以及 Web 代理侦听的是 HTTP 端口还是 HTTPS 端口。

Web 代理侦听 HTTP 端口

如果您的 Web 代理正在侦听 HTTP 端口，您可以为 HTTPS_PROXY 变量提供 Web 代理 URL 或 IP 地址。

Linux/macOS

在 Linux 或 macOS 中，在终端中运行以下命令，以配置和启动目标上的本地代理，使用侦听 HTTP 端口的 Web 代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22
```

如果您必须使用代理进行身份验证，则必须指定 **username** 和 **password** 作为 HTTPS_PROXY 变量的一部分。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
```

```
./localproxy -r us-east-1 -d 22
```

Windows

在 Windows 中，您可以配置本地代理，操作类似于 Linux 或 macOS 的配置，但是如何定义环境变量则与其它平台不同。在 cmd 窗口中运行以下命令以配置和启动目标上的本地代理，以使用侦听 HTTP 端口的 Web 代理。

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22
```

如果您必须使用代理进行身份验证，则必须指定 **username** 和 **password** 作为 HTTPS_PROXY 变量的一部分。

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22
```

侦听 HTTPS 端口的 Web 代理

如果您的 Web 代理正在侦听 HTTPS 端口，请运行以下命令。

Note

如果您正在为 Web 代理使用自签名证书，或者如果您在没有本机 OpenSSL 支持和默认配置的操作系统上运行本地代理，则必须按 GitHub 库中证书设置部分所述设置您的 Web 代理证书。

以下命令看起来与您为 HTTP 代理配置 Web 代理的方式相似，但您还需要指定您安装的证书文件的路径，如前所述。

Linux/macOS

在 Linux 或 macOS 中，在终端中运行以下命令，以配置目标上运行的本地代理，使其使用侦听 HTTPS 端口的 Web 代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

如果您必须使用代理进行身份验证，则必须指定 **username** 和 **password** 作为 HTTPS_PROXY 变量的一部分。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http://username:password@proxy.example.com:1234
./localproxy -r us-east-1 -d 22 -c /path/to/certs
```

Windows

在 Windows 中，在 cmd 窗口中运行以下命令以配置和启动目标上运行的本地代理，以使用侦听 HTTP 端口的 Web 代理。

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://proxy.example.com:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

如果您必须使用代理进行身份验证，则必须指定 **username** 和 **password** 作为 HTTPS_PROXY 变量的一部分。

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
set HTTPS_PROXY=http://username:password@10.15.20.25:1234
.\localproxy -r us-east-1 -d 22 -c \path\to\certs
```

命令和输出示例

下面显示了在 Linux 操作系统上运行的命令示例以及相应的输出。该示例显示了正在侦听 HTTP 端口的 Web 代理，以及如何在 source 和 destination 模式下使用 Web 代理配置本地代理。在运行这些命令之前，您必须已打开隧道并获得源和目标的客户端访问令牌。您还必须已构建本地代理并按照前面所述配置 Web 代理。

以下是启动本地代理后的步骤概览。本地代理：

- 标识 Web 代理 URL，以便它可以使用 URL 连接到代理服务器。
- 建立与 Web 代理的 TCP 连接。
- 发送 HTTP CONNECT 请求到 Web 代理，并等待 HTTP/1.1 200 响应，这表示连接已建立。
- 将 HTTPS 协议升级为 WebSockets 以建立长期连接。
- 开始通过与安全隧道设备终端节点的连接进行数据传输。

> **Note**
>
> 示例中使用的以下命令使用 verbosity 标志来说明运行本地代理后先前描述的不同步骤概览。建议您仅将此标记用于测试。

在源模式下运行本地代理

以下命令显示如何在源模式下运行本地代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -s 5555 -v 5 -r us-west-2
```

以下显示了在 source 模式下运行本地代理的示例输出。

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via the
 proxy.

...

Starting proxy in source mode
Attempting to establish web socket connection with endpoint wss://data.tunneling.iot.us-
west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.11
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 0a109afffee745f5-00001341-000b8138-cc6c878d80e8adb0-f186064b
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
```

```
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
Resolved bind IP: 127.0.0.1
Listening for new connection on port 5555
```

在目标模式下运行本地代理

以下命令显示如何在目标模式下运行本地代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
export HTTPS_PROXY=http:username:password@10.15.10.25:1234
./localproxy -d 22 -v 5 -r us-west-2
```

以下显示了在 destination 模式下运行本地代理的示例输出。

```
...

Parsed basic auth credentials for the URL
Found Web proxy information in the environment variables, will use it to connect via the
 proxy.

...

Starting proxy in destination mode
Attempting to establish web socket connection with endpoint wss://data.tunneling.iot.us-
west-2.amazonaws.com:443
Resolved Web proxy IP: 10.10.0.1
Connected successfully with Web Proxy
Successfully sent HTTP CONNECT to the Web proxy
Full response from the Web proxy:
HTTP/1.1 200 Connection established
TCP tunnel established successfully
Connected successfully with proxy server
Successfully completed SSL handshake with proxy server
Web socket session ID: 06717bfffed3fd05-00001355-000b8315-da3109a85da804dd-24c3d10d
Web socket subprotocol selected: aws.iot.securetunneling-2.0
Successfully established websocket connection with proxy server: wss://
data.tunneling.iot.us-west-2.amazonaws.com:443
Seting up web socket pings for every 5000 milliseconds
Scheduled next read:

...

Starting web socket read loop continue reading...
```

多路复用数据流并在安全隧道中使用同步 TCP 连接

您可以借助安全隧道多路复用功能，为每条隧道使用多个数据流。利用多路复用功能，您可以使用多个数据流对设备进行故障排除。您还可以通过取消构建、部署和启动多个本地代理或向同一设备打开多个隧道来减少操作负载。例如，对于需要发送多个 HTTP 和 SSH 数据流的 Web 浏览器，可以使用多路复用。

对于每个数据流，Amazon IoT 安全隧道支持同步 TCP 连接。使用同步连接可减小客户端发出多个请求时发生超时的几率。例如，它可以减少远程访问目标设备的本地 Web 服务器时的加载时间。

以下各节详细介绍了多路复用和使用同步 TCP 连接及其不同的使用案例。

主题

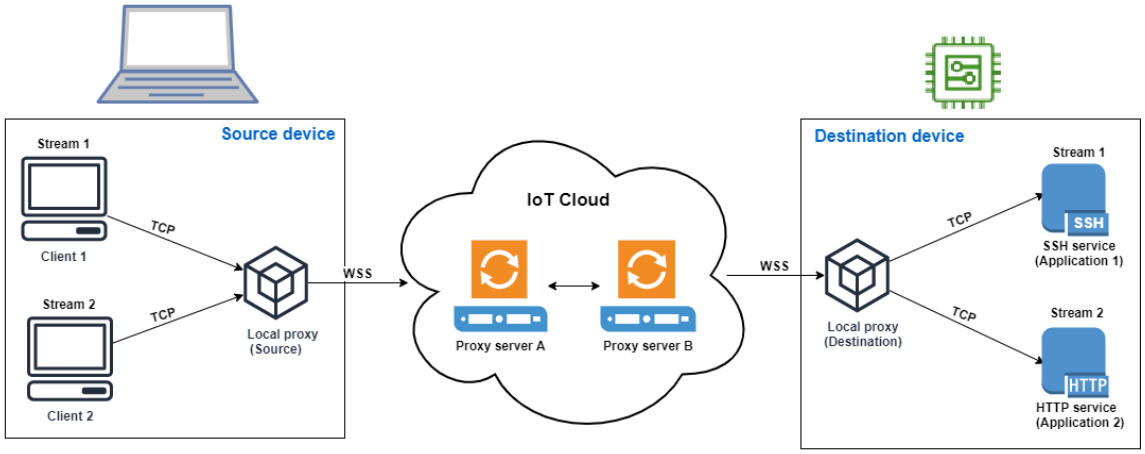
- 在安全隧道中多路复用多个数据流 (p. 758)
- 在安全隧道中使用同步 TCP 连接 (p. 760)

在安全隧道中多路复用多个数据流

您可以对使用多个连接或端口的设备使用多路复用功能。当您需要与远程设备建立多个连接以解决任何问题时，也可以使用多路复用功能。例如，对于需要发送多个 HTTP 和 SSH 数据流的 Web 浏览器，可以使用此功能。通过多路复用隧道将来自两个数据流的应用程序数据同时发送到设备。

示例使用案例

假设您需要连接到设备上的 Web 应用程序来更改某些联网参数，同时通过终端发出 shell 命令，以验证设备是否使用新的联网参数正常运行。在这种情况下，您可能需要通过 HTTP 和 SSH 连接到设备，并传输两个并行数据流，以便同时访问 Web 应用程序和终端。通过多路复用功能，这两个独立的流可以同时通过同一隧道传输。



如何设置多路复用隧道

以下流程将指导您如何设置多路复用隧道，以便使用需要连接到多个端口的应用程序对设备进行故障排除。您将设置一个隧道，其中包含两个多路复用流：一个 HTTP 流和一个 SSH 流。

1. （可选）创建配置文件

   您可以选择使用配置文件来配置源和目标设备。如果您的端口映射可能会经常发生更改，请使用配置文件。如果您希望使用 CLI 显式指定端口映射，或者如果您不需要在指定侦听端口上启动本地代理，则可跳过此步骤。有关如何使用配置文件的更多信息，请参阅中的通过--config 设置的选项 GitHub。

   1. 在源设备上，在将运行本地代理的文件夹中，创建一个名为 `Config` 的配置文件夹。在此文件夹中，使用以下内容创建一个名为 `SSHSource.ini` 的文件：

      ```
      HTTP1 = 5555
      SSH1 = 3333
      ```

   2. 在目标设备上，在将运行本地代理的文件夹中，创建一个名为 `Config` 的配置文件夹。在此文件夹中，使用以下内容创建一个名为 `SSHDestination.ini` 的文件：

      ```
      HTTP1 = 80
      ```

```
SSH1 = 22
```

2. 打开隧道

   使用 OpenTunnel API 操作或 open-tunnel CLI 命令打开隧道。通过将 SSH1 和 HTTP1 指定为与远程设备对应的 Amazon IoT 事物的服务和名称来配置目标。您的 SSH 和 HTTP 应用程序正在此远程设备上运行。您必须已在 Amazon IoT 注册表中创建 IoT 事物。有关更多信息，请参阅如何使用注册表管理事物 (p. 239)：

   ```
   aws iotsecuretunneling open-tunnel \
    --destination-config thingName=RemoteDevice1,services=HTTP1,SSH1
   ```

   运行此命令将生成源和目标访问令牌，您将使用这些令牌运行本地代理。

   ```
   {
    "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
    "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
   afa28b00cecd",
    "sourceAccessToken": source_client_access_token,
    "destinationAccessToken": destination_client_access_token
   }
   ```

3. 配置和启动本地代理

   必须先设置Amazon IoT设备客户端，或者从下载本地代理源代码GitHub并为您选择的平台构建本地代理，之后才能运行本地代理。之后，您可以启动目标和源本地代理以连接到安全隧道。有关配置和使用本地代理的更多信息，请参阅如何使用本地代理 (p. 749)。

   > **Note**
   >
   > 在源设备上，如果您不使用任何配置文件或使用 CLI 指定端口映射，则仍可以使用相同的命令来运行本地代理。在源模式下，本地代理将自动为您选取要使用的可用端口和映射。

   Start local proxy using configuration files

   运行以下命令可使用配置文件在源和目标模式下运行本地代理。

   ```
   // ----------------- Start the destination local proxy -----------------------
   ./localproxy -r us-east-1 -m dst -t destination_client_access_token

   // ----------------- Start the source local proxy ---------------------------
   // You also run the same command below if you want the local proxy to
   // choose the mappings for you instead of using configuration files.
   ./localproxy -r us-east-1 -m src -t source_client_access_token
   ```

   Start local proxy using CLI port mapping

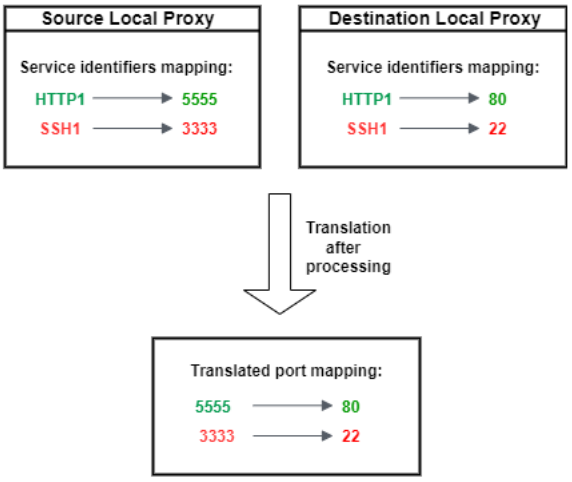
   运行以下命令可通过使用 CLI 显式指定端口映射，以便在源和目标模式下运行本地代理。

   ```
   // ----------------- Start the destination local proxy
    ----------------------------------
   ./localproxy -r us-east-1 -d HTTP1=80,SSH1=22 -t destination_client_access_token

   // ----------------- Start the source local proxy
    ---------------------------------------
   ./localproxy -r us-east-1 -s HTTP1=5555,SSH1=33 -t source_client_access_token
   ```

现在可以通过多路复用隧道同时传输来自 SSH 和 HTTP 连接的应用程序数据。如下面的地图中所示，服务标识符充当可读格式，用于转换源设备和目标设备之间的端口映射。利用此配置，安全隧道将来自源设备上

的端口 *5555* 的所有传入 HTTP 流量转发到目标设备上的端口 *80*，并将来自端口 *3333* 的所有传入 SSH 流量转发到目标设备上的端口 *22*。



在安全隧道中使用同步 TCP 连接

对于每个数据流，Amazon IoT 安全隧道同时支持多个 TCP 连接。当您需要建立与远程设备的同步连接时，可以使用此功能。使用同步 TCP 连接可减小客户端发出多个请求时发生超时的几率。例如，当访问正在运行多个组件的 Web 服务器时，同步 TCP 连接可减少加载站点所需的时间。

Note

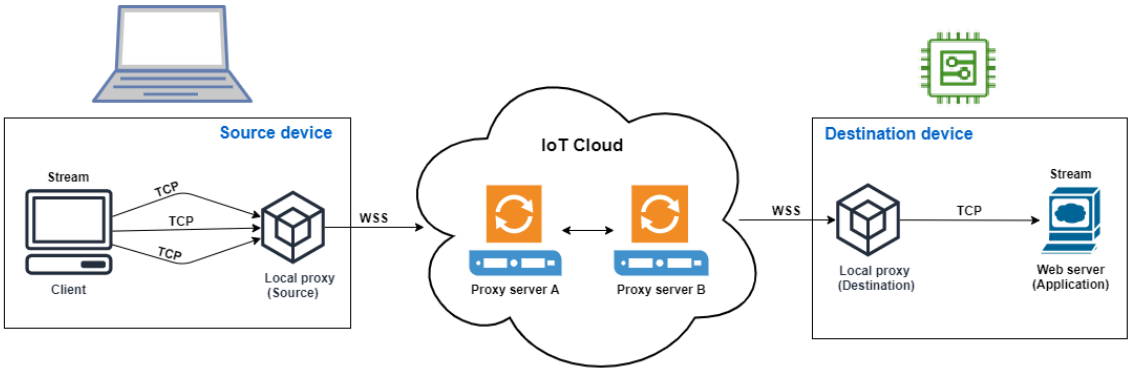
对于每个 Amazon Web Services 账户，同步 TCP 连接的带宽限制为每秒 800 KB。Amazon IoT 安全隧道会根据传入请求数为您配置此限制。

示例使用案例

假设您需要远程访问目标设备的本地 Web 服务器，并且该 Web 服务器上运行了多个组件。利用单一 TCP 连接，在尝试访问此 Web 服务器时，顺序加载会增加在站点上加载资源所花费的时间。同步 TCP 连接可以满足站点的资源要求来减少加载时间，从而减少访问时间。下图说明如何支持同步 TCP 连接，以便将数据流传输到远程设备上运行的 Web 服务器应用程序。

Note

如果要使用隧道访问远程设备上运行的多个应用程序，则可以使用隧道多路复用功能。有关更多信息，请参阅在安全隧道中多路复用多个数据流 (p. 758)：



如何使用同步 TCP 连接

以下过程将为您演练如何使用同步 TCP 连接来访问远程设备上的 Web 浏览器。当客户端发出多个请求时，Amazon IoT 安全隧道会自动设置同步 TCP 连接来处理这些请求，从而减少加载时间。

1. 打开隧道

   使用 OpenTunnel API 操作或 open-tunnel CLI 命令打开隧道。通过将 HTTP 指定为服务并指定与远程设备对应的 Amazon IoT 事物的名称来配置目标。您的 Web 服务器应用程序正在此远程设备上运行。您必须已在 Amazon IoT 注册表中创建 IoT 事物。有关更多信息，请参阅如何使用注册表管理事物 (p. 239)：

   ```
   aws iotsecuretunneling open-tunnel \
    --destination-config thingName=RemoteDevice1,services=HTTP
   ```

   运行此命令将生成源和目标访问令牌，您将使用这些令牌运行本地代理。

   ```
   {
    "tunnelId": "b2de92a3-b8ff-46c0-b0f2-afa28b00cecd",
    "tunnelArn": "arn:aws:iot:us-west-2:431600097591:tunnel/b2de92a3-b8ff-46c0-b0f2-
   afa28b00cecd",
    "sourceAccessToken": source_client_access_token,
    "destinationAccessToken": destination_client_access_token
   }
   ```

2. 配置和启动本地代理

   必须先下载本地代理源代码GitHub并为您选择的平台构建本地代理，之后才能运行本地代理。之后，您可以启动目标和源本地代理以连接到安全隧道，并开始使用远程 Web 服务器应用程序。

   Note

   要让 Amazon IoT 安全隧道使用同步 TCP 连接，您必须升级到最新版本的本地代理。如果您使用 Amazon IoT 设备客户端配置本地代理，则此功能不可用。

   ```
   // Start the destination local proxy
   ./localproxy -r us-east-1 -d HTTP=80 -t destination_client_access_token

   // Start the source local proxy
   ./localproxy -r us-east-1 -s HTTP=5555 -t source_client_access_token
   ```

   有关配置和使用本地代理的更多信息，请参阅如何使用本地代理 (p. 749)。

现在，您可以使用隧道访问 Web 服务器应用程序。当客户端发出多个请求时，Amazon IoT 安全隧道将自动设置和处理同步 TCP 连接。

配置远程设备和使用 IoT 代理

IoT 代理用于接收包含客户端访问令牌的 MQTT 消息，并在远程设备上启动本地代理。如果希望安全隧道使用 MQTT 传送客户端访问令牌，则必须在远程设备上安装并运行 IoT 代理。IoT 代理必须订阅以下保留的 IoT MQTT 主题：

Note

如果要通过订阅保留 MQTT 主题之外的方法将目标客户端访问令牌传送到远程设备，您可能需要目标客户端访问令牌 (CAT) 侦听器和本地代理。CAT 侦听器必须使用您选择的客户端访问令牌传送机制，并且能够在目标模式下启动本地代理。

IoT 代理代码段

IoT 代理必须订阅以下保留 IoT MQTT 主题，这样才能接收 MQTT 消息并启动本地代理：

`$aws/things/thing-name/tunnels/notify`

其中，`thing-name` 是与远程设备关联的 Amazon IoT 事物的名称。

以下是 MQTT 消息负载示例：

```
{
    "clientAccessToken": "destination-client-access-token",
    "clientMode": "destination",
    "region": "aws-region",
    "services": ["destination-service"]
}
```

当您收到 MQTT 消息后，IoT 代理必须在远程设备上使用适当参数启动本地代理。

以下 Java 代码演示了如何使用Device SDK 和 ProcessBuilderJava 库构建简单的 IoT 代理，以使用安全隧道。Amazon IoT

```
// Find the IoT device endpoint for your Amazon Web Services ##
final String endpoint = iotClient.describeEndpoint(new
 DescribeEndpointRequest().withEndpointType("iot:Data-ATS")).getEndpointAddress();

// Instantiate the IoT Agent with your Amazon credentials
final String thingName = "RemoteDeviceA";
final String tunnelNotificationTopic = String.format("$aws/things/%s/tunnels/notify",
 thingName);
final AWSIotMqttClient mqttClient = new AWSIotMqttClient(endpoint, thingName,
                 "your_aws_access_key", "your_aws_secret_key");

try {
    mqttClient.connect();
    final TunnelNotificationListener listener = new
 TunnelNotificationListener(tunnelNotificationTopic);
    mqttClient.subscribe(listener, true);
}
finally {
    mqttClient.disconnect();
}

private static class TunnelNotificationListener extends AWSIotTopic {
    public TunnelNotificationListener(String topic) {
        super(topic);
    }

    @Override
    public void onMessage(AWSIotMessage message) {
        try {
            // Deserialize the MQTT message
            final JSONObject json = new JSONObject(message.getStringPayload());

            final String accessToken = json.getString("clientAccessToken");
            final String region = json.getString("region");

            final String clientMode = json.getString("clientMode");
            if (!clientMode.equals("destination")) {
                throw new RuntimeException("Client mode " + clientMode + " in the MQTT
 message is not expected");
            }
```

```
            final JSONArray servicesArray = json.getJSONArray("services");
            if (servicesArray.length() > 1) {
                throw new RuntimeException("Services in the MQTT message has more than 1
 service");
            }
            final String service = servicesArray.get(0).toString();
            if (!service.equals("SSH")) {
                throw new RuntimeException("Service " + service + " is not supported");
            }

            // Start the destination local proxy in a separate process to connect to the
 SSH Daemon listening port 22
            final ProcessBuilder pb = new ProcessBuilder("localproxy",
                        "-t", accessToken,
                        "-r", region,
                        "-d", "localhost:22");
            pb.start();
        }
        catch (Exception e) {
            log.error("Failed to start the local proxy", e);
        }
    }
}
```

控制对隧道的访问

安全隧道提供特定于服务的操作、资源和条件上下文键，以供在 IAM 权限策略中使用。

隧道访问先决条件

- 了解如何使用 IAM policy保护Amazon资源。
- 了解如何创建和评估 IAM 条件。
- 了解如何使用资源标签保护Amazon资源。

隧道访问策略

您必须使用以下策略来授予使用安全隧道 API 的权限。有关 Amazon IoT 安全的更多信息，请参阅适用于 Amazon IoT 的 Identity and Access Management (p. 348)。

物联网：OpenTunnel

`iot:OpenTunnel` 策略操作授予调用 OpenTunnel 的委托人权限。

在 IAM policy 声明的 `Resource` 元素中：

- 指定通配符隧道 ARN：

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:tunnel/*`
- 指定事物 ARN 来管理对特定 IoT 事物的 `OpenTunnel` 权限：

  `arn:aws:iot:`*`aws-region`*`:`*`aws-account-id`*`:thing/`*`thing-name`*

例如，以下策略语句允许您打开到名为 `TestDevice` 的 IoT 事物的隧道。

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

iot:OpenTunnel 策略操作支持以下条件键：

- iot:ThingGroupArn
- iot:TunnelDestinationService
- aws:RequestTag/####
- aws:SecureTransport
- aws:TagKeys

如果事物属于名称以 TestGroup 开头的事物组，并且隧道上配置的目标服务是 SSH，则以下策略语句允许您打开到该事物的隧道。

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ]
        }
    }
}
```

您还可以使用资源标签来控制打开隧道的权限。例如，如果存在值为 Admin 的标签键 Owner 并且未指定其他标签，则以下策略语句允许打开隧道。有关使用标签的一般信息，请参阅给您的 Amazon IoT 资源加标签 (p. 259)。

```
{
    "Effect": "Allow",
    "Action": "iot:OpenTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:RequestTag/Owner": "Admin"
        },
        "ForAllValues:StringEquals": {
            "aws:TagKeys": "Owner"
        }
    }
}
```

物联网：RotateTunnelAccessToken

iot:RotateTunnelAccessToken 策略操作授予调用 RotateTunnelAccessToken 的委托人权限。

在 IAM policy 声明的 Resource 元素中：

- 指定完全限定的隧道 ARN：

  arn:aws:iot:*aws-region*: *aws-account-id*:tunnel/*tunnel-id*

  您也可以使用通配符隧道 ARN：

  arn:aws:iot:*aws-region*:*aws-account-id*:tunnel/*
- 指定事物 ARN 来管理对特定 IoT 事物的 RotateTunnelAccessToken 权限：

  arn:aws:iot:*aws-region*:*aws-account-id*:thing/*thing-name*

例如，以下策略语句允许您为名为 TestDevice 的 IoT 事物轮换隧道的源访问令牌或客户端的目标访问令牌。

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
}
```

iot:RotateTunnelAccessToken 策略操作支持以下条件键：

- iot:ThingGroupArn
- iot:TunnelDestinationService
- iot:ClientMode
- aws:SecureTransport

如果事物属于名称以 TestGroup 开头的事物组，隧道上配置的目标服务是 SSH，并且客户端处于 DESTINATION 模式，则以下策略语句允许您对此事物轮换目标访问令牌。

```
{
    "Effect": "Allow",
    "Action": "iot:RotateTunnelAccessToken",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "ForAnyValue:StringLike": {
            "iot:ThingGroupArn": [
                "arn:aws:iot:aws-region:aws-account-id:thinggroup/TestGroup*"
            ]
        },
        "ForAllValues:StringEquals": {
            "iot:TunnelDestinationService": [
                "SSH"
            ],
            "iot:ClientMode": "DESTINATION"
        }
    }
```

```
}
```

物联网：DescribeTunnel

iot:DescribeTunnel 策略操作授予调用 DescribeTunnel 的委托人权限。

在 IAM policy 语句的 Resource 元素中，指定完全限定的隧道 ARN：

arn:aws:iot:*aws-region*: *aws-account-id*:tunnel/*tunnel-id*

您也可以使用通配符 ARN：

arn:aws:iot:*aws-region*:*aws-account-id*:tunnel/*

iot:DescribeTunnel 策略操作支持以下条件键：

- aws:ResourceTag/*tag-key*
- aws:SecureTransport

以下策略语句允许您在请求的隧道使用值为 Admin 的键 Owner 进行标记时调用 DescribeTunnel。

```
{
    "Effect": "Allow",
    "Action": "iot:DescribeTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/Owner": "Admin"
        }
    }
}
```

物联网：ListTunnels

iot:ListTunnels 策略操作授予调用 ListTunnels 的委托人权限。

在 IAM policy 声明的 Resource 元素中：

- 指定通配符隧道 ARN：

  arn:aws:iot:*aws-region*:*aws-account-id*:tunnel/*
- 指定事物 ARN 来管理对所选 IoT 事物的 ListTunnels 权限：

  arn:aws:iot:*aws-region*:*aws-account-id*:thing/*thing-name*

iot:ListTunnels 策略操作支持条件键 aws:SecureTransport。

以下策略语句允许您列出名为 TestDevice 的事物的隧道。

```
{
    "Effect": "Allow",
    "Action": "iot:ListTunnels",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*",
        "arn:aws:iot:aws-region:aws-account-id:thing/TestDevice"
    ]
```

```
}
```

物联网：ListTagsForResource

iot:ListTagsForResource 策略操作授予调用 ListTagsForResource 的委托人权限。

在 IAM policy 语句的 Resource 元素中，指定完全限定的隧道 ARN：

arn:aws:iot:*aws-region*: *aws-account-id*:tunnel/*tunnel-id*

您也可以使用通配符隧道 ARN：

arn:aws:iot:*aws-region*:*aws-account-id*:tunnel/*

iot:ListTagsForResource 策略操作支持条件键 aws:SecureTransport。

物联网：CloseTunnel

iot:CloseTunnel 策略操作授予调用 CloseTunnel 的委托人权限。

在 IAM policy 语句的 Resource 元素中，指定完全限定的隧道 ARN：

arn:aws:iot:*aws-region*: *aws-account-id*:tunnel/*tunnel-id*

您也可以使用通配符隧道 ARN：

arn:aws:iot:*aws-region*:*aws-account-id*:tunnel/*

iot:CloseTunnel 策略操作支持以下条件键：

- iot:Delete
- aws:ResourceTag/*tag-key*
- aws:SecureTransport

以下策略语句允许您在以下情况下调用 CloseTunnel：请求的 Delete 参数是 false，并且请求的隧道使用值为 QATeam 的键 Owner 进行标记。

```
{
    "Effect": "Allow",
    "Action": "iot:CloseTunnel",
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
    ],
    "Condition": {
        "Bool": {
            "iot:Delete": "false"
        },
        "StringEquals": {
            "aws:ResourceTag/Owner": "QATeam"
        }
    }
}
```

物联网：TagResource

iot:TagResource 策略操作授予调用 TagResource 的委托人权限。

在 IAM policy 语句的 Resource 元素中，指定完全限定的隧道 ARN：

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

您也可以使用通配符隧道 ARN：

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

`iot:TagResource` 策略操作支持条件键 `aws:SecureTransport`。

物联网：UntagResource

`iot:UntagResource` 策略操作授予调用 `UntagResource` 的委托人权限。

在 IAM policy 语句的 `Resource` 元素中，指定完全限定的隧道 ARN：

`arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id`

您也可以使用通配符隧道 ARN：

`arn:aws:iot:aws-region:aws-account-id:tunnel/*`

`iot:UntagResource` 策略操作支持条件键 `aws:SecureTransport`。

通过轮换客户端访问令牌解决 Amazon IoT 安全隧道连接问题

当您使用 Amazon IoT 安全隧道时，即使隧道处于打开状态，也可能会遇到连接问题。以下各节显示了一些可能的问题，以及如何通过轮换客户端访问令牌来解决这些问题。要轮换客户端访问令牌 (CAT)，请使用 RotateTunnelAccessTokenAPI 或rotate-tunnel-access-tokenAmazon CLI。根据是在源模式还是目标模式下使用客户端时遇到错误，您可以在源模式和/或目标模式下轮换 CAT。

Note

- 如果您不确定是否需要在源或目标上轮换 CAT，则可以在使用 `RotateTunnelAccessToken` API 时通过将 `ClientMode` 设置为 `ALL`，这样就能同时在源和目标上轮换 CAT。
- 轮换 CAT 不会延长隧道持续时间。例如，假设隧道持续时间为 12 小时，隧道已经打开了 4 个小时。轮换访问令牌时，生成的新令牌只能在剩余的 8 小时内使用。

主题

- 客户端访问令牌错误无效 (p. 768)
- 客户端令牌不匹配错误 (p. 769)
- 远程设备连接问题 (p. 769)

客户端访问令牌错误无效

使用 Amazon IoT 安全隧道时，如果使用同一客户端访问令牌 (CAT) 重新连接到同一隧道，则可能会遇到连接错误。在这种情况下，本地代理无法连接到安全隧道代理服务器。如果您在源模式下使用客户端，您可能会看到以下错误消息：

```
Invalid access token: The access token was previously used and cannot be used again
```

出现此错误的原因是，本地代理只能使用一次客户端访问令牌 (CAT)，然后它变为无效。要纠正此错误，请在 SOURCE 模式下轮换客户端访问令牌以便为源生成新的 CAT。有关演示如何轮换源 CAT 的示例，请参阅轮换源 CAT 示例 (p. 769)。

客户端令牌不匹配错误

> **Note**
>
> 不建议通过客户端令牌来重复使用 CAT。我们建议您改用 RotateTunnelAccessToken API 来轮换客户端访问令牌，以重新连接到隧道。

如果您使用的是客户端令牌，则可以重复使用 CAT 以重新连接到隧道。要重复使用 CAT，您必须在首次连接到安全隧道时向 CAT 提供客户端令牌。安全隧道存储客户端令牌，因此，对于使用相同令牌的后续连接尝试，还必须提供客户端令牌。有关使用客户端令牌的更多信息，请参阅中的本地代理引用实现 GitHub。

使用客户端令牌时，如果在源模式下使用客户端，则可能会看到以下错误：

```
Invalid client token: The provided client token does not match the client token
    that was previously set.
```

出现此错误的原因是提供的客户端令牌与访问隧道时随 CAT 提供的客户端令牌不匹配。要纠正此错误，请在 SOURCE 模式下轮换 CAT 以便为源生成新的 CAT。下面是一个示例：

轮换源 CAT 示例

以下示例显示如何在 SOURCE 模式下运行 RotateTunnelAccessToken API，以便为源代码生成新的 CAT：

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --region <region> \
    --tunnel-id <tunnel-id> \
    --client-mode SOURCE
```

运行此命令将生成一个新的源访问令牌并返回隧道的 ARN。

```
{
    "sourceAccessToken": "<source-access-token>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/<tunnel-id>"
}
```

您现在可以使用新的源令牌在源模式下连接本地代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=<source-access-token>
./localproxy -r <region> -s <port>
```

下面的内容显示运行本地代理的示例输出。

```
...

[info]    Starting proxy in source mode
...
[info]    Successfully established websocket connection with proxy server ...
[info]    Listening for new connection on port <port>
...
```

远程设备连接问题

当使用 Amazon IoT 安全隧道时，即使隧道处于打开状态，设备也可能会意外断开连接。要确定设备是否仍然连接到隧道，可以使用 DescribeTunnelAPI 或 d escribe-tunnelAmazon CLI。

设备可能由于多种原因而断开连接。要解决连接问题，如果设备由于以下可能的原因断开连接，则可以在目标上轮换 CAT：

- 目标上的 CAT 变为无效。
- 令牌未通过安全隧道保留的 MQTT 主题传送到设备：

  $aws/things/*<thing-name>*/tunnels/notify

下面的示例演示了如何解决此问题：

轮换目标 CAT 示例

考虑远程设备 *<RemoteThing1>*。要为该事物打开隧道，您可以使用以下命令：

```
aws iotsecuretunneling open-tunnel \
    --region <region> \
    --destination-config thingName=<RemoteThing1>,services=SSH
```

运行此命令将为源和目标生成隧道详细信息和 CAT。

```
{
    "sourceAccessToken": "<source-access-token>",
    "destinationAccessToken": "<destination-access-token>",
    "tunnelId": "<tunnel-id>",
    "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/tunnel-id"
}
```

但是，当您使用 DescribeTunnelAPI 时，输出指示设备已断开连接，如下图所示：

```
aws iotsecuretunneling describe-tunnel \
    --tunnel-id <tunnel-id> \
    --region <region>
```

运行此命令将显示设备仍未连接。

```
{
    "tunnel": {
        ...
        "destinationConnectionState": {
            "status": "DISCONNECTED"
        },
        ...
    }
}
```

要解决此错误，请在客户端处于 DESTINATION 模式下运行 RotateTunnelAccessToken API，并为目标运行配置。运行此命令将撤销旧的访问令牌，生成新令牌，并将此令牌重新发送到 MQTT 主题：

$aws/things/*<thing-name>*/tunnels/notify

```
aws iotsecuretunneling rotate-tunnel-access-token \
    --tunnel-id <tunnel-id> \
    --client-mode DESTINATION \
    --destination-config thingName=<RemoteThing1>,services=SSH \
    --region <region>
```

运行此命令将生成新的访问令牌，如下所示。然后，如果设备代理设置正确，令牌将传送到设备以连接到隧道。

```
{
    "destinationAccessToken": "destination-access-token",
    "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

设备预调配

Amazon提供了几种不同的方法来预调配设备并在其上安装唯一的客户端证书。本部分介绍各种方式，以及如何选择适合您的 IoT 解决方案的最佳方法。这些选项在标题为在 Amazon IoT Core 中使用 X.509 证书进行设备制造和预调配的白皮书中进行了详细介绍。

选择最适合您情况的选项

- 您可以在 IoT 设备上安装证书，然后再予交付

  如果您可以在 IoT 设备上安全地安装唯一的客户端证书以供最终用户使用，则需要使用即时预调配 (JITP) (p. 780) 或者即时注册 (JITR) (p. 279)。

  通过 JITP 和 JITR，当设备首次连接时，用于签署设备证书的证书颁发机构 (CA) 会使用 Amazon IoT 注册并且被 Amazon IoT 识别。设备在首次连接时将使用其预调配模板中的详细信息在 Amazon IoT 中预调配。

  有关单件事物、JITP、JITR 和具有唯一证书的设备批量预调配的详细信息，请参阅 the section called “预调配具有设备证书的设备” (p. 779)。
- 终端用户或安装人员可以使用应用程序在其 IoT 设备上安装证书

  如果您无法在 IoT 设备上安全地安装唯一的客户端证书以供终端用户使用，但终端用户或安装程序可以使用应用程序注册设备并安装唯一设备证书，则需要执行通过信任用户预调配 (p. 775)的过程。

  使用信任用户（如终端用户或具有已知账户的安装程序）可以简化设备制造过程。设备拥有的并非唯一的客户端证书，而是一个临时证书，仅能使设备与 Amazon IoT 连接 5 分钟。在这 5 分钟的窗口中，信任用户将获得具有更长寿命的唯一客户端证书，并将其安装在设备上。申请证书的寿命有限，因此能够最大限度地减少证书受损的风险。

  有关更多信息，请参阅the section called “由可信用户预调配” (p. 775)。
- 终端用户无法使用应用程序在其 IoT 设备上安装证书

  如果上述两个选项都不适用于您的 IoT 解决方案，则可选择通过申请预调配 (p. 774)流程。通过此流程，您的 IoT 设备将拥有由实例集中其它设备共享的申请证书。设备首次使用申请证书进行连接时，Amazon IoT 使用其预调配模板注册设备，并向设备颁发其唯一的客户端证书，以便随后访问 Amazon IoT。

  此选项可在设备连接到 Amazon IoT 时启用设备自动预调配，但在申请证明受损的情况下可能会面临较大风险。如果申请证书受损，您可以停用该证书。停用申请证书可防止具有该申请证书的所有设备在将来不会被注册。但是，停用申请证书不会阻止已预调配的设备。

  有关更多信息，请参阅the section called “通过申请进行预调配” (p. 774)。

在 Amazon IoT 中预调配设备

使用 Amazon IoT 预调配设备时，您必须创建资源，以便您的设备和 Amazon IoT 能够安全地通信。您可以创建其它资源来帮助管理设备实例集。在预调配过程中可以创建以下资源：

- IoT 事物。

  IoT 事物是 Amazon IoT 设备注册表中的条目。每个事物都有一个唯一的名称和一组属性，并与物理设备相关联。事物可以使用事物类型来定义，或者分组为事物组。有关更多信息，请参阅使用 Amazon IoT 管理设备 (p. 239)。

虽然创建事物并非必需，但这使您能够按事物类型、事物组和事物属性来搜索设备，以更有效地管理设备实例集。有关更多信息，请参阅机群索引 (p. 803)。

Note

要使 Fleet Hub 对事物的连接状态数据编制索引，请预调配事物并对其进行配置，以便事物名称与连接请求中使用的客户端 ID 匹配。

- X.509 证书。

  设备使用 X.509 证书执行与 Amazon IoT 的相互身份验证。您可以注册现有证书，也可以让 Amazon IoT 为您生成并注册新证书。通过将证书附加到表示设备的事物，您可以将证书与设备关联。您还必须将证书和关联的私有密钥复制到设备上。设备在连接到 Amazon IoT 时提供证书。有关更多信息，请参阅身份验证 (p. 264)。
- IoT 策略。

  IoT 策略确定设备可在 Amazon IoT 中执行的操作。IoT 策略附加到设备证书。当设备向 Amazon IoT 提供其证书时，将向其授予在策略中指定的权限。有关更多信息，请参阅Authorization (p. 298)。每个设备都需要一个证书来与 Amazon IoT 通信。

Amazon IoT 支持使用预调配模板自动预调配实例集。预调配模板描述 Amazon IoT 预调配您的设备所需的资源。模板包含变量，使您能够使用一个模板来预调配多个设备。在您预调配设备时，您可以使用字典或映射为设备特定的变量指定值。要预调配其它设备，请在字典中指定新值。

无论设备是否具有唯一的证书（及其关联的私有密钥），您都可以使用自动预调配。

实例集预调配 API

实例集预调配中使用的 API 分为几类：

- 这些控制面板函数可创建和管理实例集预调配模板，并配置可信用户策略。
  - CreateProvisioningTemplate
  - CreateProvisioningTemplateVersion
  - DeleteProvisioningTemplate
  - DeleteProvisioningTemplateVersion
  - DescribeProvisioningTemplate
  - DescribeProvisioningTemplateVersion
  - ListProvisioningTemplates
  - ListProvisioningTemplateVersions
  - UpdateProvisioningTemplate
- 可信用户可以使用此控制面板函数生成临时信息载入申请。此临时申请在 Wi-Fi 配置或类似方法期间传递给设备。
  - CreateProvisioningClaim
- 预调配过程中设备使用的 MQTT API，其中包含嵌入设备中的预调配申请证书，或由可信用户传递到其中。
  - the section called “CreateCertificateFromCsr” (p. 798)
  - the section called “CreateKeysAndCertificate” (p. 799)
  - the section called “RegisterThing” (p. 801)

使用实例集预调配来预调配没有设备证书的设备

通过使用 Amazon IoT 实例集预调配，Amazon IoT 可以在设备首次连接到 Amazon IoT 时生成设备证书和私有密钥，并将它们安全地传递到设备。Amazon IoT 将提供由 Amazon Root 证书颁发机构 (CA) 签署的客户端证书

使用实例集预调配的方法有两种：

- 通过申请进行预调配 (p. 774)
- 由可信用户预调配 (p. 775)

通过申请进行预调配

设备在制造时可以嵌入预调配申请证书和私有密钥（这是特殊用途凭证）。如果将这些证书注册到 Amazon IoT，服务可以交换它们，作为设备可用于常规操作的唯一设备证书。此过程包括以下步骤：

在交付设备之前

1. 调用 CreateProvisioningTemplate 以创建预调配模板。此 API 返回模板 ARN。有关更多信息，请参阅设备预调配 MQTT API (p. 797)。

   您还可以从 Amazon IoT 控制台创建实例集预调配模板。

   a. 从导航窗格中选择 Connect（连接），然后选择 Fleet provisioning templates（实例集预调配模板）。

   b. 选择 Create template（创建模板），再按照提示操作。
2. 创建用作预调配申请证书的证书以及关联的私有密钥。
3. 将这些证书注册到 Amazon IoT 并关联限制证书使用的 IoT 策略。以下示例 IoT 策略限制使用与此策略关联的证书来预调配设备。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["iot:Connect"],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": ["iot:Publish","iot:Receive"],
            "Resource": [
                "arn:aws:iot:aws-region:aws-account-id:topic/$aws/certificates/create/
*",
                "arn:aws:iot:aws-region:aws-account-id:topic/$aws/provisioning-
templates/templateName/provision/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:aws:iot:aws-region:aws-account-id:topicfilter/$aws/certificates/
create/*",
                "arn:aws:iot:aws-region:aws-account-id:topicfilter/$aws/provisioning-
templates/templateName/provision/*"
            ]
```

```
        }
    ]
}
```

4. 授予 Amazon IoT 服务权限，以在预调配设备时创建或更新 IoT 资源，例如账户中的事物和证书。通过将 AWSIoTThingsRegistration 托管策略附加到信任 Amazon IoT 服务委托人的 IAM 角色（称为预调配角色）来执行此操作。
5. 制造在其中安全地嵌入了预调配申请证书的设备。

设备现在已准备就绪，可以运输到要安装以使用它的位置。

Important

预调配申请私有密钥应始终得到保护，包括在设备上时。我们建议您使用 Amazon IoT CloudWatch 指标和日志来监控滥用迹象。如果您检测到滥用，请关闭预调配申请证书，以使其不能用于设备预调配。

初始化设备以供使用

1. 设备使用 Amazon IoT Device SDK、Mobile SDK 和 Amazon IoT Device Client (p. 1283)连接到 Amazon IoT，并使用安装在设备上的预调配申请证书向其进行身份验证。

   Note

   为了安全起见，由 CreateCertificateFromCsr (p. 798) 和 CreateKeysAndCertificate (p. 799) 返回的 certificateOwnershipToken 将在一小时后过期。必须在 certificateOwnershipToken 过期前调用 RegisterThing (p. 801)。如果由 CreateCertificateFromCsr (p. 798) 或 CreateKeysAndCertificate (p. 799) 创建的证书尚未激活，并且在令牌过期时尚未附加到策略或事物，则该证书将被删除。如果令牌过期，设备可以再次调用 CreateCertificateFromCsr (p. 798) 或 CreateKeysAndCertificate (p. 799) 以生成新证书。
2. 设备通过使用以下选项之一获取永久证书和私有密钥。设备将使用此证书和密钥向 Amazon IoT 进行将来的所有身份验证。

   a. 调用 CreateKeysAndCertificate (p. 799) 以使用 Amazon 证书颁发机构创建新的证书和私有密钥。

      或者

   b. 调用 CreateCertificateFromCsr (p. 798) 以通过证书签名请求生成证书，这可确保其私有密钥安全。
3. 从设备中，调用 RegisterThing (p. 801) 以向 Amazon IoT 注册设备并创建云资源。

   实例集预调配服务使用预调配模板来定义和创建云资源，如 IoT 事物。该模板可以指定事物的属性和所属的组。在将新事物添加到事物组之前，事物组必须先存在。
4. 在设备上保存永久证书后，设备必须断开与它使用预调配申请证书发起的会话的连接，并使用此永久证书重新连接。

设备现在已准备好与 Amazon IoT 正常通信。

由可信用户预调配

在许多情况下，当可信用户（如最终用户或安装技术人员）使用移动应用程序在设备的部署位置配置设备时，设备首次连接到 Amazon IoT。

Important

您必须管理可信用户的访问和权限才能执行此流程。做到这一点的一种方法是为可信用户提供和维护一个账户，该账户对他们进行身份验证，并授予他们对执行此过程所需的 Amazon IoT 功能和 API 操作的访问权限。

在交付设备之前

1. 调用 CreateProvisioningTemplate 以创建预调配模板并返回其 *templateArn* 和 *templateName*。
2. 创建由信任用户用于启动预调配过程的 IAM 角色。预调配模板仅允许该用户预调配设备。例如：

```
{
    "Effect": "Allow",
    "Action": [
        "iot:CreateProvisioningClaim"
    ],
    "Resource": [
        "arn:aws:iot:aws-region:aws-account-id:provisioningtemplate/templateName"
    ]
}
```

3. 授予 Amazon IoT 服务权限，以在预调配设备时创建或更新 IoT 资源，例如账户中的事物和证书。您可以通过将 AWSIoTThingsRegistration 托管策略附加到信任 Amazon IoT 服务委托人的 IAM 角色（称为预调配角色）来执行此操作。
4. 提供标识您的可信用户的方法，例如，为他们提供一个账户，该账户可以对他们进行身份验证，并授权他们与注册其设备所需的 Amazon API 操作进行交互。

初始化设备以供使用

1. 可信用户登录您的预调配移动应用程序或 Web 服务。
2. 移动应用程序或 Web 应用程序使用 IAM 角色并调用 CreateProvisioningClaim，以从 Amazon IoT 获取临时预调配申请证书。

   Note

   出于安全考虑，CreateProvisioningClaim 返回的临时预调配申请证书只有五分钟的有效期。在临时预调配申请证书过期之前，以下步骤必须成功地返回有效证书。临时预调配申请证书不会显示在您账户的证书列表中。
3. 移动应用程序或 Web 应用程序将临时预调配申请证书以及任何必需的配置信息（如 Wi-Fi 凭证）提供给设备。
4. 设备使用临时预调配申请证书，通过 Amazon IoT Device SDK、Mobile SDK 和 Amazon IoT Device Client (p. 1283) 连接到 Amazon IoT。
5. 设备通过使用临时设置声明证书在连接到 Amazon IoT 的五分钟内使用其中一个选项获得永久证书和私钥。设备将使用这些选项返回的证书和密钥向 Amazon IoT 进行未来的所有身份验证。

   a. 调用 CreateKeysAndCertificate (p. 799) 以使用 Amazon 证书颁发机构创建新的证书和私有密钥。

      或者

   b. 调用 CreateCertificateFromCsr (p. 798) 以通过证书签名请求生成证书，这可确保其私有密钥安全。

> **Note**
>
> 请记住，CreateKeysAndCertificate (p. 799) 或 CreateCertificateFromCsr (p. 798) 必须在使用临时预调配申请证书连接到 Amazon IoT 后的五分钟内返回有效证书。

6. 设备调用 RegisterThing (p. 801) 以向 Amazon IoT 注册设备并创建云资源。

   实例集预调配服务使用预调配模板来定义和创建云资源，如 IoT 事物。该模板可以指定事物的属性和所属的组。在将新事物添加到事物组之前，事物组必须先存在。
7. 在设备上保存永久证书后，设备必须断开与它使用临时预调配申请证书发起的会话的连接，并使用此永久证书重新连接。

设备现在已准备好与 Amazon IoT 正常通信。

在 Amazon CLI 中使用预先预调配挂钩

以下流程使用预先预调配挂钩创建预调配模板。此处使用的 Lambda 函数是一个可以修改的示例。

创建预先预调配挂钩并将其应用到预调配模板

1. 创建一个具有定义输入和输出的 Lambda 函数。Lambda 函数高度可自定义，需要 `allowProvisioning` 和 `parameterOverrides` 来创建预调配挂钩。有关创建 Lambda 函数的更多信息，请参阅将 Amazon Lambda 与Amazon命令行界面一起使用。

   以下是 Lambda 函数输出的示例：

   ```
   {
     "allowProvisioning": True,
     "parameterOverrides": {
       "incomingKey0": "incomingValue0",
       "incomingKey1": "incomingValue1"
     }
   }
   ```
2. Amazon IoT 使用基于资源的策略来调用 Lambda，因此您需要授予 Amazon IoT 权限以调用 Lambda 函数。

   > **Important**
   >
   > 请务必在附加到 Lambda 操作策略的全局条件上下文键中包含 `source-arn` 或 `source-account`，以便防范权限操纵。有关此问题的更多信息，请参阅 防止跨服务混淆代理 (p. 308)。

   以下示例使用 add-permission 将 IoT 权限授予您的 Lambda。

   ```
   aws lambda add-permission \
       --function-name myLambdaFunction \
       --statement-id iot-permission \
       --action lambda:InvokeFunction \
       --principal iot.amazonaws.com
   ```
3. 使用 create-provisioning-template 或 update-provisioning-template 命令添加预先预调配挂钩到模板。

   以下 CLI 示例使用 create-provisioning-template 创建具有预先预调配挂钩的预调配模板：

   ```
   aws iot create-provisioning-template \
       --template-name myTemplate \
   ```

```
    --provisioning-role-arn arn:aws:iam:us-east-1:1234564789012:role/myRole \
    --template-body file://template.json \
    --pre-provisioning-hook file://hooks.json
```

此命令的输出如下所示：

```
{
    "templateArn": "arn:aws:iot:us-east-1:1234564789012:provisioningtemplate/
myTemplate",
    "defaultVersionId": 1,
    "templateName": myTemplate
}
```

您也可以从文件加载参数，而不是将其作为命令行参数值完全键入，以节省时间。有关更多信息，请参阅从文件加载 Amazon CLI 参数。下面显示了扩展 JSON 格式的 template 参数：

```
{
    "Parameters" : {
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
                "ThingName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingTypeName" : {"Fn::Join":["",["ThingTypePrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["widgets", "WA"],
                "BillingGroup": "BillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
```

```
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:504350838278:topic/foo/
bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"},
 "LocationUrl"]}
    }
}
```

下面显示了扩展 JSON 格式的 pre-provisioning-hook 参数：

```
{
     "targetArn" : "arn:aws:lambda:us-
east-1:765219403047:function:pre_provisioning_test",
     "payloadVersion" : "2020-04-01"
}
```

预调配具有设备证书的设备

Amazon IoT 提供了三种方法，用于在设备上已有设备证书（和关联的私有密钥）时预调配设备：

- 使用预调配模板进行单一事物预调配。如果您只需一次预调配一台设备，那么这是一个很好的选择。
- 使用在首次连接到 Amazon IoT 时预调配设备的模板进行即时预调配 (JITP)。如果您需要注册大量设备，但没有可以整合到批量预调配列表中的有关这些设备的信息，那么这是一个很好的选择。
- 批量注册。此选项可让您指定存储在 S3 存储桶内的文件中的单一事物预调配模板值。如果您有大量已知设备，并且可以将它们所需的特性整合到列表中，则此方法很有用。

主题

- 单个事物预调配 (p. 779)
- 即时预调配 (p. 780)
- 批量注册 (p. 784)

单个事物预调配

要预调配事物，请使用 RegisterThing API 或 register-thing CLI 命令。register-thing CLI 命令接受以下参数：

--template-body

预调配模板。

--parameters

在预调配模板中使用的参数的名称-值对列表，采用 JSON 格式（例如，{"ThingName" : "MyProvisionedThing", "CSR" : "*csr-text*"}）。

请参阅预调配模板 (p. 784)。

RegisterThing 或 register-thing 返回资源的 ARN 以及它创建的证书的文本：

```
{
    "certificatePem": "certificate-text",
    "resourceArns": {
    "PolicyLogicalName": "arn:aws:iot:us-
west-2:123456789012:policy/2A6577675B7CD1823E271C7AAD8184F44630FFD7",
    "certificate": "arn:aws:iot:us-west-2:123456789012:cert/
cd82bb924d4c6ccbb14986dcb4f40f30d892cc6b3ce7ad5008ed6542eea2b049",
    "thing": "arn:aws:iot:us-west-2:123456789012:thing/MyProvisionedThing"
    }
}
```

如果在字典中省略了参数，则使用默认值。如果未指定默认值，则不使用值来替换参数。

即时预调配

您可以在设备首次尝试连接到 Amazon IoT 时使用即时预调配（JITP）对其进行预调配。要预调配设备，您必须启用自动注册，并将预调配模板与用于对设备证书进行签名的 CA 证书关联。预调配成功和错误记录均作为 设备预配置指标 (p. 410) 录入 Amazon CloudWatch 中。

主题

- JITP 概览 (p. 780)
- 使用预调配模板注册 CA (p. 782)
- 使用预调配模板名称注册 CA (p. 783)

JITP 概览

当设备尝试使用由已注册 CA 证书签名的证书连接到 Amazon IoT 时，Amazon IoT 会从 CA 证书加载模板，并使用它来调用 RegisterThing (p. 801)。JITP 工作流首先将注册一个状态值为 PENDING_ACTIVATION 的证书。当设备预调配流程完成时，该证书的状态将更改为 ACTIVE。

Amazon IoT 定义了可在预调配模板中声明和引用的以下参数：

- AWS::IoT::Certificate::Country
- AWS::IoT::Certificate::Organization
- AWS::IoT::Certificate::OrganizationalUnit
- AWS::IoT::Certificate::DistinguishedNameQualifier
- AWS::IoT::Certificate::StateName
- AWS::IoT::Certificate::CommonName
- AWS::IoT::Certificate::SerialNumber
- AWS::IoT::Certificate::Id

这些预调配模板参数的值被限制为 JITP 可从正在预调配的设备的证书的使用者字段中提取的内容。证书必须包含模板正文中所有参数的值。AWS::IoT::Certificate::Id 参数指内部生成的 ID，而不是证书包含的 ID。您可以使用 Amazon IoT 规则内的 principal() 函数获取此 ID 的值。

> **Note**
>
> 您可以使用 Amazon IoT Core 即时预调配（JITP）功能，而无需将设备首次连接到 Amazon IoT Core 时发送整个信任链。可以选择出示 CA 证书，但当设备连接到 Amazon IoT Core 时，需要发送服务器名称指示（SNI）扩展。

示例模板正文

以下 JSON 文件是完整 JITP 模板的示例模板正文。

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
            "Status":"ACTIVE"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect
\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
```

```
}
```

此示例模板声明了从证书中提取并在 Resources 部分中使用的 AWS::IoT::Certificate::CommonName、AWS::IoT::Certificate::SerialNumber、AWS::IoT::Certific 和 AWS::IoT::Certificate::Id 预调配参数的值。JITP 工作流随后将使用此模板执行以下操作：

- 注册一个证书并将其状态设置为 PENDING_ACTIVE。
- 创建一个事物资源。
- 创建一个策略资源。
- 将策略附加到证书。
- 将证书附加到事物。
- 将证书状态更新为 ACTIVE。

请注意，如果证书不具有 templateBody 的 Parameters 部分中提及的所有属性，则设备将会预调配失败。例如，如果 AWS::IoT::Certificate::Country 包含在模板中，但证书没有 Country 属性，设备预调配操作将会失败。

您还可以使用 CloudTrail 排查 JITP 模板的问题。有关 Amazon CloudWatch 中录入的指标的信息，请参阅设备预配置指标 (p. 410)。有关预调配模板的更多信息，请参阅预调配模板 (p. 784)。

> **Note**
>
> 在预调配过程中，即时预调配（JITP）调用其它 Amazon IoT 控制面板 API 操作。这些调用可能会超过为您账户设置的 Amazon IoT 节流配额，导致调用受到限制。如有必要，请联系 Amazon 客户支持来增大节流配额。

使用预调配模板注册 CA

要使用完整的预调配模板注册 CA，请执行以下步骤：

1. 将预调配模板和角色 ARN 信息（如下例所示）保存为 JSON 文件：

```
{
     "templateBody" : "{\r\n    \"Parameters\" : {\r\n
 \"AWS::IoT::Certificate::CommonName\": {\r\n            \"Type\": \"String\"\r\n
      },\r\n        \"AWS::IoT::Certificate::SerialNumber\": {\r\n            \"Type
\": \"String\"\r\n        },\r\n        \"AWS::IoT::Certificate::Country\": {\r\n
          \"Type\": \"String\"\r\n        },\r\n        \"AWS::IoT::Certificate::Id
\": {\r\n            \"Type\": \"String\"\r\n        }\r\n    },\r\n    \"Resources
\": {\r\n        \"thing\": {\r\n            \"Type\": \"AWS::IoT::Thing\",\r\n
          \"Properties\": {\r\n                \"ThingName\": {\r\n
       \"Ref\": \"AWS::IoT::Certificate::CommonName\"\r\n                },\r\n
             \"AttributePayload\": {\r\n                    \"version\": \"v1\",
\r\n                    \"serialNumber\": {\r\n                        \"Ref\":
 \"AWS::IoT::Certificate::SerialNumber\"\r\n                    }\r\n
     },\r\n                \"ThingTypeName\": \"lightBulb-versionA\",\r\n
        \"ThingGroups\": [\r\n                    \"v1-lightbulbs\",\r\n
           {\r\n                        \"Ref\": \"AWS::IoT::Certificate::Country
\"\r\n                    }\r\n                ]\r\n            },\r\n
 \"OverrideSettings\": {\r\n                \"AttributePayload\": \"MERGE\",\r\n
             \"ThingTypeName\": \"REPLACE\",\r\n                \"ThingGroups\":
 \"DO_NOTHING\"\r\n            }\r\n        },\r\n        \"certificate\": {\r\n
     \"Type\": \"AWS::IoT::Certificate\",\r\n            \"Properties\": {\r\n
       \"CertificateId\": {\r\n                    \"Ref\": \"AWS::IoT::Certificate::Id
\"\r\n                },\r\n                \"Status\": \"ACTIVE\"\r\n            },
\r\n            \"OverrideSettings\": {\r\n                \"Status\": \"DO_NOTHING
\"\r\n            }\r\n        },\r\n        \"policy\": {\r\n            \"Type
\": \"AWS::IoT::Policy\",\r\n            \"Properties\": {\r\n
 \"PolicyDocument\": \"{ \\\"Version\\\": \\\"2012-10-17\\\", \\\"Statement\\\": [{ \\
\"Effect\\\": \\\"Allow\\\", \\\"Action\\\":[\\\"iot:Publish\\\"], \\\"Resource\\\": [\
```

```
\\"arn:aws:iot:us-east-1:123456789012:topic\/foo\/bar\\\"] }] }\"\r\n            }\r\n
      }\r\n    }\r\n}",
    "roleArn" : "arn:aws:iam::123456789012:role/JITPRole"
}
```

在本例中，templateBody 字段的值必须是指定为转义字符串的 JSON 对象，并且只能使用前述表 (p. 780)中的值。您可以使用各种工具来创建所需的 JSON 输出，例如 json.dumps (Python) 或 JSON.stringify（节点）。roleARN 字段的值必须是附加有 AWSIoTThingsRegistration 的角色的 ARN。此外，您的模板还可以在示例中使用现有PolicyName，而不是内联 PolicyDocument。

2. 使用 RegisterCACertificate API 操作或 register-ca-certificate CLI 命令注册 CA 证书。您将指定预调配模板的目录以及您在上一步中保存的角色 ARN 信息：

   下面显示了如何使用 Amazon CLI 在 DEFAULT 模式中注册 CA 证书的示例：

   ```
   aws iot register-ca-certificate --ca-certificate file://your-ca-cert --verification-
   cert file://your-verification-cert
                  --set-as-active --allow-auto-registration --registration-config
    file://your-template
   ```

   下面显示了如何使用 SNI_ONLY 在 Amazon CLI 模式中注册 CA 证书的示例：

   ```
   aws iot register-ca-certificate --ca-certificate file://your-ca-cert --certificate-
   mode SNI_ONLY
                   --set-as-active --allow-auto-registration --registration-config
    file://your-template
   ```

   有关更多信息，请参阅注册 CA 证书。

3. （可选）使用 UpdateCACertificate API 操作或 update-ca-certificate CLI 命令更新 CA 证书的设置。

   下面显示了如何使用 Amazon CLI 更新 CA 证书的示例：

   ```
   aws iot update-ca-certificate --certificate-id caCertificateId
                  --new-auto-registration-status ENABLE --registration-config
    file://your-template
   ```

使用预调配模板名称注册 CA

要使用预调配模板名称注册 CA，请执行以下步骤：

1. 将预调配模板正文保存为 JSON 文件。您可以在示例模板正文 (p. 781)中找到一个示例模板正文。
2. 要创建预调配模板，请使用 CreateProvisioningTemplate API 或 create-provisioning-template CLI 命令：

   ```
   aws iot create-provisioning-template --template-name your-template-name \
          --template-body file://your-template-body.json --type JITP \
          --provisioning-role-arn arn:aws:iam::123456789012:role/test
   ```

   Note

   对于即时预调配（JITP），您必须在创建预调配模板时将模板类型指定为 JITP。有关模板类型的更多信息，请参阅《Amazon API 参考》中的 CreateProvisioningTemplate。

3. 要使用模板名称注册 CA，请使用 RegisterCACertificate API 或register-ca-certificate CLI 命令：

```
aws iot register-ca-certificate --ca-certificate file://your-ca-cert --verification-
cert file://your-verification-cert \
        --set-as-active --allow-auto-registration --registration-config
 templateName=your-template-name
```

批量注册

您可以使用 start-thing-registration-task 命令批量注册事物。此命令接受预调配模板、S3 桶名称、键名称以及允许访问 S3 桶中文件的角色 ARN。S3 存储桶中的文件包含用于替换模板中参数的值。该文件必须为以换行符分隔的 JSON 文件。每一行包含用于注册单个设备的所有参数值。例如：

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

以下批量注册相关的 API 操作可能很有用：

- ListThingRegistrationTasks：列出当前的批量事物预调配任务。
- DescribeThingRegistrationTask：提供有关特定批量事物注册任务的信息。
- StopThingRegistrationTask：停止批量事物注册任务。
- ListThingRegistrationTaskReports：用于检查批量事物注册任务的结果和/或故障。

Note

- 一次只能运行一个批量注册操作任务（每个账户）。
- 批量注册操作会调用其它 Amazon IoT 控制面板 API 操作。这些调用可能会超出您账户的 Amazon IoT 节流配额，导致节流错误。如有必要，请联系Amazon客户支持来增大 Amazon IoT 的节流配额。

预调配模板

预调配模板是一个 JSON 文档，该文档使用参数来描述设备与 Amazon IoT 交互时必须使用的资源。预调配模板包含两个部分：Parameters 和 Resources。Amazon IoT 中有两种类型的预调配模板。一个用于即时预调配（JITP）和批量注册，另一个用于实例集预调配。

主题

- 参数部分 (p. 784)
- 资源部分 (p. 785)
- 批量注册的模板示例 (p. 788)
- 即时预调配（JITP）的模板示例 (p. 789)
- 实例集预调配 (p. 790)

参数部分

Parameters 部分声明在 Resources 部分中使用的参数。每个参数声明一个名称、一个类型以及一个可选的默认值。在随模板传入的字典不包含参数的值时，会使用默认值。模板文档的 Parameters 部分类似于以下所示：

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    }
}
```

此模板正文片段声明四个参数：ThingName、SerialNumber、Location 和 CSR。所有这些参数均为 String 类型。Location 参数声明了默认值 "WA"。

资源部分

模板正文的 Resources 部分声明了设备与 Amazon IoT 通信所需的资源：事物、证书以及一个或多个 IoT 策略。每个资源指定一个逻辑名称、一个类型和一组属性。

您可以使用逻辑名称在模板的其它位置引用资源。

类型指定您所声明的资源的种类。有效类型为：

- AWS::IoT::Thing
- AWS::IoT::Certificate
- AWS::IoT::Policy

您指定的属性取决于所声明的资源的类型。

事物资源

事物资源使用以下属性进行声明：

- ThingName: 字符串。
- AttributePayload：可选。名称-值对的列表。
- ThingTypeName：可选。用于事物的关联事物类型的字符串。
- ThingGroups：可选。事物所属的组的列表。
- BillingGroup：可选。关联账单组名称的字符串。

证书资源

您可以通过以下方式之一指定证书：

- 证书签名请求 (CSR)。
- 现有设备证书的证书 ID。（仅证书 ID 可与实例集预调配模板一起使用。）
- 使用注册到 Amazon IoT 的 CA 证书创建的设备证书。如果您有多个 CA 证书注册到同一使用者字段，则还必须传入用于对设备证书进行签名的 CA 证书。

Note

当您在模板中声明证书时，请只使用这些方法之一。例如，如果您使用了 CSR，就不能同时指定证书 ID 或设备证书。有关更多信息，请参阅X.509 客户端证书 (p. 267)。

有关更多信息，请参阅X.509 证书概览 (p. 264)。

证书资源使用以下属性进行声明：

- `CertificateSigningRequest`: 字符串。
- `CertificateId`: 字符串。
- `CertificatePem`: 字符串。
- `CACertificatePem`: 字符串。
- `Status`：可选。字符串可以是 `ACTIVE` 或 `INACTIVE`。默认值为 ACTIVE。

示例：

- 使用 CSR 指定的证书：

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateSigningRequest": {"Ref" : "CSR"},
            "Status" : "ACTIVE"
        }
    }
}
```

- 使用现有证书 ID 指定的证书：

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CertificateId": {"Ref" : "CertificateId"}
        }
    }
}
```

- 使用现有证书 .pem 和 CA 证书 .pem 指定的证书：

```
{
    "certificate" : {
        "Type" : "AWS::IoT::Certificate",
        "Properties" : {
            "CACertificatePem": {"Ref" : "CACertificatePem"},
            "CertificatePem": {"Ref" : "CertificatePem"}
        }
    }
}
```

策略资源

策略资源使用以下属性之一进行声明：

- PolicyName：可选。字符串。默认值为策略文档的哈希值。PolicyName 只能参考 Amazon IoT 策略但不是 IAM policy。如果您使用现有的 Amazon IoT 策略，则对于 PolicyName 属性，输入策略的名称。请勿包含 PolicyDocument 属性。
- PolicyDocument：可选。指定为转义字符串的 JSON 对象。如果未提供 PolicyDocument，则必须已经创建了策略。

> **Note**
>
> 如果存在 Policy 部分，则必须指定 PolicyName 或 PolicyDocument，但不能同时指定。

覆盖设置

如果模板指定了已经存在的资源，则使用 OverrideSettings 部分可以指定要采取的操作：

DO_NOTHING

将资源保留为原样。

REPLACE

使用在模板中指定的资源替换该资源。

FAIL

导致请求失败，出现 ResourceConflictsException。

MERGE

仅对 ThingGroups 的 AttributePayload 和 thing 属性有效。将事物的现有属性或组成员资格与模板中指定的同等内容合并。

当您声明事物资源时，您可以为以下属性指定 OverrideSettings：

- ATTRIBUTE_PAYLOAD
- THING_TYPE_NAME
- THING_GROUPS

当您声明证书资源时，您可以为 OverrideSettings 属性指定 Status。

OverrideSettings 不可用于策略资源。

资源示例

以下模板片段声明了一个事物、一个证书和一个策略：

```
{
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
```

```
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\", \"Statement\":
 [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

使用以下项声明事物：

- 逻辑名称 "thing"。
- 类型 AWS::IoT::Thing。
- 一组事物属性。

  事物属性包括事物名称、一组属性、一个事物类型名称 (可选) 以及事物所属的事物组列表 (可选)。

使用 {"Ref":"*parameter-name*"} 引用参数。评估模板时，使用随模板传入的字典中提供的参数值来替换参数。

使用以下项声明证书：

- 逻辑名称 "certificate"。
- 类型 AWS::IoT::Certificate。
- 一组属性。

  属性包括证书的 CSR 并将状态设置为 ACTIVE。CSR 文本作为随模板传入的字典中的参数传递。

使用以下项声明策略：

- 逻辑名称 "policy"。
- 类型 AWS::IoT::Policy。
- 现有策略的名称或策略文档的名称。

批量注册的模板示例

以下 JSON 文件是使用 CSR 指定证书的完整预调配模板的一个示例：

(PolicyDocument 字段值必须是指定为转义字符串的 JSON 对象。)

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
        "SerialNumber" : {
```

```
            "Type" : "String"
        },
        "Location" : {
            "Type" : "String",
            "Default" : "WA"
        },
        "CSR" : {
            "Type" : "String"
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "ThingName" : {"Ref" : "ThingName"},
                "AttributePayload" : { "version" : "v1", "serialNumber" :  {"Ref" :
 "SerialNumber"}},
                "ThingTypeName" :  "lightBulb-versionA",
                "ThingGroups" : ["v1-lightbulbs", {"Ref" : "Location"}]
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateSigningRequest": {"Ref" : "CSR"},
                "Status" : "ACTIVE"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : "{ \"Version\": \"2012-10-17\", \"Statement\":
 [{ \"Effect\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
east-1:123456789012:topic/foo/bar\"] }] }"
            }
        }
    }
}
```

即时预调配（JITP）的模板示例

以下 JSON 文件是使用证书 ID 指定现有证书的完整预调配模板的一个示例：

```
{
   "Parameters":{
      "AWS::IoT::Certificate::CommonName":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::SerialNumber":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Country":{
         "Type":"String"
      },
      "AWS::IoT::Certificate::Id":{
         "Type":"String"
      }
   },
   "Resources":{
      "thing":{
         "Type":"AWS::IoT::Thing",
         "Properties":{
            "ThingName":{
               "Ref":"AWS::IoT::Certificate::CommonName"
```

```
            },
            "AttributePayload":{
               "version":"v1",
               "serialNumber":{
                  "Ref":"AWS::IoT::Certificate::SerialNumber"
               }
            },
            "ThingTypeName":"lightBulb-versionA",
            "ThingGroups":[
               "v1-lightbulbs",
               {
                  "Ref":"AWS::IoT::Certificate::Country"
               }
            ]
         },
         "OverrideSettings":{
            "AttributePayload":"MERGE",
            "ThingTypeName":"REPLACE",
            "ThingGroups":"DO_NOTHING"
         }
      },
      "certificate":{
         "Type":"AWS::IoT::Certificate",
         "Properties":{
            "CertificateId":{
               "Ref":"AWS::IoT::Certificate::Id"
            },
            "Status":"ACTIVE"
         }
      },
      "policy":{
         "Type":"AWS::IoT::Policy",
         "Properties":{
            "PolicyDocument":"{ \"Version\": \"2012-10-17\", \"Statement\": [{ \"Effect
\": \"Allow\", \"Action\":[\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
east-1:123456789012:topic/foo/bar\"] }] }"
         }
      }
   }
}
```

Important

您必须在用于 JIT 预调配的模板中使用 `CertificateId`。

有关预调配模板类型的更多信息，请参阅《Amazon API 参考》中的 `CreateProvisioningTemplate`。

有关如何使用该模板进行即时预调配的更多信息，请参阅：即时预调配。

实例集预调配

实例集预调配模板由 Amazon IoT 用于设置云和设备配置。这些模板使用与 JITP 和批量注册模板相同的参数和资源。有关更多信息，请参阅预调配模板 (p. 784)。实例集预调配模板可以包含一个 `Mapping` 部分和一个 `DeviceConfiguration` 部分。您可以在实例集预调配模板中使用内置函数来生成设备特定的配置。实例集预调配模板是命名的资源，由 ARN 标识（例如，`arn:aws:iot:us-west-2:1234568788:provisioningtemplate/`*`templateName`*）。

映像

可选的 `Mappings` 部分将密钥与对应的一组命名值相匹配。例如，如果您想根据Amazon区域设置值，您可以创建将Amazon Web Services 区域名称用作密钥且其中含有您想为每个特定区域指定的值的映射。您使用 `Fn::FindInMap` 内部函数来检索映射中的值。

您不得在 Mappings 部分包含参数、虚拟参数或调用内部函数。

设备配置

设备配置部分包含要在预调配时发送到设备的任意数据。例如：

```
{
    "DeviceConfiguration": {
        "Foo":"Bar"
    }
}
```

如果您使用 avaScript Object Notation (JSON) 负载格式向设备发送消息，Amazon IoT Core 会将此数据格式化为 JSON。如果您使用 Concise Binary Object Representation (CBOR) 负载格式，Amazon IoT Core 会将此数据格式化为 CBOR。DeviceConfiguration 部分不支持嵌套 JSON 对象。

内部函数

内部函数在预调配模板中除了 Mappings 部分以外的任何部分使用。

Fn::Join

将一组值附加到单值中，中间用特定分隔符隔开。如果分隔符为空字符串，则值连接在一起而不使用分隔符。

> Important
>
> Fn::Join 不受 the section called “策略资源” (p. 786) 支持。

Fn::Select

通过索引返回对象列表中的单个对象。

> Important
>
> Fn::Select 不会检查 null 值，或检查索引是否超出数组边界。这两种情况都会导致预调配错误，因此您应确保选择了一个有效的索引值，并且列表中包含非空值。

Fn::FindInMap

返回与 Mappings 部分声明的双层映射中的键对应的值。

Fn::Split

将字符串拆分为字符串值列表，以便您可以从字符串列表中选择一个元素。您可以指定一个分隔符（如逗号），用于确定拆分字符串的位置。拆分字符串后，使用 Fn::Select 选择一个元素。

例如，如果将以逗号分隔的子网 ID 字符串导入您的堆栈模板，您可以在每个逗号的位置拆分字符串。从子网 ID 列表中，使用 Fn::Select 指定资源的子网 ID。

Fn::Sub

将输入字符串中的变量替换为您指定的值。您可以使用此函数构建命令或输出，其中包含在创建或更新堆栈之前不可用的值。

实例集预调配的模板示例

```
{
    "Parameters" : {
        "ThingName" : {
            "Type" : "String"
        },
```

```
        "SerialNumber": {
            "Type": "String"
        },
        "DeviceLocation": {
            "Type": "String"
        }
    },
    "Mappings": {
        "LocationTable": {
            "Seattle": {
                "LocationUrl": "https://example.aws"
            }
        }
    },
    "Resources" : {
        "thing" : {
            "Type" : "AWS::IoT::Thing",
            "Properties" : {
                "AttributePayload" : {
                    "version" : "v1",
                    "serialNumber" : "serialNumber"
                },
                "ThingName" : {"Ref" : "ThingName"},
                "ThingTypeName" : {"Fn::Join":["",["ThingPrefix_",
{"Ref":"SerialNumber"}]]},
                "ThingGroups" : ["v1-lightbulbs", "WA"],
                "BillingGroup": "LightBulbBillingGroup"
            },
            "OverrideSettings" : {
                "AttributePayload" : "MERGE",
                "ThingTypeName" : "REPLACE",
                "ThingGroups" : "DO_NOTHING"
            }
        },
        "certificate" : {
            "Type" : "AWS::IoT::Certificate",
            "Properties" : {
                "CertificateId": {"Ref": "AWS::IoT::Certificate::Id"},
                "Status" : "Active"
            }
        },
        "policy" : {
            "Type" : "AWS::IoT::Policy",
            "Properties" : {
                "PolicyDocument" : {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Effect": "Allow",
                        "Action":["iot:Publish"],
                        "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/foo/bar"]
                    }]
                }
            }
        }
    },
    "DeviceConfiguration": {
        "FallbackUrl": "https://www.example.com/test-site",
        "LocationUrl": {
            "Fn::FindInMap": ["LocationTable",{"Ref": "DeviceLocation"}, "LocationUrl"]}
        }
}
```

Note

可以对现有预调配模板进行更新以添加预先预调配挂钩 (p. 793)。

预先预调配挂钩

Amazon 建议在创建预调配模板时使用预调配挂钩函数，以便更好地控制将哪些设备以及多少设备的信息载入您的账户。预调配挂钩是 Lambda 函数，可在验证设备传递的参数后允许预调配设备。在您预调配设备之前，此 Lambda 函数必须存在于您的账户中，因为每次设备通过 the section called "RegisterThing" (p. 801) 发送请求时都会调用该函数。

> **Important**
>
> 请务必在附加到 Lambda 操作策略的全局条件上下文键中包含 `source-arn` 或 `source-account`，以便防范权限操纵。有关此问题的更多信息，请参阅 防止跨服务混淆代理 (p. 308)。

对于要预调配的设备，Lambda 函数必须接受输入对象并返回本节所述的输出对象。仅当 Lambda 函数返回带有 `"allowProvisioning": True` 的对象时，预调配才会继续。

预先预调配挂钩输入

在设备注册到 Amazon IoT 时，Amazon IoT 会将此对象发送到 Lambda 函数。

```
{
    "claimCertificateId" : "string",
    "certificateId" : "string",
    "certificatePem" : "string",
    "templateArn" : "arn:aws:iot:us-east-1:1234567890:provisioningtemplate/MyTemplate",
    "clientId" : "221a6d10-9c7f-42f1-9153-e52e6fc869c1",
    "parameters" : {
        "string" : "string",
        ...
    }
}
```

传递给 Lambda 函数的 `parameters` 对象包含 `parameters` 参数中的属性，该参数在 the section called "RegisterThing" (p. 801) 请求负载中传入。

预先预调配挂钩返回值

Lambda 函数必须返回一个响应，指示它是否已授权预调配请求以及要覆盖的任意属性的值。

以下是来自预先预调配函数的成功响应示例。

```
{
    "allowProvisioning": true,
    "parameterOverrides" : {
        "Key": "newCustomValue",
        ...
    }
}
```

`"parameterOverrides"` 值将添加到 the section called "RegisterThing" (p. 801) 申请负载的 `"parameters"` 参数中。

> **Note**
>
> - 如果 Lambda 函数失败，则预调配请求将失败并显示 `ACCESS_DENIED`，并且会在 CloudWatch Logs 中记录错误。
> - 如果 Lambda 函数未在响应中返回 `"allowProvisioning": "true"`，则预调配请求将失败并显示 `ACCESS_DENIED`。
> - Lambda 函数必须在 5 秒内完成执行并返回，否则预调配请求将失败。

预先预调配挂钩 Lambda 示例

Python

在 Python 中预先预调配挂钩 Lambda 示例。

```
import json

def pre_provisioning_hook(event, context):
    print(event)

    return {
        'allowProvisioning': True,
        'parameterOverrides': {
            'DeviceLocation': 'Seattle'
        }
    }
```

Java

在 Java 中预先预调配挂钩 Lambda 示例。

处理程序类：

```
package example;

import java.util.Map;
import java.util.HashMap;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class PreProvisioningHook implements RequestHandler<PreProvisioningHookRequest,
 PreProvisioningHookResponse> {

    public PreProvisioningHookResponse handleRequest(PreProvisioningHookRequest object,
 Context context) {
        Map<String, String> parameterOverrides = new HashMap<String, String>();
        parameterOverrides.put("DeviceLocation", "Seattle");

        PreProvisioningHookResponse response = PreProvisioningHookResponse.builder()
                .allowProvisioning(true)
                .parameterOverrides(parameterOverrides)
                .build();

        return response;
    }

}
```

请求类：

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@Builder
@AllArgsConstructor
```

```
@NoArgsConstructor
public class PreProvisioningHookRequest {
    private String claimCertificateId;
    private String certificateId;
    private String certificatePem;
    private String templateArn;
    private String clientId;
    private Map<String, String> parameters;
}
```

响应类：

```
package example;

import java.util.Map;
import lombok.Builder;
import lombok.Data;
import lombok.AllArgsConstructor;
import lombok.NoArgsConstructor;


@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class PreProvisioningHookResponse {
    private boolean allowProvisioning;
    private Map<String, String> parameterOverrides;
}
```

为安装设备的用户创建 IAM policy 和角色

Note

这些步骤仅在 Amazon IoT 控制台的指导下使用。
要从控制台转到此页面，请打开创建新的预调配模板。

为什么不能在 Amazon IoT 控制台上执行此操作？

为了获得最安全的体验，应在 IAM 控制台中执行 IAM 操作。本节中的过程将引导您完成创建相关 IAM 角色和策略的步骤，使用预调配模板需要这些 IAM 角色和策略。

为将安装设备的用户创建 IAM policy

此过程介绍如何创建授权用户使用预调配模板安装设备的 IAM policy。

执行此过程时，您将在 IAM 控制台和 Amazon IoT 控制台之间进行切换。我们建议您在完成此过程时同时打开这两个控制台。

为将安装设备的用户创建 IAM policy

1. 打开 IAM 控制台中的 Policies（策略）中心。
2. 请选择Create Policy（创建策略）。
3. 在创建策略页面上，选择 JSON 选项卡。
4. 切换到 Amazon IoT 控制台中您选择 Configure user policy and role（配置用户策略和角色）的页面。
5. 在 Sample provisioning policy（示例预调配策略）中，选择 Copy（复制）。
6. 切换回 IAM 控制台。

7. 在 JSON 编辑器中，粘贴您从 Amazon IoT 控制台复制的策略。此策略特定于您在 Amazon IoT 控制台中创建的模板。
8. 要继续，请选择下一步：标签。
9. 在 Add tags (Optional) [添加标签（可选）] 页面上，为要添加到此策略的每个标签选择 Add tag（添加标签）。如果没有任何标签要添加，您可以跳过该步骤。
10. 要继续，请选择下一步：审核。
11. 在查看策略页面上，执行以下操作：

    a. 对于 Name*（名称*），为策略输入可帮助您记住其作用的名称。

       记下您为该策略提供的名称，因为您将在下一步中使用该名称。

    b. 您可以选择输入您创建的策略的可选描述。

    c. 查看本策略的其余部分及其标签。
12. 要完成创建新策略，请选择 Create policy（创建策略）。

创建新策略后，继续执行the section called “为将安装设备的用户创建 IAM 角色” (p. 796)，以创建要附加此策略的用户角色条目。

为将安装设备的用户创建 IAM 角色

以下步骤介绍如何创建一个 IAM 角色，以对将使用预调配模板安装设备的用户进行身份验证。

为将安装设备的用户创建 IAM policy

1. 打开 IAM 控制台中的 Role（角色）中心。
2. 选择 Create role（创建角色）。
3. 在 Select trusted entity（选择可信实体）中，选择可信实体的类型，您要向该可信实体授予对您正在创建的模板的访问权限。
4. 选择或输入您要向其授予访问权限的可信实体的标识，然后选择 Next（下一步）。
5. 在 Add permissions（添加权限）页面的 Permission policies（权限策略）中的搜索框中，输入您在上一个过程 (p. 795)中创建的策略的名称。
6. 对于策略列表，选择在上一个过程中创建的策略，然后选择 Next（下一步）。
7. 在 Name, review, and create（命名、查看和创建）部分中，执行以下操作：

    a. 对于 Role name（角色名称），键入有助于您记住此角色的作用的角色名称。

    b. 对于 Description（描述），您可以选择输入角色的可选描述。不需要执行此操作即可继续。

    c. 查看 Step 1（步骤 1）和 Step 2（步骤 2）中的值。

    d. 对于 Add tags (Optional) [添加标签（可选）]，可以选择向该角色添加标签。不需要执行此操作即可继续。

    e. 确保该页面上的信息完整且正确无误，然后选择 Create role（创建角色）。

在创建新的角色后，返回到 Amazon IoT 控制台以继续创建模板。

更新现有策略以向新模板授权

以下步骤介绍如何向 IAM policy 添加新模板，此策略授权用户使用预调配模板安装设备。

向现有 IAM policy 添加新的模板

1. 打开 IAM 控制台中的 Policies（策略）中心。
2. 在搜索框中，输入要更新的策略的名称。

3. 在搜索框下方的列表中，找到要更新的策略并选择策略名称。
4. 对于 Policy summary（策略摘要），选择 JSON 选项卡（如果该面板尚不可见）。
5. 要编辑策略文档，请选择 Edit policy（编辑策略）。
6. 在编辑器中，选择 JSON 选项卡（如果该面板尚不可见）。
7. 在策略文档中，查找包含 `iot:CreateProvisioningClaim` 操作的策略语句。

   如果策略文档不包含带有 `iot:CreateProvisioningClaim` 操作的策略语句，请复制以下语句片段，并将其作为附加项粘贴到策略文档的 `Statement` 数组中。

   Note

   此片段必须放在 `Statement` 数组中的结束 `]` 字符之前。您可能需要在此片段之前或之后添加逗号，以更正任何语法错误。

```
{
    "Effect": "Allow",
    "Action": [
        "iot:CreateProvisioningClaim"
    ],
    "Resource": [
        "--PUT YOUR NEW TEMPLATE ARN HERE--"
    ]
}
```

8. 切换到 Amazon IoT 控制台中您选择 Modify user role permissions（修改用户角色权限）的页面。
9. 找到模板的 Resource ARN（资源 ARN）并选择 Copy（复制）。
10. 切换回 IAM 控制台。
11. 将复制的 Amazon 资源名称（ARN）粘贴到 `Statement` 数组中模板 ARN 列表的顶部，使其成为第一个条目。

    如果这是数组中唯一的 ARN，请删除刚粘贴的值末尾的逗号。
12. 查看更新的策略语句，并更正编辑器指示的任何错误。
13. 要保存更新的策略文档，请选择 Review policy（查看策略）。
14. 查看策略，然后选择 Save changes（保存更改）。
15. 返回到 Amazon IoT 控制台。

设备预调配 MQTT API

实例集预调配服务支持这些 MQTT API 操作：

- the section called “CreateCertificateFromCsr” (p. 798)
- the section called “CreateKeysAndCertificate” (p. 799)
- the section called “RegisterThing” (p. 801)

此 API 支持简洁二进制对象表示法 (CBOR) 格式和 JavaScript 对象表示法 (JSON) 格式的响应缓冲区，具体取决于主题的 *`payload-format`*。但是，为了清楚起见，本节中的响应和请求示例以 JSON 格式显示。

`payload-format`	响应格式数据类型
cbor	简洁二进制对象表示法 (CBOR)
json	JavaScript 对象表示法 (JSON)

Important

在发布请求消息主题之前，请订阅响应主题以接收响应。此 API 使用的消息使用 MQTT 的发布/订阅协议来提供请求和响应交互。
如果您未在发布请求之前 订阅响应主题，则无法收到该请求的结果。

CreateCertificateFromCsr

从证书签名请求 (CSR) 创建证书。Amazon IoT 提供由 Amazon Root 证书颁发机构 (CA) 签发的客户端证书。新证书的状态为 PENDING_ACTIVATION。当您调用 RegisterThing 使用此证书预调配事物时，证书状态将如模板中所述，更改为 ACTIVE 或 INACTIVE。

有关使用证书颁发机构证书和证书签名请求来创建客户端证书的更多信息，请参阅使用您的 CA 证书创建客户端证书 (p. 276)。

Note

为了安全起见，CreateCertificateFromCsr (p. 798) 返回的 certificateOwnershipToken 会在一小时后过期。必须在 certificateOwnershipToken 过期前调用 RegisterThing (p. 801)。如果由 CreateCertificateFromCsr (p. 798) 创建的证书尚未激活，并且在令牌过期时尚未附加到策略或事物，则该证书将被删除。如果令牌过期，设备可以调用 CreateCertificateFromCsr (p. 798) 以生成新证书。

CreateCertificateFromCsr 请求

发布包含 $aws/certificates/create-from-csr/*payload-format* 主题的消息。

payload-format

消息负载格式为 cbor 或 json。

CreateCertificateFromCsr 请求负载

```
{
    "certificateSigningRequest": "string"
}
```

certificateSigningRequest

CSR，采用 PEM 格式。

CreateCertificateFromCsr 响应

订阅 $aws/certificates/create-from-csr/*payload-format*/accepted。

payload-format

消息负载格式为 cbor 或 json。

CreateCertificateFromCsr 响应负载

```
{
    "certificateOwnershipToken": "string",
    "certificateId": "string",
    "certificatePem": "string"
```

```
}
```

certificateOwnershipToken

在预调配期间证明证书所有权的令牌。

certificateId

证书的 ID。证书管理操作仅接受 certificateId。

certificatePem

PEM 格式的证书数据。

CreateCertificateFromCsr 错误

要接收错误响应，请订阅 $aws/certificates/create-from-csr/*payload-format*/rejected。

payload-format

消息负载格式为 cbor 或 json。

CreateCertificateFromCsr 错误负载

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

statusCode

状态代码。

errorCode

错误代码。

errorMessage

错误消息。

CreateKeysAndCertificate

创建新的密钥和证书。Amazon IoT 提供由 Amazon Root 证书颁发机构 (CA) 签发的客户端证书。新证书的状态为 PENDING_ACTIVATION。当您调用 RegisterThing 使用此证书预调配事物时，证书状态将如模板中所述，更改为 ACTIVE 或 INACTIVE。

Note

为了安全起见，CreateKeysAndCertificate (p. 799) 返回的 certificateOwnershipToken 会在一小时后过期。必须在 certificateOwnershipToken 过期前调用 RegisterThing (p. 801)。如果由 CreateKeysAndCertificate (p. 799) 创建的证书尚未激活，并且在令牌过期时尚未附加到策略或事物，则该证书将被删除。如果令牌过期，设备可以调用 CreateKeysAndCertificate (p. 799) 以生成新证书。

CreateKeysAndCertificate 请求

在 $aws/certificates/create/*payload-format* 上发布带有空消息负载的消息。

payload-format

消息负载格式为 cbor 或 json。

CreateKeysAndCertificate 响应

订阅 $aws/certificates/create/*payload-format*/accepted。

payload-format

消息负载格式为 cbor 或 json。

CreateKeysAndCertificate 响应

```
{
    "certificateId": "string",
    "certificatePem": "string",
    "privateKey": "string",
    "certificateOwnershipToken": "string"
}
```

certificateId

证书 ID。

certificatePem

PEM 格式的证书数据。

privateKey

私有密钥。

certificateOwnershipToken

在预调配期间证明证书所有权的令牌。

CreateKeysAndCertificate 错误

要接收错误响应，请订阅 $aws/certificates/create/*payload-format*/rejected。

payload-format

消息负载格式为 cbor 或 json。

CreateKeysAndCertificate 错误负载

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

statusCode

状态代码。

`errorCode`

错误代码。

`errorMessage`

错误消息。

RegisterThing

使用预定义的模板预调配事物。

RegisterThing 请求

在 `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`* 上发布消息。

`payload-format`

消息负载格式为 `cbor` 或 `json`。

`templateName`

预调配模板名称。

RegisterThing 请求负载

```
{
    "certificateOwnershipToken": "string",
    "parameters": {
        "string": "string",
        ...
    }
}
```

`certificateOwnershipToken`

证明证书所有权的令牌。令牌在您通过 MQTT 创建证书时由 Amazon IoT 时生成。

`parameters`

可选。来自设备的键/值对由预先预调配挂钩 (p. 793)用于评估注册请求。

RegisterThing 响应

订阅 `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/accepted`。

`payload-format`

消息负载格式为 `cbor` 或 `json`。

`templateName`

预调配模板名称。

RegisterThing 响应负载

```
{
```

```
    "deviceConfiguration": {
        "string": "string",
        ...
    },
    "thingName": "string"
}
```

`deviceConfiguration`

在模板中定义的设备配置。

`thingName`

预调配期间创建的 IoT 事物的名称。

RegisterThing 错误响应

要接收错误响应，请订阅 `$aws/provisioning-templates/`*`templateName`*`/provision/`*`payload-format`*`/rejected`。

`payload-format`

消息负载格式为 `cbor` 或 `json`。

`templateName`

预调配模板名称。

RegisterThing 错误响应负载

```
{
    "statusCode": int,
    "errorCode": "string",
    "errorMessage": "string"
}
```

`statusCode`

状态代码。

`errorCode`

错误代码。

`errorMessage`

错误消息。

机群索引

您可以使用机群索引从以下来源对设备数据进行索引、搜索和聚合：Amazon IoT 注册处 (p. 239)、Amazon IoT Device Shadow (p. 593)、Amazon IoT 连接 (p. 1102)和 Amazon IoT Device Defender (p. 843) 违规。您可以查询一组设备，并根据设备属性的不同组合（包括状态、连接性和设备违规）聚合设备记录的统计信息。借助机群索引，您可以组织、调查设备机群并进行故障排除。

机群索引提供以下功能。

- 管理索引更新

您可以设置机群索引，将事物组、事物注册表、设备影子、设备连接和设备违规的更新索引。启用机群索引时，Amazon IoT 您的事物或事物组创建索引。AWS_Things 是您的所有事物创建的索引。AWS_ThingGroups 是包含所有事物组的索引。启用机群索引后，您可以在索引上运行查询（例如查找电池电量大于 70% 的所有手持设备）。Amazon IoT 使用您最新的数据保持索引持续更新。更多信息，请参阅管理机群索引 (p. 803)。

- 跨数据源搜索

您可以根据一种简单的查询语言 (p. 822)创建查询字符串，然后使用它在机群索引设置中对您配置的数据源进行搜索。查询字符串描述了您要查找的内容。有关支持机群索引的数据源的更多信息，请参阅管理事物索引 (p. 806)。

- 查询聚合数据

您可以在设备中搜索聚合数据，然后返回统计数据、百分位数、基数或与特定字段相关的搜索查询的内容列表。有关聚合查询的更多信息，请参阅查询聚合数据 (p. 816)。

- 使用机群指标监控聚合数据

您可以使用机群的指标自动向 CloudWatch，分析趋势并创建告警监控机群的聚合状态。有关机群指标的更多信息，请参阅机群指标 (p. 825)。

要使用机群索引，您必须设置机群索引配置。要设置实例集索引配置，您可以使用 Amazon IoT 控制台、Amazon SDK 或 Amazon Command Line Interface（Amazon CLI）。

有关定价和其它服务的信息，请参阅 Amazon IoT 设备管理定价。

管理机群索引

机群索引为您管理两种类型的索引，即事物索引和事物组索引。

事物索引

AWS_Things 是为您的所有事物创建的索引。事物索引支持以下数据源：Amazon IoT注册表 (p. 239)数据，Amazon IoTDevice Shadow (p. 593)DATA，Amazon IoT连接 (p. 1102)数据以及Amazon IoT Device Defender (p. 843)违规数据。通过将这些数据来源添加到实例集索引配置，您可以搜索事物、查询聚合数据、根据搜索查询创建动态事物组和实例集指标。

注册表-Amazon IoT提供了注册表来帮助您管理事物。您可以将注册表数据添加到实例集索引配置，以根据事物名称、描述和其他注册表属性搜索设备。有关注册表的更多信息，请参阅如何使用注册表管理事物 (p. 239)。

影子-Amazon IoT Device Shadow 服务 (p. 593)提供了有助于存储设备状态数据的影子。事物索引支持经典未命名影子和命名影子。要为命名影子编制索引，请激活您的命名影子设置并在事物索引配置中指定影子名称。默认情况下，每个 Amazon 账户最多可以添加 10 个影子名称。要了解如何增加影子名称的数量限制，请参阅《Amazon 一般参考》中的 Amazon IoT Device Management 配额。

要为索引添加命名影子，请执行以下操作：

- 如果您使用 Amazon IoT 控制台，请打开 Thing indexing（事物索引），选择 Add named shadows（添加命名的影子），然后通过 Named shadow selection（命名的影子选择）添加您的影子名称。
- 如果您使用 Amazon Command Line Interface（Amazon CLI），请将 `namedShadowIndexingMode` 设置为 `ON`，并在 `IndexingFilter` 中指定影子名称。要查看 CLI 命令示例，请参阅管理事物索引 (p. 806)。

  Important

  Amazon IoT Device Management 的 2022 年 7 月 20 日公开发行版的实例集索引与 Amazon IoT Core 命名影子和 Amazon IoT Device Defender 检测违规集成。在此 GA 版中，您可以通过指定影子名称为特定的命名影子编制索引。如果您在 2021 年 11 月 30 日至 2022 年 7 月 19 日此功能的公开预览期添加了命名影子以编制索引，我们鼓励您重新配置实例集索引设置并选择特定的影子名称，以降低索引成本并优化性能。

有关影子的更多信息，请参阅 Amazon IoTDevice Shadow 服务 (p. 593)。

连接-设备连接数据可帮助您识别设备的连接状态。此连接数据由生命期事件 (p. 1102)驱动。当客户端连接或断开连接时，Amazon IoT 向 MQTT 主题发布带消息的生命周期事件。连接或断开连接消息可以是提供连接状态详细信息的 JSON 元素的列表。有关设备连接的更多信息，请参阅生命周期事件 (p. 1102)。

Device Defender 违规-Amazon IoT Device Defender 违规数据有助于根据您在安全配置文件中定义的正常行为识别设备的异常行为。安全配置文件包含一组目标设备行为。每种行为都使用一个指标来指定设备的正常行为。有关 Device Defender 违规的更多信息，请参阅Amazon IoT Device Defender检测 (p. 947)。

有关更多信息，请参阅管理事物索引 (p. 806)。

事物组索引

`AWS_ThingGroups` 是包含所有事物组的索引。您可以使用此索引基于组名称、描述、属性和所有父组名称搜索事物组。

有关更多信息，请参阅管理事物组索引 (p. 815)。

托管字段

托管字段包含与事物、事物组、设备影子、设备连接和 Device Defender 违规关联的数据。Amazon IoT 定义了托管字段中的数据类型。您可以在创建 IoT 事物时指定每个托管字段的值。例如，事物名称、事物组和事物描述都是托管字段。机群索引服务根据您指定的索引模式为托管字段建立索引。托管字段无法更改或显示在 `customFields` 中。有关更多信息，请参阅自定义字段 (p. 806)。

以下列出了用于事物索引的托管字段：

- 注册表的托管字段

```
"managedFields" : [
  {name:thingId, type:String},
  {name:thingName, type:String},
  {name:registry.version, type:Number},
  {name:registry.thingTypeName, type:String},
```

```
  {name:registry.thingGroupNames, type:String},
]
```

- 经典未命名影子的托管字段

```
"managedFields" : [
  {name:shadow.version, type:Number},
  {name:shadow.hasDelta, type:Boolean}
]
```

- 命名影子的托管字段

```
"managedFields" : [
  {name:shadow.name.shadowName.version, type:Number},
  {name:shadow.name.shadowName.hasDelta, type:Boolean}
]
```

- 事物连接的托管字段

```
"managedFields" : [
  {name:connectivity.timestamp, type:Number},
  {name:connectivity.version, type:Number},
  {name:connectivity.connected, type:Boolean},
  {name:connectivity.disconnectReason, type:String}
]
```

- Device Defender 的托管字段

```
"managedFields" : [
  {name:deviceDefender.violationCount, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.metricName, type:String},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationTime, type:Number},
  {name:deviceDefender.securityprofile.behaviorname.lastViolationValue, type:String},
  {name:deviceDefender.securityprofile.behaviorname.inViolation, type:Boolean}
]
```

- 事物组的托管字段

```
"managedFields" : [
  {name:description, type:String},
  {name:parentGroupNames, type:String},
  {name:thingGroupId, type:String},
  {name:thingGroupName, type:String},
  {name:version, type:Number},
]
```

下表列出了不可搜索的托管字段。

数据源	无法搜索的托管字段
注册表	`registry.version`
未命名的影子	`shadow.version`
命名的影子	`shadow.name.*.version`
Device Defender	`deviceDefender.version`
事物组	`version`

自定义字段

您可以通过创建自定义字段来对事物属性、Device Shadow 数据和 Device Defender 违规数据进行汇总来进行索引。customFields 属性是字段名称和数据类型对的列表。您可以根据数据类型执行聚合查询。您选择的索引模式会影响可以在 customFields 中指定的字段。例如，如果您指定 REGISTRY 索引模式，则无法从事物影子中指定字段。您可以使用 update-indexing-configurationCLI 命令创建或更新自定义字段 (p. 808)。

- 自定义字段名称

事物和事物组属性的自定义字段名称以 attributes. 开头，后面是属性名称。如果启用了未命名的影子索引，事物可以有以 shadow.desired 或 shadow.reported 开头的自定义字段名称，然后是未命名的影子数据值名称。如果启用命名影子索引，事物可以有以 shadow.name.*.desired. 或 shadow.name.*.reported. 开头的自定义字段名称，然后是命名的影子数据值。如果 Device Defender 违规索引处于启用状态，则可以有以开头的自定义字段名称deviceDefender.，然后是 Device Defender 违规数据值。

前缀后面的属性或数据值名称只能包含字母数字、-（连字符）和 _（下划线）字符。不能有任何空格。

如果配置中的自定义字段与要为其建立索引的值类型不一致，机群索引将忽略聚合查询的不一致值。CloudWatch 日志对于排除聚合查询问题很有帮助。有关更多信息，请参阅队列索引服务的聚合查询疑难解答 (p. 1292)：

- 自定义字段类型

自定义字段类型具有以下支持的值：Number、String 和 Boolean。

管理事物索引

AWS_Things 是为您的所有事物创建的索引。您可以从以下数据源控制要索引的内容：Amazon IoT注册表 (p. 239)数据，Amazon IoTDevice Shadow (p. 593)数据，Amazon IoT连接 (p. 1102)数据以及Amazon IoT Device Defender (p. 843)违规数据。

启用事物索引

您可以使用 update-indexing-configurationCLI 命令或 UpdateIndexingConfigurationAPI 操作来创建AWS_Things索引并控制其配置。--thing-indexing-configuration (thingIndexingConfiguration) 参数允许您控制建立索引的数据类型（例如，注册表、影子和设备连接数据和Device Defender 违规数据）。

--thing-indexing-configuration 参数采用具有以下结构的字符串：

```
{
  "thingIndexingMode": "OFF"|"REGISTRY"|"REGISTRY_AND_SHADOW",
  "thingConnectivityIndexingMode": "OFF"|"STATUS",
  "deviceDefenderIndexingMode": "OFF"|"VIOLATIONS",
  "namedShadowIndexingMode": "OFF"|"ON",
  "managedFields": [
    {
      "name": "string",
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "customFields": [
    {
      "name": "string",
```

```
      "type": "Number"|"String"|"Boolean"
    },
    ...
  ],
  "filter": {
     "namedShadowNames": [ "string" ]
  }
}
```

事物索引模式

thingIndexingMode 属性控制建立索引的数据类型。

> **Important**
>
> 要启用事物索引，thingIndexingMode 不能将属性设置为 OFF。

属性	有效值	描述
thingIndexingMode	OFF	无索引。
	REGISTRY	为注册表数据建立索引。
	REGISTRY_AND_SHADOW	为注册表和事物影子数据建立索引。

thingConnectivityIndexingMode 属性指定是否为事物连接数据建立索引。

属性	有效值	描述
thingConnectivityIndexingMode	未指定。	不为事物连接数据建立索引。
	OFF	不为事物连接数据建立索引。
	STATUS	为事物连接数据建立索引。

deviceDefenderIndexingMode 属性指定是否为 Device Defender 违规数据建立索引。

属性	有效值	描述
deviceDefenderIndexingMode	未指定。	没有对 Device Defender 违规数据建立索引。
	OFF	没有对 Device Defender 违规数据建立索引。
	违规	对 Device Defender 违规数据建立索引。

namedShadowIndexingMode 属性指定是否为命名影子数据建立索引。

属性	有效值	描述
namedShadowIndexingMode	未指定。	没有对命名影子数据建立索引。

属性	有效值	描述
	OFF	没有对命名影子数据建立索引。
	开	对命名影子数据建立了索引。

Note

要选择要添加到实例集索引配置的命名影子，请将 namedShadowIndexingMode 设置为 ON并在 filter 中指定命名的影子名称。

托管字段和自定义字段

托管字段

托管字段包含与事物、事物组、设备影子、设备连接和 Device Defender 违规关联的数据。Amazon IoT 定义了托管字段中的数据类型。您可以在创建 IoT 事物时指定每个托管字段的值。例如，事物名称、事物组和事物描述都是托管字段。机群索引服务根据您指定的索引模式为托管字段建立索引。托管字段无法更改或显示在 customFields 中。

自定义字段

您可以通过创建自定义字段来对属性、Device Shadow 数据和 Device Defender 违规数据进行汇总来进行索引。customFields 属性是字段名称和数据类型对的列表。您可以根据数据类型执行聚合查询。您选择的索引模式会影响可以在 customFields 中指定的字段。例如，如果您指定 REGISTRY 索引模式，则无法从事物影子中指定字段。您可以使用 update-indexing-configurationCLI 命令创建或更新自定义字段 (p. 808)。有关更多信息，请参阅自定义字段 (p. 806)。

更新索引配置示例

您可以使用 Amazon IoT update-indexing-configuration CLI 命令更新索引配置。以下示例显示了如何使用 update-indexing-configuration 标签。

简短语法：

```
aws iot update-indexing-configuration --thing-indexing-configuration \
'thingIndexingMode=REGISTRY_AND_SHADOW,deviceDefenderIndexingMode=VIOLATIONS,namedShadowIndexingMode=ON
{name= shadow.name.thing1shadow.desired.DefaultDesired,
 type=String},{name=shadow.desired.power, type=Boolean},
 {name=deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number,
 type=Number}]'
```

JSON 语法：

```
aws iot update-indexing-configuration --cli-input-json \ '{
          "thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY_AND_SHADOW",
          "thingConnectivityIndexingMode": "STATUS",
          "deviceDefenderIndexingMode": "VIOLATIONS",
          "namedShadowIndexingMode": "ON",
          "filter": { "namedShadowNames": ["thing1shadow"]},
          "customFields": [ { "name": "shadow.desired.power", "type": "Boolean" },
          {"name": "attributes.version", "type": "Number"},
          {"name": "shadow.name.thing1shadow.desired.DefaultDesired", "type": "String"},
          {"name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number", "type":
 Number} ] } }'
```

此命令不会生成任何输出。

要检查事物索引状态，请运行 describe-index CLI 命令：

```
aws iot describe-index --index-name "AWS_Things"
```

describe-index 命令的输出如下所示：

```
{
    "indexName": "AWS_Things",
    "indexStatus": "ACTIVE",
    "schema": "MULTI_INDEXING_MODE"
}
```

Note

机群索引可能需要一点时间才能更新机群指数。我们建议 indexStatus 在使用之前显示 ACTIVE。根据配置的数据源，您可以在模式字段中具有不同的值。有关更多信息，请参阅描述事物索引 (p. 811)。

要获取事物索引配置详细信息，请运行 get-indexing-configuration CLI 命令：

```
aws iot get-indexing-configuration
```

get-indexing-configuration 命令的输出如下所示：

```
{
    "thingIndexingConfiguration": {
        "thingIndexingMode": "REGISTRY_AND_SHADOW",
        "thingConnectivityIndexingMode": "STATUS",
        "deviceDefenderIndexingMode": "VIOLATIONS",
        "namedShadowIndexingMode": "ON",
        "managedFields": [
            {
                "name": "connectivity.disconnectReason",
                "type": "String"
            },
            {
                "name": "registry.version",
                "type": "Number"
            },
            {
                "name": "thingName",
                "type": "String"
            },
            {
                "name": "deviceDefender.violationCount",
                "type": "Number"
            },
            {
                "name": "shadow.hasDelta",
                "type": "Boolean"
            },
            {
                "name": "shadow.name.*.version",
                "type": "Number"
            },
            {
                "name": "shadow.version",
                "type": "Number"
            },
            {
                "name": "connectivity.version",
                "type": "Number"
            },
            {
```

```
                "name": "connectivity.timestamp",
                "type": "Number"
            },
            {
                "name": "shadow.name.*.hasDelta",
                "type": "Boolean"
            },
            {
                "name": "registry.thingTypeName",
                "type": "String"
            },
            {
                "name": "thingId",
                "type": "String"
            },
            {
                "name": "connectivity.connected",
                "type": "Boolean"
            },
            {
                "name": "registry.thingGroupNames",
                "type": "String"
            }
        ],
        "customFields": [
            {
                "name": "shadow.name.thing1shadow.desired.DefaultDesired",
                "type": "String"
            },

            {
                "name":
 "deviceDefender.securityProfile1.NUMBER_VALUE_BEHAVIOR.lastViolationValue.number",
                "type": "Number"
            },
            {
                "name": "shadow.desired.power",
                "type": "Boolean"
            },
            {
                "name": "attributes.version",
                "type": "Number"
            }
        ],
        "filter": {
              "namedShadowNames": [
                  "thing1shadow"
              ]
          }
      },
    "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "OFF"
    }
}
```

要更新自定义字 update-indexing-configuration 段。示例如下：

```
aws iot update-indexing-configuration --thing-indexing-configuration

 'thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.version,type=Number},
{name=attributes.color,type=String},{name=shadow.desired.power,type=Boolean},
{name=shadow.desired.intensity,type=Number}]'
```

此命令已将 shadow.desired.intensity 添加到索引配置中。

> **Note**
>
> 更新自定义字段索引配置将覆盖所有现有的自定义字段。请确保在调用 update-indexing-configuration 时指定所有自定义字段。

重建索引后，您可以对新添加的字段、搜索注册表数据、影子数据和事物连接状态数据使用聚合查询。

更改索引模式时，请使用新的索引模式以确保所有自定义字段均有效。例如，如果从 REGISTRY_AND_SHADOW 模式开始使用名为 shadow.desired.temperature 的自定义字段，则必须先删除 shadow.desired.temperature 自定义字段，然后再将索引模式更改为 REGISTRY。如果您的索引配置包含未通过索引模式建立索引的自定义字段，则更新失败。

描述事物索引

以下命令说明了如何使用 describe-index CLI 命令来检索事物索引的当前状态。

```
aws iot describe-index --index-name "AWS_Things"
```

命令的响应如下所示：

```
{
    "indexName": "AWS_Things",
    "indexStatus": "BUILDING",
    "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"
}
```

首次启用机群索引时，Amazon IoT 将构建您的索引。如果 indexStatus 处于 BUILDING 状态，您无法查询索引。事物索引的 schema 指示将对什么类型的数据 (REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS) 建立索引。

更改索引的配置会导致重新生成索引。此过程中的 indexStatus 为 REBUILDING。在重建期间，您可以对事物索引中的数据运行查询。例如，如果您将索引配置从 REGISTRY 更改为 REGISTRY_AND_SHADOW，同时正在重新生成索引，则您可以查询注册表数据，包括最新的更新。但是，在重新生成操作完成之前，无法查询影子数据。生成或重新生成索引所需的时间量取决于数据量。

根据配置的数据源，您可以在模式字段中看到不同的值。下表显示了不同的模式值和相应的说明：

架构	描述
OFF	没有配置或索引数据源。
REGISTRY	对注册表数据建立索引。
REGISTRY_AND_SHADOW	对注册表数据和未命名（经典）影子数据建立索引。
REGISTRY_AND_CONNECTIVITY	为注册数据和连接数据建立索引。
REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS	对注册表数据、未命名（典型）影子数据和连接数据都建立了索引。
MULTI_INDEXING_MODE	除了注册表、未命名（经典）影子或连接数据之外，也对命名影子或 Device Defender 违规数据建立了索引。

查询事物索引

使用 search-index CLI 命令可查询索引中的数据。

```
aws iot search-index --index-name "AWS_Things" --query-string
          "thingName:mything*"
```


```
{
    "things":[{
         "thingName":"mything1",
         "thingGroupNames":[
            "mygroup1"
         ],
         "thingId":"a4b9f759-b0f2-4857-8a4b-967745ed9f4e",
         "attributes":{
            "attribute1":"abc"
         },
         "connectivity": {
            "connected":false,
            "timestamp":1556649874716,
            "disconnectReason": "CONNECTION_LOST"
         }
    },
    {
        "thingName":"mything2",
        "thingTypeName":"MyThingType",
        "thingGroupNames":[
            "mygroup1",
            "mygroup2"
        ],
        "thingId":"01014ef9-e97e-44c6-985a-d0b06924f2af",
        "attributes":{
            "model":"1.2",
            "country":"usa"
        },
        "shadow":{
            "desired":{
                "location":"new york",
                "myvalues":[3, 4, 5]
            },
            "reported":{
                "location":"new york",
                "myvalues":[1, 2, 3],
                "stats":{
                    "battery":78
                }
            },
            "metadata":{
                 "desired":{
                       "location":{
                            "timestamp":123456789
                        },
                       "myvalues":{
                            "timestamp":123456789
                       }
                  },
                  "reported":{
                        "location":{
                            "timestamp":34535454
                         },
                        "myvalues":{
                            "timestamp":34535454
                        },
                        "stats":{
                            "battery":{
                                 "timestamp":34535454
                            }
                        }
```

```
                }
            },
            "version":10,
            "timestamp":34535454
        },
        "connectivity": {
            "connected":true,
            "timestamp":1556649855046
        }
    }],
    "nextToken":"AQFCuvk7zZ3D9pOYMbFCeHbdZ+h=G"
}
```

在 JSON 响应中，"connectivity"（由 thingConnectivityIndexingMode=STATUS 设置启用）提供了一个布尔值、时间戳和 disconnectReason 来指示设备是否连接到 Amazon IoT Core。出于 CONNECTION_LOST 原因，设备 "mything1" 在 POSIX 时间 1556649874716 断开连接 (false)。有关断开原因的更多信息，请参阅生命周期事件 (p. 1102)。

```
"connectivity": {
    "connected":false,
    "timestamp":1556649874716,
    "disconnectReason": "CONNECTION_LOST"
}
```

设备 "mything2" 在 POSIX 时间 1556649855046 连接 (true)：

```
"connectivity": {
    "connected":true,
    "timestamp":1556649855046
}
```

时间戳以自纪元以来的毫秒给出，因此 1556649855046 表示 2019 年 4 月 30 日星期二上午 6:44:15.046 (UTC)。

> **Important**
>
> 如果设备已断开连接大约一小时，则连接状态的 "timestamp" 值和 "disconnectReason" 值可能会缺失。

限制和局限性

AWS_Things 具有这些限制。

具有复杂类型的影子字段

只有字段的值是简单类型、不包含数组的 JSON 对象或完全由简单类型组成的数组时，才会对影子字段编制索引。“简单类型”是指字符串、数字或文本 true 或 false 之一。例如，如果是以下影子状态，则系统将不会编制索引字段 "palette" 的值，因为它是一个包含复杂类型项目的数组。字段 "colors" 的值将会编制索引，因为该数组中的每个值都是一个字符串。

```
{
    "state": {
        "reported": {
            "switched": "ON",
            "colors": [ "RED", "GREEN", "BLUE" ],
            "palette": [
                {
                    "name": "RED",
                    "intensity": 124
                },
                {
```

```
                    "name": "GREEN",
                    "intensity": 68
                },
                {
                    "name": "BLUE",
                    "intensity": 201
                }
            ]
        }
    }
}
```

嵌套影子字段名称

嵌套影子字段的名称存储为以句点 (.) 分隔的字符串。例如，假设存在以下影子文档：

```
{
  "state": {
    "desired": {
      "one": {
        "two": {
          "three": "v2"
        }
      }
    }
  }
}
```

字段 three 的名称存储为 desired.one.two.three。如果您还有影子文档，存储方式如下：

```
{
  "state": {
    "desired": {
      "one.two.three": "v2"
    }
  }
}
```

都匹配 shadow.desired.one.two.three:v2 的查询。最佳实践是不要在影子字段名称中使用句点。

影子元数据

影子的元数据部分中的字段编制了索引，但仅当影子的 "state" 部分中的相应字段编制了索引时才会出现这种情况。（在之前的示例中，影子的元数据部分中的 "palette" 字段也不会编制索引。）

未注册的设备

实例集索引对设备的连接状态编制索引，该设备的连接 clientId 与注册表中已注册事物的 thingName 相同。

未注册的影子

如果您使用以下以下以下以下以下以下UpdateThingShadow以下以下以下以下以下以下

Null 值

未对 Null 值建立索引。

最大值

聚合查询的最大自定义字段数是 5。

聚合查询请求的最大百分位数是 100。

Authorization

您可以在 Amazon IoT 策略操作中将事物索引指定为 Amazon Resource Name (ARN)，如下所示。

操作	资源
iot:SearchIndex	索引 ARN（例如，arn:aws:iot:*your-aws-regionyour-aws-account*:index/AWS_Things）。
iot:DescribeIndex	索引 ARN（例如，arn:aws:iot:*your-aws-region*:index/AWS_Things）。

Note

如果您有权查询机群索引，可以访问整个机群的事物数据。

管理事物组索引

AWS_ThingGroups 是包含所有事物组的索引。您可以使用此索引基于组名称、描述、属性和所有父组名称搜索事物组。

启用事物组索引

您可以使用 UpdateIndexingConfigurationAPI 中的thing-group-indexing-configuration设置来创建AWS_ThingGroups索引并控制其配置。您可以使用 GetIndexingConfigurationAPI 检索当前的索引配置。

要更新对象组索引配置，请运行 update-indexing-configuration CLI 命令：

```
aws iot update-indexing-configuration --thing-group-indexing-configuration
 thingGroupIndexingMode=ON
```

您还可以在单个命令中更新事物和事物组索引的配置，如下所示：

```
aws iot update-indexing-configuration --thing-indexing-configuration
 thingIndexingMode=REGISTRY --thing-group-indexing-configuration thingGroupIndexingMode=ON
```

thingGroupIndexingMode 的有效值如下所示。

OFF

无索引/删除索引。

开

创建或配置 AWS_ThingGroups 索引。

要检索当前的事物和事物组索引配置，运行 get-indexing-configuration CLI 命令。

```
aws iot get-indexing-configuration
```

命令的响应如下所示：

```
{
   "thingGroupIndexingConfiguration": {
        "thingGroupIndexingMode": "ON"
    }
}
```

描述组索引

要检索 AWS_ThingGroups 索引的当前状态，使用describe-indexCLI 命令。

```
aws iot describe-index --index-name "AWS_ThingGroups"
```

命令的响应如下所示：

```
{
   "indexStatus": "ACTIVE",
   "indexName": "AWS_ThingGroups",
   "schema": "THING_GROUPS"
}
```

首次启用索引时，Amazon IoT 将构建您的索引。如果 indexStatus 为 BUILDING，则您无法查询索引。

查询事物组索引

要查询索引中的数据，使用 search-index CLI 命令：

```
aws iot search-index --index-name "AWS_ThingGroups" --query-string
 "thingGroupName:mythinggroup*"
```

Authorization

您可以指定事物组索引作为 Amazon IoT 策略操作中的资源 ARN，如下所示：

操作	资源
iot:SearchIndex	索引 ARN（例如，arn:aws:iot:*your-aws-region*:index/AWS_ThingGroups）。
iot:DescribeIndex	索引 ARN（例如，arn:aws:iot:*your-aws-region*:index/AWS_ThingGroups）。

查询聚合数据

Amazon IoT 提供了四个 API（GetStatistics、GetCardinality、GetPercentiles 和 GetBucketsAggregation），允许您搜索设备机群以获取聚合数据。

> **Note**
>
> 有关聚合API缺少或意外值的问题，请参阅机群索引故障排查指南 (p. 1292)。

GetStatistics

GetStatisticsAPI 和get-statistics CLI 命令返回指定聚合字段的计数、平均值、总和、最小值、最大值、平方和、方差和标准差。

get-statistics CLI 命令使用以下参数：

`index-name`

要搜索的索引的名称。默认值为 `AWS_Things`。

`query-string`

用于搜索索引的查询。您可以为查询指定 `"*"` 来获取您的Amazon Web Services 账户中所有已编制索引的事物的计数。

`aggregationField`

可选。要聚合的字段。此字段必须是在调用 update-indexing-configuration 时定义的托管字段或自定义字段。如果不指定聚合字段，则 `registry.version` 用作聚合字段。

`query-version`

要使用的查询的版本。默认值为 `2017-09-30`。

聚合字段的类型可能会影响返回的统计信息。

GetStatistics 带字符串值

如果在字符串字段上进行聚合，则调用 `GetStatistics` 会返回具有与查询匹配的属性的设备计数。例如：

```
aws iot get-statistics --aggregation-field 'attributes.stringAttribute'
           --query-string '*'
```

此命令会返回包含 `stringAttribute` 属性的设备数量：

```
{
  "statistics": {
    "count": 3
  }
}
```

GetStatistics 使用布尔值

当您使用布尔聚合字段调用 `GetStatistics` 时：

- AVERAGE 是匹配查询的设备的百分比。
- 根据以下规则，MINIMUM 为 0 或 1：
  - 如果聚合字段的所有值均为 `false`，则 MINIMUM 为 0。
  - 如果聚合字段的所有值均为 `true`，则 MINIMUM 为 1。
  - 如果聚合字段的值既有 `false` 又有 `true`，则 MINIMUM 为 0。
- 根据以下规则，MAXIMUM 为 0 或 1：
  - 如果聚合字段的所有值均为 `false`，则 MAXIMUM 为 0。
  - 如果聚合字段的所有值均为 `true`，则 MAXIMUM 为 1。
  - 如果聚合字段的值既有 `false` 又有 `true`，则 MAXIMUM 为 1。
- SUM 是等效于布尔值的整数的总和。
- COUNT 是符合查询字符串条件且包含有效聚合字段值的内容计数。

GetStatistics 用数值

调用 GetStatistics 并指定 Number 类型的聚合字段时，GetStatistics 返回以下值：

count

与查询字符串条件匹配并包含有效聚合字段值的事物的计数。

average

与查询匹配的数值的平均值。

sum

与查询匹配的数值的总和。

minimum

与查询匹配的数值中的最小值。

最大值

与查询匹配的数值中的最大值。

sumOfSquares

与查询匹配的数值的平方和。

variance

与查询匹配的数值的方差。一组值的方差是每个值与该组值平均值之间差值的平方的平均值。

stdDeviation

与查询匹配的数值的标准差。一组值的标准差用于衡量值的分布程度。

以下示例演示了如何使用数值自定义字段调用 get-statistics。

```
aws iot get-statistics --aggregation-field 'attributes.numericAttribute2'
            --query-string '*'
```

```
{
  "statistics": {
    "count": 3,
    "average": 33.333333333333336,
    "sum": 100.0,
    "minimum": -125.0,
    "maximum": 150.0,
    "sumOfSquares": 43750.0,
    "variance": 13472.22222222222,
    "stdDeviation": 116.06990230986766
  }
}
```

对于数值聚合字段，如果字段值超过最大双精度值，则统计值为空

GetCardinality

GetCardinalityAPI 和get-cardinality CLI 命令返回与查询相匹配的唯一值的大致数量。例如，您可能需要查找电池电量低于 50% 的设备数量：

```
aws iot get-cardinality --index-name AWS_Things --query-string "batterylevel
          > 50" --aggregation-field "shadow.reported.batterylevel"
```

此命令返回电池电量超过 50% 的事物数量：

```
{
    "cardinality": 100
}
```

即使没有匹配字段，get-cardinality 也始终返回 `cardinality`。例如：

```
aws iot get-cardinality --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_STR"
```

```
{
    "cardinality": 0
}
```

get-cardinality CLI 命令使用以下参数：

`index-name`

要搜索的索引的名称。默认值为 `AWS_Things`。

`query-string`

用于搜索索引的查询。您可以为查询指定 `"*"` 来获取您的Amazon Web Services 账户中所有已编制索引的事物的计数。

`aggregationField`

要聚合的字段。

`query-version`

要使用的查询的版本。默认值为 `2017-09-30`。

GetPercentiles

GetPercentilesAPI 和get-percentiles CLI 命令将与查询匹配的聚合值分组为百分位数分组。默认百分位数分组为：1,5,25,50,75,95,99，不过您可以在调用 `GetPercentiles` 时指定自己的百分位数分组。此函数为指定的每个百分位数组（或默认百分位数分组）返回一个值。百分位数组“1”包含在与查询匹配的值的大约 1% 中出现的聚合字段值。百分位数组“5”包含在与查询匹配的值的大约 5% 中出现的聚合字段值，以此类推。结果是近似值，与查询匹配的值越多，百分位数值就越准确。

以下示例演示了如何调用 get-percentiles CLI 命令。

```
aws iot get-percentiles --query-string "thingName:*" --aggregation-field
          "attributes.customField_NUM" --percents 10 20 30 40 50 60 70 80 90 99
```

```
{
    "percentiles": [
        {
            "value": 3.0,
            "percent": 80.0
        },
        {
            "value": 2.5999999999999996,
            "percent": 70.0
        },
        {
            "value": 3.0,
```

```
            "percent": 90.0
        },
        {
            "value": 2.0,
            "percent": 50.0
        },
        {
            "value": 2.0,
            "percent": 60.0
        },
        {
            "value": 1.0,
            "percent": 10.0
        },
        {
            "value": 2.0,
            "percent": 40.0
        },
        {
            "value": 1.0,
            "percent": 20.0
        },
        {
            "value": 1.4,
            "percent": 30.0
        },
        {
            "value": 3.0,
            "percent": 99.0
        }
    ]
}
```

以下命令显示没有匹配文档时从 get-percentiles 返回的输出。

```
aws iot get-percentiles --query-string "thingName:Non-existent*"
          --aggregation-field "attributes.customField_NUM"
```

```
{
    "percentiles": []
}
```

get-percentile CLI 命令使用以下参数：

`index-name`

要搜索的索引的名称。默认值为 `AWS_Things`。

`query-string`

用于搜索索引的查询。您可以为查询指定 `"*"` 来获取您的Amazon Web Services 账户中所有已编制索引的事物的计数。

`aggregationField`

要聚合的字段，必须为 `Number` 类型。

`query-version`

要使用的查询的版本。默认值为 `2017-09-30`。

`percents`

可选。您可以使用此参数指定自定义百分位数分组。

GetBucketsAggregation

GetBucketsAggregationAPI 和get-buckets-aggregation CLI 命令使用以下以下以下

操作	资源
	arn:aws:iot:*your-aws-region*:index/AWS_ThingGroups ）。

查询语法

在机群索引中，您可以使用查询语法来指定查询。

支持的功能

查询语法支持以下功能：

- 术语和短语
- 搜索字段
- 前缀搜索
- 范围搜索
- 布尔运算符 AND、OR、NOT 和 -。使用连字符从搜索结果中排除某事物（例如 thingName:(tv* AND -plasma)）。
- 分组
- 字段分组
- 对特殊字符转义（如，使用 \）

不支持的 功能

查询语法不支持以下功能：

- 前导通配符搜索（例如“*xyz”），搜索“*”将匹配所有事物
- 正则表达式
- 提升
- 排名
- 模糊搜索
- 近似搜索
- 排序
- 聚合

注意

关于查询语言需要注意的几点：

- 默认运算符为 AND。"thingName:abc thingType:xyz" 的查询等同于 "thingName:abc AND thingType:xyz"。
- 如果未指定字段， Amazon IoT 在所有注册表、Device Shadow 和 Device Defender 字段中搜索术语。
- 所有字段名称均区分大小写。
- 搜索不区分大小写。用空格字符分隔单词，如 Java 的 Character.isWhitespace(int) 中所定义的。
- Device Shadow 数据（未命名影子和命名影子）的索引包括“reported”、“desired”、“delta”和“metadata”部分。
- 设备影子和注册表版本不可搜索，但会在响应中提供。

- 查询中术语数量上限为 2 个。

事物查询示例

使用查询语法在查询字符串中指定查询。查询将传递给SearchIndexAPI。下表列出了一些查询字符串示例。

查询字符串	结果
abc	在任何注册表、影子（经典未命名影子和命名影子）或 Device Defender 违规字段中查询“abc”。
thingName：myThingName	查询名为 “myThingName” 的事物。
thingName:my*	查询名称以“my”开头的事物。
thingName:ab?	查询名称为“ab”加另外一个字符的事物（例如：“aba”、“abb”、“abc”等）。
thingTypeName:aa	查询与类型 aa 关联的事物。
attributes.myAttribute:75	查询属性名为“myAttribute”且属性值为 75 的事物。
attributes.myAttribute:[75 TO 80]	查询属性名为“myAttribute”，且属性值在数字范围（75-80，含 75 和 80）之内的事物。
attributes.myAttribute:{75 TO 80]	查询属性名为“myAttribute”，且属性值在数字范围（大于 75 且小于等于 80）之内的事物。
attributes.serialNumber:["abcd" TO "abcf"]	查询属性名为“serialNumber”，且属性值在字母数字字符串范围之内的事物。此查询将返回属性名为“serialNumber”，且其值为“abcd”、“abce”或“abcf”的事物。
attributes.myAttribute:i*t	查询属性名为“myAttribute”，且属性值为以“i”开头、以“t”结尾、中间有任意数量字符的事物。
attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10	使用布尔表达式组合术语来查询事物。此查询将返回具有下列特征的事物：属性名为“attr1”且值为“abc”；属性名为“attr2”且值小于 5；以及属性名为“attr3”且值不大于 10。
shadow.hasDelta:true	查询具有增量元素的未命名影子的事物。
NOT attributes.model:legacy	查询属性名为“model”且不是“legacy”的事物。
shadow.reported.stats.battery:{70 TO 100} (v2 OR v3) NOT attributes.model:legacy	查询具有以下特征的事物： • 事物的影子 `stats.battery` 属性具有介于 70 到 100 之间的值。 • 文本“v2”或“v3”出现在事物的名称、类型名称或属性值中。 • 事物的 `model` 属性未设置为“legacy”。
shadow.reported.myvalues:2	查询具有以下特征的事物：影子的“reported”部分中的 `myvalues` 数组包含值 2。

查询字符串	结果
shadow.reported.location:* NOT shadow.desired.stats.battery:*	查询具有以下特征的事物： • 影子的 `location` 部分中存在 `reported` 属性。 • 影子的 `desired` 部分中不存在 `stats.battery` 属性。
shadow.name.<shadowName>.hasDelta:true	查询具有给定名称和增量元素影子的东西。
shadow.name.<shadowName>.desired.filament:*	查询具有给定名称和所需灯丝特性的阴影的对象。
shadow.name.<shadowName>.reported.location:*	查询具有具有给定命名影子且 `location` 属性存在于命名影子的报告部分中的事物。
connectivity.connected:true	查询所有连接的设备。
connectivity.connected:false	查询所有断开连接的设备。
connectivity.connected:true AND connectivity.timestamp : [1557651600000 TO 1557867600000]	查询所有已连接且连接时间戳 >= 1557651600000 且 <= 1557867600000 的设备。时间戳以自纪元以来的毫秒数给出。
connectivity.connected:false AND connectivity.timestamp : [1557651600000 TO 1557867600000]	查询所有已断开连接且断开连接时间戳 >= 1557651600000 且 <= 1557867600000 的设备。时间戳以自纪元以来的毫秒数给出。
connectivity.connected:true AND connectivity.timestamp > 1557651600000	查询所有已连接且连接时间戳 > 1557651600000 的设备。时间戳以自纪元以来的毫秒数给出。
connectivity.connected:*	查询具有连接信息的所有设备。
connectivity.disconnectReason:*	查询具有连接断开原因的所有设备。
connectivity.disconnectReason:CLIENT_INITIATED_DISCONNECT	查询由于 CLIENT_INITATE_DISKINATE_DISKINTION 而断开的所有设备
deviceDefender.violationCount:[0 TO 100]	查询设备计数值在数字范围（0-100）内的 Device Defender 违规事物。
设备防御者。 <device-SecurityProfile >.disconnect Beavior.invional：	查询违反安全配置文件 `device-SecurityProfile` 中定义的行为 `disconnectBehavior` 的事物。请注意，inViolation:false 不是有效的查询。
设备防御者。 <device-SecurityProfile >.断开连接行为。 lastViolationValue.number>2	查询上次违规事件值大于 2 的安全配置文件设备中定义SecurityProfile 的行为`disconnectBehavior`。
设备防御者。 <device-SecurityProfile >.断开连接行为。 lastViolationTime>1634227200000	查询上次违规事件特定纪元时间后的安全配置文件设备。`disconnectBehaviorSecurityProfile`

事物组查询示例

系统在查询字符串中使用查询语法指定查询，并将其传送至 `SearchIndex` API。下表列出了一些查询字符串示例。

查询字符串	结果
abc	在所有字段中查询“abc”。
thingGroupNames:myGroupThing 姓名	查询名为 “myGroupThing名称” 的事物组。
thingGroupNames:我的*	查询名称以“my”开头的事物组。
thingGroupNames:ab ?	查询名称为“ab”加另外一个字符的事物组（例如：“aba”、“abb”、“abc”等。）
attributes.myAttribute:75	查询属性名为“myAttribute”且属性值为 75 的事物组。
attributes.myAttribute:[75 TO 80]	查询属性名为“myAttribute”，且属性值在数字范围（75-80，含 75 和 80）之内的事物组。
attributes.myAttribute:[75 TO 80]	查询属性名为“myAttribute”，且属性值在数字范围（大于 75 且小于等于 80）之内的事物组。
attributes.myAttribute:["abcd" TO "abcf"]	查询属性名为“myAttribute”，且属性值在字母数字字符串范围之内的事物组。此查询将返回属性名为“serialNumber”，且其值为“abcd”、“abce”或“abcf”的事物组。
attributes.myAttribute:i*t	查询属性名为“myAttribute”，且其值为以“i”开头、以“t”结尾、中间有任意数量字符的事物组。
attributes.attr1:abc AND attributes.attr2<5 NOT attributes.attr3>10	使用布尔表达式组合术语来查询事物组。此查询将返回具有下列特征的事物组：属性名为“attr1”且值为“abc”；属性名为“attr2”且值小于 5；以及属性名为“attr3”且值不大于 10。
NOT attributes.myAttribute:cde	查询属性名为“myAttribute”且不是“cde”的事物组。
parentGroupNames:(myParentThingGroupName)	查询父组名称与 “myParentThingGroupName” 匹配的事物组。
parentGroupNames:(myParentThingGroupName 或 myRootThingGroupName)	查询父组名称与 “” 或 “myParentThingGroupName” 匹配的事物组。myRootThingGroupName
parentGroupNames:(myParentThingGroupNa*)	查询父组名以 “myParentThingGroupNa” 开头的事物组。

机群指标

机群指标是机群索引 (p. 803)的一个功能，是一项允许您可以在 Amazon IoT 中索引、搜索和聚合设备数据的托管服务。. 借助机群指标，随着时间的推移，您可以在CloudWatch 中监控机群设备的聚合状态，包括查看您的机群设备的断开率或指定时间内的平均电池电量变化。

使用机群指标，您可以构建聚合查询 (p. 816)，他们的结果不断被发射到 CloudWatch 用作分析趋势和创建告警的指标。对于监控任务，您可以指定不同聚合类型的聚合查询（统计数据、基数和百分位)。您可以保存所有聚合查询创建机群指标供将来重复使用。

入门教程

在本教程中，您将创建一个机群指标 (p. 825)监控传感器的温度来检测潜在的异常。创建机群指标时，您需要定义检测温度超过 80 华氏度的传感器数量的聚合查询 (p. 816) 您指定查询每 60 秒运行一次，查询结果

将发送到 CloudWatch，您可以在其中查看具有潜在高温风险的传感器数量并设置告警。要完成此教程，需要使用 Amazon CLI：

在本教程中，您将学习如何：

- 设置 (p. 826)
- 创建机群指标 (p. 827)
- 查看 CloudWatch 中的指标 (p. 829)
- 清理资源 (p. 830)

完成本教程需要大约 15 分钟。

先决条件

- 安装 Amazon CLI 的最新版本
- 让自己熟悉查询聚合数据
- 让自己熟悉使用 Amazon CloudWatch 指标

设置

要使用机群指标，您应启用机群索引。要为具有指定数据源和相关配置的事物或事物组启用机群索引，请按照管理事物索引 (p. 806)和管理事物组索引 (p. 815)中的说明操作。

设置

1. 运行以下命令以启用机群索引并指定要搜索的数据源。

```
aws iot update-indexing-configuration \
--thing-indexing-configuration
 "thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.temperature,type=Number},
{name=attributes.rackId,type=String},
{name=attributes.stateNormal,type=Boolean}],thingConnectivityIndexingMode=STATUS" \
```

   上述的 CLI 命令示例启用了震源组索引，支持使用 AWS_Things 索引搜索注册表数据、阴影数据和对象连接状态。

   可能需要几分钟才能完成此配置更改。在创建机群指标之前，请确保已启用机群索引。

   要检查您的机群索引是否已启用，请运行以下 CLI 命令：

```
aws --region us-east-1 iot describe-index --index-name "AWS_Things"
```

   有关更多信息，请参阅 启用事物索引 (p. 806)。

2. 运行以下 bash 脚本创建十个事物并对其进行描述。

```
# Bash script. Type `bash` before running in other shells.

Temperatures=(70 71 72 73 74 75 47 97 98 99)
Racks=(Rack1 Rack1 Rack2 Rack2 Rack3 Rack4 Rack5 Rack6 Rack6 Rack6)
IsNormal=(true true true true true false false false false false)

for ((i=0; i < 10; i++))
do
```

```
  thing=$(aws iot create-thing --thing-name "TempSensor$i" --attribute-payload
 attributes="{temperature=${Temperatures[@]:$i:1},rackId=${Racks[@]:$i:1},stateNormal=
${IsNormal[@]:$i:1}}")
  aws iot describe-thing --thing-name "TempSensor$i"
done
```

这个脚本创建了十个事物来表示十个传感器。每个事物都有 temperature、rackId 和 stateNormal 属性如下表所述：

属性	数据类型	描述
temperature	数字	温度（单位“华氏”）
rackId	字符串	包含传感器的服务器机架 ID
stateNormal	布尔值	传感器的温度值是否正常

此脚本的输出包含十个 JSON 文件。其中一个 JSON 文件如下所示：

```
{
    "version": 1,
    "thingName": "TempSensor0",
    "defaultClientId": "TempSensor0",
    "attributes": {
        "rackId": "Rack1",
        "stateNormal": "true",
        "temperature": "70"
    },
    "thingArn": "arn:aws:iot:region:account:thing/TempSensor0",
    "thingId": "example-thing-id"
}
```

有关更多信息，请参阅创建事物。

创建机群指标

要创建机群指标

1. 运行以下命令创建名为*high_temp_FM*的机群指标：您可以创建机群指标来监控 CloudWatch 中温度超过 80 华氏度的传感器数量。

```
aws iot create-fleet-metric --metric-name "high_temp_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature >80" --period 60 --aggregation-field
 "attributes.temperature" --aggregation-type name=Statistics,values=count
```

指标名称

数据类型：字符串 --metric-name 参数指定机群指标名称。在此示例中，您正在创建名为 high_temp_FM的机群指标。

--查询-字符串

数据类型：字符串 --query-string参数指定查询字符串。在此示例中，查询字符串意味着查询名称以 TempSensor 开头的所有事物以及气温高于华氏 80 度的事物。有关更多信息，请参阅。查询语法 (p. 822)。

--时期

数据类型：整数 --period参数指定检索聚合数据的时间（以秒为单位）。在此示例中，您指定要创建的机群指标每 60 秒检索一次聚合数据。

--字段-聚合

数据类型：字符串 --aggregation-field 参数指定要评估的属性。在此示例中，要评估温度属性。

--聚合-类型

--aggregation-type 参数指定要在机群指标中显示的统计摘要。对于监控任务，您可以为不同聚合类型自定义聚合查询属性（统计数据、基数和百分位）。在本示例中，您将指定对于聚合类型统计数据的计数返回具有与查询匹配的属性的设备计数，换言之，返回名称以 temSensor 开头的设备计数且温度高于华氏 80 度的设备计数。有关更多信息，请参阅查询聚合数据 (p. 816)。

此命令的输出如下所示：

```
{
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "metricName": "high_temp_FM"
}
```

Note

数据点可能需要一点时间才能在 CloudWatch 中显示。

要了解有关如何创建机群指标的更多信息，请参阅管理机群指标 (p. 830)。

如果您无法创建机群指标，请阅读机群指标故障排查 (p. 1293)。

2. （可选）运行以下命令以描述名为 high_temp_FM 的机群指标：

```
aws iot describe-fleet-metric --metric-name "high_temp_FM"
```

此命令的输出如下所示：

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625249775.834,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
            "sum"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625249775.834,
    "metricName": "high_temp_FM"
}
```

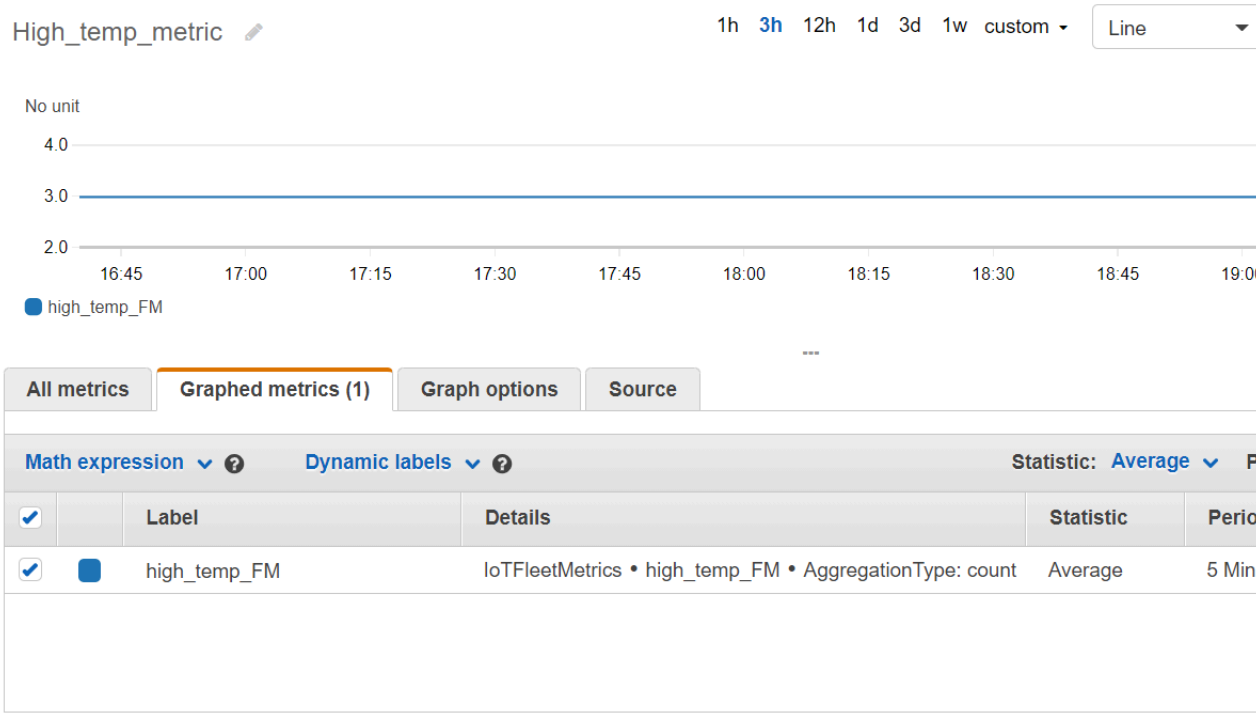
查看 CloudWatch 中的机群指标

创建机群指标后，您可以在 CloudWatch 控制台中查看指标数据。在本教程中，您将看到指标显示了以 temSensor 开头的传感器以及温度高于华氏 80 度的传感器数量。

在 CloudWatch 中查看数据点

1. 通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 在左侧的 CloudWatch 菜单上，选择 Metrics（指标）展开子菜单然后选择 All metrics（所有指标）。这将打开上半部分的页面显示图表，下半部分包含四个选项卡式部分。
3. 第一个选项卡式部分所有指标列出您可以在组中查看的所有指标，选择包含所有机群指标的 IotFleetMetrics。
4. 在所有指标选项卡上的聚合类型部分，选择 Aggregation type（聚合类型）查看您创建的所有机群指标。
5. 在 聚合类型左侧选择机群指标显示图片。您将在指标名称左侧看到值的 ##，这是您在本教程的创建机群指标 (p. 827)部分中指定的聚合类型值。
6. 选择所有指标选项卡右侧叫做图表化指标的第二个选项卡，查看从上一步中选择的机群指标。

   您能够看到一张图表，显示温度高于 80 华氏度的传感器数量，如下所示：



   Note

   CloudWatch 中的时期属性原定设置为 5 分钟。这是 CloudWatch 中数据点之间的时间间隔。您可以根据您的需求更改时期设置。

7. （可选）您可以设置指标告警。

   1. 在左侧 CloudWatch 菜单上，选择 Alarms（告警）展开子菜单然后选择 All alarms（所有告警）。
   2. 在 Alarms（告警）页面上，在右上角选择 Create alarm（创建告警）。按照控制台中 Create alarm（创建告警）的说明根据需要创建警报。有关更多信息，请参阅使用 Amazon CloudWatch 告警。

要了解更多信息，请阅读使用 Amazon CloudWatch 指标。

如果您在 CloudWatch 中看不到数据点，请阅读机群指标故障排除 (p. 1293)。

清除

要删除机群指标

您使用 delete-fleet-metric CLI 命令删除机群指标。

运行以下命令删除名为 high_temp_FM 的机群指标。

```
aws iot delete-fleet-metric --metric-name "high_temp_FM"
```

要清除事物

您可以使用 delete-thing CLI 命令删除事物。

运行以下脚本删除您创建的十个事物：

```
# Bash script. Type `bash` before running in other shells.

for ((i=0; i < 10; i++))
do
  thing=$(aws iot delete-thing --thing-name "TempSensor$i")
done
```

要清除 CloudWatch 中的指标

CloudWatch 不支持删除指标。指标根据保留时间表过期。要了解更多信息，请参阅使用 Amazon CloudWatch 指标。

管理机群指标

本主题介绍如何使用 Amazon IoT 控制台和 Amazon CLI 来管理您的机群指标。

管理机群指标 (控制台)

在创建机群指标之前，请确保您已使用关联的数据源和配置启用了机群索引。

启用机群索引

如果您已经启用了机群索引，请跳过此部分。

如果还没有启用机群索引，请按照以下说明操作。

1. 在 https://console.aws.amazon.com/iot/ 打开您的Amazon IoT 控制台。
2. 在 Amazon IoT 菜单上，选择 Settings (设置)。
3. 要查看详细设置，请在设置页面，向下滚动到机群索引部分。
4. 要更新您的机群索引设置，请在机群索引部分右侧选择 Manage indexing （管理索引）。
5. 在管理机群索引页面上，根据您的需求更新机群索引设置。
   - 配置

     要打开事物索引，请开启事物索引，然后选择要从中索引的数据源。

要启用事物组索引，请打开事物组索引。

- 聚合的自定义字段 - 可选

  自定义字段是字段名称和字段类型对的列表。

  要添加自定义字段对，请选择 Add new field（添加新字段）。输入自定义字段名称，如 `attributes.temperature`，然后从字段类型菜单选择一个字段类型。请注意，自定义以 `attributes.` 开头的字段名称并将保存为属性来运行事物聚合查询。

  要更新和保存设置，请选择 Update（更新）。

创建机群指标

1. 在 https://console.aws.amazon.com/iot/ 打开您的Amazon IoT 控制台。
2. 在 Amazon IoT 菜单中，选择Manage（管理），然后选择 Fleet metrics（机群指标）。
3. 在机群指标页面上，选择 Create fleet metric（创建机群指标）然后完成创建步骤。
4. 在步骤 1 中配置机群指标
   - 在查询部分中，输入查询字符串以指定要执行聚合搜索的事物或事物组。查询由属性和值组成的字符串。对于属性，选择所需的属性，或，如果该属性没有出现在列表中，则在字段中输入属性。在 : 之后输入值。一个查询字符串示例可以是 `thingName:TempSensor*`。对于输入的每个查询字符串，请按键盘上的输入。如果输入多个查询字符串，请通过选择and、or、and not 或, or or not 指定他们之间的关系。
   - 在报告属性中，从各自的列表中选择 Index name（索引名称）、Aggregation type（聚合类型）和 Aggregation field（聚合字段）。接下来，在选择数据中选择您想要聚合的数据，您可以在其中选择多个数据值。
   - 选择 Next (下一步)。
5. 在步骤 2 中指定机群指标属性
   - 在机群指标名称字段中，输入要创建机群指标的名称。
   - 在说明-可选在段中，输入要创建机群指标的描述。此字段为可选项。
   - 在小时和分钟字段中，输入您希望机群指标向 CloudWatch 发送数据的时间（频率）。
   - 选择 Next (下一步)。
6. 步骤 3：审核并创建
   - 查看步骤 1 和步骤 2 的设置。要编辑设置，请选择 Edit（编辑）。
   - 选择创建机群指标。

成功创建后，机群指标页面会列出机群指标。

更新机群指标

1. 在机群指标页面上，选择要更新的机群指标。
2. 在机群指标 Details(详细信息) 选项卡上，选择 Edit (编辑)。这将打开创建步骤，您可以在这三个步骤中的任何一个步骤中更新机群指标。
3. 完成更新机群指标后，选择 Update fleet metric（更新机群指标）。

删除机群指标

1. 在 Fleet metric（机群指标）页面上，选择要删除的机群。
2. 在显示机群指标详细信息的下一页上，选择 Delete（删除）。
3. 在对话框中，输入机群指标的名称确认删除。

4. 选择 Delete。此步骤将永久删除您的机群指标。

管理机群指标 (CLI)

以下部分介绍如何使用 Amazon CLI 来管理您的机群指标。在创建机群指标之前，请确保您已使用关联的数据源和配置启用了机群索引。要为您的事物或事物组启用机群索引，请按照管理事物索引 (p. 806)或管理事物组索引 (p. 815)中的说明操作。

创建机群指标

您可以使用创建机群指标 CLI 命令创建机群指标。

```
aws iot create-fleet-metric --metric-name "YourFleetMetricName" --query-string "*" --period
 60 --aggregation-field "registry.version" --aggregation-type name=Statistics,values=sum
```

此命令的输出包含机群指标的名称和 Amazon Resource Name (ARN)。输出如下所示：

```
{
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "metricName": "YourFleetMetricName"
}
```

列出机群指标

您可以使用列表机群指标 CLI 命令列出账户中的所有机群指标。

```
aws iot list-fleet-metrics
```

此命令的输出包含所有机群指标。输出如下所示：

```
{
    "fleetMetrics": [
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric1",
            "metricName": "YourFleetMetric1"
        },
        {
            "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/
YourFleetMetric2",
            "metricName": "YourFleetMetric2"
        }
    ]
}
```

描述机群指标

您可以使用描述船队指标 CLI 命令来显示有关机群指标的更详细信息。

```
aws iot describe-fleet-metric --metric-name "YourFleetMetricName"
```

此命令输出包含有关指定机群指标的详细信息。输出如下所示：

```
{
    "queryVersion": "2017-09-30",
```

```
    "lastModifiedDate": 1625790642.355,
    "queryString": "*",
    "period": 60,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 1,
    "aggregationType": {
        "values": [
            "sum"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625790642.355,
    "metricName": "YourFleetMetricName"
}
```

更新机群指标

您可以使用更新机群指标 CLI 命令更新机群指标。

```
aws iot update-fleet-metric --metric-name "YourFleetMetricName" --query-string
 "*" --period 120 --aggregation-field "registry.version" --aggregation-type
 name=Statistics,values=sum,count --index-name AWS_Things
```

更新机群指标命令不会产生任何输出。您可以使用描述机群指标 CLI 命令查看结果。

```
{
    "queryVersion": "2017-09-30",
    "lastModifiedDate": 1625792300.881,
    "queryString": "*",
    "period": 120,
    "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
    "aggregationField": "registry.version",
    "version": 2,
    "aggregationType": {
        "values": [
            "sum",
            "count"
        ],
        "name": "Statistics"
    },
    "indexName": "AWS_Things",
    "creationDate": 1625792300.881,
    "metricName": "YourFleetMetricName"
}
```

删除机群指标

您可以使用删除机群指标 CLI 命令删除机群指标。

```
aws iot delete-fleet-metric --metric-name "YourFleetMetricName"
```

如果删除成功或者您指定的机群指标不存在，此命令将不产生任何输出。

有关更多信息，请参阅机群指标故障排除 (p. 1293)。

基于 MQTT 的文件传输

基于 MQTT 的文件传输让您可以管理文件并将其传输到机群中的 Amazon IoT 设备上。借助 Amazon 云中的此功能，您可以创建包含多个文件的流，您可以更新流数据（文件列表和描述）、获取流数据等。Amazon IoT基于 MQTT 的文件传输，可以使用支持 JSON 或 CBOR 格式的请求和响应消息的 MQTT 协议，将数据以小数据块的形式，传输到您的 IoT 设备。

有关使用 Amazon IoT 在 IoT 设备之间传输数据方法的更多信息，请参阅 将设备连接到 Amazon IoT (p. 72)。

主题

- 什么是流？ (p. 834)
- 在Amazon云中管理流 (p. 835)
- 在设备中使用 Amazon IoT 基于 MQTT 的文件传输 (p. 836)
- FreeRTOS OTA 中的一个示例使用案例 (p. 842)

什么是流？

在 Amazon IoT 中，流是一种可公开寻址的资源，它是可以传输到 IoT 设备的文件列表的抽象化展示。通常而言，流包含以下信息：

- Amazon Resource Name (ARN)，在给定时间唯一标识流。此 ARN 具有模式 `arn:partition:iot:region:account-ID:stream/stream ID`。
- 流 ID，它将标识您的流，并且在 Amazon Command Line Interface (Amazon CLI) 或 SDK 命令中使用（且通常是必要的）。
- 流描述提供对流资源的说明。
- 流版本用来标识流的特定版本。由于流数据可以在设备开启数据传输之前立即修改，因此设备可以使用流版本来强制执行一致性检查。
- 一系列文件，可以传输到设备。对于列表中的每个文件，流都会记录文件 ID、文件大小以及文件的地址信息（例如，Amazon S3 存储桶名称、对象密钥和对象版本）。
- Amazon Identity and Access Management (IAM) 角色，授予 Amazon IoT 基于 MQTT 的文件传输读取存储在数据存储中的流文件的权限。

Amazon IoT 基于 MQTT 的文件传输提供了以下功能，以便设备可以从 Amazon 云传输数据：

- 使用 MQTT 协议进行数据传输。
- 支持 JSON 或 CBOR 格式。
- 通过描述流 (`DescribeStream` API) 来获取流文件列表、流版本和相关信息的能力。
- 能够在小数据块中发送数据 (`GetStream` API)，以便具有硬件约束的设备可以接收数据块的能力。
- 支持每个请求的动态数据块大小，以支持具有不同内存容量的设备。
- 当多个设备从同一流文件请求数据块时，对并发流请求的优化。
- Amazon S3 作为流文件的数据存储。
- Support 数据传输日志将消息从Amazon IoT基于 MQTT 的文件传输发布到 MQTT CloudWatch。

有关基于 MQTT 的文件传输配额，请参阅《Amazon 一般参考》中的 Amazon IoT Core 服务配额。

在Amazon云中管理流

Amazon IoT 提供的 Amazon SDK 和 Amazon CLI 命令，您可用来在 Amazon 云中管理流。您可以使用这些命令来进行以下操作：

- 创建流 CLI / SDK
- 描述流以获取其信息。CLI / SDK
- 列出您Amazon Web Services 账户中的流。CLI / SDK
- 更新流中的文件列表或流描述。CLI / SDK
- 删除流。CLI / SDK

> Note
>
> 此时，流在 Amazon Web Services Management Console 中不可见。您必须使用 Amazon CLI 或者 Amazon SDK 来管理 Amazon IoT 中的流。

在您使用设备上的 Amazon IoT 基于 MQTT 的文件传输前，您必须按照下面几个部分中的步骤操作，以确保您的设备已获得正确授权，并且可以连接到 Amazon IoT Device Gateway。

向您的设备授予权限

您可以按照创建 Amazon IoT 策略中的步骤创建设备策略，或使用现有设备策略。将策略附加到与您的设备关联的证书，并将以下权限添加到设备策略中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {   "Effect": "Allow",
            "Action": [ "iot:Connect" ],
            "Resource": [
                "arn:partition:iot:region:accountID:client/
${iot:Connection.Thing.ThingName}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [ "iot:Receive", "iot:Publish" ],
            "Resource": [
                "arn:partition:iot:region:accountID:topic/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:Subscribe",
            "Resource": [
                "arn:partition:iot:region:accountID:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/streams/*"
            ]
        }
    ]
}
```

将设备连接到 Amazon IoT

使用 Amazon IoT 基于 MQTT 的文件传输的设备需要连接到 Amazon IoT。Amazon IoT基于 MQTT 的文件传输与 Amazon 云中的 Amazon IoT 集成，因此您的设备应直接连接到 Amazon IoT 数据层面的端点。

Note

Amazon IoT 数据层面的端点特定于 Amazon Web Services 账户 和区域。您必须将终端节点用于 Amazon Web Services 账户 和您的设备在 Amazon IoT 中注册的区域。

参阅 连接到 Amazon IoT Core (p. 64) 了解更多信息。

在设备中使用 Amazon IoT 基于 MQTT 的文件传输

要启动数据传输过程，设备必须接收初始数据集，其中至少包含一个流 ID。您可以通过在任务文档中包含初始数据集来使用 任务 (p. 637) 计划设备的数据传输任务。当设备收到初始数据集时，它应该开始与 Amazon IoT 基于 MQTT 的文件传输进行交互。若要与 Amazon IoT 基于 MQTT 的文件传输交换数据，设备应：

- 使用 MQTT 协议订阅 基于 MQTT 的文件传输主题 (p. 106)。
- 发送请求，然后使用 MQTT 消息等待接收响应。

您可以选择在初始数据集中包含流文件 ID 和流版本。将流文件 ID 发送到设备可以简化设备固件/软件的编程，因为设备无需发送 `DescribeStream` 请求来获取此 ID。设备可以在 `GetStream` 请求中指定流版本以强制执行一致性检查，以防流意外更新。

使用 DescribeStream Dream 获取流数据

Amazon IoT 基于 MQTT 的文件传输提供 `DescribeStream` API 将流数据发送到设备。此 API 返回的流数据包括流 ID、流版本、流描述和流文件列表，每个流文件都有一个文件 ID 和文件大小信息（以字节为单位）。通过这些信息，设备可以选择任意文件来启动数据传输过程。

Note

如果您的设备在初始数据集中收到了所有必需的流文件 ID，则无需使用 `DescribeStream` API。

请按照以下步骤发送 `DescribeStream` 请求。

1. 订阅“已接受”主题筛选条件 `$aws/things/ThingName/streams/StreamId/description/json`。
2. 订阅“已被拒绝”主题筛选条件 `$aws/things/ThingName/streams/StreamId/rejected/json`。
3. 发送消息到 `$aws/things/ThingName/streams/StreamId/describe/json` 以发布 `DescribeStream` 请求。
4. 如果请求被接受，则您的设备会在“已接受”主题筛选条件中收到 `DescribeStream` 响应。
5. 如果请求被拒绝，则您的设备会在“已被拒绝”主题筛选条件条件中收到的错误响应。

Note

如果您在显示的主题和主题筛选条件中将 `json` 替换为 `cbor`，则您的设备将收到 CBOR 格式的消息，该格式比 JSON 更紧凑。

DescribeStream 请求

使用 JSON 格式的典型 `DescribeStream` 请求类似于以下示例。

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039"
```

```
}
```

- （可选）“c”是客户端令牌字段。

  客户端令牌不能超过 64 字节。超过 64 字节的客户端令牌将导致错误响应和 InvalidRequest 错误消息。

DescribeStream 回应

使用 JSON 格式的 DescribeStream 响应类似于以下示例。

```
{
    "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039",
    "s": 1,
    "d": "This is the description of stream ABC.",
    "r": [
        {
            "f": 0,
            "z": 131072
        },
        {
            "f": 1,
            "z": 51200
        }
    ]
}
```

- “c”是客户端令牌字段。如果它在 DescribeStream 请求中给出，则将会返回。使用客户端令牌将响应与其请求相关联。
- “s”是整数形式的流版本。您可以使用此版本对 GetStream 请求执行一致性检查。
- “r”包含流中的文件列表。
  - “f”是整数形式的流文件 ID。
  - “z”是以字节为单位的流文件大小。
- “d”包含流的描述。

从流文件中获取数据块

您可以使用 GetStream API，以便设备可以以小数据块的形式接收流文件，因此可以用于那些对处理大型数据块大小有限制的设备。要接收整个数据文件，设备可能需要发送或接收多个请求和响应，直到接收和处理了所有数据块。

GetStream 请求

请按照以下步骤发送 GetStream 请求。

1. 订阅“已接受”主题筛选条件 $aws/things/*ThingName*/streams/*StreamId*/data/json。
2. 订阅“已被拒绝”主题筛选条件 $aws/things/*ThingName*/streams/*StreamId*/rejected/json。
3. 发布 GetStream 请求到主题 $aws/things/*ThingName*/streams/*StreamId*/get/json。
4. 如果请求被接受，您的设备将在“已接受”主题筛选条件下收到一个或多个 GetStream 响应。每个响应消息都包含单个数据块的基本信息和数据负载。
5. 重复步骤 3 和 4 以接收所有数据块。如果请求的数据量大于 128 KB，则必须重复这些步骤。您必须对您的设备进行编程，以使用多个 GetStream 请求接收所有请求的数据。
6. 如果请求被拒绝，您的设备将在“已被拒绝”主题筛选条件下收到错误响应。

Note

- 如果您在显示的主题和主题筛选条件中将“json”替换为“cbor”，则您的设备将收到 CBOR 格式的消息，这种格式比 JSON 更紧凑。
- Amazon IoT 基于 MQTT 的文件传输将数据块的大小限制为 128 KB。如果您对超过 128 KB 的数据块发出请求，请求将失败。
- 您可以对总大小大于 128 KB 的多个数据块发出请求（例如，如果您请求 5 个数据块，各 32 KB，则共有 160 KB 的数据）。在这种情况下，请求不会失败，但您的设备必须发出多个请求才能接收请求的所有数据。当您的设备提出额外请求时，该服务将发送额外的数据块。我们建议您仅在正确接收和处理之前的响应之后，才继续执行新的请求。
- 无论请求的数据总大小如何，您都应该对设备进行编程，以便在未收到数据块或未正确接收数据块时启动重试。

使用 JSON 格式的典型 `GetStream` 请求类似于以下示例。

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "s": 1,
    "f": 0,
    "l": 4096,
    "o": 2,
    "n": 100,
    "b": "..."
}
```

- [可选]“c”是客户端令牌字段。

  客户端令牌不能超过 64 字节。超过 64 字节的客户端令牌将导致错误响应和 `InvalidRequest` 错误消息。
- [可选]“s”是流版本字段（整数）。

  基于 MQTT 的文件传输应用基于此请求的版本和云中的最新流版本的一致性检查。如果从设备发送的 `GetStream` 请求中的流版本与云中的最新流版本不匹配，则服务会发送错误响应和 `VersionMismatch` 错误消息。通常，设备会在初始数据集或对 `DescribeStream` 的响应中接收预期（最新）的流版本。
- “f”是流文件 ID（范围为 0 到 255 的整数）。

  使用 Amazon CLI 或 SDK 创建或更新流时，需要流文件 ID。如果设备请求了 ID 不存在的流文件，则该服务将发送错误响应和 `ResourceNotFound` 错误消息。
- “l”是以字节为单位的数据块大小（范围为 256 到 131072 的整数）。

  请参阅 为 GetStream 请求构建位图 (p. 839) 以了解如何使用位图字段指定流文件的哪些部分将在 `GetStream` 响应中返回。如果设备指定的数据块大小超出范围，则服务会发送错误响应和 `BlockSizeOutOfBounds` 错误消息。
- [可选]“o”是流文件中数据块的偏移量（范围为 0 到 98304 的整数）。

  请参阅 为 GetStream 请求构建位图 (p. 839) 以了解如何使用位图字段指定流文件的哪些部分将在 `GetStream` 响应中返回。最大值 98304 是基于 24 MB 流文件大小限制和 256 字节的最小数据块大小计算得出的。如果未指定，默认值为 0。
- [可选]“n”是请求的数据块数量（范围为 0 到 98304 的整数）。

  “n”字段指定 (1) 请求的数据块数量，或 (2) 使用位图字段 (“b”) 时，位图请求将返回的数据块数量的限制。第二种用法是可选的。如果没有定义，则默认为 131072/ *`DataBlockSize`*。
- [可选]“b”是一个位图，表示正在请求的数据块。

  使用位图，您的设备可以请求非连续的数据块，这使得处理错误后重试的操作更加方便。请参阅 为 GetStream 请求构建位图 (p. 839) 了解如何使用位图字段指定流文件的哪一部分将在 `GetStream` 响应

中返回。对于此字段，将位图转换为以十六进制表示法表示位图值的字符串。位图必须小于 12288 个字节。

Important

应指定 "n" 或 "b"。如果两者都未指定，当文件小于 131072 字节 (128 KB) 时，GetStream 请求可能无效。

GetStream 回应

JSON 格式的 GetStream 响应对于每个请求的数据块而言均如此示例所示。

```
{
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
    "f": 0,
    "l": 4096,
    "i": 2,
    "p": "..."
}
```

- “c”是客户端令牌字段。如果它在 GetStream 请求中给出，则将会返回。使用客户端令牌将响应与其请求相关联。
- “f”是当前数据块负载所属的流文件的 ID。
- “l”是数据块负载的大小（以字节为单位）。
- “i”是负载中包含的数据块的 ID。数据块从 0 开始编号。
- “p”包含数据块负载。此字段是一个字符串，表示 Base64 编码中数据块的值。

为 GetStream 请求构建位图

您可以使用 GetStream 请求中的位图字段 (b) 从流文件中获取非连续数据块。这有助于 RAM 容量有限的设备处理网络交付问题。设备只能请求那些未接收或未正确接收的数据块。位图确定将返回流文件的哪些数据块。对于位图中设置为 1 的每个位，将返回相应的流文件数据块。

下面的示例将演示如何在 GetStream 请中指定位图机器支持的字段。例如，您希望以 256 字节的数据块（块大小）接收流文件。将每个 256 字节的数据块视为有一个指定其在文件中位置的数字，从 0 开始。因此，数据块 0 是文件中第一个 256 字节的数据块，数据块 1 是第二个，依此类推。您想要从文件中请求数据块 20、21、24 和 43。

数据块偏移

由于第一个数据块是第 20 号，因此指定偏移量（字段 o）为 20，以节省位图中的空间。

数据块数量

为了确保您的设备收到的数据块数量不会超过其在有限内存资源中所能处理的数量，您可以指定基于 MQTT 的文件传输发送的每条消息中应返回的最大数据块数量。请注意，如果位图本身指定的数据块数量小于此数值，或者如果它会使基于 MQTT 的文件传输发送的响应消息总大小大于每个 GetStream 请求 128 KB 的服务限制，则将忽略此值。

数据块位图

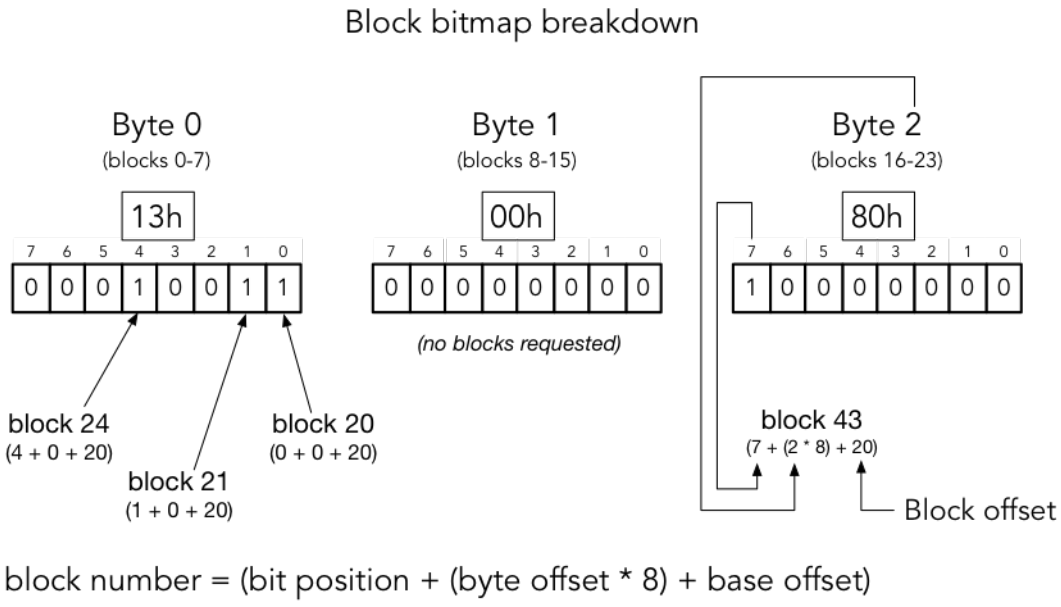
位图本身是一个以十六进制表示法表示的无签名字节数组，并以数字字符串的形式包含在 GetStream 请求中。但要构造这个字符串，我们首先要将位图视为一个长序列的位（二进制数）。如果此序列中的位设置为 1，则流文件中的相应数据块将被发送回设备。在我们的示例中，我们希望接收数据块 20、21、24 和 43，因此我们必须在位图中设置位 20、21、24 和 43。我们可以使用数据块偏移来节省空间，因此在从每个数据块编号中减去偏移后，我们希望设置位 0、1、4 和 23，如下例所示。

```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

一次取一个字节（8 位），这个字节通常被写作：“0b00010011”、“0b00000000”和“0b10000000”。位 0 显示在第一个字节末尾的二进制表示中，位 23 显示在最后一个字节的开头。除非您早已了解惯例，否则这种表示方式可能会令人困惑不已。第一个字节包含位 7-0（按该顺序），第二个字节包含位 15-8，第三个字节包含位 23-16，依此类推。在十六进制表示法中，这将转换为“0x130080”。

Tip

您可以将标准二进制转换为十六进制表示法。一次取四个二进制数字，并将它们转换为十六进制等效数字。例如，“0001”变为“1”，“0011”变成 “3”，依此类推。



将这一切结合起来，我们的 GetStream 请求在 JSON 格式中看起来与下方类似。

```
{
    "c" : "1",       // client token
    "s" : 1,         // expected stream version
    "l" : 256,       // block size
    "f" : 1,         // source file index id
    "o" : 20,        // block offset
    "n" : 32,        // number of blocks
    "b" : "0x130080" // A missing blockId bitmap starting from the offset
}
```

处理来自 Amazon IoT 基于 MQTT 的文件传输的错误

为 DescribeStream 和 GetStream API 发送到设备的错误响应包含客户端令牌、错误代码和错误消息。典型的错误响应类似于以下示例。

```
{
    "o": "BlockSizeOutOfBounds",
    "m": "The block size is out of bounds",
    "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380"
}
```

- 如果出现错误，则“o”为指示错误原因的代码。有关更多详细信息，请参阅本部分后面的错误代码。
- “m”是包含错误详细信息的错误消息。
- “c”是客户端令牌字段。如果它在 `DescribeStream` 请求中提供，则可能会返回。您可以使用客户端令牌将响应与其请求相关联。

  客户端令牌字段并不总是包含在错误响应中。当请求中给出的客户端令牌无效或格式不正确时，它不会在错误响应中返回。

  Note

  为了向后兼容，错误响应中的字段可能采用非缩写形式。例如，错误代码可能由“code”或“o”字段表示，客户端令牌字段可以由“clientToken”或“c”字段表示。建议您使用上面显示的缩写形式。

InvalidTopic

流消息的 MQTT 主题无效。

InvalidJson

流请求不是有效的 JSON 文档。

InvalidCbor

流请求不是有效的 CBOR 文档。

InvalidRequest

请求整体被标识为格式错误。有关更多信息，请参阅错误消息。

Unauthorized

请求未获授权访问存储介质（如 Amazon S3）中的流数据文件。有关更多信息，请参阅错误消息。

BlockSizeOutOfBounds

数据块大小超出了界限。请参阅 Amazon IoT Core Service Quotas 中的“基于 MQTT 的文件传输”部分。

OffsetOutOfBounds

偏移超出界限。请参阅 Amazon IoT Core Service Quotas 中的“基于 MQTT 的文件传输”部分。

BlockCountLimitExceeded

请求数据块的数量超出了界限。请参阅 Amazon IoT Core Service Quotas 中的“基于 MQTT 的文件传输”部分。

BlockBitmapLimitExceeded

请求位图的大小超出了界限。请参阅 Amazon IoT Core Service Quotas 中的“基于 MQTT 的文件传输”部分。

ResourceNotFound

找不到请求的流、文件、文件版本或数据块。有关更多详细信息，请参阅错误消息。

VersionMismatch

请求中的流版本与基于 MQTT 的文件传输功能中的流版本不匹配。这表示自设备最初接收流版本以来，流数据已被修改。

E | ETagMismatch

流中的 S3 ETag 与最新 S3 对象版本的 ETag 不匹配。

InternalError

基于 MQTT 的文件传输中发生内部错误。

FreeRTOS OTA 中的一个示例使用案例

FreeRTOS OTA (over-the-air) 代理使用Amazon IoT基于 MQTT 的文件传输，将 FreeRTOS 固件映像传输到 FreeRTOS 设备。为将初始数据集发送到设备，它使用 Amazon IoT Job 服务将 OTA 更新任务安排到 FreeRTOS 设备。

有关基于 MQTT 的文件传输客户端的实施案例参考，请参阅 FreeRTOS 文档中的 FreeRTOS OTA 代理代码。

Amazon IoT Device Defender

Amazon IoT Device Defender 是一项安全服务，您可以借助此服务审核设备的配置、监控互联设备以检测异常行为，并降低安全风险。有了此服务，您可以在整个 Amazon IoT 设备实例集中实施一致的安全策略，并能够在设备遭到破坏时快速响应。

IoT 实例集可能由大量具有不同功能、长期存在且地理位置分散的设备组成。这些特性导致实例集设置复杂且容易出错。由于设备的计算能力、内存和存储功能通常有限，因而限制了在设备本身上对加密和其它形式的安全功能的使用。此外，设备经常使用具有已知漏洞的软件。这些因素不仅令 IoT 实例集成为吸引黑客的目标，而且导致难以持续保护设备实例集安全。

Amazon IoT Device Defender 提供了识别安全问题和最佳实践偏离情况的工具，从而解决了这些难题。Amazon IoT Device Defender 可以审计设备实例集，以确保它们遵守安全最佳实践并检测设备上的异常行为。

Amazon IoT Device Defender 入门

您可以使用以下教程来学习使用 Amazon IoT Device Defender。

主题

- 设置 (p. 843)
- 审计指南 (p. 844)
- ML Detect 指南 (p. 856)
- 自定义查看 Amazon IoT Device Defender 审计结果的时间和方式 (p. 879)

设置

首次使用 Amazon IoT Device Defender 前，请完成以下任务：

主题

- 注册一个 Amazon Web Services 账户 (p. 16)
- 保护 IAM 用户 (p. 17)

注册一个 Amazon Web Services 账户

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

注册 Amazon Web Services 账户

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

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

   当您注册 Amazon Web Services 账户时，系统将会创建一个 Amazon Web Services 账户根用户。根用户有权访问该账户中的所有 Amazon Web Services和资源。作为安全最佳实践，请 为管理用户分配管理访问权限，并且只使用根用户执行 需要根用户访问权限的任务。

Amazon注册过程完成后，会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择 My Account (我的账户) 来查看当前的账户活动并管理您的账户。

保护 IAM 用户

注册 Amazon Web Services 账户后，启用多重身份验证（MFA）来保护您的管理员用户。有关说明，请参阅《IAM 用户指南》中的为 IAM 用户启用虚拟 MFA 设备（控制台）。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。要保护您的 IAM 用户，请开启 MFA 并仅向 IAM 用户授予执行任务所需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅《IAM 用户指南》中的以下主题：

- 在您的 Amazon Web Services 账户 中创建 IAM 用户
- 适用于 Amazon 资源的访问管理
- 基于 IAM 身份的策略示例

这些任务创建一个 Amazon Web Services 账户和一个具有该账户的管理员权限的用户。

审计指南

本教程提供有关如何配置定期审计、设置告警、查看审计结果和减少审计问题的说明。

主题

- 先决条件 (p. 844)
- 启用审计检查 (p. 844)
- 查看审计结果 (p. 848)
- 创建审计缓解操作 (p. 849)
- 将缓解操作应用于审计查找结果 (p. 851)
- 启用 SNS 通知（可选）(p. 852)
- 启用日志记录（可选）(p. 854)
- 创建 Amazon IoT Device Defender 审计 IAM 角色（可选）(p. 855)

先决条件

要完成本教程，您需要：

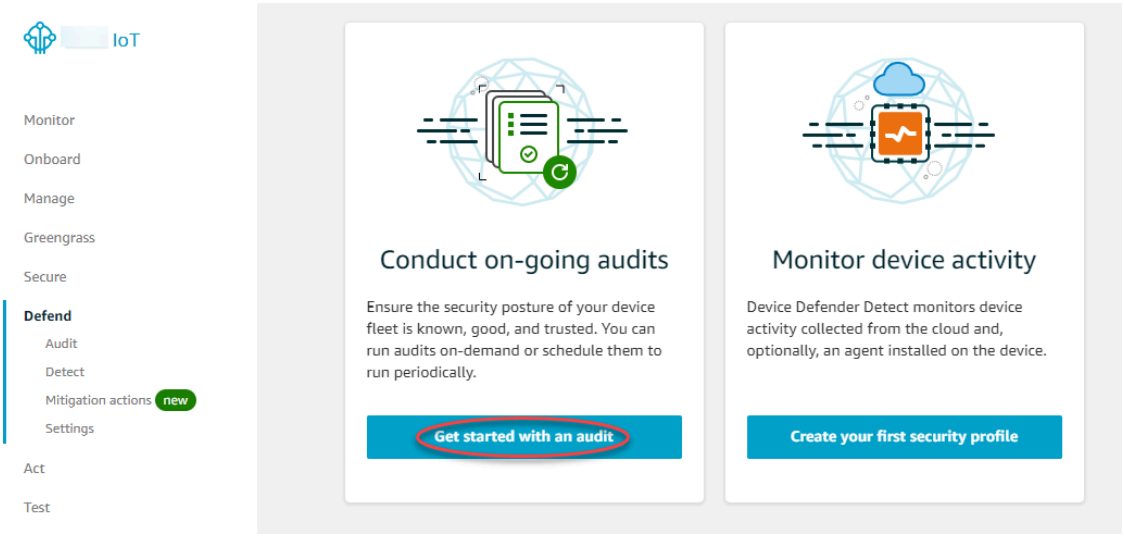
- 一个 Amazon Web Services 账户。如果您尚未拥有账户，请参阅设置。

启用审计检查

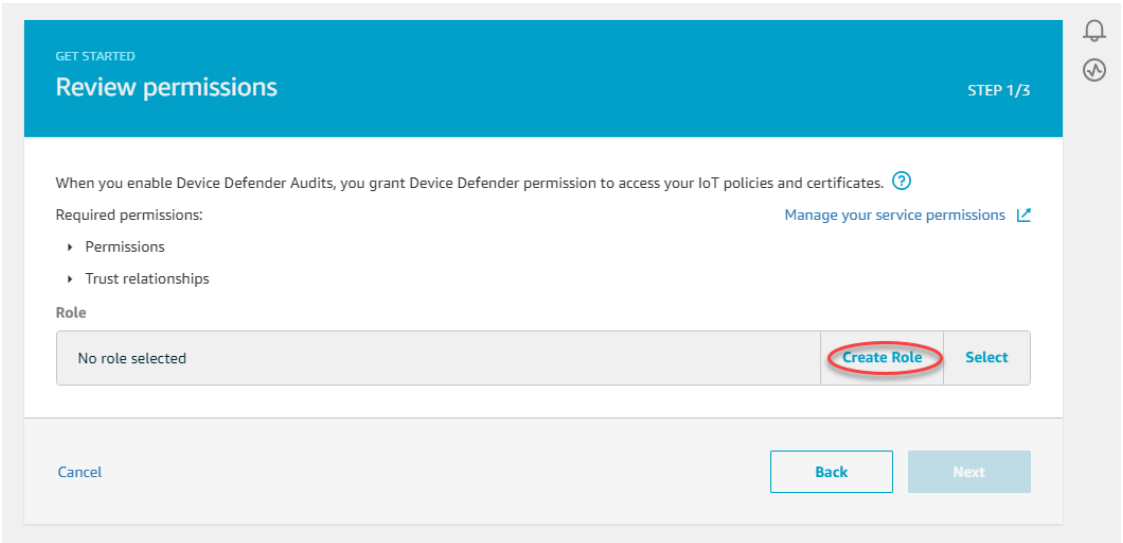
在以下流程中，您可以启用审计检查以查看账户和设备设置及策略，以确保安全措施已就绪。在本教程中，我们指导您启用所有审计检查，但您可以仅选择所需的检查。

审计定价是按每月每台设备计数制定的（连接到 Amazon IoT 机群设备）。因此，在使用此功能时，添加或删除审计检查不会影响您的月度账单。

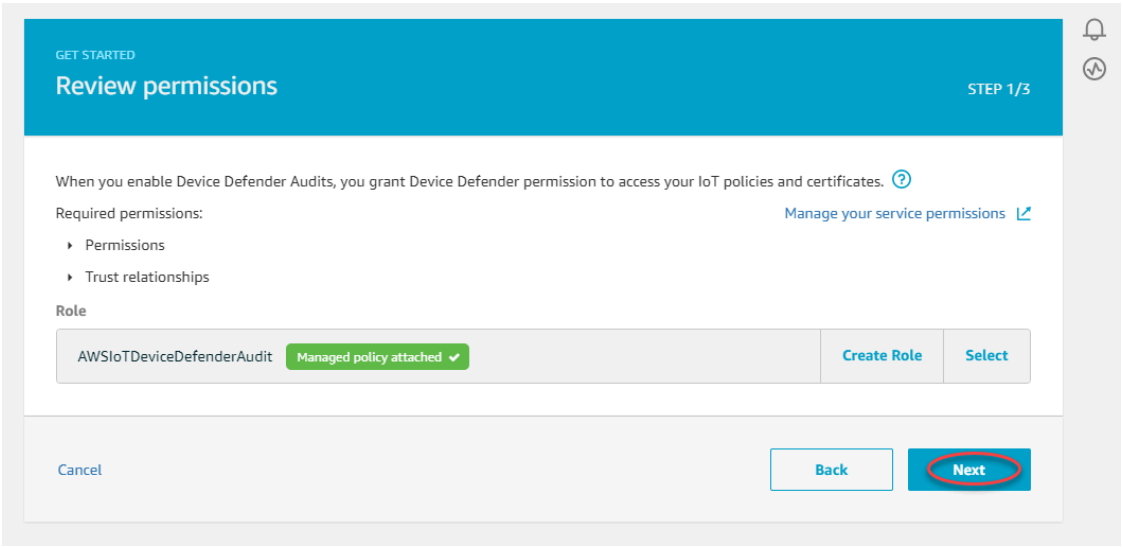
1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Get started with an audit（审计入门）。



2. Get started with Device Defender Audit（Device Defender 审计入门）页面概览了启用审计检查所需的步骤。查看完该页面的内容后，选择 Next（下一步）。
3. 如果您已有要使用的角色，可以选择该角色。否则请选择 Create Role（创建角色）并将其命名为 *AWSIoTDeviceDefenderAudit*。

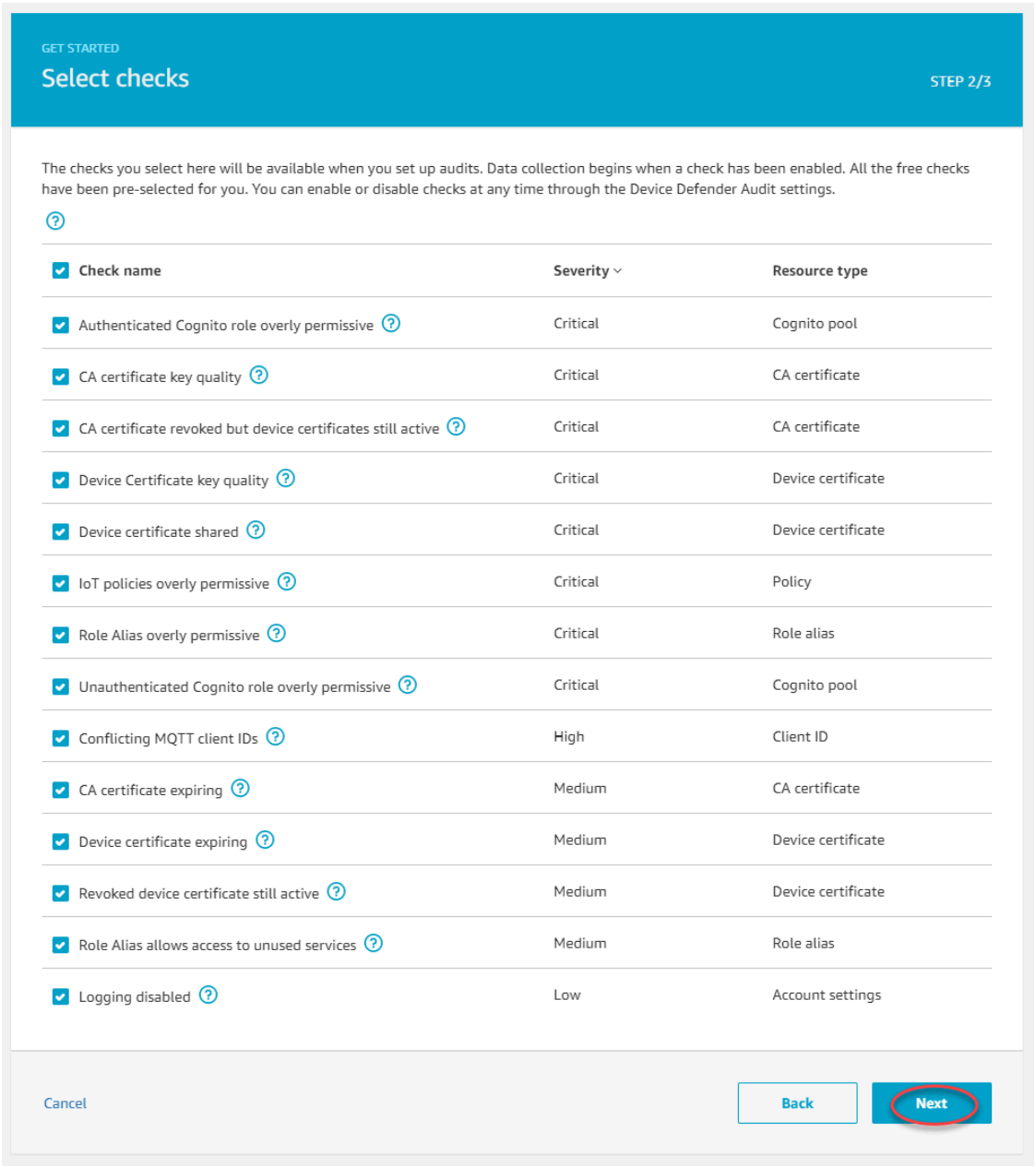


您应看到所需权限自动附加到了角色。请选择 Permissions（权限）和 Trust relationship（信任关系）胖的三角形以查看授予了哪些权限。当您准备好继续前进时，选择 Next（下一步）。



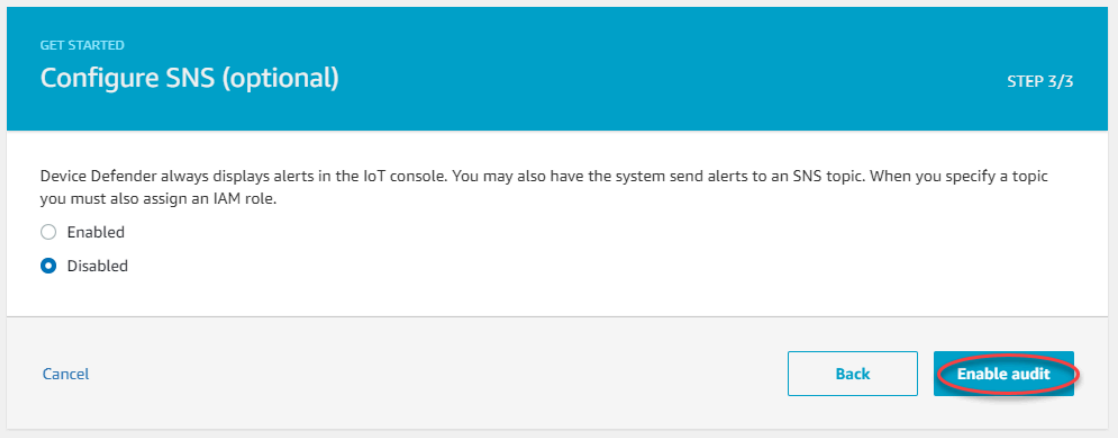
4. 在 Select checks（选择检查）页面，您将看到可以选择的所有审计检查。在本教程中，我们将指示您选择所有检查，但您也可以选择任何所需的检查。每个审计检查旁边有一个帮助图标，用于描述审计检查的功能。有关审计检查的详细信息，请参阅审核检查。

   选择好您的检查后，请选择 Next（下一步）。

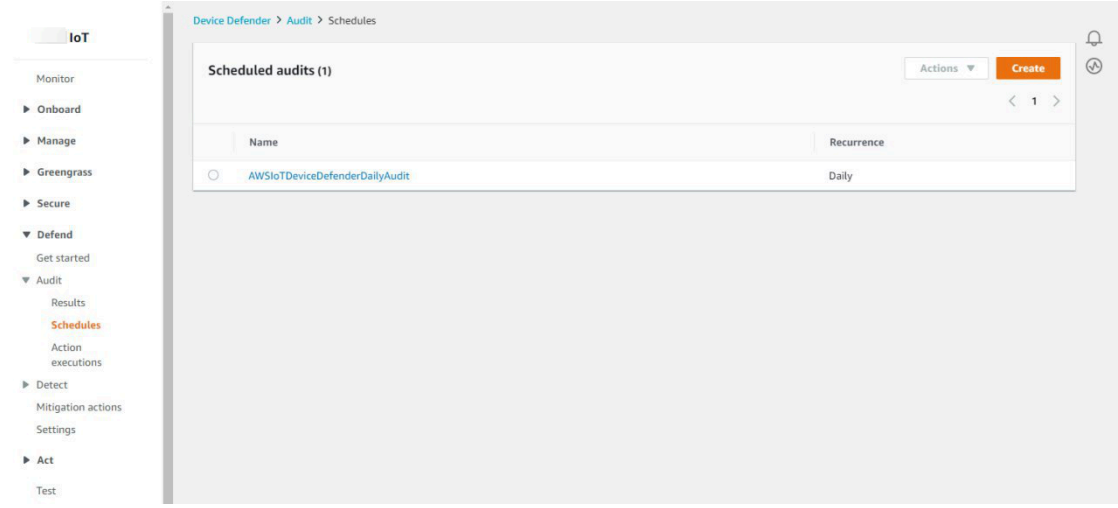


您可以在 Settings（设置）中随时更改已配置的审计检查。

5. 在 Configure SNS (optional)（配置 SNS（可选））页面中，选择 Enable audit（启用审计）。如果您希望启用 SNS 通知，请参阅 启用 SNS 通知（可选）(p. 852)。



6. 您将被重定向至 Audit（审计）项下的 Schedules（安排）。

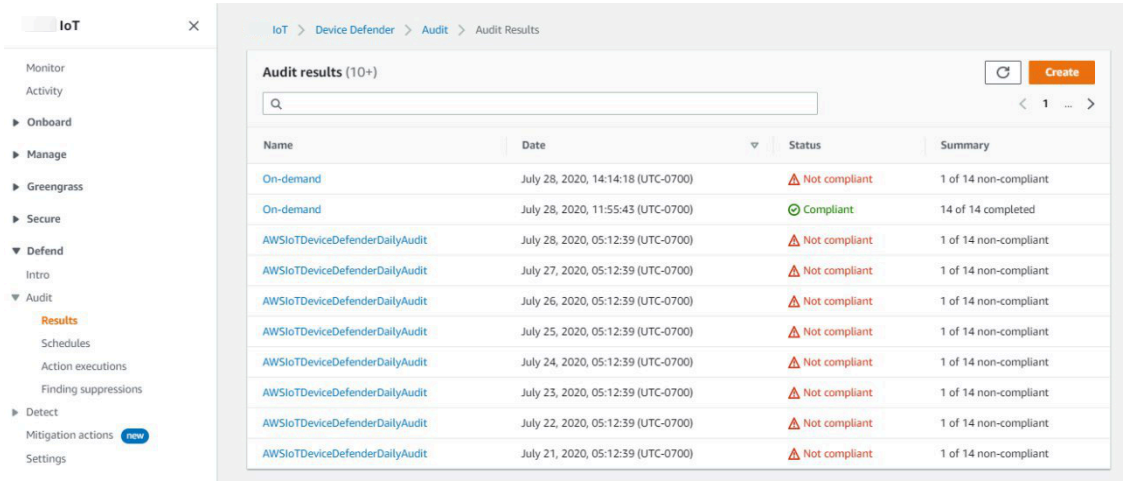


查看审计结果

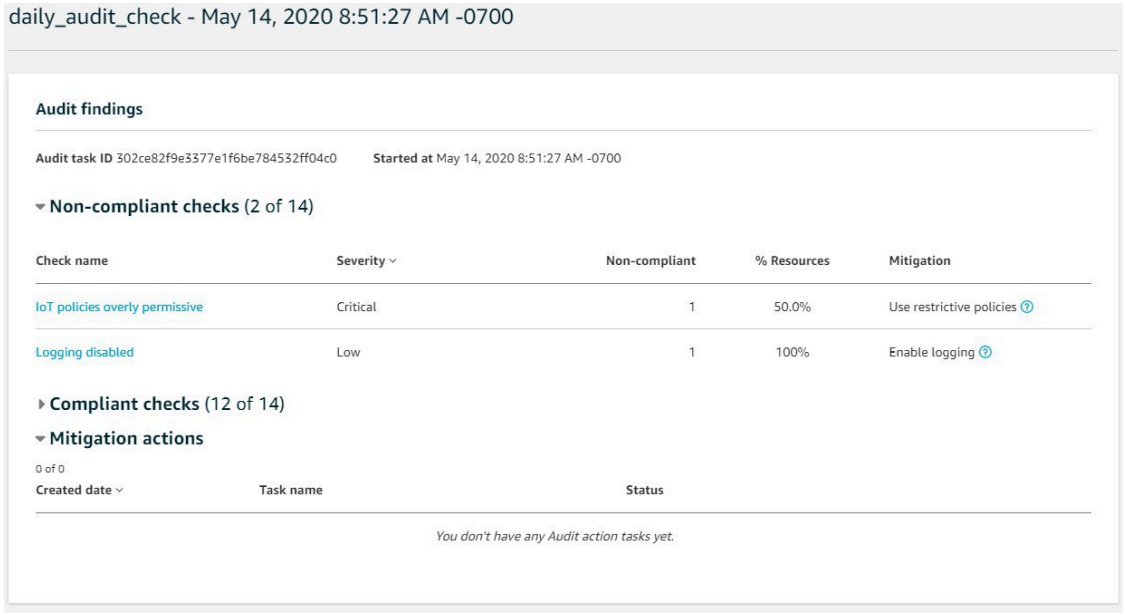
以下流程介绍如何查看审计结果。在本教程中，您将看到在 启用审计检查 (p. 844) 教程设置的审计检查的审计结果。

查看审计结果

1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），选择 Audit（审计），然后选择 Results（结果）。
2. Summary（摘要）会告诉您是否存在任何不合规的检查。



3. 选择您想要调查的审计检查的 Name（名称）。



4. 使用问号获取有关如何使您的不合规检查合规的指导。例如，您可以按照 启用日志记录（可选）(p. 854) 中的步骤让“禁用日志记录”检查合规。

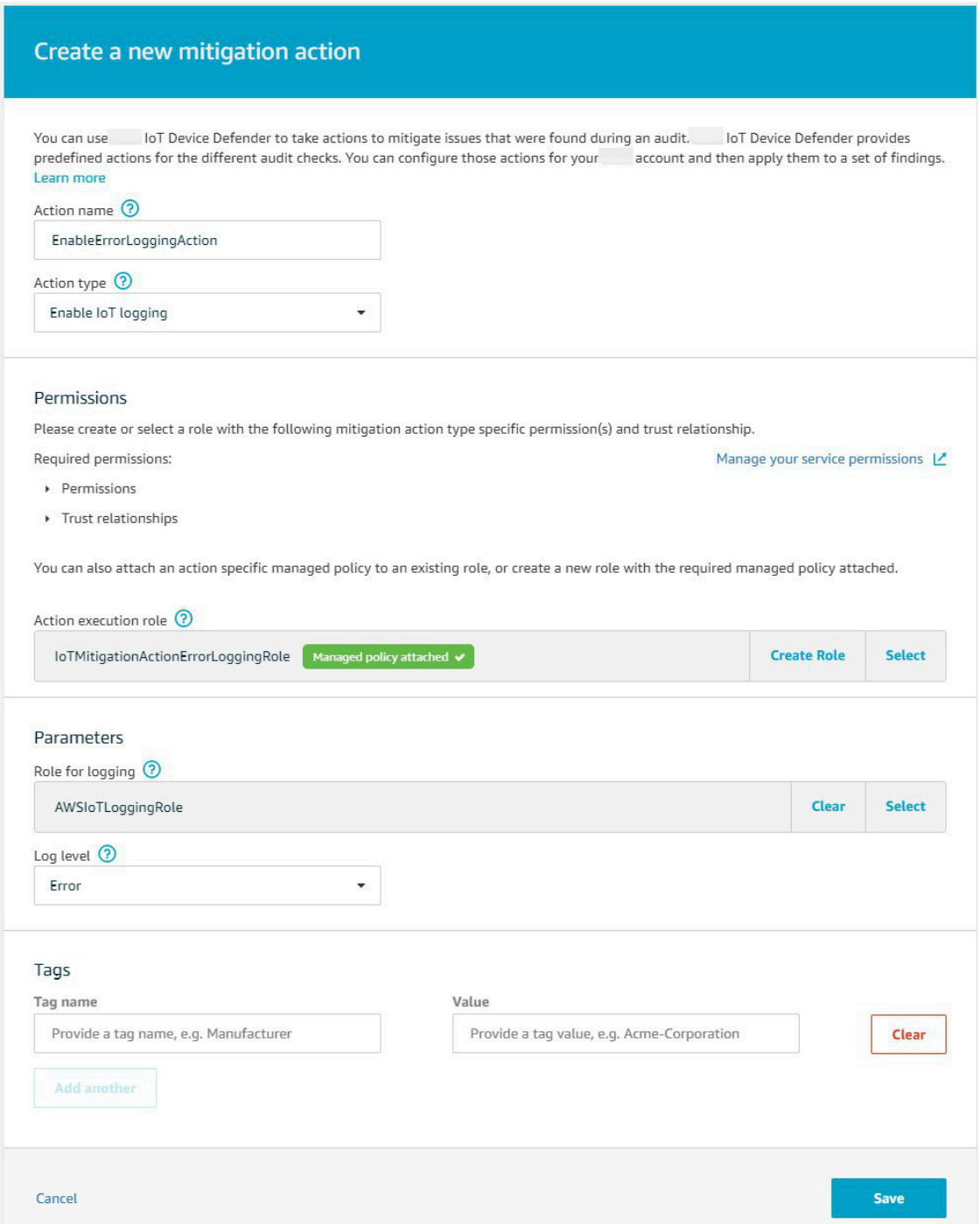
创建审计缓解操作

在以下流程中，您将创建一个 Amazon IoT Device Defender 审计缓解操作以启用 Amazon IoT 日志记录。每个审计检查都已映射缓解操作，这些操作将影响您为想要修复的审计检查所选择的操作类型。有关更多信息，请参阅缓解操作。

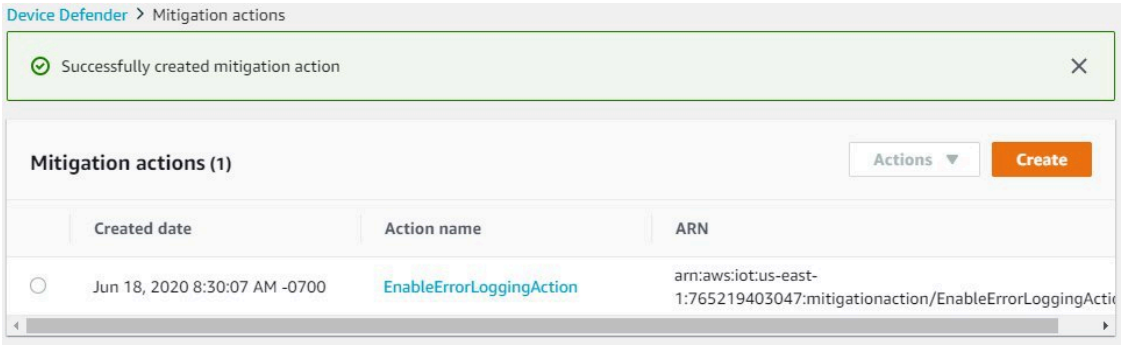
使用 Amazon IoT 控制台创建缓解操作

1. 打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择防护，然后选择 Mitigation Actions (缓解操作)。
3. 在 Mitigation Actions (缓解操作) 页面上，选择创建。
4. 在 Create a Mitigation Action（创建缓解操作）页面，在 Action name（操作名称）中为您的缓解操作输入唯一名称，例如 *EnableErrorLoggingAction*。

5. 在 Action type（操作类型）中，请选择 Enable IoT logging（启用 IoT 日志记录）。
6. 在 Action execution role（操作执行角色）中，选择 Create Role（创建角色）。对于 Name（名称），使用 *IoTMitigationActionErrorLoggingRole*。然后选择 Create role（创建角色）。
7. 在 Parameters（参数）中，在 Role for logging（用于日志记录的角色）中，选择 `AWSIoTLoggingRole`。对于 Log level（日志级别），选择 `Error`。



8. 选择保存可将缓解操作保存到您的 Amazon 账户。
9. 创建后，您将看到以下页面指示，表明您的缓解操作已成功创建。

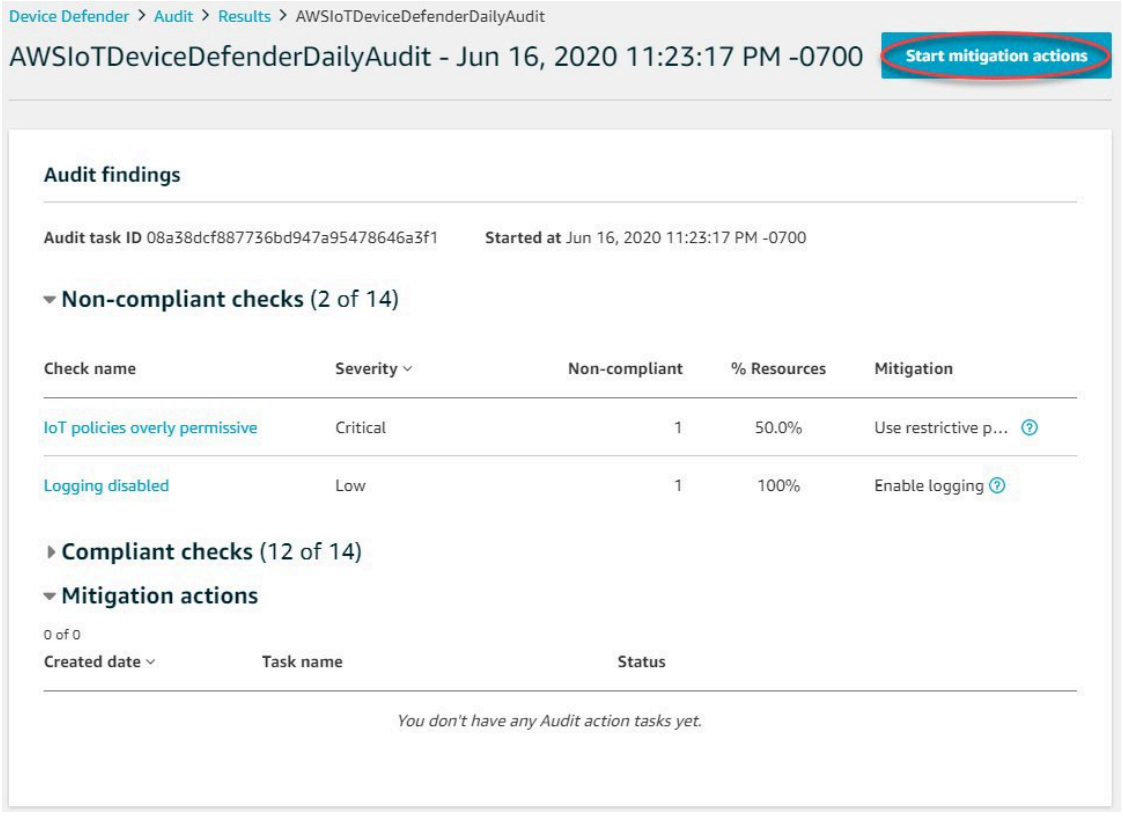


将缓解操作应用于审计查找结果

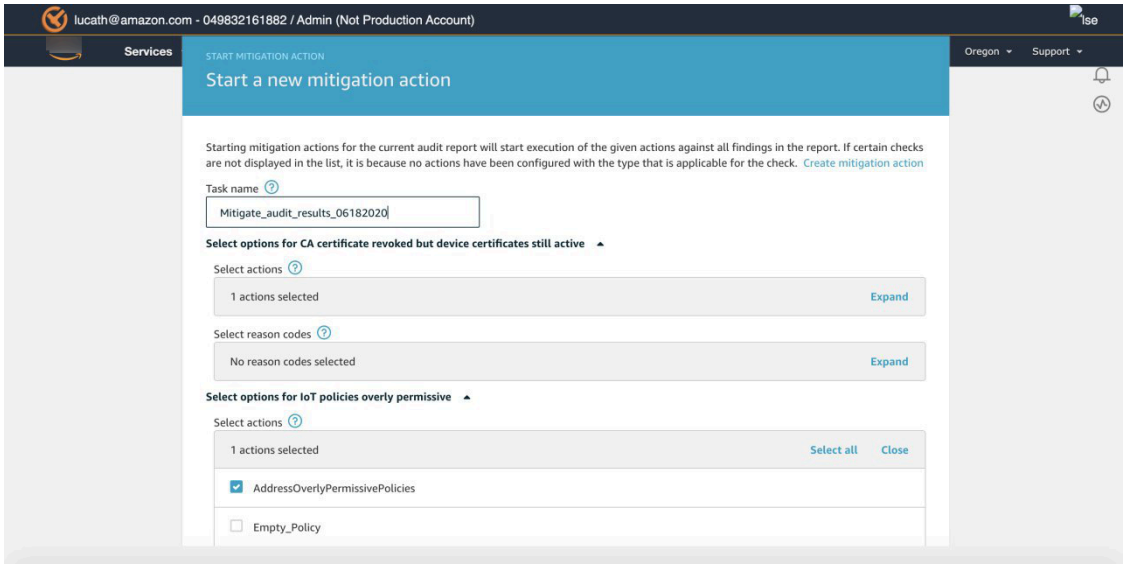
以下流程介绍如何将缓解操作应用于审计结果。

减少不合规审计查找结果

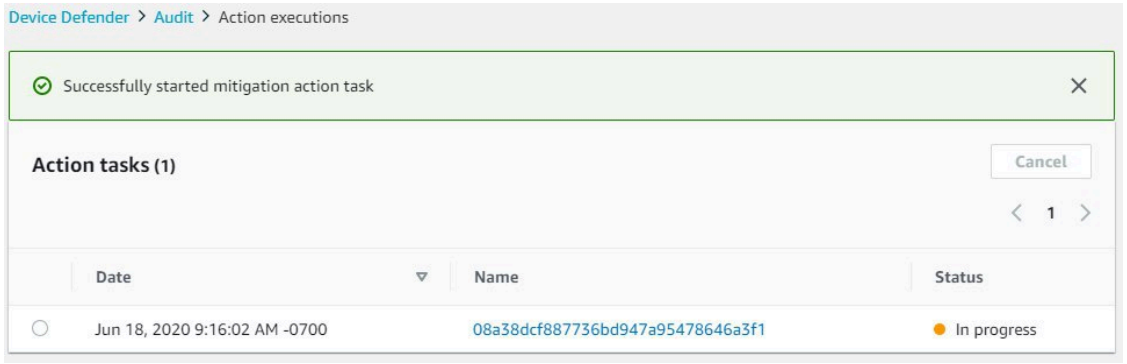
1. 打开 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择 Audit（审计），然后选择 Results（结果）。选择要响应的审计名称。
3. 检查您的结果。注意 `Logging disabled` 位于 Non-compliant checks（不合规检查）项下。
4. 选择 Start mitigation actions（启动缓解操作）。



5. 在 Select actions（选择操作）中，为每个不合规的结果选择适当的操作以解决这些问题。



6. 选择 Confirm（确认）。
7. 缓解操作一旦启动，可能需要几分钟时间才能运行。



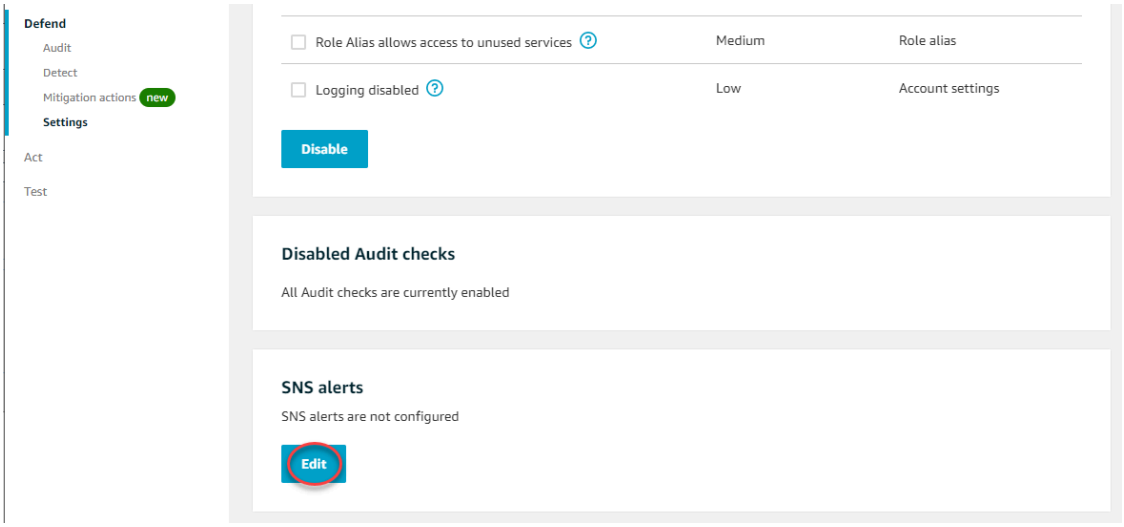
要检查缓解操作是否有效

1. 在 Amazon 控制台的导航窗格中，选择 Settings（设置）。
2. 确认 Logs（日志）为 `Enabled` 且 Level of verbosity（详细程度级别）为 `Error`。

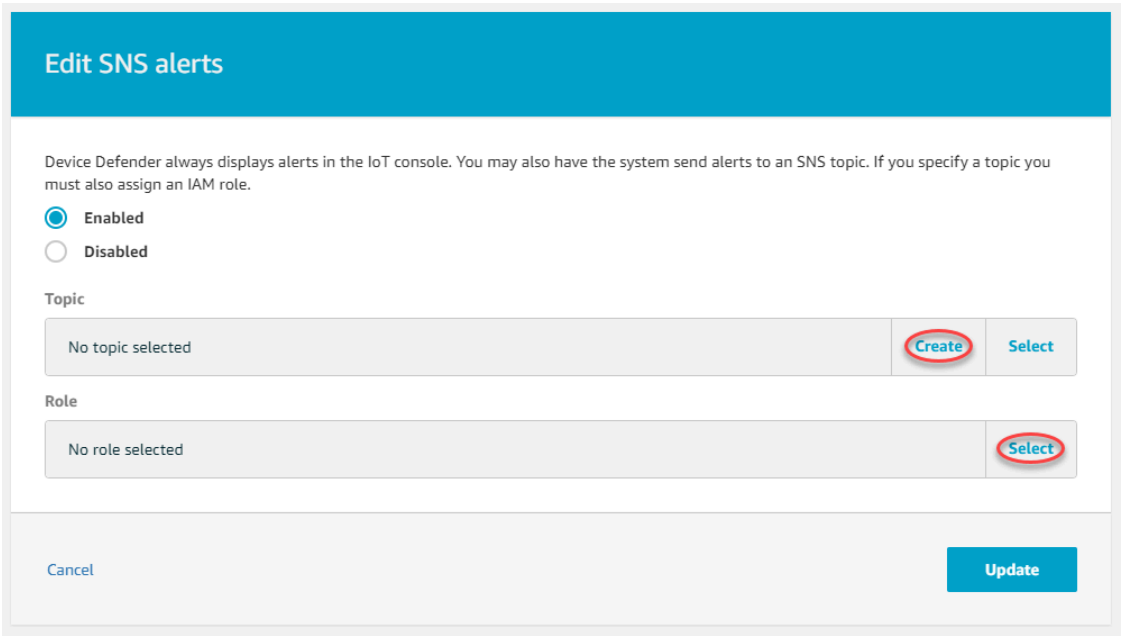
启用 SNS 通知（可选）

在以下流程中，您可以启用 Simple Notifications Service (SNS) 通知，以便在审计发现任何不合规的资源时向您发出提示。在本教程中，您将为 启用审计检查 (p. 844) 教程中启用的审计检查设置通知。

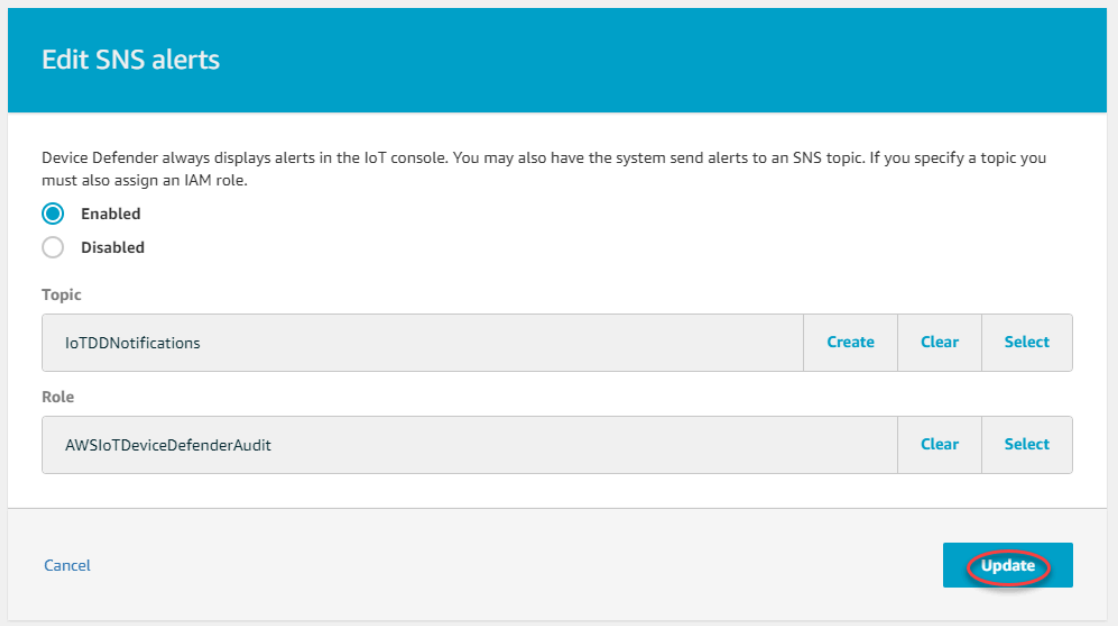
1. 首先，您需要创建一个 IAM 策略以通过 Amazon 管理控制台提供 Amazon SNS 的访问权限。您可以按照 创建 Amazon IoT Device Defender 审计 IAM 角色（可选）(p. 855) 进程操作，但应在第 8 步中选择 AWSIoTDeviceDefenderPublishFindingsToSNSMitigationAction。
2. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Settings（设置）。
3. 在 SNS alerts（SNS 提示）中，选择 Edit（编辑）。



4. 在 Edit SNS alerts（编辑 SNS 提示）页面中，选择 Enabled（已启用）。在 Topic（主题）中，选择 Create（创建）。将主题命名为 *IoTDDNotifications*，然后选择 Create（创建）。在 Role（角色）中，选择您创建的 *AWSIoTDeviceDefenderAudit* 的角色。



Select Update（更新）。



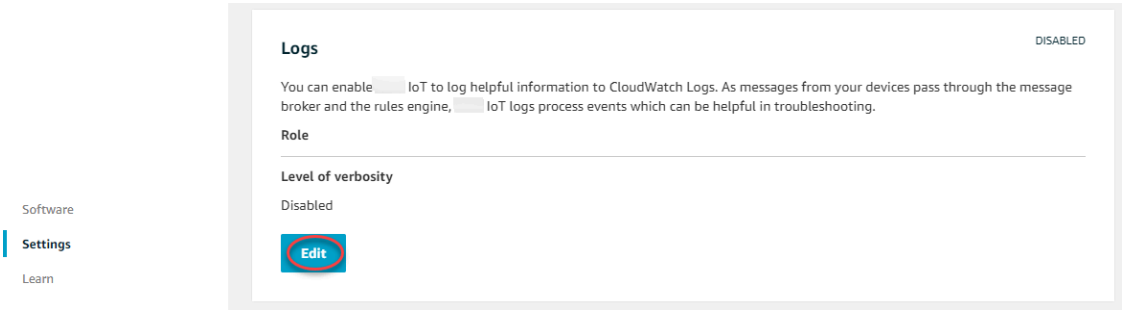
如果您希望通过 SNS 在运维平台上接收电子邮件或文本，请参阅使用 Amazon SNS 发送用户通知。

启用日志记录（可选）

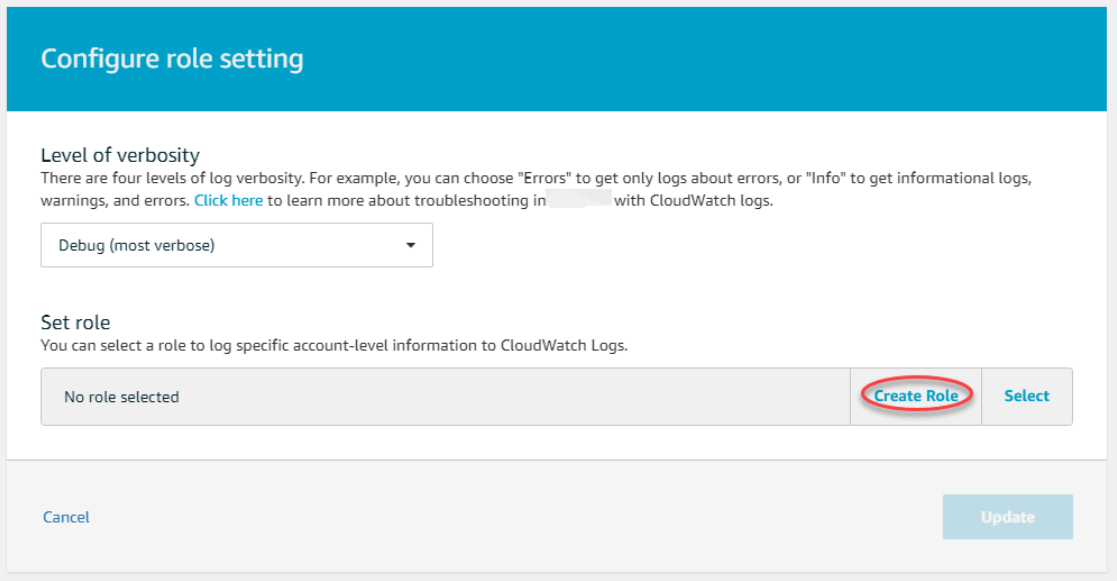
本流程介绍如何启用 Amazon IoT 将信息记录到 CloudWatch Logs 中。这将允许您查看您的审计结果。启用日志记录可能会导致费用产生。

要启用日志记录：

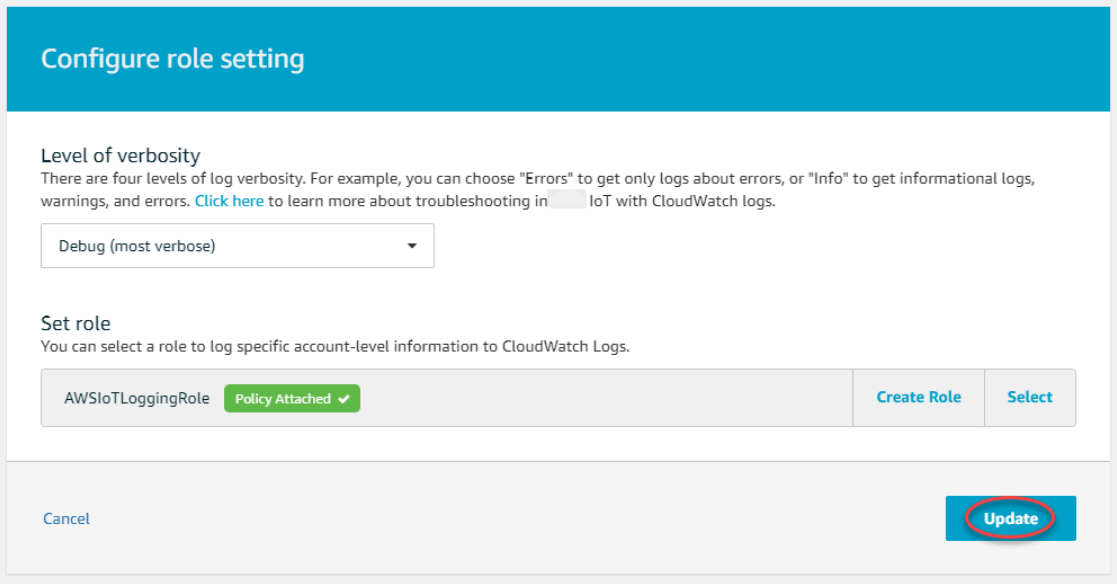
1. 在 Amazon 控制台的导航窗格中，选择 Settings（设置）。
2. 在 Logs（日志）项下，选择 Edit（编辑）。



3. 在 Leve of verbosity（详细程度级别）项下，选择 Debug (most verbosity（调试（最详细））。
4. 在 Set role（设置角色）项下，选择 Ceate Role（创建角色）并将角色命名为 *AWSIoTLoggingRole*。策略将自动附加。



Select Update（更新）。



创建 Amazon IoT Device Defender 审计 IAM 角色（可选）

在以下流程中，您将创建一个 Amazon IoT Device Defender 审计 IAM 角色，为 Amazon IoT Device Defender 提供 Amazon IoT 的读取访问权限。

1. 通过以下网址导航到 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Users（用户），然后选择 Add user（添加用户）。
3. 对于用户名，输入 **`Administrator`**。
4. 选中 Amazon Management Console access（亚马逊云科技管理控制台访问权限）旁边的复选框。然后选择自定义密码，并在文本框中输入新密码。
5. （可选）预设情况下，Amazon 要求新用户在首次登录时创建新密码。您可以清除用户必须在下次登录时创建新密码旁边的复选框以允许新用户在登录后重置其密码。

6. 选择下一步: 权限。
7. 在 Set permissions (设置权限) 下，选择 Attach existing policies directly (直接附加现有策略)。
8. 在策略列表中，选中 AWSIoTDeviceDefenderAudit 旁边的复选框。
9. 选择 Next: Tags（下一步: 标签）。
10. 选择下一步：审核以查看要添加到新用户的组成员资格的列表。如果您已准备好继续，请选择创建用户。

ML Detect 指南

在本入门指南中，您将创建一个 ML Detect 安全配置文件，该配置文件使用机器学习 (ML) 根据设备中的历史指标数据创建预期行为模型。当 ML Detect 正在创建 ML 模型时，您可以监控其进度。构建 ML 模型后，您可以持续查看和调查告警，并缓解已发现的问题。

有关 ML Detect 及其 API 和 CLI 命令的更多信息，请参阅 ML Detect (p. 955)。

本章包含以下部分：

- 先决条件 (p. 856)
- 如何在控制台中使用 ML Detect (p. 856)
- 如何将 （ ML Detect 与 CLI 一起使用 (p. 870)

先决条件

- 一个 Amazon Web Services 账户。如果您尚未拥有账户，请参阅设置。

如何在控制台中使用 ML Detect

教程

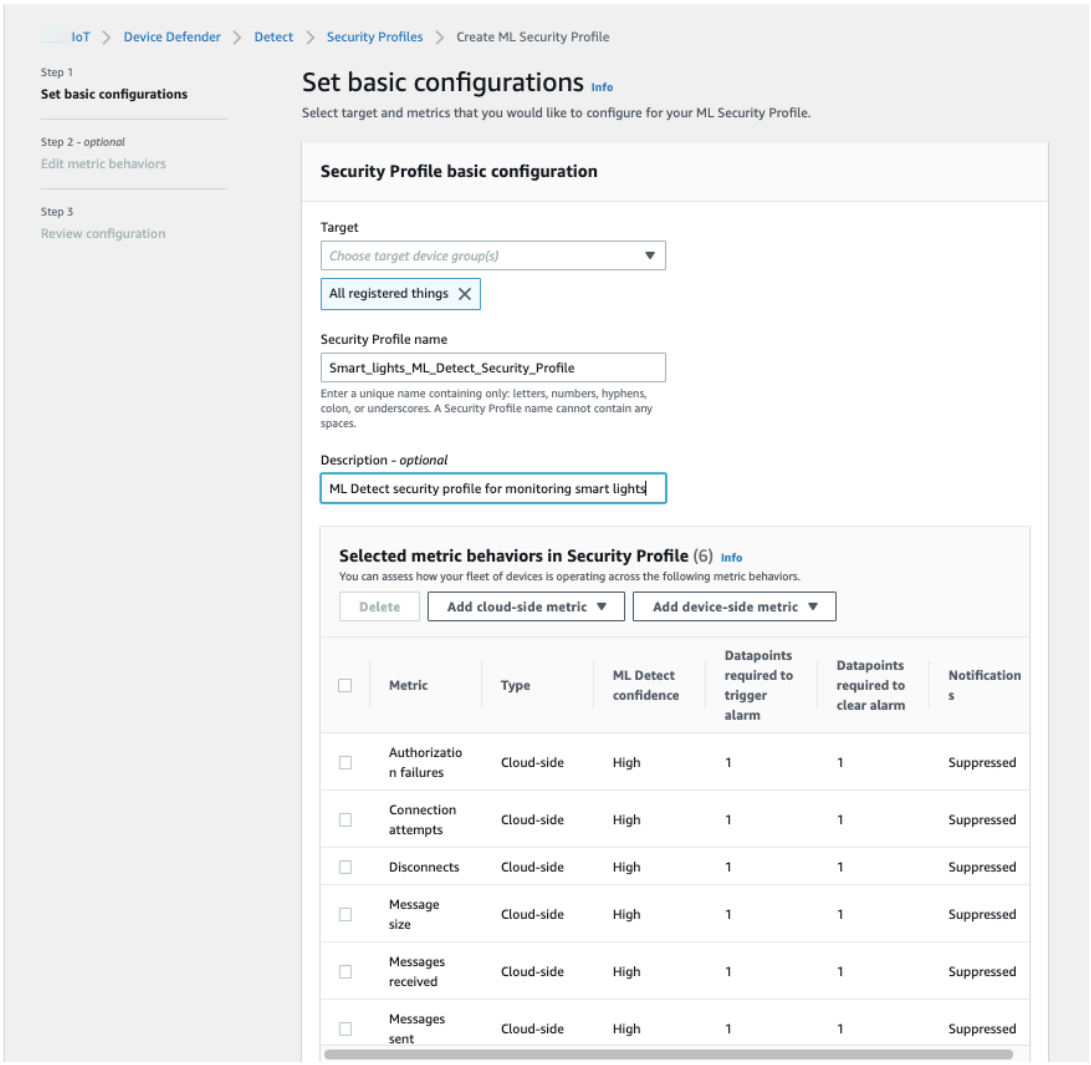
- 启用 ML Detect (p. 856)
- 监控您的 ML 模型状态 (p. 860)
- 查看您的 ML Detect 告警 (p. 861)
- 微调您的机器学习（ ML ）告警 (p. 863)
- 标记告警的验证状态 (p. 865)
- 缓解已确定的设备问题 (p. 866)

启用 ML Detect

以下流程详细介绍了如何在控制台中设置 ML Detect。

1. 首先，确保您的设备将按 ML Detect 最低要求 (p. 956)中的定义要求创建最小的数据点，以进行持续训练和刷新模型。要进行数据收集，请确保您的安全配置文件已附加到目标，该目标可以是事物或事物组。
2. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护）。选择 Detect（检测）、Security profiles（安全配置文件）、Create security profile（创建安全配置文件），然后选择 Create ML anomaly Detect profile（创建 ML 异常检测配置文件）。
3. 在 Set basic configurations（设置基本配置）页面上，执行以下操作。
   - 在 Target（目标）项下，选择您的目标设备组。
   - 在 Security profile name（安全配置文件名称）中，输入您的安全配置文件的名称。
   - （可选）在 Description（说明）项下，您可以编写 ML 配置文件的简述。

- 在 Selected metric behaviors in Security Profile（安全配置文件中特定的指标行为）项下，选择要监控的指标。

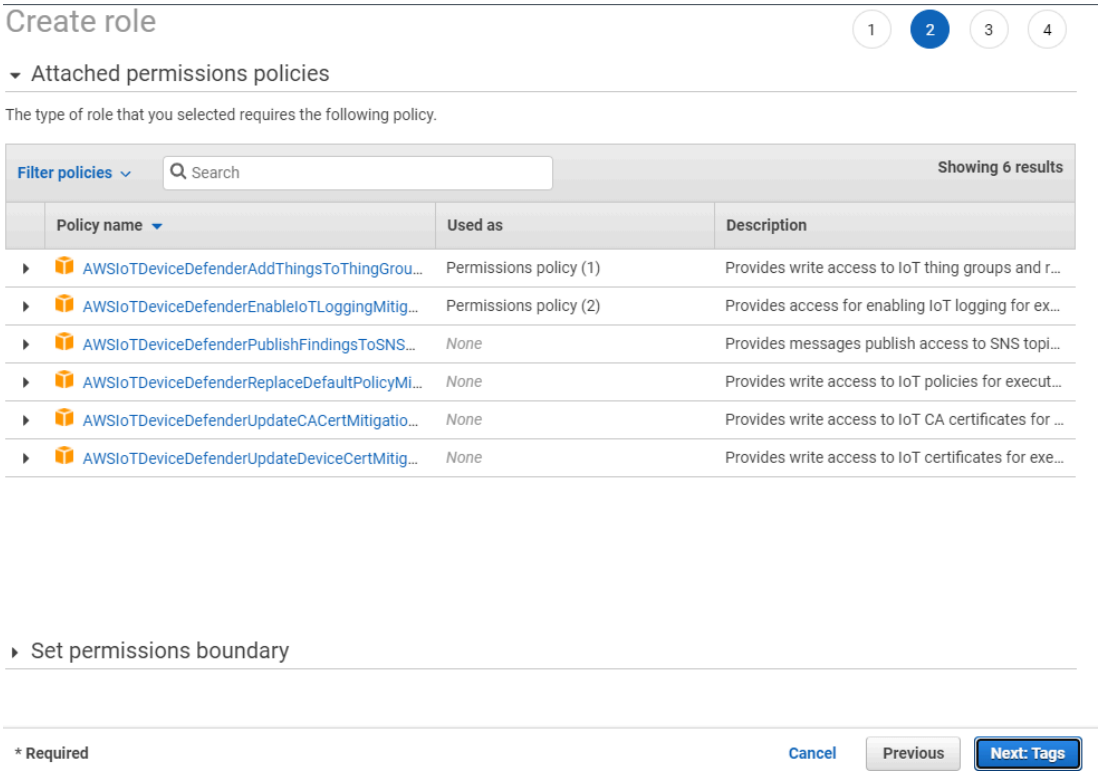


完成此操作后，选择 Next (下一步)。

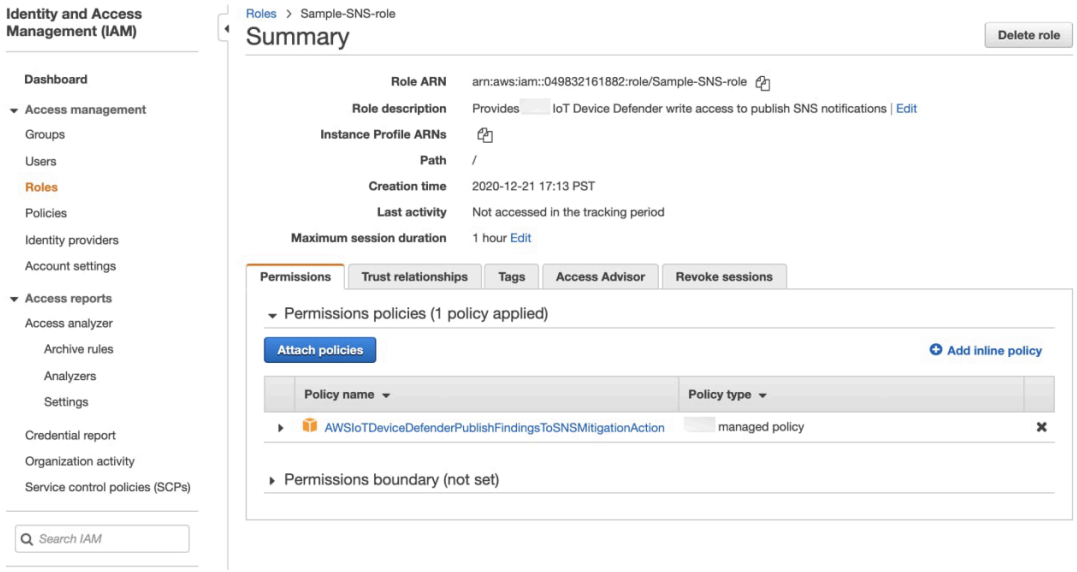
4. 在 Set SNS (optional)（设置 SNS（可选））页面上，为设备违反配置文件中的行为时的告警通知指定 SNS 主题。选择要用于发布到选定 SNS 主题的 IAM 角色。

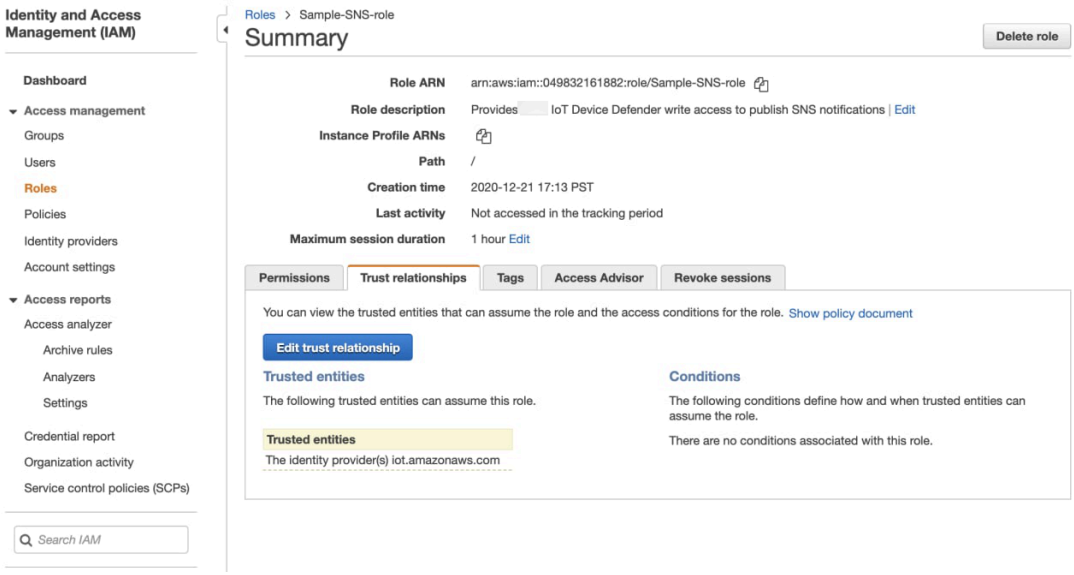
   如果您还没有 SNS 角色，请使用以下步骤创建具有所需的适当权限和信任关系的角色。

   - 导航到 IAM 控制台。在导航窗格中，选择 Roles，然后选择 Create role。
   - 在Select type of trusted entity（选择信任实体的类型）项下，选择 Amazon service（亚马逊云科技服务）。然后，在 Choose a use case（选择使用案例）中，选择 IoT，在 Select your use case（选择您的使用案例）项下，选择 IoT - Device Defender Mitigation Actions（IoT - Device Defender 缓解操作）。完成操作后，选择 Next: Permissions（下一步：权限）。
   - 在“附加权限策略”下，确保AWSIoTDeviceDefenderPublishFindingsToSNSMitigationAction已选中，然后选择“下一步：标签”。

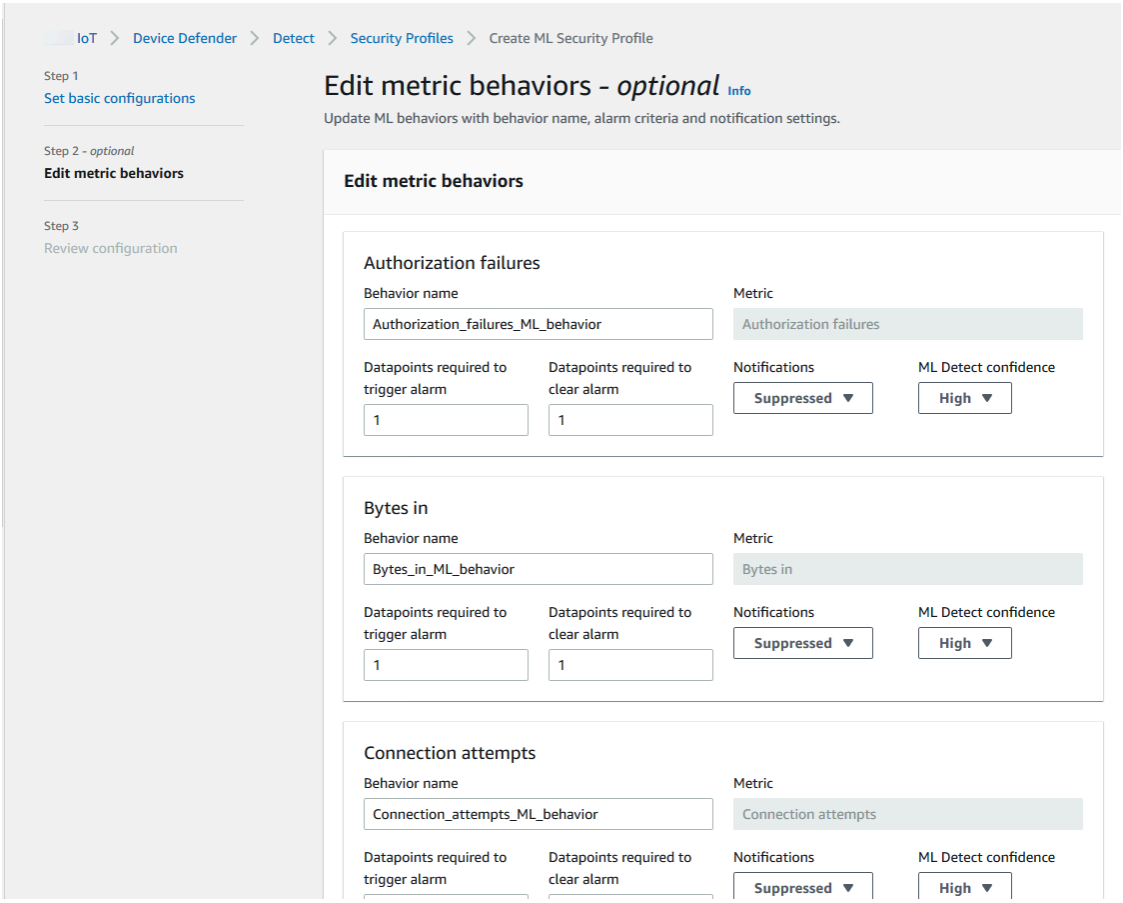


- 在 Add tags (optional)（添加标签（可选））项下，您可以添加要与角色关联的任何标签。完成此操作后，选择 Next: Review (下一步：审核)。
- 在“审核”下，为您的角色命名并确保其列AWSIoTDeviceDefenderPublishFindingsToSNSMitigationAction在“权限和Amazon服务”下：iot.amzonAws.com 列在“信任关系”下。完成后，选择 Create role（创建角色）。

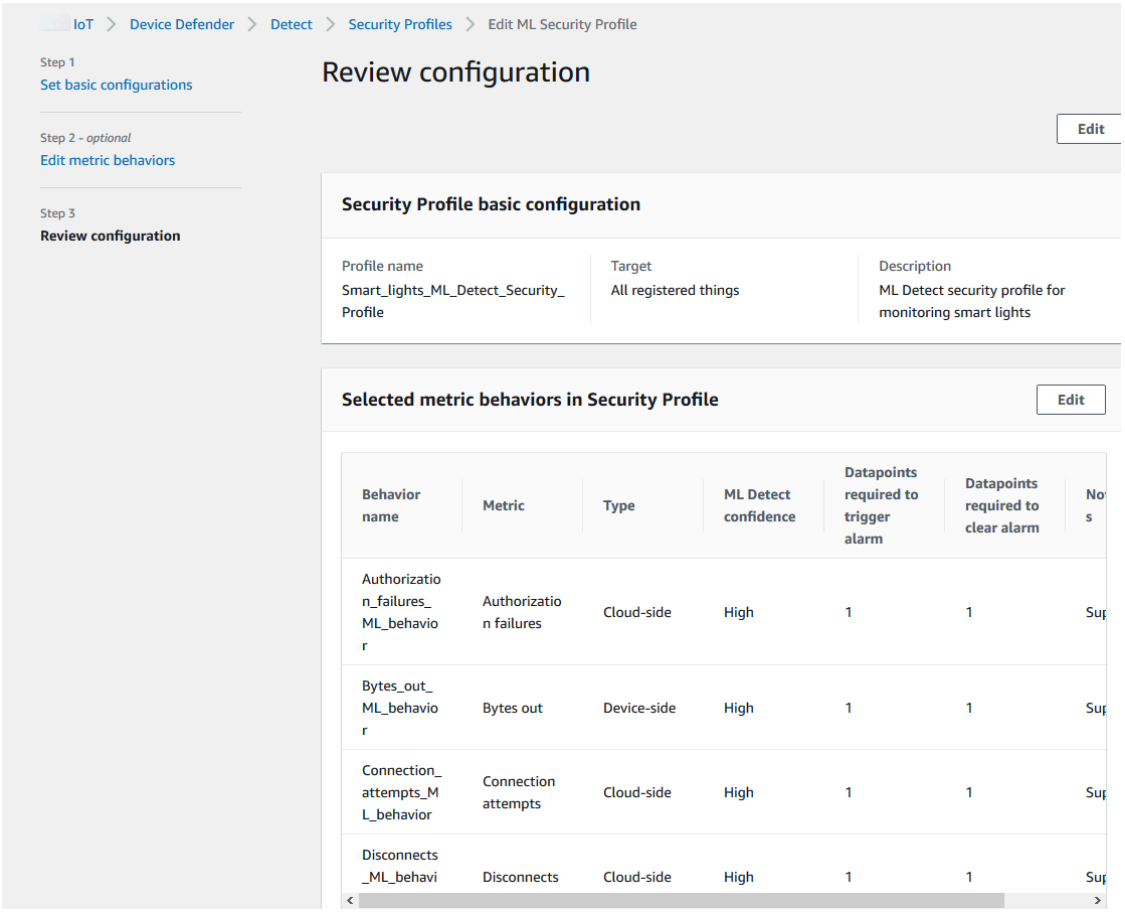




5. 在 Edit Metric behavior（编辑指标行为）页面上，您可以自定义您的机器学习（ML）行为设置。



6. 完成此操作后，选择 Next (下一步)。
7. 在 Review configuration（查看配置）页面上，验证您希望机器学习监控的行为，然后选择 Next（下一步）。



8. 创建了安全配置文件之后，您将会被重定向到 Security Profiles（安全配置文件）页面，此处将显示新创建的安全配置文件。

   Note

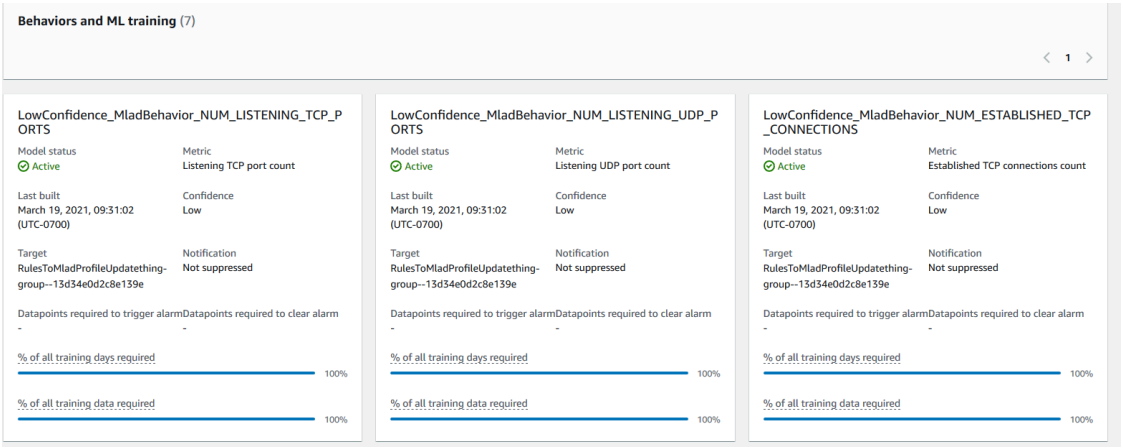
   初始 ML 模型训练和创建需要 14 天才能完成。如果您的设备上存在任何异常活动，则在完成后您应会看到告警。

监控您的 ML 模型状态

当您的 ML 模型处于初始训练阶段时，您可以通过执行以下步骤随时监控其进度。

1. 在 Amazon IoT 控制台中的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Security profiles（安全配置文件）。
2. 在 Security Profiles（安全配置文件）页面上，选择您要查看的安全配置文件。然后，选择 Behaviors and ML training（行为和机器学习（ML）训练）。
3. 在 Behaviors and ML training（行为和机器学习（ML）训练）页面上，检查 ML 模型的训练进度。

   在模型状态为 Active（激活）后，它将开始根据您的使用情况做出 Detect 决策，并每天更新配置文件。



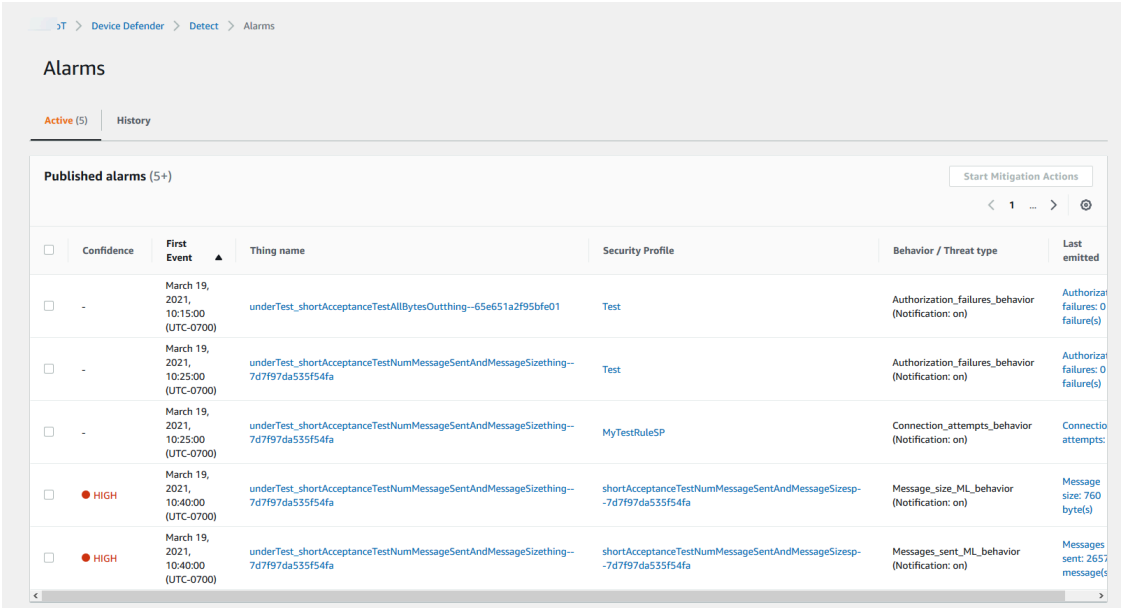
Note

如果您的模型未按预期进行，请确保您的设备满足 最低要求 (p. 956) 的要求。

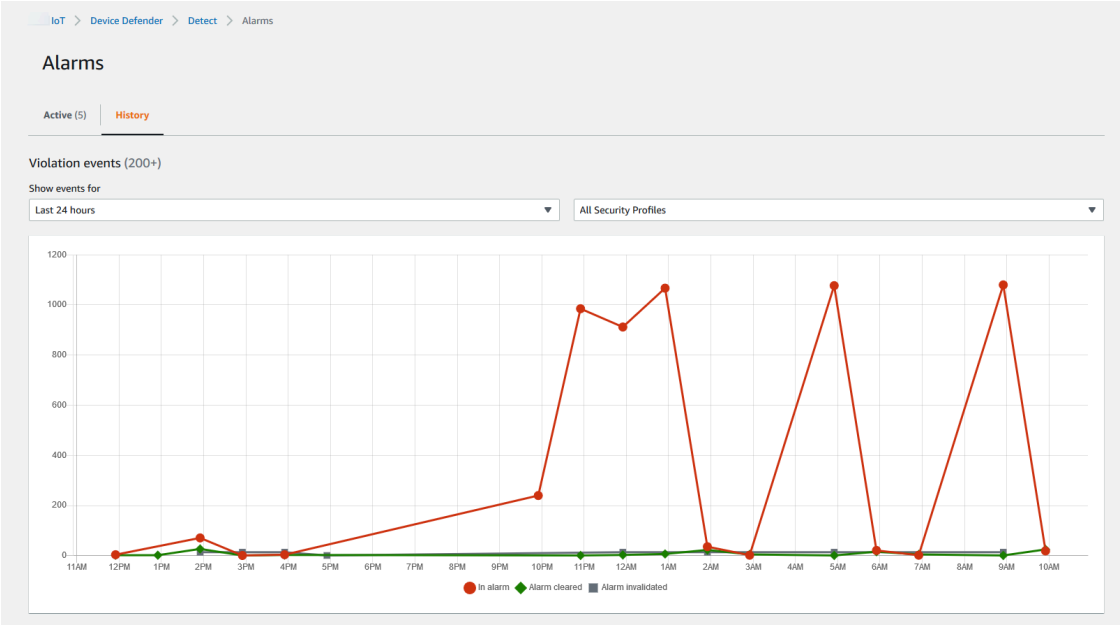
查看您的 ML Detect 告警

在构建 ML 模型并准备进行数据推理之后，您可以定期查看和调查由模型识别的告警。

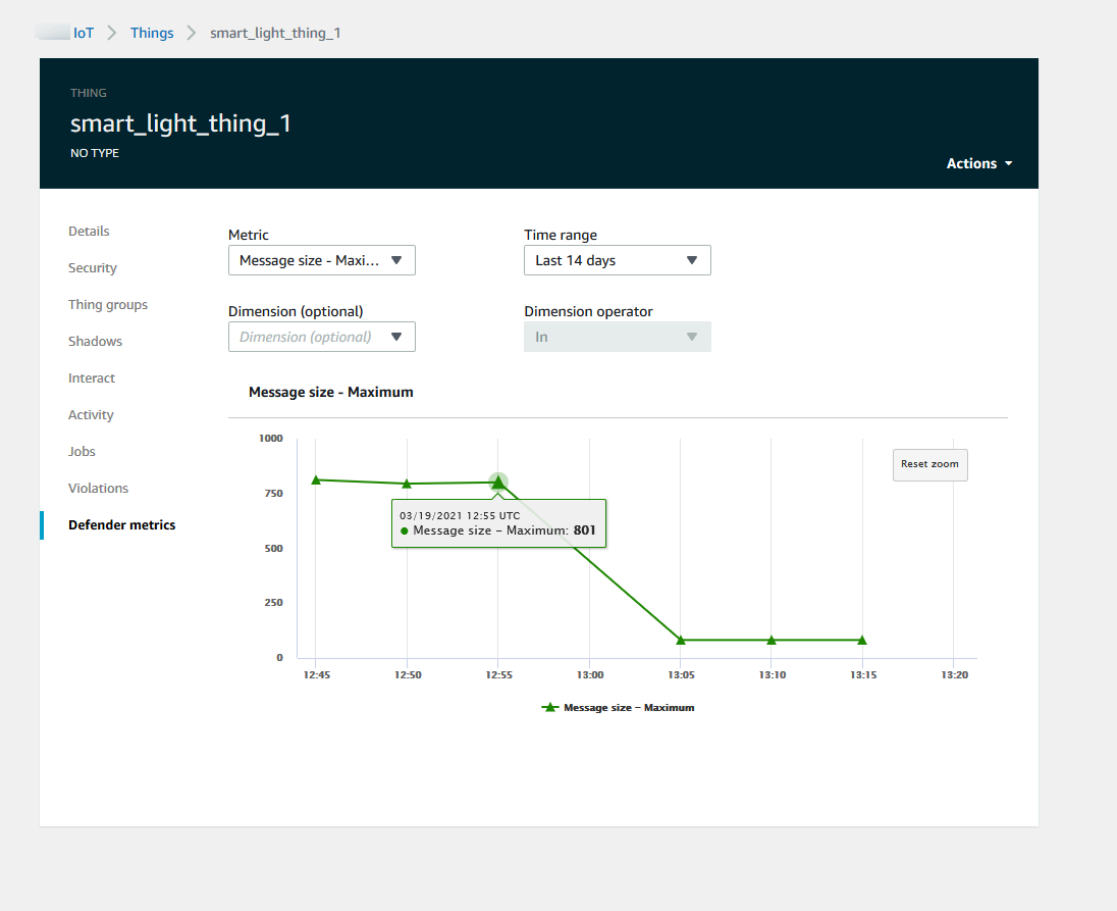
1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Alarms（告警）。



2. 如果您导航到 History（历史记录）选项卡上，您还可以查看不再处于告警状态的设备相关详细信息。



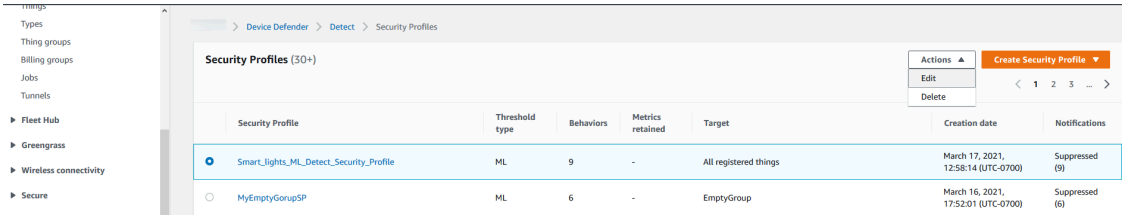
若要获取更多信息，请在 Manage（管理）项下选择 Things（事物），选择您想要查看更多详细信息的事物，然后导航到 Defender metrics（Defender 指标）。您可以访问 Defender metrics graph（Defender 指标图形）并对 Active（激活）选项卡中告警的任何内容进行调查。在这种情况下，图形显示了启动告警的消息大小峰值。您可以看到告警随后被清除。



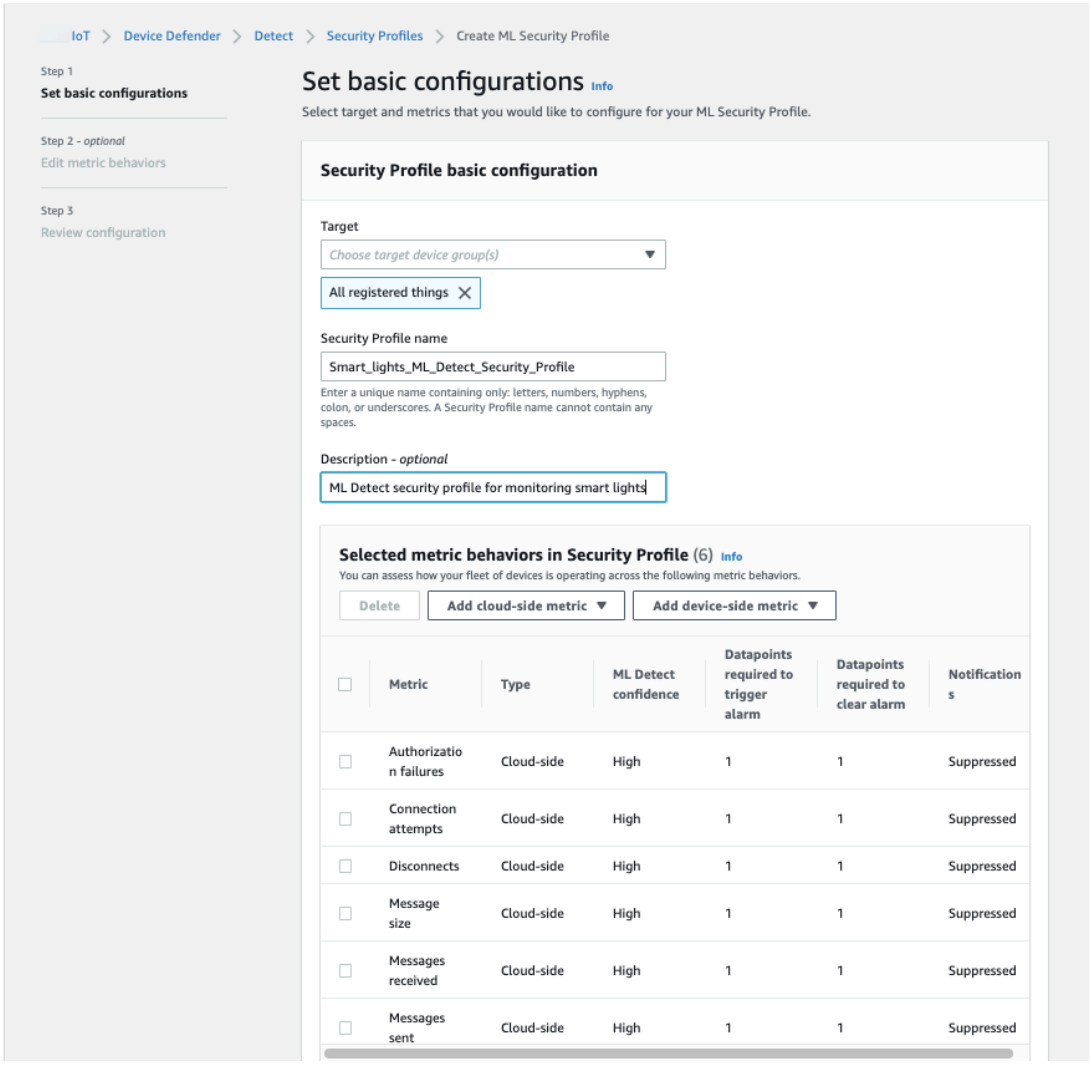
微调您的机器学习（ML）告警

在构建 ML 模型并准备好进行数据评估之后，您可以更新安全配置文件的机器学习（ML）行为设置以更改配置。以下流程介绍如何在 Amazon CLI 中更新您的安全配置文件的机器学习（ML）行为设置。

1. 在 Amazon IoT 控制台中的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Security profiles（安全配置文件）。
2. 在 Security Profiles（安全配置文件）页面上，选中要查看的安全配置文件旁边的复选框。然后依次选择 Actions（操作）、Edit（编辑）。

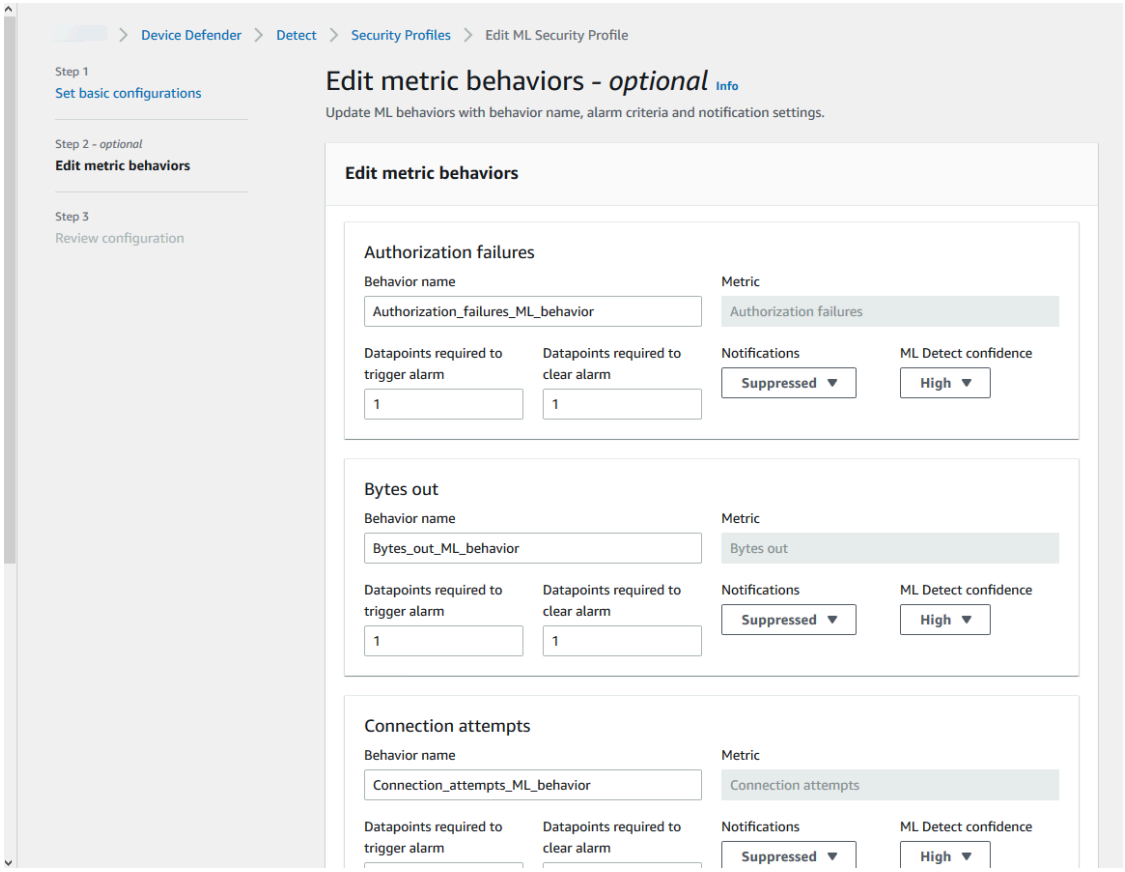


3. 在 Set basic configurations（设置基本配置）项下，您可以调整安全配置文件目标事物组或更改要监控的指标。



4. 您可以通过导航到 Edit metric behaviors（编辑指标行为）更新以下任意一项。

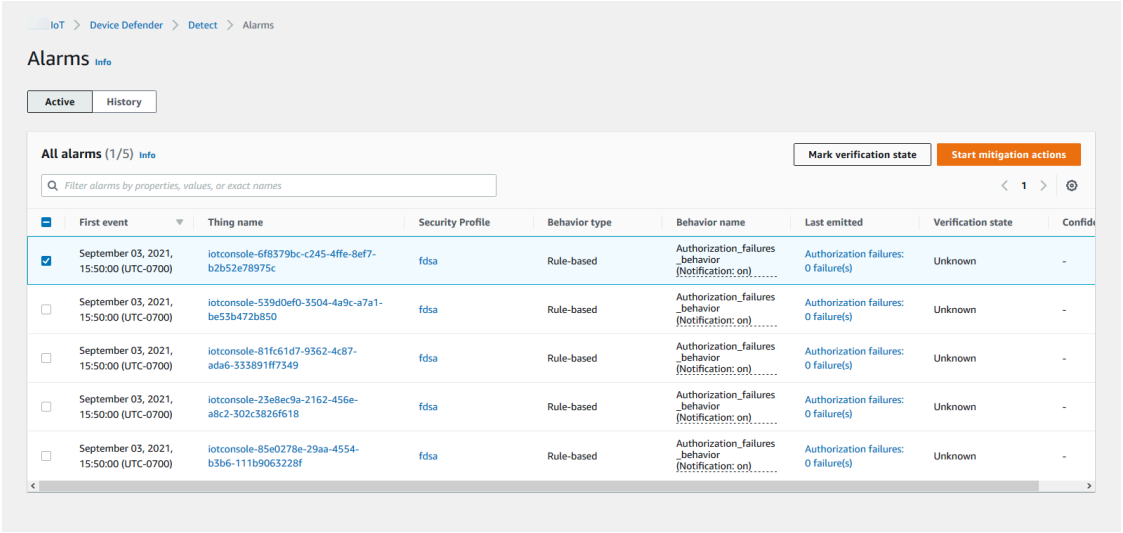
   - 启动告警所需的 ML 模型数据点
   - 清除告警所需的 ML 模型数据点
   - ML Detect 置信级别
   - ML Detect 通知，例如 Not suppressed（未隐藏）、Suppressed（隐藏）。



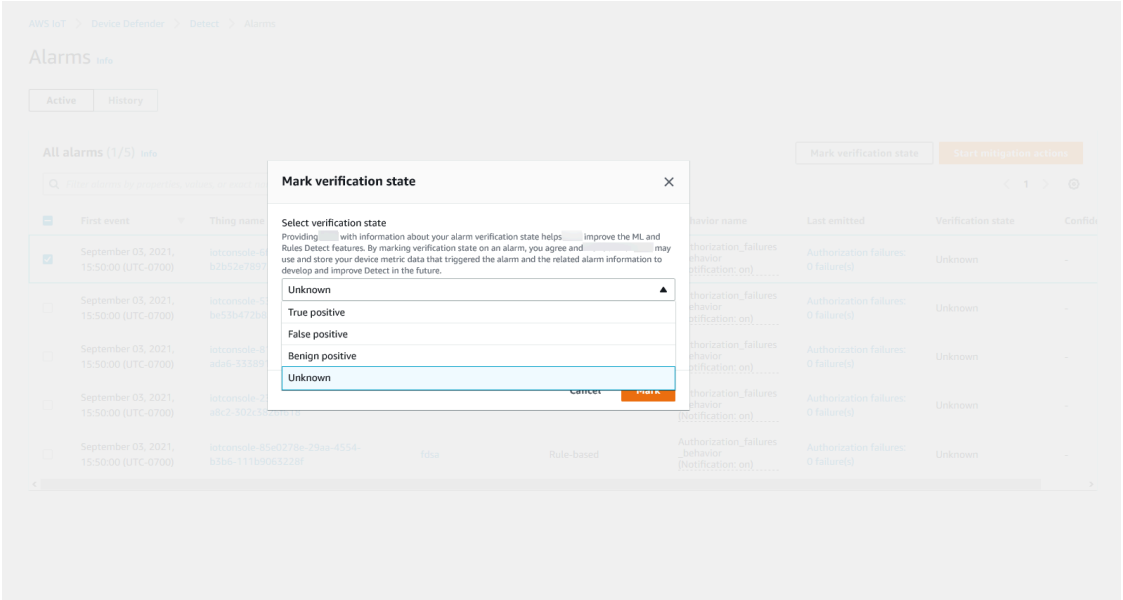
标记告警的验证状态

通过设置验证状态并提供该验证状态的描述来标记告警。这会帮助您和您的团队识别不必回应的告警。

1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Alarms（告警）。选择告警以标记其验证状态。



2. 选择 Mark verification state（标记验证状态）。验证状态模式打开。
3. 选择适当的验证状态，输入验证说明（可选），然后选择 Mark（标记）。此操作将验证状态和描述分配给选定的告警。

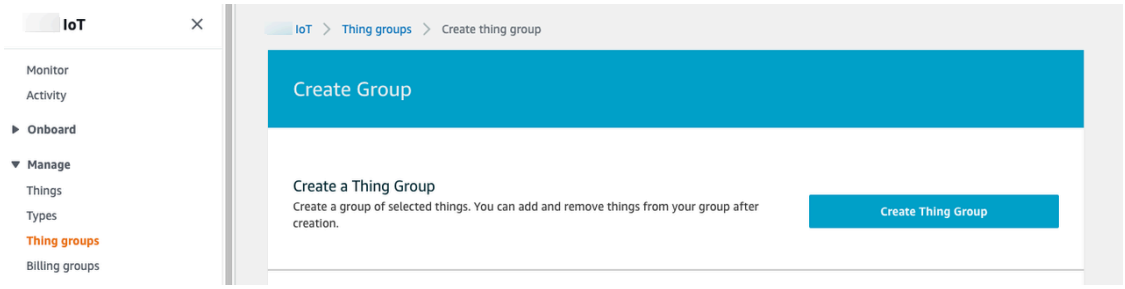


缓解已确定的设备问题

1. （可选）在设置隔离缓解措施之前，让我们设置一个隔离组，我们会将违规的设备移动到该组。您也可以使用现有组。
2. 导航到 Manage（管理）、Thing groups（事物组），然后选择 Create Thing Group（创建事物组）。为您的事物组命名。在本教程中，我们将事物组命名为 `Quarantine_group`。在 Thing Group（事物组）、Security（安全）中，将以下策略应用于事物组。

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

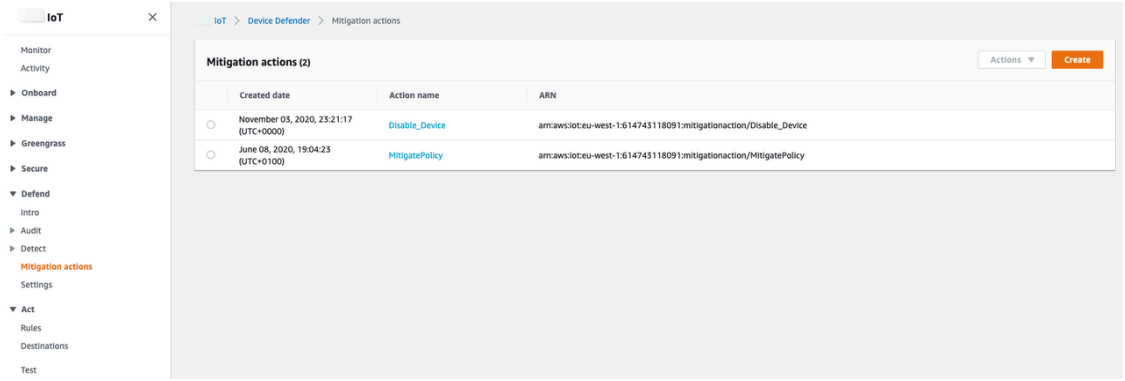
```
      "Action": "iot:*",
      "Resource": "*",
    }
  ]
}
```



完成后，选择 Create thing group（创建事务组）。

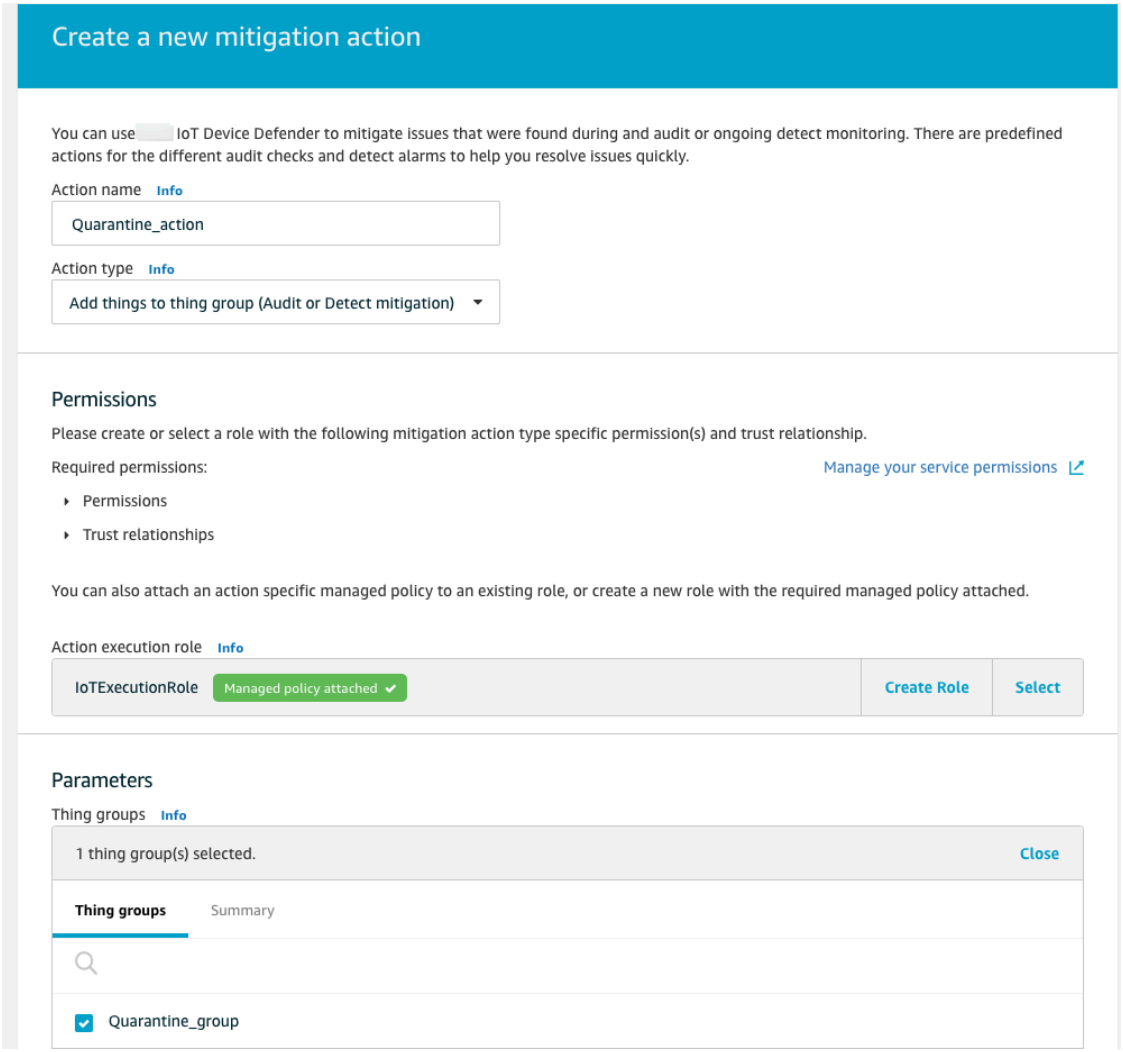
3. 现在我们已经创建了事物组，让我们创建一个缓解操作，将告警中的设备移动到 `Quarantine_group`。

   在 Defend（防护）、Mitigation actions（缓解操作）中，选择 Create（创建）。



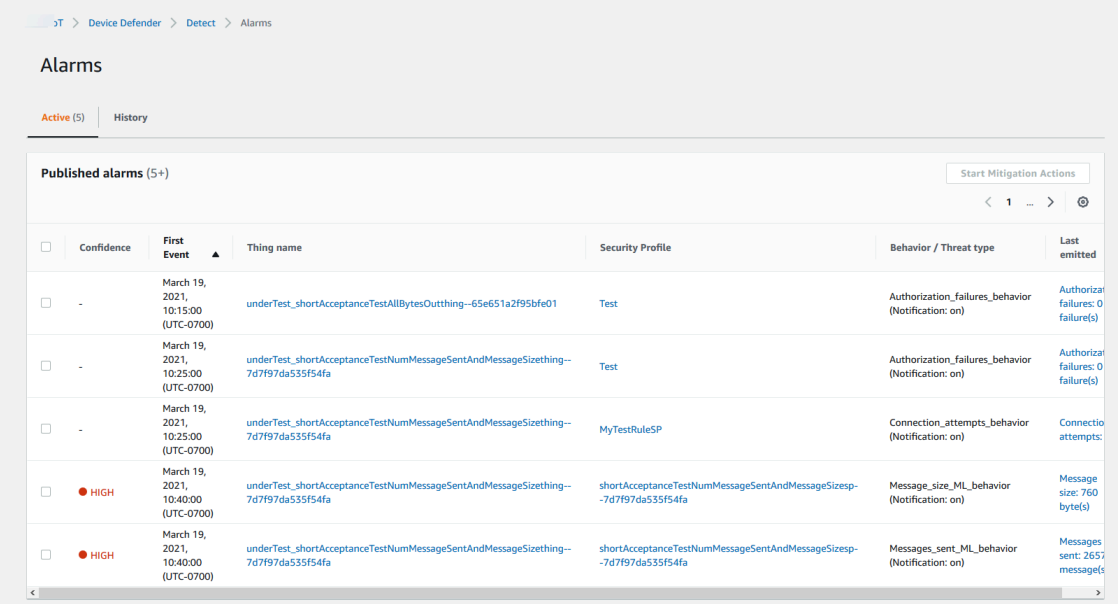
4. 在 Create new mitigation action（创建新的缓解操作）页面上，输入以下信息。

   - Action name（操作名称）：指定缓解操作的名称，例如 **`Quarantine_action`**。
   - Action type（操作类型）：选择操作的类型。我们将选择 Add things to things group (Audit or Detect mitigation)（将事物添加到事物组（审计或 Detect 缓解））。
   - Action execution role（操作执行角色）：创建角色或选择现有角色（如果您之前创建了角色）。
   - Parameters（参数）：选择事物组。我们可以使用之前创建的 `Quarantine_group`。



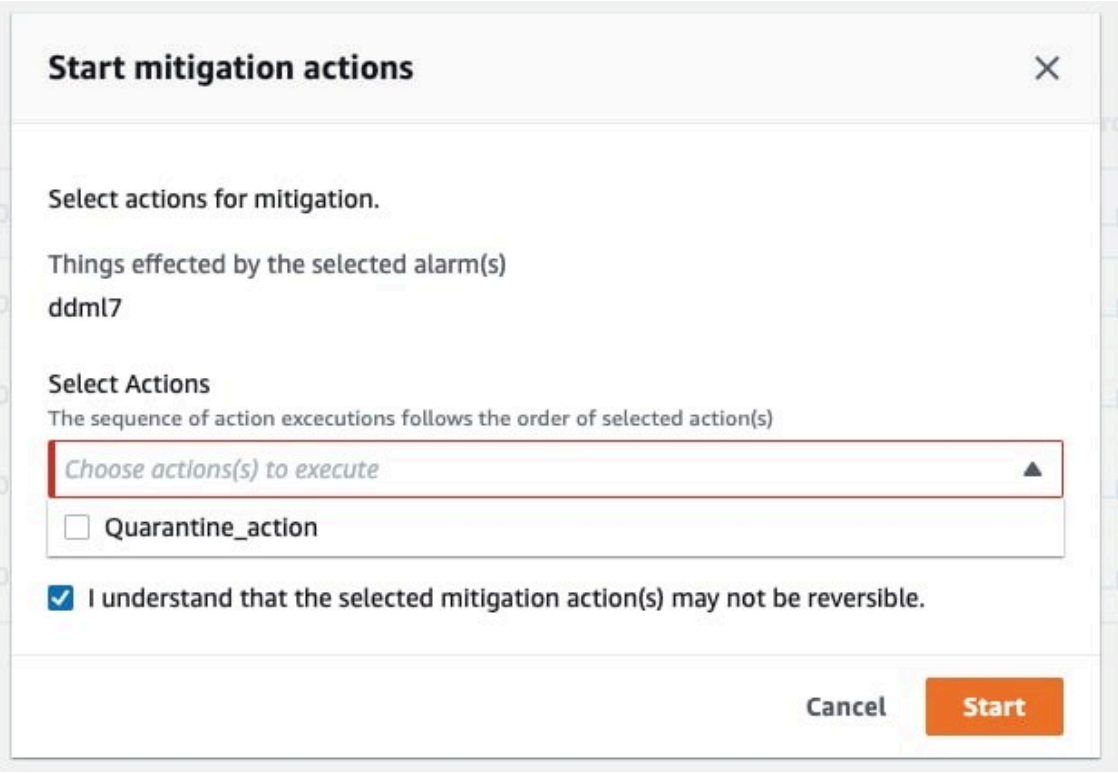
完成此操作后，选择 Save。现在，您可以执行缓解操作，将告警中的设备移动到隔离事物组，并在调查期间执行缓解操作来隔离设备。

5. 导航到 Defender、Detect（检测）、Alarms（告警），您可以查看 Active（激活）状态下有哪些设备处于告警状态。

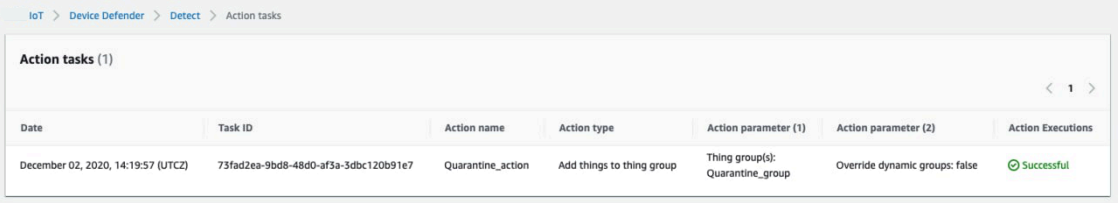


选择要移动到隔离组的设备，然后选择 Start Mitigation Actions（启动缓解操作）。

6. 在 Start mitigation actions（启动缓解操作）、Start Actions（启动操作）项下，选择您之前创建的缓解操作。例如，我们将选择 **`Quarantine_action`**，然后选择 Start（启动）。此时将打开操作任务页面。



7. 设备现在被隔离在 **`Quarantine_group`** 中，您可以调查引发告警的问题的根本原因。完成调查后，您可以将设备移出事物组或采取进一步操作。



如何将 （ ML Detect 与 CLI 一起使用

下面的介绍了如何使用 CLI 设置 ML Detect。

教程

- 启用 ML Detect (p. 870)
- 监控您的 ML 模型状态 (p. 871)
- 查看您的 ML Detect 告警 (p. 873)
- 微调您的机器学习 （ ML ）告警 (p. 874)
- 标记告警的验证状态 (p. 876)
- 缓解已确定的设备问题 (p. 876)

启用 ML Detect

以下流程介绍如何在 Amazon CLI 中启用 ML Detect。

1. 确保您的设备将如 ML Detect 最低要求 (p. 956)中定义创建所需的最小数据点，以进行持续训练和模型更新。为了进行数据收集，请确保您的事物位于附加到安全配置文件的事物组中。
2. 使用 `create-security-profile` 命令创建 ML Detect 安全配置文件。以下示例创建一个名为 *`security-profile-for-smart-`* lights 的安全配置文件，用于检查发送的消息数量、授权失败次数、连接尝试次数以及断开连接次数。此示例使用 `mlDetectionConfig` 以确定该指标将使用 ML Detect 模型。

```
aws iot create-security-profile \
    --security-profile-name security-profile-for-smart-lights \
    --behaviors \
     '[{
    "name": "num-messages-sent-ml-behavior",
    "metric": "aws:num-messages-sent",
    "criteria": {
      "consecutiveDatapointsToAlarm": 1,
      "consecutiveDatapointsToClear": 1,
      "mlDetectionConfig": {
        "confidenceLevel": "HIGH"
      }
    },
    "suppressAlerts": true
  },
  {
    "name": "num-authorization-failures-ml-behavior",
    "metric": "aws:num-authorization-failures",
    "criteria": {
      "consecutiveDatapointsToAlarm": 1,
      "consecutiveDatapointsToClear": 1,
      "mlDetectionConfig": {
        "confidenceLevel": "HIGH"
      }
```

```
    },
    "suppressAlerts": true
  },
  {
    "name": "num-connection-attempts-ml-behavior",
    "metric": "aws:num-connection-attempts",
    "criteria": {
      "consecutiveDatapointsToAlarm": 1,
      "consecutiveDatapointsToClear": 1,
      "mlDetectionConfig": {
        "confidenceLevel": "HIGH"
      }
    },
    "suppressAlerts": true
  },
  {
    "name": "num-disconnects-ml-behavior",
    "metric": "aws:num-disconnects",
    "criteria": {
      "consecutiveDatapointsToAlarm": 1,
      "consecutiveDatapointsToClear": 1,
      "mlDetectionConfig": {
        "confidenceLevel": "HIGH"
      }
    },
    "suppressAlerts": true
  }]'
```

输出：

```
{
    "securityProfileName": "security-profile-for-smart-lights",
    "securityProfileArn": "arn:aws:iot:eu-west-1:123456789012:securityprofile/security-profile-for-smart-lights"
  }
```

3. 接下来，将您的安全配置文件与一个或多个事物组关联。使用 attach-security-profile 命令将事物组附加到您的安全配置文件。*####### ML_Detect_beta_static_Group #####security-profile-for-smart*

```
aws iot attach-security-profile \
--security-profile-name security-profile-for-smart-lights \
--security-profile-target-arn arn:aws:iot:eu-west-1:123456789012:thinggroup/ML_Detect_beta_static_group
```

输出：

无。

4. 创建完整的安全配置文件后，ML 模型将开始训练。初始 ML 模型训练和构建需要 14 天才能完成。14 天后，如果您的设备上有异常活动，则可能会看到告警。

监控您的 ML 模型状态

以下流程介绍如何监控正在进行训练的 ML 模型。

- 使用 get-behavior-model-training-summaries 命令查看 ML 模型的进度。以下示例获取 *security-profile-for-smart-* lights 安全配置文件的 ML 模型训练进度摘要。 modelStatus显示模型是否已完成特定行为的训练或仍在等待构建。

```
aws iot get-behavior-model-training-summaries \
   --security-profile-name security-profile-for-smart-lights
```

输出：

```
{
    "summaries": [
        {
            "securityProfileName": "security-profile-for-smart-lights",
            "behaviorName": "Messages_sent_ML_behavior",
            "trainingDataCollectionStartDate": "2020-11-30T14:00:00-08:00",
            "modelStatus": "ACTIVE",
            "datapointsCollectionPercentage": 29.408,
            "lastModelRefreshDate": "2020-12-07T14:35:19.237000-08:00"
        },
        {
            "securityProfileName": "security-profile-for-smart-lights",
            "behaviorName": "Messages_received_ML_behavior",
            "modelStatus": "PENDING_BUILD",
            "datapointsCollectionPercentage": 0.0
        },
        {
            "securityProfileName": "security-profile-for-smart-lights",
            "behaviorName": "Authorization_failures_ML_behavior",
            "trainingDataCollectionStartDate": "2020-11-30T14:00:00-08:00",
            "modelStatus": "ACTIVE",
            "datapointsCollectionPercentage": 35.464,
            "lastModelRefreshDate": "2020-12-07T14:29:44.396000-08:00"
        },
        {
            "securityProfileName": "security-profile-for-smart-lights",
            "behaviorName": "Message_size_ML_behavior",
            "trainingDataCollectionStartDate": "2020-11-30T14:00:00-08:00",
            "modelStatus": "ACTIVE",
            "datapointsCollectionPercentage": 29.332,
            "lastModelRefreshDate": "2020-12-07T14:30:44.113000-08:00"
        },
        {
            "securityProfileName": "security-profile-for-smart-lights",
            "behaviorName": "Connection_attempts_ML_behavior",
            "trainingDataCollectionStartDate": "2020-11-30T14:00:00-08:00",
            "modelStatus": "ACTIVE",
            "datapointsCollectionPercentage": 32.891999999999996,
            "lastModelRefreshDate": "2020-12-07T14:29:43.121000-08:00"
        },
        {
            "securityProfileName": "security-profile-for-smart-lights",
            "behaviorName": "Disconnects_ML_behavior",
            "trainingDataCollectionStartDate": "2020-11-30T14:00:00-08:00",
            "modelStatus": "ACTIVE",
            "datapointsCollectionPercentage": 35.46,
            "lastModelRefreshDate": "2020-12-07T14:29:55.556000-08:00"
        }
    ]
}
```

Note

如果您的模型未按预期进行，请确保您的设备满足 最低要求 (p. 956) 的要求。

查看您的 ML Detect 告警

在构建 ML 模型并准备好进行数据评估之后，您可以定期查看由模型推断的任何告警。以下流程介绍如何在 Amazon CLI 中查看您的告警。

- 要查看所有活动告警，请使用 list-active-violations 命令。

```
aws iot list-active-violations \
--max-results 2
```

输出：

```
{
    "activeViolations": []
}
```

或者，您可以使用 list-violation-events 命令查看在给定时间内发现的所有违规行为。以下示例列出了从 2020 年 9 月 22 日 5:42:13 GMT 到 2020 年 10 月 26 日 5:42:13 GMT 之间的违规事件。

```
aws iot list-violation-events \
    --start-time 1599500533 \
    --end-time 1600796533 \
    --max-results 2
```

输出：

```
{
    "violationEvents": [
        {
            "violationId": "1448be98c09c3d4ab7cb9b6f3ece65d6",
            "thingName": "lightbulb-1",
            "securityProfileName": "security-profile-for-smart-lights",
            "behavior": {
                "name": "LowConfidence_MladBehavior_MessagesSent",
                "metric": "aws:num-messages-sent",
                "criteria": {
                    "consecutiveDatapointsToAlarm": 1,
                    "consecutiveDatapointsToClear": 1,
                    "mlDetectionConfig": {
                        "confidenceLevel": "HIGH"
                    }
                },
                "suppressAlerts": true
            },
            "violationEventType": "alarm-invalidated",
            "violationEventTime": 1600780245.29
        },
        {
            "violationId": "df4537569ef23efb1c029a433ae84b52",
            "thingName": "lightbulb-2",
            "securityProfileName": "security-profile-for-smart-lights",
            "behavior": {
                "name": "LowConfidence_MladBehavior_MessagesSent",
                "metric": "aws:num-messages-sent",
                "criteria": {
                    "consecutiveDatapointsToAlarm": 1,
                    "consecutiveDatapointsToClear": 1,
                    "mlDetectionConfig": {
                        "confidenceLevel": "HIGH"
                    }
```

```
                },
                "suppressAlerts": true
            },
            "violationEventType": "alarm-invalidated",
            "violationEventTime": 1600780245.281
        }
    ],
    "nextToken":
 "Amo6XIUrsOohsojuIG6TuwSR3X9iUvH2OCksBZg6bed2j21VSnD1uP1pflxKX1+a3cvBRSosIB0xFv40kM6RYBknZ/
vxabMe/ZW31Ps/WiZHlr9Wg7R7eEGli59IJ/U0iBQ1McP/ht0E2XA2TTIvYeMmKQQPsRj/
eoV9j7P/wveu7skNGepU/mvpV0O2Ap7hnV5U+Prx/9+iJA/341va
+pQww7jpUeHmJN9Hw4MqW0ysw0Ry3w38hOQWEpz2xwFWAxAARxeIxCxt5c37RK/lRZBlhYqoB
+w2PZ74730h8pICGY4gktJxkwHyyRabpSM/G/f5DFrD9O5v8idkTZzBxW2jrbzSUIdafPtsZHL/
yAMKr3HAKtaABz2nTsOBNre7X2d/jIjjarhon0Dh9l+8I9Y5Ey
+DIFBcqFTvhibKAafQt3gs6CUiqHdWiCenfJyb8whmDE2qxvdxGElGmRb
+k6kuN5jrZxxw95gzfYDgRHv11iEn8h1qZLD0czkIFBpMppHj9cetHPvM
+qffXGAzKi8tL6eQuCdMLXmVE3jbqcJcjk9ItnaYJi5zKDz9FVbrz9qZZPtZJFHp"
}
```

微调您的机器学习（ML）告警

构建 ML 模型并准备好进行数据评估后，您可以更新安全配置文件的 ML 行为设置以更改配置。以下流程介绍如何在 Amazon CLI 中更新您的安全配置文件的机器学习（ML）行为设置。

- 要更改安全配置文件的 ML 行为设置，请使用 update-security-profile 命令。以下示例通过更改confidenceLevel部分行为并取消隐藏所有行为的通知，从而更新了 *security-profile-for-smart-* lights 安全配置文件的行为。

```
aws iot update-security-profile \
    --security-profile-name security-profile-for-smart-lights \
    --behaviors \
     '[{
      "name": "num-messages-sent-ml-behavior",
      "metric": "aws:num-messages-sent",
      "criteria": {
          "mlDetectionConfig": {
              "confidenceLevel" : "HIGH"
          }
      },
      "suppressAlerts": false
  },
  {
      "name": "num-authorization-failures-ml-behavior",
      "metric": "aws:num-authorization-failures",
      "criteria": {
          "mlDetectionConfig": {
              "confidenceLevel" : "HIGH"
          }
      },
      "suppressAlerts": false
  },
  {
      "name": "num-connection-attempts-ml-behavior",
      "metric": "aws:num-connection-attempts",
      "criteria": {
          "mlDetectionConfig": {
              "confidenceLevel" : "HIGH"
          }
      },
      "suppressAlerts": false
  },
  {
```

```
      "name": "num-disconnects-ml-behavior",
      "metric": "aws:num-disconnects",
      "criteria": {
          "mlDetectionConfig": {
              "confidenceLevel" : "LOW"
          }
      },
      "suppressAlerts": false

  }]'
```

输出：

```
 {
    "securityProfileName": "security-profile-for-smart-lights",
    "securityProfileArn": "arn:aws:iot:eu-west-1:123456789012:securityprofile/security-
profile-for-smart-lights",
    "behaviors": [
        {
            "name": "num-messages-sent-ml-behavior",
            "metric": "aws:num-messages-sent",
            "criteria": {
                "mlDetectionConfig": {
                    "confidenceLevel": "HIGH"
                }
            }
        },
        {
            "name": "num-authorization-failures-ml-behavior",
            "metric": "aws:num-authorization-failures",
            "criteria": {
                "mlDetectionConfig": {
                    "confidenceLevel": "HIGH"
                }
            }
        },
        {
            "name": "num-connection-attempts-ml-behavior",
            "metric": "aws:num-connection-attempts",
            "criteria": {
                "mlDetectionConfig": {
                    "confidenceLevel": "HIGH"
                }
            },
            "suppressAlerts": false
        },
        {
            "name": "num-disconnects-ml-behavior",
            "metric": "aws:num-disconnects",
            "criteria": {
                "mlDetectionConfig": {
                    "confidenceLevel": "LOW"
                }
            },
            "suppressAlerts": true
        }
    ],
    "version": 2,
    "creationDate": 1600799559.249,
    "lastModifiedDate": 1600800516.856
}
```

标记告警的验证状态

您可以使用验证状态标记告警，帮助对告警进行分类并调查异常情况。

- 用验证状态和状态描述标记告警。例如，要将告警的验证状态设置为误报，请使用以下命令：

```
aws iot put-verification-state-on-violation --violation-id 12345  --verification-
state FALSE_POSITIVE --verification-state-description "This is dummy description"  --
endpoint https://us-east-1.iot.amazonaws.com --region us-east-1
```

输出：

无。

缓解已确定的设备问题

1. 使用 create-thing-group 命令为缓解操作创建事物组。在下面的示例中，我们创建了一个名为ThingGroupForDetectMitigationAction。

```
aws iot create-thing-group –thing-group-name ThingGroupForDetectMitigationAction
```

输出：

```
{
 "thingGroupName": "ThingGroupForDetectMitigationAction",
 "thingGroupArn": "arn:aws:iot:us-
east-1:123456789012:thinggroup/ThingGroupForDetectMitigationAction",
 "thingGroupId": "4139cd61-10fa-4c40-b867-0fc6209dca4d"
}
```

2. 接下来，使用 create-mitigation-action 命令创建缓解操作。在以下示例中，我们使用了 IAM 角色的 ARN 创建了一个名为 detect_mitigation_action 的缓解操作，该 IAM 角色专用于应用缓解操作。我们还将定义操作类型和该操作的参数。在这种情况下，我们的缓解措施会将事物移动到我们之前创建的名为 “事物组” ThingGroupForDetectMitigationAction。

```
aws iot create-mitigation-action --action-name detect_mitigation_action \
--role-arn arn:aws:iam::123456789012:role/MitigationActionValidRole \
--action-params \
'{
     "addThingsToThingGroupParams": {
         "thingGroupNames": ["ThingGroupForDetectMitigationAction"],
         "overrideDynamicGroups": false
     }
 }'
```

输出：

```
{
 "actionArn": "arn:aws:iot:us-
east-1:123456789012:mitigationaction/detect_mitigation_action",
 "actionId": "5939e3a0-bf4c-44bb-a547-1ab59ffe67c3"
}
```

3. 使用 start-detect-mitigation-actions-task 命令启动缓解操作任务。task-id、target 和 actions 是必要的参数。

```
aws iot start-detect-mitigation-actions-task \
```

```
    --task-id taskIdForMitigationAction \
    --target '{ "violationIds" : [ "violationId-1", "violationId-2" ] }' \
    --actions "detect_mitigation_action" \
    --include-only-active-violations \
    --include-suppressed-alerts
```

输出：

```
{
    "taskId": "taskIdForMitigationAction"
}
```

4. （可选）要查看任务中包含的缓解操作执行，请使用 list-detect-mitigation-actions-executions 命令。

```
aws iot list-detect-mitigation-actions-executions \
    --task-id taskIdForMitigationAction \
    --max-items 5 \
    --page-size 4
```

输出：

```
{
    "actionsExecutions": [
        {
            "taskId": "e56ee95e - f4e7 - 459 c - b60a - 2701784290 af",
            "violationId": "214_fe0d92d21ee8112a6cf1724049d80",
            "actionName": "underTest_MAThingGroup71232127",
            "thingName": "cancelDetectMitigationActionsTaskd143821b",
            "executionStartDate": "Thu Jan 07 18: 35: 21 UTC 2021",
            "executionEndDate": "Thu Jan 07 18: 35: 21 UTC 2021",
            "status": "SUCCESSFUL",
        }
    ]
}
```

5. （可选）使用 describe-detect-mitigation-actions-task 命令获取有关缓解操作任务的信息。

```
aws iot describe-detect-mitigation-actions-task \
    --task-id taskIdForMitigationAction
```

输出：

```
{
    "taskSummary": {
        "taskId": "taskIdForMitigationAction",
        "taskStatus": "SUCCESSFUL",
        "taskStartTime": 1609988361.224,
        "taskEndTime": 1609988362.281,
        "target": {
            "securityProfileName": "security-profile-for-smart-lights",
            "behaviorName": "num-messages-sent-ml-behavior"
        },
        "violationEventOccurrenceRange": {
            "startTime": 1609986633.0,
            "endTime": 1609987833.0
        },
        "onlyActiveViolationsIncluded": true,
        "suppressedAlertsIncluded": true,
```

```
        "actionsDefinition": [
            {
                "name": "detect_mitigation_action",
                "id": "5939e3a0-bf4c-44bb-a547-1ab59ffe67c3",
                "roleArn": "arn:aws:iam::123456789012:role/MitigatioActionValidRole",
                "actionParams": {
                    "addThingsToThingGroupParams": {
                        "thingGroupNames": [
                            "ThingGroupForDetectMitigationAction"
                        ],
                        "overrideDynamicGroups": false
                    }
                }
            }
        ],
        "taskStatistics": {
            "actionsExecuted": 0,
            "actionsSkipped": 0,
            "actionsFailed": 0
        }
    }
}
```

6. （可选）要获取缓解操作任务的列表，请使用 list-detect-mitigation-actions-tasks 命令。

```
aws iot list-detect-mitigation-actions-tasks \
    --start-time 1609985315 \
    --end-time 1609988915 \
    --max-items 5 \
    --page-size 4
```

输出：

```
{
    "tasks": [
        {
            "taskId": "taskIdForMitigationAction",
            "taskStatus": "SUCCESSFUL",
            "taskStartTime": 1609988361.224,
            "taskEndTime": 1609988362.281,
            "target": {
                "securityProfileName": "security-profile-for-smart-lights",
                "behaviorName": "num-messages-sent-ml-behavior"
            },
            "violationEventOccurrenceRange": {
                "startTime": 1609986633.0,
                "endTime": 1609987833.0
            },
            "onlyActiveViolationsIncluded": true,
            "suppressedAlertsIncluded": true,
            "actionsDefinition": [
                {
                    "name": "detect_mitigation_action",
                    "id": "5939e3a0-bf4c-44bb-a547-1ab59ffe67c3",
                    "roleArn": "arn:aws:iam::123456789012:role/
MitigatioActionValidRole",
                    "actionParams": {
                        "addThingsToThingGroupParams": {
                            "thingGroupNames": [
                                "ThingGroupForDetectMitigationAction"
                            ],
                            "overrideDynamicGroups": false
                        }
                    }
```

```
                }
            ],
            "taskStatistics": {
                "actionsExecuted": 0,
                "actionsSkipped": 0,
                "actionsFailed": 0
            }
        }
    ]
}
```

7. （可选）要取消缓解操作任务，请使用 cancel-detect-mitigation-actions-task 命令。

```
aws iot cancel-detect-mitigation-actions-task \
    --task-id taskIdForMitigationAction
```

输出：

无。

自定义查看 Amazon IoT Device Defender 审计结果的时间和方式

Amazon IoT Device Defender 审计提供定期安全检查以确认 Amazon IoT 设备和资源都遵循了最佳实践。对于每次检查，审计结果都被归类为合规或不合规，其中不合规结果会导致控制台出现警告图标。为了减少重复已知问题产生的噪音，审计查找结果隐藏功能允许您临时将这些不合规通知静音。

您可以在预定时间段内隐藏指定资源或账户的特定审计检查。已隐藏的审计检查结果将被归类为隐藏的查找结果，与合规类别和不合规类别区分开。此新类别不会触发像不合规结果那样的告警。这样，您就可以在已知维护期间或计划完成更新之前减少不合规通知干扰。

开始使用

以下各节详细介绍了如何使用审计查找结果隐藏功能来隐藏控制台和 CLI 中的 `Device certificate expiring` 检查。如果您想遵循其中一个演示，您必须首先创建两个即将到期的证书，以便 Device Defender 检测。

请使用以下命令创建您的证书。

- 创建并注册 CA 证书 (p. 270)
- 使用您的 CA 证书创建客户端证书 在步骤 3 中，将您的 `days` 参数设置为 **1**。

如果您正在使用 CLI 创建证书，请输入以下命令。

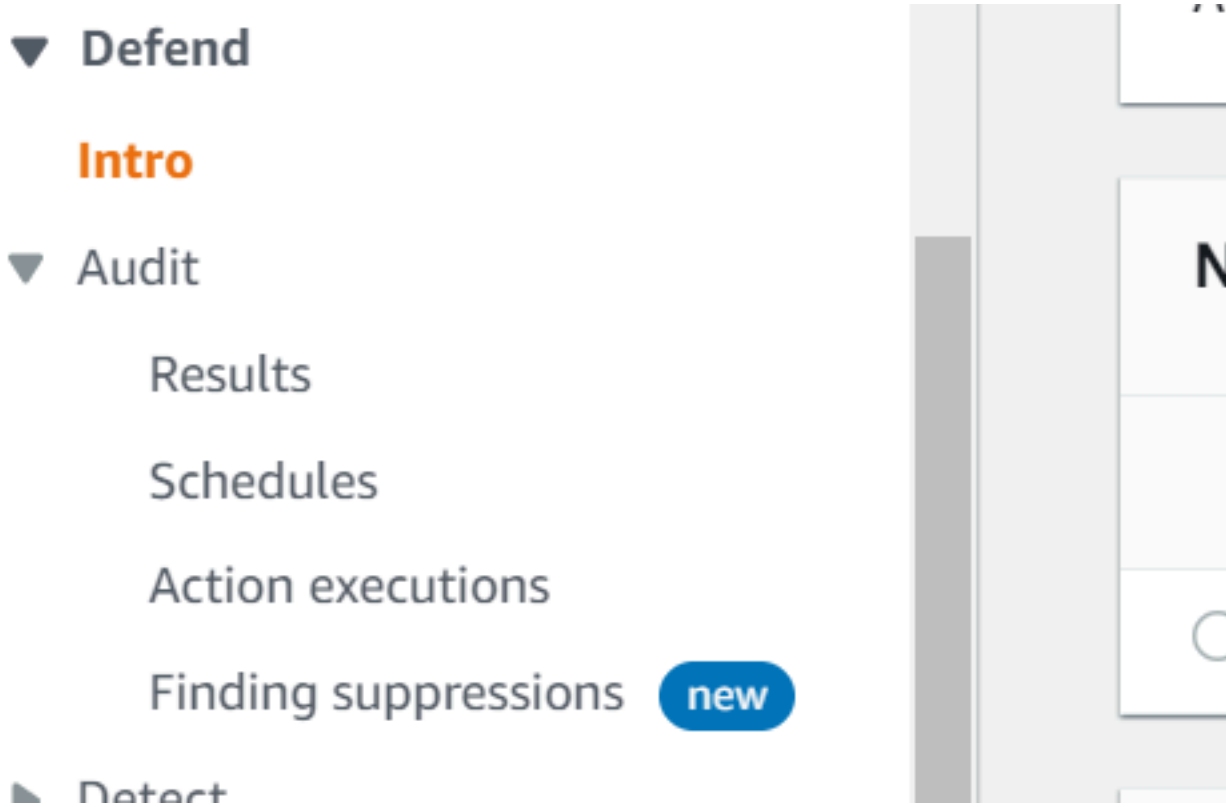
```
openssl x509 -req \
    -in device_cert_csr_filename \
    -CA root_ca_pem_filename \
    -CAkey root_ca_key_filename \
    -CAcreateserial \
    -out device_cert_pem_filename \
    -days 1 -sha256
```

在控制台中自定义审计查找结果

以下演练使用具有两个过期设备证书的账户，这些证书将触发不合规审计检查。在这种情况下，我们希望禁用警告，因为我们的开发人员正在测试一项新功能来解决问题。我们为每个证书创建审计查找结果抑制，以阻止下周审计结果出现不合规的情况。

1. 我们将首先运行按需审计，以显示过期的设备证书检查不合规。

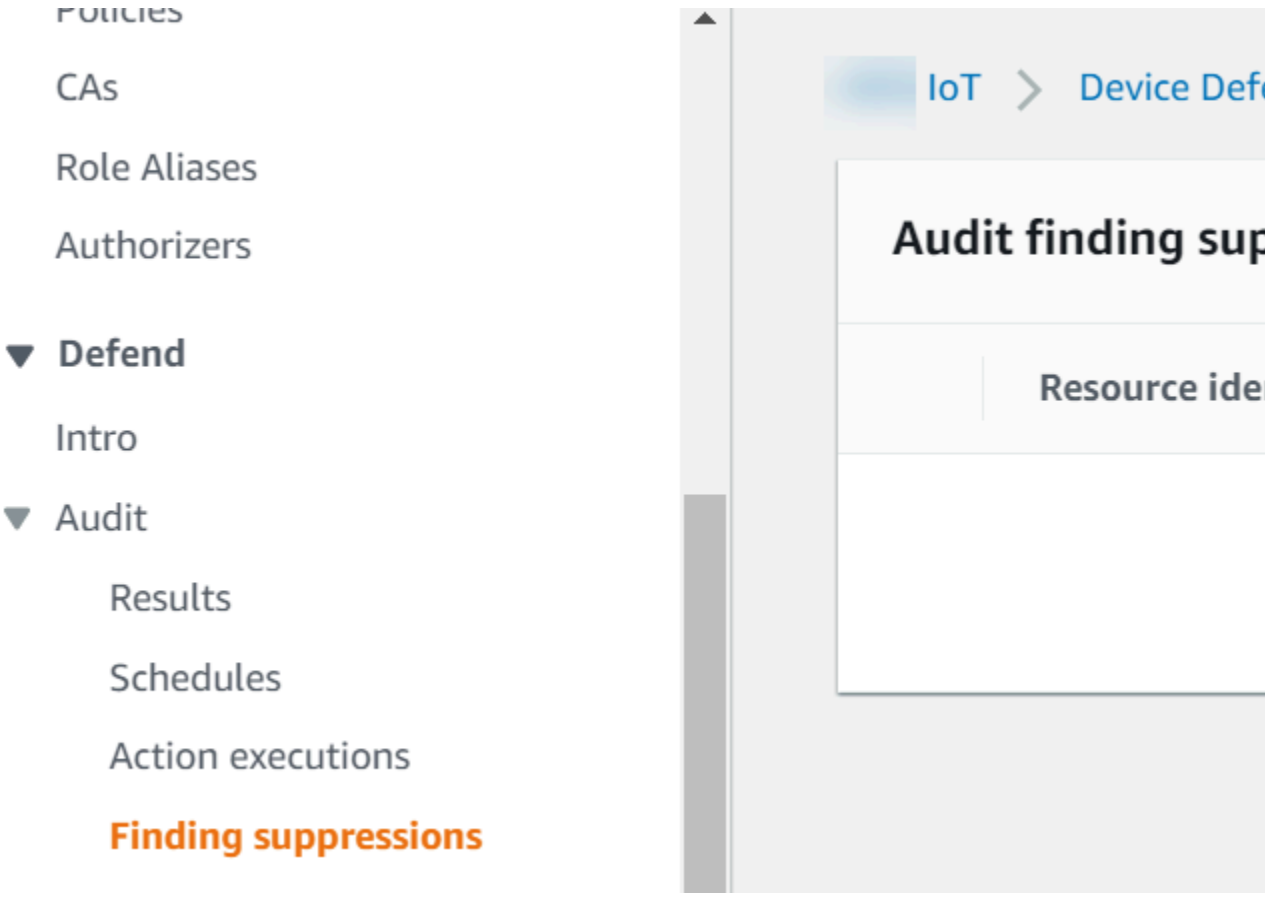
   前往 https://console.aws.amazon.com/ses/ 登入 Amazon Web Services Management Console 並開啟 Amazon SES 主控台。在 Audit Results（审计结果）页面上，选择 Create（创建）。Create a new audit（创建新审计）窗口将会打开。选择创建。



   从按需审计结果中，我们可以看到两个资源的“设备证书即将到期”不合规。

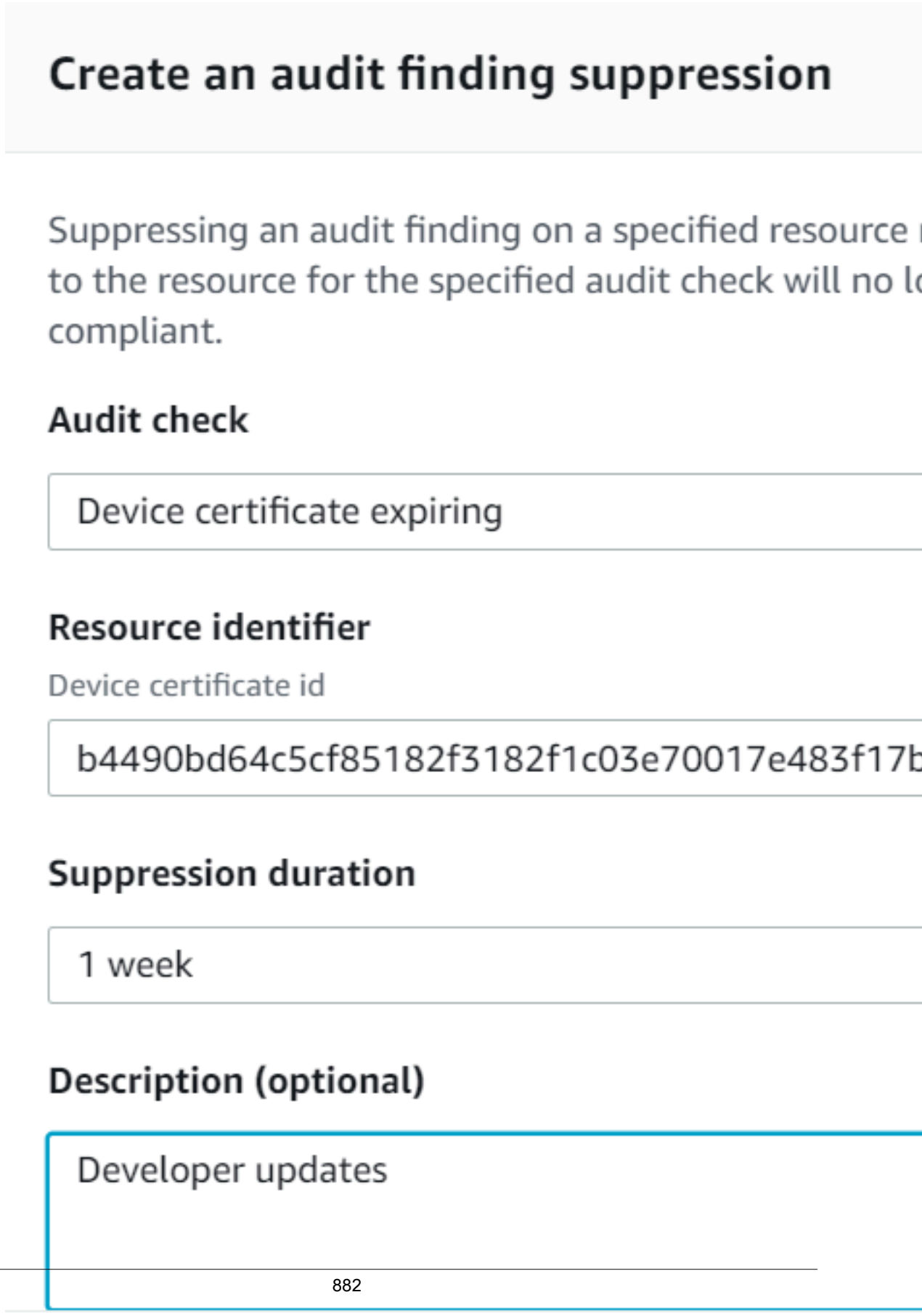
2. 现在，我们希望禁用“设备证书即将过期”的不合规检查警告，因为我们的开发人员正在测试新功能来修复警告。

   从左侧边栏的 Defend（防护）下，选择 Audit（审计），然后选择 Finding suppressions（查找结果隐藏）。在 Audit finding suppressions（审计查找结果隐藏）页面上，选择 Create（创建）。



3. 在 Create an audit finding suppression（创建审计查找结果隐藏）窗口中，我们需要填写以下内容。

   - Audit check:（审计检查）：我们选择 `Device certificate expiring`，因为这是我们想要隐藏的审计检查。
   - Resource identifier（资源标识符）：我们输入希望隐藏审计查找结果的证书之一的设备证书 ID。
   - Suppression duration（隐藏时长）：我们选择 `1 week`，因为这是我们希望将 `Device certificate expiring` 审计检查隐藏的时长。
   - Description (optional)（描述（可选））：我们添加了一个说明，描述了为什么我们要隐藏此审计查找结果。

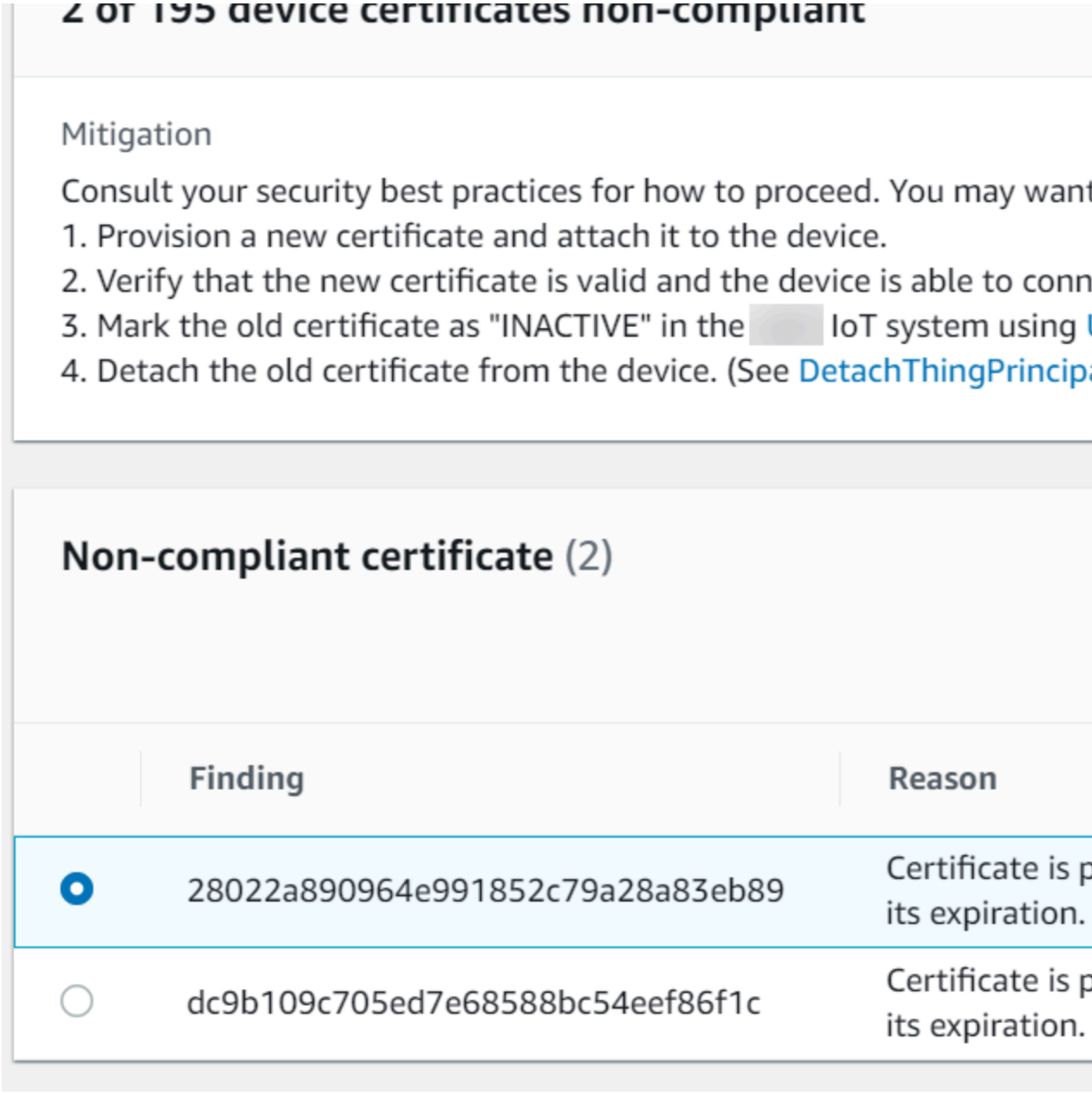

Create an audit finding suppression
Suppressing an audit finding on a specified resource
to the resource for the specified audit check will no l
compliant.
Audit check
Device certificate expiring
Resource identifier
Device certificate id
b4490bd64c5cf85182f3182f1c03e70017e483f17b
Suppression duration
1 week
Description (optional)
Developer updates

填写完这些字段后，选择 Create（创建）。创建审计查找结果隐藏后，我们会看到一个成功的提示。

4. 我们隐藏了其中一个证书的审计查找结果，现在我们需要隐藏第二个证书的审计查找结果。我们可以使用与步骤 3 中相同的隐藏方法，但此次我们将使用不同的方法进行演示。

   从 Defend（防护）的左侧边栏中，选择 Audit（审计），然后选择 Results（结果）。在 Audit results（审计结果）页面上，选择包含不合规资源的审计。然后，在 Non-compliant checks（不合规检查）项下选择资源。在我们的例子中，我们选择“设备证书即将到期”。

5. 在 Device certificate expiring（设备证书即将到期）页面，在 Non-compliant policy（不合规的策略）项下选择需要隐藏的查找结果旁的选项按钮。接下来，选择 Actions（操作）下拉菜单，然后选择为要隐藏的查找结果选择时长。在我们的例子中，我们跟其它证书一样选择 1 week。在 Confirm suppression（确认隐藏）窗口中，选择 Enable suppression（启用隐藏）。



创建审计查找结果隐藏后，我们会看到一个成功的提示。现在，两个审计查找结果均已被隐藏 1 周，同时我们的开发人员正在研制解决方案来解决警告。

在 CLI 中自定义您的审计查找结果

以下演练使用具有过期设备证书并触发了不合规审计检查的账户。在这种情况下，我们希望禁用警告，因为我们的开发人员正在测试一项新功能来解决问题。我们为证书创建审计查找结果隐藏，以防下周的审计结果出现不合规的情况。

我们将使用以下 CLI 命令。

- create-audit-suppression
- describe-audit-suppression
- update-audit-suppression
- delete-audit-suppression
- list-audit-suppressions

1. 使用以下命令来启用审计。

```
aws iot update-account-audit-configuration \
   --audit-check-configurations "{\"DEVICE_CERTIFICATE_EXPIRING_CHECK\":{\"enabled
\":true}}"
```

输出：

无。

2. 使用以下命令运行按需审计，以针对 DEVICE_CERTIFICATE_EXPIRING_CHECK 审计检查。

```
aws iot start-on-demand-audit-task \
    --target-check-names DEVICE_CERTIFICATE_EXPIRING_CHECK
```

输出：

```
{
    "taskId": "787ed873b69cb4d6cdbae6ddd06996c5"
}
```

3. 使用 describe-account-audit-configuration 命令来描述审计配置。我们希望确认我们已开启 DEVICE_CERTIFICATE_EXPIRING_CHECK 的审计检查。

```
aws iot describe-account-audit-configuration
```

输出：

```
{
    "roleArn": "arn:aws:iam::<accountid>:role/service-role/project",
    "auditNotificationTargetConfigurations": {
        "SNS": {
            "targetArn": "arn:aws:sns:us-east-1:<accountid>:project_sns",
            "roleArn": "arn:aws:iam::<accountid>:role/service-role/project",
            "enabled": true
        }
```

```
    },
    "auditCheckConfigurations": {
        "AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK": {
            "enabled": false
        },
        "CA_CERTIFICATE_EXPIRING_CHECK": {
            "enabled": false
        },
        "CA_CERTIFICATE_KEY_QUALITY_CHECK": {
            "enabled": false
        },
        "CONFLICTING_CLIENT_IDS_CHECK": {
            "enabled": false
        },
        "DEVICE_CERTIFICATE_EXPIRING_CHECK": {
            "enabled": true
        },
        "DEVICE_CERTIFICATE_KEY_QUALITY_CHECK": {
            "enabled": false
        },
        "DEVICE_CERTIFICATE_SHARED_CHECK": {
            "enabled": false
        },
        "IOT_POLICY_OVERLY_PERMISSIVE_CHECK": {
            "enabled": true
        },
        "IOT_ROLE_ALIAS_ALLOWS_ACCESS_TO_UNUSED_SERVICES_CHECK": {
            "enabled": false
        },
        "IOT_ROLE_ALIAS_OVERLY_PERMISSIVE_CHECK": {
            "enabled": false
        },
        "LOGGING_DISABLED_CHECK": {
            "enabled": false
        },
        "REVOKED_CA_CERTIFICATE_STILL_ACTIVE_CHECK": {
            "enabled": false
        },
        "REVOKED_DEVICE_CERTIFICATE_STILL_ACTIVE_CHECK": {
            "enabled": false
        },
        "UNAUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK": {
            "enabled": false
        }
    }
}
```

DEVICE_CERTIFICATE_EXPIRING_CHECK 应具有 值。true

4. 使用 list-audit-task 命令来标识已完成的审计任务。

```
aws iot list-audit-tasks \
    --task-status "COMPLETED" \
    --start-time 2020-07-31 \
    --end-time 2020-08-01
```

输出：

```
{
    "tasks": [
        {
            "taskId": "787ed873b69cb4d6cdbae6ddd06996c5",
            "taskStatus": "COMPLETED",
```

```
            "taskType": "SCHEDULED_AUDIT_TASK"
        }
    ]
}
```

您在步骤 1 中运行的审计的 taskId 应该具有一个为 COMPLETED 的 taskStatus。

5. 使用 describe-audit-task 命令，获取有关使用来自前述步骤的 taskId 输出完成的审计的详细信息。此命令列出了有关审计的详细信息。

```
aws iot describe-audit-task \
    --task-id "787ed873b69cb4d6cdbae6ddd06996c5"
```

输出：

```
{
    "taskStatus": "COMPLETED",
    "taskType": "SCHEDULED_AUDIT_TASK",
    "taskStartTime": 1596168096.157,
    "taskStatistics": {
        "totalChecks": 1,
        "inProgressChecks": 0,
        "waitingForDataCollectionChecks": 0,
        "compliantChecks": 0,
        "nonCompliantChecks": 1,
        "failedChecks": 0,
        "canceledChecks": 0
    },
    "scheduledAuditName": "AWSIoTDeviceDefenderDailyAudit",
    "auditDetails": {
        "DEVICE_CERTIFICATE_EXPIRING_CHECK": {
            "checkRunStatus": "COMPLETED_NON_COMPLIANT",
            "checkCompliant": false,
            "totalResourcesCount": 195,
            "nonCompliantResourcesCount": 2
        }
    }
}
```

6. 使用 list-audit-findings 命令查找不合规的证书 ID，以便我们可以暂停此资源的审计提示。

```
aws iot list-audit-findings \
    --start-time 2020-07-31 \
    --end-time 2020-08-01
```

输出：

```
{
    "findings": [
        {
            "findingId": "296ccd39f806bf9d8f8de20d0ceb33a1",
            "taskId": "787ed873b69cb4d6cdbae6ddd06996c5",
            "checkName": "DEVICE_CERTIFICATE_EXPIRING_CHECK",
            "taskStartTime": 1596168096.157,
            "findingTime": 1596168096.651,
            "severity": "MEDIUM",
            "nonCompliantResource": {
                "resourceType": "DEVICE_CERTIFICATE",
                "resourceIdentifier": {
                    "deviceCertificateId": "b4490<shortened>"
                },
```

```
                "additionalInfo": {
                "EXPIRATION_TIME": "1582862626000"
                }
            },
            "reasonForNonCompliance": "Certificate is past its expiration.",
            "reasonForNonComplianceCode": "CERTIFICATE_PAST_EXPIRATION",
            "isSuppressed": false
        },
        {
            "findingId": "37ecb79b7afb53deb328ec78e647631c",
            "taskId": "787ed873b69cb4d6cdbae6ddd06996c5",
            "checkName": "DEVICE_CERTIFICATE_EXPIRING_CHECK",
            "taskStartTime": 1596168096.157,
            "findingTime": 1596168096.651,
            "severity": "MEDIUM",
            "nonCompliantResource": {
                "resourceType": "DEVICE_CERTIFICATE",
                "resourceIdentifier": {
                    "deviceCertificateId": "c7691<shortened>"
                },
                "additionalInfo": {
                "EXPIRATION_TIME": "1583424717000"
                }
            },
            "reasonForNonCompliance": "Certificate is past its expiration.",
            "reasonForNonComplianceCode": "CERTIFICATE_PAST_EXPIRATION",
            "isSuppressed": false
        }
    ]
}
```

7. 使用 create-audit-suppression 命令将 ID 为 *c7691e<shortened>* 的设备证书隐藏 DEVICE_CERTIFICATE_EXPIRING_CHECK 审计检查的通知，直至 *2020-08-20*。

```
aws iot create-audit-suppression \
    --check-name DEVICE_CERTIFICATE_EXPIRING_CHECK \
    --resource-identifier deviceCertificateId="c7691e<shortened>" \
    --no-suppress-indefinitely \
    --expiration-date 2020-08-20
```

8. 使用 list-audit-suppression 命令确认审计隐藏设置并获取有关隐藏的详细信息。

```
aws iot list-audit-suppressions
```

输出：

```
{
    "suppressions": [
        {
        "checkName": "DEVICE_CERTIFICATE_EXPIRING_CHECK",
            "resourceIdentifier": {
                "deviceCertificateId": "c7691e<shortened>"
            },
        "expirationDate": 1597881600.0,
        "suppressIndefinitely": false
        }
    ]
}
```

9. update-audit-suppression 命令可用于更新审计查找结果隐藏。下面的示例将 expiration-date 更新为了 08/21/20。

```
aws iot update-audit-suppression \
    --check-name DEVICE_CERTIFICATE_EXPIRING_CHECK \
    --resource-identifier deviceCertificateId=c7691e<shortened> \
    --no-suppress-indefinitely \
    --expiration-date 2020-08-21
```

10. delete-audit-suppression 命令可用于删除审计查找结果隐藏。

```
aws iot delete-audit-suppression \
    --check-name DEVICE_CERTIFICATE_EXPIRING_CHECK \
    --resource-identifier deviceCertificateId="c7691e<shortened>"
```

要确认删除，请使用 list-audit-suppressions 命令。

```
aws iot list-audit-suppressions
```

输出：

```
{
 "suppressions": []
}
```

在本教程中，我们向您展示了如何在控制台和 CLI 中隐藏 `Device certificate expiring` 检查。有关审计查找结果隐藏的更多信息，请参阅 审计查找结果隐藏 (p. 938)。

审核

Amazon IoT Device Defender 审核会查看与账户相关的和与设备相关的设置及策略，以确保安全措施就绪。审核可帮助您检测与安全最佳实践或访问策略的任何偏离（例如，使用同一身份的多个设备，或允许一个设备读取和更新许多其它设备数据的过度宽松的权限策略。）您可以根据需要运行审核（按需审核）或安排审核定期运行（计划审核）。

Amazon IoT Device Defender 审核会针对常见的 IoT 安全最佳实践和设备漏洞运行一组预定义检查。预定义检查示例包括，授权读取或更新多个设备上数据的策略、共享身份（X.509 证书）的设备或者即将过期或已吊销但仍处于活动状态的证书。

问题严重性

问题严重性指示与每个已标识的不合规实例相关的关注级别，以及建议的补救时间。

重大

具有此严重性的不合规审计检查将标识需要迫切关注的问题。关键问题通常使得只掌握了少量高级知识且没有中间人知识或特殊凭证的不良行为者，可以轻松获取您资产的访问或控制权限。

高

具有此严重性的不合规审计检查需要立即进行调查，并在解决关键问题之后提供补救计划。与关键问题一样，高严重性问题通常会向不良行为者提供对您资产的访问或控制权限。但是，高严重性问题通常更难以利用。它们可能需要特殊的工具、中间人知识或特定的设置。

中

具有此严重性的不合规审计检查会提出问题，需要在持续的安保状况维护过程中加以注意。中等严重性问题可能会导致负面操作影响，例如由于安全控制故障而导致的意外中断。这些问题也可能向不良行为者提供对您资产的有限访问或控制，或者可能有助于其部分恶意行为的实施。

低

具有此严重性的不合规审计检查通常表明忽略或绕过了安全最佳实践。虽然它们本身可能不会对安全造成直接影响，但这些失误可能被不良行为者利用。与中等严重性问题一样，低严重性问题需要在持续的安全状况维护过程中加以注意。

后续步骤

要了解可执行的审计检查类型，请参阅 审核检查 (p. 889)。有关适用于审计的服务配额的信息，请参阅 Service Quotas。

审核检查

Note

当您启用检查后，数据收集就会立即开始。如果账户中有大量数据需要收集，那么在启用检查之后可能需要等待一定时间才会获得结果。

以下为受支持的审核检查：

- 已吊销中间 CA 以进行活动设备证书检查 (p. 889)
- 已吊销的 CA 证书仍处于活动状态 (p. 890)
- 共享设备证书 (p. 891)
- 设备证书密钥质量 (p. 891)
- CA 证书密钥质量 (p. 892)
- 未经身份验证的 Cognito 角色过于宽容 (p. 893)
- 经过身份验证的 Cognito 角色过于宽容 (p. 898)
- Amazon IoT 策略过于宽容 (p. 904)
- Amazon IoT 策略可能配置错误 (p. 908)
- CA 证书即将过期 (p. 911)
- 冲突的 MQTT 客户端 ID (p. 911)
- 设备证书即将过期 (p. 912)
- 已撤销的设备证书仍处于激活状态 (p. 913)
- 日志记录已禁用 (p. 913)

已吊销中间 CA 以进行活动设备证书检查

使用此检查来识别尽管撤消了中间 CA，但仍处于活动状态的所有相关设备证书。

此检查在 CLI 和 API 中显示为 `INTERMEDIATE_CA_REVOKED_FOR_ACTIVE_DEVICE_CERTIFICATES_CHECK`。

严重性：危急

详细信息

此检查发现不合规问题时，会返回以下原因代码：

- INTERMEDIATE_CA_REVOKED_BY_ISSUER

为什么这非常重要

已吊销中间 CA 以进行活动设备证书检查通过确定 CA 链中已吊销中间颁发 CA 的 Amazon IoT Core 中是否存在活动设备证书来评估设备身份和信任度。

不应再使用已吊销的中间 CA 来签署 CA 链中的任何其他 CA 或设备证书。在吊销中间 CA 后，具有使用此 CA 证书签名的证书的新添加设备将构成安全威胁。

如何修复

查看 CA 证书被吊销后一段时间的设备证书注册活动。按照您的安全最佳做法来缓解这种情况。您可能需要：

1. 为受影响的设备配置由其他 CA 签名的新证书。
2. 验证新证书是否有效，以及设备能否使用新证书进行连接。
3. 用于在中UpdateCertificate将旧证书标记为 “已撤销”Amazon IoT。您还可以使用缓解操作实现以下目的：
   - 对您的审计查找结果应用 UPDATE_DEVICE_CERTIFICATE 缓解操作以进行此更改。
   - 应用 ADD_THINGS_TO_THING_GROUP 缓解操作，以将设备添加到可以对其执行操作的组。
   - 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 PUBLISH_FINDINGS_TO_SNS 缓解操作。
   - 对吊销中间 CA 证书后的设备证书注册活动进行审核，并考虑吊销在此期间可能使用它颁发的任何设备证书。可以ListRelatedResourcesForAuditFinding使用 CA 证书签发的设备证书和UpdateCertificate吊销设备证书。
   - 将旧证书从设备分离。（请参见 DetachThingPrincipal。）

   有关更多信息，请参阅缓解操作 (p. 995)：

已吊销的 CA 证书仍处于活动状态

CA 证书已被吊销，但在 Amazon IoT 中仍处于活动状态。

此检查在 CLI 和 API 中显示为 REVOKED_CA_CERTIFICATE_STILL_ACTIVE_CHECK。

严重性：危急

详细信息

CA 证书在发证机构所维护的证书吊销列表中标记为已吊销，但在 Amazon IoT 中仍标记为“ACTIVE”或“PENDING_TRANSFER”。

此检查发现不合规的 CA 证书时，会返回以下原因代码：

- CERTIFICATE_REVOKED_BY_ISSUER

为什么这非常重要

已吊销的 CA 证书不应再用于签发设备证书。证书可能因遭破坏而被吊销。如果新添加的设备具有使用此 CA 证书签发的证书，可能会造成安全威胁。

如何修复

1. 使用 UpdateCACertificate 在 Amazon IoT 中将该 CA 证书标记为“INACTIVE”。您还可以使用缓解操作实现以下目的：
   - 对您的审计查找结果应用 UPDATE_CA_CERTIFICATE 缓解操作以进行此更改。
   - 应用 PUBLISH_FINDINGS_TO_SNS 缓解操作，以实施自定义响应来响应 Amazon SNS 消息。

   有关更多信息，请参阅缓解操作 (p. 995)：

2. 对吊销 CA 证书后的设备证书注册活动进行审核，并考虑吊销在此期间可能使用它颁发的任何设备证书。可以使用 ListCertificatesByCA 列出通过该 CA 证书签发的设备证书 UpdateCertificate，并吊销设备证书。

共享设备证书

多个并发连接使用相同的 X.509 证书向 Amazon IoT 进行身份验证。

此检查在 CLI 和 API 中显示为 `DEVICE_CERTIFICATE_SHARED_CHECK`。

严重性：危急

详细信息

在按需审计过程中进行此检查时，它会检查在审计开始之前 31 天至检查运行前 2 小时内设备用于连接的证书和客户端 ID。对于计划审计，此检查会查看从上次运行审计前 2 小时到该审计实例开始前 2 小时期间的数据。如果在检查期间已采取措施来缓解这种状况，请记录执行并行连接的时间，以判断问题是否持续存在。

此检查发现不合规的证书时，会返回以下原因代码：

- CERTIFICATE_SHARED_BY_MULTIPLE_DEVICES

此外，此检查返回的查找结果还包含共享证书的 ID、使用证书进行连接的客户端的 ID，以及连接/连接断开次数。最近的结果列在最前面。

为什么这非常重要

每个设备应具有唯一证书以向 Amazon IoT 进行身份验证。如果多个设备使用相同的证书，这可能表示设备已遭破坏。其身份可能已遭克隆，会进一步危害系统。

如何修复

验证设备证书是否已遭破坏。如果已遭破坏，请遵照安全最佳实践来缓解此情况。

如果在多个设备上使用同一证书，则需要执行以下操作：

1. 预置新的唯一证书并将其附加到每个设备。
2. 验证新证书是否有效，以及设备能否使用它们进行连接。
3. 用于在中UpdateCertificate将旧证书标记为 “已撤销”Amazon IoT。您还可以使用缓解操作实现以下目的：
   - 对您的审计查找结果应用 `UPDATE_DEVICE_CERTIFICATE` 缓解操作以进行此更改。
   - 应用 `ADD_THINGS_TO_THING_GROUP` 缓解操作，以将设备添加到可以对其执行操作的组。
   - 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 `PUBLISH_FINDINGS_TO_SNS` 缓解操作。

   有关更多信息，请参阅缓解操作 (p. 995)：
4. 将旧证书从各个设备中分离。

设备证书密钥质量

Amazon IoT 客户通常依赖使用 X.509 证书的 TLS 相互身份验证，用于对 Amazon IoT 消息代理进行身份验证。这些证书及其证书颁发机构证书必须先在其 Amazon IoT 账户中注册，然后才能使用。在注册这些证书时，Amazon IoT 对这些证书执行基本的完整性检查。这些检查包括：

- 必须采用有效的格式。
- 必须由注册的证书颁发机构签名。

- 必须仍然在其有效期内（换句话说，尚未过期）。
- 其加密密钥大小必须满足所需的最小大小（对于 RSA 密钥，它们必须为 2048 位或更大）。

此审计检查为您的加密密钥质量提供了以下附加测试：

- CVE-2008-0166 — 检查是否在基于 Debian 的操作系统上，使用 OpenSSL 0.9.8c-1 到 0.9.8g-9 之间的版本生成了密钥。这些版本的 OpenSSL 使用随机数生成器生成可预测的数字，使远程攻击者更容易对加密密钥进行暴力猜测攻击。
- CVE-2017-15361 - 检查密钥是否由 Infineon RSA 库 1.02.013 在 Infineon Trusted Platform Module (TPM) 固件中生成，例如 0000000000000422 - 4.34 之前的版本、000000000000062b - 6.43 之前的版本以及 0000000000008521 - 133.33 之前的版本。该库不当地处理 RSA 密钥生成，使攻击者更容易通过针对性的攻击，破解某些加密保护机制。受影响技术的示例包括 BitLocker TPM 1.2、 YubiKey 4（4.3.5 之前）生成 PGP 密钥以及 Chrome 操作系统中的 “缓存用户数据” 加密功能。

如果证书未通过这些测试，Amazon IoT Device Defender 会将证书报告为不合规。

此检查在 CLI 和 API 中显示为 `DEVICE_CERTIFICATE_KEY_QUALITY_CHECK`。

严重性：危急

详细信息

此检查适用于状态为“ACTIVE”或“PENDING_TRANSFER”的设备证书。

此检查发现不合规的证书时，会返回以下原因代码：

- CERTIFICATE_KEY_VULNERABILITY_CVE-2017-15361
- CERTIFICATE_KEY_VULNERABILITY_CVE-2008-0166

为什么这非常重要

当设备使用易受攻击的证书时，攻击者会更容易盗用该设备。

如何修复

更新您的设备证书以替换具有已知漏洞的这些证书。

如果在多个设备上使用同一证书，则需要执行以下操作：

1. 预置新的唯一证书并将其附加到每个设备。
2. 验证新证书是否有效，以及设备能否使用它们进行连接。
3. 用于在中UpdateCertificate将旧证书标记为 “已撤销”Amazon IoT。您还可以使用缓解操作实现以下目的：
   - 对您的审计查找结果应用 `UPDATE_DEVICE_CERTIFICATE` 缓解操作以进行此更改。
   - 应用 `ADD_THINGS_TO_THING_GROUP` 缓解操作，以将设备添加到可以对其执行操作的组。
   - 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 `PUBLISH_FINDINGS_TO_SNS` 缓解操作。

   有关更多信息，请参阅缓解操作 (p. 995)：
4. 将旧证书从各个设备中分离。

CA 证书密钥质量

Amazon IoT 客户通常依赖使用 X.509 证书的 TLS 相互身份验证，用于对 Amazon IoT 消息代理进行身份验证。这些证书及其证书颁发机构证书必须先在其 Amazon IoT 账户中注册，然后才能使用。在注册这些证书时，Amazon IoT 对这些证书执行基本的完整性检查，包括：

- 证书采用有效格式。
- 证书必须在其有效期内（换句话说，尚未过期）。
- 其加密密钥大小满足所需的最小大小（对于 RSA 密钥，它们必须为 2048 位或更大）。

此审计检查为您的加密密钥质量提供了以下附加测试：

- CVE-2008-0166 — 检查是否在基于 Debian 的操作系统上，使用 OpenSSL 0.9.8c-1 到 0.9.8g-9 之间的版本生成了密钥。这些版本的 OpenSSL 使用随机数生成器生成可预测的数字，使远程攻击者更容易对加密密钥进行暴力猜测攻击。
- CVE-2017-15361 - 检查密钥是否由 Infineon RSA 库 1.02.013 在 Infineon Trusted Platform Module (TPM) 固件中生成，例如 0000000000000422 - 4.34 之前的版本、000000000000062b - 6.43 之前的版本以及 0000000000008521 - 133.33 之前的版本。该库不当地处理 RSA 密钥生成，使攻击者更容易通过针对性的攻击，破解某些加密保护机制。受影响技术的示例包括 BitLocker TPM 1.2、 YubiKey 4（4.3.5 之前）生成 PGP 密钥以及 Chrome 操作系统中的 “缓存用户数据” 加密功能。

如果证书未通过这些测试，Amazon IoT Device Defender 会将证书报告为不合规。

此检查在 CLI 和 API 中显示为 `CA_CERTIFICATE_KEY_QUALITY_CHECK`。

严重性：危急

详细信息

此检查适用于状态为“ACTIVE”或“PENDING_TRANSFER”的 CA 证书。

此检查发现不合规的证书时，会返回以下原因代码：

- CERTIFICATE_KEY_VULNERABILITY_CVE-2017-15361
- CERTIFICATE_KEY_VULNERABILITY_CVE-2008-0166

为什么这非常重要

如果新添加的设备使用此 CA 证书签名，可能会造成安全威胁。

如何修复

1. 使用 UpdateCACertificate 在 Amazon IoT 中将该 CA 证书标记为“INACTIVE”。您还可以使用缓解操作实现以下目的：
   - 对您的审计查找结果应用 `UPDATE_CA_CERTIFICATE` 缓解操作以进行此更改。
   - 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 `PUBLISH_FINDINGS_TO_SNS` 缓解操作。

   有关更多信息，请参阅缓解操作 (p. 995)：
2. 对吊销 CA 证书后的设备证书注册活动进行审核，并考虑吊销在此期间可能使用它颁发的任何设备证书。（使用 ListCertificatesByCA 列出由 CA 证书签名的设备证书UpdateCertificate并吊销设备证书。）

未经身份验证的 Cognito 角色过于宽容

附加到未经身份验证的 Amazon Cognito 身份池角色的策略被视为过于宽容，因为它会授权执行以下任意 Amazon IoT 操作：

- 管理或修改事物。
- 读取事物管理数据。

- 管理非事物相关的数据或资源。

或者，因为它授权在一系列设备上执行以下 Amazon IoT 操作：

- 使用 MQTT 连接、发布或订阅预留主题（包括影子或任务执行数据）。
- 使用 API 命令读取或修改影子或任务执行数据。

一般情况下，使用未经身份验证的 Amazon Cognito 身份池角色进行连接的设备只应具备有限的权限，发布和订阅事物特定 MQTT 主题，或者使用 API 命令读取和修改与影子或任务执行数据相关的事物特定数据。

此检查在 CLI 和 API 中显示为 `UNAUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK`。

严重性：危急

详细信息

对于此检查，Amazon IoT Device Defender 会审计在执行审计之前 31 天内用于连接到 Amazon IoT 消息代理的所有 Amazon Cognito 身份池。经过身份验证或未经身份验证的 Amazon Cognito 身份连接的所有 Amazon Cognito 身份池都在审核范围内。

此检查发现不合规的未经身份验证的 Amazon Cognito 身份池角色时，会返回以下原因代码：

- ALLOWS_ACCESS_TO_IOT_ADMIN_ACTIONS
- ALLOWS_BROAD_ACCESS_TO_IOT_DATA_PLANE_ACTIONS

为什么这非常重要

由于未经身份验证的身份从未经过用户的的身份验证，因此它们带来风险的可能性远远高于经过身份验证的 Amazon Cognito 身份。如果未经身份验证的身份遭到破坏，它可能使用管理操作来修改账户设置、删除资源或获取对敏感数据的访问权限。或者，凭借对设备设置宽泛的访问权限，它可以访问或修改您账户中所有设备的影子和任务。来宾用户可能使用这些权限破坏您的整个队列或通过消息发动 DDOS 攻击。

如何修复

附加到未经身份验证的 Amazon Cognito 身份池角色的策略应仅为设备授予执行任务所需的这些权限。我们建议您完成以下步骤：

1. 创建新的合规角色。
2. 创建新的 Amazon Cognito 身份池并为其附加合规角色。
3. 验证您的身份能否使用新池访问 Amazon IoT。
4. 验证完成后，将合规角色附加到标记为不合规的 Amazon Cognito 身份池。

您还可以使用缓解操作实现以下目的：

- 应用 `PUBLISH_FINDINGS_TO_SNS` 缓解操作，以实施自定义响应来响应 Amazon SNS 消息。

有关更多信息，请参阅缓解操作 (p. 995)：

管理或修改事物。

以下 Amazon IoT API 操作用于管理或修改事物。不应向通过未经身份验证的 Amazon Cognito 身份池进行连接的设备授予执行这些操作的权限。

- `AddThingToThingGroup`

- AttachThingPrincipal
- CreateThing
- DeleteThing
- DetachThingPrincipal
- ListThings
- ListThingsInThingGroup
- RegisterThing
- RemoveThingFromThingGroup
- UpdateThing
- UpdateThingGroupsForThing

任何授权执行这些操作的角色，即便是在单一资源上执行这些操作，也会被视为不合规。

读取事物管理数据

以下 Amazon IoT API 操作用于读取或修改事物数据。不应向通过未经身份验证的 Amazon Cognito 的身份池连接的设备授予执行这些操作的权限。

- DescribeThing
- ListJobExecutionsForThing
- ListThingGroupsForThing
- ListThingPrincipals

Example

- 不合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iot:DescribeThing",
          "iot:ListJobExecutionsForThing",
          "iot:ListThingGroupsForThing",
          "iot:ListThingPrincipals"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:/thing/MyThing"
      ]
    }
  ]
}
```

此示例允许设备执行指定操作，即便该设备仅获得了针对一个事物的权限。

管理非事物对象

通过未经身份验证的 Amazon Cognito 身份池进行连接的设备，不应该被授予执行除这些部分中所述之外的 Amazon IoT API 操作的权限。可以通过创建一个不供设备使用的单独身份池，使用通过未经身份验证的 Amazon Cognito 身份池进行连接的应用程序来管理账户。

订阅/发布到 MQTT 主题

MQTT 消息通过 Amazon IoT 消息代理发送，并由设备用来执行多个操作，包括访问和修改影子状态和任务执行状态。为设备授权以连接、发布或订阅 MQTT 消息的策略，应该限定对特定资源执行如下操作：

连接

- 不合规：

```
arn:aws:iot:region:account-id:client/*
```

通配符 * 允许任何设备连接到 Amazon IoT。

```
arn:aws:iot:region:account-id:client/${iot:ClientId}
```

除非 iot:Connection.Thing.IsAttached 在条件键中设置为 True，这相当于上例中的通配符 *。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Connect" ],
      "Resource": [
        "arn:aws:iot:region:account-id:client/${iot:Connection.Thing.ThingName}"
      ],
      "Condition": {
        "Bool": { "iot:Connection.Thing.IsAttached": "true" }
      }
    }
  ]
}
```

资源规范包含与用于连接的设备名称匹配的变量。条件语句通过检查 MQTT 客户端所用的证书是否与附加到使用此名称的事物的证书匹配，进一步限制权限。

发布

- 不合规：

```
arn:aws:iot:region:account-id:topic/$aws/things/*/shadow/update
```

此示例允许设备更新任何设备的影子（* = 所有设备）。

```
arn:aws:iot:region:account-id:topic/$aws/things/*
```

此示例允许设备读取、更新或删除任何设备的影子。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Publish" ],
      "Resource": [
```

```
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
      ],
    }
  ]
}
```

资源规范包含通配符，但仅匹配其事物名称用于连接的设备的任何影子相关主题。

Subscribe

- 不合规：

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

此示例允许设备订阅为所有设备预留的影子或任务主题。

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

与上一个示例相同，不过使用的是 # 通配符。

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/+/shadow/update
```

此示例允许设备查看任何设备上的影子更新（+ = 所有设备）。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Subscribe" ],
      "Resource": [
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/jobs/*"
      ],
    }
  ]
}
```

资源规范包含通配符，但仅匹配其事物名称用于连接的设备的任何影子相关主题和任务相关主题。

接收

- 合规：

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

这是允许的，因为设备只能从其有权订阅的主题接收消息。

读取/修改影子或任务数据

为设备授予执行 API 操作权限以访问或修改设备影子或任务执行数据的策略，应该限定对特定资源执行以下操作。下面是一些 API 操作：

- DeleteThingShadow

- GetThingShadow
- UpdateThingShadow
- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

Example

- 不合规：

```
arn:aws:iot:region:account-id:thing/*
```

此示例允许设备对任何事物执行指定操作。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iot:DeleteThingShadow",
          "iot:GetThingShadow",
          "iot:UpdateThingShadow",
          "iot:DescribeJobExecution",
          "iot:GetPendingJobExecutions",
          "iot:StartNextPendingJobExecution",
          "iot:UpdateJobExecution"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:/thing/MyThing1",
        "arn:aws:iot:region:account-id:/thing/MyThing2"
      ]
    }
  ]
}
```

此示例允许设备仅对两个事物执行指定操作。

经过身份验证的 Cognito 角色过于宽容

附加到经身份验证的 Amazon Cognito 身份池角色的策略被视为过于宽容，因为它会授权执行以下 Amazon IoT 操作：

- 管理或修改事物。
- 管理非事物相关的数据或资源。

或者，因为它授权在一系列设备上执行以下 Amazon IoT 操作：

- 读取事物管理数据。
- 使用 MQTT 连接/发布/订阅预留主题（包括影子或任务执行数据）。
- 使用 API 命令读取或修改影子或任务执行数据。

一般情况下，使用经身份验证的 Amazon Cognito 身份池角色进行连接的设备只应具备有限的权限，读取事物特定管理数据、发布和订阅事物特定 MQTT 主题，或使用 API 命令读取和修改与影子或任务执行数据相关的事物特定数据。

此检查在 CLI 和 API 中显示为 `AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK`。

严重性：危急

详细信息

对于此检查，Amazon IoT Device Defender 会审计在执行审计之前 31 天内用于连接到 Amazon IoT 消息代理的所有 Amazon Cognito 身份池。经过身份验证或未经身份验证的 Amazon Cognito 身份连接的所有 Amazon Cognito 身份池都在审核范围内。

此检查发现不合规的经身份验证的 Amazon Cognito 身份池角色时，会返回以下原因代码：

- ALLOWS_BROAD_ACCESS_TO_IOT_THING_ADMIN_READ_ACTIONS
- ALLOWS_ACCESS_TO_IOT_NON_THING_ADMIN_ACTIONS
- ALLOWS_ACCESS_TO_IOT_THING_ADMIN_WRITE_ACTIONS

为什么这非常重要

如果经身份验证的身份遭到破坏，它可能使用管理操作来修改账户设置、删除资源或获取对敏感数据的访问权限。

如何修复

附加到经过身份验证的 Amazon Cognito 身份池角色的策略应仅为设备授予执行任务所需的这些权限。我们建议您完成以下步骤：

1. 创建新的合规角色。
2. 创建新的 Amazon Cognito 身份池并为其附加合规角色。
3. 验证您的身份能否使用新池访问 Amazon IoT。
4. 验证完成后，将合规角色附加到标记为不合规的 Amazon Cognito 身份池。

您还可以使用缓解操作实现以下目的：

- 应用 `PUBLISH_FINDINGS_TO_SNS` 缓解操作，以实施自定义响应来响应 Amazon SNS 消息。

有关更多信息，请参阅缓解操作 (p. 995)：

管理或修改事物。

以下 Amazon IoT API 操作可用于管理或修改事物，因此不应该为通过经身份验证的 Amazon Cognito 身份池进行连接的设备授予执行以下操作的权限：

- `AddThingToThingGroup`
- `AttachThingPrincipal`
- `CreateThing`
- `DeleteThing`
- `DetachThingPrincipal`
- `ListThings`

- ListThingsInThingGroup
- RegisterThing
- RemoveThingFromThingGroup
- UpdateThing
- UpdateThingGroupsForThing

任何授权执行这些操作的角色，即便是在单一资源上执行这些操作，也会被视为不合规。

管理非事物对象

通过经身份验证的 Amazon Cognito 身份池进行连接的设备，不应该被授予执行除这些部分中所述之外的 Amazon IoT API 操作的权限。要使用通过经身份验证 Amazon Cognito 身份池进行连接的应用程序来管理账户，请创建一个不供设备使用的单独身份池。

读取事物管理数据

以下 Amazon IoT API 操作可用于读取事物数据，因此应该为通过经身份验证的 Amazon Cognito 身份池进行连接的设备授予仅对有限事物集执行以下操作的权限：

- DescribeThing
- ListJobExecutionsForThing
- ListThingGroupsForThing
- ListThingPrincipals

- 不合规：

```
arn:aws:iot:region:account-id:thing/*
```

此示例允许设备对任何事物执行指定操作。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iot:DescribeThing",
          "iot:ListJobExecutionsForThing",
          "iot:ListThingGroupsForThing",
          "iot:ListThingPrincipals"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:/thing/MyThing"
      ]
    }
  ]
}
```

此示例允许设备仅对一个事物执行指定操作。

- 合规：

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

```
    {
      "Effect": "Allow",
      "Action": [
          "iot:DescribeThing",
          "iot:ListJobExecutionsForThing",
          "iot:ListThingGroupsForThing",
          "iot:ListThingPrincipals"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:/thing/MyThing*"
      ]
    }
  ]
}
```

此示例之所以合规，是因为尽管该资源是使用通配符 (*) 指定的，但它的前面是特定字符串，这将仅限名称包含给定前缀的设备访问事物集。

- 不合规：

```
arn:aws:iot:region:account-id:thing/*
```

此示例允许设备对任何事物执行指定操作。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iot:DescribeThing",
          "iot:ListJobExecutionsForThing",
          "iot:ListThingGroupsForThing",
          "iot:ListThingPrincipals"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:/thing/MyThing"
      ]
    }
  ]
}
```

此示例允许设备仅对一个事物执行指定操作。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iot:DescribeThing",
          "iot:ListJobExecutionsForThing",
          "iot:ListThingGroupsForThing",
          "iot:ListThingPrincipals"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:/thing/MyThing*"
      ]
```

```
    }
  ]
}
```

此示例之所以合规，是因为尽管该资源是使用通配符 (*) 指定的，但它的前面是特定字符串，这将仅限名称包含给定前缀的设备访问事物集。

订阅/发布到 MQTT 主题

MQTT 消息通过 Amazon IoT 消息代理发送，并由设备用来执行许多不同的操作，包括访问和修改影子状态和任务执行状态。为设备授权以连接、发布或订阅 MQTT 消息的策略，应该限定对特定资源执行如下操作：

连接

- 不合规：

  ```
  arn:aws:iot:region:account-id:client/*
  ```

  通配符 * 允许任何设备连接到 Amazon IoT。

  ```
  arn:aws:iot:region:account-id:client/${iot:ClientId}
  ```

  除非 `iot:Connection.Thing.IsAttached` 在条件键中设置为 True，这相当于上例中的通配符 *。

- 合规：

  ```
  {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [ "iot:Connect" ],
        "Resource": [
          "arn:aws:iot:region:account-id:client/${iot:Connection.Thing.ThingName}"
        ],
        "Condition": {
          "Bool": { "iot:Connection.Thing.IsAttached": "true" }
        }
      }
    ]
  }
  ```

  资源规范包含与用于连接的设备名称匹配的变量，且条件语句通过检查 MQTT 客户端所用的证书是否与附加到使用此名称的事物的证书匹配，进一步限制权限。

发布

- 不合规：

  ```
  arn:aws:iot:region:account-id:topic/$aws/things/*/shadow/update
  ```

  此示例允许设备更新任何设备的影子（* = 所有设备）。

  ```
  arn:aws:iot:region:account-id:topic/$aws/things/*
  ```

  此示例允许设备读取/更新/删除任何设备的影子。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Publish" ],
      "Resource": [
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
      ],
    }
  ]
}
```

资源规范包含通配符，但仅匹配其事物名称用于连接的设备的任何影子相关主题。

Subscribe

- 不合规：

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

此示例允许设备订阅为所有设备预留的影子或任务主题。

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/#
```

与上一个示例相同，不过使用的是 # 通配符。

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/+/shadow/update
```

此示例允许设备查看任何设备上的影子更新（+ = 所有设备）。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Subscribe" ],
      "Resource": [
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/jobs/*"
      ],
    }
  ]
}
```

资源规范包含通配符，但仅匹配其事物名称用于连接的设备的任何影子相关主题和任务相关主题。

接收

- 合规：

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

这是合规的，因为设备只能从其有权订阅的主题接收消息。

读取或修改影子或任务数据

为设备授予执行 API 操作权限以访问或修改设备影子或任务执行数据的策略，应该限定对特定资源执行以下操作。下面是一些 API 操作：

- DeleteThingShadow
- GetThingShadow
- UpdateThingShadow
- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

示例

- 不合规：

```
arn:aws:iot:region:account-id:thing/*
```

此示例允许设备对任何事物执行指定操作。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iot:DeleteThingShadow",
          "iot:GetThingShadow",
          "iot:UpdateThingShadow",
          "iot:DescribeJobExecution",
          "iot:GetPendingJobExecutions",
          "iot:StartNextPendingJobExecution",
          "iot:UpdateJobExecution"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:/thing/MyThing1",
        "arn:aws:iot:region:account-id:/thing/MyThing2"
      ]
    }
  ]
}
```

此示例允许设备仅对两个事物执行指定操作。

Amazon IoT 策略过于宽容

Amazon IoT 策略提供的权限过于宽泛或不受限制。它为一系列设备授予发送或接收 MQTT 消息的权限，或为一系列设备授予访问或修改影子和任务执行数据的权限。

一般而言，针对某个设备的策略应该授予仅与该设备关联的资源的访问权限，而不应该牵涉其它设备，或只牵涉少量其他设备。除了某些例外情况，在此类策略中使用通配符（例如“*”）指定资源被视为过于宽泛或不受限制。

此检查在 CLI 和 API 中显示为 IOT_POLICY_OVERLY_PERMISSIVE_CHECK。

严重性：危急

详细信息

此检查发现不合规的 Amazon IoT 策略时，会返回以下原因代码：

- ALLOWS_BROAD_ACCESS_TO_IOT_DATA_PLANE_ACTIONS

为什么这非常重要

策略过于宽容的证书、Amazon Cognito 身份或事物组一旦遭到破坏，就会影响您的整个账户。攻击者可以使用此类宽泛的访问权限，读取或修改所有设备的影子、任务或任务执行数据。或者，攻击者可以使用已遭破坏的证书连接您网络中的恶意设备或发动 DDOS 攻击。

如何修复

按照以下步骤来修复附加到事物、事物组或其它实体的任何不合规策略：

1. CreatePolicyVersion用于创建新的、合规的策略版本。将 `setAsDefault` 标记设置为 True。（这可使此新版本适用于使用策略的所有实体。）
2. ListTargetsForPolicy用于获取策略所关联的目标（证书、事物组）列表，并确定组中包含哪些设备或哪些设备使用证书进行连接。
3. 验证所有关联的设备能否连接到 Amazon IoT。如果设备无法连接，请使用 SetPolicyVersion将默认策略还原到以前的版本，修改策略，然后重试。

您可以使用缓解操作实现以下目的：

- 对您的审计查找结果应用 `REPLACE_DEFAULT_POLICY_VERSION` 缓解操作以进行此更改。
- 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 `PUBLISH_FINDINGS_TO_SNS` 缓解操作。

有关更多信息，请参阅缓解操作 (p. 995)：

使用 Amazon IoT 策略变量 (p. 303)在策略中动态引用 Amazon IoT 资源。

MQTT 权限

MQTT 消息通过 Amazon IoT 消息代理发送，并由设备用来执行多个操作，包括访问和修改影子状态和任务执行状态。为设备授权以连接、发布或订阅 MQTT 消息的策略，应该限定对特定资源执行如下操作：

连接

- 不合规：

```
arn:aws:iot:region:account-id:client/*
```

通配符 * 允许任何设备连接到 Amazon IoT。

```
arn:aws:iot:region:account-id:client/${iot:ClientId}
```

除非 `iot:Connection.Thing.IsAttached` 在条件键中设置为 True，这相当于上例中的通配符 *。

- 合规：

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Connect" ],
      "Resource": [
        "arn:aws:iot:region:account-id:client/${iot:Connection.Thing.ThingName}"
      ],
      "Condition": {
        "Bool": { "iot:Connection.Thing.IsAttached": "true" }
      }
    }
  ]
}
```

资源规范包含与用于连接的设备名称匹配的变量。条件语句通过检查 MQTT 客户端所用的证书是否与附加到使用此名称的事物的证书匹配，进一步限制权限。

发布

- 不合规：

```
arn:aws:iot:region:account-id:topic/$aws/things/*/shadow/update
```

此示例允许设备更新任何设备的影子（* = 所有设备）。

```
arn:aws:iot:region:account-id:topic/$aws/things/*
```

此示例允许设备读取、更新或删除任何设备的影子。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Publish" ],
      "Resource": [
        "arn:aws:iot:region:account-id:topic/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
      ],
    }
  ]
}
```

资源规范包含通配符，但仅匹配其事物名称用于连接的设备的任何影子相关主题。

Subscribe

- 不合规：

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

此示例允许设备订阅为所有设备预留的影子或任务主题。

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/*
```

与上一个示例相同，不过使用的是 # 通配符。

```
arn:aws:iot:region:account-id:topicfilter/$aws/things/+/shadow/update
```

此示例允许设备查看任何设备上的影子更新（+ = 所有设备）。

- 合规：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "iot:Subscribe" ],
      "Resource": [
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/shadow/*"
        "arn:aws:iot:region:account-id:topicfilter/$aws/things/
${iot:Connection.Thing.ThingName}/jobs/*"
      ],
    }
  ]
}
```

资源规范包含通配符，但仅匹配其事物名称用于连接的设备的任何影子相关主题和任务相关主题。

接收

- 合规：

```
arn:aws:iot:region:account-id:topic/$aws/things/*
```

这是合规的，因为设备只能从其有权订阅的主题接收消息。

影子和任务权限

为设备授予执行 API 操作权限以访问或修改设备影子或任务执行数据的策略，应该限定对特定资源执行以下操作。下面是一些 API 操作：

- DeleteThingShadow
- GetThingShadow
- UpdateThingShadow
- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution
- UpdateJobExecution

示例

- 不合规：

```
arn:aws:iot:region:account-id:thing/*
```

此示例允许设备对任何事物执行指定操作。

- 合规：

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

```
      "Effect": "Allow",
      "Action": [
          "iot:DeleteThingShadow",
          "iot:GetThingShadow",
          "iot:UpdateThingShadow",
          "iot:DescribeJobExecution",
          "iot:GetPendingJobExecutions",
          "iot:StartNextPendingJobExecution",
          "iot:UpdateJobExecution"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:/thing/MyThing1",
        "arn:aws:iot:region:account-id:/thing/MyThing2"
      ]
    }
  ]
}
```

此示例允许设备仅对两个事物执行指定操作。

Amazon IoT 策略可能配置错误

已确定某项 Amazon IoT 策略可能配置错误。错误配置的策略，包括过于宽松的策略，可能会导致安全事件，例如允许设备访问意外资源。

可能配置错误的 Amazon IoT 策略检查是一种警告，提醒您确保在更新策略之前仅允许预期的操作。

在此 CLI 和 API 中，该检查显示为 `IOT_POLICY_POTENTIAL_MISCONFIGURATION_CHECK`。

严重性：中

详细信息

当此检查发现可能配置错误的 Amazon IoT 策略时，Amazon IoT 会返回以下原因代码：

- POLICY_CONTAINS_MQTT_WILDCARDS_IN_DENY_STATEMENT
- TOPIC_FILTERS_INTENDED_TO_DENY_ALLOWED_USING_WILDCARDS

为什么这非常重要

错误配置的策略可能会向设备提供超出所需的权限，从而导致意外后果。我们建议仔细考虑该政策，以限制对资源的访问并防止安全威胁。

策略在拒绝语句示例中包含 MQTT 通配符

可能配置错误的 Amazon IoT 策略检查会检查 deny 语句中是否有 MQTT 通配符（+ 或 # ）。通配符被 Amazon IoT 策略视为文字字符串，可能会使策略过于宽松。

以下示例旨在通过在策略中使用 MQTT 通配符 # 来拒绝订阅与 `building/control_room` 相关的主题。但是，MQTT 通配符在 Amazon IoT 策略中没有通配符含义，设备可以订阅 `building/control_room/data1`。

可能配置错误的 Amazon IoT 策略检查将使用原因代码 `POLICY_CONTAINS_MQTT_WILDCARDS_IN_DENY_STATEMENT` 标记此策略。

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

```
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:region:account-id:topicfilter/building/*"
    },
    {
      "Effect": "Deny",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:region:account-id:topicfilter/building/control_room/#"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:region:account-id:topic/building/*"
    }
  ]
}
```

以下是正确配置的策略的示例。设备无权订阅 building/control_room/ 的子主题，也无权接收来自 building/control_room/ 的子主题的消息。

```
{
"Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:region:account-id:topicfilter/building/*"
    },
    {
      "Effect": "Deny",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:region:account-id:topicfilter/building/control_room/*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:region:account-id:topic/building/*"
    },
    {
      "Effect": "Deny",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:region:account-id:topic/building/control_room/*"
    }
  ]
}
```

旨在拒绝使用通配符允许的主题筛选条件示例

以下示例策略旨在通过拒绝资源 building/control_room/* 来拒绝订阅与 building/control_room 相关的主题。但是，设备可以发送 building/# 订阅请求，并接收来自与 building 相关的所有主题的消息，包括 building/control_room/data1。

可能配置错误的 Amazon IoT 策略检查将使用原因代码 TOPIC_FILTERS_INTENDED_TO_DENY_ALLOWED_USING_WILDCARDS 标记此策略。

以下示例策略有权接收关于 building/control_room topics 的消息：

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

```
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:region:account-id:topicfilter/building/*"
    },
    {
      "Effect": "Deny",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:region:account-id:topicfilter/building/control_room/*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:region:account-id:topic/building/*"
    }
  ]
}
```

以下是正确配置的策略的示例。设备无权订阅 building/control_room/ 的子主题，也无权接收来自 building/control_room/ 的子主题的消息。

```
{
"Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:region:account-id:topicfilter/building/*"
    },
    {
      "Effect": "Deny",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:region:account-id:topicfilter/building/control_room/*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:region:account-id:topic/building/*"
    },
    {
      "Effect": "Deny",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:region:account-id:topic/building/control_room/*"
    }
  ]
}
```

Note

此检查可能会误报。我们建议您评估所有已标记的策略，并使用审计抑制功能标记误报资源。

如何修复

此检查会标记可能配置错误的策略，因此可能会出现误报。使用审计抑制功能 (p. 938)标记所有误报，这样以后就不会再标记误报。

您可以按照以下步骤来修复附加到事物、事物组或其他实体的任何不合规策略：

1. CreatePolicyVersion用于创建新的、合规的策略版本。将 setAsDefault 标记设置为 True。（这可使此新版本适用于使用策略的所有实体。）

   有关为常见用例创建 Amazon IoT 策略的示例，请参阅 Amazon IoT Core 开发者指南中的发布/订阅策略示例 (p. 316)。

2. 验证所有关联的设备能否连接到 Amazon IoT。如果设备无法连接，请使用 SetPolicyVersion将默认策略还原到以前的版本，修改策略，然后重试。

您可以使用缓解操作实现以下目的：

- 对您的审计查找结果应用 REPLACE_DEFAULT_POLICY_VERSION 缓解操作以进行此更改。
- 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 PUBLISH_FINDINGS_TO_SNS 缓解操作。

有关更多信息，请参阅缓解操作 (p. 995)：

使用 Amazon IoT 策略变量 (p. 303)在策略中动态引用 Amazon IoT 资源。

CA 证书即将过期

CA 证书将在 30 天内过期或已经过期。

此检查在 CLI 和 API 中显示为 CA_CERTIFICATE_EXPIRING_CHECK。

严重性：中

详细信息

此检查适用于状态为“ACTIVE”或“PENDING_TRANSFER”的 CA 证书。

此检查发现不合规的 CA 证书时，会返回以下原因代码：

- CERTIFICATE_APPROACHING_EXPIRATION
- CERTIFICATE_PAST_EXPIRATION

为什么这非常重要

已过期的 CA 证书不应被用于签发新设备证书。

如何修复

有关如何继续，请参阅安全最佳实践。您可能需要：

1. 向 Amazon IoT 注册新的 CA 证书。
2. 验证能否使用新的 CA 证书签发设备证书。
3. 使用 UpdateCACertificate 在 Amazon IoT 中将旧的 CA 证书标记为“INACTIVE”。您还可以使用缓解操作实现以下目的：
   - 对您的审计查找结果应用 UPDATE_CA_CERTIFICATE 缓解操作以进行此更改。
   - 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 PUBLISH_FINDINGS_TO_SNS 缓解操作。

   有关更多信息，请参阅缓解操作 (p. 995)：

冲突的 MQTT 客户端 ID

多个设备使用同一客户端 ID 连接。

此检查在 CLI 和 API 中显示为 CONFLICTING_CLIENT_IDS_CHECK。

严重性：高

详细信息

使用同一客户端 ID 建立了多个连接，从而导致已连接的设备断开连接。MQTT 规范只允许每个客户端 ID 有一个活动连接，因此当另一个设备使用同一客户端 ID 连接时，它会使前一个连接断开。

在按需审核过程中进行此检查时，它会检查在审核开始之前 31 天内客户端 ID 是如何用于连接的。对于计划审核，此检查会查看从上次运行审核到该审核实例开始期间的数据。如果您已在检查期间采取措施来缓解这种状况，请记录连接/连接断开的时间，以判断问题是否持续存在。

此检查发现不合规问题时，会返回以下原因代码：

- DUPLICATE_CLIENT_ID_ACROSS_CONNECTIONS

此检查返回的查找结果还包含用于连接的客户端 ID、委托人 ID 和连接断开次数。最近的结果列在最前面。

为什么这非常重要

ID 相冲突的设备将被迫不断重新连接，这可能导致消息丢失或致使设备无法连接。

这可能表示设备或设备的凭证已遭破坏，并可能是 DDoS 攻击的一部分。也有可能是设备未在账户中得到正确配置，或者设备连接效果不佳，被迫每分钟重新连接多次。

如何修复

将每个设备注册为 Amazon IoT 中的唯一事物，并使用事物名称作为客户端 ID 进行连接。或者，在通过 MQTT 连接设备时使用 UUID 作为客户端 ID。您还可以使用缓解操作实现以下目的：

- 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 `PUBLISH_FINDINGS_TO_SNS` 缓解操作。

有关更多信息，请参阅缓解操作 (p. 995)：

设备证书即将过期

设备证书将在 30 天内过期或已经过期。

此检查在 CLI 和 API 中显示为 `DEVICE_CERTIFICATE_EXPIRING_CHECK`。

严重性：中

详细信息

此检查适用于状态为“ACTIVE”或“PENDING_TRANSFER”的设备证书。

此检查发现不合规的设备证书时，会返回以下原因代码：

- CERTIFICATE_APPROACHING_EXPIRATION
- CERTIFICATE_PAST_EXPIRATION

为什么这非常重要

设备证书过期后不应再投入使用。

如何修复

有关如何继续，请参阅安全最佳实践。您可能需要：

1. 预置新证书并将它附加到设备。
2. 验证新证书是否有效以及设备能否使用它进行连接。
3. UpdateCertificate用于在中将旧证书标记为非活动Amazon IoT。您还可以使用缓解操作实现以下目的：
   - 对您的审计查找结果应用 UPDATE_DEVICE_CERTIFICATE 缓解操作以进行此更改。
   - 应用 ADD_THINGS_TO_THING_GROUP 缓解操作，以将设备添加到可以对其执行操作的组。
   - 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 PUBLISH_FINDINGS_TO_SNS 缓解操作。

   有关更多信息，请参阅缓解操作 (p. 995)：
4. 将旧证书从设备分离。（请参见 DetachThingPrincipal。）

已撤销的设备证书仍处于激活状态

已吊销的设备证书仍处于活动状态。

此检查在 CLI 和 API 中显示为 REVOKED_DEVICE_CERTIFICATE_STILL_ACTIVE_CHECK。

严重性：中

详细信息

设备证书位于其 CA 的证书吊销列表中，但它在 Amazon IoT 中仍处于活动状态。

此检查适用于状态为“ACTIVE”或“PENDING_TRANSFER”的设备证书。

此检查发现不合规问题时，会返回以下原因代码：

- CERTIFICATE_REVOKED_BY_ISSUER

为什么这非常重要

设备证书通常因为遭到破坏而被吊销。证书也可能因为错误或疏忽而尚未在 Amazon IoT 中吊销。

如何修复

验证设备证书是否已遭破坏。如果已遭破坏，请遵照安全最佳实践来缓解此情况。您可能需要：

1. 为设备预置新证书。
2. 验证新证书是否有效以及设备能否使用它进行连接。
3. 用于在中UpdateCertificate将旧证书标记为 “已撤销”Amazon IoT。您还可以使用缓解操作实现以下目的：
   - 对您的审计查找结果应用 UPDATE_DEVICE_CERTIFICATE 缓解操作以进行此更改。
   - 应用 ADD_THINGS_TO_THING_GROUP 缓解操作，以将设备添加到可以对其执行操作的组。
   - 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 PUBLISH_FINDINGS_TO_SNS 缓解操作。

   有关更多信息，请参阅缓解操作 (p. 995)：
4. 将旧证书从设备分离。（请参见 DetachThingPrincipal。）

日志记录已禁用

Amazon IoT亚马逊未启用日志 CloudWatch。验证 V1 和 V2 日志记录。

此检查在 CLI 和 API 中显示为 LOGGING_DISABLED_CHECK。

严重性：低

详细信息

此检查发现不合规问题时，会返回以下原因代码：

- LOGGING_DISABLED

为什么这非常重要

Amazon IoT借 CloudWatch 助日志，您可以了解中的行为Amazon IoT，包括身份验证失败、意外的连接和连接断开，这些行为可能表明设备遭到破坏。

如何修复

启用Amazon IoT登录 CloudWatch。请参阅监控工具 (p. 384)。您还可以使用缓解操作实现以下目的：

- 对您的审计查找结果应用 ENABLE_IOT_LOGGING 缓解操作以进行此更改。
- 如果要实现自定义响应以响应 Amazon SNS 消息，请应用 PUBLISH_FINDINGS_TO_SNS 缓解操作。

有关更多信息，请参阅 缓解操作 (p. 995)。

审核命令

管理审核设置

使用UpdateAccountAuditConfiguration 为账户配置审核设置。通过此命令，您可以启用希望用于审核、设置可选通知及配置权限的检查。

使用 DescribeAccountAuditConfiguration 检查这些设置。

使用 DeleteAccountAuditConfiguration 删除审核设置。这会还原所有默认值，并有效地禁用审核，因为所有检查默认处于禁用状态。

UpdateAccountAuditConfiguration

针对此账户配置或重新配置 Device Defender 审核设置。设置包括审核通知的发送方式以及启用或禁用的审核检查类型。

摘要

```
aws iot  update-account-audit-configuration \
    [--role-arn <value>] \
    [--audit-notification-target-configurations <value>] \
    [--audit-check-configurations <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

cli-input-json 格式

```
{
  "roleArn": "string",
  "auditNotificationTargetConfigurations": {
    "string": {
      "targetArn": "string",
      "roleArn": "string",
      "enabled": "boolean"
```

```
    }
  },
  "auditCheckConfigurations": {
    "string": {
      "enabled": "boolean"
    }
  }
}
```

cli-input-json 字段

名称	类型	描述
roleArn	字符串 最大长度：2048，最小长度：20	角色的 ARN，该角色授权 Amazon IoT 在执行审核时访问设备、策略、证书及其它项相关信息。
auditNotificationTargetConfigurations	映射	审核通知要发送到的目标的相关信息。
targetArn	字符串	审核通知要发送到的目标（SNS 主题）的 ARN。
roleArn	字符串 最大长度：2048，最小长度：20	可授权向目标发送通知的角色的 ARN。
已启用	布尔值	如果已启用向目标发送通知，则为 True。
auditCheckConfigurations	映射	为此账户指定要启用和禁用的审核检查。使用 `DescribeAccountAuditConfiguration` 查看所有检查的列表，包括当前已启用的检查。 某些检查在启用后，可能会立即启动某些数据收集。如果禁用某项检查，则将删除迄今为止收集的与该检查有关的所有数据。 任何计划审核所使用的检查无法禁用。必须先从计划审核中删除检查，或删除计划审核自身。 第一次调用 `UpdateAccountAuditConfiguration` 时，必须提供此参数，且必须至少指定一个已启用的检查。
已启用	布尔值	如果已为此账户启用审核检查，则为 True。

输出

无

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

DescribeAccountAuditConfiguration

获取此账户的 Device Defender 审核设置信息。设置包括审核通知的发送方式以及启用或禁用的审核检查类型。

摘要

```
aws iot  describe-account-audit-configuration  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

cli-input-json 格式

```
{
}
```

输出

```
{
  "roleArn": "string",
  "auditNotificationTargetConfigurations": {
    "string": {
      "targetArn": "string",
      "roleArn": "string",
      "enabled": "boolean"
    }
  },
  "auditCheckConfigurations": {
    "string": {
      "enabled": "boolean"
    }
  }
}
```

CLI 输出字段

名称	类型	描述
roleArn	字符串 最大长度：2048，最小长度：20	角色的 ARN，该角色授权 Amazon IoT 在执行审核时访问设备、策略、证书及其它项相关信息。 第一次调用 `UpdateAccountAuditConfiguration` 时，必须提供此参数。

名称	类型	描述
auditNotificationTargetConfigurations	映射	有关此账户的审核通知要发送到的目标的相关信息。
targetArn	字符串	审核通知要发送到的目标（SNS 主题）的 ARN。
roleArn	字符串 最大长度：2048，最小长度：20	可授权向目标发送通知的角色的 ARN。
已启用	布尔值	如果已启用向目标发送通知，则为 True。
auditCheckConfigurations	映射	针对此账户启用和禁用的审核检查。
已启用	布尔值	如果已为此账户启用审核检查，则为 True。

– 错误 –

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

DeleteAccountAuditConfiguration

还原此账户的默认 Device Defender 审核设置。您输入的所有配置数据将被删除，所有审计检查均重置为已禁用。

摘要

```
aws iot  delete-account-audit-configuration \
    [--delete-scheduled-audits | --no-delete-scheduled-audits]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` 格式

```
{
  "deleteScheduledAudits": "boolean"
}
```

`cli-input-json` 字段

名称	类型	描述
deleteScheduledAudits	布尔值	如果为 True，则所有计划审核都将被删除。

输出

无

– 错误 –

InvalidRequestException

请求的内容无效。

ResourceNotFoundException

指定的资源不存在。

ThrottlingException

速率超过限制。

InternalFailureException

出现意外错误。

计划审核

使用 CreateScheduledAudit 创建一个或多个计划审核。通过此命令，您可以指定在审核期间要执行的检查以及审核应该运行的频率。

使用 ListScheduledAudits 和 DescribeScheduledAudit 跟踪计划审核。

使用 UpdateScheduledAudit 更改现有的计划审核，或使用 DeleteScheduledAudit 将其删除。

CreateScheduledAudit

创建计划审核，使之按指定的时间间隔运行。

摘要

```
aws iot  create-scheduled-audit \
    --frequency <value> \
    [--day-of-month <value>] \
    [--day-of-week <value>] \
    --target-check-names <value> \
    [--tags <value>] \
    --scheduled-audit-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

cli-input-json 格式

```
{
  "frequency": "string",
  "dayOfMonth": "string",
  "dayOfWeek": "string",
  "targetCheckNames": [
    "string"
  ],
  "tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "scheduledAuditName": "string"
}
```

cli-input-json 字段

名称	类型	描述
frequency	字符串	计划审核的运行频率。可以是以下任一值：“DAILY”、“WEEKLY”、“BIWEEKLY”或“MOI每个审核的实际开始时间取决于系统。 枚举：DAILY | WEEKLY | BIWEEKLY | MONTHLY
dayOfMonth	字符串 模式：^([1-9]|[12][0-9]|3[01])$\|^LAST$	运行计划审核当月的具体日期。可以是“1”到“31”的任意值或“LAST”。如果 `frequency` 参数设置为“MONTHLY”，则此字段为必填字段。如果指定第 29-31 日，但当月没有那么多天，那么系统会在当月的最后一天运行审核。
dayOfWeek	字符串	运行计划审核当周的具体日期。可以是以下任一值：“SUN”、“MON”、“TUE”、“WED”、“THU”、“如果 `frequency` 参数段设置为“WEEKLY”或“BIWEEKLY”，则此字段为必填字段。 枚举：SUN | MON | TUE | WED | THU | FRI | SAT
targetCheckNames	列表 成员：AuditCheckName	计划审核期间执行的检查。账户必须已启用检查。（使用 `DescribeAccountAuditConfiguration` 查看所有检查的列表，包括已启用的检查，或使用 `UpdateAccountAuditConfiguration` 选择启用的检查。）
标签	列表 成员：Tag java 类：java.util.List	可用于管理计划审核的元数据。
密钥	字符串	标签的键。
值	字符串	标签的值。
scheduledAuditName	字符串 最大长度：128，最小长度：1 模式：[a-zA-Z0-9_-]+	要为计划审核指定的名称。（最多 128 个字符）

输出

```
{
```

```
  "scheduledAuditArn": "string"
}
```

CLI 输出字段

名称	类型	描述
scheduledAuditArn	字符串	计划审核的 ARN。

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

`LimitExceededException`

已超出限制。

ListScheduledAudits

列出所有计划审核。

摘要

```
aws iot  list-scheduled-audits \
    [--next-token <value>] \
    [--max-results <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` 格式

```
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

`cli-input-json` 字段

名称	类型	描述
nextToken	字符串	用于获取下一组结果的令牌。
maxResults	integer 范围 - 最大值：250，最小值：1	一次性返回的最大结果数。默认值为 25。

输出

```
{
```

```
  "scheduledAudits": [
    {
      "scheduledAuditName": "string",
      "scheduledAuditArn": "string",
      "frequency": "string",
      "dayOfMonth": "string",
      "dayOfWeek": "string"
    }
  ],
  "nextToken": "string"
}
```

CLI 输出字段

名称	类型	描述
scheduledAudits	列表 成员：ScheduledAuditMetadata java 类：java.util.List	计划审核的列表。
scheduledAuditName	字符串 最大长度：128，最小长度：1 模式：[a-zA-Z0-9_-]+	计划审核的名称。
scheduledAuditArn	字符串	计划审核的 ARN。
frequency	字符串	计划审核的运行频率。 枚举：DAILY | WEEKLY | BIWEEKLY | MONTHLY
dayOfMonth	字符串 模式：^([1-9]|[12][0-9]|3[01])$\|^LAST$	运行计划审核当月的具体日期（如果 `frequency` 为“MONTHLY”）。如果指定第 29-31 日，但当月没有那么多天，那么系统会在当月的最后一天运行审核。
dayOfWeek	字符串	运行计划审核当周的具体日期（如果 `frequency` 为“WEEKLY”或“BIWEEKLY”）。 枚举：SUN | MON | TUE | WED | THU | FRI | SAT
nextToken	字符串	用于检索下一组结果的令牌，没有更多结果时为 `null`。

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

DescribeScheduledAudit

获取有关计划审核的信息。

摘要

```
aws iot  describe-scheduled-audit \
    --scheduled-audit-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` 格式

```
{
  "scheduledAuditName": "string"
}
```

`cli-input-json` 字段

名称	类型	描述
scheduledAuditName	字符串 最大长度：128，最小长度：1 模式：[a-zA-Z0-9_-]+	要获取其信息的计划审核的名称。

输出

```
{
  "frequency": "string",
  "dayOfMonth": "string",
  "dayOfWeek": "string",
  "targetCheckNames": [
    "string"
  ],
  "scheduledAuditName": "string",
  "scheduledAuditArn": "string"
}
```

CLI 输出字段

名称	类型	描述
frequency	字符串	计划审核的运行频率。可以是以下任一值：“DAILY”、“WEEKLY”、“BIWEEKLY”或“MOI每个审核的实际开始时间取决于系统。 枚举：DAILY | WEEKLY | BIWEEKLY | MONTHLY

名称	类型	描述
dayOfMonth	字符串 模式：^([1-9]|[12][0-9]|3[01])$\|^LAST$	运行计划审核当月的具体日期。可以是“1”到“31”的任意值或“LAST”。如果指定第 29-31 日，但当月没有那么多天，那么系统会在当月的最后一天运行审核。
dayOfWeek	字符串	运行计划审核当周的具体日期。可以是以下任一值：“SUN”、“MON”、“TUE”、“WED”、“THU”、“ 枚举：SUN | MON | TUE | WED | THU | FRI | SAT
targetCheckNames	列表 成员：AuditCheckName	计划审核期间执行的检查。账户必须已启用检查。（使用 `DescribeAccountAuditConfiguration` 查看所有检查的列表，包括已启用的检查，或使用 `UpdateAccountAuditConfiguration` 选择启用的检查。）
scheduledAuditName	字符串 最大长度：128，最小长度：1 模式：[a-zA-Z0-9_-]+	计划审核的名称。
scheduledAuditArn	字符串	计划审核的 ARN。

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ResourceNotFoundException`

指定的资源不存在。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

UpdateScheduledAudit

更新计划审核，包括执行的检查和审核执行的频率。

摘要

```
aws iot  update-scheduled-audit \
    [--frequency <value>] \
    [--day-of-month <value>] \
    [--day-of-week <value>] \
    [--target-check-names <value>] \
```

```
    --scheduled-audit-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

cli-input-json 格式

```
{
  "frequency": "string",
  "dayOfMonth": "string",
  "dayOfWeek": "string",
  "targetCheckNames": [
    "string"
  ],
  "scheduledAuditName": "string"
}
```

cli-input-json 字段

名称	类型	描述
frequency	字符串	计划审核的运行频率。可以是以下任一值：“DAILY”、“WEEKLY”、“BIWEEKLY”或“MOI每个审核的实际开始时间取决于系统。 枚举：DAILY | WEEKLY | BIWEEKLY | MONTHLY
dayOfMonth	字符串 模式：^([1-9]|[12][0-9]|3[01])$\|^LAST$	运行计划审核当月的具体日期。可以是“1”到“31”的任意值或“LAST”。如果 frequency 参数设置为“MONTHLY”，则此字段为必填字段。如果指定第 29-31 日，但当月没有那么多天，那么系统会在当月的最后一天运行审核。
dayOfWeek	字符串	运行计划审核当周的具体日期。可以是以下任一值：“SUN”、“MON”、“TUE”、“WED”、“THU”、“如果 frequency 参数段设置为“WEEKLY”或“BIWEEKLY”，则此字段为必填字段。 枚举：SUN | MON | TUE | WED | THU | FRI | SAT
targetCheckNames	列表 成员：AuditCheckName	计划审核期间执行的检查。账户必须已启用检查。（使用 DescribeAccountAuditConfiguration 查看所有检查的列表，包括已启用的检查，或使用 UpdateAccountAuditConfiguration 选择启用的检查。）
scheduledAuditName	字符串 最大长度：128，最小长度：1	计划审核的名称。（最多 128 个字符）

名称	类型	描述
	模式：[a-zA-Z0-9_-]+	

输出

```
{
  "scheduledAuditArn": "string"
}
```

CLI 输出字段

名称	类型	描述
scheduledAuditArn	字符串	计划审核的 ARN。

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ResourceNotFoundException`

指定的资源不存在。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

DeleteScheduledAudit

删除计划审核。

摘要

```
aws iot  delete-scheduled-audit \
    --scheduled-audit-name <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` 格式

```
{
  "scheduledAuditName": "string"
}
```

`cli-input-json` 字段

名称	类型	描述
scheduledAuditName	字符串 最大长度：128，最小长度：1	要删除的计划审核的名称。

名称	类型	描述
	模式：[a-zA-Z0-9_-]+	

输出

无

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ResourceNotFoundException`

指定的资源不存在。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

运行按需审核

使用 `StartOnDemandAuditTask` 指定要执行的检查并立即开始运行审核。

StartOnDemandAuditTask

启动按需 Device Defender 审核。

摘要

```
aws iot  start-on-demand-audit-task \
    --target-check-names <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` 格式

```
{
  "targetCheckNames": [
    "string"
  ]
}
```

`cli-input-json` 字段

名称	类型	描述
targetCheckNames	列表 成员：AuditCheckName	在审核过程中执行的检查。必须为账户启用指定的检查，否则会出现异常。（使用 `DescribeAccountAuditConfiguration` 查看所有检查的列表，包括已启用的检查，或使用

名称	类型	描述
		UpdateAccountAuditConfiguration 选择启用的检查。)

输出

```
{
  "taskId": "string"
}
```

CLI 输出字段

名称	类型	描述
taskId	字符串 最大长度：40，最小长度：1 模式：[a-zA-Z0-9-]+	启动的按需审核的 ID。

– 错误 –

InvalidRequestException

请求的内容无效。

ThrottlingException

速率超过限制。

InternalFailureException

出现意外错误。

LimitExceededException

已超出限制。

管理审核实例

使用 DescribeAuditTask 获取关于特定审核实例的信息。如果审核已经运行，则结果会包括失败的检查、通过的检查、系统无法完成的检查，以及仍在运行中的检查（如果审核仍在进行中）。

使用 ListAuditTasks 查找在特定时间间隔内运行的审核。

使用 CancelAuditTask 停止正在进行的审核。

DescribeAuditTask

获取有关 Device Defender 审核的信息。

摘要

```
aws iot  describe-audit-task \
    --task-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

cli-input-json 格式

```
{
  "taskId": "string"
}
```

cli-input-json 字段

名称	类型	描述
taskId	字符串 最大长度：40，最小长度：1 模式：[a-zA-Z0-9-]+	要获取其信息的审核的 ID。

输出

```
{
  "taskStatus": "string",
  "taskType": "string",
  "taskStartTime": "timestamp",
  "taskStatistics": {
    "totalChecks": "integer",
    "inProgressChecks": "integer",
    "waitingForDataCollectionChecks": "integer",
    "compliantChecks": "integer",
    "nonCompliantChecks": "integer",
    "failedChecks": "integer",
    "canceledChecks": "integer"
  },
  "scheduledAuditName": "string",
  "auditDetails": {
    "string": {
      "checkRunStatus": "string",
      "checkCompliant": "boolean",
      "totalResourcesCount": "long",
      "nonCompliantResourcesCount": "long",
      "errorCode": "string",
      "message": "string"
    }
  }
}
```

CLI 输出字段

名称	类型	描述
taskStatus	字符串	审核的状态，可以是以下任一值：“IN_PROGRESS”、“COMPLETED”、“FAIL 枚举：IN_PROGRESS | COMPLETED | FAILED | CANCELED
taskType	字符串	审核的类型：“ON_DEMAND_AUDIT_TASK”或“SCHEDU 枚举：ON_DEMAND_AUDIT_TASK | SCHEDULED_AUDIT_TASK

名称	类型	描述
taskStartTime	timestamp	审核的开始时间。
taskStatistics	TaskStatistics	有关审核的统计信息。
totalChecks	integer	此审核中检查的数量。
inProgressChecks	integer	正在进行的检查的数量。
waitingForDataCollectionChecks	integer	等待数据收集的检查的数量。
compliantChecks	integer	发现合规资源的检查的数量。
nonCompliantChecks	integer	发现不合规资源的检查的数量。
failedChecks	integer	检查的数量。
canceledChecks	integer	因审核取消而未运行的检查的数量。
scheduledAuditName	字符串 最大长度：128，最小长度：1 模式：[a-zA-Z0-9_-]+	计划审核的名称（仅当审核是计划审核时）。
auditDetails	映射	有关在此审核过程中执行的每项检查的详细信息。
checkRunStatus	字符串	此检查的完成状态，可以是以下任一值：“IN_PROGRESS”、“WAITING_FOR_DATA 枚举：IN_PROGRESS | WAITING_FOR_DATA_COLLECTION | CANCELED | COMPLETED_COMPLIANT | COMPLETED_NON_COMPLIANT | FAILED
checkCompliant	布尔值	如果检查完成并找到所有合规资源，则为 True。
totalResourcesCount	long	已执行检查的资源的数量。
nonCompliantResourcesCount	long	检查发现不合规的资源的数量。
errorCode	字符串	在此审核过程中执行此检查时遇到的所有错误代码。可以是以下任一值：“INSUFFICIENT_PERMISSIONS”或“AUDIT
message	字符串 最大长度：2048	与在此审核过程中执行此检查时遇到的所有错误相关的消息。

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ResourceNotFoundException`

指定的资源不存在。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

ListAuditTasks

列出已在指定时间段内执行的 Device Defender 审核。

摘要

```
aws iot  list-audit-tasks \
    --start-time <value> \
    --end-time <value> \
    [--task-type <value>] \
    [--task-status <value>] \
    [--next-token <value>] \
    [--max-results <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` 格式

```
{
  "startTime": "timestamp",
  "endTime": "timestamp",
  "taskType": "string",
  "taskStatus": "string",
  "nextToken": "string",
  "maxResults": "integer"
}
```

`cli-input-json` 字段

名称	类型	描述
startTime	timestamp	时间段的开始时间。审核信息只保留有限的时间（180 天）。在保留信息之前请求开始时间将引发 `InvalidRequestException`。
endTime	timestamp	时间段的结束时间。
taskType	字符串	筛选条件，用于将输出限定到指定审核类型，可以是以下任一值：“ON_DEMAND_AUDIT_TASK”或“SCHEDU 枚举： ON_DEMAND_AUDIT_TASK | SCHEDULED_AUDIT_TASK

名称	类型	描述
taskStatus	字符串	筛选条件，用于将输出限定到具有指定完成状态的审核，可以是以下任一值：“IN_PROGRESS”、“COMPLETED”、“FAIL 枚举：IN_PROGRESS | COMPLETED | FAILED | CANCELED
nextToken	字符串	用于获取下一组结果的令牌。
maxResults	integer 范围 - 最大值：250，最小值：1	一次性返回的最大结果数。默认值为 25。

输出

```
{
  "tasks": [
    {
      "taskId": "string",
      "taskStatus": "string",
      "taskType": "string"
    }
  ],
  "nextToken": "string"
}
```

CLI 输出字段

名称	类型	描述
tasks	列表 成员：AuditTaskMetadata java 类：java.util.List	在指定时间段内执行的审核。
taskId	字符串 最大长度：40，最小长度：1 模式：[a-zA-Z0-9-]+	此审核的 ID。
taskStatus	字符串	此审核的状态，可以是以下任一值：“IN_PROGRESS”、“COMPLETED”、“FAIL 枚举：IN_PROGRESS | COMPLETED | FAILED | CANCELED
taskType	字符串	此审核的类型，可以是以下任一值：“ON_DEMAND_AUDIT_TASK”或“SCHEDU 枚举：ON_DEMAND_AUDIT_TASK | SCHEDULED_AUDIT_TASK

名称	类型	描述
nextToken	字符串	用于检索下一组结果的令牌，没有更多结果时为 null。

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

CancelAuditTask

取消正在进行的审核。审核可能是计划审核，也可能是按需审核。如果审核未处于进行中，则将引发 `InvalidRequestException`。

摘要

```
aws iot  cancel-audit-task \
    --task-id <value>  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` 格式

```
{
  "taskId": "string"
}
```

`cli-input-json` 字段

名称	类型	描述
taskId	字符串 最大长度：40，最小长度：1 模式：[a-zA-Z0-9-]+	要取消的审核的 ID。只能取消状态为“IN_PROGRESS”的审核。

输出

无

– 错误 –

`ResourceNotFoundException`

指定的资源不存在。

`InvalidRequestException`

请求的内容无效。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

检查审核结果

使用 `ListAuditFindings` 查看审核结果。可以按照检查类型、特定资源或审核时间筛选结果。您可以使用此信息来解决发现的任何问题。

您可以定义缓解操作并将其应用于审核的结果。有关更多信息，请参阅 缓解操作 (p. 995)。

ListAuditFindings

列出 Device Defender 审核的结果或在指定时间段内审核执行的结果。（结果保留 180 天。）

摘要

```
aws iot  list-audit-findings \
    [--task-id <value>] \
    [--check-name <value>] \
    [--resource-identifier <value>] \
    [--max-results <value>] \
    [--next-token <value>] \
    [--start-time <value>] \
    [--end-time <value>]  \
    [--cli-input-json <value>] \
    [--generate-cli-skeleton]
```

`cli-input-json` 格式

```
{
  "taskId": "string",
  "checkName": "string",
  "resourceIdentifier": {
    "deviceCertificateId": "string",
    "caCertificateId": "string",
    "cognitoIdentityPoolId": "string",
    "clientId": "string",
    "policyVersionIdentifier": {
      "policyName": "string",
      "policyVersionId": "string"
    },

    "roleAliasArn": "string",
    "account": "string"
  },
  "maxResults": "integer",
  "nextToken": "string",
  "startTime": "timestamp",
  "endTime": "timestamp"
}
```

cli-input-json 字段

名称	类型	描述
taskId	字符串 最大长度：40，最小长度：1 模式：[a-zA-Z0-9-]+	筛选条件，用于将结果限定到具有指定 ID 的审核。必须指定 taskId 或 startTime 和 endTime，但不能同时指定这两项。
checkName	字符串	筛选条件，用于将结果限定到指定审核检查结果。
resourceIdentifier	ResourceIdentifier	用于标识不合规资源的信息。
deviceCertificateId	字符串 最大长度：64，最小长度：64 模式：(0x)?[a-fA-F0-9]+	附加到资源的证书的 ID。
caCertificateId	字符串 最大长度：64，最小长度：64 模式：(0x)?[a-fA-F0-9]+	授权证书所用的 CA 证书的 ID。
cognitoIdentityPoolId	字符串	Amazon Cognito 身份池的 ID。
clientId	字符串	客户端 ID。
policyVersionIdentifier	PolicyVersionIdentifier	与资源关联的策略的版本。
policyName	字符串 最大长度：128，最小长度：1 模式：[w+=,.@-]+	策略的名称。
policyVersionId	字符串 模式：[0-9]+	与资源关联的策略的版本 ID。
roleAliasArn	字符串	具有过于宽松操作的角色别名的 ARN。 最大长度：2048，最小长度：1
账户	字符串 最大长度：12，最小长度：12 模式：[0-9]+	资源所关联的账户。
maxResults	integer 范围 - 最大值：250，最小值：1	一次性返回的最大结果数。默认值为 25。
nextToken	字符串	用于获取下一组结果的令牌。
startTime	timestamp	筛选条件，用于将结果限定到在指定时间之后发现的结果。必须

名称	类型	描述
		指定 startTime 和 endTime 或 taskId，但不能同时指定这两项。
endTime	timestamp	筛选条件，用于将结果限定到在指定时间之前发现的结果。必须指定 startTime 和 endTime 或 taskId，但不能同时指定这两项。

输出

```
{
  "findings": [
    {
      "taskId": "string",
      "checkName": "string",
      "taskStartTime": "timestamp",
      "findingTime": "timestamp",
      "severity": "string",
      "nonCompliantResource": {
        "resourceType": "string",
        "resourceIdentifier": {
          "deviceCertificateId": "string",
          "caCertificateId": "string",
          "cognitoIdentityPoolId": "string",
          "clientId": "string",
          "policyVersionIdentifier": {
            "policyName": "string",
            "policyVersionId": "string"
          },
          "account": "string"
        },
        "additionalInfo": {
          "string": "string"
        }
      },
      "relatedResources": [
        {
          "resourceType": "string",
          "resourceIdentifier": {
            "deviceCertificateId": "string",
            "caCertificateId": "string",
            "cognitoIdentityPoolId": "string",
            "clientId": "string",

            "iamRoleArn": "string",

            "policyVersionIdentifier": {
              "policyName": "string",
              "policyVersionId": "string"
            },
            "account": "string"
          },

          "roleAliasArn": "string",

          "additionalInfo": {
            "string": "string"
          }
        }
      ],
      "reasonForNonCompliance": "string",
```

```
      "reasonForNonComplianceCode": "string"
    }
  ],
  "nextToken": "string"
}
```

CLI 输出字段

名称	类型	描述
findings	列表 成员：AuditFinding	审核的结果。
taskId	字符串 最大长度：40，最小长度：1 模式：[a-zA-Z0-9-]+	生成此结果的审核的 ID。
checkName	字符串	生成此结果的审核检查。
taskStartTime	timestamp	审核的开始时间。
findingTime	timestamp	结果的发现时间。
severity	字符串	结果的严重性。 枚举：CRITICAL | HIGH | MEDIUM | LOW
nonCompliantResource	NonCompliantResource	经发现不符合审核检查规定的资源。
resourceType	字符串	不合规资源的类型。 枚举：DEVICE_CERTIFICATE | CA_CERTIFICATE | IOT_POLICY | COGNITO_IDENTITY_POOL | CLIENT_ID | ACCOUNT_SETTINGS
resourceIdentifier	ResourceIdentifier	用于标识不合规资源的信息。
deviceCertificateId	字符串 最大长度：64，最小长度：64 模式：(0x)?[a-fA-F0-9]+	附加到资源的证书的 ID。
caCertificateId	字符串 最大长度：64，最小长度：64 模式：(0x)?[a-fA-F0-9]+	授权证书所用的 CA 证书的 ID。
cognitoIdentityPoolId	字符串	Amazon Cognito 身份池的 ID。
clientId	字符串	客户端 ID。
policyVersionIdentifier	PolicyVersionIdentifier	与资源关联的策略的版本。

名称	类型	描述
policyName	字符串 最大长度：128，最小长度：1 模式：[w+=,.@-]+	策略的名称。
policyVersionId	字符串 模式：[0-9]+	与资源关联的策略的版本 ID。
账户	字符串 最大长度：12，最小长度：12 模式：[0-9]+	资源所关联的账户。
additionalInfo	映射	有关不合规资源的其它信息。
relatedResources	列表 成员：RelatedResource	相关资源的列表。
resourceType	字符串	资源的类型。 枚举：DEVICE_CERTIFICATE | CA_CERTIFICATE | IOT_POLICY | COGNITO_IDENTITY_POOL | CLIENT_ID | ACCOUNT_SETTINGS
resourceIdentifier	ResourceIdentifier	用于标识资源的信息。
deviceCertificateId	字符串 最大长度：64，最小长度：64 模式：(0x)?[a-fA-F0-9]+	附加到资源的证书的 ID。
caCertificateId	字符串 最大长度：64，最小长度：64 模式：(0x)?[a-fA-F0-9]+	授权证书所用的 CA 证书的 ID。
cognitoIdentityPoolId	字符串	Amazon Cognito 身份池的 ID。
clientId	字符串	客户端 ID。
policyVersionIdentifier	PolicyVersionIdentifier	与资源关联的策略的版本。
iamRoleArn	字符串 最大长度：2048，最小长度：20	具有过于宽松操作的 IAM 角色的 ARN。
policyName	字符串 最大长度：128，最小长度：1 模式：[w+=,.@-]+	策略的名称。

名称	类型	描述
policyVersionId	字符串 模式：[0-9]+	与资源关联的策略的版本 ID。
roleAliasArn	字符串 最大长度：2048，最小长度：1	具有过于宽松操作的角色别名的 ARN。
账户	字符串 最大长度：12，最小长度：12 模式：[0-9]+	资源所关联的账户。
additionalInfo	映射	有关资源的其它信息。
reasonForNonCompliance	字符串	资源不合规的原因。
reasonForNonComplianceCode	字符串	用于表明资源不合规原因的代码。
nextToken	字符串	用于检索下一组结果的令牌，没有更多结果时为 `null`。

– 错误 –

`InvalidRequestException`

请求的内容无效。

`ThrottlingException`

速率超过限制。

`InternalFailureException`

出现意外错误。

审计查找结果隐藏

当您运行审计时，它会报告所有不合规资源的查找结果。这意味着您的审计报告包含您正在努力缓解问题的资源的查找结果以及已知不合规资源（如测试设备或损坏设备）的查找结果。审计将继续报告在连续审计运行中仍然不合规的资源的查找结果，这可能会向您的报告中添加不需要的信息。使用审计查找结果，您可以在定义的时间段内隐藏或筛选出查找结果，直到资源的问题被解决，若是针对与测试或损坏设备关联的资源，则可无限期地隐藏结果。

Note

缓解操作不适用于被隐藏的审计结果。有关缓解操作的更多信息，请参阅 缓解操作 (p. 995)。

有关审计检查结果隐藏配额的信息，请参阅 Amazon IoT Device Defender 端点与配额。

审计查找结果隐藏的工作原理

当您为不合规的资源创建审计查找结果隐藏时，您的审计报告和通知的行为会有所不同。

您的审计报告将包含一个新部分，其中列出与报告关联的所有隐藏的查找结果。当我们评估审计检查是否合规时，不会考虑隐藏查找结果。当您在命令行界面 (CLI) 中使用 describe-audit-task 命令时，每个审计检查也会返回隐藏的资源计数。

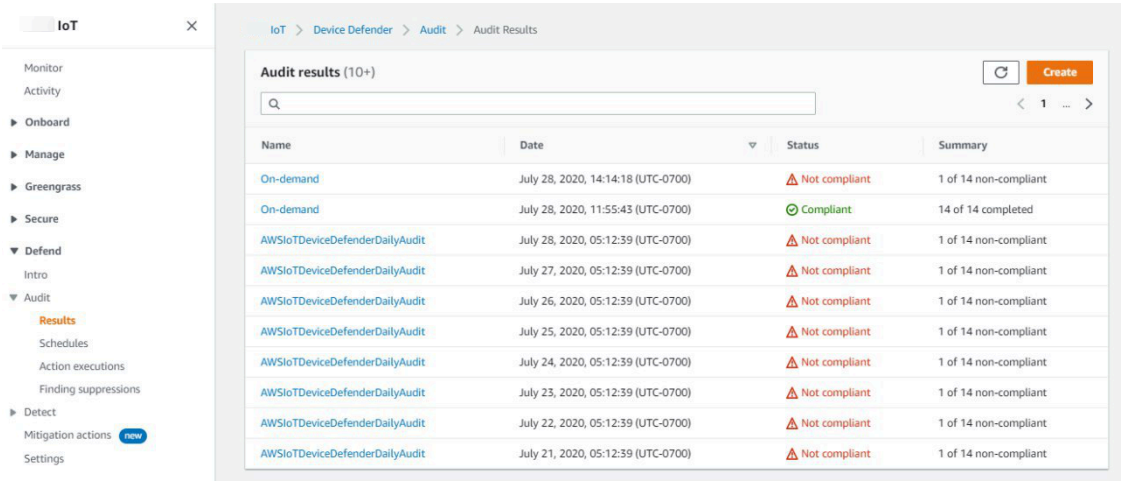
对于审计通知，当我们评估审计检查是否合规时，不会考虑隐藏的查找结果。Amazon IoT Device Defender 发布到 Amazon CloudWatch 和 Amazon Simple Notification Service (Amazon SNS) 的每个审计检查通知中也包含隐藏资源计数。

如何在控制台中使用审计查找结果隐藏

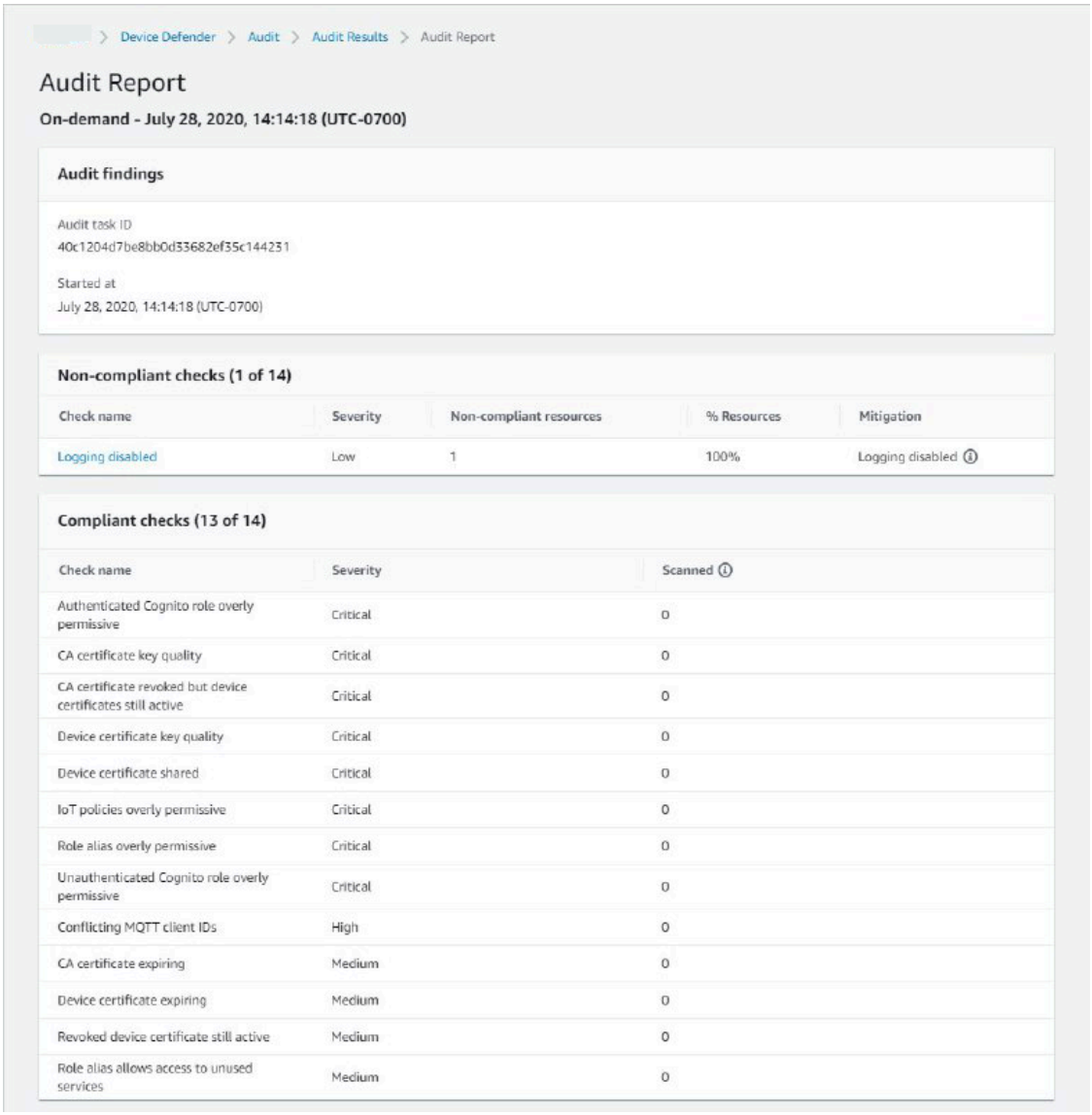
要隐藏审计报告中的查找结果

以下流程介绍了如何在 Amazon IoT 控制台中创建审计查找结果隐藏。

1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Audit（审计）、Results（结果）。
2. 选择您要查看的审计报告。



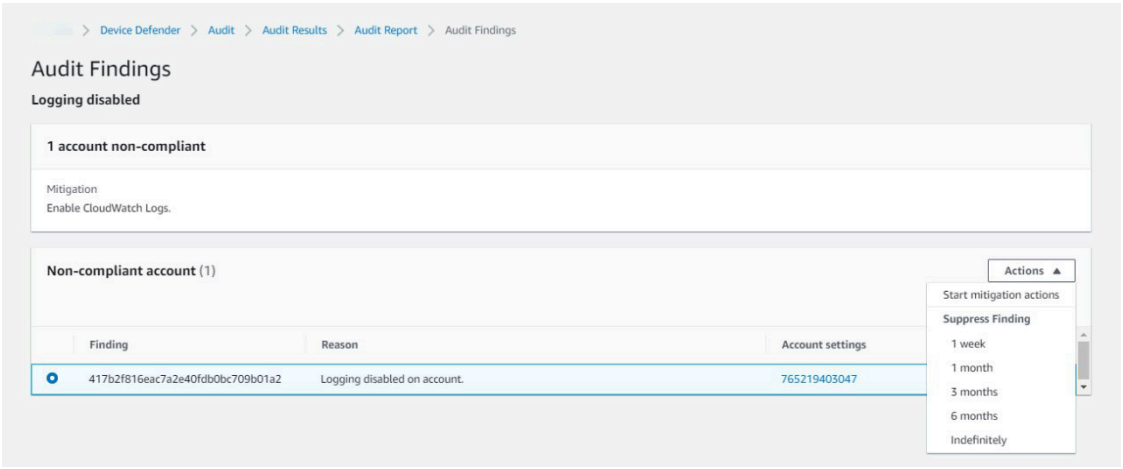
3. 在 Non-compliant checks（不合规检查）部分，在 Check name（检查名称）项下，选择您感兴趣的审计检查。



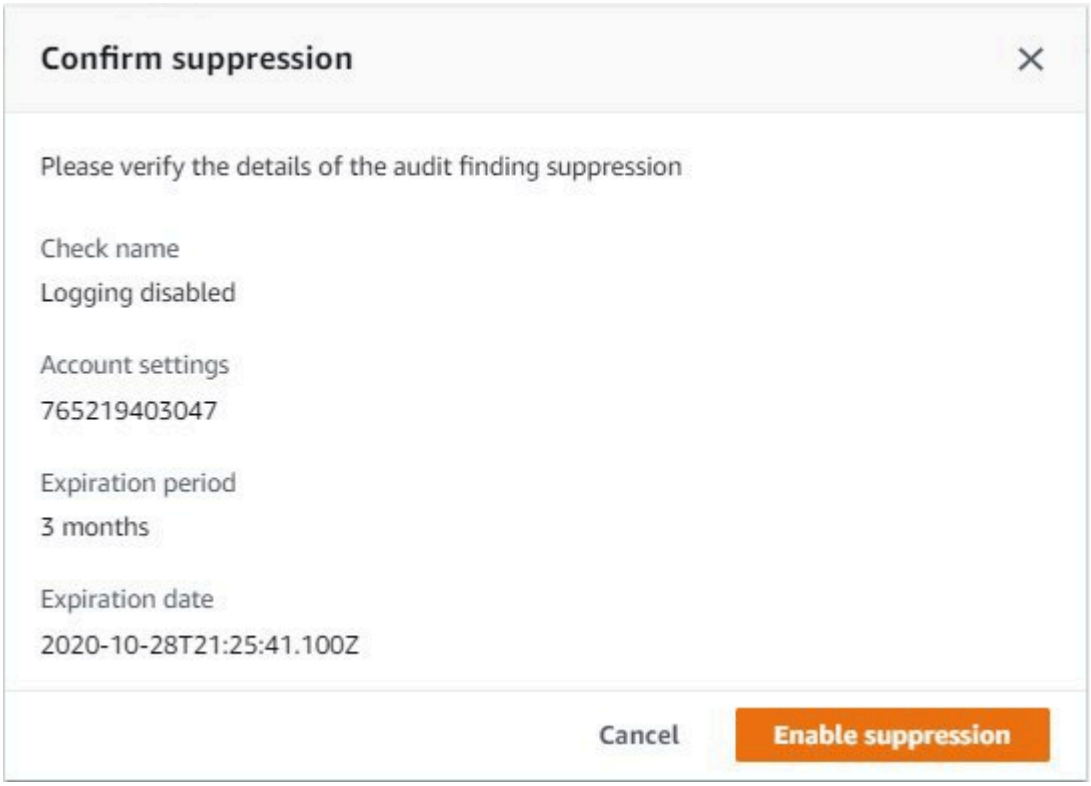
4. 在审计检查详细信息屏幕上，如果存在您不想看到的查找结果，请选择查找结果旁边的选项按钮。然后，选择 Actions（操作），接着选择您希望审计查找结果隐藏持续的时长。

   Note

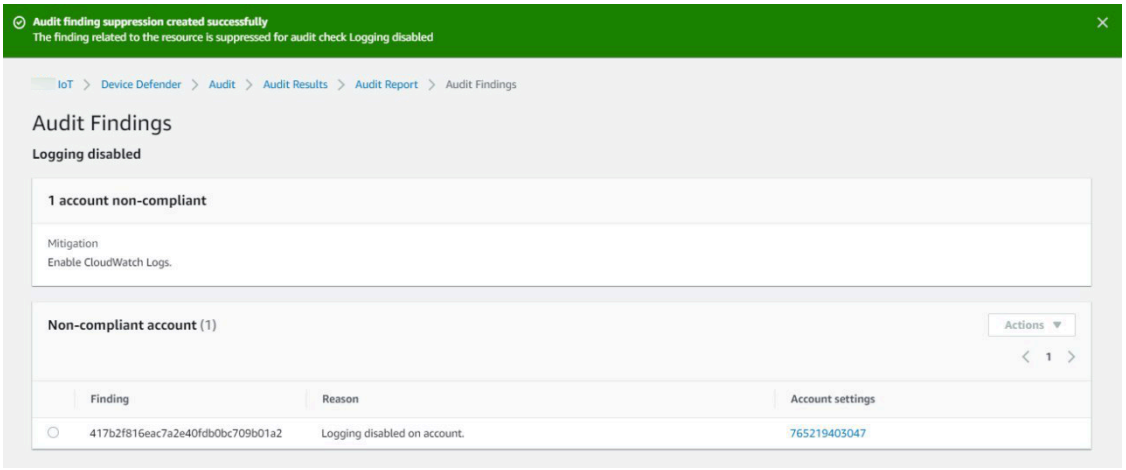
   您可以在控制台中选择 1 week（1 周）、1 month（1 个月）、3 months（3 个月）、6 months（6 个月）或 Indefinitely（无限期），作为审计查找结果隐藏的到期日期。如果要设置特定的到期日期，则只能在 CLI 或 API 中执行此操作。无论到期日期为何，您都可以随时取消审计查找结果隐藏。



5. 确认隐藏详细信息，然后选择 Enable suppresion（启用隐藏）。

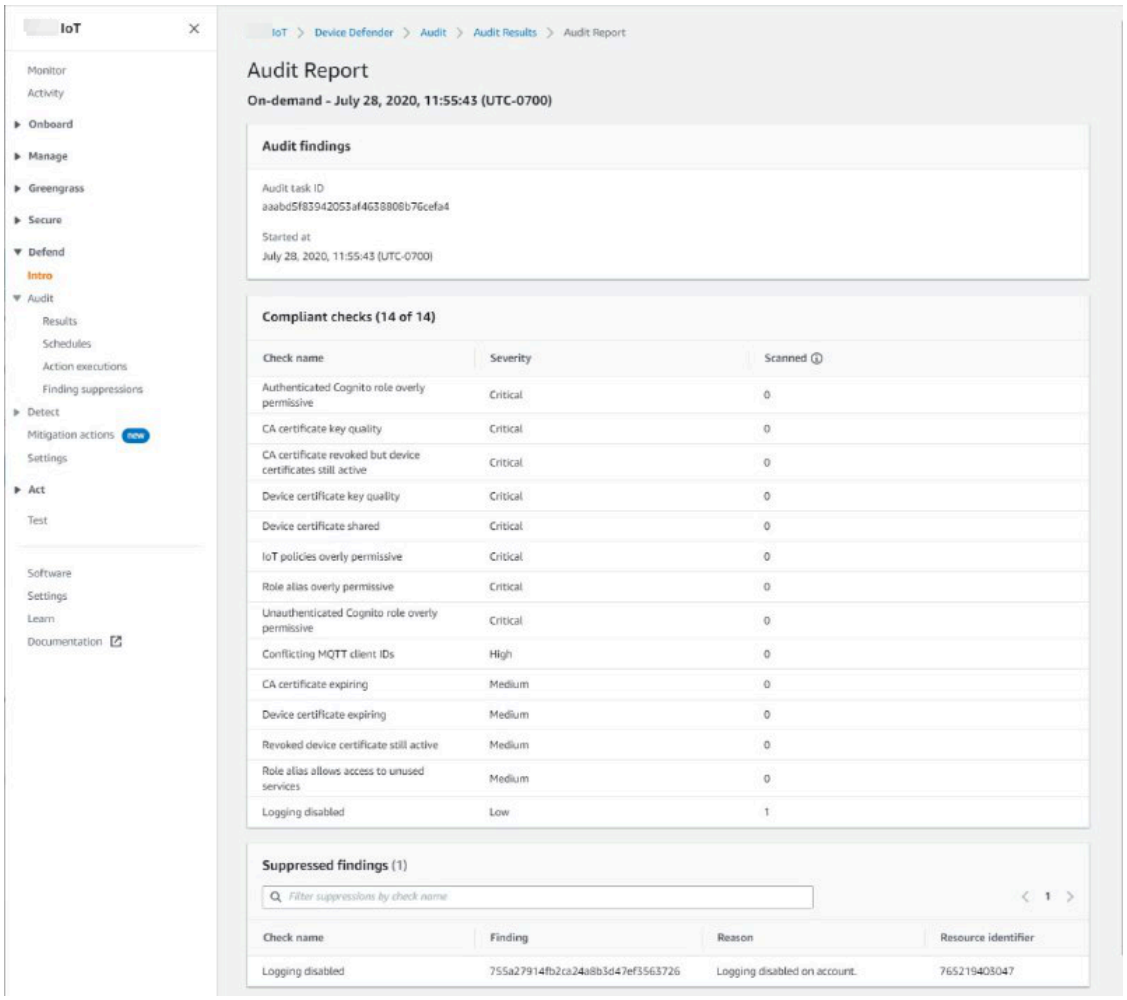


6. 创建审计查找结果隐藏后，将显示一个提示，确认审计查找结果隐藏已创建。



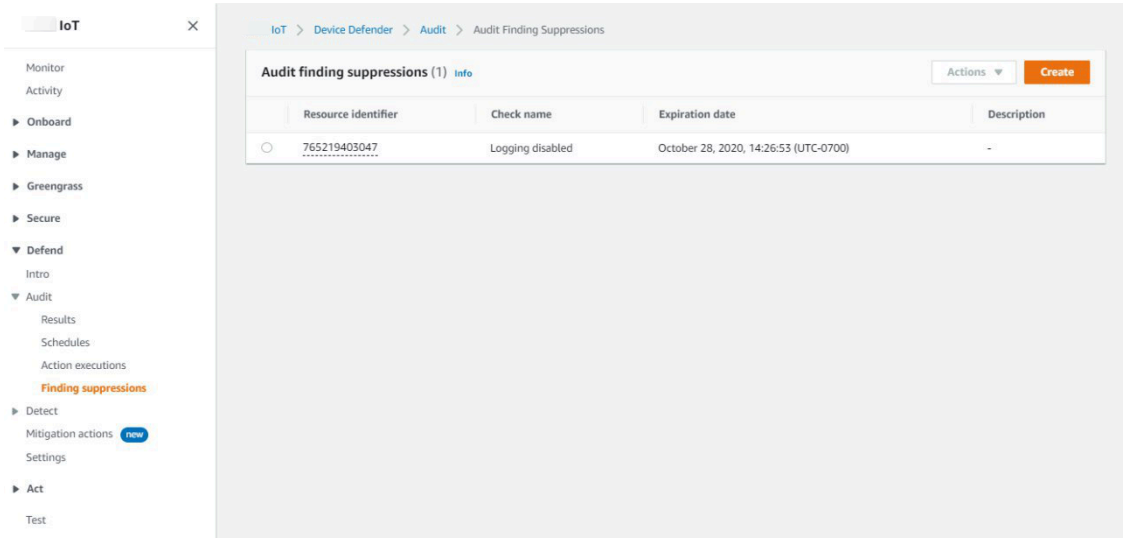
在审计报告中查看隐藏的结果

1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Audit（审计）、Results（结果）。
2. 选择您要查看的审计报告。
3. 在 Suppressed findings（隐藏查找结果）部分，查看已针对您选择的审计报告隐藏了哪些审计结果。



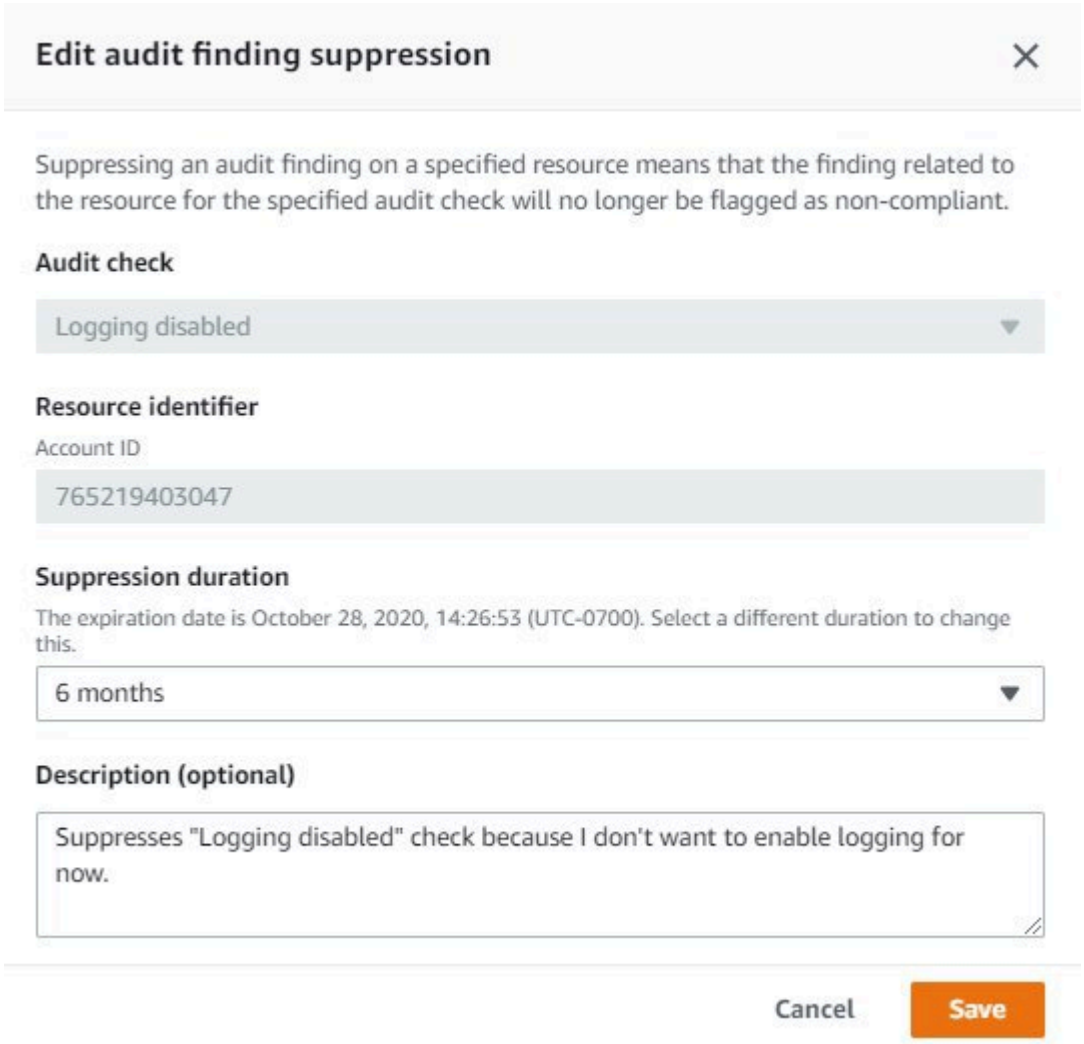
列出审计查找结果隐藏

- 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Audit（审计）、Finding suppressions（查找结果隐藏）。



要编辑审计查找结果隐藏

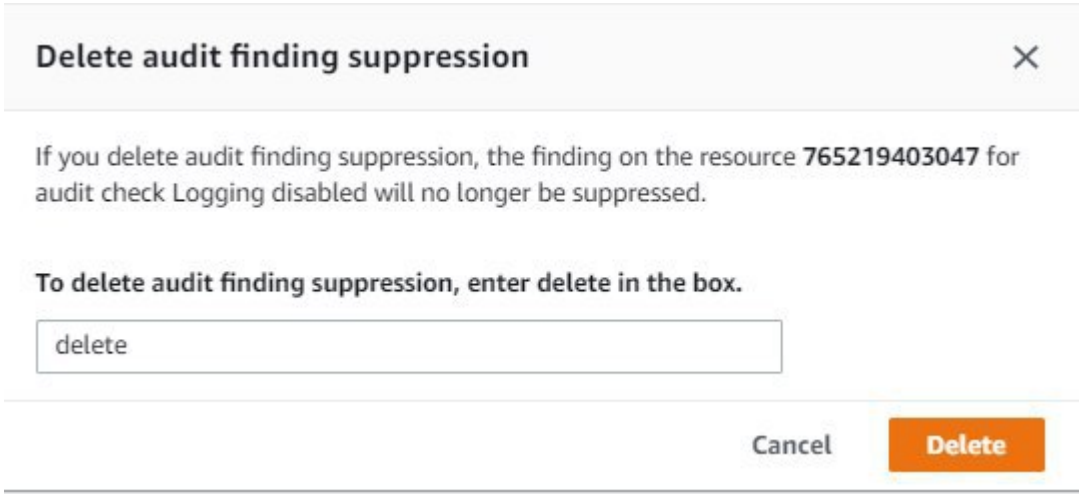
1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Audit（审计）、Finding suppressions（查找结果隐藏）。
2. 选择要编辑的审计查找结果隐藏旁边的选项按钮。下一步，选择 Actions（操作）、Edit（编辑）。
3. 在 Edit audit finding suppression（编辑查找结果隐藏）窗口中，您可以更改 Suppression duration（隐藏时长）或者 Description (optional)（描述（可选））。



4. 执行您的更改后，选择 Save（保存）。Finding suppressions（查找结果隐藏）窗口将打开。

要删除审计查找结果隐藏

1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Audit（审计）、Finding suppressions（查找结果隐藏）。
2. 选择要删除的审计查找结果隐藏旁边的选项按钮，然后选择 Actions（操作）、Delete（删除）。
3. 在 Delete audit finding suppression（删除审计查找结果隐藏）窗口中，在文本框中输入 `delete` 以确认删除，然后选择 Delete（删除）。Finding suppressions（查找结果隐藏）窗口将打开。



如何在 CLI 中使用审计查找结果隐藏

您可以使用以下 CLI 命令来创建和管理审计查找结果隐藏。

- create-audit-suppression
- describe-audit-suppression
- update-audit-suppression
- delete-audit-suppression
- list-audit-suppressions

您输入的 `resource-identifier` 取决于您需要隐藏查找结果的 `check-name`。下表详细说明了哪些检查需要哪个 `resource-identifier` 来用于创建和编辑隐藏。

> **Note**
>
> 隐藏命令不意味着关闭审计。审计仍会在您的 Amazon IoT 设备上运行。隐藏仅适用于审计结果。

check-name	resource-identifier
AUTHENTICATE_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK	cognitoIdentityPoolId
CA_CERT_APPROACHING_EXPIRATION_CHECK	caCertificateId
CA_CERTIFICATE_KEY_QUALITY_CHECK	caCertificateId
CONFLICTING_CLIENT_IDS_CHECK	clientId
DEVICE_CERT_APPROACHING_EXPIRATION_CHECK	deviceCertificateId
DEVICE_CERTIFICATE_KEY_QUALITY_CHECK	deviceCertificateId
DEVICE_CERTIFICATE_SHARED_CHECK	deviceCertificateId
IOT_POLICY_OVERLY_PERMISSIVE_CHECK	policyVersionIdentifier
IOT_ROLE_ALIAS_ALLOWS_ACCESS_TO_UNUSED_SERVICES_CHECK	roleAliasArn
IOT_ROLE_ALIAS_OVERLY_PERMISSIVE_CHECK	roleAliasArn

check-name	resource-identifier
LOGGING_DISABLED_CHECK	account
REVOKED_CA_CERT_CHECK	caCertificateId
REVOKED_DEVICE_CERT_CHECK	deviceCertificateId
UNAUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK	cognitoIdentityPoolId

要创建和应用审计查找结果隐藏

以下流程介绍如何在 Amazon CLI 中创建审计查找结果隐藏。

- 使用 `create-audit-suppression` 命令创建审计查找结果隐藏。以下示例为根据 Logging disabled（禁用的日志记录）这项检查为 Amazon Web Services 账户 *123456789012* 创建了审计查找结果隐藏。

```
aws iot create-audit-suppression \
    --check-name LOGGING_DISABLED_CHECK \
    --resource-identifier account=123456789012 \
    --client-request-token 28ac32c3-384c-487a-a368-c7bbd481f554 \
    --suppress-indefinitely \
    --description "Suppresses logging disabled check because I don't want to enable
 logging for now."
```

此命令无任何输出。

审计查找结果隐藏 API

以下 API 可用于创建和管理审计查找结果隐藏。

- CreateAuditSuppression
- DescribeAuditSuppression
- UpdateAuditSuppression
- DeleteAuditSuppression
- ListAuditSuppressions

要对特定审计查找结果进行筛选，您可以使用 ListAuditFindings API。

Detect

借助 Amazon IoT Device Defender Detect，您可以监控设备的行为，以识别可能表明设备遭到危害的异常行为。使用云端指标（来自 Amazon IoT）和设备端指标（来自您在设备上安装的代理）的组合，您可以检测到：

- 连接模式的变化。
- 与未经授权或无法识别的终端节点通信的设备。
- 入站和出站设备流量模式的变化。

您可以创建安全配置文件（其中包含设备预期行为的定义），并将它们分配到实例集中的一组设备或所有设备。Amazon IoT Device Defender Detect 使用这些安全配置文件来检测异常，并通过 Amazon CloudWatch 指标和 Amazon Simple Notification Service 通知发送提醒。

Amazon IoT Device Defender Detect 能够检测相连设备中频繁出现的安全问题：

- 从设备到已知恶意 IP 地址，或表示潜在恶意命令和控制渠道的未经授权终端节点的流量。
- 表明设备正在参与 DDoS 攻击的恶意流量，例如出站流量高峰。
- 其远程管理接口和端口支持远程访问的设备。
- 发送到您账户的消息速率激增（例如，来自可能导致过量按每条消息收费的流氓设备）。

使用案例：

评估攻击面

可以使用 Amazon IoT Device Defender Detect 评估设备的攻击面。例如，可以识别服务端口通常成为攻击活动目标的设备（在端口 23/2323 上运行的 telnet 服务，在端口 22 上运行的 SSH 服务，在端口 80/443/8080/8081 上运行的 HTTP/S 服务）。虽然在设备上使用这些服务端口可能有合法的原因，但是它们也通常是敌人攻击面的一部分，并且具有相关风险。在 Amazon IoT Device Defender Detect 向您发出攻击面告警后，您可以将攻击面降至最低（通过消除未使用的网络服务），或运行其它评估来识别安全漏洞（例如，使用常见、默认或弱密码配置的 telnet）。

检测设备行为异常以及可能的安全根本原因

您可以使用 Amazon IoT Device Defender Detect 提醒您可能表明安全违规的意外设备行为指标（开放端口数量、连接数量、意外开放端口、与意外 IP 地址的连接）。例如，TCP 连接数量高于预期，可能表明设备正被用于 DDoS 攻击。侦听的端口不是预计的端口，可能表明设备上安装了用以进行远程控制的后门。您可以使用 Amazon IoT Device Defender Detect 探测设备实例集的运行状况，并验证安全假设（例如，没有设备在侦听端口 23 或 2323）。

您可以启用基于机器学习 (ML) 的威胁检测，以自动识别潜在威胁。

检测配置错误的设备

从设备发送到账户的信息数量或大小激增，可能表明设备配置错误。此类设备可能会增加每条消息的费用。同样，如果设备多次授权失败，则可能要求重新配置策略。

监控未注册设备的行为

Amazon IoT Device Defender Detect 使得识别未在 Amazon IoT 注册表中注册的设备的异常行为成为可能。您可以定义特定于以下目标类型之一的安全配置文件：

- 所有设备
- 所有已注册的设备（Amazon IoT 注册表中的事物）
- 所有未注册的设备
- 事物组中的设备

安全配置文件为账户中的设备定义一组预期行为，并指定在检测到异常时要执行的操作。安全配置文件应附加到最具体的目标，以便您精细控制要针对该配置文件评估哪些设备。

未注册的设备在整个设备生命周期内必须提供一致的 MQTT 客户端标识符或事物名称（对于报告设备指标的设备），以便所有违规和指标都归属到相同设备。

Important

如果事物名称包含控制字符，或事物名称的长度超过 128 字节的 UTF-8 编码字符，则将拒绝设备报告的消息。

安全使用案例

本部分介绍威胁您设备机群的不同类型的攻击，以及可用于监控这些攻击的建议指标。我们建议使用指标异常作为调查安全问题的起始操作，但您不应仅仅根据指标异常来确定任何安全威胁。

要调查异常告警，请将告警详细信息与其它上下文信息（如设备属性、设备指标历史趋势、安全配置文件指标历史趋势、自定义指标和日志）相关联，以确定是否存在安全威胁。

云端使用案例

Device Defender 可以在 Amazon IoT 云端监控以下用例。

知识产权盗窃涉及盗窃个人或公司的知识产权，包括商业秘密、硬件或软件。它经常发生在设备的制造阶段。知识产权盗窃可能以盗版、设备盗窃或设备证书盗窃的形式出现。由于存在允许意外访问 IoT 资源的策略，因此可能会发生基于云的知识产权盗窃。您应检视您的 IoT 策略并启用审计过于宽容的检查以确定过于宽容的策略。

相关指标：

指标	理由
源 IP	如果设备被盗，则其源 IP 地址将超出正常供应链中流通的设备的正常预期 IP 地址范围。
收到的消息数量	由于攻击者可能会在基于云的 IP 窃取中使用设备，因此与从 Amazon IoT 云发送至设备的消息计数或消息大小有关的指标可能会激增，这表明可能存在安全问题。
消息大小	

基于 MT 的数据：

当恶意行为者从 IoT 部署或设备执行未经授权的数据传输时，就会发生数据泄露。攻击者通过 MQTT 对云端数据源发起此类攻击。

相关指标：

指标	理由
源 IP	如果设备被盗，则其源 IP 地址将超出标准供应链中流通设备的正常预期 IP 地址范围。
收到的消息数量	由于攻击者可能会在基于 MQTT 的数据泄漏中使用设备，因此与从 Amazon IoT 云发送至设备消息计数或消息大小有关的指标可能会激增，这表明可能存在安全问题。
消息大小	

模仿：

模拟攻击是指攻击者装作已知或受信任的实体，试图访问 Amazon IoT 云端服务、应用程序、数据或对 IoT 设备发布命令和加以控制。

相关指标：

指标	理由
授权失败	当攻击者使用被盗的身份装作受信任实体时，与连接相关的指标往往会激增，因为凭证可能不

指标	理由
连接尝试	再有效，或者可能已经被信任设备使用。授权失败、连接尝试或断开连接中的异常行为通常指向潜在的模拟攻击情况。
断开连接	

云基础：

Amazon IoT 云服务的滥用在发布或订阅消息量较大的主题或大型消息的主题时会发生。过度宽容的策略或设备漏洞攻击针对命令和控制，也可能导致云基础设施滥用。这次攻击的主要目标之一是增加您的Amazon账单。您应检视您的 IoT 策略并启用审计过于宽容的检查以确定过于宽容的策略。

相关指标：

指标	理由
收到的消息数量	这次攻击的目的是增加您的Amazon账单，监控消息计数、收到的消息和消息大小等活动的指标将会激增。
发送的消息数量	
消息大小	
源 IP	可能会出现可疑的源 IP 列表，攻击者将从中生成其消息收发量。

设备端使用案例

Device Defender 可以在您的设备端监控以下使用案例。

Denial-of-service 攻击：

denial-of-service (DoS) 攻击旨在关闭设备或网络，使目标用户无法访问该设备或网络。DoS 攻击通过使目标流量泛滥，或发送请求以使系统启动速度减慢或导致系统失败来阻止访问。您的 IoT 设备可用于 DoS 攻击。

相关指标：

指标	理由
数据包出去	DoS 攻击通常涉及来自给定设备较高的出站通信速率，而且根据 DoS 攻击的类型，输出数据包数和输出字节数可能会单独增加或同时增加。
字节输出	
目的地 IP	如果您定义了设备应与之通信的 IP 地址 /CIDR 范围，则目标 IP 中的异常可能表示设备出现未经授权的 IP 通信。
监听 TCP 端口	DoS 攻击通常需要更大的命令和控制基础设施，在此基础设施中，安装在设备上的恶意软件将接收有关攻击者和攻击时间的命令和信息。因此，为了接收此类信息，恶意软件通常会侦听一般不被设备使用的端口。
侦听 TP 端口计数	
监听 UDP 端口	
侦听 UP 端口计数	

横威胁：

横向威胁升级通常从攻击者获得对网络中某个点（例如连网的设备）的访问权限开始。随后，攻击者会试图通过被盗凭证或漏洞利用等方法提高其权限级别或对其它设备的访问权限。

相关指标：

指标	理由
数据包出去	在典型情况下，攻击者必须在局域网上运行扫描，以执行侦测并识别可用设备，从而缩小攻击目标的选择范围。这种扫描可能会导致字节激增和数据包输出计数。
字节输出	
目的地 IP	如果设备将与一组已知的 IP 地址或 CIDR 通信，您可以确定它是否尝试与异常 IP 地址进行通信，在横向威胁升级的使用案例中，该 IP 地址通常是本地网络上的私有 IP 地址。
授权失败	当攻击者试图提高他们在 IoT 网络中的权限级别时，他们可能会使用被撤销或已过期的被盗凭证，从而导致更多的授权失败。

数据

当恶意软件或恶意参与者从设备或网络端点进行未经授权的数据传输时，就会发生数据泄露。对于攻击者来说，数据泄露通常有两种目的：获取数据或知识产权，或者对网络进行侦测。监控意味着恶意代码被用于监控用户活动，其目的是窃取凭证和收集信息。以下指标可作为调查任一类型攻击的起点。

相关指标：

指标	理由
数据包出去	当数据泄露或监控攻击发生时，攻击者通常会对从设备发送的数据进行镜像处理，而不是简单地重新导向数据，当防御程序看不到预期数据时，这些操作将被防御程序识别。此类镜像数据会显著增加从设备发送的数据总量，从而导致输出数据包和字节计数激增。
字节输出	
目的地 IP	当攻击者使用设备进行数据泄露或监控攻击时，必须将数据发送到攻击者控制的异常 IP 地址。监控目标 IP 有助于识别此类攻击。

加密货币挖矿

攻击者利用设备的处理能力挖掘加密数字货币。加密挖掘是一个计算密集型过程，通常需要与其它采矿同行和池进行网络通信。

相关指标：

指标	理由
目的地 IP	加密挖掘过程中通常需要有网络通信。拥有一个严格控制的、设备应与之通信的 IP 地址列表可以帮助识别设备上的意外通信，如加密数字货币挖掘。

指标	理由
CPU 使用率自定义指标	加密数字货币挖掘需要依靠密集的计算，这会导致设备 CPU 出现高利用率。如果您选择收集 higher-than-normal 和监控此指标，

命令和控制、恶意软件和勒索

恶意软件或勒索软件会限制您对设备的控制，并限制您的设备功能。在出现勒索软件攻击的情况下，数据访问将会因勒索软件使用的加密手段而丢失。

相关指标：

指标	理由
目的地 IP	网络或远程攻击占 IoT 设备攻击的很大一部分。严格控制的、设备应与之通信的 IP 地址列表有助于识别恶意软件或勒索软件攻击导致的异常目标 IP。
监听 TCP 端口	多种恶意软件攻击涉及启动命令以在设备上执行。 command-and-control 此类服务器对恶意软件或勒索软件操作至关重要，可通过严密监控打开的 TCP/UDP 端口和端口计数来加以识别。
侦听 TP 端口计数	
监听 UDP 端口	
侦听 UP 端口计数	

概念

指标

Amazon IoT Device Defender Detect 使用指标来检测设备的异常行为。Amazon IoT Device DefenderDetect 将指标的报告值与您提供的预期值进行比较。这些指标可以从两个来源获取：云端指标和设备端指标。ML Detect 支持 6 个云端指标和 7 个设备端指标。有关 ML Detect 支持指标的列表，请参阅 受支持的指标 (p. 957)。

使用云端指标（例如，授权失败次数、设备通过 Amazon IoT 发送或接收的消息数量或大小）检测 Amazon IoT 网络中的异常行为。

Amazon IoT Device Defender Detect 也可以收集、聚合和监控 Amazon IoT 设备生成的指标数据（例如，设备侦听的端口、发送的字节或数据包数量，或设备的 TCP 连接）。

可以只将 Amazon IoT Device Defender Detect 与云端指标相结合使用。要使用设备端指标，必须首先在连接 Amazon IoT 的设备或设备网关上部署 Amazon IoT SDK，以收集指标并发送到 Amazon IoT。请参阅从设备发送指标 (p. 978)。

安全配置文件

安全配置文件为账户中的一组设备（事物组 (p. 246)）或所有设备定义异常行为，并指定在检测到异常时要执行的操作。您可以使用 Amazon IoT 控制台或 API 命令创建安全配置文件，并将其与一组设备关联。Amazon IoT Device DefenderDetect 开始记录安全相关数据，并使用安全配置文件中定义的行为检测设备行为中的异常情况。

行为

行为告诉 Amazon IoT Device Defender Detect 如何识别设备是否发生异常。任何设备操作与行为不匹配时，均会触发提示。规则检测行为由指标和绝对值或统计阈值以及运算符组成（例如，小于等于、大

于等于），用于描述预期的设备行为。ML Detect 行为由指标和 ML Detect 配置组成，这些行为可以设置 ML 模型来了解设备的正常行为。

ML 模型

ML 模型是一种机器学习模型，用于监控客户配置的每个行为。该模型根据目标设备组的指标数据模式进行训练，并为基于指标的行为生成三个异常置信阈值（高、中和低）。它根据设备级别的摄取指标数据推断异常。在 ML Detect 上下文中，将创建一个 ML 模型来评估一个基于指标的行为。有关更多信息，请参阅ML Detect (p. 955)。

置信级别

ML Detect 支持三个置信级别：`High`、`Medium` 和 `Low`。`High` 置信度意味着异常行为评估的灵敏度较低，而且告警数量往往较少。`Medium` 置信度意味着中等灵敏度，`Low` 置信度意味着高灵敏度和通常更高的告警数量。

维度

您可以定义维度以调整行为的作用域。例如，您可以定义一个主题筛选条件维度，该维度将行为应用于与模式匹配的 MQTT 主题。有关定义要在安全配置文件中使用的维度的信息，请参阅 CreateDimension。

告警

检测到异常时，系统会通过 CloudWatch 指标（请参阅 使用 Amazon IoT 指标 (p. 401)）或 SNS 通知发送告警通知。Amazon IoT 控制台中也会显示告警通知，以及告警的相关信息和设备告警历史记录。当监控的设备不再表现出异常行为，或者已经触发告警但在较长时间内停止报告时，系统也会发送告警。

告警验证状态

创建告警后，您可以验证告警为真、良、误报或未知。您还可以为告警验证状态添加描述。您可以通过四种验证状态之一查看、组织和筛选 Amazon IoT Device Defender 告警。您可以使用告警验证状态和相关说明来通知团队成员。这有助于您的团队采取后续行动，例如，对真告警执行缓解措施、跳过良性告警或继续对未知告警进行调查。所有告警的原定设置验证状态为 “未知”。

告警隐藏

通过将行为通知设置为 on 或者 `suppressed` 来管理 Detect 告警 SNS 通知。隐藏告警不会阻止 Detect 执行设备行为评估；Detect 会继续将异常行为标记为违规告警。但是，隐藏的告警不会转发 SNS 通知。它们只能通过 Amazon IoT 控制台或 API 访问。

行为

安全配置文件包含一组行为。每个行为都包含一个指标，用于指定账户中的一组设备或所有设备的正常行为。行为分为两类：Rule Detect 行为和 ML Detect 行为。使用 Rules Detect 行为，您可以定义设备的行为方式，而 ML Detect 使用基于历史设备数据构建的 ML 模型来评估设备应有的行为方式。

安全配置文件可以是以下两种阈值类型之一：ML 或者 基于规则。ML 安全配置文件会借助过去的数据自动检测机群中的设备级操作和安全异常。基于规则的安全配置文件要求您手动设置静态规则以监控您的设备行为。

下面描述了 `behavior` 定义中使用的一些字段：

Rules Detect 和 ML Detect 的共同点

`name`

行为的名称。

`metric`

所用指标的名称（即行为评估的对象）。

consecutiveDatapointsToAlarm

如果设备在指定数量的连续数据点违反了行为，则会发出警报。如果未指定，默认值为 1。

consecutiveDatapointsToClear

如果发出警报后，违规设备不再违反指定数量的连续数据点的行为，则警报将被解除。如果未指定，默认值为 1。

threshold type

安全性配置文件可以是两种阈值类型之一：ML 或基于规则。ML 安全配置文件会借助过去的数据自动检测机群中的设备级操作和安全异常。基于规则的安全配置文件要求您手动设置静态规则以监控您的设备行为。

alarm suppressions

通过将行为通知设置为 on 或者 suppressed 来管理 Detect 告警 SNS 通知。隐藏告警不会阻止 Detect 执行设备行为评估；Detect 会继续将异常行为标记为违规告警。但是，隐藏告警不会转发 SNS 通知。它们只能通过 Amazon IoT 控制台或 API 进行访问。

Rule Detect

`dimension`

您可以定义维度以调整行为的作用域。例如，您可以定义一个主题筛选条件维度，该维度将行为应用于与模式匹配的 MQTT 主题。要定义要在安全配置文件中使用的维度，请参阅 CreateDimension。仅适用于 Rules Detect。

`criteria`

确定设备与 `metric` 相关的行为是否正常的标准。

`comparisonOperator`

将评估的事物 (`metric`) 与标准（`value` 或 `statisticalThreshold`）关联在一起的运算符。

可能的值为：“less-than”、“less-than-equals”、“greater-than”、“greater-than-equals”、“in-cidr-set”、“not-in-cidr-set”、“in-port-set”和“not-in-port-set”。并非所有运算符对于每个指标都是有效的。针对 CIDR 集和端口的运算符仅用于与此类实体有关的指标。

`value`

与 `metric` 进行比较的值。根据不同的指标类型，应当包含 `count`（一个值）、`cidrs`（CIDR 列表）或 `ports`（端口列表）。

`statisticalThreshold`

作为行为违规判断依据的统计阈值。此字段包含一个 `statistic` 字段，后者具有以下可能的值：“p0”、“p0.1”、“p0.01”、“p1”、“p10”、“p50”、“p90”、“p99”、“p99.9”、“p99.99”或“p100”。

此 `statistic` 表示一个百分位数。它解析为用于确定符合行为的值。在指定持续时间 (`durationSeconds`) 内从与此安全配置文件关联的所有报告设备收集指标一次或多次，并根据该数据计算百分位数。之后，收集设备的测量值并在相同的持续时间内累积。如果设备的结果值高于或低于 (`comparisonOperator`) 与指定百分位数关联的值，则认为设备与行为相符。否则，该设备违反了该行为。

百分位数表示所有考虑的测量值中低于相关值的百分比。例如，如果与“p90”（第 90 个百分位数）相关的值是 123，则 90% 的测量值都低于 123。

`durationSeconds`

使用此参数为具有时间维度的条件（例如，`NUM_MESSAGES_SENT`）指定评估行为的时间段。对于 `statisticalThreshhold` 指标比较，这是一个时间段，期间将收集所有设备的测量值以确定 `statisticalThreshold` 值，然后为每个设备确定其行为在比较中的排名方式。

ML Detect

`ML Detect confidence`

ML Detect 支持三个置信级别：`High`、`Medium` 和 `Low`。`High` 置信度意味着异常行为评估的敏感度较低，同时告警数量往往较少,`Medium` 置信度意味着中等灵敏度，`Low` 置信度意味着高灵敏度和通常更高的告警数量。

ML Detect

通过机器学习 Detect (ML Detect)，您可以创建安全配置文件，使用机器学习通过基于历史设备数据自动创建模型来学习预期的设备行为，并将这些配置文件分配给一组设备或您实例集中的所有设备。Amazon IoT Device Defender 然后会使用 ML 模型识别异常并触发告警。

Note

ML Detect 现在支持监控您的实例集特有的运行状况指标。您可以使用设备端自定义指标 (p. 959)，并通过维度功能 (p. 985)对您的实例集进行更精细的监控。除了使用 Rules Detect 手动设置静态告警外，您现在还可以使用机器学习来自动了解实例集对自定义指标的预期行为。您还可以基于维度筛选出云端指标。

有关 ML Detect 入门的信息，请参阅 ML Detect 指南 (p. 856)。

本章包含以下部分：

- ML Detect 的使用案例 (p. 955)
- ML Detect 的工作原理 (p. 955)
- 最低要求 (p. 956)
- 限制 (p. 956)
- 在告警中标记误报和其它验证状态 (p. 956)
- 受支持的指标 (p. 957)
- Service Quotas (p. 957)
- ML Detect CLI 命令 (p. 957)
- ML Detect API (p. 957)
- 暂停或删除 ML Detect 安全配置文件 (p. 958)

ML Detect 的使用案例

当难以设置设备的预期行为时，您可以使用 ML Detect 来监控您的实例集设备。例如，要监控断开连接数量指标，可能不清楚什么是可接受的阈值。在这种情况下，您可以启用 ML Detect 功能，根据设备报告的历史数据来识别异常断开连接指标数据点。

ML Detect 的另一个使用案例是监控随着时间的推移动态变化的设备行为。ML Detect 根据设备中不断变更的数据模式，定期了解动态预期的设备行为。例如，发送的设备消息量在工作日和周末之间可能会有所不同，而 ML Detect 将了解此动态行为。

ML Detect 的工作原理

使用 ML Detect 功能，您可以创建行为以跨 6 个云端指标 (p. 957)和 7 个设备端指标 (p. 957)识别运营和安全异常。在初始模型训练期之后，ML Detect 会根据最后 14 天的数据每天刷新模型。它使用 ML 模型监控这些指标的数据点，并在检测到异常时触发告警。

如果将安全配置文件附加到具有类似预期行为的设备集合，则 ML Detect 将发挥出最大的作用。例如，如果您的某些设备在客户家中使用，而其他设备在企业办公室中使用，则这两个组之间的设备行为模式可能会有很大差异。您可以将设备组织到家用设备事物组和办公设备事物组。为了获得最佳异常检测效果，请将每个事物组附加到单独的 ML Detect 安全配置文件。

虽然 ML Detect 将构建初始模型，但在随后的 14 天期间内，每个指标需要 14 天和至少 25000 个数据点才能生成模型。之后，它每天都会更新模型，达到最少数量的指标数据点。如果未满足最低要求，ML Detect 会在第二天尝试构建模型，并在接下来的 30 天内每天重试，然后停止模型以进行评估。

最低要求

对于训练和创建初始 ML 模型，ML Detect 具有以下最低要求。

最低训练时间

构建初始模型需要 14 天的时间。之后，模型每天都会使用 14 天后续周期的指标数据进行刷新。

最低总数据点

在过去 14 天内，构建 ML 模型所需的最低数据点为每个指标 25000 个数据点。对于模型的持续训练和刷新，ML Detect 要求受监控设备满足最少的数据点。它大致相当于以下设置：

- 60 台设备连接并在 Amazon IoT 上每隔 45 分钟执行一次活动。
- 40 台设备则每隔 30 分钟一次。
- 15 台设备则每隔 10 分钟一次。
- 7 台设备则每隔 5 分钟一次。

设备组目标

为收集数据，您必须在安全配置文件的目标事物组中包含事物。

创建初始模型后，ML 模型每天刷新，并在 14 天的后续周期内至少需要 25000 个数据点。

限制

您可以将 ML Detect 与有关以下云端指标的维度一起使用：

- 授权失败 (aws:num-authorization-failures) (p. 981)
- 已收到的消息 (aws:num-messages-received) (p. 980)
- 已发送的消息 (aws:num-messages-sent) (p. 979)
- 消息大小 (aws:message-byte-size) (p. 978)

ML Detect 不支持以下指标。

ML Detect 不支持的云端指标：

- 源 IP (aws:source-ip-address) (p. 982)

ML Detect 不支持的设备端指标：

- 目标 IP (aws:destination-ip-addresses) (p. 970)
- 侦听 TCP 端口 (aws:listening-tcp-ports) (p. 971)
- 侦听 UDP 端口 (aws:listening-udp-ports) (p. 971)

自定义指标仅支持 number（数值）类型。

在告警中标记误报和其它验证状态

如果您通过调查验证 ML 检测告警是否为误报，可以将告警的验证状态设置为误报。这可以帮助您和您的团队识别不必回应的告警。您还可以将告警标记为“真”、“良性”或“未知”。

您可以通过Amazon IoT Device Defender控制台或使用 PutVerificationStateOnViolationAPI 操作标记警报。

受支持的指标

您可以将以下云端指标与 ML Detect 结合使用：

- 授权失败 (aws:num-authorization-failures) (p. 981)
- 连接尝试 (aws:num-connection-attempts) (p. 983)
- 断开连接 (aws:num-disconnects) (p. 984)
- 消息大小 (aws:message-byte-size) (p. 978)
- 已发送的消息 (aws:num-messages-sent) (p. 979)
- 已收到的消息 (aws:num-messages-received) (p. 980)

您可以将以下设备端指标与 ML Detect 结合使用：

- 输出字节数 (aws:all-bytes-out) (p. 964)
- 字节数 (aws:all-bytes-in) (p. 965)
- 侦听 TCP 端口计数 (aws:num-listening-tcp-ports) (p. 966)
- 侦听 UDP 端口计数 (aws:num-listening-udp-ports) (p. 967)
- 输出数据包数 (aws:all-packets-out) (p. 968)
- 数据包数 (aws:all-packets-in) (p. 969)
- 已建立的 TCP 连接计数 (aws:num-established-tcp-connections) (p. 972)

Service Quotas

有关 ML Detect 服务配额和限制的信息，请参阅 Amazon IoT Device Defender 端点和配额。

ML Detect CLI 命令

您可以使用以下 CLI 命令来创建和管理 ML Detect。

- create-security-profile
- attach-security-profile
- list-security-profiles
- describe-security-profile
- update-security-profile
- delete-security-profile
- get-behavior-model-training-摘要
- list-active-violations
- list-violation-events

ML Detect API

以下 API 可用于创建和管理 ML Detect 安全配置文件。

- CreateSecurityProfile
- AttachSecurityProfile
- ListSecurityProfiles

- DescribeSecurityProfile
- UpdateSecurityProfile
- DeleteSecurityProfile
- GetBehaviorModelTrainingSummaries
- ListActiveViolations
- ListViolationEvents
- PutVerificationStateOnViolation

暂停或删除 ML Detect 安全配置文件

您可以暂停 ML Detect 安全配置文件以暂时停止监控设备行为，也可以删除 ML Detect 安全配置文件以长时间停止监控设备行为。

使用控制台暂停 ML Detect 安全配置文件

要使用控制台暂停 ML Detect 安全配置文件，您必须首先有一个空的事物组。若要创建空事物组，请参阅 静态事物组 (p. 246)。如果已创建空事物组，则将空事物组设置为 ML Detect 安全配置文件的目标。

> Note
>
> 您需要在 30 天内将安全配置文件的目标设置回具有设备的设备组，否则您将无法重新激活安全配置文件。

使用控制台删除 ML Detect 安全配置文件

若要删除安全配置文件，请按照下列步骤操作：

1. 在 Amazon IoT 控制台导航到边栏，然后选择 Defend（防护）部分。
2. 在 Defend（防护）中，选择 Detect（检测）然后选择 Security Profiles（安全配置文件）。
3. 选择要删除的 ML Detect 安全配置文件。
4. 选择 Actions（操作），然后从选项中选择 Delete（删除）。

> Note
>
> 删除 ML Detect 安全配置文件后，您将无法重新激活安全配置文件。

使用 CLI 暂停 ML Detect 安全配置文件

要使用 CLI 暂停 ML Detect 安全配置文件，请使用 `detach-security-security-profile` 命令：

```
$aws iot detach-security-profile --security-profile-name SecurityProfileName --
security-profile-target-arn arn:aws:iot:us-east-1:123456789012:all/registered-things
```

> Note
>
> 此选项只在 Amazon CLI 中可用。与控制台工作流程类似，您需要在 30 天内将安全配置文件的目标设置回具有设备的设备组，否则您将无法重新激活安全配置文件。要将安全配置文件附加到设备组，请使用 `attach-security-profile` 命令。

使用 CLI 删除 ML Detect 安全配置文件

您可以通过使用如下 `delete-security-profile` 命令删除安全配置文件：

```
delete-security-profile --security-profile-name SecurityProfileName
```

Note

删除 ML Detect 安全配置文件后，您将无法重新激活安全配置文件。

自定义指标

借助 Amazon IoT Device Defender 自定义指标，您可以定义和监控实例集或使用案例独有的指标，例如连接到 Wi-Fi 网关的设备数量、电池的充电量或智能插头的电源周期次数。自定义指标行为在安全配置文件中定义，其为设备组（事物组）或所有设备指定了预期行为。您可以通过设置告警来监控行为，该告警可用来检测和响应特定于设备的问题。

本章包含以下部分：

- 如何在控制台中使用自定义指标 (p. 959)
- 如何使用 CLI 中的自定义指标 (p. 961)
- 自定义指标 CLI 命令 (p. 963)
- 自定义指标 API (p. 963)

如何在控制台中使用自定义指标

教程

- Amazon IoT Device Defender 代理 SDK (Python) (p. 959)
- 创建自定义指标并将其添加到安全配置文件 (p. 959)
- 查看自定义指标详细信息 (p. 960)
- 更新自定义指标 (p. 960)
- 删除自定义指标 (p. 960)

Amazon IoT Device Defender 代理 SDK (Python)

首先，请下载 Amazon IoT Device Defender 代理 SDK (Python) 示例代理。代理将收集指标并发布报告。设备端指标发布后，您可以查看正在收集的指标并确定设置告警的阈值。有关设置设备代理的说明，请参阅 Amazon IoT Device Defender 代理 SDK (Python) 自述文件。有关更多信息，请参阅 Amazon IoT Device Defender 代理 SDK (Python)。

创建自定义指标并将其添加到安全配置文件

以下流程介绍了如何使用控制台创建自定义指标。

1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Metrics（指标）。
2. 在 Custom metrics（自定义指标）页面上，选择 Create（创建）。
3. 在 Create custom metric（创建自定义指标）页面上，执行以下操作。
    1. 在 Name（名称）项下，输入自定义指标的名称。创建自定义指标之后，您无法修改此名称。
    2. 在 Display name (optional)（显示名称（可选））项下，您可以输入自定义指标的友好名称。它不需要是唯一的，并且可以在创建后进行修改。
    3. 在 Type（类型）中，选择要监控的指标类型。指标类型包括 string-list（字符串列表）、ip-address-list（IP 地址列表）、number-list（编号列表）和 number（编号）。创建后无法修改类型。

        Note

        ML Detect 仅允许 number（数值）类型。
    4. 在 Tags（标签）项下，您可以选择要与资源关联的标签。

完成后，选择 Confirm（确认）。

4. 创建自定义指标后，将会显示 Custom metrics（自定义指标）页面，您可以在其中查看新创建的自定义指标。
5. 接下来，您需要将自定义指标添加到安全配置文件。在 Amazon IoT 控制台中的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Security profiles（安全配置文件）。
6. 选择您要添加自定义指标的安全配置文件。
7. 选择 Actions（操作）和 Edit（编辑）。
8. 选择 Addtional Metrics to retain（要保留的其它指标），然后选择您的自定义指标。在以下页面选择 Next（下一步），直到您到达 Confirm（确认）页面。选择 Save（保存）和 Continue（继续）。成功添加自定义指标后，将会显示安全配置文件详细信息页面。

   Note

   当任何指标值为负数时，百分位数统计数据不可用于指标。

查看自定义指标详细信息

以下流程介绍了如何在控制台中查看自定义指标的详细信息。

1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Metrics（指标）。
2. 选择您想要查看详细信息的自定义指标的 Metric name（指标名称）。

更新自定义指标

以下流程说明如何使用控制台更新自定义指标。

1. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Metrics（指标）。
2. 选择要更新的自定义指标旁边的选项按钮。然后，在 Actions（操作）中，选择 Edit（编辑）。
3. 在 Update custom metrics（更新自定义指标）页面上，您可以编辑显示名称及删除或添加标签。
4. 完成后，选择 Update（更新）。Custom metrics（自定义指标）页面。

删除自定义指标

以下流程介绍了如何使用控制台删除自定义指标。

1. 首先，从引用的任何安全配置文件中删除您的自定义指标。您可以在自定义指标详细信息页面上查看哪些安全配置文件包含了您的自定义指标。在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Metrics（指标）。
2. 选择您要删除的自定义指标。从自定义指标详细新页面上的 Security Profiles（安全配置文件）下列示的任何安全配置文件中删除自定义指标。
3. 在 Amazon IoT 控制台的导航窗格中，展开 Defend（防护），然后选择 Detect（检测）、Metrics（指标）。
4. 选择要删除的自定义指标旁边的选项按钮。然后，对于 Actions（操作），选择 Delete（删除）。
5. 在 Are you sure you want to delete custom metric?（是否确定要删除自定义指标）消息上，选择 Delete custom metric（删除自定义指标）。

   Warning

   删除自定义指标后，将丢失与该指标关联的所有数据。此操作无法撤消。

如何使用 CLI 中的自定义指标

教程

- Amazon IoT Device Defender 代理 SDK (Python) (p. 961)
- 创建自定义指标并将其添加到安全配置文件 (p. 961)
- 查看自定义指标详细信息 (p. 962)
- 更新自定义指标 (p. 962)
- 删除自定义指标 (p. 962)

Amazon IoT Device Defender 代理 SDK (Python)

首先，请下载 Amazon IoT Device Defender 代理 SDK (Python) 示例代理。代理将收集指标并发布报告。发布设备端指标后，您可以查看正在收集的指标并确定设置告警的阈值。有关设置设备代理的说明，请参阅 Amazon IoT Device Defender 代理 SDK (Python) 自述文件。有关更多信息，请参阅 Amazon IoT Device Defender 代理 SDK (Python)。

创建自定义指标并将其添加到安全配置文件

以下流程介绍如何从 CLI 创建自定义指标并将其添加到安全配置文件。

1. 使用 `create-custom-metric` 命令创建自定义指标。以下示例创建用于测量电池百分比的自定义指标。

```
aws iot create-custom-metric \
    --metric-name "batteryPercentage" \
    --metric-type "number" \
    --display-name "Remaining battery percentage." \
    --region us-east-1
    --client-request-token "02ccb92b-33e8-4dfa-a0c1-35b181ed26b0" \
```

输出：

```
{
    "metricName": "batteryPercentage",
    "metricArn": "arn:aws:iot:us-east-1:1234564789012:custommetric/batteryPercentage"
}
```

2. 创建自定义量度指标后，您可以使用 `update-security-profile` 将自定义指标添加到现有配置文件或使用`create-security-profile` 创建新的配置文件以便添加自定义指标。在这里，我们新建一个名为 *batteryUsage* 的安全配置文件，以便将我们的新 *batteryPercentage* 自定义指标添加到其中。我们还添加了一个名为 *cellularBandwidth* 的 Rules Detect 指标。

```
aws iot create-security-profile \
    --security-profile-name batteryUsage \
    --security-profile-description "Shows how much battery is left in percentile."  \
    --behaviors "[{\"name\":\"great-than-75\",\"metric\":\"batteryPercentage\",
\"criteria\":{\"comparisonOperator\":\"greater-than\",\"value\":{\"number
\":75},\"consecutiveDatapointsToAlarm\":5,\"consecutiveDatapointsToClear
\":1}},{\"name\":\"cellularBandwidth\",\"metric\":\"aws:message-byte-size\",
\"criteria\":{\"comparisonOperator\":\"less-than\",\"value\":{\"count\":128},
\"consecutiveDatapointsToAlarm\":1,\"consecutiveDatapointsToClear\":1}}]" \
    --region us-east-1
```

输出：

```
{
```

```
    "securityProfileArn": "arn:aws:iot:us-east-1:1234564789012:securityprofile/
batteryUsage",
    "securityProfileName": "batteryUsage"
}
```

Note

当任何指标值为负数时，百分位数统计数据不可用于指标。

查看自定义指标详细信息

以下流程介绍如何从 CLI 查看自定义指标的详细信息。

- 使用 list-custom-metrics 命令查看所有自定义指标。

```
aws iot list-custom-metrics \
    --region us-east-1
```

此命令的输出如下所示。

```
{
    "metricNames": [
        "batteryPercentage"
    ]
}
```

更新自定义指标

以下流程介绍如何从 CLI 更新自定义指标。

- 使用 update-custom-metric 命令更新自定义指标。以下示例将更新 display-name。

```
aws iot update-custom-metric \
    --metric-name batteryPercentage \
    --display-name 'remaining battery percentage on device' \
    --region us-east-1
```

此命令的输出如下所示。

```
{
    "metricName": "batteryPercentage",
    "metricArn": "arn:aws:iot:us-east-1:1234564789012:custommetric/batteryPercentage",
    "metricType": "number",
    "displayName": "remaining battery percentage on device",
    "creationDate": "2020-11-17T23:01:35.110000-08:00",
    "lastModifiedDate": "2020-11-17T23:02:12.879000-08:00"
}
```

删除自定义指标

以下流程介绍如何从 CLI 中删除自定义指标。

1. 要删除自定义指标，请先将其从附加到的任何安全配置文件中删除。使用 list-security-profiles 命令查看具有特定自定义指标的安全配置文件。

2. 要从安全配置文件中删除自定义指标，请使用 update-security-profiles 命令。输入要保留的所有信息，但排除自定义指标。

```
aws iot update-security-profile \
  --security-profile-name batteryUsage \
  --behaviors "[{\"name\":\"cellularBandwidth\",\"metric\":\"aws:message-byte-size
\",\"criteria\":{\"comparisonOperator\":\"less-than\",\"value\":{\"count\":128},
\"consecutiveDatapointsToAlarm\":1,\"consecutiveDatapointsToClear\":1}}]"
```

此命令的输出如下所示。

```
{
  "behaviors": [{\"name\":\"cellularBandwidth\",\"metric\":\"aws:message-byte-size
\",\"criteria\":{\"comparisonOperator\":\"less-than\",\"value\":{\"count\":128},
\"consecutiveDatapointsToAlarm\":1,\"consecutiveDatapointsToClear\":1}}],
  "securityProfileName": "batteryUsage",
  "lastModifiedDate": 2020-11-17T23:02:12.879000-09:00,
  "securityProfileDescription": "Shows how much battery is left in percentile.",
  "version": 2,
  "securityProfileArn": "arn:aws:iot:us-east-1:1234564789012:securityprofile/
batteryUsage",
  "creationDate": 2020-11-17T23:02:12.879000-09:00
}
```

3. 分离自定义指标后，使用 delete-custom-metric 命令删除自定义指标。

```
aws iot delete-custom-metric  \
  --metric-name batteryPercentage \
  --region us-east-1
```

此命令的输出如下所示

```
HTTP 200
```

自定义指标 CLI 命令

可以使用以下 CLI 命令来创建和管理自定义指标。

- create-custom-metric
- describe-custom-metric
- list-custom-metrics
- update-custom-metric
- delete-custom-metric
- list-security-profiles

自定义指标 API

以下 API 可用于创建和管理自定义指标。

- CreateCustomMetric
- DescribeCustomMetric
- ListCustomMetrics
- UpdateCustomMetric

- DeleteCustomMetric
- ListSecurityProfiles

设备端指标

创建安全配置文件时，您可以通过为 IoT 设备生成的指标配置行为和阈值来指定 IoT 设备的预期行为。以下是设备端指标，它们是来自您在设备上安装的代理的指标。

输出字节数 (aws:all-bytes-out)

给定时间段内从设备发出的出站字节数量。

使用此指标指定给定时间段内设备应该发送的最大或最小出站流量（以字节为单位）。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：字节

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "TCP outbound traffic",
  "metric": "aws:all-bytes-out",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 4096
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **statisticalThreshold** 的示例

```
{
  "name": "TCP outbound traffic",
  "metric": "aws:all-bytes-out",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p50"
    },
    "durationSeconds": 900,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Outbound traffic ML behavior",
  "metric": "aws:all-bytes-out",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

字节数 (aws:all-bytes-in)

给定时间段内发往设备的入站字节数量。

使用此指标指定给定时间段内设备应该接收的最大或最小入站流量（以字节为单位）。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：字节

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "TCP inbound traffic",
  "metric": "aws:all-bytes-in",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 4096
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **statisticalThreshold** 的示例

```
{
  "name": "TCP inbound traffic",
  "metric": "aws:all-bytes-in",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p90"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
```

```
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Inbound traffic ML behavior",
  "metric": "aws:all-bytes-in",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

侦听 TCP 端口计数 (aws:num-listening-tcp-ports)

设备正在侦听的 TCP 端口数量。

使用此指标指定每个设备应该监控的最大 TCP 端口数量。

兼容：Rules Detect | ML Detect

单位：失败次数

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：失败次数

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "Max TCP Ports",
  "metric": "aws:num-listening-tcp-ports",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 5
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **statisticalThreshold** 的示例

```
{
  "name": "Max TCP Ports",
  "metric": "aws:num-listening-tcp-ports",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
```

```
      "statistic": "p50"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Max TCP Port ML behavior",
  "metric": "aws:num-listening-tcp-ports",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

侦听 UDP 端口计数 (aws:num-listening-udp-ports)

设备正在侦听的 UDP 端口数量。

使用此指标指定每个设备应该侦听的最大 UDP 端口数量。

兼容：Rules Detect | ML Detect

单位：失败次数

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：失败次数

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "Max UDP Ports",
  "metric": "aws:num-listening-udp-ports",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 5
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **statisticalThreshold** 的示例

```
{
```

```
  "name": "Max UDP Ports",
  "metric": "aws:num-listening-udp-ports",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p50"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Max UPD Port ML behavior",
  "metric": "aws:num-listening-tcp-ports",
  "criteria": {
  "consecutiveDatapointsToAlarm": 1,
  "consecutiveDatapointsToClear": 1,
  "mlDetectionConfig": {
    "confidenceLevel": "HIGH"
   }
 },
  "suppressAlerts": true
}
```

输出数据包数 (aws:all-packets-out)

给定时间段内从设备发出的出站数据包数量。

使用此指标指定给定时间段内设备应该发送的最大或最小出站总流量。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：数据包

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "TCP outbound traffic",
  "metric": "aws:all-packets-out",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 100
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **`statisticalThreshold`** 的示例

```
{
  "name": "TCP outbound traffic",
  "metric": "aws:all-packets-out",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p90"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Outbound sent ML behavior",
  "metric": "aws:all-packets-out",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

数据包数 (aws:all-packets-in)

给定时间段内发往设备的入站数据包数量。

使用此指标指定给定时间段内设备应该接收的最大或最小入站总流量。

兼容：Rule Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：数据包

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "TCP inbound traffic",
  "metric": "aws:all-packets-in",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 100
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
```

```
  },
  "suppressAlerts": true
}
```

Example

使用 statisticalThreshold 的示例

```
{
  "name": "TCP inbound traffic",
  "metric": "aws:all-packets-in",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p90"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Inbound sent ML behavior",
  "metric": "aws:all-packets-in",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

目标 IP (aws:destination-ip-addresses)

一组目标 IP 地址。

使用此指标指定一组允许（以前称为列入白名单）或拒绝（以前称为列入黑名单）的无类域间路由 (CIDR)，每个设备必须或不得通过它们连接到 Amazon IoT。

兼容：Rule Detect

运算符：in-cidr-set | not-in-cidr-set

值：CIDR 列表

单位：n/a

Example

```
{
  "name": "Denied source IPs",
  "metric": "aws:destination-ip-address",
  "criteria": {
    "comparisonOperator": "not-in-cidr-set",
    "value": {
```

```
      "cidrs": [ "12.8.0.0/16", "15.102.16.0/24" ]
    }
  },
  "suppressAlerts": true
}
```

侦听 TCP 端口 (aws:listening-tcp-ports)

设备正在侦听的 TCP 端口。

使用此指标指定一组允许（以前称为列入白名单）或拒绝（以前称为列入黑名单）的 TCP 端口，每个设备必须或不得通过它们进行侦听。

兼容：Rule Detect

运算符：in-port-set | not-in-port-set

值：端口列表

单位：n/a

Example

```
{
  "name": "Listening TCP Ports",
  "metric": "aws:listening-tcp-ports",
  "criteria": {
    "comparisonOperator": "in-port-set",
    "value": {
      "ports": [ 443, 80 ]
    }
  },
  "suppressAlerts": true
}
```

侦听 UDP 端口 (aws:listening-udp-ports)

设备正在侦听的 UDP 端口。

使用此指标指定一组允许（以前称为列入白名单）或拒绝（以前称为列入黑名单）的 UDP 端口，每个设备必须或不得通过它们进行侦听。

兼容：Rule Detect

运算符：in-port-set | not-in-port-set

值：端口列表

单位：n/a

Example

```
{
  "name": "Listening UDP Ports",
  "metric": "aws:listening-udp-ports",
  "criteria": {
    "comparisonOperator": "in-port-set",
    "value": {
      "ports": [ 1025, 2000 ]
    }
  }
```

```
}
```

已建立的 TCP 连接计数 (aws:num-established-tcp-connections)

设备的 TCP 连接数。

使用此指标指定每个设备应该具有的最大或最小活动 TCP 连接数量（所有 TCP 状态）。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：连接

Example

```
{
  "name": "TCP Connection Count",
  "metric": "aws:num-established-tcp-connections",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 3
    },
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **statisticalThreshold** 的示例

```
{
  "name": "TCP Connection Count",
  "metric": "aws:num-established-tcp-connections",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p90"
    },
    "durationSeconds": 900,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Connection count ML behavior",
  "metric": "aws:num-established-tcp-connections",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
```

```
    }
  },
  "suppressAlerts": true
}
```

设备指标文档规范

整体结构

长名称	短名称	必填	类型	约束	注意
header	hed	Y	对象		格式正确的报告所必需的完整数据块。
指标	met	Y	对象		报告可以同时包含两个或至少一个 `metrics` 或者 `custom_metrics` 数据块。
custom_metrics	cmet	Y	对象		报告可以同时包含两个或至少一个 `metrics` 或者 `custom_metrics` 数据块。

标头数据块

长名称	短名称	必填	类型	约束	注意
report_id	rid	Y	整数		单调递增值。推荐采用纪元时间戳。
version	v	Y	字符串	Major.Minor	添加字段时为次要增量。如果指标移除，则为主要增量。

指标数据块：

TCP 连接

长名称	短名称	父元素	必填	类型	约束	注意
tcp_connections	tc	指标	否	对象		
established_connections	ec	tcp_connections	否	对象		已建立 TCP 状态
connections	cs	established_connections	否	List<Object>		
remote_addr	rad	connections	Y	数字	ip:port	IP 可能是 IPv6 或 IPv4

长名称	短名称	父元素	必填	类型	约束	注意
local_port	lp	connections	否	数字	>= 0	
local_interface	li	connections	否	字符串		接口名称
total	t	established_connections	否	数字	>= 0	已建立的连接数

侦听 TCP 端口

长名称	短名称	父元素	必填	类型	约束	注意
listening_tcp_ports	lpts	指标	否	对象		
ports	pts	listening_tcp_ports	否	List<Object>	> 0	
port	pt	ports	否	数字	> 0	端口应该是大于 0 的数字
interface	if	ports	否	字符串		接口名称
total	t	listening_tcp_ports	否	数字	>= 0	

侦听 UDP 端口

长名称	短名称	父元素	必填	类型	约束	注意
listening_udp_ports	lpts	指标	否	对象		
ports	pts	listening_udp_ports	否	List<Port>	> 0	
port	pt	ports	否	数字	> 0	端口应该是大于 0 的数字
interface	if	ports	否	字符串		接口名称
total	t	listening_udp_ports	否	数字	>= 0	

网络统计数据

长名称	短名称	父元素	必填	类型	约束	注意
network_stats	ns	metrics	否	对象		
bytes_in	bi	network_stats	否	数字	Delta Metric, >= 0	
bytes_out	bo	network_stats	否	数字	Delta Metric, >= 0	
packets_in	pi	network_stats	否	数字	Delta Metric, >= 0	
packets_out	po	network_stats	否	数字	Delta Metric, >= 0	

Example

以下 JSON 结构使用长名称。

```
{
  "header": {
    "report_id": 1530304554,
    "version": "1.0"
  },
  "metrics": {
    "listening_tcp_ports": {
      "ports": [
        {
          "interface": "eth0",
          "port": 24800
        },
        {
          "interface": "eth0",
          "port": 22
        },
        {
          "interface": "eth0",
          "port": 53
        }
      ],
      "total": 3
    },
    "listening_udp_ports": {
      "ports": [
        {
          "interface": "eth0",
          "port": 5353
        },
        {
          "interface": "eth0",
          "port": 67
        }
      ],
      "total": 2
    },
    "network_stats": {
      "bytes_in": 29358693495,
      "bytes_out": 26485035,
      "packets_in": 10013573555,
      "packets_out": 11382615
    },
    "tcp_connections": {
      "established_connections": {
        "connections": [
          {
            "local_interface": "eth0",
            "local_port": 80,
            "remote_addr": "192.168.0.1:8000"
          },
          {
            "local_interface": "eth0",
            "local_port": 80,
            "remote_addr": "192.168.0.1:8000"
          }
        ],
        "total": 2
      }
    }
  },
  "custom_metrics": {
```

```
    "MyMetricOfType_Number": [
      {
        "number": 1
      }
    ],
    "MyMetricOfType_NumberList": [
      {
        "number_list": [
          1,
          2,
          3
        ]
      }
    ],
    "MyMetricOfType_StringList": [
      {
        "string_list": [
          "value_1",
          "value_2"
        ]
      }
    ],
    "MyMetricOfType_IpList": [
      {
        "ip_list": [
          "172.0.0.0",
          "172.0.0.10"
        ]
      }
    ]
  }
}
```

Example 使用短名称的 JSON 结构示例

```
{
  "hed": {
    "rid": 1530305228,
    "v": "1.0"
  },
  "met": {
    "tp": {
      "pts": [
        {
          "if": "eth0",
          "pt": 24800
        },
        {
          "if": "eth0",
          "pt": 22
        },
        {
          "if": "eth0",
          "pt": 53
        }
      ],
      "t": 3
    },
    "up": {
      "pts": [
        {
          "if": "eth0",
          "pt": 5353
        },
```

```
        {
          "if": "eth0",
          "pt": 67
        }
      ],
      "t": 2
    },
    "ns": {
      "bi": 29359307173,
      "bo": 26490711,
      "pi": 10014614051,
      "po": 11387620
    },
    "tc": {
      "ec": {
        "cs": [
          {
            "li": "eth0",
            "lp": 80,
            "rad": "192.168.0.1:8000"
          },
          {
            "li": "eth0",
            "lp": 80,
            "rad": "192.168.0.1:8000"
          }
        ],
        "t": 2
      }
    }
  },
  "cmet": {
    "MyMetricOfType_Number": [
      {
        "number": 1
      }
    ],
    "MyMetricOfType_NumberList": [
      {
        "number_list": [
          1,
          2,
          3
        ]
      }
    ],
    "MyMetricOfType_StringList": [
      {
        "string_list": [
          "value_1",
          "value_2"
        ]
      }
    ],
    "MyMetricOfType_IpList": [
      {
        "ip_list": [
          "172.0.0.0",
          "172.0.0.10"
        ]
      }
    ]
  }
}
```

从设备发送指标

Amazon IoT Device Defender Detect 可以收集、聚合和监控 Amazon IoT 设备生成的指标数据，以识别表现出异常行为的设备。本部分介绍如何将指标从设备发送到 Amazon IoT Device Defender。

您必须在连接 Amazon IoT 的设备或设备网关上安全部署 Amazon IoT SDK 版本二，以收集设备端指标。查看在此处查看 SDK 的完整列表。

您可以使用 Amazon IoT Device Client 发布指标，因为它提供的单个代理涵盖了 Amazon IoT Device Defender 和 Amazon IoT Device Management 中的功能。这些功能包括任务、安全隧道、Amazon IoT Device Defender 指标发布等。

您可以将设备端指标发布到 Amazon IoT 中的预留主题，便于 Amazon IoT Device Defender 进行收集和评估。

使用 Amazon IoT Device Client 发布指标

要安装 Amazon IoT Device Client，您可以从 Github 下载。在要收集设备端数据的设备上安装 Amazon IoT 后，您必须进行配置以将设备端指标发送到 Amazon IoT Device Defender。确认 Amazon IoT Device Client 配置文件在 `device-defender` 部分设置了以下参数：

```
"device-defender":    {
      "enabled":    true,
      "interval-in-seconds": 300
  }
```

> **Warning**
>
> 时间间隔至少应设置为 300 秒。如果将时间间隔设置为小于 300 秒，则您的指标数据可能会被节流。

更新配置后，您可以在 Amazon IoT Device Defender 控制台创建安全配置文件和行为来监控设备发布到云的指标。您可以依次选择 Defend（防护）、Detect（检测）和 Metrics（指标），从而在 Amazon IoT Core 控制台中发布指标。

云端指标

创建安全配置文件时，您可以通过为 IoT 设备生成的指标配置行为和阈值来指定 IoT 设备的预期行为。以下是云端指标，来自 Amazon IoT。

消息大小 (aws:message-byte-size)

消息中的字节数。使用此指标指定从设备传输到 Amazon IoT 的每条消息的最大或最小大小（以字节为单位）。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：字节

Example

```
{
  "name": "Max Message Size",
  "metric": "aws:message-byte-size",
```

```
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 1024
    },
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **`statisticalThreshold`** 的示例

```
{

  "name": "Large Message Size",
  "metric": "aws:message-byte-size",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p90"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Message size ML behavior",
  "metric": "aws:message-byte-size",
  "criteria": {
  "consecutiveDatapointsToAlarm": 1,
  "consecutiveDatapointsToClear": 1,
  "mlDetectionConfig": {
    "confidenceLevel": "HIGH"
   }
 },
  "suppressAlerts": true
}
```

如果设备在连续三个五分钟时段内，传输的消息累积大小超过针对该安全配置文件行为报告的所有其他设备累积大小的 90%，则会发出告警。

已发送的消息 (aws:num-messages-sent)

设备在给定时间段内发送的消息数量。

使用此指标指定给定时间段内 Amazon IoT 与各个设备之间可发送的最大或最小消息数量。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：消息

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{

  "name": "Out bound message count",
  "metric": "aws:num-messages-sent",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 50
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
    },
  "suppressAlerts": true
}
```

Example 使用 **`statisticalThreshold`** 的示例

```
{

  "name": "Out bound message rate",
  "metric": "aws:num-messages-sent",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p99"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Messages sent ML behavior",
  "metric": "aws:num-messages-sent",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

已收到的消息 (aws:num-messages-received)

设备在给定时间段内接收的消息数量。

使用此指标指定给定时间段内 Amazon IoT 与各个设备之间可接收的最大或最小消息数量。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：消息

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "In bound message count",
  "metric": "aws:num-messages-received",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 50
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
    },
  "suppressAlerts": true
}
```

Example 使用 **`statisticalThreshold`** 的示例

```
{
  "name": "In bound message rate",
  "metric": "aws:num-messages-received",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p99"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Messages received ML behavior",
  "metric": "aws:num-messages-received",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

授权失败 (aws:num-authorization-failures)

使用此指标指定给定时间段内每个设备允许的最大授权失败次数。如果从设备发往 Amazon IoT 的请求遭到拒绝（例如，设备试图发布到某个主题，但没有足够的权限），则会发生授权失败。

兼容：Rules Detect | ML Detect

单位：失败次数

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "Authorization Failures",
  "metric": "aws:num-authorization-failures",
  "criteria": {
    "comparisonOperator": "less-than",
    "value": {
      "count": 5
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **`statisticalThreshold`** 的示例

```
{
  "name": "Authorization Failures",
  "metric": "aws:num-authorization-failures",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p50"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Authorization failures ML behavior",
  "metric": "aws:num-authorization-failures",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

源 IP (aws:source-ip-address)

设备连接到 Amazon IoT 的 IP 地址。

使用此指标指定一组允许（以前称为列入白名单）或拒绝（以前称为列入黑名单）的无类域间路由 (CIDR)，每个设备必须或不得通过它们连接到 Amazon IoT。

兼容：Rule Detect

运算符：in-cidr-set | not-in-cidr-set

值：CIDR 列表

单位：n/a

Example

```
{
  "name": "Denied source IPs",
  "metric": "aws:source-ip-address",
  "criteria": {
    "comparisonOperator": "not-in-cidr-set",
    "value": {
      "cidrs": [ "12.8.0.0/16", "15.102.16.0/24" ]
    }
  },
  "suppressAlerts": true
}
```

连接尝试 (aws:num-connection-attempts)

设备在给定时间段内尝试建立连接的次数。

使用此指标指定每个设备尝试建立连接的最大或最小次数。成功和失败的尝试都会计算在内。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：连接尝试次数

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "Connection Attempts",
  "metric": "aws:num-connection-attempts",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 5
    },
    "durationSeconds": 600,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **`statisticalThreshold`** 的示例

```
{
  "name": "Connection Attempts",
  "metric": "aws:num-connection-attempts",
  "criteria": {
    "comparisonOperator": "less-than-equals",
```

```
    "statisticalThreshold": {
      "statistic": "p10"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Connection attempts ML behavior",
  "metric": "aws:num-connection-attempts",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": false
}
```

断开连接 (aws:num-disconnects)

设备在给定时间段内与 Amazon IoT 断开连接的次数。

使用此指标来指定设备在给定时间段内与 Amazon IoT 断开连接的最大或最小次数。

兼容：Rules Detect | ML Detect

运算符：less-than | less-than-equals | greater-than | greater-than-equals

值：非负整数

单位：连接断开次数

持续时间：非负整数。有效值为 300、600、900、1800 或 3600 秒。

Example

```
{
  "name": "Disconnections",
  "metric": "aws:num-disconnects",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "value": {
      "count": 5
    },
    "durationSeconds": 600,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 **`statisticalThreshold`** 的示例

```
{
```

```
  "name": "Disconnections",
  "metric": "aws:num-disconnects",
  "criteria": {
    "comparisonOperator": "less-than-equals",
    "statisticalThreshold": {
      "statistic": "p10"
    },
    "durationSeconds": 300,
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1
  },
  "suppressAlerts": true
}
```

Example 使用 ML Detect 的示例

```
{
  "name": "Disconnects ML behavior",
  "metric": "aws:num-disconnects",
  "criteria": {
    "consecutiveDatapointsToAlarm": 1,
    "consecutiveDatapointsToClear": 1,
    "mlDetectionConfig": {
      "confidenceLevel": "HIGH"
    }
  },
  "suppressAlerts": true
}
```

使用维度确定安全配置文件中指标的作用域

维度是您可以定义的属性，用于获取安全配置文件中有关指标和行为的更精确数据。您通过提供用作筛选条件的值或模式来定义作用域。例如，您可以定义一个主题筛选条件维度，该维度仅将指标应用于与特定值（例如“data/bulb/+/activity”）匹配的 MQTT 主题。有关定义要在安全配置文件中使用的维度的信息，请参阅 CreateDimension。

维度值支持 MQTT 通配符。MQTT 通配符可帮助您同时订阅多个主题。有两种不同类型的通配符：单级 (+) 和多级（#)。例如，维度值 `Data/bulb/+/activity` 创建一个和与 + 处于同一级别的所有主题匹配的订阅。维度值还支持 MQTT 客户端 ID 替换变量 ${iot:ClientId}。

TOPIC_FILTER 类型的维度与以下云端衡量指标集兼容：

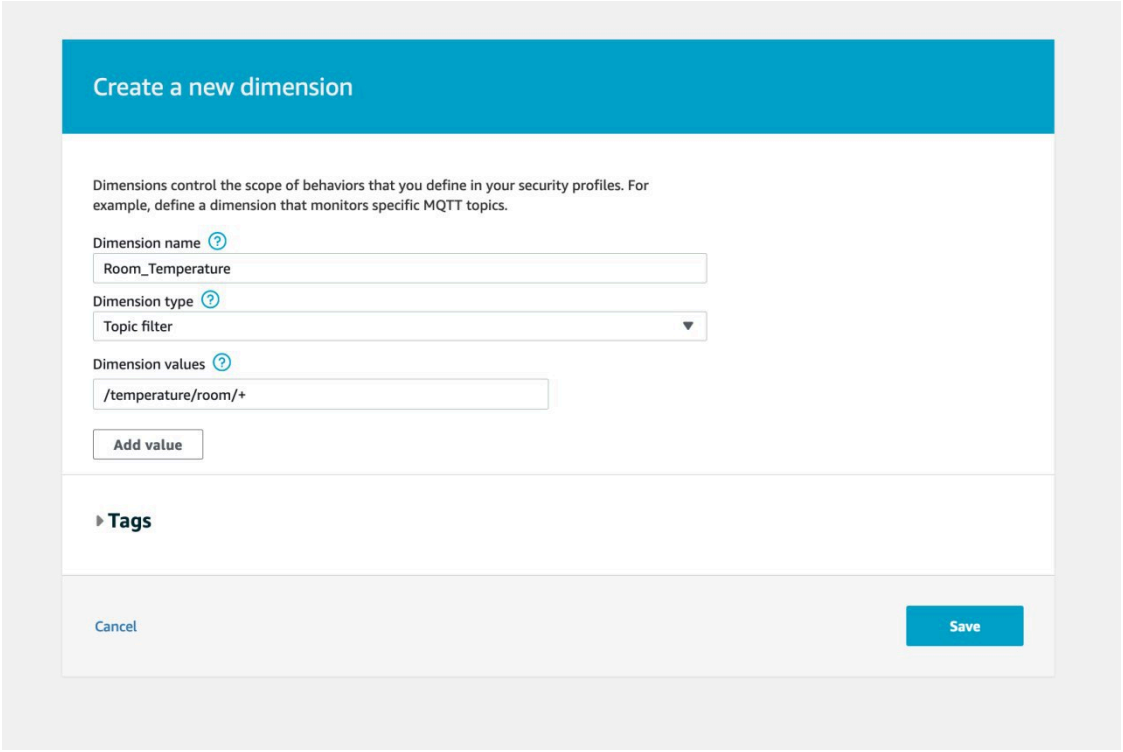
- 发送的消息数量
- 收到的消息数量
- 消息字节大小
- 源 IP 地址（仅适用于 Rules Detect）
- 授权失败次数

如何在控制台中使用维度

创建维度并将其应用于安全配置文件行为

1. 在 Amazon IoT 控制台中的导航窗格中，展开 Defend（防护），展开 Detect（检测），然后选择 Security profiles（安全配置文件）。
2. 在 Security profiles (安全配置文件) 页面上，选择 Create (创建) 以添加新的安全配置文件，或选择 Edit (编辑) 以将维度应用于现有安全配置文件。

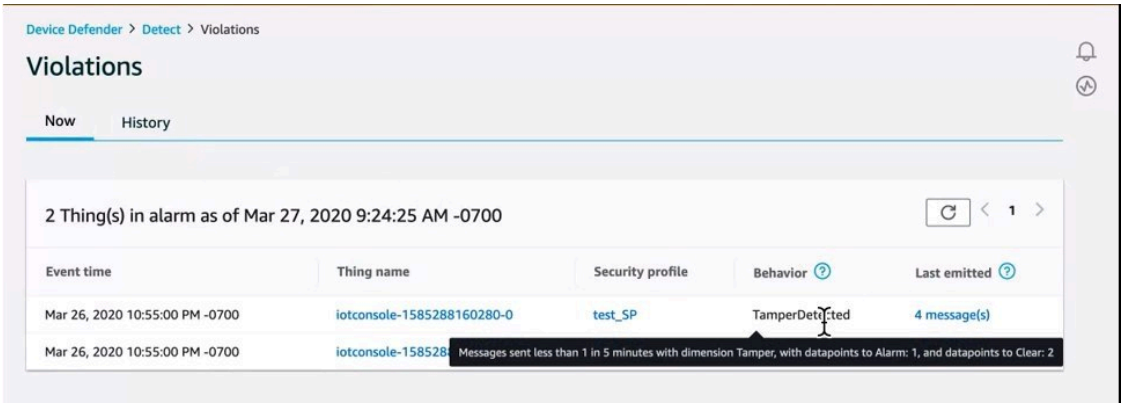
3. 在 Expected behaviors (预期行为) 页面上，在 Metric (指标) 下选择五个云端指标维度支持之一。此时将显示 Dimension (维度) 和 Dimension operator (维度运算符) 框。
4. 对于 Dimension（维度），选择 Add dimension（添加维度）。
5. 在 Create a new dimension (创建新维度) 页面上，输入新维度的详细信息。Dimensions values (维度值) 支持 MQTT 通配符 # 和 + 以及 MQTT 客户端 ID 替换变量 ${iot:Clientid}。



6. 选择保存。
7. 您还可以选择将维度添加到 Additional Metrics to retain（要保留的其它指标）下的指标中。
8. 要完成行为创建，请在其它必填字段中输入信息，然后选择 Next（下一步）。
9. 完成剩余步骤以完成安全配置文件的创建。

查看违规项

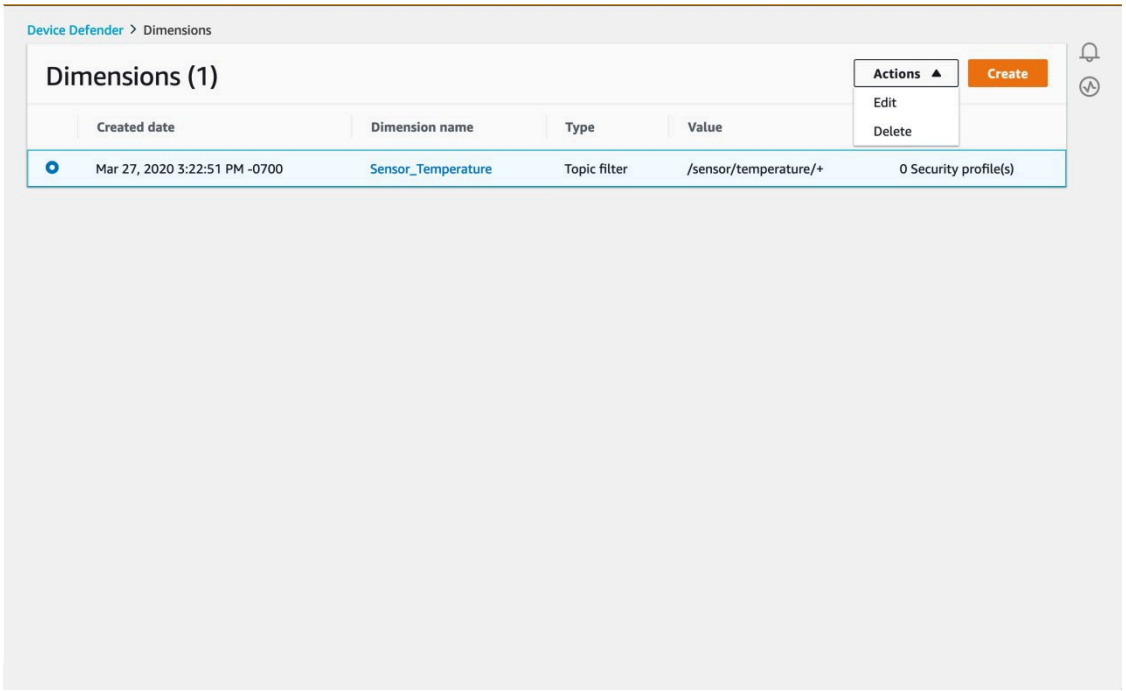
1. 在 Amazon IoT 控制台中的导航窗格中，展开 Defend（下一步），展开 Detect（检测），然后选择 Violations（违规）。



2. 在 Behavior（行为）列中，将鼠标悬停在要查看其违规信息的行为上。

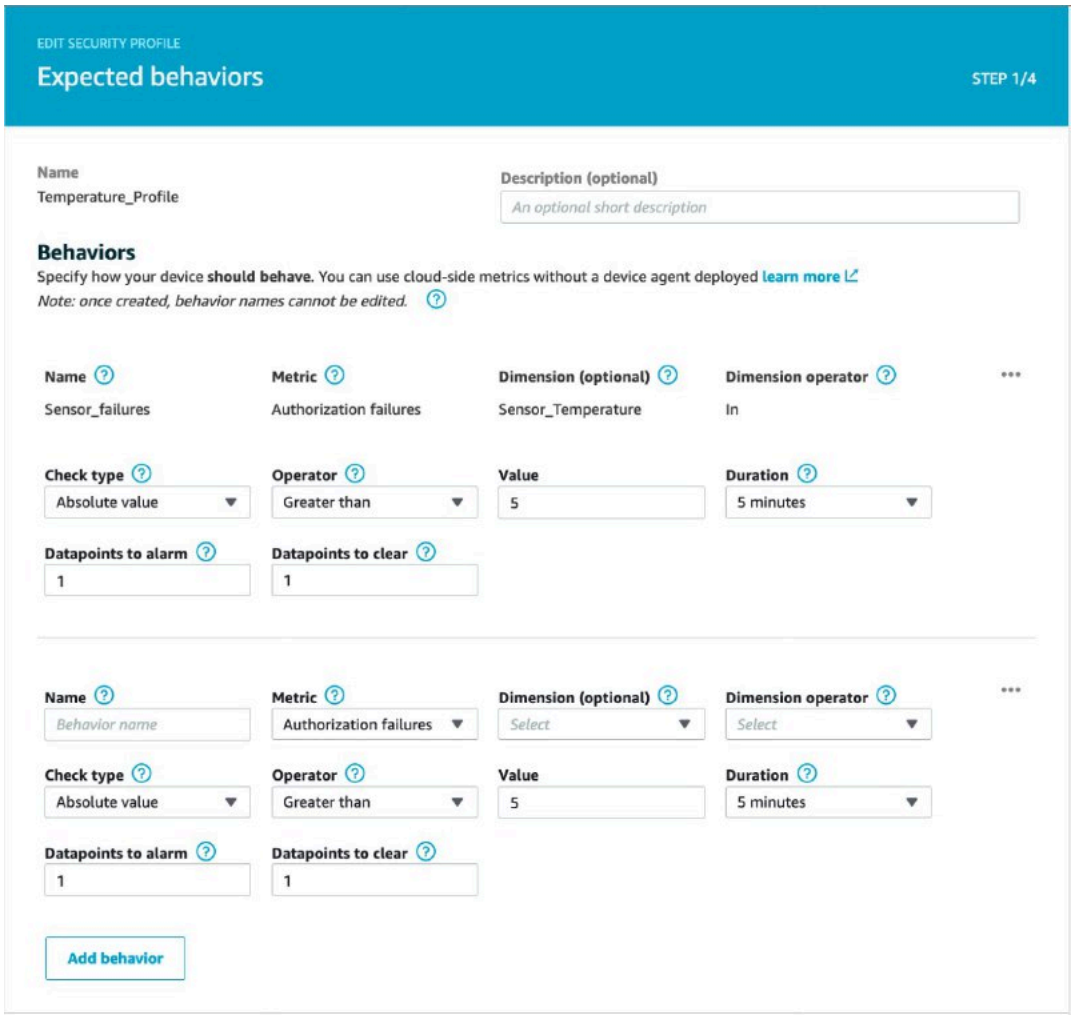
查看和更新您的维度

1. 在 Amazon IoT 控制台中的导航窗格中，展开 Defend（防护），展开 Detect（检测），然后选择 Dimensions（维度）。
2. 选择要编辑的维度。
3. 选择 Actions，然后选择 Edit。



删除维度

1. 在 Amazon IoT 控制台中的导航窗格中，展开 Defend（防护），展开 Detect（检测），然后选择 Dimensions（维度）。
2. 选择要删除的维度。
3. 通过选中 Used in (用于) 列，确认维度未附加到安全配置文件。如果维度附加到安全配置文件，请打开左侧的 Security profiles (安全配置文件) 页面，然后编辑维度附加到的安全配置文件。您删除该维度时，还会删除行为。如果要保留此行为，请选择省略号，然后选择 Copy (复制)。然后，您可以继续删除该行为。如果要删除其它维度，请按照本节中的步骤操作。



4. 选择 Actions，然后选择 Delete。

如何在 Amazon CLI CLI 上使用维度

创建维度并将其应用于安全配置文件行为

1. 首先创建维度，然后再将其附加到安全配置文件。使用 CreateDimension 命令创建维度：

```
aws iot create-dimension \
  --name TopicFilterForAuthMessages \
  --type TOPIC_FILTER \
  --string-values device/+/auth
```

此命令的输出如下所示：

```
{
    "arn": "arn:aws:iot:us-west-2:123456789012:dimension/TopicFilterForAuthMessages",
    "name": "TopicFilterForAuthMessages"
}
```

2. 使用 UpdateSecurityProfile 将维度添加到现有安全配置文件中，或使用 CreateSecurityProfile 将维度添加到新的安全配置文件中。在以下示例中，我们创建了一个新的安全配置文件，用于检查 TopicFilterForAuthMessages 的消息是否在 128 字节以下，并保留发送到非身份验证主题的消息数。

```
aws iot create-security-profile \
  --security-profile-name ProfileForConnectedDevice \
  --security-profile-description "Check to see if messages to
 TopicFilterForAuthMessages are under 128 bytes and retains the number of messages sent
 to non-auth topics." \
  --behaviors "[{\"name\":\"CellularBandwidth\",\"metric\":\"aws:message-byte-size
\",\"criteria\":{\"comparisonOperator\":\"less-than\",\"value\":{\"count\":128},
\"consecutiveDatapointsToAlarm\":1,\"consecutiveDatapointsToClear\":1}},{\"name
\":\"Authorization\",\"metric\":\"aws:num-authorization-failures\",\"criteria\":
{\"comparisonOperator\":\"less-than\",\"value\":{\"count\":10},\"durationSeconds\":300,
\"consecutiveDatapointsToAlarm\":1,\"consecutiveDatapointsToClear\":1}}]" \
  --additional-metrics-to-retain-v2 "[{\"metric\": \"aws:num-authorization-failures\",
\"metricDimension\": {\"dimensionName\": \"TopicFilterForAuthMessages\",\"operator\":
 \"NOT_IN\"}}]"
```

此命令的输出如下所示：

```
{
    "securityProfileArn": "arn:aws:iot:us-west-2:1234564789012:securityprofile/
ProfileForConnectedDevice",
    "securityProfileName": "ProfileForConnectedDevice"
}
```

您也可以从文件加载参数，而不是将其作为命令行参数值完全键入，以节省时间。有关更多信息，请参阅从文件加载 Amazon CLI 参数。下面显示了扩展 JSON 格式的 behavior 参数：

```
[
  {
    "criteria": {
      "comparisonOperator": "less-than",
      "consecutiveDatapointsToAlarm": 1,
      "consecutiveDatapointsToClear": 1,
      "value": {
        "count": 128
      }
    },
    "metric": "aws:message-byte-size",
    "metricDimension": {
      "dimensionName:": "TopicFilterForAuthMessages"
    },
    "name": "CellularBandwidth"
  }
]
```

或者，通过将维度与 ML 结合使用来使用 CreateSecurityProfile，如以下示例所示：

```
aws iot create-security-profile --security-profile-name ProfileForConnectedDeviceML \
   --security-profile-description “Check to see if messages to
 TopicFilterForAuthMessages are abnormal”  \
   --behaviors “[{\“name\”:\“test1\”,\“metric\”:\“aws:message-byte-size\”,
\“metricDimension\”:{\“dimensionName\”: \“TopicFilterForAuthMessages\”,\“operator
\”: \“IN\”},\“criteria\”:{\“mlDetectionConfig\”:{\“confidenceLevel\”:\“HIGH\”},
\“consecutiveDatapointsToAlarm\”:1,\“consecutiveDatapointsToClear\”:1}}]” \
   --region us-west-2
```

查看具有某个维度的安全配置文件

- 使用 ListSecurityProfiles 命令查看具有特定维度的安全配置文件：

```
aws iot list-security-profiles \
  --dimension-name TopicFilterForAuthMessages
```

此命令的输出如下所示：

```
{
    "securityProfileIdentifiers": [
        {
            "name": "ProfileForConnectedDevice",
            "arn": "arn:aws:iot:us-west-2:1234564789012:securityprofile/
ProfileForConnectedDevice"
        }
    ]
}
```

更新维度

- 使用 UpdateDimension 命令更新维度：

```
aws iot update-dimension \
  --name TopicFilterForAuthMessages \
  --string-values device/${iot:ClientId}/auth
```

此命令的输出如下所示：

```
{
    "name": "TopicFilterForAuthMessages",
    "lastModifiedDate": 1585866222.317,
    "stringValues": [
        "device/${iot:ClientId}/auth"
    ],
    "creationDate": 1585854500.474,
    "type": "TOPIC_FILTER",
    "arn": "arn:aws:iot:us-west-2:1234564789012:dimension/TopicFilterForAuthMessages"
}
```

删除维度

1. 要删除维度，请先将其从附加到的任何安全配置文件中分离。使用 ListSecurityProfiles 命令查看具有特定维度的安全配置文件。
2. 要从安全配置文件中删除维度，请使用 UpdateSecurityProfile 命令。输入要保留的所有信息，但排除维度：

```
aws iot update-security-profile \
  --security-profile-name ProfileForConnectedDevice \
  --security-profile-description "Check to see if authorization fails 10 times in 5
 minutes or if cellular bandwidth exceeds 128" \
  --behaviors "[{\"name\":\"metric\":\"aws:message-byte-size\",\"criteria
\":{\"comparisonOperator\":\"less-than\",\"value\":{\"count\":128},
\"consecutiveDatapointsToAlarm\":1,\"consecutiveDatapointsToClear\":1}},{\"name
\":\"Authorization\",\"metric\":\"aws:num-authorization-failures\",\"criteria\":
```

```
{\comparisonOperator\":\"less-than\",\"value\"{\"count\":10},\"durationSeconds\":300,
\"consecutiveDatapointsToAlarm\":1,\"consecutiveDatapointsToClear\":1}}]"
```

此命令的输出如下所示：

```
{
  "behaviors": [
    {
      "metric": "aws:message-byte-size",
      "name": "CellularBandwidth",
      "criteria": {
        "consecutiveDatapointsToClear": 1,
        "comparisonOperator": "less-than",
        "consecutiveDatapointsToAlarm": 1,
        "value": {
          "count": 128
        }
      }
    },
    {
      "metric": "aws:num-authorization-failures",
      "name": "Authorization",
      "criteria": {
        "durationSeconds": 300,
        "comparisonOperator": "less-than",
        "consecutiveDatapointsToClear": 1,
        "consecutiveDatapointsToAlarm": 1,
        "value": {
          "count": 10
        }
      }
    }
  ],
  "securityProfileName": "ProfileForConnectedDevice",
  "lastModifiedDate": 1585936349.12,
  "securityProfileDescription": "Check to see if authorization fails 10 times in 5
 minutes or if cellular bandwidth exceeds 128",
  "version": 2,
  "securityProfileArn": "arn:aws:iot:us-west-2:123456789012:securityprofile/Preo/
ProfileForConnectedDevice",
  "creationDate": 1585846909.127
}
```

3. 分离维度后，使用 DeleteDimension 命令删除维度：

```
aws iot delete-dimension \
  --name TopicFilterForAuthMessages
```

权限

本部分包含有关如何设置管理 Amazon IoT Device Defender Detect 所需的 IAM 角色和策略的信息。有关更多信息，请参阅 IAM 用户指南。

授予 Amazon IoT Device Defender Detect 向 SNS 主题发布告警的权限

如果在 CreateSecurityProfile 中使用 `alertTargets` 参数，则必须为 IAM 角色指定两个策略：一个权限策略和一个信任策略。权限策略授予 Amazon IoT Device Defender 向 SNS 主题发布通知的权限。信任策略授予 Amazon IoT Device Defender 代入所需角色的权限。

权限策略

```
{
  "Version":"2012-10-17",
  "Statement":[
      {
        "Effect":"Allow",
        "Action":[
          "sns:Publish"
        ],
        "Resource":[
          "arn:aws:sns:region:account-id:your-topic-name"
        ]
    }
  ]
}
```

信任策略

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

传递角色策略

您还需要附加至 IAM 用户的 IAM 权限策略，允许该用户传递角色。请参阅向用户授予权限以将角色传递给 Amazon 服务

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

Detect 命令

您可以使用本部分中的 Detect 命令来配置 ML Detect 或规 Rules Detect 安全配置文件，以识别和监控可能表明设备受到危害的异常行为。

DetectMitigation 操作命令

启动和管理 Detect 执行
CancelDetectMitigationActionsTask
DescribeDetectMitigationActionsTask
ListDetectMitigationActionsTasks
StartDetectMitigationActionsTask
ListDetectMitigationActionsExecutions

维度操作命令

启动和管理维度执行
CreateDimension
DescribeDimension
ListDimensions
DeleteDimension
UpdateDimension

CustomMetric 操作命令

启动和管理 CustomMetric 执行
CreateCustomMetric
UpdateCustomMetric
DescribeCustomMetric
ListCustomMetrics
DeleteCustomMetric

安全配置文件操作命令

启动和管理安全配置文件执行
CreateSecurityProfile
AttachSecurityProfile
DetachSecurityProfile
DeleteSecurityProfile
DescribeSecurityProfile
ListTargetsForSecurityProfile
UpdateSecurityProfile

启动和管理安全配置文件执行
ValidateSecurityProfileBehaviors
ListSecurityProfilesForTarget

告警操作命令

管理告警和目标
ListActiveViolations
ListViolationEvents
PutVerificationStateOnViolation

ML Detect 操作命令

列出 ML 模型训练数据
GetBehaviorModelTrainingSummaries

如何使用 Amazon IoT Device Defender detect

1. 可以只将 Amazon IoT Device Defender Detect 与云端指标相结合使用，但如果计划使用设备报告的指标，必须首先在连接 Amazon IoT 的设备或设备网关上部署 Amazon IoT SDK。有关更多信息，请参阅从设备发送指标 (p. 978)。
2. 在定义行为并创建警报之前，请先考虑查看设备生成的指标。Amazon IoT 可以从设备收集指标，因此您可以首先识别账户中的一组设备或所有设备的常见或异常行为。使用 CreateSecurityProfile，但只指定您感兴趣的那些 `additionalMetricsToRetain`。此时不要指定 `behaviors`。

   使用 Amazon IoT 控制台查看设备指标，以了解设备的典型行为。
3. 为安全配置文件创建一组行为。行为包含为账户中的一组设备或所有设备指定正常行为的指标。有关详细信息和示例，请参阅云端指标 (p. 978)和 设备端指标 (p. 964)。创建一组行为后，可以使用 ValidateSecurityProfileBehaviors 进行验证。
4. 使用 CreateSecurityProfile 操作创建包含行为的安全配置文件。当设备违反行为时，可以使用 `alertTargets` 参数向目标（SNS 主题）发送告警。（如果使用 SNS 发送告警，请注意系统将根据您Amazon Web Services 账户的 SNS 主题配额统计数量。）大规模的违规项可能会超过您的 SNS 主题配额。也可以使用 CloudWatch 指标检查违规。有关更多信息，请参阅使用 Amazon IoT 指标 (p. 401)。
5. 使用 AttachSecurityProfile 操作将安全配置文件附加到一组设备（事物组）、账户中所有已注册的事物、所有未注册的事物或所有设备。Amazon IoT Device DefenderDetect 开始检测异常行为，如果检测到任何行为违规，则发送告警。您可能想要将安全配置文件附加到所有未注册的事物，例如，您希望与不在账户事物注册表中的移动设备进行交互。您可以根据需求，为不同设备组定义不同的行为集。

   要将安全配置文件附加到一组设备，必须指定包含这些设备的事物组的 ARN。事物组 ARN 采用以下格式。

   ```
   arn:aws:iot:region:account-id:thinggroup/thing-group-name
   ```

   要将安全配置文件附加到Amazon Web Services 账户中所有已注册的事物（忽略未注册的事物），必须指定以下格式的 ARN：

```
arn:aws:iot:region:account-id:all/registered-things
```

要将安全配置文件附加到所有未注册的事物，必须指定以下格式的 ARN。

```
arn:aws:iot:region:account-id:all/unregistered-things
```

要将安全配置文件附加到所有设备，必须指定以下格式的 ARN。

```
arn:aws:iot:region:account-id:all/things
```

6. 也可以使用 ListActiveViolations 操作跟踪违规项，从而了解对于给定的安全配置文件或目标设备，检测到了哪些违规项。

   使用 ListViolationEvents 操作查看在指定时间段内检测到哪些违规项。您可以按安全配置文件、设备或告警验证状态筛选这些结果。
7. 通过使用 PutVerificationStateOnViolation 操作标记告警的验证状态并提供该验证状态的说明，可以验证、组织和管理报警。
8. 如果设备违反定义的行为过于频繁或太不频繁，那么应该调整行为定义。
9. 要查看您设置的安全配置文件和正在受到监控的设备，请使用 ListSecurityProfiles、ListSecurityProfilesForTarget 和 ListTargetsForSecurityProfile 操作。

   使用 DescribeSecurityProfile 操作获取有关安全配置文件的更多详细信息。
10. 要更新安全配置文件，请使用 UpdateSecurityProfile 操作。使用 DetachSecurityProfile 操作从账户或目标事物组分离安全配置文件。使用 DeleteSecurityProfile 操作完全删除安全配置文件。

缓解操作

您可以使用 Amazon IoT Device Defender 以采取操作缓解 Audit 查找结果或 Detect 告警中发现的问题。

Note

不会对被隐藏的审计查找结果执行缓解操作。有关审计查找结果隐藏的更多信息，请参阅 审计查找结果隐藏 (p. 938)。

审计缓解操作

Amazon IoT Device Defender 为不同的审计检查提供了预定义的操作。您可以为您的Amazon Web Services 账户配置这些操作，然后将其应用于一组查找结果。这些结果可以是：

- 审核中发现的所有结果。此选项可通过 Amazon IoT 控制台和 Amazon CLI 来使用。
- 各个结果的列表。此选项仅在使用 Amazon CLI 的情况下可用。
- 筛选的审核结果集。

下表列出了针对每种情况的审核检查类型和支持的缓解操作：

审核检查到缓解操作映射

审核检查	支持的缓解操作
REVOKED_CA_CERT_CHECK	PUBLISH_FINDING_TO_SNS、UPDATE_CA_CERTIFICATE

审核检查	支持的缓解操作
INTERMEDIATE_CA_REVOKED_FOR_ACTIVE_DEVICE_CERTIFICATES_CHECK	PUBLISH_FINDING_TO_SNS、UPDATE_DEVICE_CERTIFICATE
DEVICE_CERTIFICATE_SHARED_CHECK	PUBLISH_FINDING_TO_SNS、UPDATE_DEVICE_CERTIFICATE
UNAUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK	PUBLISH_FINDING_TO_SNS
AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK	PUBLISH_FINDING_TO_SNS
IOT_POLICY_OVERLY_PERMISSIVE_CHECK	PUBLISH_FINDING_TO_SNS、REPLACE_DEFAULT_POLICY_V
IOT_POLICY_POTENTIAL_MISCONFIGURATION_CHECK	PUBLISH_FINDING_TO_SNS、REPLACE_DEFAULT_POLICY_V
CA_CERT_APPROACHING_EXPIRATION_CHECK	PUBLISH_FINDING_TO_SNS、UPDATE_CA_CERTIFICATE
CONFLICTING_CLIENT_IDS_CHECK	PUBLISH_FINDING_TO_SNS
DEVICE_CERT_APPROACHING_EXPIRATION_CHECK	PUBLISH_FINDING_TO_SNS、UPDATE_DEVICE_CERTIFICATE
REVOKED_DEVICE_CERT_CHECK	PUBLISH_FINDING_TO_SNS、UPDATE_DEVICE_CERTIFICATE
LOGGING_DISABLED_CHECK	PUBLISH_FINDING_TO_SNS、ENABLE_IOT_LOGGING

所有审计检查都支持向 Amazon SNS 发布审计查找结果，以便您可以采取自定义操作来响应通知。每种类型的审核检查可以支持其它缓解操作：

REVOKED_CA_CERT_CHECK

- 在 Amazon IoT 中更改证书的状态以将其标记为非活动状态。

DEVICE_CERTIFICATE_SHARED_CHECK

- 在 Amazon IoT 中更改设备证书的状态以将其标记为非活动状态。
- 将使用该证书的设备添加到事物组。

UNAUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK

- 无额外受支持的操作。

AUTHENTICATED_COGNITO_ROLE_OVERLY_PERMISSIVE_CHECK

- 无额外受支持的操作。

IOT_POLICY_OVERLY_PERMISSIVE_CHECK

- 添加一个空白 Amazon IoT 策略版本以限制权限。

IOT_POLICY_POTENTIAL_MISCONFIGURATION_CHECK

- 识别 Amazon IoT 策略中的潜在错误配置。

CA_CERT_APPROACHING_EXPIRATION_CHECK

- 在 Amazon IoT 中更改证书的状态以将其标记为非活动状态。

CONFLICTING_CLIENT_IDS_CHECK

- 无额外受支持的操作。

DEVICE_CERT_APPROACHING_EXPIRATION_CHECK

- 在 Amazon IoT 中更改设备证书的状态以将其标记为非活动状态。
- 将使用该证书的设备添加到事物组。

REVOKED_DEVICE_CERT_CHECK

- 在 Amazon IoT 中更改设备证书的状态以将其标记为非活动状态。
- 将使用该证书的设备添加到事物组。

LOGGING_DISABLED_CHECK

- 启用日志记录。

Amazon IoT Device Defender 支持对 Audit 查找结果执行以下类型的缓解操作：

操作类型	注意
ADD_THINGS_TO_THING_GROUP	您指定要向其中添加设备的组。您还可以指定是否应覆盖一个或多个动态组中的成员身份（如果它超过了该事物可能所属组的最大数量）。
ENABLE_IOT_LOGGING	您可以指定日志记录级别和具有日志记录权限的角色。您不能指定 DISABLED 日志记录级别。
PUBLISH_FINDING_TO_SNS	您指定结果应发布到的主题。
REPLACE_DEFAULT_POLICY_VERSION	您指定模板名称。使用默认策略或空白策略替换策略版本。目前仅支持值 BLANK_POLICY。
UPDATE_CA_CERTIFICATE	您指定 CA 证书的新状态。目前仅支持值 DEACTIVATE。
UPDATE_DEVICE_CERTIFICATE	您指定设备证书的新状态。目前仅支持值 DEACTIVATE。

通过配置在审核期间发现问题时要执行的标准操作，您可以一致地响应这些问题。使用这些定义的缓解操作还可以帮助您更快地解决问题，并减少人为错误的可能性。

> **Important**
>
> 应用旨在更改证书、将事物添加到新事物组或替换策略的缓解操作可能会对您的设备和应用程序产生影响。例如，设备可能无法连接。请考虑缓解操作的影响，然后再应用这些操作。您可能需要采取其它操作来纠正该问题，然后您的设备和应用程序才可正常工作。例如，您可能需要提供更新的设备证书。缓解操作可帮助您快速限制您的风险，但您仍必须采取纠正措施以解决底层问题。

某些操作（例如重新激活设备证书）只能手动执行。 Amazon IoT Device Defender 不提供机制来自动回滚已应用的缓解操作。

检测缓解操作

Amazon IoT Device Defender 支持对 Detect 告警执行以下类型的缓解操作：

操作类型	注意
ADD_THINGS_TO_THING_GROUP	您指定要向其中添加设备的组。您还可以指定是否应覆盖一个或多个动态组中的成员身份（如果它超过了该事物可能所属组的最大数量）。

如何定义和管理缓解操作

您可以使用 Amazon IoT 控制台或 Amazon CLI 为您的Amazon Web Services 账户定义和管理缓解操作。

创建缓解操作

您定义的每个缓解操作是预定义的操作类型和特定于您账户的参数的组合。

使用 Amazon IoT 控制台创建缓解操作

1. 打开 Amazon IoT 控制台中的“Mitigation actions”（缓解操作）页面。

2. 在 Mitigation Actions（缓解操作）页面上，选择 Create（创建）。
3. 在 Create a new mitigation action（创建新的缓解操作）页面上，在 Action name（操作名称）中输入缓解操作的唯一名称。
4. 在操作类型中，指定您要定义的操作的类型。
5. 在 Permissions（权限）中，选择应用操作的权限所属的 IAM 角色。
6. 每个活动类型请求不同的一组参数。输入操作的参数。例如，如果选择将事物添加到事物组操作类型，请选择目标组，然后选择或清除 Override dynamic groups (覆盖动态组)。
7. 选择 Create（创建）以将缓解操作保存到您的 Amazon 账户。

使用 Amazon CLI 创建缓解操作

- 使用 CreateMitigationAction 命令创建您的缓解操作。当您将操作应用于审核结果时，将使用您为该操作提供的唯一名称。选择一个有意义的名称。

使用 Amazon IoT 控制台查看和修改缓解操作

1. 打开 Amazon IoT 控制台中的“Mitigation actions”（缓解操作）页面。

   Mitigation actions（缓解操作）页面显示为您的 Amazon Web Services 账户定义的所有缓解操作的列表。
2. 选择您要更改的缓解操作的操作名称链接。
3. 选择 Edit（编辑），然后对缓解操作进行更改。由于缓解操作的名称用于标识此操作，因此，您不能更改此名称。
4. 选择 Update（更新）以将对缓解操作的更改保存到您的 Amazon Web Services 账户。

使用 Amazon CLI 列出缓解操作

- 使用 ListMitigationAction 命令可列出您的缓解操作。如果您想更改或删除缓解操作，请记下名称。

使用 Amazon CLI 更新缓解操作

- 使用 UpdateMitigationAction 命令更改您的缓解操作。

使用 Amazon IoT 控制台删除迁移操作

1. 打开 Amazon IoT 控制台中的“Mitigation actions”（缓解操作）页面。

   Mitigation actions（缓解操作）页面显示为您的 Amazon Web Services 账户定义的所有缓解操作。
2. 选择要删除的缓解操作，然后选择 Delete（删除）。
3. 在 Are you sure you want to delete（是否确定要删除）窗口中，选择 Delete（删除）。

使用 Amazon CLI 删除缓解操作

- 使用 UpdateMitigationAction 命令更改您的缓解操作。

使用 Amazon IoT 控制台查看缓解操作详细信息

1. 打开 Amazon IoT 控制台中的“Mitigation actions”（缓解操作）页面。

   Mitigation actions（缓解操作）页面显示为您的 Amazon Web Services 账户定义的所有缓解操作。
2. 选择您要查看的缓解操作的操作名称链接。

使用 Amazon CLI 查看缓解操作详细信息

- 使用 DescribeMitigationAction 命令可查看您的缓解操作的详细信息。

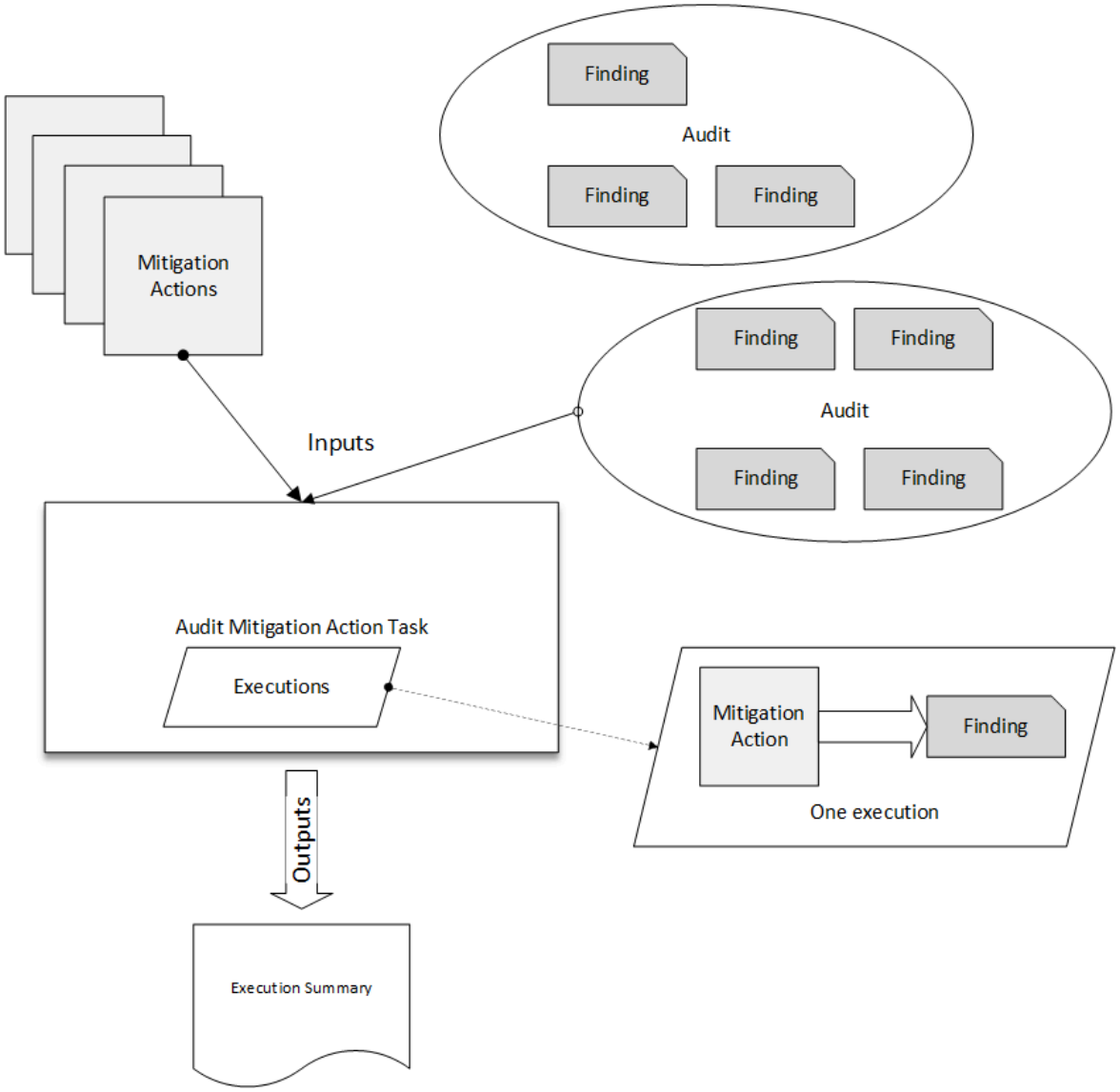
应用缓解操作

在定义了一组缓解操作后，您可以将这些操作应用于审核的结果。当您应用操作时，您就启动了审核缓解操作任务。此任务可能需要一些时间才能完成，具体取决于结果集和您应用于结果集的操作。例如，如果您有一个证书已过期的大型设备池，则可能需要一些时间来停用所有这些证书或将这些设备移到隔离组。其它操作（如启用日志记录）可以快速完成。

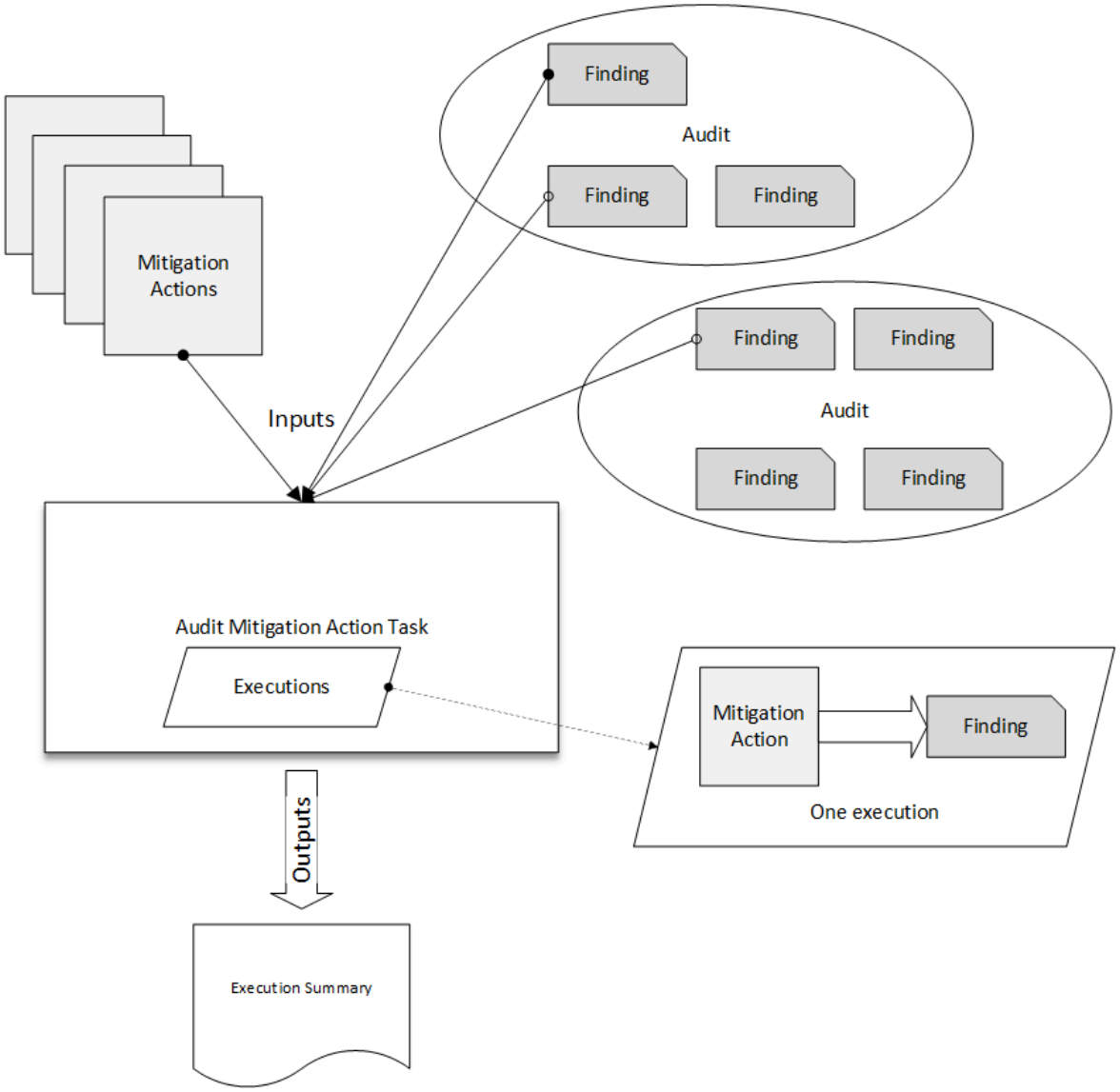
您可以查看操作执行的列表并取消尚未完成的执行。已作为已取消操作执行的一部分执行的操作不会回滚。如果您对一组结果应用多个操作而其中一个操作失败，则会对该结果跳过后续操作（但仍应用于其它结果）。结果的任务状态为 FAILED（失败）。如果在应用于结果时一个或多个操作失败，则 `taskStatus` 设置为失败。操作按指定的顺序应用。

每个操作执行将一组操作应用到目标。该目标可以是结果列表，也可以是来自某个审核的所有结果。

下图显示了您如何可以定义审核缓解任务，该任务从一个审核中获取所有结果并对这些结果应用一组操作。单次执行将一个操作应用于一个结果。审核缓解操作任务输出执行摘要。



下图显示了您如何可以定义审核缓解任务，该任务从一个或多个审核中获取各个结果的列表并对这些结果应用一组操作。单次执行将一个操作应用于一个结果。审核缓解操作任务输出执行摘要。



您可以使用 Amazon IoT 控制台或 Amazon CLI 以应用缓解操作。

使用 Amazon IoT 控制台通过启动操作执行以应用缓解操作

1. 在 Amazon IoT 控制台中打开“Audit results”（审计结果）页面。
2. 选择您要应用操作的审核的名称。
3. 选择 Start mitigation actions（启动缓解操作）。如果所有检查都符合要求，则此按钮不可用。
4. 在 Start a new mitigation action（启动新的缓解操作）中，任务名称默认为审核 ID，但您可以将其更改为更有意义的内容。
5. 对于在审核中具有一个或多个不合规结果的每种类型的检查，您可以选择一个或多个要应用的操作。仅显示对检查类型有效的操作。

   Note

   如果尚未为Amazon Web Services 账户配置操作，则操作列表为空。您可以选择 Create mitigation action（创建缓解操作）链接来创建一个或多个缓解操作。
6. 指定要应用的所有操作后，请选择 Start task（启动任务）。

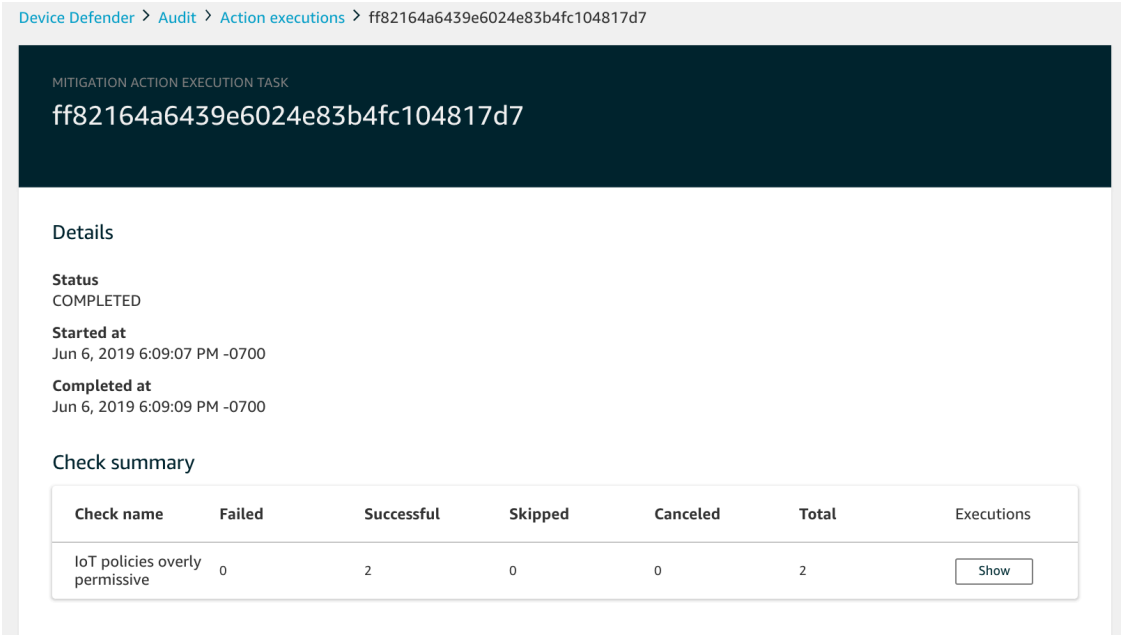
使用 Amazon CLI 通过启动审核缓解操作执行以应用缓解操作

1. 如果您要对审计的所有查找结果应用操作，请使用 ListAuditTasks 命令查找任务 ID。
2. 如果您要仅对选定的查找结果应用操作，请使用 ListAuditFindings 命令获取结果 ID。
3. 使用 ListMitigationActions 命令，并记下要应用的缓解操作的名称。
4. 使用 StartAuditMitigationActionsTask 命令将操作应用到目标。记下任务 ID。您可以使用该 ID 来检查操作执行的状态，查看详细信息，或取消它。

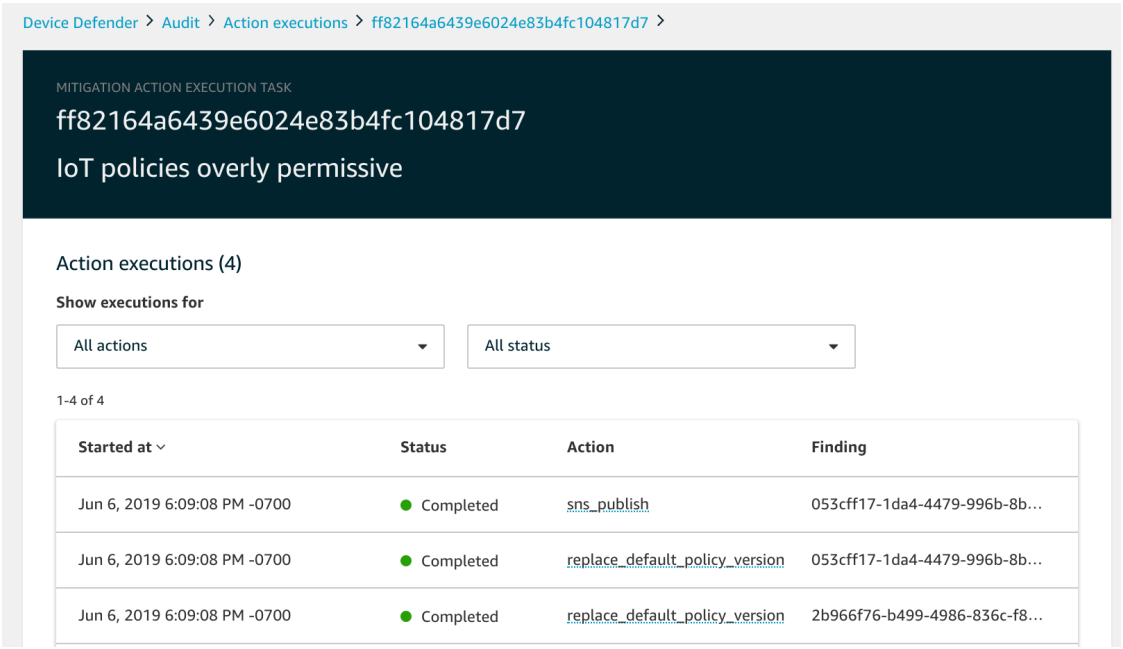
使用 Amazon IoT 控制台查看您的操作执行

1. 打开 Amazon IoT 控制台中的“Action tasks”（操作任务）页面。

   操作任务列表显示每个操作任务的启动时间和当前状态。
2. 选择名称链接以查看任务的详细信息。详细信息包括任务应用的所有操作、其目标及其状态。



   您可以使用 Show executions for (显示的执行面向) 筛选条件来重点关注某些类型的操作或操作状态。
3. 要查看任务的详细信息，请在 Executions (执行) 中选择 Show (显示)。



使用 Amazon CLI 列出已启动的任务

1. 使用 ListAuditMitigationActionsTasks 查看您的审计缓解操作任务。您可以提供筛选条件来缩小结果范围。如果要查看任务的详细信息，请记下任务 ID。
2. 使用 ListAuditMitigationActionsExecutions 查看特定审计缓解操作任务的执行详细信息。
3. 使用 DescribeAuditMitigationActionsTask 查看有关任务的详细信息，例如启动时指定的参数。

使用 Amazon CLI 取消正在运行的审核缓解操作任务

1. 使用 ListAuditMitigationActionsTasks 命令找到您要取消其执行的任务的 ID。您可以提供筛选条件来缩小结果范围。
2. 使用 ListDetectMitigationActionsExecutions 命令通过任务 ID 取消您的审计缓解操作任务。您无法取消已完成的任务。当您取消任务时，不会应用剩余的操作，但不会回滚已应用的缓解操作。

权限

对于您定义的每个缓解操作，您必须提供用于应用该操作的角色。

缓解操作的权限

操作类型	权限策略模板	

UPDATE_DEVICE_CERTIFICATE	<pre>{
	"Version":"2012-10-17",
	"Statement":[
	{
	"Effect":"Allow",
	"Action":[</pre>	

操作类型	权限策略模板	

	<pre>"iot:UpdateCertificate"
	            ],
	            "Resource":[
	                "*"
	            ]
	        }
	    ]
	}</pre>	

UPDATE_CA_CERTIFICATE	<pre>{
	    "Version":"2012-10-17",
	    "Statement":[
	        {
	
	"Effect":"Allow",
	            "Action":[
	
	"iot:UpdateCACertificate"
	            ],
	            "Resource":[
	                "*"
	            ]
	        }
	    ]
	}</pre>	

ADD_THINGS_TO_THING_GROUP	<pre>{
	    "Version":"2012-10-17",
	    "Statement":[
	        {
	
	"Effect":"Allow",
	            "Action":[
	
	"iot:ListPrincipalThings",
	
	"iot:AddThingToThingGroup"
	            ],
	            "Resource":[
	                "*"
	            ]
	        }
	    ]
	}</pre>	

操作类型	权限策略模板	

REPLACE_DEFAULT_POLICY_VERSION	{
	"Version":"2012-10-17",
	"Statement":[
	{
	"Effect":"Allow",
	"Action":[
	"iot:CreatePolicyVersion"
	],
	"Resource":[
	"*"
	]
	}
	]
	}	

ENABLE_IOT_LOGGING	{
	"Version":"2012-10-17",
	"Statement":[
	{
	"Effect":"Allow",
	"Action":[
	"iot:SetV2LoggingOptions"
	],
	"Resource":[
	"*"
	]
	},
	{
	"Effect":"Allow",
	"Action":[
	"iam:PassRole"
	],
	"Resource":[
	"*<IAM role ARN used for setting up logging>*"
	]
	}
	]
	}	

操作类型	权限策略模板	

PUBLISH_FINDING_TO_SNS	{
	"Version":"2012-10-17",
	"Statement":[
	{
	"Effect":"Allow",
	"Action":[
	"sns:Publish"
	],
	"Resource":[
	"*<The SNS topic to which the finding is published>*"
	]
	}
	]
	}	

对于所有缓解操作类型，请使用以下信任策略模板：


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:iot:*:111122223333::*"
        },
        "StringEquals": {
          "aws:SourceAccount": "111122223333:"
        }
      }
    }
  ]
}
```


缓解操作命令

您可以使用这些缓解操作命令为您的Amazon Web Services 账户定义一组操作，以后可以将这些操作应用于一组或多组审计查找结果。有三种命令类别：

- 用于定义和管理操作的命令。
- 用于启动和管理将这些操作应用于 Audit 查找结果的命令。
- 用于启动和管理将这些操作应用 Detect 告警的命令。

缓解操作命令

定义和管理操作	启动和管理 Audit 执行	启动和管理 Detect 执行
CreateMitigationAction	CancelAuditMitigationActionsTask	CancelDetectMitigationActionsTask
DeleteMitigationAction	DescribeAuditMitigationActionsTask	DescribeDetectMitigationActionsTask
DescribeMitigationAction	ListAuditMitigationActionsTasks	ListDetectMitigationActionsTasks
ListMitigationActions	StartAuditMitigationActionsTask	StartDetectMitigationActionsTask
UpdateMitigationAction	ListAuditMitigationActionsExecutions	ListDetectMitigationActionsExecutions

将 Amazon IoT Device Defender 与其它 Amazon 产品结合使用

在运行 Amazon IoT Greengrass 的设备上使用 Amazon IoT Device Defender

Amazon IoT Greengrass 提供了与 Amazon IoT Device Defender 的预构建集成以持续监控设备行为。

- Device Defender 与 Amazon IoT Greengrass V1 集成
- Device Defender 与 Amazon IoT Greengrass V2 集成

将 Amazon IoT Device Defender 与 FreeRTOS 和嵌入式设备搭配使用

要在 FreeRTOS 设备上使用 Amazon IoT Device Defender，则您的设备必须安装了 FreeRTOS Embedded C SDK 或 Amazon IoT Device Defender 库。FreeRTOS Embedded C SDK 包含 Amazon IoT Device Defender 库。有关如何将 Amazon IoT Device Defender 与您的 FreeRTOS 设备集成的信息，请参阅以下演示：

- Amazon IoT Device Defender适用于 FreeRTOS 标准指标和自定义指标演示
- 使用 MQTT 代理将指标提交到 Amazon IoT Device Defender
- 使用 MQTT 核心库将指标提交到 Amazon IoT Device Defender

要在没有 FreeRTOS 的嵌入式设备上使用 Amazon IoT Device Defender，您的设备就必须具有 Amazon IoT 嵌入式 C 开发工具包或 Amazon IoT Device Defender 库。Amazon IoT Embedded C SDK 包括 Amazon IoT Device Defender 库。有关如何集成 Amazon IoT Device Defender 和嵌入式设备的信息，请参阅以下演示，Amazon IoT Device Defender 了解 Amazon IoT 嵌入式SDK标准和自定义指标演示。

配合使用 Amazon IoT Device Defender 和 Amazon IoT Device Management

您可以使用 Amazon IoT Device Management 实例集索引为索引、搜索和聚合您的 Amazon IoT Device Defender 检测违规情况。在实例集索引中建立 Device Defender 违规数据之后，您可以访问和查询来自

Fleet Hub 应用程序的 Device Defender 违规数据，根据违规数据创建实例集告警以监控设备实例集中的异常情况，以及在 Fleet Hub 仪表板中查看实例集告警。

Note

支持索引 Amazon IoT Device Defender 违规数据的实例集索引功能已在 Amazon IoT Device Management 的预览版本中，可能会发生更改。

- 管理实例集索引
- 查询语法
- 管理 Fleet Hub 应用程序的实例集索引
- 入门

与 Amazon Security Hub 集成

Amazon Security Hub 提供了您在 Amazon 中的安全状态的全面视图，可帮助您检查环境是否符合安全行业标准和最佳实践。Security Hub 从跨 Amazon Web Services 账户、服务和受支持的第三方产品中收集安全数据。您可以使用 Security Hub 分析安全趋势并确定优先级最高的安全问题。

Amazon IoT Device Defender 与 Security Hub 的集成使您能够从 Amazon IoT Device Defender 中的 Security Hub 接收调查结果。Security Hub 会在其对您的安全状况分析中包含这些结果。

目录

- 启用和配置集成 (p. 1008)
- Amazon IoT Device Defender 将结果发送到 Security Hub 的方式 (p. 1008)
    - Amazon IoT Device Defender 发送的结果类型 (p. 1009)
    - 发送结果的延迟 (p. 1009)
    - 当 Security Hub 不可用时重试 (p. 1009)
    - 更新 Security Hub 中的现有结果 (p. 1009)
- 来自 Amazon IoT Device Defender 的典型结果 (p. 1010)
- 停止 Amazon IoT Device Defender 向 Security Hub 发送调查结果 (p. 1013)

启用和配置集成

将 Amazon IoT Device Defender 与 Security Hub 集成之前，您必须先启用 Security Hub。有关如何启用 Security Hub 的信息，请参阅 Amazon Security Hub 用户指南中的设置 Security Hub。

同时启用 Amazon IoT Device Defender 和 Security Hub 后，打开 Security Hub 控制台中的“Integrations”（集成）页面，然后选择 Accept findings for Audit, Detect, or both（接受审计或检测的调查结果或同时接受两者的调查结果）。Amazon IoT Device Defender 将开始向 Security Hub 发送调查结果。

Amazon IoT Device Defender 将结果发送到 Security Hub 的方式

在 Security Hub 中，安全问题按调查结果进行跟踪。一些调查结果来自其他 Amazon 服务或第三方产品检测到的问题。

Security Hub 提供了管理来自所有这些来源的结果的工具。您可以查看和筛选结果列表，并查看结果的详细信息。有关更多信息，请参阅 Amazon Security Hub 用户指南中的查看结果。您还可以跟踪结果的调查状态。有关更多信息，请参阅 Amazon Security Hub 用户指南中对结果采取行动。

Security Hub 中的所有调查结果都使用名为 Amazon 安全检测结果格式 (ASFF) 的标准 JSON 格式。ASFF 包括有关问题根源、受影响资源以及结果当前状态的详细信息。有关 ASFF 的更多信息，请参阅 Amazon Security Hub 用户指南中的 Amazon 安全检测结果格式 (ASFF)。

Amazon IoT Device Defender 是一项 Amazon 服务，可将检测结果发送到 Security Hub。

Amazon IoT Device Defender 发送的结果类型

启用 Security Hub 集成后，Amazon IoT Device Defender 审计会将其生成的检测结果（称为检查摘要）发送到 Security Hub。检查摘要是关于特定审计检查类型和特定审计任务的一般信息。有关更多信息，请参阅审计检查。

Amazon IoT Device Defender 审计会向 Security Hub 发送有关每项审计任务中的“审计检查摘要”和“审计结果”的检测结果更新。如果在“审计检查”中找到的所有资源均符合要求，或者审计任务被取消，审计会将 Security Hub 中的检查摘要更新为“已存档”记录状态。如果某项资源在审计检查中被报告为不合规，但在上一次审计任务中被报告为合规，则审计会将其更改为合规，并将 Security Hub 中的调查结果更新为“已存档”记录状态。

Amazon IoT Device Defender 检测会将违规调查结果发送到 Security Hub。这些违规调查结果包括机器学习 (ML)、统计和静态行为。

为了将调查结果发送到 Security Hub，Amazon IoT Device Defender 使用 Amazon 安全检测结果格式 (ASFF)。在 ASFF 中，`Types` 字段提供结果类型。来自 Amazon IoT Device Defender 的结果可能具有 `Types` 的以下值。

不寻常的行为

冲突的 MQTT 客户端 ID 和设备证书共享检查的调查结果类型，以及检测的调查结果类型。

软件和配置检查/漏洞

所有其他审计检查的调查结果类型。

发送结果的延迟

Amazon IoT Device Defender 审计创建新调查结果时，将在审计任务完成后立即将结果发送到 Security Hub。延迟取决于审计任务中生成的调查结果的数量。Security Hub 通常会在一小时内收到调查结果。

Amazon IoT Device Defender 检测将近乎实时发送违规调查结果。在违规进入或退出警报（意味着已创建或删除警报）后，会立即创建或存档相应的 Security Hub 调查结果。

当 Security Hub 不可用时重试

如果 Security Hub 不可用，Amazon IoT Device Defender 审计和 Amazon IoT Device Defender 检测会重试发送调查结果，直到收到这些结果。

更新 Security Hub 中的现有结果

将 Amazon IoT Device Defender 审计调查结果发送到 Security Hub 后，您可以通过选中的资源标识符和审计检查类型对其进行识别。如果在后续审计任务中针对同一资源和审计检查生成了新的审计调查结果，Amazon IoT Device Defender 审计会向 Security Hub 发送更新，以反映调查结果活动的其他观察结果。如果在后续审计任务中没有针对同一资源和审计检查生成其他审计调查结果，则资源将更改为符合审计检查。 Amazon IoT Device Defender然后，审计会将调查结果存档到 Security Hub 中。

Amazon IoT Device Defender 审计还会更新 Security Hub 中的检查摘要。如果在审计检查中发现不合规资源或检查失败，Security Hub 调查结果的状态将变为活动。否则，Amazon IoT Device Defender 审计会将调查结果存档到 Security Hub 中。

当发生冲突（例如，在警报中）时，Amazon IoT Device Defender 检测会创建一个 Security Hub 调查结果。仅当满足以下条件之一时，才会更新调查结果：

- 由于调查结果将很快在 Security Hub 中过期，因此 Amazon IoT Device Defender 会发送更新以使该调查结果保持最新状态。调查结果将在最新更新后 90 天或创建日期后 90 天（如果未发生更新）被删除。有关更多信息，请参阅 Amazon Security Hub 用户指南中的 Security Hub 配额。
- 相应的违规将解除警报，因此 Amazon IoT Device Defender 会将其调查结果状态更新为“已存档”。

来自 Amazon IoT Device Defender 的典型结果

Amazon IoT Device Defender 使用 Amazon 安全检测结果格式 (ASFF) 将调查结果发送到 Security Hub。

以下示例显示了 Security Hub 中审计调查结果的典型调查结果。`ProductFields` 中的 `ReportType` 是 `AuditFinding`。


```
  {
  "SchemaVersion": "2018-10-08",
  "Id": "336757784525/IOT_POLICY/policyexample/1/IOT_POLICY_OVERLY_PERMISSIVE_CHECK/
ALLOWS_BROAD_ACCESS_TO_IOT_DATA_PLANE_ACTIONS",
  "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/iot-device-defender-audit",
  "ProductName": "IoT Device Defender - Audit",
  "CompanyName": "AWS",
  "Region": "us-west-2",
  "GeneratorId": "1928b87ab338ee2f541f6fab8c41c4f5",
  "AwsAccountId": "123456789012",
  "Types": [
    "Software and Configuration Check/Vulnerabilities"
  ],
  "CreatedAt": "2022-11-06T22:11:40.941Z",
  "UpdatedAt": "2022-11-06T22:11:40.941Z",
  "Severity": {
    "Label": "CRITICAL",
    "Normalized": 90
  },
  "Title": "IOT_POLICY_OVERLY_PERMISSIVE_CHECK:
 ALLOWS_BROAD_ACCESS_TO_IOT_DATA_PLANE_ACTIONS",
  "Description": "IOT_POLICY policyexample:1 is reported as non-compliant for
 IOT_POLICY_OVERLY_PERMISSIVE_CHECK by Audit task 9f71b6e90cfb57d4ac671be3a4898e6a.
 The non-compliant reason is Policy allows broad access to IoT data plane actions:
 [iot:Connect].",
  "SourceUrl": "https://us-west-2.console.aws.amazon.com/iot/home?region=us-west-2#/policy/
policyexample",
  "ProductFields": {
    "CheckName": "IOT_POLICY_OVERLY_PERMISSIVE_CHECK",
    "TaskId": "9f71b6e90cfb57d4ac671be3a4898e6a",
    "TaskType": "ON_DEMAND_AUDIT_TASK",
    "PolicyName": "policyexample",
    "IsSuppressed": "false",
    "ReasonForNonComplianceCode": "ALLOWS_BROAD_ACCESS_TO_IOT_DATA_PLANE_ACTIONS",
    "ResourceType": "IOT_POLICY",
    "FindingId": "1928b87ab338ee2f541f6fab8c41c4f5",
    "PolicyVersionId": "1",
    "ReportType": "AuditFinding",
    "TaskStartTime": "1667772700554",
    "aws/securityhub/FindingId": "arn:aws:securityhub:us-west-2::product/aws/iot-device-
defender-audit/336757784525/IOT_POLICY/policyexample/1/IOT_POLICY_OVERLY_PERMISSIVE_CHECK/
ALLOWS_BROAD_ACCESS_TO_IOT_DATA_PLANE_ACTIONS",
    "aws/securityhub/ProductName": "IoT Device Defender - Audit",
    "aws/securityhub/CompanyName": "AWS"
  },
  "Resources": [
    {
      "Type": "AwsIotPolicy",
      "Id": "policyexample",
      "Partition": "aws",
      "Region": "us-west-2",
      "Details": {
        "Other": {
          "PolicyVersionId": "1"
        }
      }
```

```
    }
  ],
  "WorkflowState": "NEW",
  "Workflow": {
    "Status": "NEW"
  },
  "RecordState": "ACTIVE",
  "FindingProviderFields": {
    "Severity": {
      "Label": "CRITICAL"
    },
    "Types": [
      "Software and Configuration Check/Vulnerabilities"
    ]
  }
}
```

以下示例显示了 Security Hub 中审计检查摘要的典型调查结果。ProductFields 中的 ReportType 是 CheckSummary。

```
  {
  "SchemaVersion": "2018-10-08",
  "Id": "615243839755/SCHEDULED_AUDIT_TASK/daily_audit_schedule_checks/
DEVICE_CERTIFICATE_KEY_QUALITY_CHECK",
  "ProductArn": "arn:aws:securityhub:us-east-1::product/aws/iot-device-defender-audit",
  "ProductName": "IoT Device Defender - Audit",
  "CompanyName": "AWS",
  "Region": "us-east-1",
  "GeneratorId": "f3021945485adf92487c273558fcaa51",
  "AwsAccountId": "123456789012",
  "Types": [
    "Software and Configuration Check/Vulnerabilities/CVE"
  ],
  "CreatedAt": "2022-10-18T14:20:13.933Z",
  "UpdatedAt": "2022-10-18T14:20:13.933Z",
  "Severity": {
    "Label": "CRITICAL",
    "Normalized": 90
  },
  "Title": "DEVICE_CERTIFICATE_KEY_QUALITY_CHECK Summary: Completed with 2 non-compliant
 resources",
  "Description": "Task f3021945485adf92487c273558fcaa51 of weekly scheduled Audit
 daily_audit_schedule_checks completes. 2 non-cimpliant resources are found for
 DEVICE_CERTIFICATE_KEY_QUALITY_CHECK out of 1000 resources in the account. The percentage
 of non-compliant resources is 0.2%.",
  "SourceUrl": "https://us-east-1.console.aws.amazon.com/iot/home?region=us-east-1#/dd/
audit/results/f3021945485adf92487c273558fcaa51/DEVICE_CERTIFICATE_KEY_QUALITY_CHECK",
  "ProductFields": {
    "TaskId": "f3021945485adf92487c273558fcaa51",
    "TaskType": "SCHEDULED_AUDIT_TASK",
    "ScheduledAuditName": "daily_audit_schedule_checks",
    "CheckName": "DEVICE_CERTIFICATE_KEY_QUALITY_CHECK",
    "ReportType": "CheckSummary",
    "CheckRunStatus": "COMPLETED_NON_COMPLIANT",
    "NonComopliantResourcesCount": "2",
    "SuppressedNonCompliantResourcesCount": "1",
    "TotalResourcesCount": "1000",
    "aws/securityhub/FindingId": "arn:aws:securityhub:us-east-1::product/aws/
iot-device-defender-audit/615243839755/SCHEDULED/daily_audit_schedule_checks/
DEVICE_CERTIFICATE_KEY_QUALITY_CHECK",
    "aws/securityhub/ProductName": "IoT Device Defender - Audit",
    "aws/securityhub/CompanyName": "AWS"
```

```
  },
  "Resources": [
    {
      "Type": "AwsIotAuditTask",
      "Id": "f3021945485adf92487c273558fcaa51",
      "Region": "us-east-1"
    }
  ],
  "WorkflowState": "NEW",
  "Workflow": {
    "Status": "NEW"
  },
  "RecordState": "ACTIVE",
  "FindingProviderFields": {
    "Severity": {
      "Label": "CRITICAL"
    },
    "Types": [
      "Software and Configuration Check/Vulnerabilities/CVE"
    ]
  }
}
```

以下示例显示了 Security Hub 中 Amazon IoT Device Defender 检测违规的典型调查结果。

```
  {
  "SchemaVersion": "2018-10-08",
  "Id": "e92a782593c6f5b1fc7cb6a443dc1a12",
  "ProductArn": "arn:aws:securityhub:us-east-1::product/aws/iot-device-defender-detect",
  "ProductName": "IoT Device Defender - Detect",
  "CompanyName": "AWS",
  "Region": "us-east-1",
  "GeneratorId": "arn:aws:iot:us-east-1:123456789012:securityprofile/MySecurityProfile",
  "AwsAccountId": "123456789012",
  "Types": [
    "Unusual Behaviors"
  ],
  "CreatedAt": "2022-11-09T22:45:00Z",
  "UpdatedAt": "2022-11-09T22:45:00Z",
  "Severity": {
    "Label": "MEDIUM",
    "Normalized": 40
  },
  "Title": "Registered thing MyThing is in alarm for STATIC behavior MyBehavior.",
  "Description": "Registered thing MyThing violates STATIC behavior MyBehavior of security
 profile MySecurityProfile. Violation was triggered because the device did not conform to
 aws:num-disconnects less-than 1.",
  "SourceUrl": "https://us-east-1.console.aws.amazon.com/iot/home?region=us-east-1#/dd/
securityProfile/MySecurityProfile?tab=violations",
  "ProductFields": {
    "ComparisonOperator": "less-than",
    "BehaviorName": "MyBehavior",
    "ViolationId": "e92a782593c6f5b1fc7cb6a443dc1a12",
    "ViolationStartTime": "1668033900000",
    "SuppressAlerts": "false",
    "ConsecutiveDatapointsToAlarm": "1",
    "ConsecutiveDatapointsToClear": "1",
    "DurationSeconds": "300",
    "Count": "1",
    "MetricName": "aws:num-disconnects",
    "BehaviorCriteriaType": "STATIC",
    "ThingName": "MyThing",
    "SecurityProfileName": "MySecurityProfile",
```

```
    "aws/securityhub/FindingId": "arn:aws:securityhub:us-east-1::product/aws/iot-device-
defender-detect/e92a782593c6f5b1fc7cb6a443dc1a12",
    "aws/securityhub/ProductName": "IoT Device Defender - Detect",
    "aws/securityhub/CompanyName": "AWS"
  },
  "Resources": [
    {
      "Type": "AwsIotRegisteredThing",
      "Id": "MyThing",
      "Region": "us-east-1",
      "Details": {
        "Other": {
          "SourceUrl": "https://us-east-1.console.aws.amazon.com/iot/home?region=us-
east-1#/thing/MyThing?tab=violations",
          "IsRegisteredThing": "true",
          "ThingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyThing"
        }
      }
    }
  ],
  "WorkflowState": "NEW",
  "Workflow": {
    "Status": "NEW"
  },
  "RecordState": "ACTIVE",
  "FindingProviderFields": {
    "Severity": {
      "Label": "MEDIUM"
    },
    "Types": [
      "Unusual Behaviors"
    ]
  }
}
```

停止 Amazon IoT Device Defender 向 Security Hub 发送调查结果

要停止向 Security Hub 发送结果，您可以使用 Security Hub 控制台或 API。

有关更多信息，请参阅 Amazon Security Hub 用户指南中的禁用和启用来自集成的结果流（控制台）或禁用来自集成的结果流（Security Hub API、Amazon CLI）。

跨服务混淆代理问题防范

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

混淆代理安全问题会影响 Amazon IoT Device Defender 从您处访问的三种资源：运行审计、发送安全配置文件违规事件的 SNS 通知以及运行缓解操作。对于其中每个操作，`aws:SourceArn` 的值必须如下所示：

- 对于在 UpdateAccountAuditConfiguration API 中传递的资源（RoleArn 和 notificationTarget RoleArn 属性），应将 `aws:SourceArn` 作为 `arn:`*`arnPartition`*`:iot:`*`region`*`:`*`accountId`*`:` 来缩小资源策略的范围。
- 对于在 CreateMitigationAction API 中传递的资源（RoleArn 属性），应将 `aws:SourceArn` 作为 `arn:`*`arnPartition`*`:iot:`*`region`*`:`*`accountId`*`:mitigationaction/`*`mitigationActionName` 来缩小资源策略的范围。

- 对于在 CreateSecurityProfile API 中传递的资源（alertTargets 属性），应将 `aws:SourceArn` 作为 `arn:arnPartition:iot:region:accountId:securityprofile/securityprofileName` 来缩小资源策略的范围。

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

以下示例演示如何使用 Amazon IoT Device Defender 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件上下文键来防范混淆代理问题。

```
 {
"Version": "2012-10-17",
"Statement": {
  "Sid": "ConfusedDeputyPreventionExamplePolicy",
  "Effect": "Allow",
  "Principal": {
    "Service": "iot.amazonaws.com"
  },
  "Action": "sts:AssumeRole",
  "Condition": {
    "ArnLike": {
      "aws:SourceArn": "arn:aws:iot:*:123456789012::*"
    },
    "StringEquals": {
      "aws:SourceAccount": "123456789012:"
    }
  }
}
}
```

设备代理的安全最佳实践

最低权限

应仅为代理进程授予履行其职责所需的最低权限。

基本机制

- 代理应该以非根用户身份运行。
- 代理应该以专用用户身份在其自己的组中运行。
- 应该为用户/组授予收集和传输指标所需资源的只读权限。
- 示例：针对代理示例的 /proc /sys 只读权限。
- 有关如何设置进程从而以较低权限运行的示例，请参阅 Python 示例代理附带的设置说明。

有一些知名的 Linux 机制可帮助您进一步限制或隔离代理进程：

高级机制

- CGroups
- SELinux
- Chroot
- Linux Namespaces

运营弹性

代理进程必须能够灵活应对意外的运营错误和异常，不能崩溃或永久退出。代码需要从容地处理异常，并且作为一项预防措施，它必须配置为在发生意外终止（例如，因为系统重启或未捕获的异常）时自动重启。

最少依赖项

代理在实施中必须使用尽可能少的依赖项（即第三方库）。如果出于任务复杂性（例如，传输层安全）这类正当理由使用了库，只能使用维护良好的依赖项，并建立机制以确保依赖项保持更新。如果添加的依赖项包含代理不使用的功能且默认处于活动状态（例如，打开的端口、域套接字），请在代码中禁用这些功能或通过库的配置文件禁用这些功能。

进程隔离

代理进程只能包含执行设备指标收集和传输所必需的功能。不得利用其它系统进程作为容器，或实施用于使用案例范围之外的功能。此外，代理进程必须避免创建入站通信渠道，如域套接字和网络服务端口，这种渠道将允许本地或远程进程干扰其操作并影响其完整性和隔离。

隐匿性

代理进程不得使用表明其用途和安全价值的关键字（如安全、监控或审核）命名。首选通用代码名称或每个设备唯一的随机进程名称。命名代理的二进制文件所在的目录以及进程参数的任何名称和值时，必须遵循相同的原则。

最少信息共享

部署到设备的任何代理项目均不得包含敏感信息，如特权凭证、调试代码和死码，或揭露有关代理收集类指标的服务器端处理详情或有关后端系统的其它相关详情的内联注释或文档文件。

传输层安全

要为数据传输建立 TLS 安全通道，代理进程必须在应用程序级别强制执行所有客户端验证（如果没有默认启用），例如证书链和域名验证。此外，代理必须使用包含受信任机构、不包含属于遭破坏证书发布者的证书的根证书存储区。

安全部署

任何代理部署机制（例如代码推送或同步）以及包含其二进制文件、源代码和任何配置文件（包括受信任的根证书）的存储库，都必须实施访问控制以防止未经授权的代码注入或篡改。如果部署机制依赖于网络通信，则使用加密方法来保护传输中的部署项目的完整性。

阅读更多内容

- Amazon IoT 中的安全性 (p. 263)
- 了解 Amazon IoT 安全模型
- Redhat：尝试 Python
- Python 中的 10 种常见安全问题和规避方法
- 什么是最小特权以及为什么需要它？
- 十大 OWASP 嵌入式安全实践
- OWASP IoT 项目

Device Advisor

Device Advisor 是一种基于云的完全托管式测试功能，用于在设备软件开发过程中验证 IoT 设备。Device Advisor 提供预构建的测试，您可以使用这些测试验证 IoT 设备与 Amazon IoT Core 之间的连接是否安全可靠，然后再将设备部署到生产环境。Device Advisor 的预构建测试可帮助您根据最佳实践验证您的设备软件，以便使用 TLS、MQTT、Device Shadow 和 IoT Jobs。您还可以下载已签名的资格报告，以提交 Amazon 合作伙伴网络，让您的设备符合 Amazon 合作伙伴设备目录的要求，而无需将您的设备发送到其中，更无需等待它进行测试。

Note

Device Advisor 在 us-east-1、us-west-2、ap-northeast-1、eu-west-1 区域中获得支持。Device Advisor 支持使用 X509 客户端证书的 MQTT。
Device Advisor 目前支持 RSA 服务器证书。

本章包含以下部分：

- 设置 (p. 1016)
- 在控制台中开始使用 Device Advisor (p. 1020)
- Device Advisor 工作流 (p. 1027)
- Device Advisor 详细控制台工作流 (p. 1031)
- 长时间测试控制台的工作流程 (p. 1040)
- Device Advisor VPC 端点（Amazon PrivateLink）(p. 1046)
- Device Advisor 测试用例 (p. 1048)

任何已构建连接到 Amazon IoT Core 的设备均可以使用 Device Advisor。您可以通过 Amazon IoT 控制台，或者使用 Amazon CLI 或 SDK，来访问 Device Advisor。准备好测试设备时，请使用 Amazon IoT Core 注册设备，然后使用 Device Advisor 终端节点配置设备软件。然后选择预构建的测试，加以配置，在您的设备上运行测试，随之获取测试结果以及详细的日志或资格报告。

Device Advisor 是 Amazon 云中的测试终端节点。若要测试设备，您可以配置设备使其连接到 Device Advisor 提供的测试终端节点。将设备配置成能够连接到测试终端节点后，您可以访问 Device Advisor 的控制台，或使用 Amazon SDK 选择要在设备上运行的测试。然后，Device Advisor 会管理测试的整个生命周期，包括资源调配、测试过程调度、管理状态机、记录设备行为、记录结果并以测试报告的形式提供最终结果。

设置

首次使用 Device Advisor 之前，请完成以下任务。

先决条件

- 创建 IoT 事物 (p. 1016)
- 创建要用作设备角色的 IAM 角色 (p. 1017)
- 为 IAM 用户创建自定义托管策略来使用设备顾问 (p. 1018)
- 创建 IAM 用户来使用 Device Advisor。 (p. 1018)
- 配置您的设备 (p. 1019)

创建 IoT 事物

首先，您需要创建事物并将证书附加到事物。使用下列教程创建事物：创建事物对象。

创建要用作设备角色的 IAM 角色

Note

您可以使用 Device Advisor 控制台快速创建设备角色。请参阅控制台中的 Device Advisor 入门了解使用 Device Advisor 控制台设置设备角色的步骤。

1. 前往 Amazon I IAM 控制台，然后登录您用于 Device Advisor 测试的账户。
2. 在左侧导航窗格中，选择 Policies（策略）。
3. 选择 Create policy（创建策略）。
4. 在 Create Policy（创建策略）下，执行以下操作。
   1. 对于 Service (服务)，请选择 IoT。
   2. 在 Actions（操作）下，或者根据附加到 IoT 事物或上一节中创建的证书（推荐）的策略选择操作，或者在筛选条件操作框中搜索然后选择下列操作。
      - 连接
      - 发布
      - Subscribe
      - 接收
      - RetainPublish
   3. 在 Resources（资源）下，要使用最佳安全实践，我们建议您使用以下步骤限制 client（客户端）、topic（主题）和 topicfilter（主题筛选条件）资源：
      a. 选择 Specify client resource ARN for the Connect action（为 Connect 操作指定客户端资源 ARN）。
         i. 选择 Add ARN (添加 ARN)。
         ii. 在可视化 ARN 编辑器中指定 region、accountId 和 clientId 或手动指定要用于运行测试用例的 IoT 主题的 Amazon Resource Names (ARN)。clientId 是您的设备用于与 Device Advisor 交互的 MQTT clientId。
         iii. 选择 Add（添加）。
      b. 选择 Specify topic resource ARN for the Receive and 1 more action（为 Receive 及另 1 个操作指定主题资源 ARN）。
         i. 选择 Add ARN (添加 ARN)。
         ii. 在可视化 ARN 编辑器中指定 region（区域）、accountId 和 topic name（主题名称）或手动指定要用于运行测试用例的 IoT 主题的 ARN。topic name（主题名称）是您的设备要用于发布消息的 MQTT topic（主题）。
         iii. 选择 Add（添加）。
      c. 选择 Specify topicfilter resource ARN for the Subscribe action（为 Subscribe 操作指定主题筛选条件资源 ARN）。
         i. 选择 Add ARN (添加 ARN)。
         ii. 在可视化 ARN 编辑器中指定 region（区域）、accountId 和 topic name（主题名称）或手动指定要用于运行测试用例的 IoT 主题的 ARN。topic name（主题名称）是您的设备用于订阅的 MQTT topic（主题）。
         iii. 选择 Add (添加)。
5. 选择Review policy（查看策略）。
6. 在 Review policy（查看策略）项下，输入一个 Name（名称）。
7. 选择 Create policy（创建策略）。
8. 在左侧导航窗格中，选择 Roles（角色）。
9. 请选择 Create Role(创建角色)。
10. 在Or select a service to view its use cases（或选择一个服务以查看其用例）中，选择 IoT。
11. 在选择您的使用案例下面，选择 IoT。

12. 选择Next: Permissions（下一步: 权限）。
13. （可选）在 Set permissions boundary（设置权限边界）项下，选择 Use a permissions boundary to control the maximum role permissions（使用权限边界控制最大角色权限），然后选择您刚刚创建的策略。
14. 请选择下一步：标签。
15. 请选择下一步：审核。
16. 输入 Role name（角色名称）和 Role description（角色描述）。
17. 选择 Create role（创建角色）。
18. 去到您创建的角色。
19. 在 Permissions（权限）选项卡中，选择 Attach policies（附加策略），然后选择您在步骤 4 中创建的策略。
20. 选择 Attach policy（附上策略）。
21. 选择 Trust relationships（信任关系）选项卡，然后选择 Edit trust relationship（编辑信任关系）。
22. 输入此策略：

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

23. 选择 Update Trust Policy（更新信任策略）。

为 IAM 用户创建自定义托管策略来使用设备顾问

1. 通过以下网址导航到 IAM 控制台：https://console.aws.amazon.com/iam/。如果提示，请输入您的 Amazon 凭证。
2. 在左侧导航窗格中，选择 Policies (策略)。
3. 选择 Create policy（创建策略），然后选择 JSON 选项卡。
4. 添加必要的权限以使用 Device Advisor。可在安全最佳实践主题中找到策略文档。
5. 请选择查看策略。
6. 输入名称和描述。
7. 请选择Create Policy（创建策略）。

创建 IAM 用户来使用 Device Advisor。

Note

我们建议您创建一个要在运行 Device Advisor 测试时使用的 IAM 用户。尽管可以使用管理员用户运行 Device Advisor 测试，但不建议这样做。

1. 导航到 IAM 控制台：https://console.aws.amazon.com/iam/如果出现提示，请输入要登录的 Amazon 凭证。

2. 在左侧导航窗格中，选择 Users（用户）。
3. 选择添加用户。
4. 输入 User name（用户名称）。
5. 如果用户需要在 Amazon Web Services Management Console 之外与 Amazon 交互，则需要编程式访问权限。Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

   要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中将临时凭证用于 Amazon 资源中的说明进行操作。
IAM	使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。 （不推荐使用）	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

6. 选择Next: Permissions（下一步: 权限）。
7. 要提供访问权限，请为您的用户、组或角色添加权限：
   - 通过身份提供商在 IAM 中托管的用户：

     创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。
   - IAM 用户：
     - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
     - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。
8. 在搜索框中输入您创建的自定义管理策略名称，然后选中左侧的 Policy name（策略名称）复选框。
9. 请选择下一步：标签。
10. 选择 Next: Review (下一步: 审核)。
11. 选择 Create user。
12. 选择 Close（关闭）。

Device Advisor 需要代表您访问 Amazon 资源（事物、证书和终端节点）的权限。您的 IAM 用户必须具有必要的权限。 CloudWatch 如果您将必要的权限策略附加到您的 IAM 用户，Device Advisor 也会将日志发布到 Amazon。

配置您的设备

Device Advisor 使用服务器名称指示 (SNI) TLS 扩展来应用 TLS 配置。设备在连接时必须使用此扩展，并传递与 Device Advisor 测试终端节点相同的服务器名称。

当测试处于 `Running` 状态时，Device Advisor 允许 TLS 连接，并会在每次测试运行之前和之后拒绝 TLS 连接。因此，我们还建议使用设备连接重试机制，以便在使用 Device Advisor 期间获得完全自动的测试体验。如果您运行具有多个测试用例的测试套件（例如 TLS 连接、MQTT 连接和 MQTT 发布），我们建议您为您的设备构建一个机制，每五秒便尝试连接到测试终端节点。然后，您能够以自动的方式按顺序运行多个测试用例。

Note

为了使您的设备软件准备好进行测试，我们建议您获取一个 SDK，该 SDK 可以连接 Amazon IoT Core 并使用为您的账户提供的 Device Advisor 测试终端节点更新 SDK。

Device Advisor 支持两种类型的终端节点：账户级终端节点和设备级终端节点。请选择最符合您使用案例的终端节点。要使用不同的设备同时运行多个测试套件，请使用设备级终端节点。运行以下命令以获取设备级别终端节点：

```
aws iotdeviceadvisor get-endpoint --thing-arn your-thing-arn
```

or

```
aws iotdeviceadvisor get-endpoint --certificate-arn your-certificate-arn
```

要一次运行一个测试套件，请选择账户级终端节点。运行以下命令获取账户级别终端节点：

```
aws iotdeviceadvisor get-endpoint
```

在控制台中开始使用 Device Advisor

本教程可帮助您快速开始在控制台中使用 Device Advisor。Device Advisor 提供了一些功能，例如必需的测试和签名的资格报告，以便根据 Amazon IoT Core 资格认证程序对 Amazon合作伙伴设备目录中的设备进行资格认证并加以列示。

有关使用 Device Advisor 的更多信息，请参阅 Device Advisor 工作流 (p. 1027) 和 Device Advisor 详细控制台工作流 (p. 1031)。

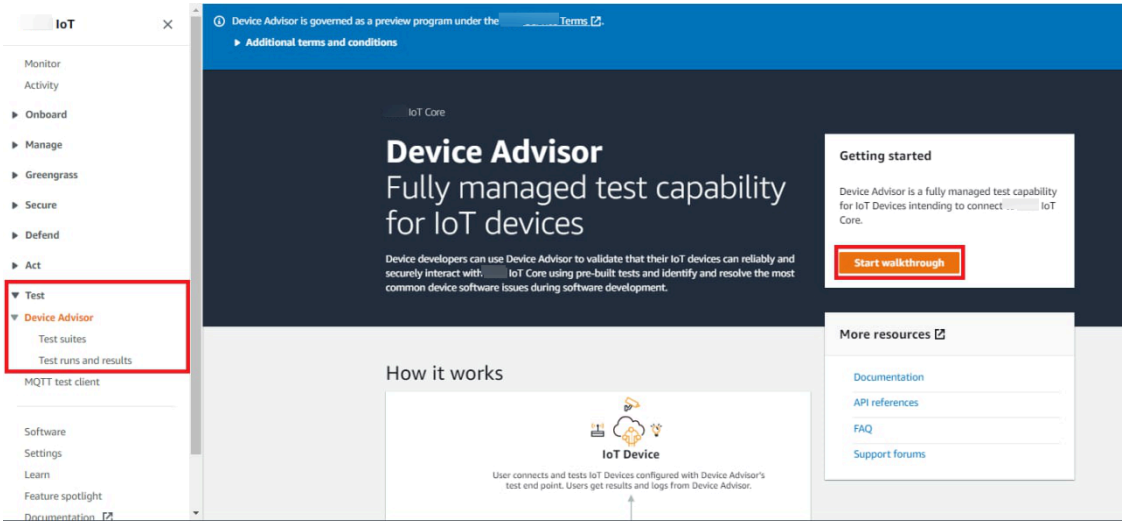
要完成本教程，请执行 设置 (p. 1016) 中概述的步骤。

Note

Device Advisor 在 us-east-1、us-west-2、ap-northeast-1、eu-west-1 区域中获得支持。

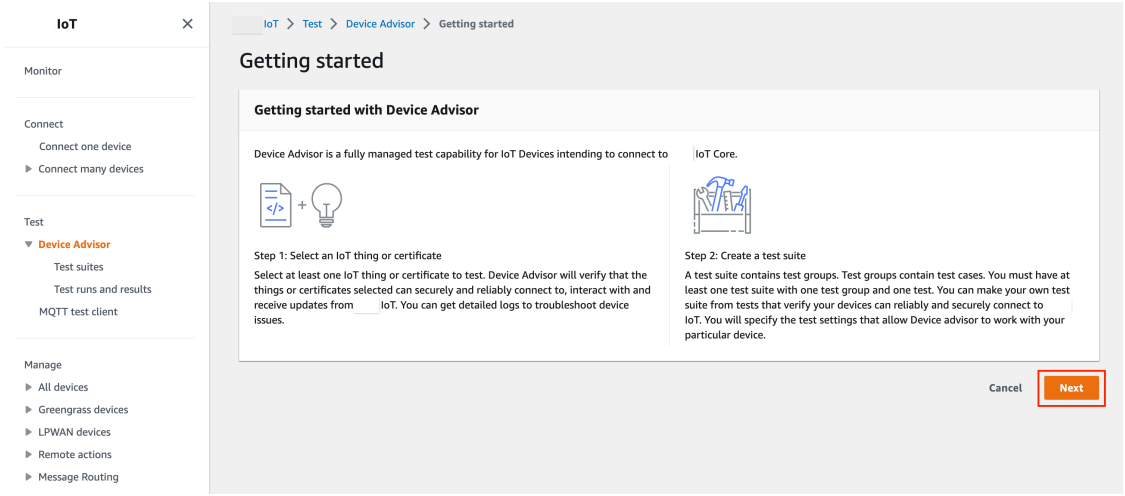
开始使用

1. 在 Amazon IoT 控制台的导航窗格中，展开 Test（测试）、Device Advisor，然后选择 Start walkthrough（开始演练）。



2. Getting started with Device Advisor（Device Advisor 入门）页面概览了创建测试套件和对您的设备运行测试所需的各个步骤。您还可以查找适用于您账户的 Device Advisor 测试终端节点。您必须在将用于测试的设备上配置固件或软件，以连接到此测试终端节点。

   要完成此教程，您需要创建事物和证书。



   查看信息后，选择 Next（下一步）。

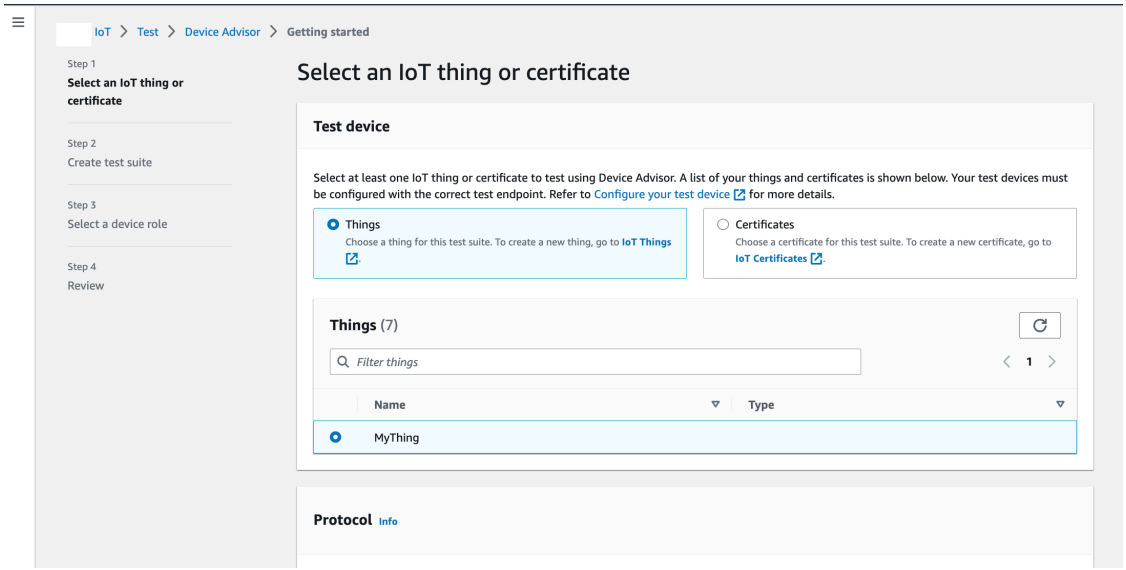
3. 在 Step 1（步骤 1）中，使用 Device Advisor 选择要进行测试的 Amazon IoT 事物或证书。如果您没有任何现有的事物或证书，请参阅 设置。

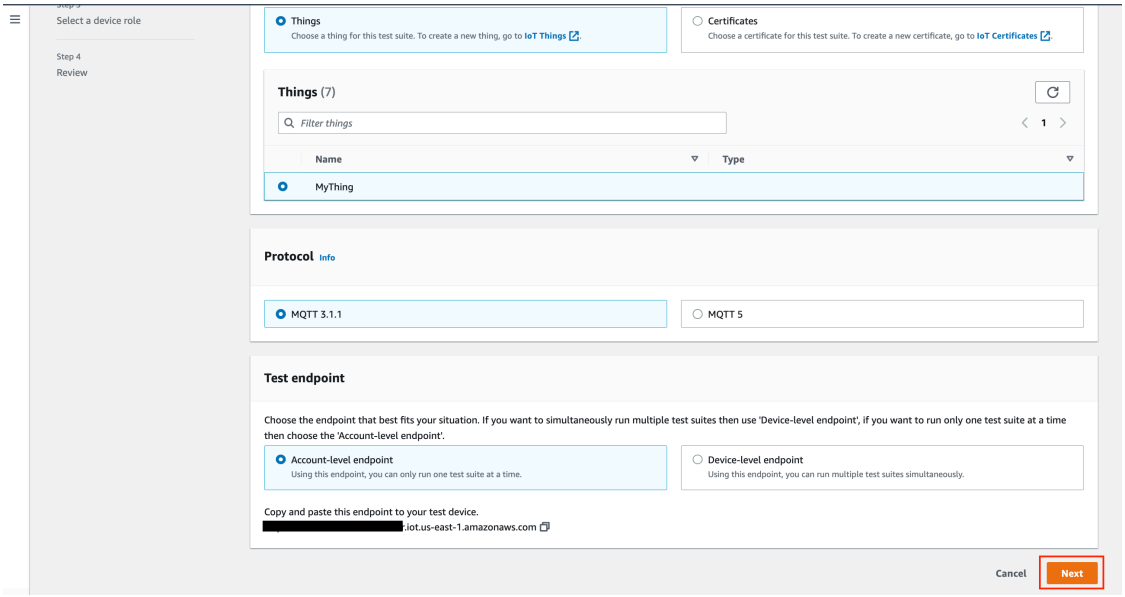
   对于协议，选择 MQTT 3.1.1 或 MQTT 5。

   在测试端节点部分中，选择最适合我们使用案例的终端节点。如果您计划使用相同的 Amazon 账户同时运行多个测试套件，选择设备级终端节点。否则，要一次运行一个测试套件，请选择账户级别终端节点。

   使用选定的 Device Advisor 的测试终端节点配置测试设备。

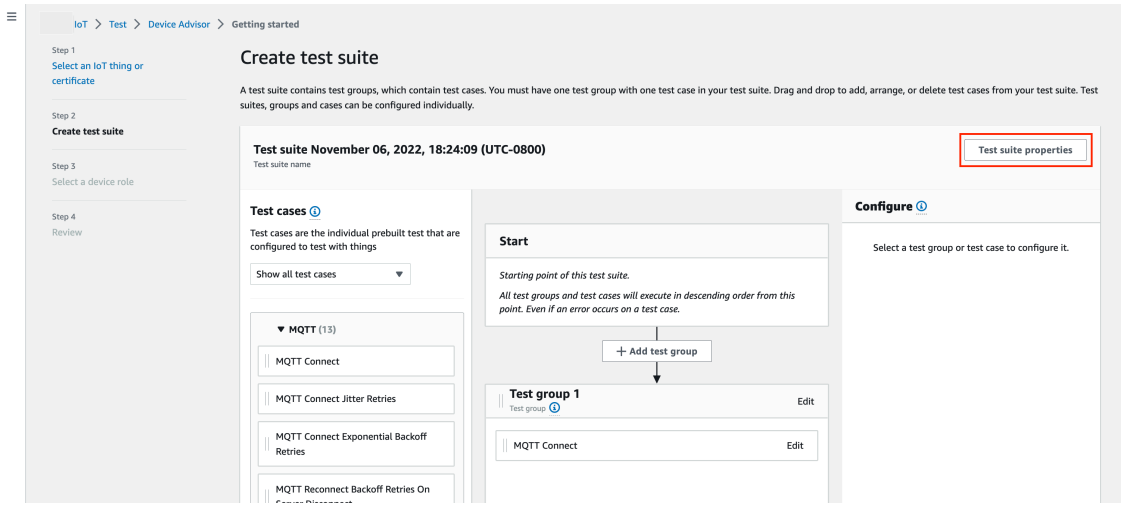
   选择下一步。





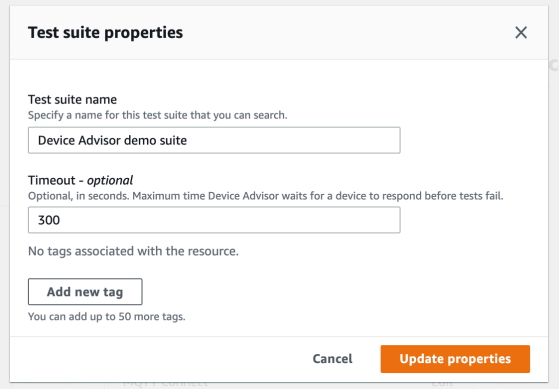
4. 在 Step 2（步骤 2）中，您可以创建和配置自定义测试套件。自定义测试套件必须拥有至少一个测试组，并且每个测试组必须拥有至少一个测试用例。我们为您添加了 MQTT Connect 测试用例，以便您能开始操作。

   选择 Test suite properties（测试套件属性）。创建测试套件时，必须提供测试套件属性。



   您可以配置如下套件级属性：

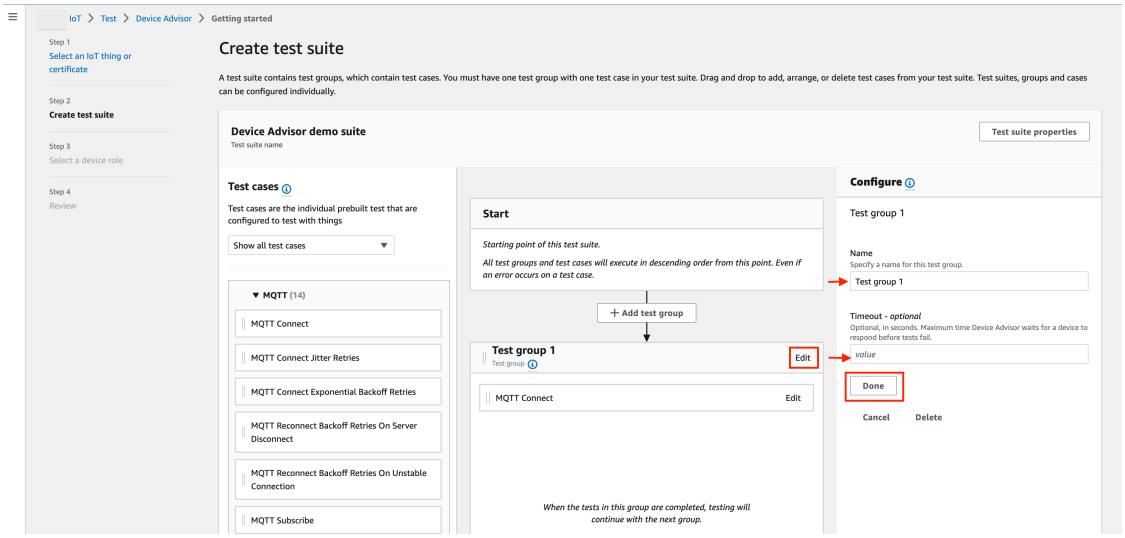
   - 测试套件名称：输入测试套件的名称。
   - Timeout（超时）（可选）：当前测试套件中每个测试用例的超时时间（以秒为单位）。如果您未指定超时值，则将使用默认值。
   - Tags（标签）（可选）：将标签添加到要创建的测试套件中。



完成后，选择 Update properties（更新属性）。

5. （可选）您可以选择测试组名称旁的 Edit（编辑）以更新测试套件组配置。

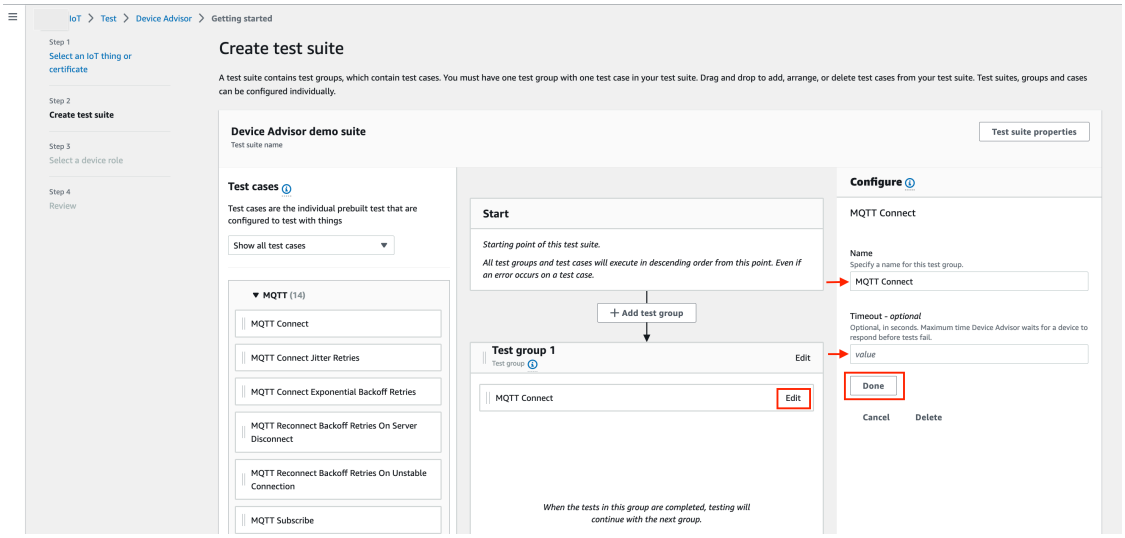
   - 名称：输入测试套件组的自定义名称。
   - Timeout（超时）（可选）：当前测试套件中每个测试用例的超时时间（以秒为单位）。如果您未指定超时值，则将使用默认值。



选择完成。

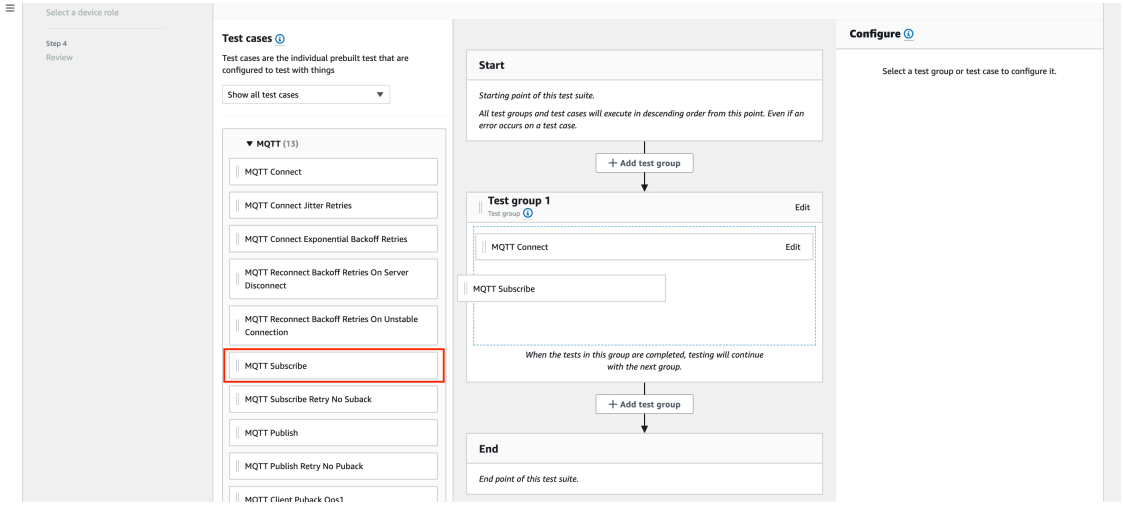
6. （可选）您可以选择测试用例名称旁的 Edit（编辑）以更新测试用例配置。

   - 名称：输入测试套件组的自定义名称。
   - Timeout（超时）（可选）：所选测试用例的超时时间（以秒为单位）。如果您未指定超时值，则将使用默认值。



选择完成。

7. （可选）要向测试套件添加更多测试组，请选择添加测试组，然后按照步骤 5 中的说明进行操作。
8. （可选）要添加更多测试用例，请将出现的 Test cases（测试用例）下的测试用例拖动到您拥有的任何测试组中。

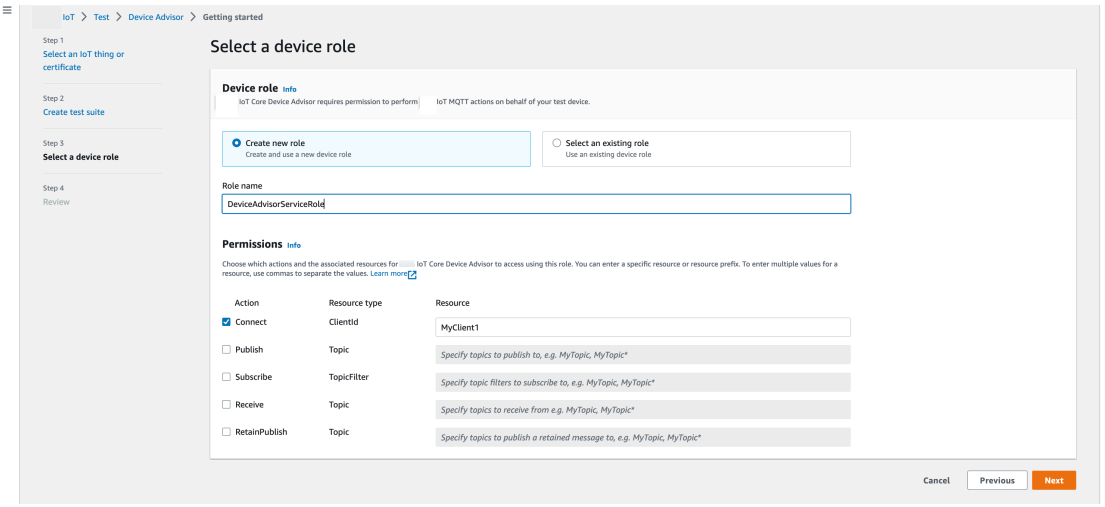


9. 通过选择并拖动列出的测试用例，可以对测试组和测试用例进行重新排序。Device Advisor 将按照测试用例列出的顺序运行测试。

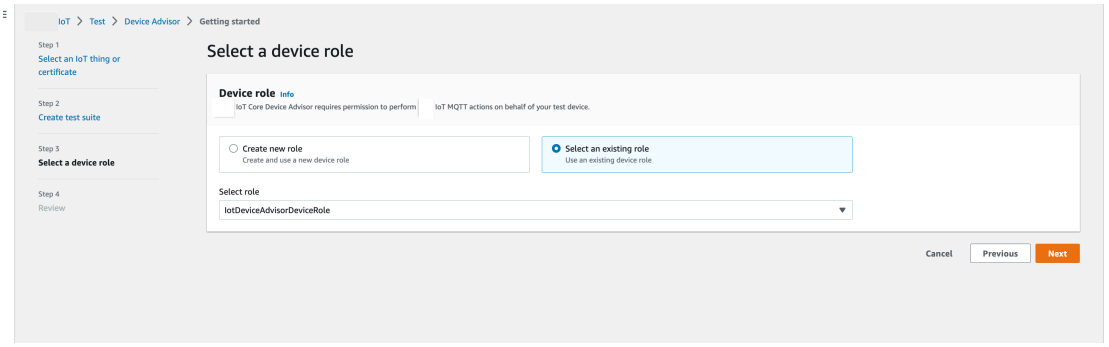
   配置测试套件后，选择 Next（下一步）。

10. 在步骤 3 中，您可以配置设备角色， Device Advisor 将使用该角色代表您的测试设备来执行 Amazon IoT MQTT 操作。如果您选择 MQTT 连接仅限测试用例，将自动选择 Connect（连接）操作，因为运行此测试套件需要对设备角色具有该权限。如果您选择了其他测试用例，将选择相应的操作。

    为每个选定的操作提供资源值。例如，对于 Connect（连接）操作，提供客户端 ID，您的设备将使用该 ID 连接到 Device Advisor 终端节点。您可以通过使用逗号分隔值来提供多个值，也可以使用通配符 (*) 字符提供前缀值。例如，为任何 `MyTopic` 开头的主题提供发布权限，您可以将“`MyTopic*`”作为资源值。



如果您之前已经在设置创建了设备角色，您希望使用该角色，请选择 Select an existing role（选择现有角色），然后在 Select role（选择角色）下面选择您的设备角色。



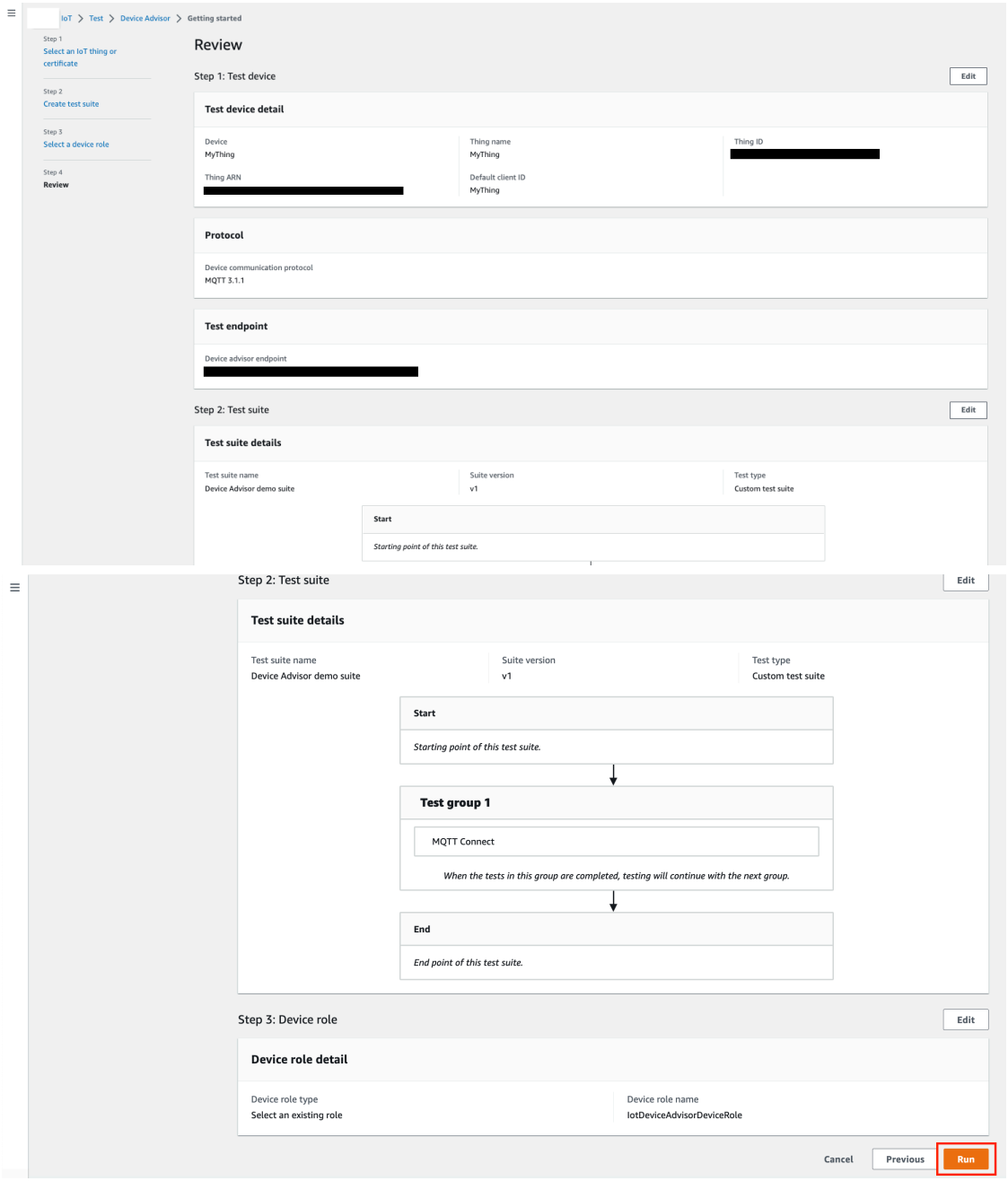
使用提供的两个选项之一配置设备角色，然后选择 Next（下一步）。

11. Step 4（步骤 4）说明了已配置的所选测试设备、测试终端节点、测试套件和测试设备角色的概览。如果要对这节进行更改，请选择要编辑的部分上方的 Edit（编辑）。

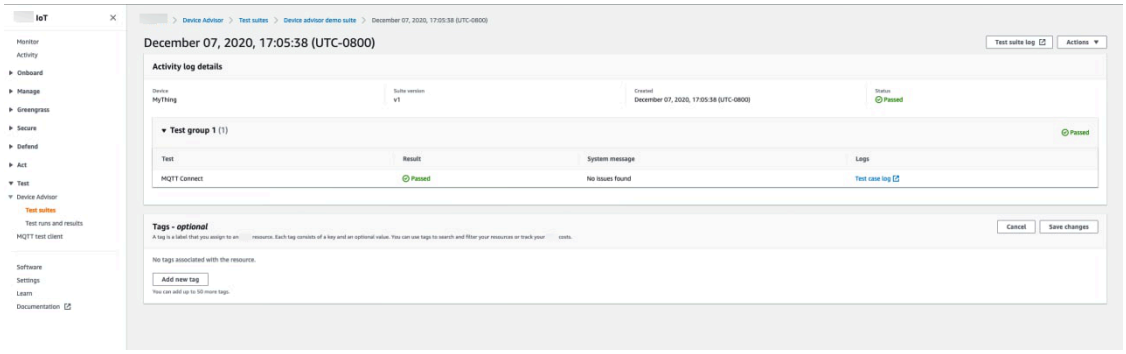
    Note

    为了获得最佳结果，您可以在启动套件运行之前将选定的测试设备连接到 Device Advisor 测试终端节点。我们建议您为设备建立一个机制，以便每五秒便尝试一次连接到测试终端节点，每次持续一到两分钟。

    要创建测试套件并对您的设备运行所选的测试，请选择 Run（运行）。



12. 在导航窗格中，展开 Test（测试）、Device Advisor，然后选择 Test runs and results（测试运行和结果），以查看运行详细信息和日志。选择已启动的测试套件运行查看运行详细信息和日志。



13. 要访问运行套件件件 CloudWatch 日志：
    - 选择测试套件 CloudWatch 日志。
    - 选择任意测试用例的 Test case l og（测试用例日 CloudWatch 志）。
14. 根据您的测试结果，对您的设备进行故障排除操作，直到通过所有测试。

Device Advisor 工作流

本教程将教您如何创建自定义测试套件，以及如何对要在控制台中测试的设备运行测试。测试完成后，您可以查看测试结果和详细日志。

教程

- 先决条件 (p. 1027)
- 创建测试套件定义 (p. 1027)
- 获取测试套件定义 (p. 1029)
- 获取测试终端节点 (p. 1029)
- 启动测试套件运行 (p. 1029)
- 运行一个测试套件 (p. 1030)
- 停止测试套件运行 (p. 1030)
- 获取成功的资格测试套件运行的资格报告 (p. 1030)

先决条件

要完成本教程，请完成 设置 (p. 1016) 中概述的步骤。

创建测试套件定义

首先，安装 Amazon SDK。

rootGroup 语法

根组是 JSON 字符串，用于指定测试套件中包含哪些测试用例以及这些测试用例的任何必要配置。使用根组以您喜欢的任何方式构建并排列测试套件。测试套件的层次结构如下：

```
test suite # test group(s) # test case(s)
```

测试套件必须至少有一个测试组，并且每个测试组必须至少有一个测试用例。Device Advisor 将按照定义的测试组和测试用例顺序运行测试。

每个根组都遵循此基本结构：

```
{
    "configuration": {  // for all tests in the test suite
        "": ""
    }
    "tests": [{
        "name": ""
        "configuration": {  // for all sub-groups in this test group
            "": ""
        },
        "tests": [{
            "name": ""
            "configuration": {  // for all test cases in this test group
                "": ""
            },
            "test": {
                "id": ""
                "version": ""
            }
        }]
    }]
}
```

一个包含 "name"、"configuration" 和 "tests" 的数据块被称为“组定义”。一个包含 "name"、"configuration" 和 "test" 的数据库被称为“测试用例定义”。每个包含 "id" 和 "version" 的 "test" 数据块都被称为“测试用例”。

有关如何填写每个测试用例（"test" 数据块）中 "id" 和 "version" 字段的信息，请参阅 Device Advisor 测试用例 (p. 1048)。该部分还包含有关可用 "configuration" 设置的信息。

以下是根组配置的示例，该示例指定了“MQTT Connect Happy Case”和“MQTT Connect Exponential Backoff Retries”测试用例，以及配置字段的说明。

```
{
    "configuration": {},  // Suite-level configuration
    "tests": [            // Group definitions should be provided here
      {
        "name": "My_MQTT_Connect_Group",  // Group definition name
        "configuration": {}               // Group definition-level configuration,
        "tests": [                        // Test case definitions should be provided here
        {
            "name": "My_MQTT_Connect_Happy_Case",  // Test case definition name
            "configuration": {
                "EXECUTION_TIMEOUT": 300        // Test case definition-level
 configuration, in seconds
            },
            "test": {
                "id": "MQTT_Connect",              // test case id
                "version": "0.0.0"                 // test case version
            }
        },
        {
            "name": "My_MQTT_Connect_Jitter_Backoff_Retries",  // Test case definition name
            "configuration": {
                "EXECUTION_TIMEOUT": 600                // Test case definition-level
 configuration,  in seconds
            },
            "test": {
                "id": "MQTT_Connect_Jitter_Backoff_Retries",  // test case id
                "version": "0.0.0"                                  // test case version
            }
        }]
    }]
```

```
}
```

创建测试套件定义时，您必须提供根组配置。保存在响应对象中返回的 suiteDefinitionId。此 ID 用于检索测试套件定义信息和运行测试套件。

以下是一个 Java SDK 示例：

```
response = iotDeviceAdvisorClient.createSuiteDefinition(
        CreateSuiteDefinitionRequest.builder()
            .suiteDefinitionConfiguration(SuiteDefinitionConfiguration.builder()
                .suiteDefinitionName("your-suite-definition-name")
                .devices(
                    DeviceUnderTest.builder()
                        .thingArn("your-test-device-thing-arn")
                        .certificateArn("your-test-device-certificate-arn")
                        .build()
                )
                .rootGroup("your-root-group-configuration")
                .devicePermissionRoleArn("your-device-permission-role-arn")
                .protocol("MqttV3_1_1 || MqttV5"")
                .build()
            )
            .build()
)
```

获取测试套件定义

创建测试套件定义后，您会在 CreateSuiteDefinition API 的响应对象中收到 suiteDefinitionId。

您可能会看到返回的根组的各个组和测试用例定义中都有新的 id 字段。这是预期的结果，您可以使用这些 ID 来运行测试套件定义中的子集。

Java SDK 示例：

```
response = iotDeviceAdvisorClient.GetSuiteDefinition(
    GetSuiteDefinitionRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .build()
)
```

获取测试终端节点

使用 GetEndpoint API 获取设备使用的测试端点。选择终端节点时，请选择最适合情况的终端节点。要同时运行多个测试套件，请通过提供 thing ARN 或 certificate ARN 使用设备级端节。要运行单个测试套件，不提供参数选择账户级终端节点。

SDK 示例：

```
response = iotDeviceAdvisorClient.getEndpoint(GetEndpointRequest.builder()
.certificateArn("your-test-device-certificate-arn")
.thingArn("your-test-device-thing-arn")
.build())
```

启动测试套件运行

在成功创建测试套件定义并配置测试设备以连接到 Device Advisor 测试终端节点之后，请使用 StartSuiteRun API 运行您的测试套件。使用 certificateArn 或者 thingArn 运行测试套件。如果两者都完成了配置，如果证书属于该事物，则将使用该证书。

对于 .parallelRun()，如果通过一个 Amazon 帐户使用设备级终端节点并行运行多个测试套件，使用 true。

SDK 示例：

```
response = iotDeviceAdvisorClient.startSuiteRun(StartSuiteRunRequest.builder()
.suiteDefinitionId("your-suite-definition-id")
.suiteRunConfiguration(SuiteRunConfiguration.builder()
    .primaryDevice(DeviceUnderTest.builder()
        .certificateArn("your-test-device-certificate-arn")
        .thingArn("your-test-device-thing-arn")
        .build())
    .parallelRun(true | false)
    .build())
.build())
```

保存在响应中返回的 suiteRunId。您会用此来检索此测试套件运行的结果。

运行一个测试套件

启动测试套件运行后，您可以使用 GetSuiteRun API 检查其进度及结果。

SDK 示例：

```
// Using the SDK, call the GetSuiteRun API.

response = iotDeviceAdvisorClient.GetSuiteRun(
GetSuiteRunRequest.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

停止测试套件运行

要停止仍在运行的测试套件，可以调用 StopSuiteRun API。在您调用 StopSuiteRun API 后，则该服务将启动清理程序。当服务运行清理程序时，测试套件运行状态将会升级到 Stopping。清理程序需要几分钟时间，一旦完成，测试套件运行状态将会升级到 Stopped 状态。测试运行完全停止后，您将能够启动另一个测试套件运行。您可以使用 GetSuiteRun API 定期检查套件运行状态，如上一节所示。

SDK 示例：

```
// Using the SDK, call the StopSuiteRun API.

response = iotDeviceAdvisorClient.StopSuiteRun(
StopSuiteRun.builder()
    .suiteDefinitionId("your-suite-definition-id")
    .suiteRunId("your-suite-run-id")
.build())
```

获取成功的资格测试套件运行的资格报告

如果您运行的资格测试套件已完成，则可以使用 GetSuiteRunReport API 检索资格报告。您可以使用此资格报告来使用 Amazon IoT Core 资格计划为您的设备获取资格。要确定测试套件是否为合资格测试套件，请检查 intendedForQualification 参数是否设置为 true。在您调用 GetSuiteRunReport API 后，返回的下载 URL 可在 90 秒内供您下载。如果从上次调用 GetSuiteRunReport API 之后超过 90 秒，请再次调用该 API 以检索有效的 URL。

SDK 示例：

```
// Using the SDK, call the getSuiteRunReport API.

response = iotDeviceAdvisorClient.getSuiteRunReport(
    GetSuiteRunReportRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .suiteRunId("your-suite-run-id")
        .build()
)
```

Device Advisor 详细控制台工作流

在本教程中，您将创建自定义测试套件，并对要在控制台中测试的设备运行测试。测试完成后，您可以查看测试结果和详细日志。

教程

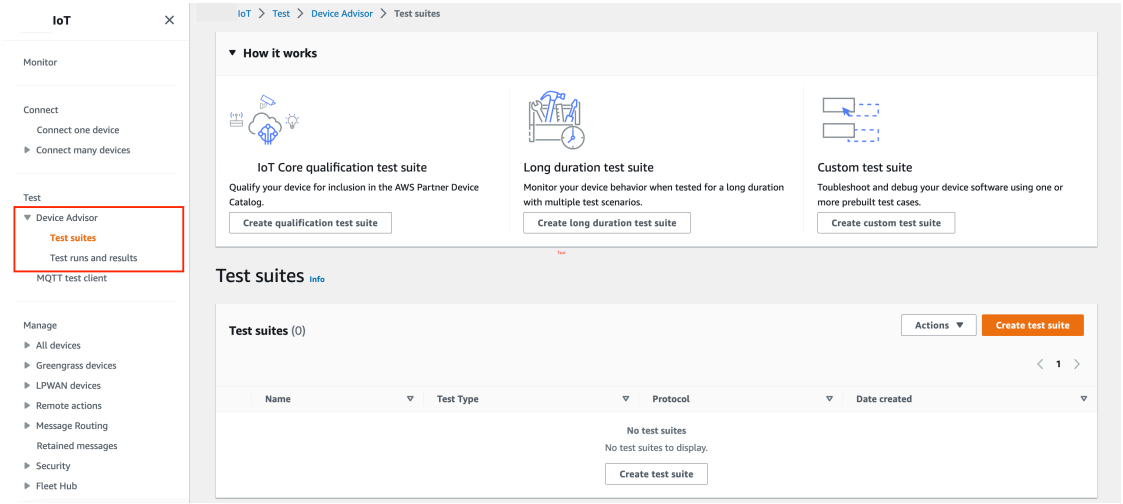
- 先决条件 (p. 1031)
- 创建测试套件定义 (p. 1031)
- 启动测试套件运行 (p. 1036)
- 停止测试套件运行（可选） (p. 1038)
- 查看测试套件运行详细信息和日志 (p. 1039)
- 下载 Amazon IoT 资格报告 (p. 1040)

先决条件

要完成此教程，您需要创建事物和证书。

创建测试套件定义

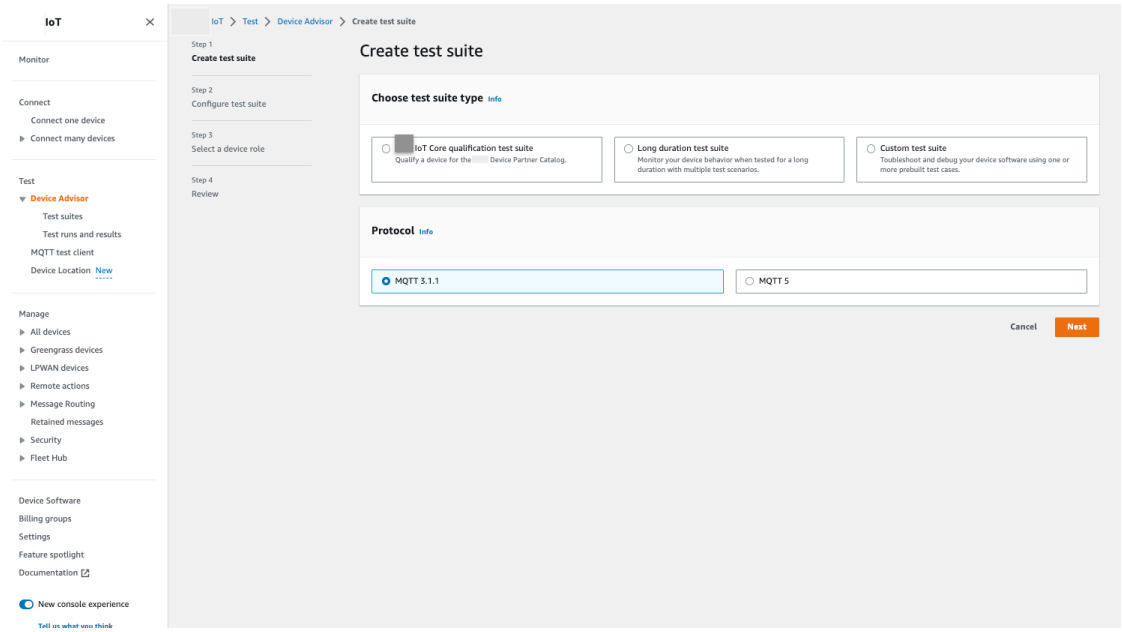
1. 在 Amazon IoT 控制台的导航窗格中，展开 Test（测试）、Device Advisor，然后选择 Test suites（测试套件）。



选择 Create Test Suite（创建测试套件）。

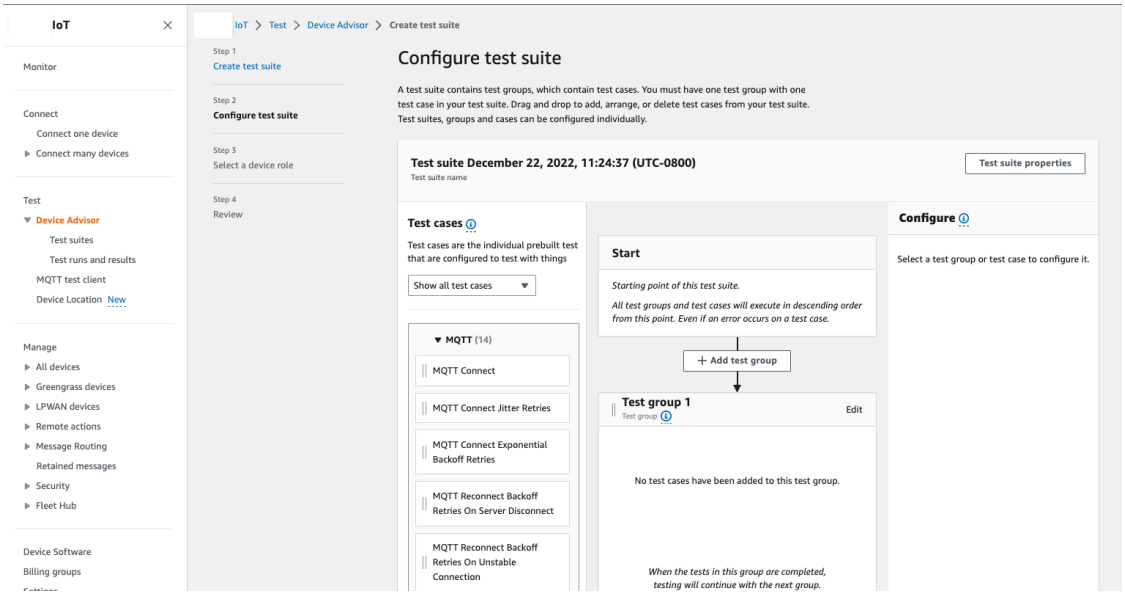
2. 选择 `Use the Amazon Qualification test suite` 或 `Create a new test suite`。

   对于协议，选择 MQTT 3.1.1 或 MQTT 5。

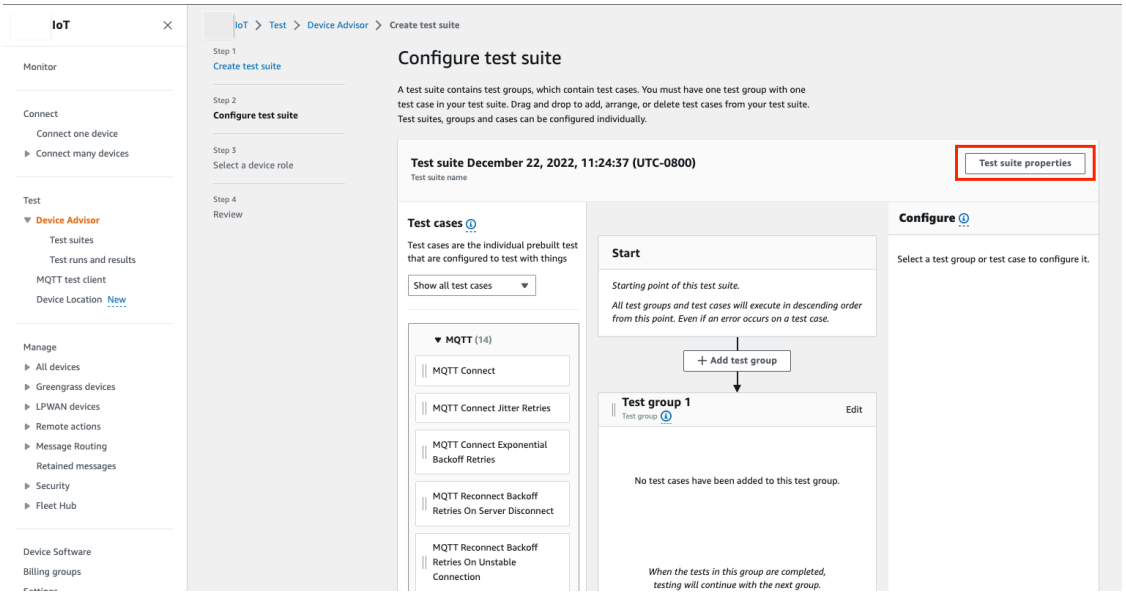


选择 `Use the Amazon Qualification test suite` 授予设备资格并将其列在 Amazon 合作伙伴设备目录中。通过选择此选项，可以预先选择使您的设备符合 Amazon IoT Core 资格计划所需资格的测试用例。无法添加或删除测试组和测试用例。您仍然需要配置测试套件属性。

选择 `Create a new test suite` 来创建并配置自定义测试套件。我们建议从此选项开始进行初始测试和故障排除。自定义测试套件必须拥有至少一个测试组，并且每个测试组必须拥有至少一个测试用例。在本教程中，我们将选择此选项，然后选择 Next（下一步）。

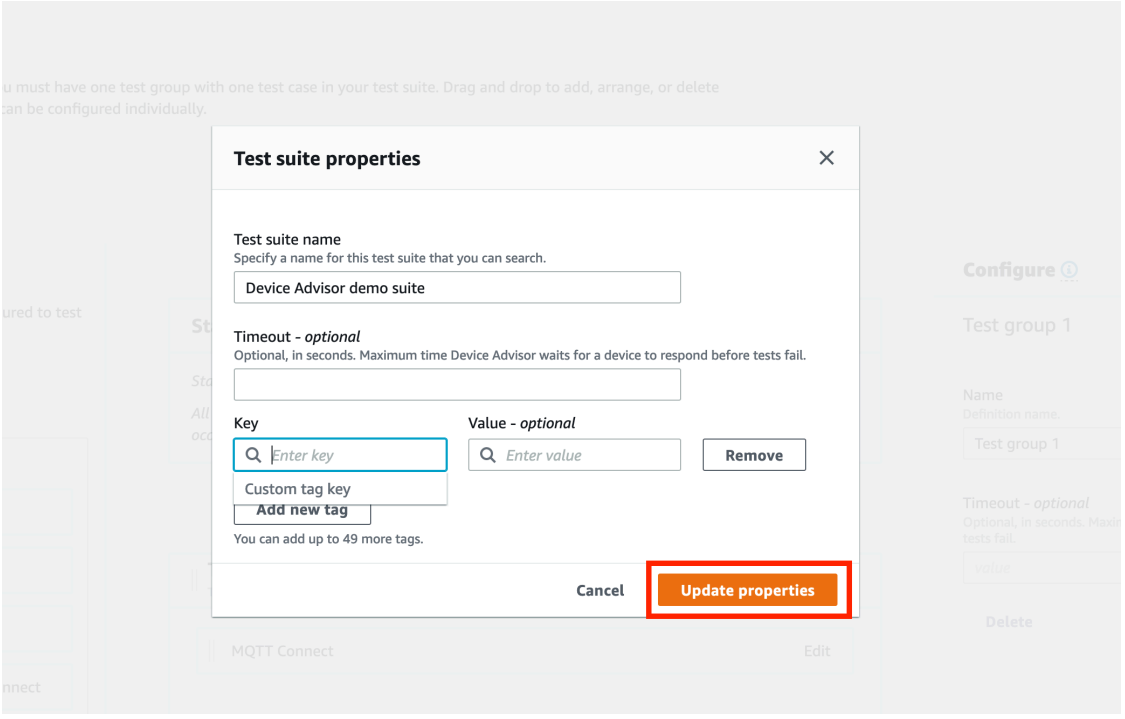


3. 选择 Test suite properties（测试套件属性）。创建测试套件时，您必须创建测试套件属性。



在 Test suite properties（测试套件属性）项下，请填写以下内容：

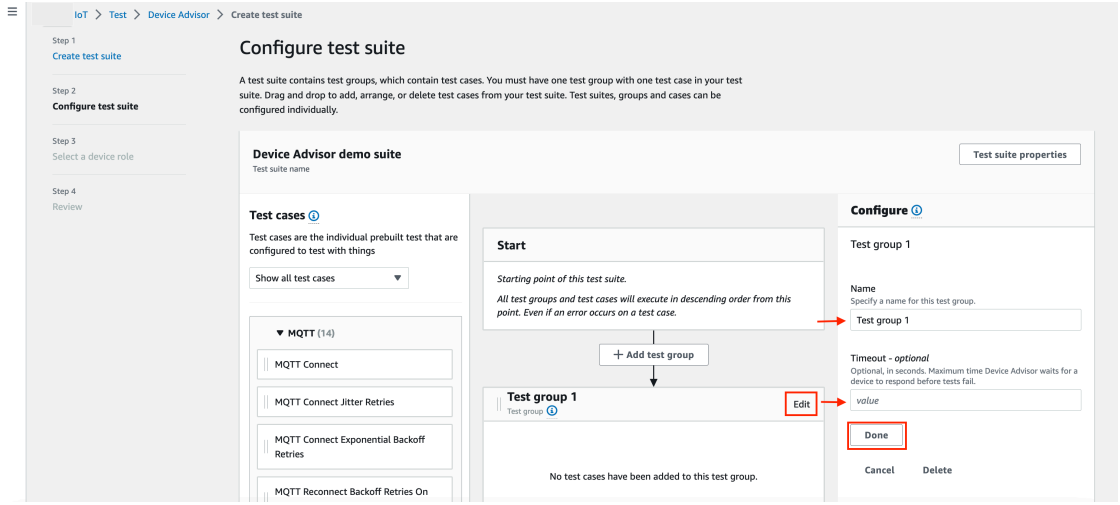
- Test suite name（测试套件名称）：您可以使用自定义名称创建套件。
- Timeout（超时）（可选）：当前测试套件中每个测试用例的超时时间（以秒为单位）。如果您未指定超时值，则将使用默认值。
- 标签（可选）：向测试套件添加标签。



完成后，选择 Update properties（更新属性）。

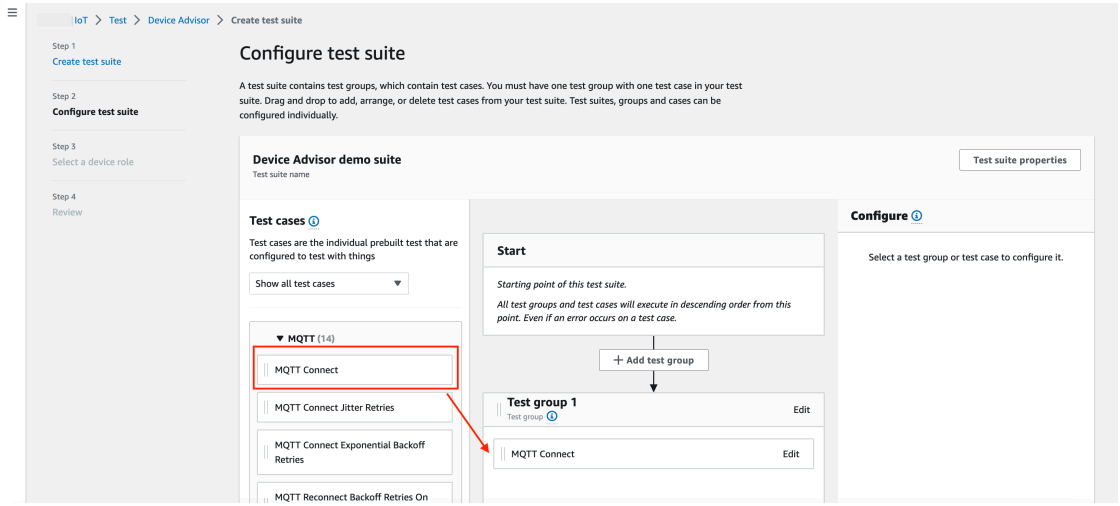
4. 要修改组级别配置，请在 `Test group 1` 下，选择 Edit（编辑）。然后，输入 Name（名称）为组指定一个自定义名称。

   或者，您也可以在所选的测试组下输入 Timeout（超时）值（以秒为单位）。如果您未指定超时值，则将使用默认值。



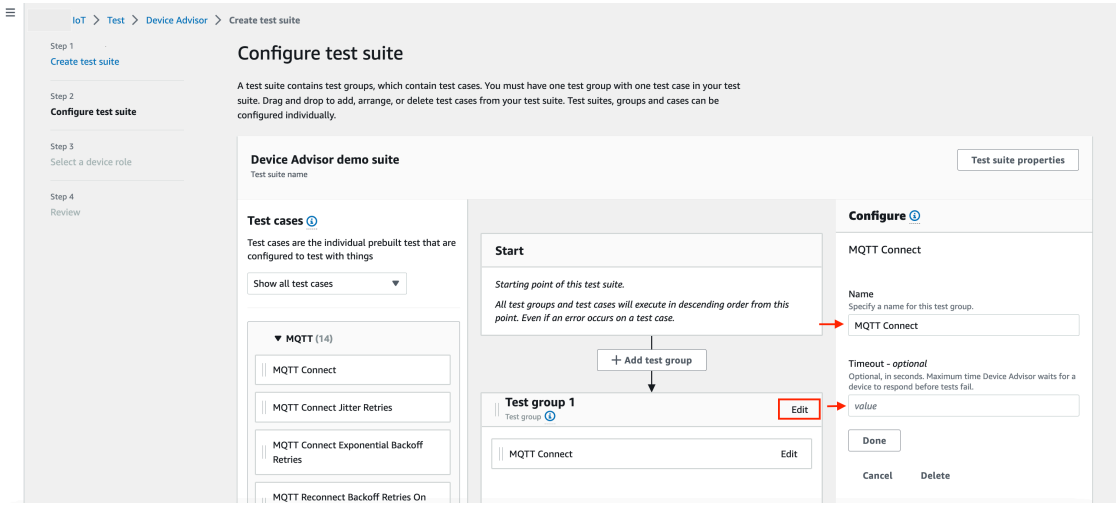
   选择完成。

5. 从 Test cases（测试用例）中拖动一个可用的测试用例加入测试组。



6. 要修改您添加到测试组的测试用例级别配置，请选择 Edit（编辑）。然后，输入 Name（名称）为组指定一个自定义名称。

   或者，您也可以在所选的测试组下输入 Timeout（超时）值（以秒为单位）。如果您未指定超时值，则将使用默认值。



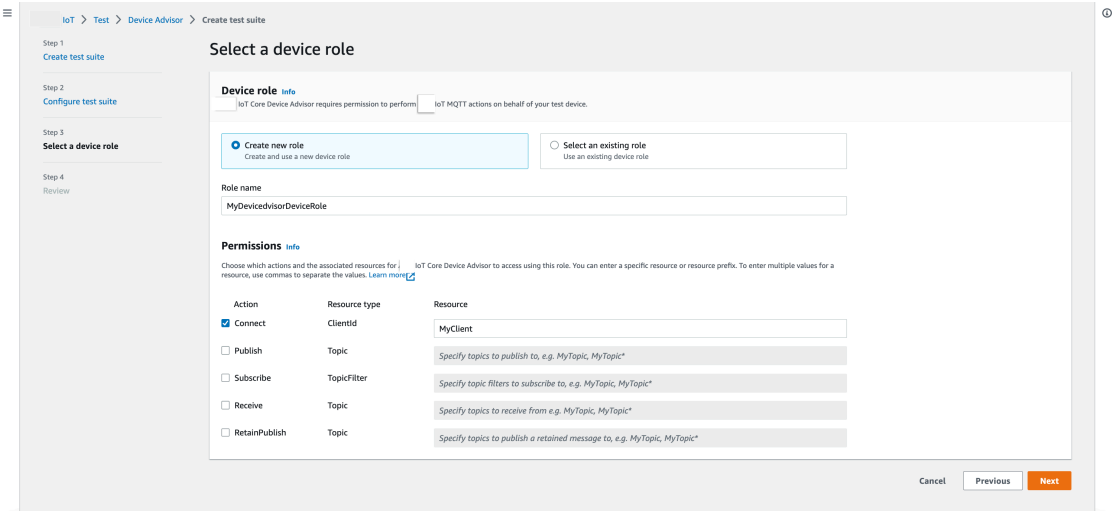
选择完成。

Note

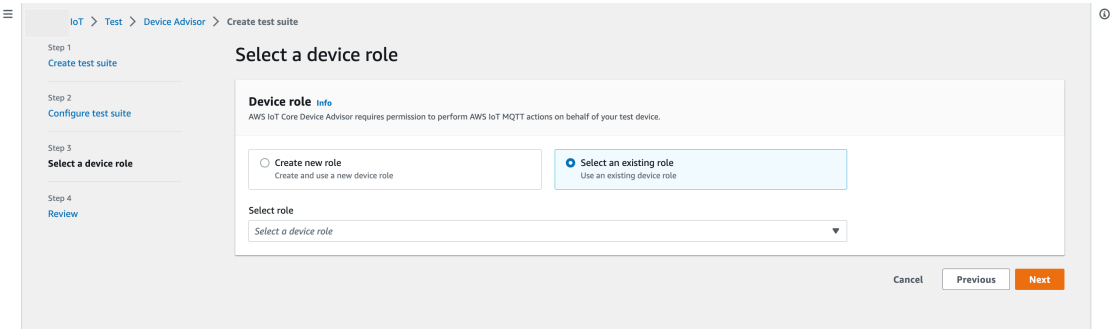
要向测试套件添加更多测试组，请选择 Add test group（添加测试组）。按照上述步骤创建和配置更多的测试组，或将更多测试用例添加到一个或多个测试组。可以通过选择并拖动测试用例到目标位置来重新排序测试组和测试用例。Device Advisor 将按照定义的测试组和测试用例顺序运行测试。

7. 选择下一步。
8. 在步骤 3中，配置 Device Advisor 将用于代表测试设备执行 Amazon IoTMQTT 操作的设备角色。

如果您仅在步骤 2中选择 MQTT Connect 测试用例，将自动检查 Connect(连接)操作，因为运行此测试套件需要对设备角色具有该权限。如果您选择了其他测试用例，将检查相应的所需操作。确保提供了每个操作的资源值。例如，对于 Connect(连接)操作，提供设备将连接到 Device Advisor 终端节点的客户端 id。您可以通过使用逗号分隔值来提供多个值，也可以使用通配符 (*) 字符提供前缀值。例如，为任何 `MyTopic` 开头的主题提供发布权限，您可以将“`MyTopic*`”作为资源值。



如果您之前已经创建了设备角色，您希望使用该角色，请选择 Select an existing role（选择现有角色）然后在 Select role（选择角色）下面选择您的设备角色。



使用提供的两个选项之一配置设备角色，然后选择 Next（下一步）。

9. 在步骤 4中，请确保每个步骤中提供的配置准确无误。要编辑为特定步骤提供的配置，请选择 Edit（编辑）用于相应的步骤。

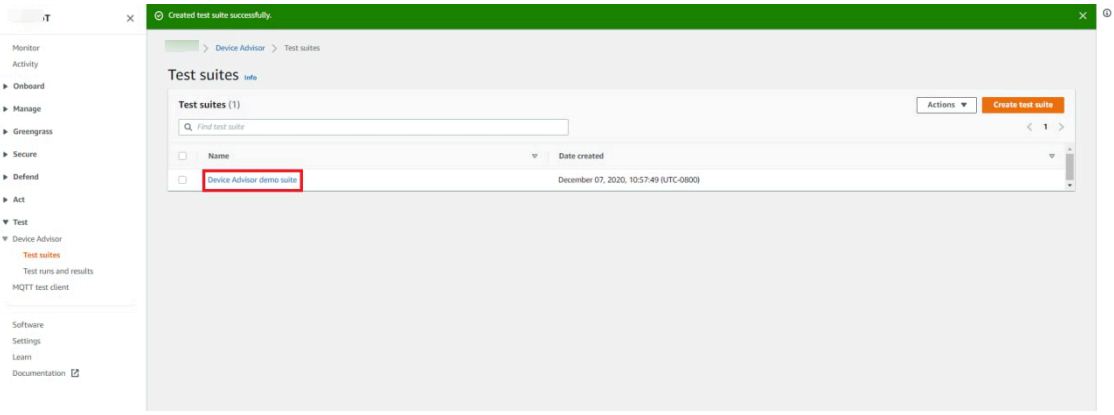
   验证配置后，选择 Create test suite（创建测试套件）。

   测试套件应能成功创建，并且您将被重定向到 Test suite（测试套件）页面，您可以在其中查看已创建的所有测试套件。

   如果测试套件创建失败，请确保已按照之前的说明配置测试套件、测试组、测试用例和设备角色。

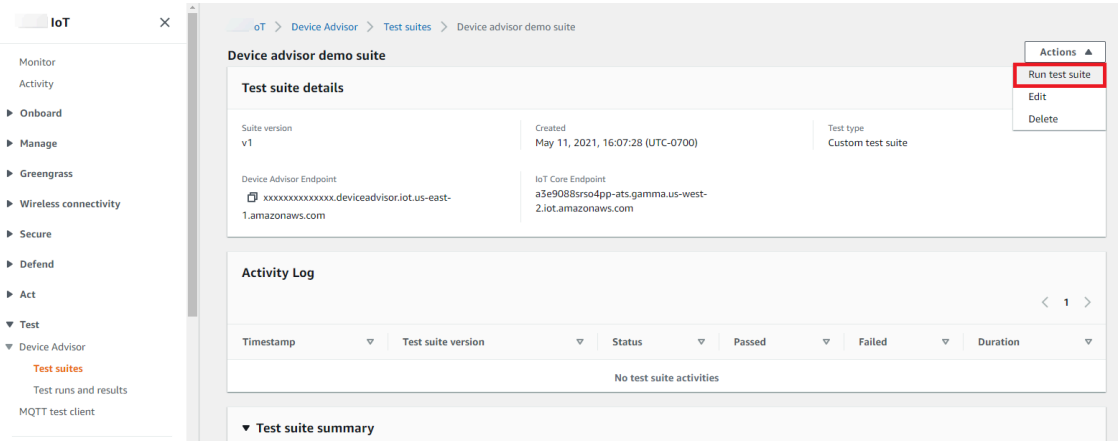
启动测试套件运行

1. 在Amazon IoT控制台的导航窗格中，展开测试、Device Advisor，然后选择 Test suite（测试套件）。
2. 选择要查看其测试套件详细信息的测试套件。



测试套件详细信息页面将显示与测试套件相关的所有信息。

3. 选择 Actions（操作），然后选择 Run test suite（运行测试套件）。

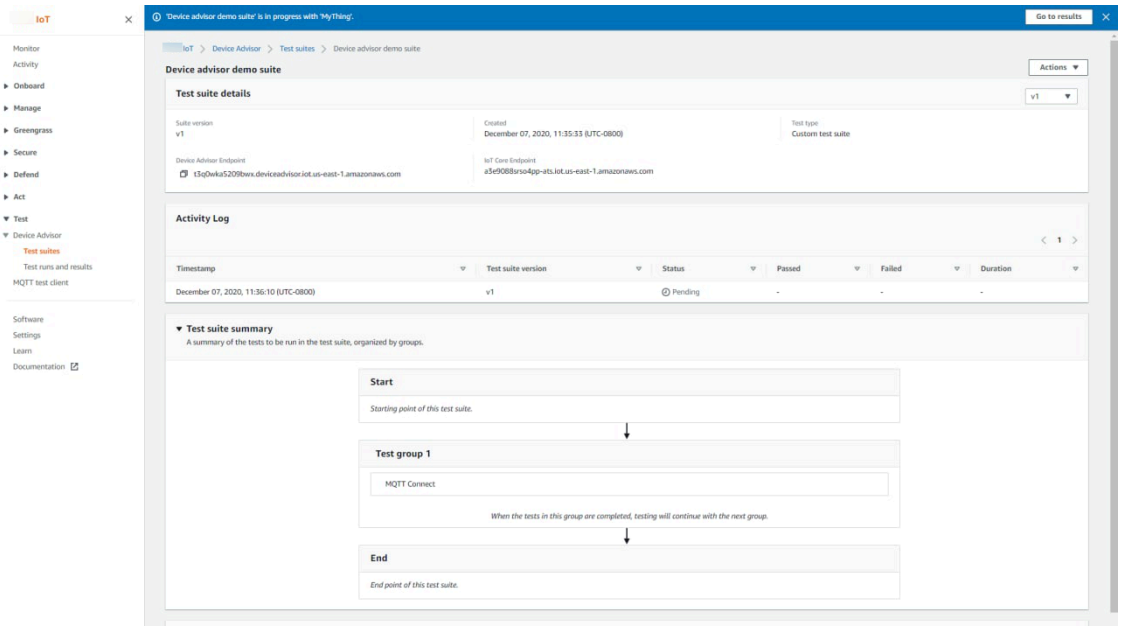


4. 在 Run configuration（运行配置）项下，您需要选择 Amazon IoT 事物或证书以使用 Device Advisor 进行测试。如果您没有任何现有事物或证书，请先创建 Amazon IoT Core 资源 (p. 1016)。

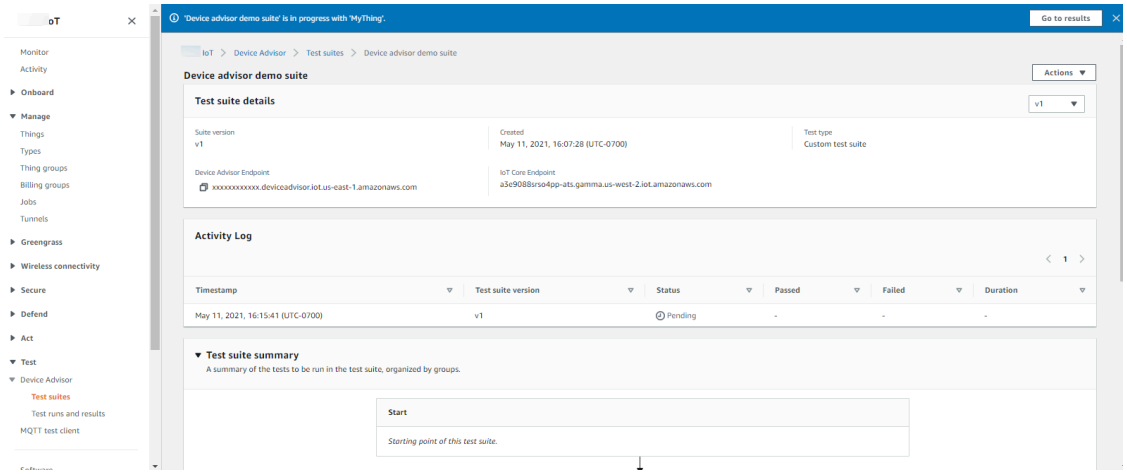
   在测试终端节点中，选择最适合您用例的终端节点。如果您计划使用相同的 Amazon 账户同时运行多个测试套件，选择 Device-level endpoint（设备级终端节点）。否则，如果您计划一次只运行一个测试套件，请选择 Account-level endpoint（账户级别终端节点）。

   使用选定的 Device Advisor 的测试终端节点配置测试设备。

   选择事物或证书以及 Device Advisor 终端节点后，选择 Run test（运行测试）。

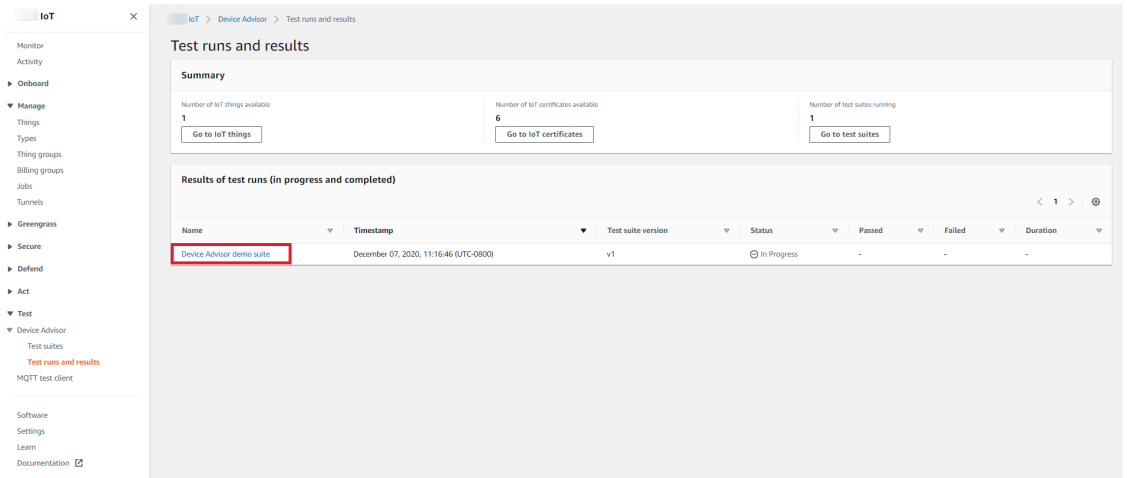


5. 选择顶部广告条的 Go to result（转到结果），以查看测试运行详细信息。

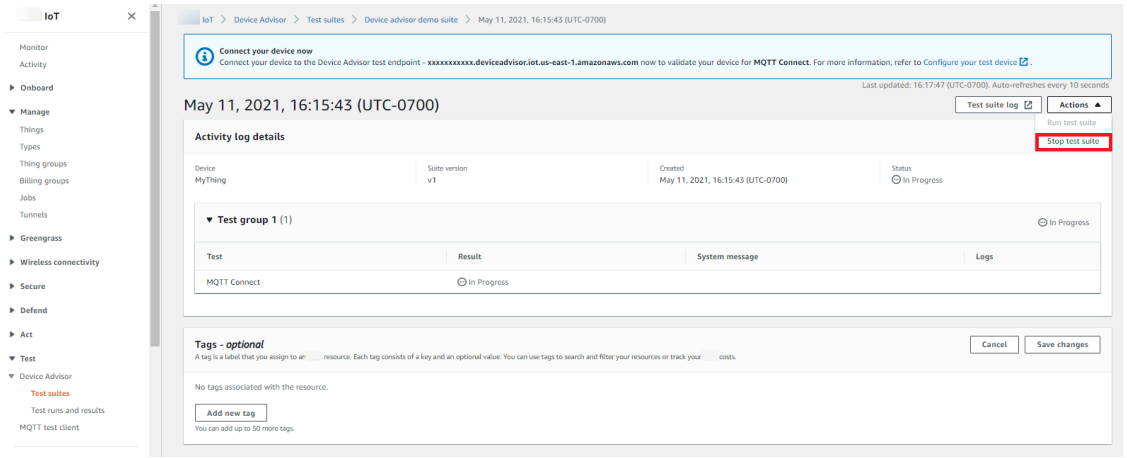


停止测试套件运行（可选）

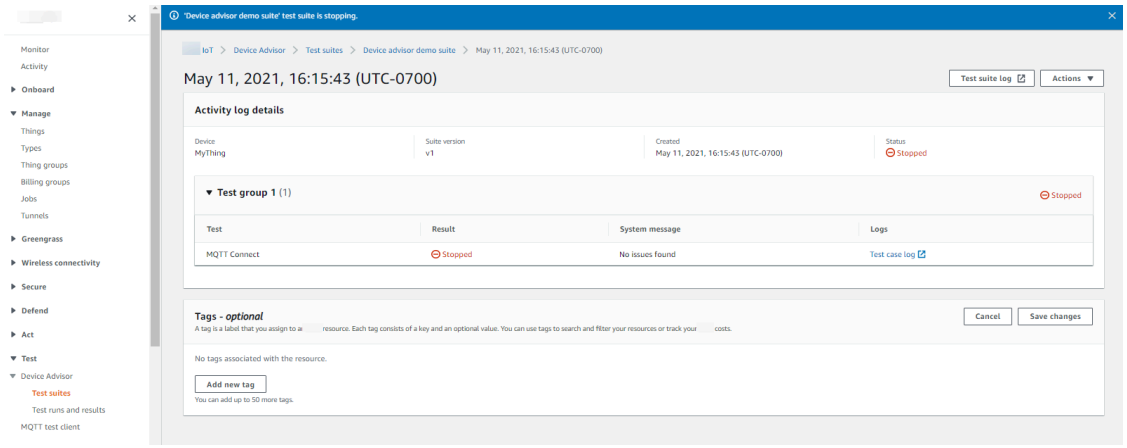
1. 在 Amazon IoT控制台的导航窗格中，展开 Test（测试）、Device Advisor，然后选择 Test run and results（测试运行和结果）。
2. 选择要停止的正在进行的测试套件。



3. 选择 Actions（操作），然后选择 Stop test suite（停止测试套件）。



4. 此清理过程可能需要几分钟才能完成。清理程序运行时，测试运行状态将为 STOPPING。等待清理程序完成，并等待测试套件状态更改为 STOPPED 状态，然后再开始运行新套件。



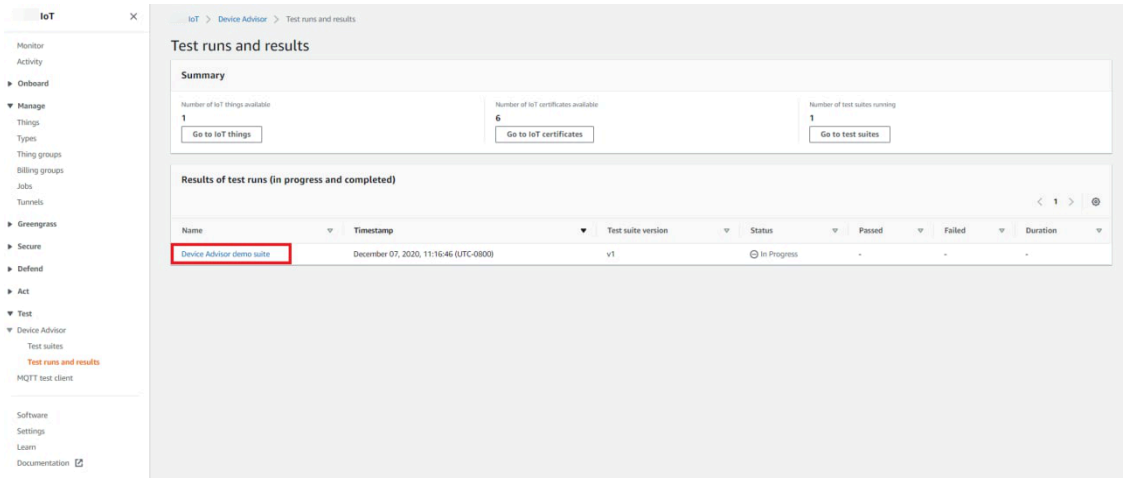
查看测试套件运行详细信息和日志

1. 在 Amazon IoT 控制台的导航窗格中，展开 Test（测试）、Device Advisor，然后选择 Test run and results（测试运行和结果）。

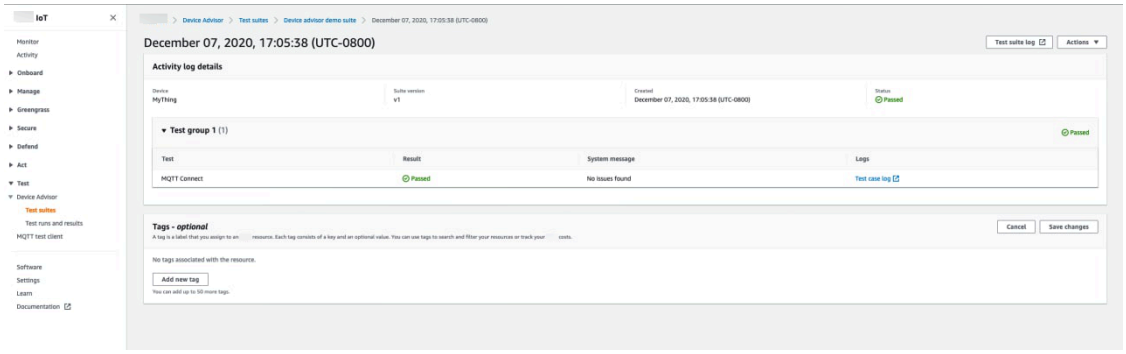
   本页面将显示：

   - IoT 事物的数量
   - IoT 证书的数量
   - 当前运行的测试套件数
   - 所有已创建的测试套件运行

2. 选择要查看其运行详细信息和日志的测试套件。



运行摘要页面显示当前测试套件运行的状态。此页面每 10 秒自动刷新一次。我们建议您为设备建立一个机制，以便每五秒便尝试一次连接到测试终端节点，每次持续一到两分钟。然后，您能够以自动的方式按顺序运行多个测试用例。

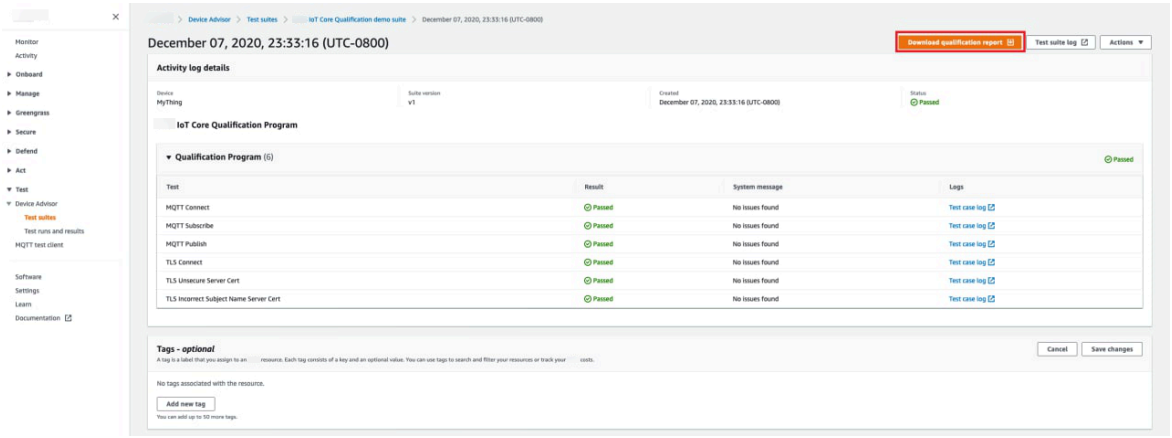


3. 要访问测试套件运行套件运行套件运行套件运行套件运行套件日志。 CloudWatch

   要访问任何测试用例 CloudWatch 的日志，请选择 Test case log（测试用例日志）
4. 根据您的测试结果，对您的设备进行故障排除操作，直到通过所有测试。

下载 Amazon IoT 资格报告

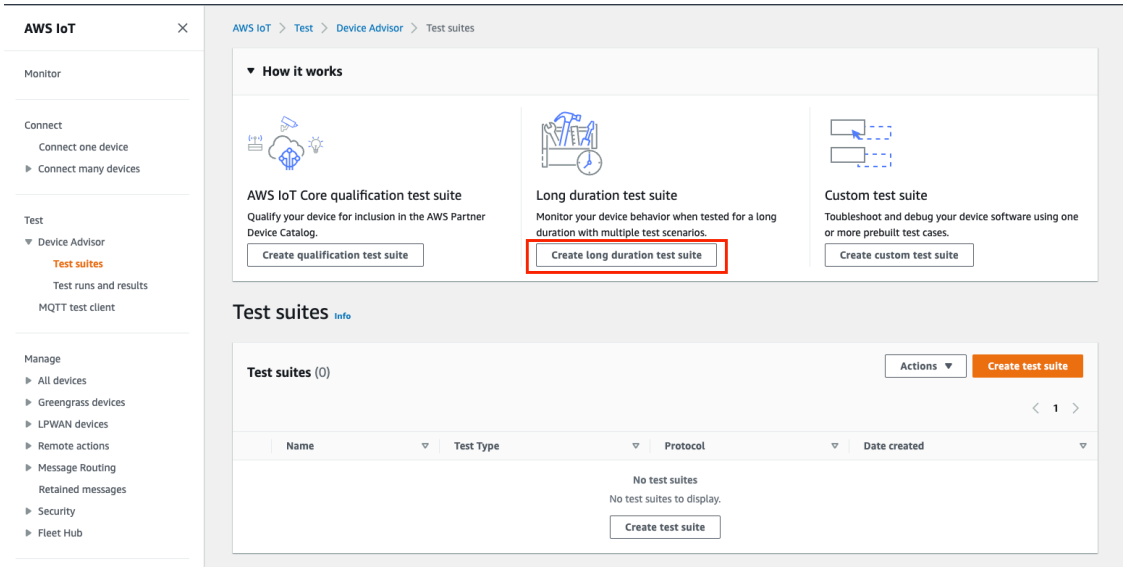
如果您在创建测试用例时选择了 Use the Amazon IoT Qualification test suite（使用 IoT 资格测试套件）选项，并能够运行资格测试套件，则可以在测试运行摘要页面选择 Download qualification report（下载资格报告）。



长时间测试控制台的工作流程

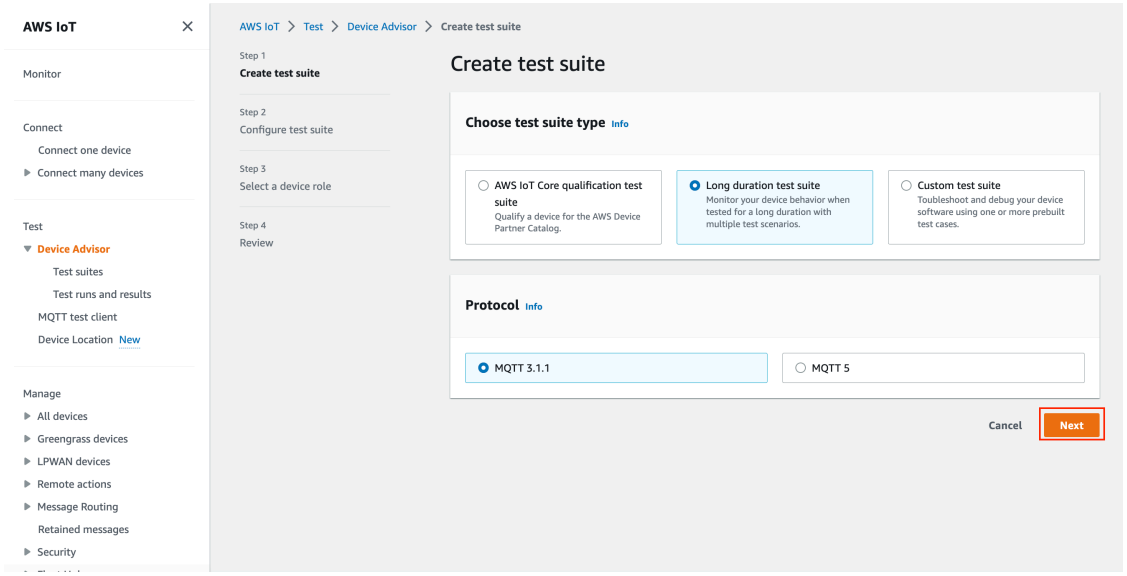
本教程可帮助您使用控制台开始在 Device Advisor 上进行长时间测试。要完成本教程，请执行设置 (p. 1016)中的步骤。

1. 在 Amazon IoT 控制台的导航窗格中，依次展开 Test（测试）、Device Advisor 和 Test suites（测试套件）。在此页面上，选择 Create long duration test suite（创建长时间测试套件）。



2. 在 Create test suite（创建测试套件）页面上，选择 Long duration test suite（长时间测试套件），然后选择 Next（下一步）。

   对于协议，选择 MQTT 3.1.1 或 MQTT 5。



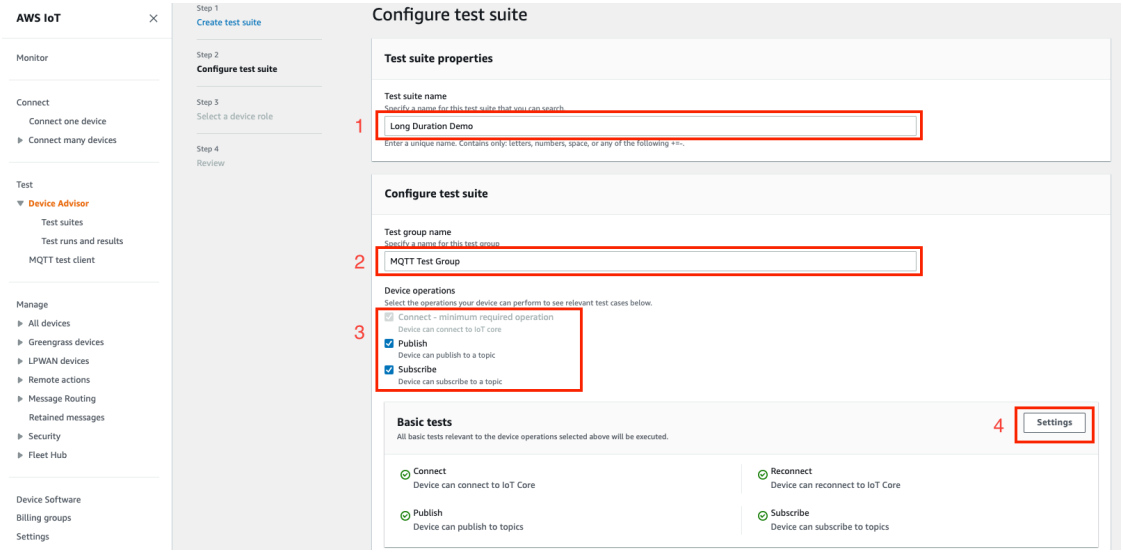
3. 在 Configure test suite（配置测试套件）页面上，执行以下操作：

   a. 更新 Test suite name（测试套件名称）字段。

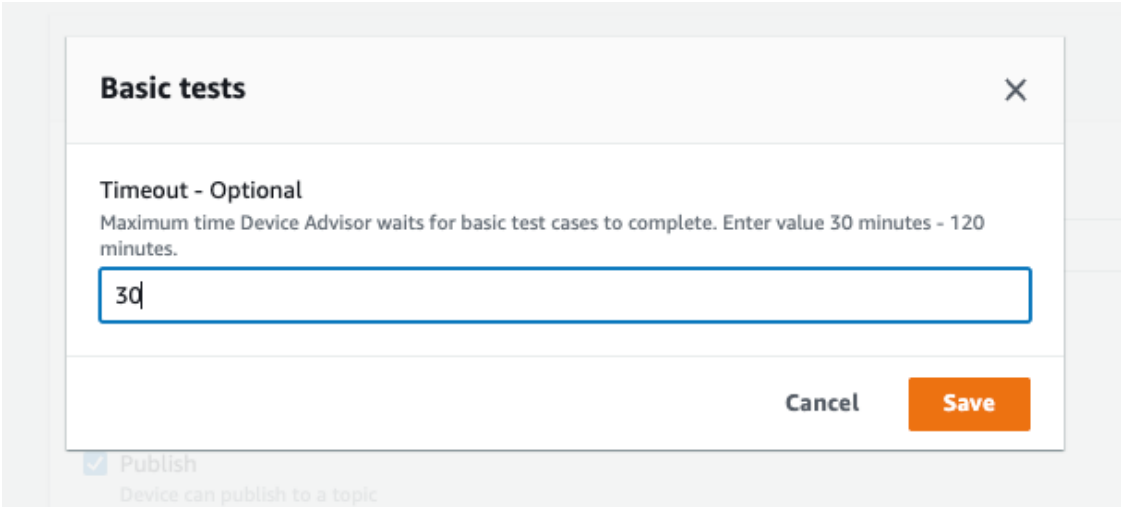
   b. 更新 Test group name（测试组名称）字段。

   c. 选择设备可以执行的 Device operations（设备操作）。这将选择要运行的测试。

   d. 选择 Settings（设置）选项。



4. （可选）输入 Device Advisor 必须等待基本测试完成的最长时间。选择 Save（保存）。



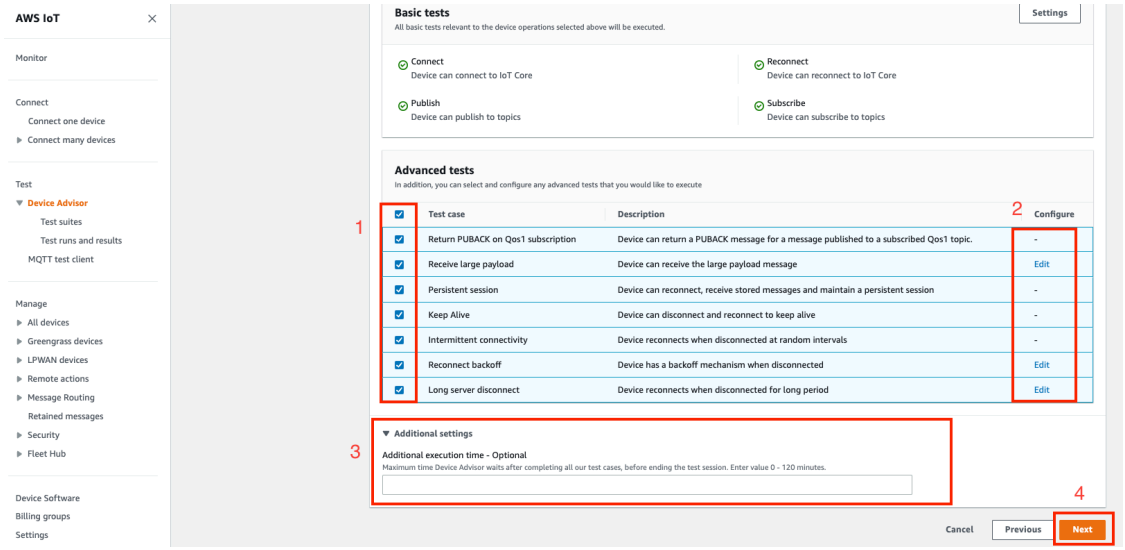
5. 在 Advanced tests（高级测试）和 Additional settings（其他设置）部分中执行以下操作。

   a. 选择或取消选择要在此测试中运行的 Advanced tests（高级测试）。

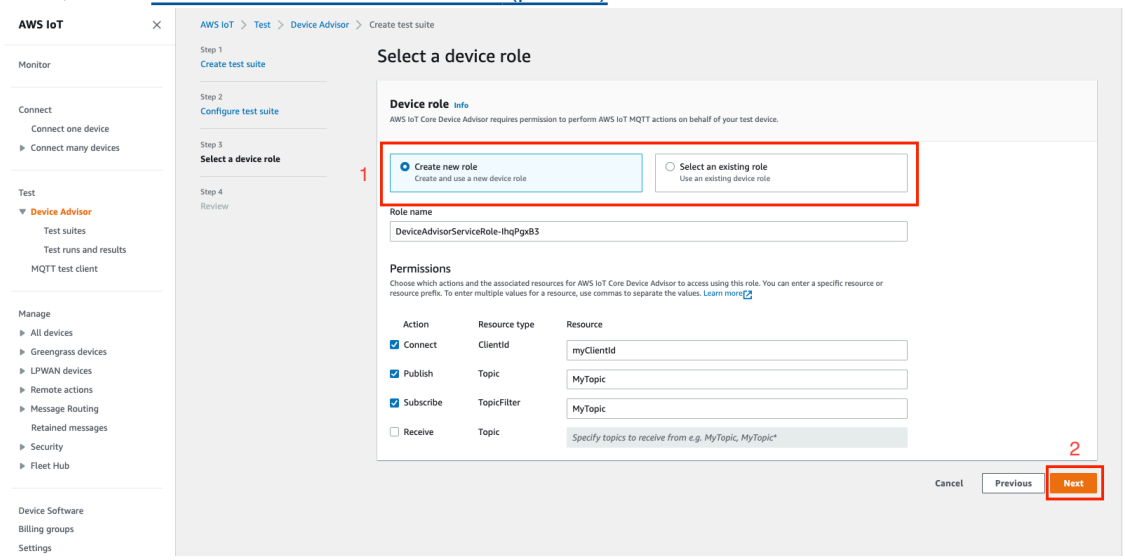
   b. 如果需要，Edit（编辑）测试的配置。

   c. 在 Additional settings（其他设置）部分下，配置 Additional execution time（其他执行时间）。

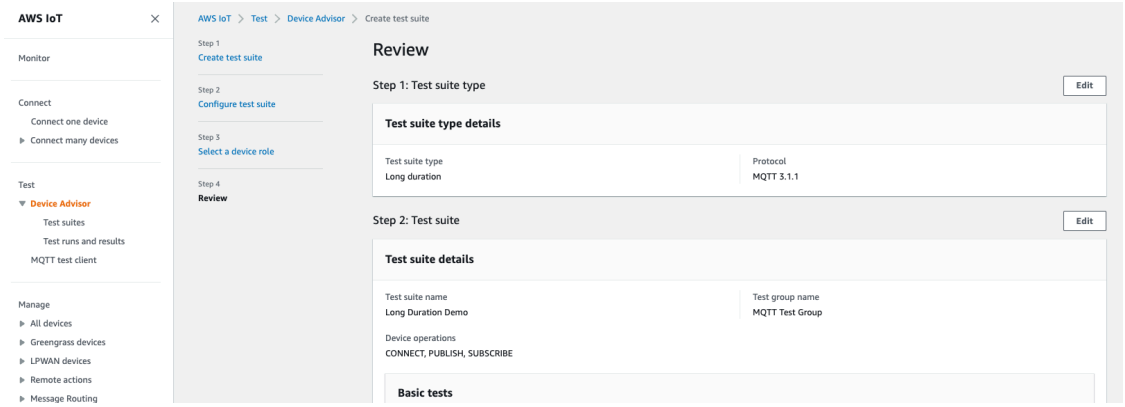
   d. 选择 Next（下一步）以继续执行下一步。

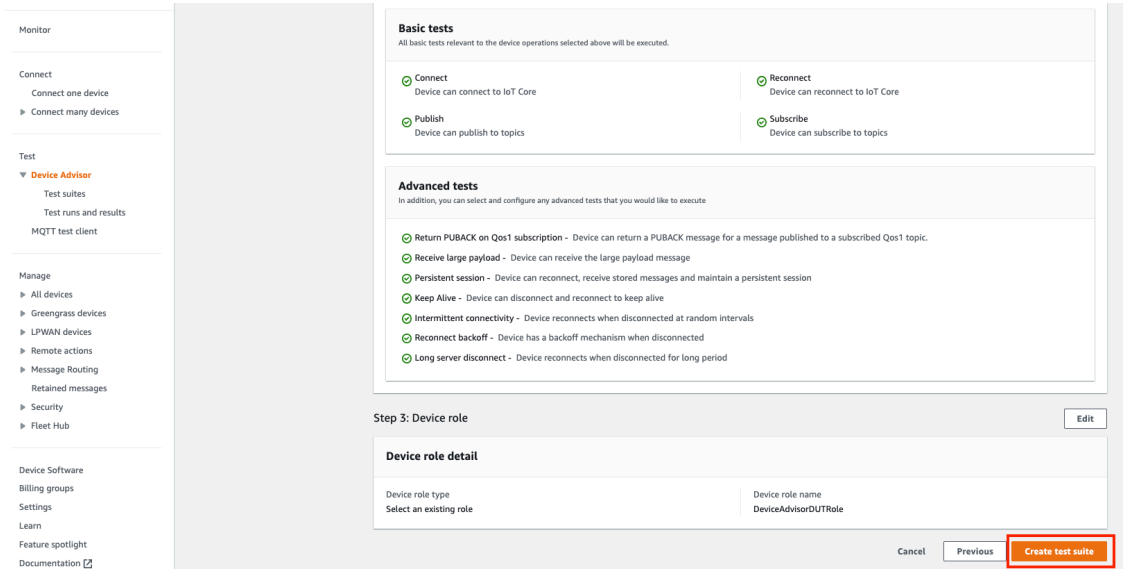


6. 在此步骤中，Create a new role（创建新角色）或 Select an existing role（选择现有角色）。有关详细信息，请参阅创建要用作设备角色的 IAM 角色 (p. 1017)。

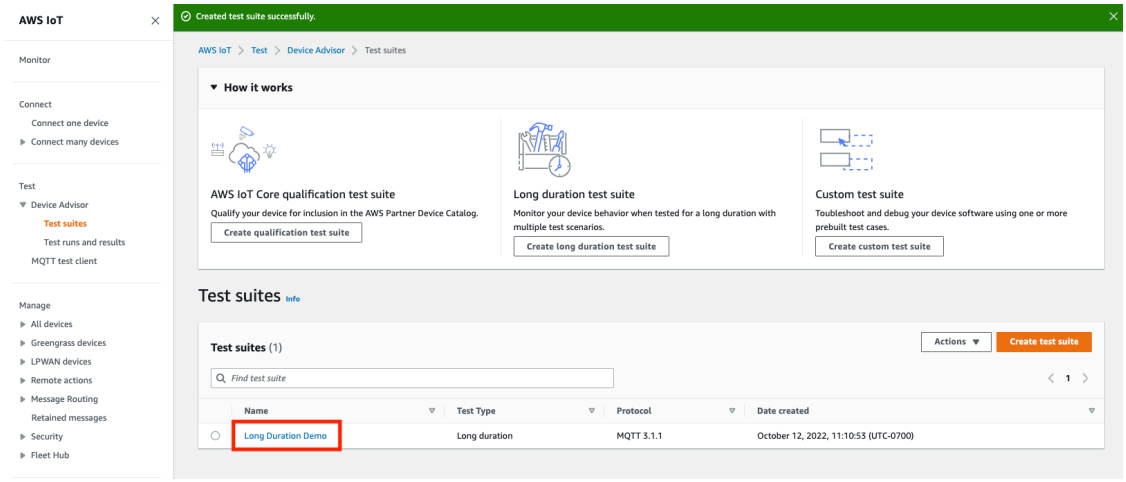


7. 查看在此步骤之前创建的所有配置，然后选择 Create test suite（创建测试套件）。

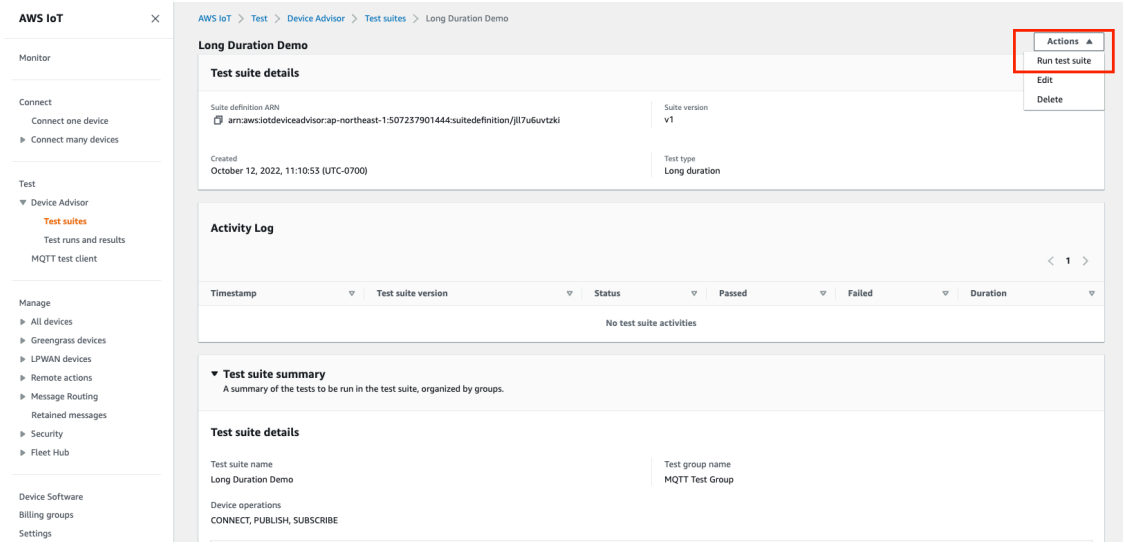




8. 创建的测试套件位于 Test suites（测试套件）部分下。选择此套件以查看详细信息。



9. 要运行创建的测试套件，请选择 Actions（操作），然后选择 Run test suite（运行测试套件）。

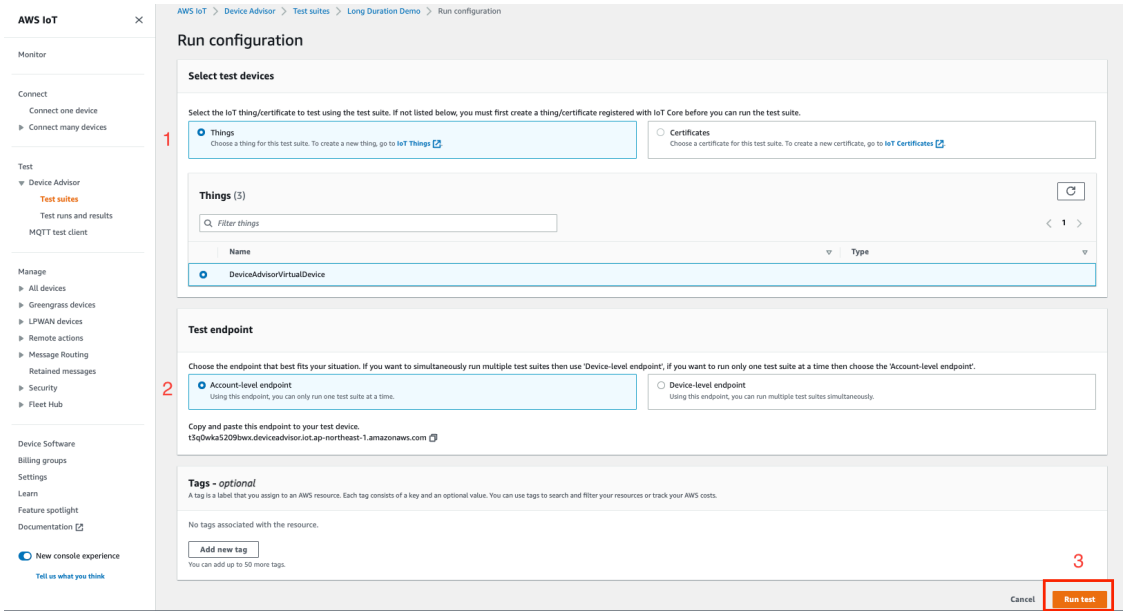


10. 在 Run configuration（运行配置）页面中选择配置选项。

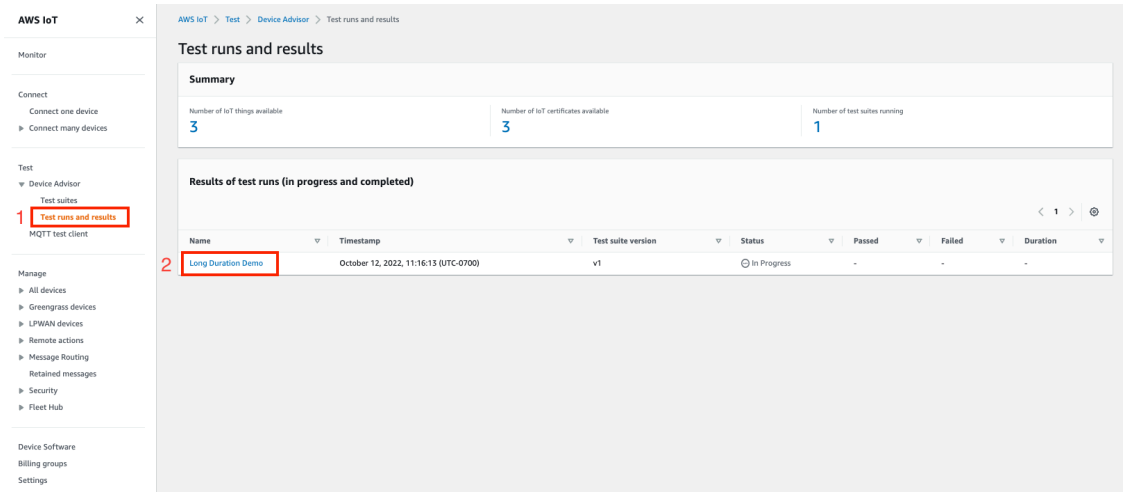
    a. 选择要运行测试的 Things（事物）或 Certificate（证书）。

    b. 选择 Account-level endpoint（账户级端点）或 Device-level endpoint（设备级端点）。

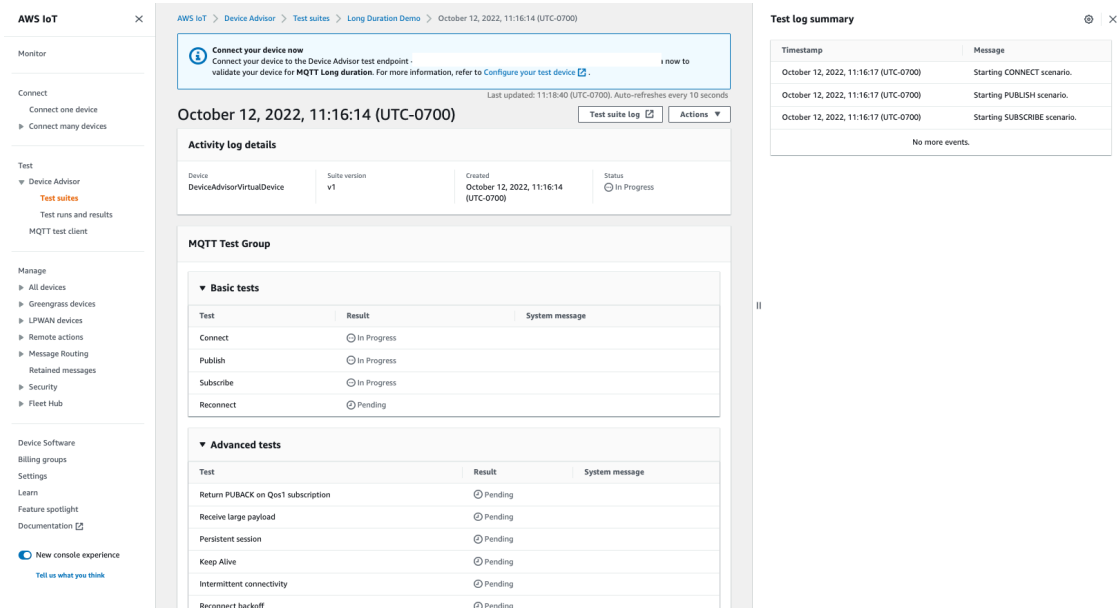
    c. 选择 Run test（运行测试）以运行此测试。



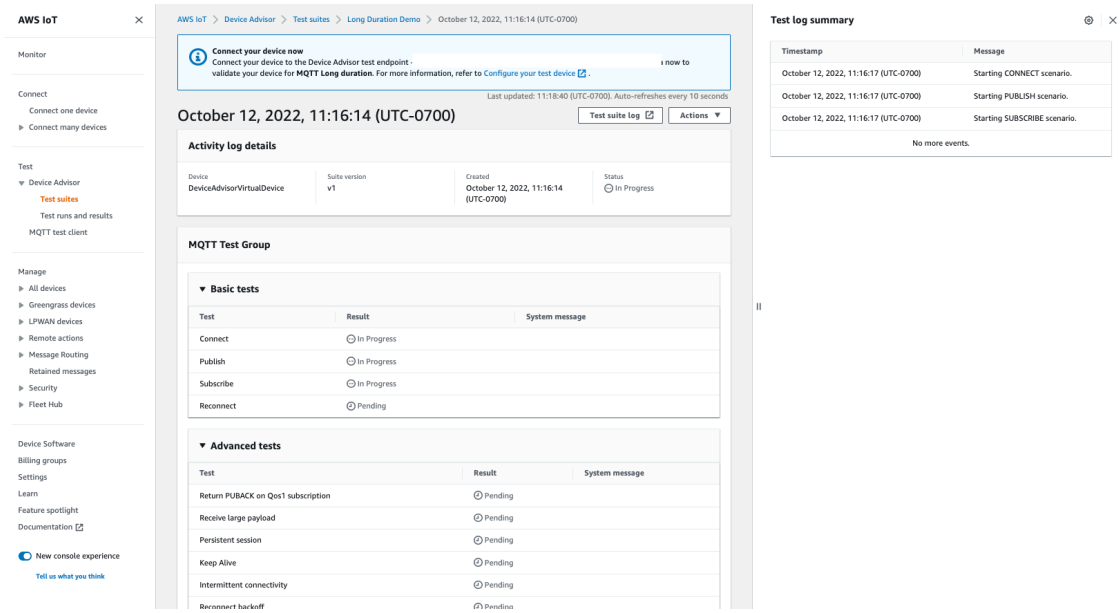
11. 要查看测试套件的运行结果，请在左侧导航窗格中选择 Test runs and results（测试运行和结果）。选择运行的测试套件以查看结果的详细信息。



12. 上一步将打开“test summary”（测试摘要）页面。测试运行的所有详细信息都显示在此页面中。当控制台提示启动设备连接时，将您的设备连接到提供的端点。在此页面上可以看到测试的进度。



13. 长时间测试在侧面板上提供了额外的 Test log summary（测试日志摘要），该摘要几乎实时显示设备和代理之间发生的所有重要事件。要查看更深入的详细日志，请单击 Test case log（测试用例日志）。



Device Advisor VPC 端点（Amazon PrivateLink）

您可以通过创建接口 VPC 端点 在 VPC 和 Amazon IoT Core Device Advisor 测试端点（数据面板）之间建立私有连接。您可以使用此端点验证 Amazon IoT 设备与 Amazon IoT Core 之间的连接是否安全和可靠，然后将设备部署到生产环境。Device Advisor 的预构建测试可帮助您根据最佳实践验证您的设备软件，以便使用 TLS、MQTT、设备影子和 Amazon IoT Jobs。

Amazon PrivateLink可为与您的 IoT 设备结合使用的接口端点提供支持。此服务可帮助您以私有方式访问 Amazon IoT Core Device Advisor 测试端点，而无需互联网网关、NAT 设备、VPN 连接或 Amazon Direct Connect 连接。VPC 中发送 TCP 和 MQTT 数据包的实例不需要公有 IP 地址来与 Amazon IoT Core Device

Advisor 测试端点进行通信。您的 VPC 和 Amazon IoT Core Device Advisor 之间的流量不会脱离 Amazon Web Services 云。IoT 设备和 Device Advisor 测试案例之间的任何 TLS 和 MQTT 通信均在您 Amazon Web Services 账户中的资源范围内。

每个接口端点均由子网中的一个或多个弹性网络接口表示。

要了解有关使用接口 VPC 端点的更多信息，请参阅《Amazon VPC 用户指南》中的接口 VPC 端点（Amazon PrivateLink）。

Amazon IoT Core Device Advisor VPC 终端节点注意事项

在设置接口 VPC 端点之前，请先查看《Amazon VPC 用户指南》中的接口端点属性和限制。继续之前，请注意以下各项：

- Amazon IoT Core Device Advisor 目前支持从您的 VPC 调用 Device Advisor 测试端点（数据面板）。消息代理使用数据面板通信发送和接收数据。这是在 TLS 和 MQTT 数据包的帮助下实现的。Amazon IoT Core Device Advisor 的 VPC 端点将您的 Amazon IoT 设备连接到 Device Advisor 测试端点。此 VPC 端点不使用控制面板 API 操作。为了创建或运行测试套件或其他控制面板 API，请通过公共互联网使用控制台、Amazon SDK 或 Amazon 命令行接口。
- 以下 Amazon Web Services 区域对于 Amazon IoT Core Device Advisor 支持 VPC 端点：
  - 美国东部（弗吉尼亚州北部）
  - 美国西部（俄勒冈州）
  - 亚太地区（东京）
  - 欧洲（爱尔兰）
- Device Advisor 支持使用 X.509 客户端证书和 RSA 服务器证书的 MQTT。
- 目前不支持 VPC 端点策略。
- 查看 VPC 端点先决条件，了解有关如何创建连接 VPC 端点的资源的说明。您必须创建 VPC 和私有子网才能使用 Amazon IoT Core Device Advisor VPC 端点。
- 您的 Amazon PrivateLink 资源上有配额。有关更多信息，请参阅 Amazon PrivateLink 配额。
- VPC 端点仅支持 IPv4 流量。

为 Amazon IoT Core Device Advisor 创建接口 VPC 终端节点

要开始使用 VPC 端点，请创建接口 VPC 端点。然后，选择 Amazon IoT Core Device Advisor 作为 Amazon Web Service。如果您使用Amazon IoT Core Device Advisor的是Amazon CLI，请describe-vpc-endpoint-services致电确认您的可用区中是否存在Amazon Web Services 区域。确认连接到端点的安全组允许对 MQTT 和 TLS 流量使用 TCP 协议通信。例如，在美国东部（弗吉尼亚州北部）区域中，使用以下命令：

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-
east-1.deviceadvisor.iot
```

注意，您可以使用以下服务名称为 Amazon IoT Core 创建 VPC 端点：

- com.amazonaws.region.deviceadvisor.iot

原定设置情况下，对于此端点开启私有 DNS。这确保原定设置测试端点的使用保持在您的私有子网内。要获取您的账户或设备级端点，请使用控制台、Amazon CLI 或 Amazon SDK。例如，如果您在公有子网或公共互联网上运行 get-endpoint，则可以获取您的端点并使用它来连接到 Device Advisor。有关更多信息，请参阅 Amazon VPC 用户指南中的通过接口端点访问服务。

要将 MQTT 客户端连接到 VPC 端点接口，Amazon PrivateLink 服务将在附加到 VPC 的私有托管区中创建 DNS 记录。这些 DNS 记录将 Amazon IoT 设备的请求定向到 VPC 端点。

通过 VPC 端点控制对 Amazon IoT Core Device Advisor 的访问

您可以将设备访问限制为 Amazon IoT Core Device Advisor，并仅允许使用 VPC 条件上下文键通过 VPC 端点进行访问。Amazon IoT Core 支持以下 VPC 相关的上下文键：

- SourceVpc
- SourceVpce
- VPCSourceIp

> **Note**
>
> Amazon IoT Core Device Advisor 目前不支持 VPC 端点策略。

以下策略授予使用与事物名称匹配的客户端 ID 连接到 Amazon IoT Core Device Advisor 的权限。它还会发布到任何以事物名称为前缀的主题。该策略以设备连接到具有特定 VPC 端点 ID 的 VPC 端点为条件。此策略拒绝连接到您的公有 Amazon IoT Core Device Advisor 测试端点的尝试。

```
{
"Version": "2012-10-17",
    "Statement": [
        {
"Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:client/
${iot:Connection.Thing.ThingName}"
            ],
            "Condition": {
"StringEquals": {
"aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }

        },
        {
"Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:us-east-1:123456789012:topic/
${iot:Connection.Thing.ThingName}/*"
            ]
        }
    ]
}
```

Device Advisor 测试用例

Device Advisor 提供五个类别的预构建测试。

- TLS (p. 1049)
- MQTT (p. 1053)
- 影子 (p. 1061)
- 任务执行 (p. 1062)
- 权限与策略 (p. 1064)
- 长时间测试 (p. 1064)

用于测试是否符合 Amazon 设备资格认证计划资格的 Device Advisor 测试案例。

根据 Amazon 设备资格认证计划，您的设备需要通过以下测试才符合资格。

- TLS Incorrect Subject Name Server Cert (p. 1052) (TLS 不正确的主题名称服务器证书) ("Incorrect Subject Common Name (CN) / Subject Alternative Name (SAN)") (“不正确的主题通用名称 (CN)/主题备用名称 (SAN)”)
- TLS Unsecure Server Cert (p. 1051) (TLS 不安全的服务器证书) ("Not Signed By Recognized CA") (“未被认可的 CA 签名”)
- TLS Connect (p. 1049) (“TLS Connect”)
- MQTT Connect (p. 1053) ("Device send CONNECT to Amazon IoT Core(Happy case)") (“设备将 CONNECT 发送到 IoT Core （ 快乐用例 ） ”)
- MQTT 订阅 (p. 1058) ("Can Subscribe (Happy Case)") (“可以订阅 (快乐用例) ”)
- MQTT 发布 (p. 1056) ("QoS0 (Happy Case)") (“QoS0 (快乐用例) ”)

TLS

可以使用这些测试来确定设备和 Amazon IoT 之间的传输层安全协议 (TLS) 是否安全。

满意路径

“TLS Connect”

验证被测设备是否可以完成与 Amazon IoT 的 TLS 握手。此测试不会验证客户端设备的 MQTT 执行。

API 测试用例定义：

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。为了获得最佳效果，我们建议超时值为 2 分钟。

```
"tests":[
   {
      "name":"my_tls_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Connect",
         "version":"0.0.0"
      }
   }
]
```

Example 测试用例输出：

- 通过 — 被测设备完成了与 Amazon IoT 的 TLS 握手。
- 通过但带有警告 — 被测设备完成了与 Amazon IoT 的 TLS 握手，但是存在来自设备或 Amazon IoT 的 TLS 警告消息。
- 失败 — 由于握手错误，被测设备无法完成与 Amazon IoT 的 TLS 握手。

“TLS 接收最大大小片段”

此测试用例可帮助您验证设备是否能够接收和处理 TLS 最大大小片段。您的测试设备需要使用 QoS 1 订阅预配置的主题，才能接收大量有效负载。您可以使用配置 ${payload} 自定义有效负载。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。为了获得最佳效果，我们建议超时值为 2 分钟。

```
"tests":[
   {
      "name":"TLS Receive Maximum Size Fragments",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
         "PAYLOAD_FORMAT":"{"message":"${payload}"}", // A string with a placeholder
 ${payload}, or leave it empty to receive a plain string.
         "TRIGGER_TOPIC": "test_1" // A topic to which a device will subscribe, and to
 which a test case will publish a large payload.
      },
      "test":{
         "id":"TLS_Receive_Maximum_Size_Fragments",
         "version":"0.0.0"
      }
   }
]
```

密码套件

“适用于 Amazon IoT 推荐的密码套件的 TLS 设备支持”

验证来自被测设备的 TLS Client Hello 消息中的密码套件是否包含 Amazon IoT 推荐的密码套件 (p. 346)。它提供了有关设备支持的密码套件的更多洞察。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_tls_support_aws_iot_cipher_suites_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Support_AWS_IoT_Cipher_Suites",
         "version":"0.0.0"
      }
   }
```

```
]
```

Example 测试用例输出：

- 通过 — 被测设备的密码套件至少包含一个 Amazon IoT 推荐的密码套件，并且不包含任何不受支持的密码套件。
- 通过但带有警告 —设备密码套件至少包含一个 Amazon IoT 密码套件，但 1) 不包含任何推荐的密码套件，或 2) 包含 Amazon IoT 不支持的密码套件。我们建议验证不受支持的密码套件是否安全。
- 失败 — 被测设备密码套件不包含任何由 Amazon IoT 支持的密码套件。

较大的服务器证书

“TLS 较大服务器证书”

验证您的设备是否可以通过接收和处理较大的服务器证书来完成与 Amazon IoT 的 TLS 握手。此测试使用的服务器证书的大小（以字节为单位）比“TLS Connect”测试用例和 IoT Core 中当前使用的证书大小大 20%。在此测试用例中，将测试设备的 TLS 缓冲区空间。如果缓冲区空间足够大，TLS 握手将完成而不会出现错误。此测试不验证设备的 MQTT 实施。测试用例在 TLS 握手过程完成后终止。

API 测试用例定义：

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。为了获得最佳效果，我们建议超时值为 2 分钟。如果此测试用例失败但“TLS Connect”测试用例通过，我们建议您将设备的 TLS 缓冲区空间限制提高到更高的字节数。提高缓冲区空间限制将确保设备能够在服务器证书大小增加的情况下处理较大的服务器证书。

```
"tests":[
   {
      "name":"my_tls_large_size_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"TLS_Large_Size_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example 测试用例输出：

- 通过 — 被测设备完成了与 Amazon IoT 的 TLS 握手。
- 通过但带有警告 — 被测设备完成了与 Amazon IoT 的 TLS 握手，但是存在来自设备或 Amazon IoT 的 TLS 警告消息。
- 失败 — 由于在握手过程中出现错误，被测设备未能完成与 Amazon IoT 的 TLS 握手。

TLS 不安全服务器证书

“未被认可的 CA 签名”

验证如果提供不具有 ATS CA 有效签名的服务器证书，被测设备是否会关闭连接。设备只能连接到提供有效证书的终端节点。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_tls_unsecure_server_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Unsecure_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example 测试用例输出：

- 通过 — 被测设备关闭了连接。
- 失败 — 被测设备完成了与 Amazon IoT 的 TLS 握手。

“TLS 不正确的主题名称服务器证书/不正确的主题通用名称 (CN, Common Name)/主题备用名称 (SAN, Subject Alternative Name)”

验证如果提供与请求的域名不同的域名服务器证书，被测设备是否会关闭连接。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_tls_incorrect_subject_name_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"TLS_Incorrect_Subject_Name_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example 测试用例输出：

- 通过 — 被测设备关闭了连接。
- 失败 — 被测设备完成了与 Amazon IoT 的 TLS 握手。

TLS 已过期的服务器证书

“已过期的服务器证书”

验证如果被测试的设备的服务器证书已过期，该设备是否关闭连接。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_tls_expired_cert_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",  //in seconds
      },
      "test":{
         "id":"TLS_Expired_Server_Cert",
         "version":"0.0.0"
      }
   }
]
```

Example 测试用例输出：

- 通过 — 在关闭连接之前，被测设备未能完成与 Amazon IoT 的 TLS 握手，但发送了 TLS 警报消息。
- 通过但带有警告 — 在关闭连接之前，被测设备未能完成与 Amazon IoT 的 TLS 握手，也未能发送 TLS 警报消息。
- 失败 — 被测设备完成了与 Amazon IoT 的 TLS 握手。

MQTT

CONNECT、DISCONNECT 和 RECONNECT

"Device send CONNECT to Amazon IoT Core (Happy case)" (“设备将 CONNECT 发送到 IoT Core (快乐用例) ”)

验证被测设备是否发送 CONNECT 请求。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_mqtt_connect_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",   // in seconds
      },
      "test":{
         "id":"MQTT_Connect",
         "version":"0.0.0"
      }
   }
]
```

“设备可以将 PUBACK 返回到 QoS1 的任意主题”

如果客户端在使用 QoS1 订阅到主题后收到来自代理的发布消息，则此测试用例将检查设备 (客户端) 是否能够返回 PUBACK 消息。

可针对此测试用例配置负载内容和负载大小。如果配置了负载大小，Device Advisor 将覆盖负载内容的值，并将预定义的具有所需大小的负载发送到设备。负载大小的值介于 0 到 128 之间，不能超过 128KB。Amazon IoT Core 拒绝大于 128KB 的发布和连接请求，如 Amazon IoT Core 消息代理及协议限制和限额页面所示。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议超时值为 2 分钟。PAYLOAD_SIZE 可以配置为 0 到 128KB 之间的值。定义负载大小会覆盖负载内容，因为 Device Advisor 会将具有给定大小的预定义负载发送回设备。

```
"tests":[
{
        "name":"my_mqtt_client_puback_qos1",
        "configuration": {
            // optional:"TRIGGER_TOPIC": "myTopic",
            "EXECUTION_TIMEOUT":"300", // in seconds
            "PAYLOAD_FOR_PUBLISH_VALIDATION":"custom payload",
            "PAYLOAD_SIZE":"100" // in kilobytes
        },
        "test": {
            "id": "MQTT_Client_Puback_QoS1",
            "version": "0.0.0"
        }
    }
]
```

"Device re-connect with jitter backoff - No CONNACK response" (“设备以抖动退避功能重新连接 - 没有 CONNACK 响应”)

验证被测设备与代理重新连接至少五次时是否使用了正确的抖动退避。代理记录被测设备的 CONNECT 请求时间戳，执行数据包验证，暂停而不向被测设备发送 CONNACK，并等待被测设备重新发送请求。允许第六次连接尝试通过，并允许 CONNACK 流回到被测设备。

将再次执行上述过程。总体而言，此测试用例要求设备总共至少连接 12 次。收集的时间戳用于验证被测设备是否使用了抖动退避。如果被测设备具有严格的指数退避延迟，则此测试用例将通过但带有警告。

我们建议实施指数退避和抖动在所测试设备上的机制通过此测试案例。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 4 分钟。

```
"tests":[
   {
      "name":"my_mqtt_jitter_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300",    // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Jitter_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

“设备使用指数退避重新连接 - 没有 CONNACK 响应”

验证被测设备与代理重新连接至少五次时是否使用了正确的指数退避。代理记录被测设备的 CONNECT 请求时间戳，执行数据包验证，暂停而不向客户端设备发送 CONNACK，并等待被测设备重新发送请求。收集的时间戳用于验证被测设备是否使用了指数退避。

我们建议实施指数退避和抖动在所测试设备上的机制通过此测试案例。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 4 分钟。

```
"tests":[
   {
      "name":"my_mqtt_exponential_backoff_retries_test",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"600",  // in seconds
      },
      "test":{
         "id":"MQTT_Connect_Exponential_Backoff_Retries",
         "version":"0.0.0"
      }
   }
]
```

“使用抖动退避重新连接设备-服务器断开连接后”

验证正在测试的设备在与服务器断开连接后重新连接时是否使用必要的抖动和回退。Device Advisor 至少将设备与服务器断开五次，并观察设备的 MQTT 重新连接的行为。Device Advisor 记录被测设备的连接请求的时间戳，执行数据包验证，暂停而不向客户端设备发送连接，并等待被测设备重新发送请求。收集的时间戳用于验证被测设备在重新连接时是否使用抖动和退避。如果被测设备具有严格的指数退避或未实现适当的抖动退避机制，此测试用例将通过但是带有警告。如果被测设备实施了线性退避或恒定退避机制，测试将失败。

要通过此测试用例，我们建议在所测试设备上实施指数退避和抖动机制。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 4 分钟。
通过指定 RECONNECTION_ATTEMPTS，可以更改验证退避的重新连接尝试次数。该数字必须介于 5 到 10 之间。默认值为 5。

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_server_disconnect",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Server_Disconnect",
         "version":"0.0.0"
      }
   }
```

```
]
```

"Device re-connect with jitter backoff - On unstable connection" (“设备以抖动退避功能重新连接 - 连接不稳定”)

验证被测设备在不稳定连接上重新连接时是否使用了必要的抖动和退避。Device Advisor 在五次成功连接后将设备与服务器断开连接，并观察设备的 MQTT 重新连接的行为。Device Advisor 记录被测设备的 CONNECT (连接) 请求的时间戳，执行数据包验证，发送回 CONNACK，断开连接，记录断开连接的时间戳，并等待被测设备重新发送请求。收集的时间戳用于验证被测设备在成功但不稳定的连接后重新连接时，是否使用抖动和退避。如果被测设备具有严格的指数退避或未实现适当的抖动退避机制，此测试用例将通过但是带有警告。如果被测设备实施了线性退避或恒定退避机制，测试将失败。

要通过此测试用例，我们建议在所测试设备上实施指数退避和抖动机制。

API 测试用例定义：

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 4 分钟。
> 通过指定 RECONNECTION_ATTEMPTS，可以更改验证退避的重新连接尝试次数。该数字必须介于 5 到 10 之间。默认值为 5。

```
"tests":[
   {
      "name":"my_mqtt_reconnect_backoff_retries_on_unstable_connection",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "RECONNECTION_ATTEMPTS": 5
      },
      "test":{
         "id":"MQTT_Reconnect_Backoff_Retries_On_Unstable_Connection",
         "version":"0.0.0"
      }
   }
]
```

发布

"QoS0 (Happy Case)" (“QoS0 (快乐用例) ”)

验证被测设备是否使用 QoS0 和 QoS1 发布消息。您还可以通过在测试设置中指定此主题值和有效负载来验证消息的主题和有效负载。

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_mqtt_publish_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish",
```

```
         "version":"0.0.0"
      }
   }
]
```

"QoS1 publish retry - No PUBACK" (“QoS1 发布重试 - 没有 PUBACK”)

如果代理没有发送 PUBACK，则验证被测设备是否重新发布了随 QoS1 发送的消息。您还可以通过在测试设置中指定此主题来验证消息的主题。在重新发布消息之前，客户端设备不得断开连接。此测试还验证重新发布的消息与原始消息是否具有相同的数据包标识符。在测试执行期间，如果设备丢失连接并重新连接，则测试用例将顺利重置，并且设备必须重新执行测试用例步骤。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。建议至少 4 分钟。

```
"tests":[
   {
      "name":"my_mqtt_publish_retry_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_FOR_PUBLISH_VALIDATION": "my_TOPIC_FOR_PUBLISH_VALIDATION",
         "PAYLOAD_FOR_PUBLISH_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retry_No_Puback",
         "version":"0.0.0"
      }
   }
]
```

"Publish Retained messages" (“发布保留消息”)

验证被测设备是否发布 `retainFlag` 设置为 true 的消息。您可以通过在测试设置中设置主题值和有效负载来验证消息的主题和有效负载。如果未将 PUBLISH 数据包中发送的 `retainFlag` 设置为 true，则测试用例将失败。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。要运行此测试用例，请在您的 device role (设备角色) 中添加 `iot:RetainPublish` 操作。

```
"tests":[
   {
      "name":"my_mqtt_publish_retained_messages_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds

 "TOPIC_FOR_PUBLISH_RETAINED_VALIDATION": "my_TOPIC_FOR_PUBLISH_RETAINED_VALIDATION",

 "PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION": "my_PAYLOAD_FOR_PUBLISH_RETAINED_VALIDATION",
      },
      "test":{
         "id":"MQTT_Publish_Retained_Messages",
         "version":"0.0.0"
```

```
      }
   }
]
```

"Publish with User Property" (“使用用户属性发布”)

验证被测设备是否使用正确的用户属性发布消息。您可以通过在测试设置中设置名称-值对来验证用户属性。如果未提供用户属性或用户属性不匹配，则测试用例将失败。

API 测试用例定义：

Note

这是仅限 MQTT5 的测试用例。
EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_mqtt_user_property_test",
      "test":{
        "USER_PROPERTIES": [
            {"name": "name1", "value":"value1"},
            {"name": "name2", "value":"value2"}
        ],
        "EXECUTION_TIMEOUT":"300",  // in seconds
      },
      "test":{
         "id":"MQTT_Publish_User_Property",
         "version":"0.0.0"
      }
   }
]
```

Subscribe

"Can Subscribe (Happy Case)" (“可以订阅（快乐用例）”)

验证被测设备是否订阅了 MQTT 主题。您还可以通过在测试设置中指定此主题来验证被测设备订阅的主题。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_mqtt_subscribe_test",
      "configuration":{
         // optional:
         "EXECUTION_TIMEOUT":"300",  // in seconds
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION_ID":
["my_TOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe",
         "version":"0.0.0"
      }
   }
```

```
]
```

"Subscribe Retry - No SUBACK" (“订阅重试 - 无 SUBACK”)

验证被测设备是否重试 MQTT 主题的失败订阅。然后，服务器会等待，不发送 SUBACK。如果客户端设备未重试订阅，则测试失败。客户端设备必须使用相同的数据包 ID 重试失败的订阅。您还可以通过在测试设置中指定此主题来验证被测设备订阅的主题。在测试执行期间，如果设备丢失连接并重新连接，则测试用例将顺利重置，并且设备必须重新执行测试用例步骤。

API 测试用例定义：

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 4 分钟。

```
"tests":[
   {
      "name":"my_mqtt_subscribe_retry_test",
      "configuration":{
         "EXECUTION_TIMEOUT":"300",  // in seconds
         // optional:
         "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION_ID":
["myTOPIC_FOR_PUBLISH_VALIDATION_a","my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
      },
      "test":{
         "id":"MQTT_Subscribe_Retry_No_Suback",
         "version":"0.0.0"
      }
   }
]
```

Keep-Alive

“Mqt No Ack PingResp”

此测试用例验证被测设备在未收到 ping 响应时是否断开连接。作为此测试用例的一部分，Device Advisor 会阻止从 Amazon IoT Core 发送的发布、订阅和 ping 请求响应。它还验证正在测试的设备是否断开了 MQTT 连接。

API 测试用例定义：

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议超时时间大于 keepAliveTime 值的 1.5 倍。

```
"tests":[
    {
       "name":"Mqtt No Ack PingResp",
       "configuration":
          //optional:
          "EXECUTION_TIMEOUT":"306",   // in seconds
       },
       "test":{
          "id":"MQTT_No_Ack_PingResp",
          "version":"0.0.0"
       }
    }
]
```

持久会话

“持久会话（快乐用例）”

此测试用例验证设备在与持久会话断开连接时的行为。该测试用例检查设备是否可以重新连接，恢复对其触发器主题的订阅而无需显式重新订阅，接收主题中存储的消息以及在持久会话期间按预期工作。当此测试用例通过时，它表示客户端设备能够按照预期方式与 Amazon IoT Core 代理保持持久会话。有关 Amazon IoT 持久会话的更多信息，请参阅使用 MQTT 持久会话。

在此测试用例中，客户端设备应该与 Amazon IoT Core 进行连接 [clean session（清理会话）标志设置为 false]，然后订阅一个触发器主题。成功订阅后，Amazon IoT Core Device Advisor 将断开设备的连接。当设备处于断开连接状态时，该主题中将存储 QoS 1 消息负载。然后，Device Advisor 将允许客户端设备与测试终端节点重新连接。此时，由于存在持久会话，客户端设备应该恢复其主题订阅而不发送任何其他 SUBSCRIBE 数据包，然后接收来自代理的 QoS 1 消息。重新连接后，如果客户端设备通过发送额外的 SUBSCRIBE 数据包再次订阅其主题触发器和/或客户端未能从触发器主题接收存储的消息，则测试用例将失败。

API 测试用例定义：

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为至少 4 分钟。在第一个连接中，客户端设备需要显式订阅之前没有订阅的 TRIGGER_TOPIC。要通过测试用例，客户端设备必须使用 QoS 1 成功订阅 TRIGGER_TOPIC。重新连接后，客户端设备应该了解存在活动的持久会话；因此它应该接受由触发器主题发送的已存储消息，然后对该特定消息返回 PUBACK。

```
"tests":[
   {
      "name":"my_mqtt_persistent_session_happy_case",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         // if Payload not provided, a string will be stored in the trigger topic to be
 sent back to the client device
         "PAYLOAD": "The message which should be received from AWS IoT Broker after re-
connecting to a persistent session from the specified trigger topic.",
         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Persistent_Session_Happy_Case",
         "version":"0.0.0"
      }
   }
]
```

“持久会话 - 会话到期”

此测试用例有助于验证断开连接的设备重新连接到过期的持久会话时的设备行为。会话过期后，我们希望设备通过显式发送新的 SUBSUBE 数据包来重新订阅之前订阅的主题。

在第一次连接期间，我们希望测试设备与 Amazon IoT 代理进行连接，因为其 CleanSession 标志设置为 false 以启动持久会话。然后，设备应该订阅触发器主题。接着，在成功订阅并启动持久会话后，Amazon IoT Core Device Advisor 断开设备连接。断开连接之后，Amazon IoT Core Device Advisor 允许测试设备与测试终端节点重新连接。此时，当测试设备发送另一个 CONNECT 数据包时，Amazon IoT Core Device Advisor 会发回一个 CONNACK 数据包，指示持久会话已过期。测试设备需要正确地解释此数据包，并且在持久会话终止时，它应该会再次重新订阅同一触发器主题。如果测试设备不再重新订阅其主题触发器，测试用例将失败。要通过测试，设备需要了解持久会话已经结束，然后为第二个连接中的相同触发器主题发回一个新的 SUBSCRIBE 数据包。

如果测试设备的此测试用例获得通过，则表示该设备能够按照预期方式在持久会话到期时进行重新连接。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为至少 4 分钟。测试设备需要显式订阅之前没有订阅的 TRIGGER_TOPIC。要通过测试用例，测试设备必须发送 CONNECT 数据包（CleanSession 标志设置为 false），并使用 QoS 1 成功订阅触发器主题。成功订阅后，Amazon IoT Core Device Advisor 会断开设备连接。断开连接之后，Amazon IoT Core Device Advisor 允许设备重新连接，预计该设备将重新订阅同一个 TRIGGER_TOPIC，因为 Amazon IoT Core Device Advisor 已终止持久会话。

```
"tests":[
   {
      "name":"my_expired_persistent_session_test",
      "configuration":{
         //required:
         "TRIGGER_TOPIC": "myTrigger/topic",
         // optional:
         "EXECUTION_TIMEOUT":"300" // in seconds
      },
      "test":{
         "id":"MQTT_Expired_Persistent_Session",
         "version":"0.0.0"
      }
   }
]
```

影子

使用这些测试验证您的被测设备是否正确使用了 Amazon IoT 设备影子服务。参阅 Amazon IoT Device Shadow 服务 (p. 593) 了解更多信息。如果在测试套件中配置了这些测试用例，则在启动套件运行时需要提供一个事物。

发布

“设备连接后发布状态（快乐用例）”

验证设备是否可以在连接到 Amazon IoT Core 之后发布其状态。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_shadow_publish_reported_state",
      "configuration": {
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME",
         "REPORTED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
         }
      },
```

```
      "test":{
         "id":"Shadow_Publish_Reported_State",
         "version":"0.0.0"
      }
   }
]
```

REPORTED_STATE 可以在设备连接后对其确切影子状态进行额外验证。预设情况下，此测试用例会验证您的设备发布状态。

如果未提供 *SHADOW_NAME*，则测试用例将默认查找发布到 Unnamed（经典）影子类型的主题前缀的消息。如果您的设备使用命名的影子类型，请提供影子名称。请参阅在设备中使用影子，了解更多信息。

更新

“设备更新报告状态为理想状态（快乐用例）”

验证设备是否读取所有收到的更新消息，并同步设备的状态以与所需的状态属性匹配。您的设备应在同步后发布其最新报告状态。如果您的设备在运行测试之前已存在影子，请确保为测试用例配置的所需状态与现有报告状态不匹配。您可以通过查看 Shadow 文档中的ClientToken字段来识别由 Device Advisor 发送的影子更新消息DeviceAdvisorShadowTestCaseSetup。

API 测试用例定义：

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 2 分钟。

```
"tests":[
   {
      "name":"my_shadow_update_reported_state",
      "configuration": {
         "DESIRED_STATE": {
            "STATE_ATTRIBUTE": "STATE_VALUE"
         },
         // optional:
         "EXECUTION_TIMEOUT":"300", // in seconds
         "SHADOW_NAME": "SHADOW_NAME"
      },
      "test":{
         "id":"Shadow_Update_Reported_State",
         "version":"0.0.0"
      }
   }
]
```

DESIRED_STATE 应具有至少一个属性和关联的值。

如果未提供 SHADOW_NAME，则测试用例将默认查找发布到 Unnamed（经典）影子类型的主题前缀的消息。如果您的设备使用命名的影子类型，请提供影子名称。请参阅在设备中使用影子，了解更多信息。

任务执行

"Device can complete a job execution"（“设备可以完成任务执行”）

此测试用例可帮助您验证设备是否能够使用 Amazon IoT Jobs 接收更新，并发布成功更新的状态。有关 Amazon IoT Jobs 的更多信息，请参阅任务。

为了成功运行此测试用例，提供了您授予设备角色所需的两个保留 Amazon 主题。要订阅与消息相关的作业活动，请使用 notify 和 notify-next 主题。您的设备角色必须为以下主题授予 PUBLISH 操作：

- $aws/things/thingName/jobs/jobId/get
- $aws/things/thingName/jobs/jobId/update

建议为以下主题授予 SUBSCRIBE 和 RECEIVE 操作：

- $aws/things/thingName/jobs/get/accepted
- $aws/things/thingName/jobs/jobId/get/rejected
- $aws/things/thingName/jobs/jobId/update/accepted
- $aws/things/thingName/jobs/jobId/update/rejected

建议为以下主题授予 SUBSCRIBE 操作：

- $aws/things/thingName/jobs/notify-next

有关这些保留主题的更多信息，请参阅 Amazon IoT 作业的保留主题。

API 测试用例定义：

> **Note**
>
> EXECUTION_TIMEOUT 的默认值为 5 分钟。我们建议将超时值设置为 3 分钟。根据提供的 Amazon IoT 作业文档或源，调整超时值（例如，如果作业需要很长时间才能运行，则为测试用例定义一个更大的超时值）。要运行测试，需要提供有效的 Amazon IoT 作业文档或现有的作业 ID。可采用 JSON 文档或 S3 链接形式提供 Amazon IoT 作业文档。如果已提供作业文档，则可以选择提供作业 ID。如果已提供作业 ID，Device Advisor 将在代表您创建 Amazon IoT 作业时使用该 ID。如果未提供作业文档，您可以提供用于运行测试用例的同一区域中的现有 ID。在此情况下，Device Advisor 将在运行测试用例时使用该 Amazon IoT 作业。

```
"tests": [
   {
      "name":"my_job_execution",
      "configuration": {
         // optional:
         // Test case will create a job task by using either JOB_DOCUMENT or
 JOB_DOCUMENT_SOURCE.
         // If you manage the job task on your own, leave it empty and provide the
 JOB_JOBID (self-managed job task).
         // JOB_DOCUMENT is a JSON formatted string
         "JOB_DOCUMENT": "{
            \"operation\":\"reboot\",
            \"files\" : {
               \"fileName\" : \"install.py\",
               \"url\" : \"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket-
name/key}\"
            }
         }",
         // JOB_DOCUMENT_SOURCE is an S3 link to the job document. It will be used only
 if JOB_DOCUMENT is not provided.
         "JOB_DOCUMENT_SOURCE": "https://s3.amazonaws.com/bucket-name/key",
         // JOB_JOBID is mandatory, only if neither document nor document source is
 provided. (Test case needs to know the self-managed job task id).
         "JOB_JOBID": "String",
         // JOB_PRESIGN_ROLE_ARN is used for the presign Url, which will replace the
 placeholder in the JOB_DOCUMENT field
         "JOB_PRESIGN_ROLE_ARN": "String",
         // Presigned Url expiration time. It must be between 60 and 3600 seconds, with
 the default value being 3600.
         "JOB_PRESIGN_EXPIRES_IN_SEC": "Long"
```

```
         "EXECUTION_TIMEOUT": "300", // in seconds
      },
      "test": {
         "id": "Job_Execution",
         "version": "0.0.0"
      }
   }
]
```

有关创建和使用任务文档的更多信息，请参阅任务文档。

权限与策略

您可以使用下列测试来确定附加到设备证书的策略是否符合下列标准最佳实践。

“设备证书附加策略不包含通配符”

验证与设备关联的权限策略是否遵循最佳实践，并且未向设备授予除所需权限以外的其它权限。

API 测试用例定义：

Note

EXECUTION_TIMEOUT 的默认值为 1 分钟。我们建议将超时设置为至少 30 秒。

```
"tests":[
   {
        "name":"my_security_device_policies",
        "configuration": {
            // optional:
            "EXECUTION_TIMEOUT":"60"    // in seconds
        },
        "test": {
            "id": "Security_Device_Policies",
            "version": "0.0.0"
        }
    }
]
```

长时间测试

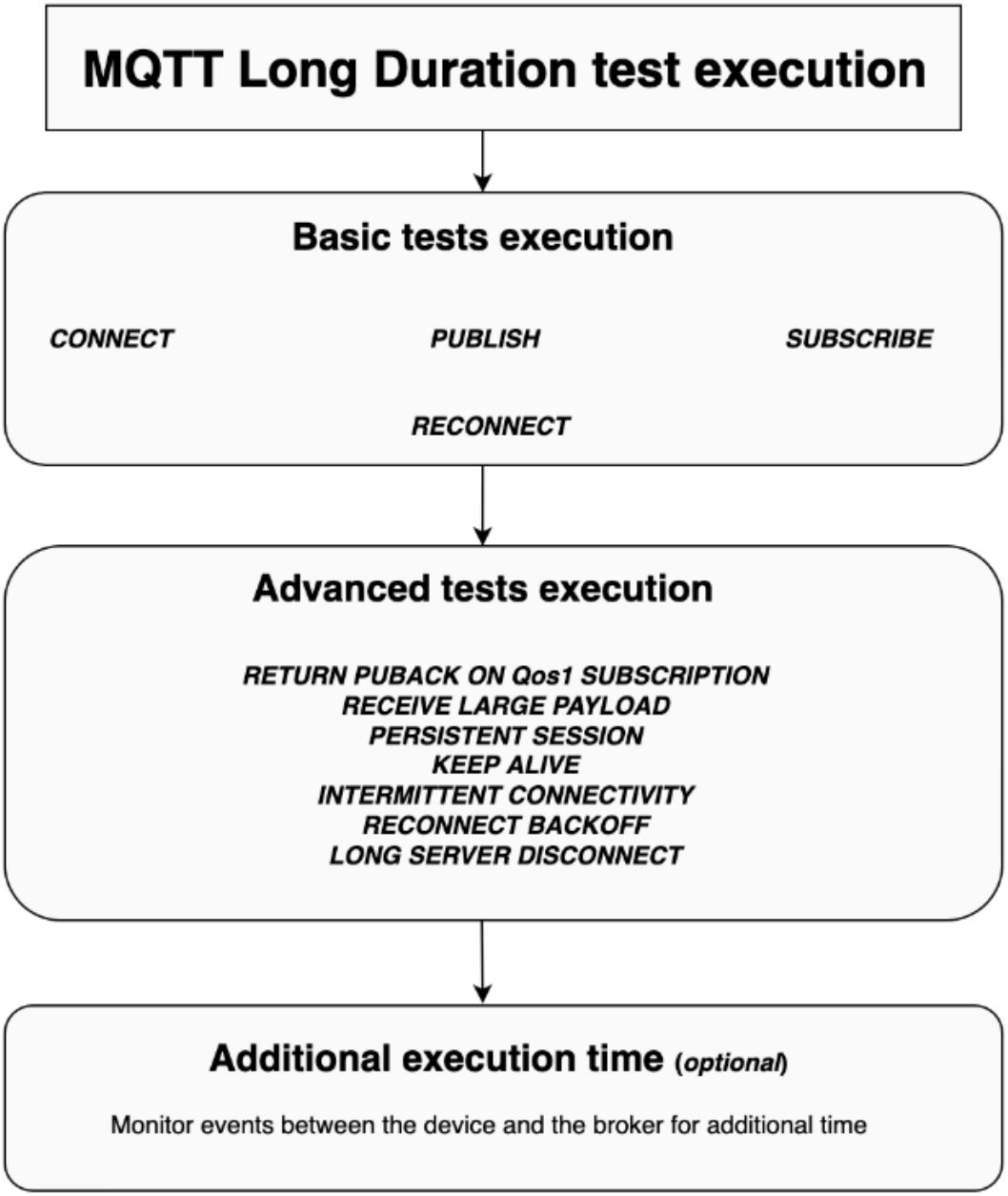
长时间测试是一种新的测试套件，可以在设备长时间运行时监控设备的行为。与运行专注于设备特定行为的单独测试相比，长时间测试可在设备的使用寿命内检查设备在各种真实场景中的行为。Device Advisor 会以尽可能最有效的顺序编排测试。测试生成结果和日志，包括摘要日志，其中包含有关设备在测试期间的性能的有用指标。

MQTT 长时间测试用例

在 MQTT 长时间测试用例中，设备的行为最初是在诸如 MQTT 连接、订阅、发布和重新连接之类的快乐用例场景中观察的。然后，在多种复杂的故障场景中观察该设备，例如 MQTT 重新连接回退、长时间服务器断开连接和间歇性连接。

MQTT 长时间测试用例执行流程

MQTT 长时间测试用例的执行分为三个阶段：



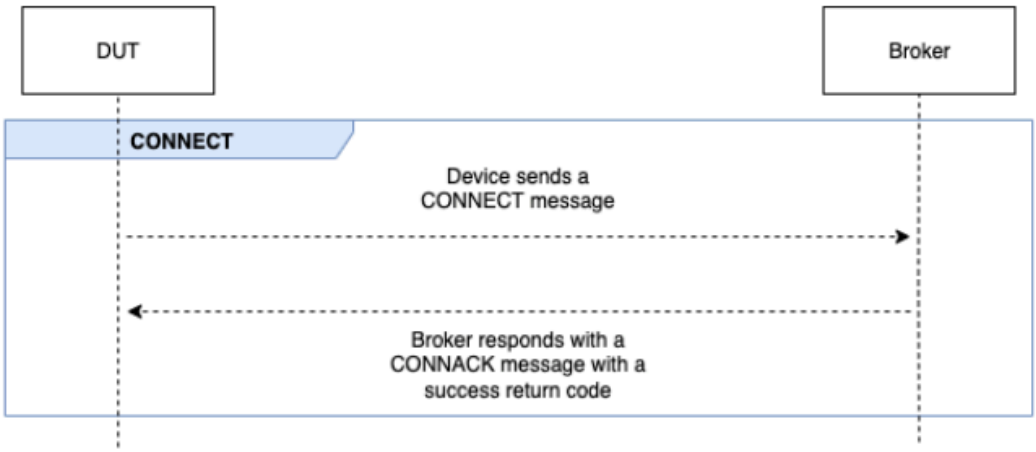
执行基本测试

在此阶段，测试用例并行运行多项简单测试。该测试验证设备是否在配置中选择了这些操作。

根据所选操作，基本测试集可以包括以下内容：

连接

此场景验证设备是否能够成功与代理建立连接。

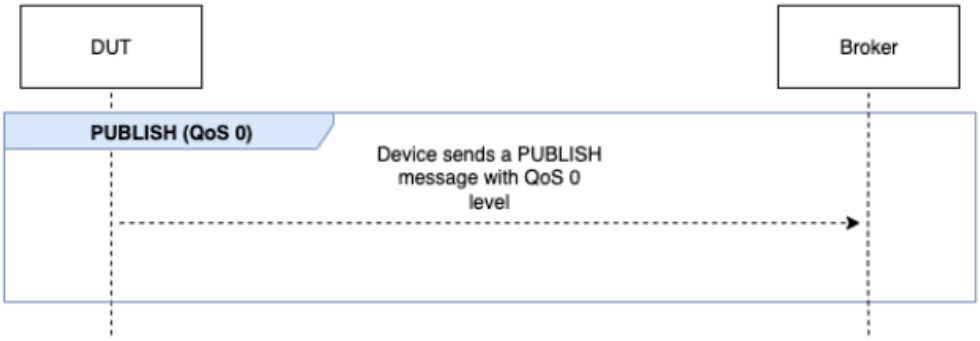


发布

此场景验证设备是否能够成功向代理发布。

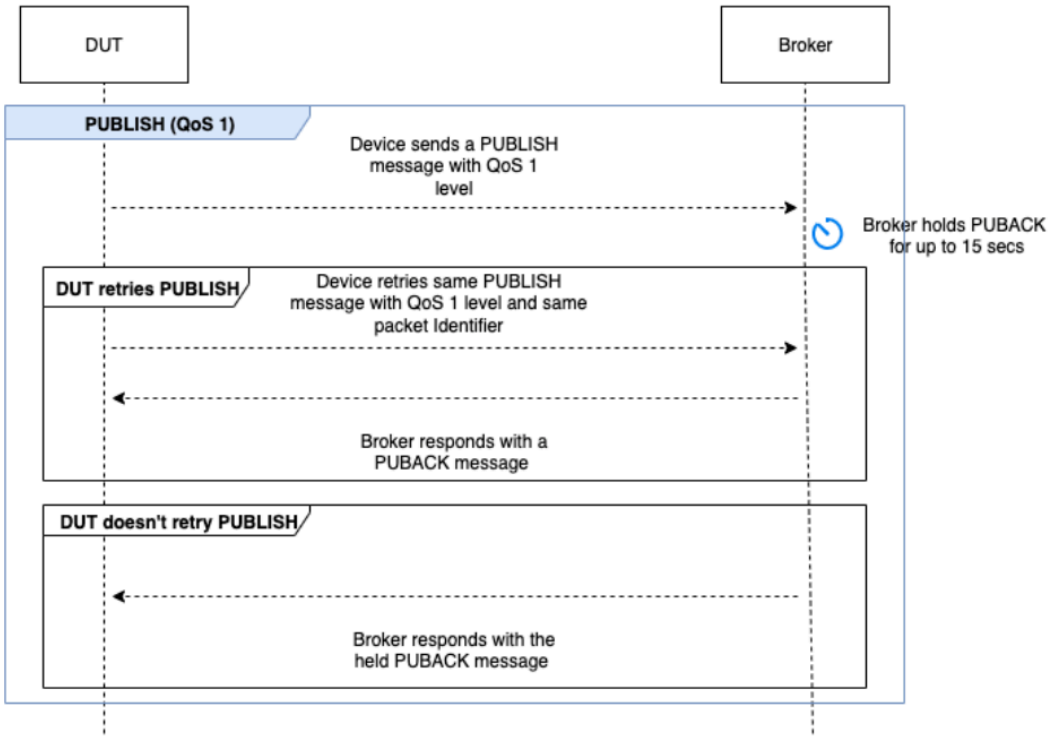
QoS 0

此测试用例验证设备在使用 QoS 0 发布期间是否能够成功向代理发送 PUBLISH 消息。测试不会等待设备收到 PUBACK 消息。



QoS 1

在此测试用例中，设备应使用 QoS 1 向代理发送两条 PUBLISH 消息。在收到第一条 PUBLISH 消息后，代理最多等待 15 秒钟就会做出响应。设备必须在 15 秒的期限内使用相同的数据包标识符重试原始 PUBLISH 消息。如果是这样，代理会回复一条 PUBACK 消息，测试将进行验证。如果设备不重试 PUBLISH，会将原始 PUBACK 发送到设备，测试将被标记为 Pass with warnings（通过但带有警告），并会显示系统消息。在测试执行期间，如果设备丢失连接并重新连接，则测试场景将顺利重置，并且设备必须重新执行测试场景步骤。

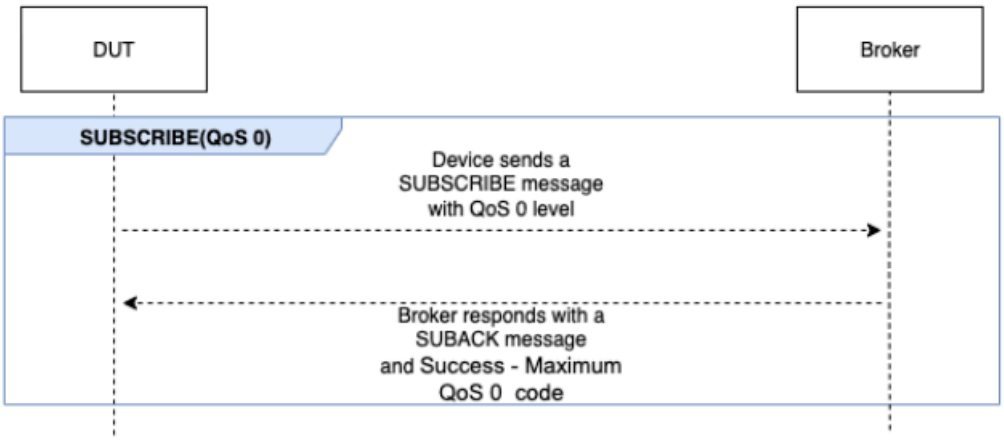


订阅

此场景验证设备是否能够成功向代理订阅。

QoS 0

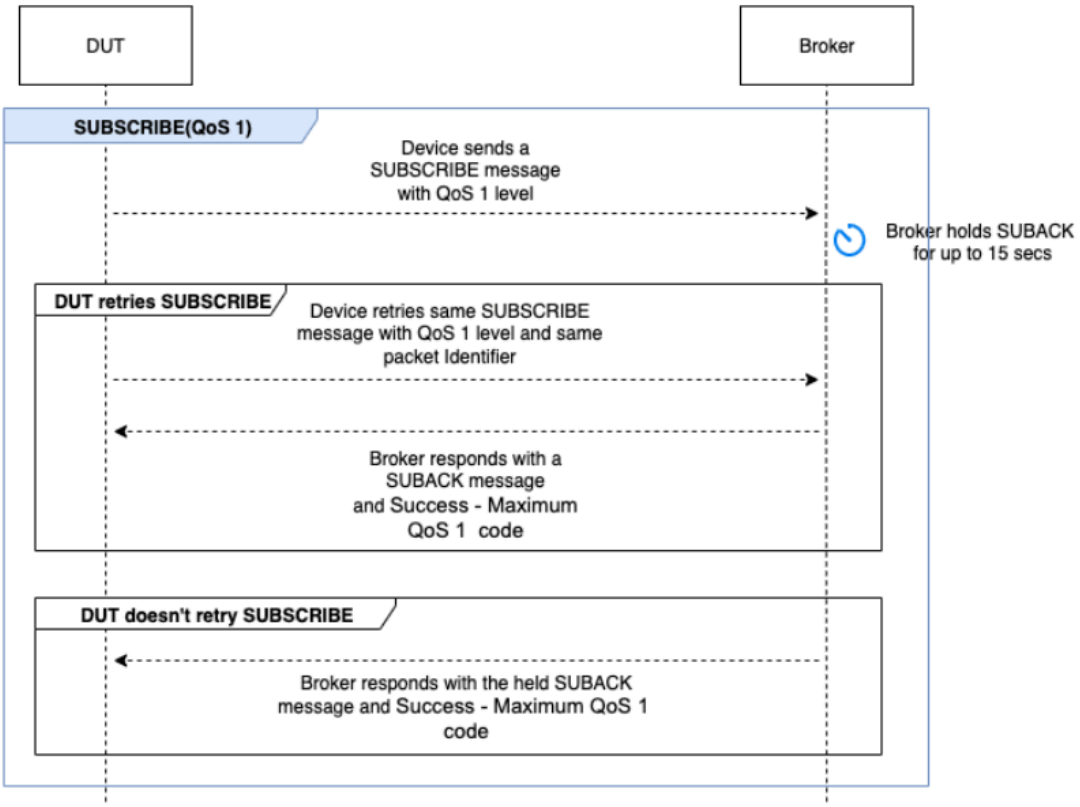
此测试用例验证设备在使用 QoS 0 订阅期间是否能够成功向代理发送 SUBSCRIBE 消息。测试不会等待设备收到 SUBACK 消息。



QoS 1

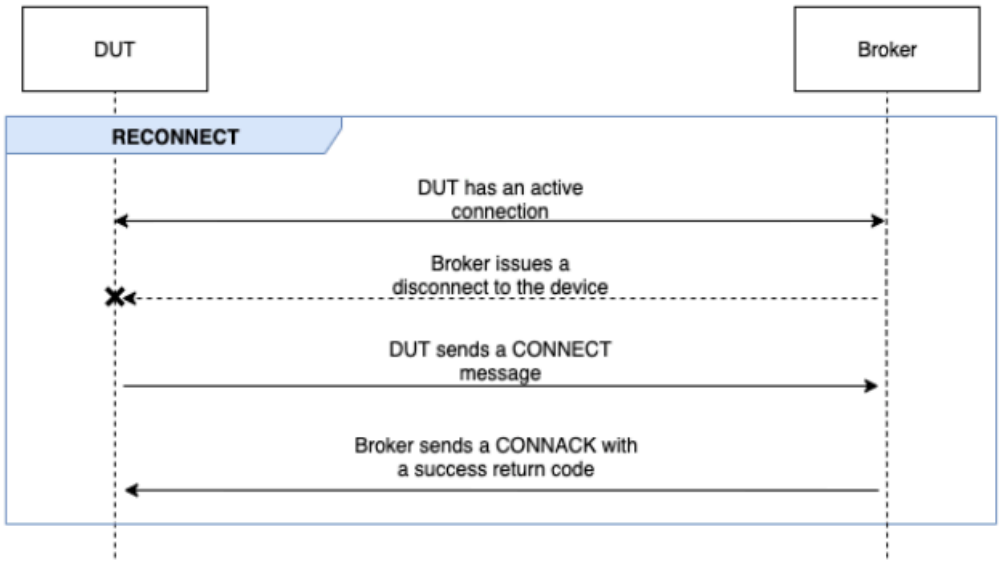
在此测试用例中，设备应使用 QoS 1 向代理发送两条 SUBSCRIBE 消息。在收到第一条 SUBSCRIBE 消息后，代理最多等待 15 秒钟就会做出响应。设备必须在 15 秒的期限内使用相同的数据包标识符重试原

始 SUBSCRIBE 消息。如果是这样，代理会回复一条 SUBACK 消息，测试将进行验证。如果设备不重试 SUBSCRIBE，会将原始 SUBACK 发送到设备，测试将被标记为 Pass with warnings（通过但带有警告），并会显示系统消息。在测试执行期间，如果设备丢失连接并重新连接，则测试场景将顺利重置，并且设备必须重新执行测试场景步骤。



重新连接

此场景验证在设备从成功连接断开连接后，是否能够成功与代理重新连接。如果在之前的测试套件期间多次连接设备，Device Advisor 不会断开设备此连接。相反，它会将测试标记为 Pass（通过）。



执行高级测试

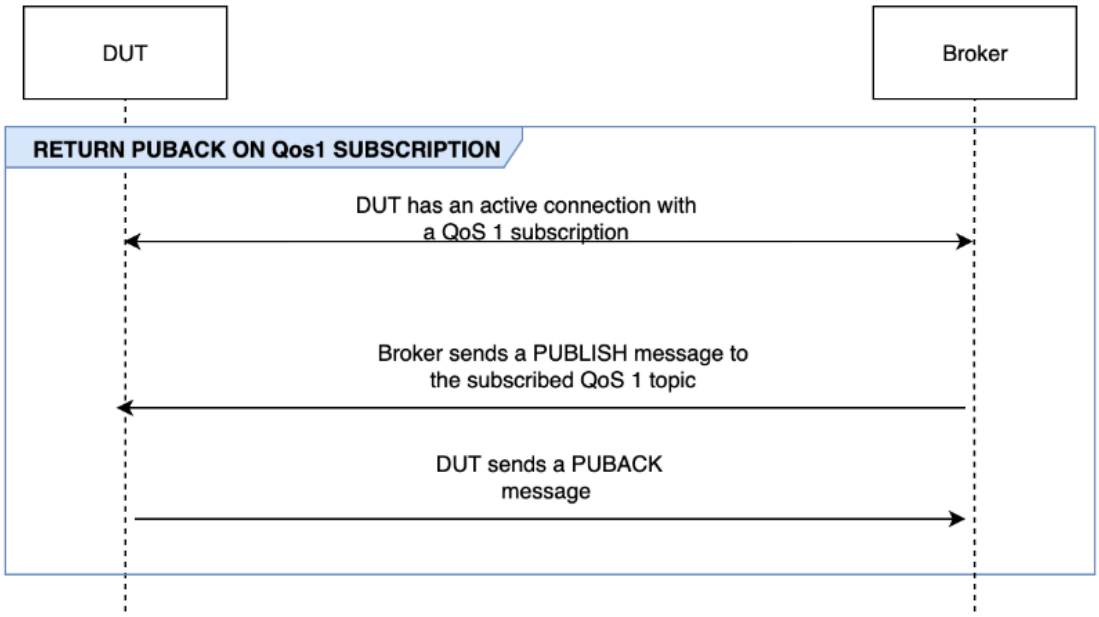
在此阶段，测试用例连续运行更复杂的测试，以验证设备是否遵循最佳实践。这些高级测试可供选择，如果不需要，可以选择退出。根据场景的要求，每个高级测试都有自己的超时值。

订阅 QoS 1 时返回 PUBACK

Note

仅当您的设备能够执行 QoS 1 订阅时才选择此场景。

此场景验证设备订阅主题并收到来自代理的 PUBLISH 消息后是否返回 PUBACK 消息。

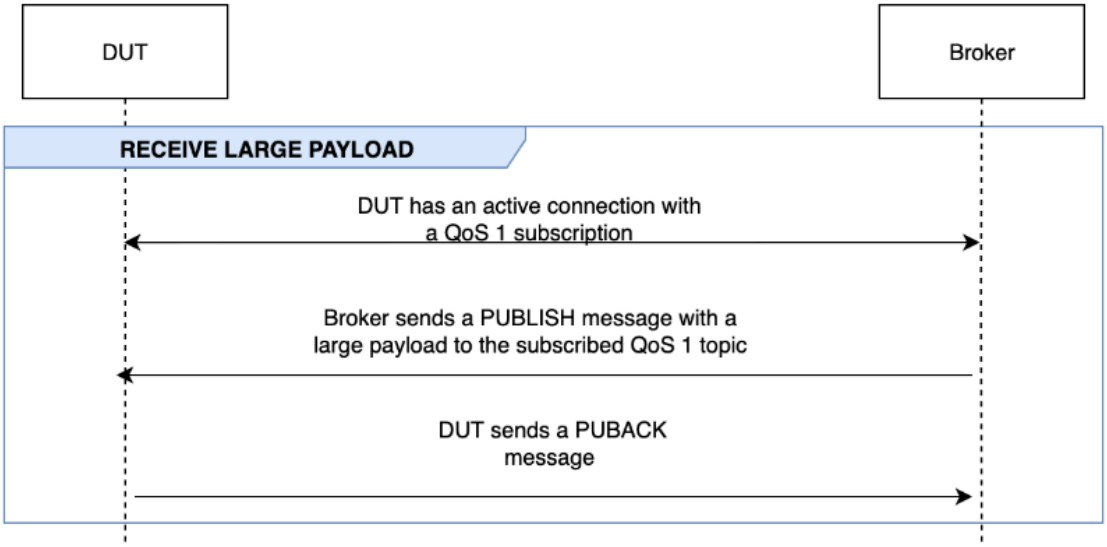


接收较大有效负载

Note

仅当您的设备能够执行 QoS 1 订阅时才选择此场景。

此场景验证设备在从代理接收到 QoS 1 主题的具有较大有效负载的 PUBLISH 消息后，是否以 PUBACK 消息进行响应。可以使用 LONG_PAYLOAD_FORMAT 选项配置预期有效负载的格式。



持久性会话

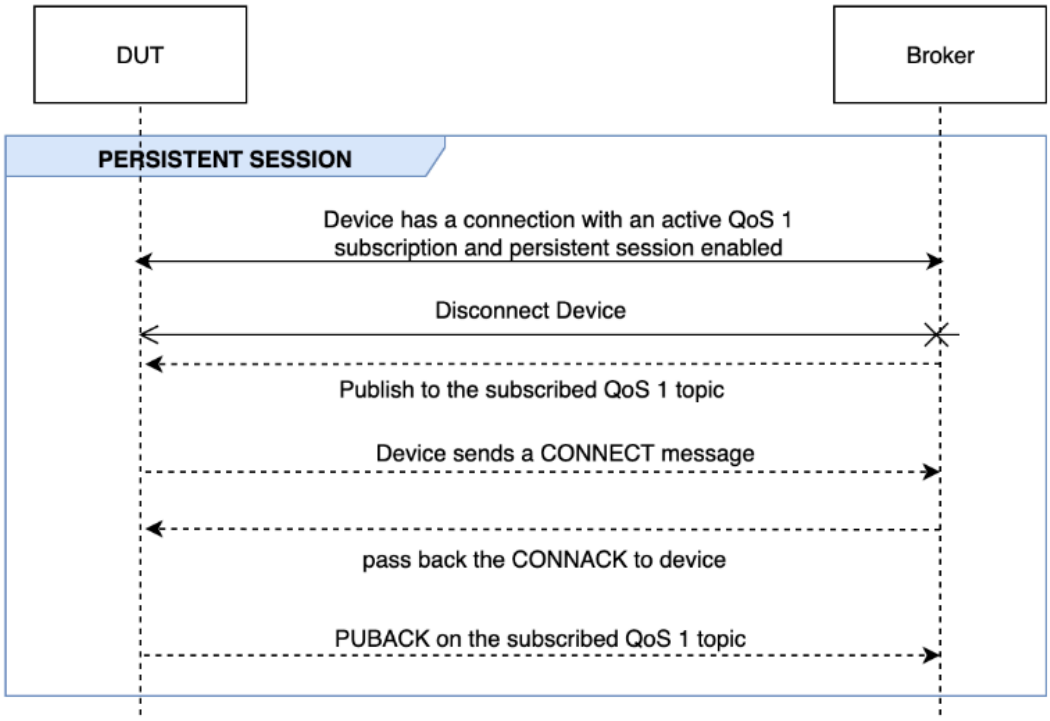
Note

仅当您的设备能够执行 QoS 1 订阅，并且能够保留持久性会话时才选择此场景。

此场景验证设备在保留持续性会话方面的行为。当满足以下条件时，此测试将进行验证：

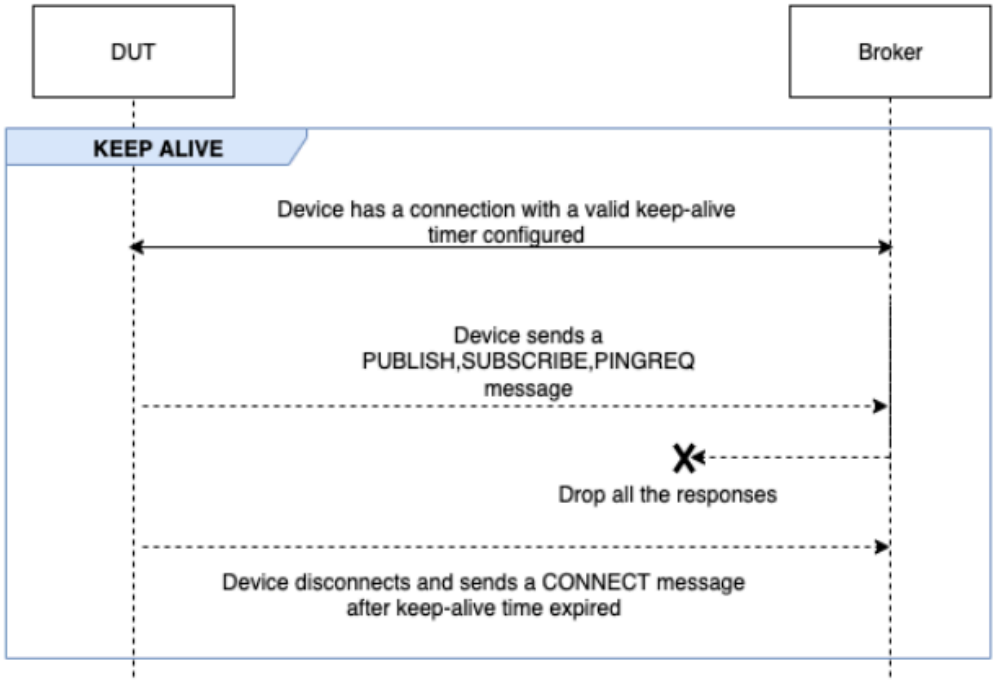
- 设备在具有有效 QoS 1 订阅并启用持久性会话的情况下连接到代理。
- 设备在会话期间成功断开与代理的连接。
- 设备重新连接到代理并恢复对其触发器主题的订阅，而没有明确重新订阅这些主题。
- 设备成功接收代理存储的有关其订阅主题的消息，并按预期运行。

有关 Amazon IoT 持久性会话的更多信息，请参阅使用 MQTT 持久性会话。



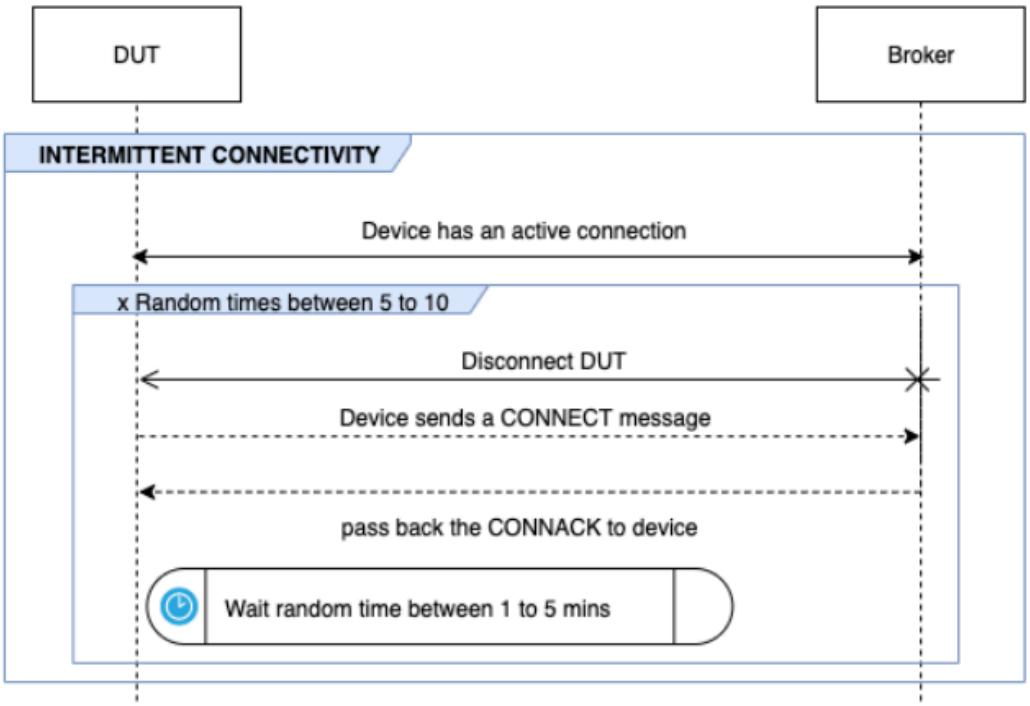
保持活动

此场景验证设备在未收到来自代理的 ping 响应后是否成功断开连接。连接必须配置有效的保持活动计时器。作为此测试的一部分，代理会阻止针对 PUBLISH、SUBSCRIBE 和 PINGREQ 消息发送的所有响应。它还验证正在测试的设备是否断开了 MQTT 连接。



间歇性连接

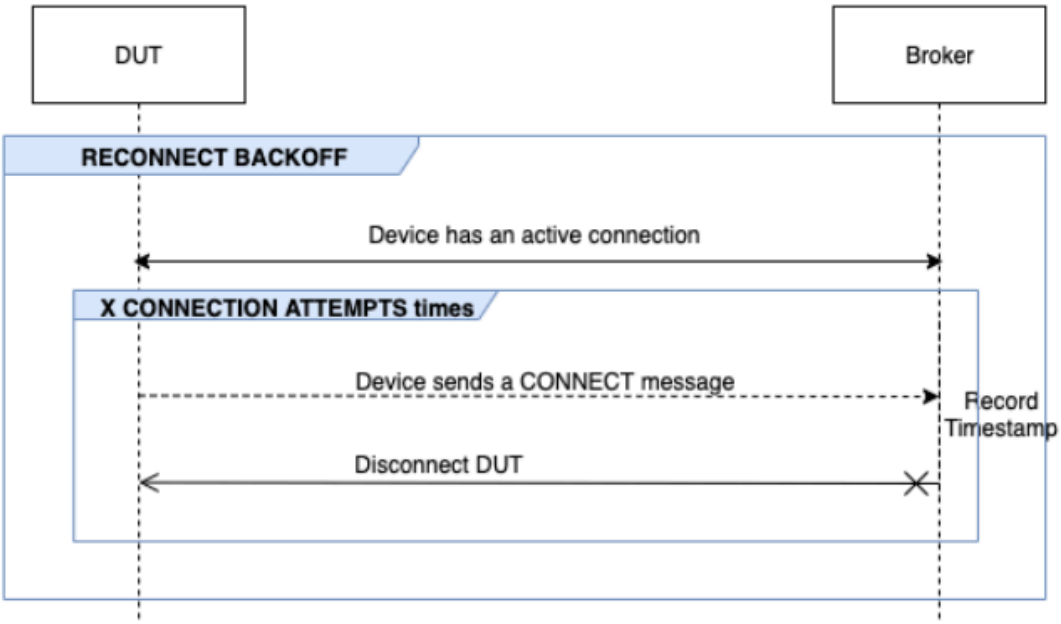
此场景验证在代理以随机间隔断开设备连接一段随机时间后，设备是否可以连接回代理。



重新连接回退

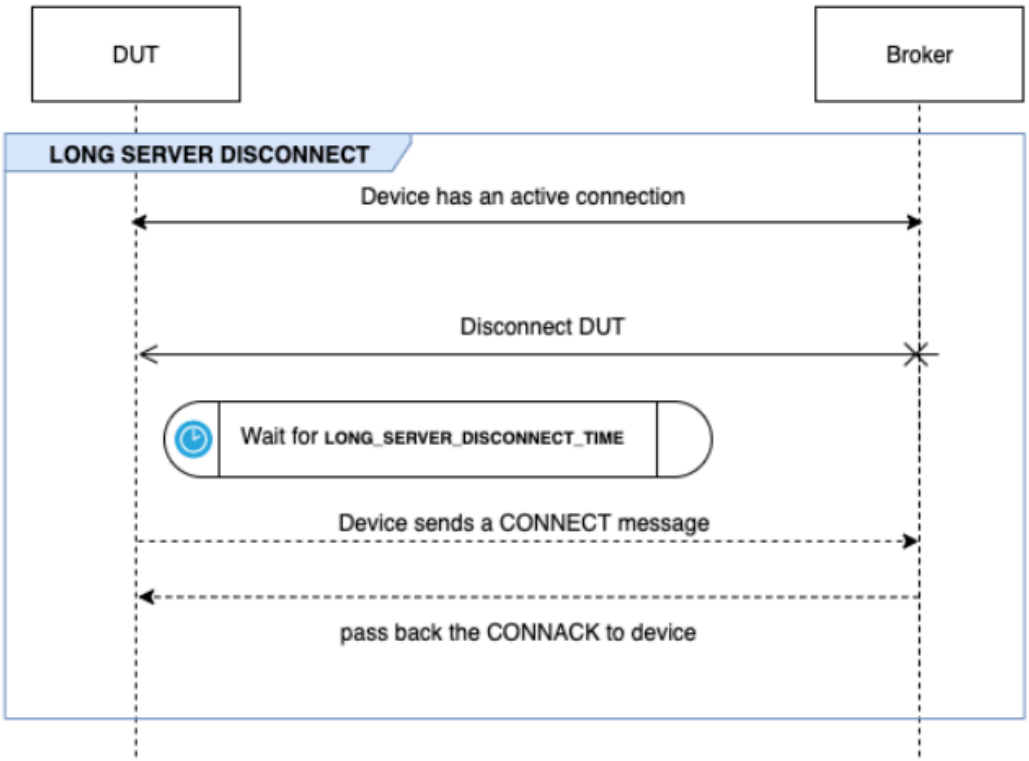
此场景验证代理多次与设备断开连接时，设备是否实现了回退机制。Device Advisor 将回退类型报告为指数回退、抖动回退、线性回退或恒定回退。回退尝试次数可使用 `BACKOFF_CONNECTION_ATTEMPTS` 选项进行配置。默认值为 5。此值可配置为 5 到 10。

要通过此测试，我们建议在所测试设备上实施指数回退和抖动机制。



服务器长时间断开连接

此场景验证代理在较长一段时间（最多 120 分钟）断开设备连接后，设备是否可以成功重新连接。可以使用 LONG_SERVER_DISCONNECT_TIME 选项配置服务器断开连接的时间。默认值为 120 分钟。此值可配置为 30 到 120 分钟。



额外执行时间

额外执行时间是测试在完成所有上述测试之后，以及结束测试用例之前等待的时间。客户利用这段额外时间来监控和记录设备与代理之间的所有通信。可以使用 ADDITIONAL_EXECUTION_TIME 选项配置额外执行时间。默认情况下，此选项设置为 0 分钟，也可以设置为 0 到 120 分钟。

MQTT 长时间测试配置选项

为 MQTT 长时间测试提供的所有配置选项都是可选的。以下选项可用：

OPERATIONS（操作）

设备执行的操作列表，例如 CONNECT、PUBLISH 和 SUBSCRIBE。测试用例根据指定的操作运行场景。未指定的操作假定为有效。

```
{
"OPERATIONS": ["PUBLISH", "SUBSCRIBE"]
//by default the test assumes device can CONNECT
}
```

SCENARIOS（场景）

测试用例根据所选操作运行场景来验证设备的行为。有两种类型的场景：

- 基本场景是一些简单的测试，用于验证设备是否可以执行上面作为配置的一部分选择的操作。这些场景是根据配置中指定的操作预先选择的。配置中不需要额外输入。
- 高级场景是对设备执行的更为复杂的场景，用于验证设备在满足现实条件时是否遵循了最佳实践。这些场景是可选的，可以作为场景数组传递给测试套件的配置输入。

```
{
    "SCENARIOS": [      // list of advanced scenarios
                "PUBACK_QOS_1",
                "RECEIVE_LARGE_PAYLOAD",
                "PERSISTENT_SESSION",
                "KEEP_ALIVE",
                "INTERMITTENT_CONNECTIVITY",
                "RECONNECT_BACK_OFF",
                "LONG_SERVER_DISCONNECT"
    ]
}
```

BASIC_TESTS_EXECUTION_TIME_OUT：

测试用例等待所有基本测试完成的最长时间。默认值为 60 分钟。此值可配置为 30 到 120 分钟。

LONG_SERVER_DISCONNECT_TIME：

在“服务器长时间断开连接”测试期间，测试用例断开连接和重新连接设备所用的时间。默认值为 60 分钟。此值可配置为 30 到 120 分钟。

ADDITIONAL_EXECUTION_TIME：

配置此选项可在所有测试完成后提供一个时间范围，用于监控设备和代理之间的事件。默认值为 5 分钟。此值可配置为 0 到 120 分钟。

BACKOFF_CONNECTION_ATTEMPTS：

此选项可配置测试用例断开连接设备的次数。此选项由重新连接回退测试使用。默认值为 5 次。此值可配置为 5 到 10。

LONG_PAYLOAD_FORMAT：

当测试用例发布到设备订阅的 QoS 1 主题时，设备期望的消息有效负载格式。

API 测试用例定义：

```
{
"tests":[
   {
      "name":"my_mqtt_long_duration_test",
      "configuration": {
         // optional
         "OPERATIONS": ["PUBLISH", "SUBSCRIBE"],
         "SCENARIOS": [
            "LONG_SERVER_DISCONNECT",
            "RECONNECT_BACK_OFF",
            "KEEP_ALIVE",
            "RECEIVE_LARGE_PAYLOAD",
            "INTERMITTENT_CONNECTIVITY",
            "PERSISTENT_SESSION",
         ],
         "BASIC_TESTS_EXECUTION_TIMEOUT": 60, // in minutes (60 minutes by default)
         "LONG_SERVER_DISCONNECT_TIME": 60,   // in minutes (120 minutes by default)
         "ADDITIONAL_EXECUTION_TIME": 60,     // in minutes (0 minutes by default)
         "BACKOFF_CONNECTION_ATTEMPTS": "5",
         "LONG_PAYLOAD_FORMAT":"{"message":"${payload}"}"
      },
      "test":{
         "id":"MQTT_Long_Duration",
         "version":"0.0.0"
      }
   }
 ]
}
```

MQTT 长时间测试用例摘要日志

MQTT 长时间测试用例的运行时间比常规测试用例长。提供了单独的摘要日志，其中列出了运行期间的重要事件，例如设备连接、发布和订阅。详细信息包括测试的内容、未测试的内容和失败的内容。测试会在日志的末尾包含测试用例运行期间发生的所有事件的摘要。摘要包括：

- 设备上配置的“保持活动”计时器。
- 设备上配置的持久性会话标志。
- 测试运行期间的设备连接数。
- 设备重新连接回退类型（如果已通过重新连接回退测试验证）。
- 在测试用例运行期间，设备发布到的主题。
- 在测试用例运行期间，设备订阅的主题。

Amazon IoT Core 设备位置

在使用 Amazon IoT Core 设备位置功能之前，请查看关于此功能的条款和条件。请注意，Amazon 可能会将您的地理位置搜索请求参数（例如用于运行搜索的位置数据）和其他信息传输给您选择的第三方数据提供商，这些提供商可能不在您当前使用的 Amazon Web Services 区域范围内。有关更多信息，请参阅 Amazon 服务条款。

通过 Amazon IoT Core 设备位置功能，可以使用第三方求解器来测试 IoT 设备的位置。求解器是第三方供应商提供的算法，用于解析测量数据并估计设备的位置。通过确定设备的位置，您可以在现场跟踪和调试设备以解决任何问题。

将解析从各种来源收集的测量数据，并将地理位置信息报告为 GeoJSON 有效负载。GeoJSON 格式是一种用于对地理数据结构进行编码的格式。此有效负载中包含设备位置的纬度和经度坐标，这些坐标基于世界大地坐标系统 (WGS84)。

主题

- 测量类型和求解器 (p. 1076)
- Amazon IoT Core 设备定位的工作原理 (p. 1076)
- 如何使用 Amazon IoT Core 设备位置 (p. 1077)
- 解析 IoT 设备的位置 (p. 1078)
- 位置求解器和设备有效负载 (p. 1082)

测量类型和求解器

Amazon IoT Core 设备位置功能与第三方供应商合作来解析测量数据并提供估计的设备位置。下表显示了测量类型和第三方位置求解器，以及有关支持设备的信息。有关 LoRa WAN 设备以及为这些设备配置设备位置的信息，请参阅使用 Amazon IoT Core for LoRaWAN 配置无线资源的位置 (p. 1129)。

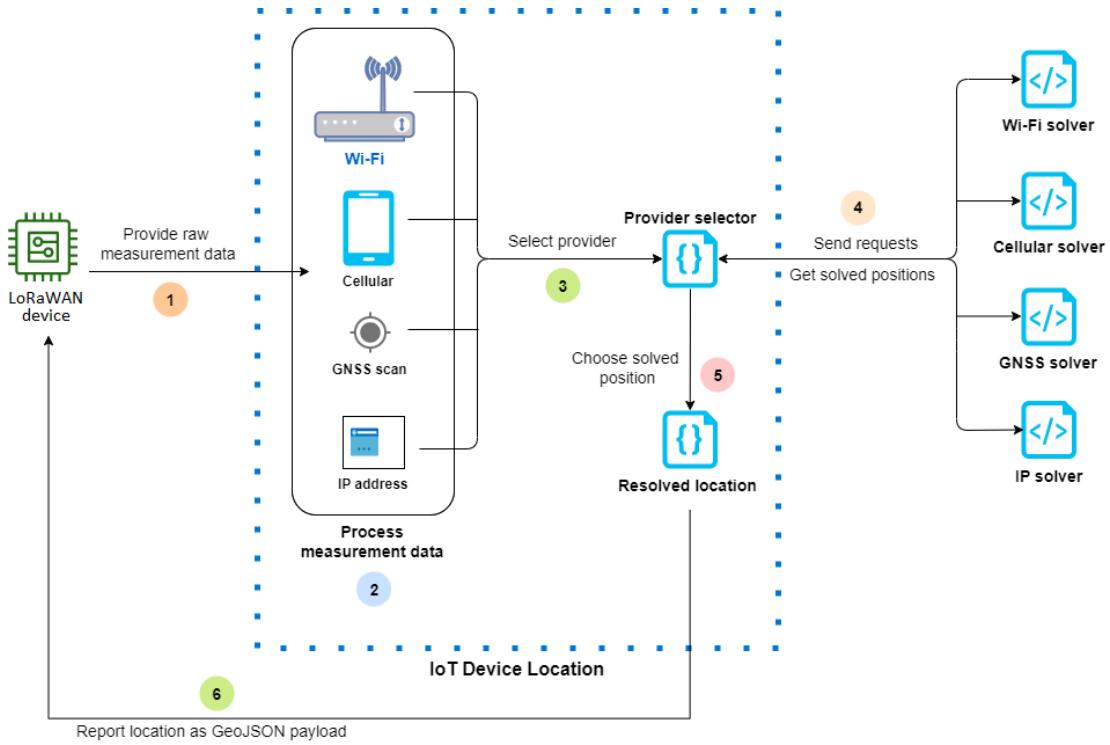
测量类型和求解器

测量类型	第三方求解器	支持的设备
Wi-Fi 接入点	基于 Wi-Fi 的求解器	一般 IoT 设备和 LoRa WAN 设备
蜂窝无线电发射塔：GSM、LTE、CDMA、SCDMA、WCMDA 和 TD-SCDMA 数据	基于蜂窝的求解器	一般 IoT 设备和 LoRa WAN 设备
IP 地址	IP 反向查找求解器	一般 IoT 设备
GNSS 扫描数据（导航消息）	GNSS 求解器	一般 IoT 设备和 LoRa WAN 设备

有关位置求解器的更多信息，以及显示各种测量类型的设备有效负载的示例，请参阅位置求解器和设备有效负载 (p. 1082)。

Amazon IoT Core 设备定位的工作原理

下图显示了 DevAmazon IoT Core ice Locate 如何收集设备测量数据和求解设备位置信息。



以下步骤显示了 Amazon IoT Core 设备位置功能的工作原理。

1. 接收测量数据

   首先从您的设备发送与设备位置相关的原始测量数据。测量数据被指定为 JSON 有效负载。
2. 处理测量数据

   处理测量数据，Amazon IoT Core 设备位置功能会选择要使用的测量数据，这些数据可能是 Wi-Fi、蜂窝网络、GNSS 扫描或 IP 地址信息。
3. 选择求解器

   根据测量数据选择第三方求解器。例如，如果测量数据中包含 Wi-Fi 和 IP 地址信息，则会选择 Wi-Fi 求解器和 IP 反向查找求解器。
4. 获取已解析的位置

   向求解器提供商发送 API 请求，以请求解析位置。Amazon IoT Core然后，设备位置功能会从求解器那里获取估计的地理位置信息。
5. 选择已解析的位置

   比较已解析的位置信息及其准确性，然后 Amazon IoT Core 设备位置功能会选择准确性最高的地理定位结果。
6. 输出位置信息

   将地理位置信息作为 GeoJSON 有效负载发送给您。此有效负载中包含 WGS84 地理坐标、准确性信息、置信度，以及获得解析位置的时间戳。

如何使用 Amazon IoT Core 设备位置

以下步骤显示了如何使用 Amazon IoT Core 设备位置功能。

1. 提供测量数据

   将与设备位置相关的原始测量数据指定为 JSON 有效负载。要检索有效负载测量数据，请转到您的设备日志或使用 CloudWatch Logs，然后复制有效负载数据信息。JSON 有效负载中必须包含一种或多种类型的数据测量。有关显示各种求解器有效负载格式的示例，请参阅位置求解器和设备有效负载 (p. 1082)。
2. 解析位置信息

   使用Amazon IoT控制台中的 Device Locate（设备位置）页面或 GetPositionEstimateAPI 操作，传递有效负载测量数据并解设备位置。 Amazon IoT Core然后，设备位置会选择准确性最高的求解器并报告设备位置。有关更多信息，请参阅解析 IoT 设备的位置 (p. 1078)：
3. 复制位置信息

   验证由 Amazon IoT Core 设备位置解析并报告为 GeoJSON 有效负载的地理位置信息。您可以复制有效负载以用于您的应用程序和其他 Amazon Web Service。例如，您可以使用 位置 (p. 505) Amazon IoT 规则操作将地理位置数据发送到 Amazon Location Service。

以下主题说明了如何使用 Amazon IoT Core 设备位置，以及设备位置有效负载的示例。

- 解析 IoT 设备的位置 (p. 1078)
- 位置求解器和设备有效负载 (p. 1082)

解析 IoT 设备的位置

通过 Amazon IoT Core 设备位置功能来解码来自设备的测量数据，并使用第三方求解器来解析设备位置。解析后的位置以 GeoJSON 有效负载的形式生成，其中包含地理坐标和准确性信息。您可以通过 Amazon IoT 控制台、Amazon IoT Wireless API 或 Amazon CLI 解析设备位置。

主题

- 解析设备位置（控制台）(p. 1078)
- 解析设备位置 (API) (p. 1080)
- 对解析位置时出现的错误进行故障排查 (p. 1081)

解析设备位置（控制台）

解析设备位置（控制台）

1. 转到 Amazon IoT 控制台中的 Device Location（设备位置）页面。
2. 从您的设备日志或日志获取有效负载测量数据，然后将其输入到 Enter JSON pay CloudWatch load data（输入 JSON 有效负载数据）部分中。

   以下代码显示了 JSON 有效负载示例。有效负载包含蜂窝和 Wi-Fi 测量数据。如果您的有效负载包含其他类型的测量数据，将使用准确性最高的求解器。有关更多信息以及有效负载示例，请参阅the section called “位置求解器和设备有效负载” (p. 1082)。

   Note

   JSON 有效负载中必须包含一种类型的数据测量。

```
{
    "Timestamp": 1664313161,
    "WiFiAccessPoints": [{
        "MacAddress": "A0:EC:F9:1E:32:C1",
        "Rss": -77
```

```
    }],
    "CellTowers": {
        "Gsm": [{
            "Mcc": 262,
            "Mnc": 1,
            "Lac": 5126,
            "GeranCid": 16504,
            "GsmLocalId": {
                "Bsic": 6,
                "Bcch": 82
            },
            "GsmTimingAdvance": 1,
            "RxLevel": -110,
            "GsmNmr": [{
                "Bsic": 7,
                "Bcch": 85,
                "RxLevel": -100,
                "GlobalIdentity": {
                    "Lac": 1,
                    "GeranCid": 1
                }
            }]
        }],
        "Wcdma": [{
            "Mcc": 262,
            "Mnc": 7,
            "Lac": 65535,
            "UtranCid": 14674663,
            "WcdmaNmr": [{
                    "Uarfcndl": 10786,
                    "UtranCid": 14674663,
                    "Psc": 149
                },
                {
                    "Uarfcndl": 10762,
                    "UtranCid": 14674663,
                    "Psc": 211
                }
            ]
        }],
        "Lte": [{
            "Mcc": 262,
            "Mnc": 2,
            "EutranCid": 2898945,
            "Rsrp": -50,
            "Rsrq": -5,
            "LteNmr": [{
                    "Earfcn": 6300,
                    "Pci": 237,
                    "Rsrp": -60,
                    "Rsrq": -6,
                    "EutranCid": 2898945
                },
                {
                    "Earfcn": 6300,
                    "Pci": 442,
                    "Rsrp": -70,
                    "Rsrq": -7,
                    "EutranCid": 2898945
                }
            ]
        }]
    }
}
```

3. 要解析位置信息，请选择 Resolve（解析）。

   位置信息作为使用 GeoJSON 格式的有效负载返回，此格式用于对地理数据结构进行编码。有效负载中包含：

   - WGS84 地理坐标，包括纬度和经度信息。还可能包含海拔信息。
   - 报告的位置信息类型，例如 Point（点）。点位置类型将位置表示为 WGS84 纬度和经度，编码为 GeoJSON 点。
   - 水平和垂直准确性信息，表示求解器估计的位置信息与实际设备位置之间的差异。
   - 置信水平，表示位置估计响应中的不确定性。默认值为 0.68，表示实际设备位置在估计位置的不确定性半径内的概率为 68%。
   - 时间戳信息与解析该位置的日期和时间相对应。它使用 Unix 时间戳格式。

   以下代码显示了解析位置后返回的 GeoJSON 有效负载示例。

   Note

   如果 Amazon IoT Core 设备位置在尝试解析位置时报告错误，可以先对错误进行故障排除，然后再解析该位置。有关更多信息，请参阅对解析位置时出现的错误进行故障排查 (p. 1081)：

```
{
    "coordinates": [
        13.376076698303223,
        52.51823043823242
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 45,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 303,
        "horizontalConfidenceLevel": 0.68,
        "timestamp": "2022-11-18T12:23:58.189Z"
    }
}
```

4. 转到 Resource location（资源位置）部分，并验证 Amazon IoT Core 设备位置报告的地理位置信息是否正确。您可以复制有效负载以用于您的应用程序和 Amazon Web Service。例如，您可以使用位置 (p. 505)将地理位置数据发送到 Amazon Location Service。

解析设备位置 (API)

要使用Amazon IoT Wireless API 解析设备位置，请使用 GetPositionEstimateAPI 操作或 get-position-estimateCLI 命令。将有效负载测量数据指定为输入，然后运行 API 操作以解析设备位置。

Note

`GetPositionEstimate` API 操作不存储任何设备或状态信息，也不能用于检索历史位置数据。它执行一次性操作，解析测量数据并生成估计位置。要检索位置信息，每次执行此 API 操作时都必须指定有效负载信息。

以下命令显示了如何使用此 API 操作解析位置的示例。

```
aws iotwireless get-position-estimate --cli-input-json file://location.json
```

以下代码显示了 *location.json* 文件的内容。

location.json 的内容

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress":"54.240.198.35"
      },
     "Gnss":{

 "Payload":"01e82567880c420da31c76020b1c44372661ac4884eae1a0f2e018fd8e8e2fe0010a498b817c4414e91b4974ca2
              "CaptureTime":1330798212.8811495
      },
     "CellTowers": {
        "Lte": [{
            "Mcc": 262,
            "Mnc": 2,
            "EutranCid": 2898945,
            "Rsrp": -50,
            "Rsrq": -5,
            "LteNmr": [{
                    "Earfcn": 6300,
                    "Pci": 237,
                    "Rsrp": -60,
                    "Rsrq": -6,
                    "EutranCid": 2898945
                },
                {
                    "Earfcn": 6300,
                    "Pci": 442,
                    "Rsrp": -70,
                    "Rsrq": -7,
                    "EutranCid": 2898945
                }
            ]
        }]
    }
}
```

此示例中包含蜂窝数据和 IP 地址作为测量类型。Amazon IoT Core设备位置会在基于蜂窝的求解器和 IP 反向查找求解器之间进行选择，并且会选择准确性较高的求解器。解析的位置作为使用 GeoJSON 格式的有效负载返回，此格式用于对地理数据结构进行编码。此有效负载中包含 WGS84 经度和纬度坐标、准确性和置信度信息、位置数据类型，以及解析位置的时间戳。

```
{
    "coordinates": [
        13.37704086303711,
        52.51865005493164
    ],
    "type": "Point",
    "properties": {
        "verticalAccuracy": 707,
        "verticalConfidenceLevel": 0.68,
        "horizontalAccuracy": 389,
        "horizontalConfidenceLevel": 0.68,
        "timestamp": "2022-11-18T14:03:57.391Z"
    }
}
```

对解析位置时出现的错误进行故障排查

当您尝试解析位置时，可能会看到以下错误代码。Amazon IoT Core设备位置在使用 GetPositionEstimate API 操作时可能会生成错误，或者引用与 Amazon IoT 控制台中的错误对应的行号。

- 400 错误

  此错误表示设备有效负载的格式 JSON 无法通过 Amazon IoT Core 设备位置进行验证。出现此错误可能是因为：

  - JSON 测量数据的格式不正确。
  - 有效负载中仅包含时间戳信息。
  - 诸如 IP 地址之类的测量数据参数无效。

  要解决此错误，请检查您的 JSON 格式是否正确，是否包含来自一种或多种测量类型的数据作为输入。如果 IP 地址无效，有关如何提供有效 IP 地址来解决此错误的信息，请参阅 IP 反向查找求解器 (p. 1086)。
- 403 错误

  此错误表示您无权执行 API 操作或无权使用 Amazon IoT 控制台检索设备位置。要解决此错误，请验证您是否拥有执行此操作所需的权限。如果您的 Amazon Web Services Management Console会话或 Amazon CLI 会话令牌已过期，可能会发生此错误。要解决此错误，请刷新会话令牌以使用 Amazon CLI，或者注销 Amazon Web Services Management Console，然后使用您的凭证登录。
- 404 错误

  此错误表明 Amazon IoT Core 设备位置未找到或未解析任何位置信息。此错误可能是由于测量数据输入中数据不足等原因造成的。例如：

  - MAC 地址或蜂窝发射塔信息不足。
  - IP 地址不可用于查找和检索位置。
  - GNSS 有效负载不足。

  要解决此类情况下出现的错误，请检查您的测量数据是否包含解析设备位置所需的足够信息。
- 404 错误

  此错误表明，当 Amazon IoT Core 设备位置功能尝试解析位置时，出现了内部服务器异常。要尝试修复此错误，请刷新会话，并重试发送需要解析的测量数据。

位置求解器和设备有效负载

位置求解器是可用于解析 IoT 设备位置的算法。Amazon IoT Core设备位置支持以下位置求解器。您将看到这些测量类型的 JSON 有效负载格式的示例，求解器支持的设备，以及如何解析位置。

要解析设备位置，请指定一种或多种这些测量数据类型。将为所有测量数据组合返回单个解析位置。

主题

- 基于 Wi-Fi 的求解器 (p. 1082)
- 基于蜂窝的求解器 (p. 1083)
- IP 反向查找求解器 (p. 1086)
- GNSS 求解器 (p. 1086)

基于 Wi-Fi 的求解器

使用基于 Wi-Fi 的求解器和来自 Wi-Fi 接入点的扫描信息来解析位置。此求解器支持 WLAN 技术，可用于计算一般 IoT 设备和 W LoRa AN 无线设备的设备位置。

LoRaWAN 设备必须具有 LoRa Edge 芯片组，该芯片组可以解码传入的 Wi-Fi 扫描信息。 LoRa Edge 是一个超低功耗平台，集成了针对地理定位应用的远程 LoRa 收发器、多星座 GNSS 扫描仪和无源 Wi-Fi MAC 扫描仪。当收到来自设备的上行链路消息时，会将 Wi-Fi 扫描数据发送到 Amazon IoT Core 设备位置，并根据 Wi-Fi 扫描结果估算位置。然后将解码后的信息传递给基于 Wi-Fi 的求解器，以检索位置信息。

基于 Wi-Fi 的求解器有效负载示例

以下代码显示了来自包含测量数据的设备的 JSON 有效负载示例。当 Amazon IoT Core 设备位置接收这些数据作为输入时，它会向求解器提供商发送 HTTP 请求以解析位置信息。要检索此信息，请指定 MAC 地址和 RSS（接收的信号强度）的值。为此，要么使用此格式提供 JSON 有效负载，要么使用 GetPositionEstimateAPI 操作的WiFiAccessPoints对象参数。

```
{
    "Timestamp": 1664313161,    // optional
    "WiFiAccessPoints": [
        {
            "MacAddress": "A0:EC:F9:1E:32:C1",  // required
            "Rss": -75                          // required
        }
    ]
}
```

基于蜂窝的求解器

您可以使用基于蜂窝的求解器和从蜂窝无线发射塔获得的测量数据来解析位置。此求解器支持以下技术。即使包含来自任何或所有这些技术的测量数据，也只能获得单个已解析的位置信息。

- GSM
- CDMA
- WCDMA
- TD-SCDMA
- LTE

基于蜂窝的求解器有效负载示例

以下代码显示了来自包含蜂窝测量数据的设备的 JSON 有效负载示例。当 Amazon IoT Core 设备位置接收这些数据作为输入时，它会向求解器提供商发送 HTTP 请求以解析位置信息。要检索信息，您可以在控制台中使用此格式提供 JSON 负载，或者为 GetPositionEstimateAPI 操作的CellTowers参数指定值。您可以通过使用任何或所有这些蜂窝技术指定参数值来提供测量数据。

LTE（长期演变）

使用此测量数据时，必须指定诸如移动网络的网络和国家/地区代码之类的信息，以及包括有关本地 ID 的信息在内的其他可选参数。以下代码显示了有效负载格式的示例。有关这些参数的更多信息，请参阅 LTE 对象。

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Lte": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "EutranCid": int,             // required
          "Tac": int,                   // optional
          "LteLocalId": {               // optional
              "Pci": int,               // required
              "Earfcn": int,            // required
          },
          "LteTimingAdvance": int,      // optional
          "Rsrp": int,                  // optional
          "Rsrq": float,                // optional
```

```
          "NrCapable": boolean,         // optional
          "LteNmr": [
                {
                    "Pci": int,         // required
                    "Earfcn": int,      // required
                    "EutranCid": int,   // required
                    "Rsrp": int,        // optional
                    "Rsrq": float       // optional
                }
            ]
        }
      ]
   }
}
```

GSM（全球移动通信系统）

使用此测量数据时，必须指定诸如移动网络的网络和国家/地区代码之类的信息、基站信息，以及其他可选参数等信息。以下代码显示了有效负载格式的示例。有关这些参数的更多信息，请参阅 GSM 对象。

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Gsm": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "Lac": int,                   // required
          "GeranCid": int,              // required
          "GsmLocalId": {               // optional
              "Bsic": int,              // required
              "Bcch": int,              // required
          },
          "GsmTimingAdvance": int,      // optional
          "RxLevel": int,               // optional
          "GsmNmr": [                   // optional
            {
                "Bsic": int,            // required
                "Bcch": int,            // required
                "RxLevel": int,         // optional
                "GlobalIdentity": {
                    "Lac": int,         // required
                    "GeranCid": int     // required
                }
             }
          ]
       }
    ]
}
```

CDMA（码分多址）

使用此测量数据时，必须指定诸如信号功率和标识信息、基站信息，以及其他可选参数等信息。以下代码显示了有效负载格式的示例。有关这些参数的更多信息，请参阅 CDMA 对象。

```
{
    "Timestamp": 1664313161,              // optional
    "CellTowers": {
        "Cdma": [
        {
            "SystemId": int,               // required
            "NetworkId": int,              // required
            "BaseStationId": int,          // required
```

```
            "RegistrationZone": int,        // optional
            "CdmaLocalId": {                // optional
              "PnOffset": int,              // required
              "CdmaChannel": int,           // required
            },
            "PilotPower": int,              // optional
            "BaseLat": float,               // optional
            "BaseLng": float,               // optional
            "CdmaNmr": [                    // optional
                {
                    "PnOffset": int,        // required
                    "CdmaChannel": int,     // required
                    "PilotPower": int,      // optional
                    "BaseStationId": int    // optional
                }
             ]
          }
       ]
    }
}
```

WCDMA（宽带码分多址）

使用此测量数据时，必须指定诸如网络和国家/地区代码、信号功率和标识信息、基站信息，以及其他可选参数等信息。以下代码显示了有效负载格式的示例。有关这些参数的更多信息，请参阅 CDMA 对象。

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
        "Wcdma": [
        {
          "Mcc": int,                   // required
          "Mnc": int,                   // required
          "UtranCid": int,              // required
          "Lac": int,                   // optional
          "WcdmaLocalId": {             // optional
              "Uarfcndl": int,          // required
              "Psc": int,               // required
          },
          "Rscp": int,                  // optional
          "Pathloss": int,              // optional
          "WcdmaNmr": [                 // optional
                {
                  "Uarfcndl": int,      // required
                  "Psc": int,           // required
                  "UtranCid": int,      // required
                  "Rscp": int,          // optional
                  "Pathloss": int,      // optional
                }
             ]
          }
       ]
    }
}
```

TD-SCDMA（时分同步码分多址）

使用此测量数据时，必须指定诸如网络和国家/地区代码、信号功率和标识信息、基站信息，以及其他可选参数等信息。以下代码显示了有效负载格式的示例。有关这些参数的更多信息，请参阅 CDMA 对象。

```
{
    "Timestamp": 1664313161,           // optional
    "CellTowers": {
```

```
        "Tdscdma": [
        {
          "Mcc": int,                    // required
          "Mnc": int,                    // required
          "UtranCid": int,               // required
          "Lac": int,                    // optional
          "TdscdmaLocalId": {            // optional
              "Uarfcn": int,             // required
              "CellParams": int,         // required
          },
          "TdscdmaTimingAdvance": int,   // optional
          "Rscp": int,                   // optional
          "Pathloss": int,               // optional
          "TdscdmaNmr": [                // optional
                {
                  "Uarfcn": int,         // required
                  "CellParams": int,     // required
                  "UtranCid": int,       // optional
                  "Rscp": int,           // optional
                  "Pathloss": int,       // optional
                }
            ]
        }
      ]
   }
}
```

IP 反向查找求解器

您可以使用 IP 反向查找求解器，将 IP 地址作为输入来解析位置。此求解器可从已预置 Amazon IoT 的设备中获取位置信息。使用 IPv4 或 IPv6 标准模式或 IPv6 十六进制压缩模式的格式指定 IP 地址信息。然后，将获得已解析的估计位置，包括设备所在的城市和国家/地区等其他信息。

Note

使用 IP 反向查找，即表示您同意不将其用于识别或定位特定的家庭或街道地址。

IP 反向查找求解器有效负载示例

以下代码显示了来自包含测量数据的设备的 JSON 有效负载示例。当 Amazon IoT Core 设备位置收到测量数据中的 IP 地址信息时，它会在求解器提供商的数据库中查找该信息，然后使用该信息来解析位置信息。要检索此信息，要么使用此格式提供 JSON 有效负载，要么为 GetPositionEstimateAPI 操作的 I p 参数指定值。

```
{
    "Timestamp": 1664313161,
    "Ip":{
        "IpAddress":"54.240.198.35"
      }
}
```

GNSS 求解器

使用 GNSS（全球导航卫星系统）求解器和 GNSS 扫描结果消息或 NAV 消息中包含的信息检索设备位置。可以选择提供额外的 GNSS 辅助信息，从而减少求解器搜索信号时必须使用的变量数量。通过提供这种辅助信息（包括位置、高度，以及捕获时间和准确度信息），求解器可以轻松识别视野中的卫星并计算设备位置。

此求解器可以与 LoRa WAN 设备，以及其他预置了的设备结合使用Amazon IoT。对于一般 IoT 设备，如果设备支持使用 GNSS 进行位置估计，当从设备接收 GNSS 扫描信息时，收发器会解析位置信息。对于 LoRa

WAN 设备，这些设备必须配备 LoRa Edge 芯片组。当收到来自设备的上行链路消息时，会将 GNSS 扫描数据发送到 Amazon IoT Core for LoRaWAN，并根据来自收发器的扫描结果估算位置。

GNSS 求解器有效负载示例

以下代码显示了来自包含测量数据的设备的 JSON 有效负载示例。当 Amazon IoT Core 设备位置收到包含测量数据中有效负载的 GNSS 扫描信息时，它会使用收发器和所包含的任何其他辅助信息来搜索信号并解析位置信息。要检索此信息，要么使用此格式提供 JSON 有效负载，要么为 GetPositionEstimateAPI 操作的 Gnss 参数指定值。

> **Note**
>
> 在 Amazon IoT Core Device Location 可以解析设备位置之前，必须从有效负载中删除目标字节。

```
{
    "Timestamp": 1664313161,                  // optional
    "Gnss": {
        "AssistAltitude": number,             // optional
        "AssistPosition": [ number ],         // optional
        "CaptureTime": number,                // optional
        "CaptureTimeAccuracy": number,        // optional
        "Payload": "string",                  // required
        "Use2DSolver": boolean                // optional
    }
}
```

事件消息

本部分中的信息介绍在更新或更改事物或任务时 Amazon IoT 所发布的消息。有关 Amazon IoT Events 服务（该服务允许您创建探测器以监控设备的故障或操作变化，并在上述情况发生时触发操作）的信息，请参阅 Amazon IoT Events。

事件消息的生成方式

在发生特定事件时，Amazon IoT 将发布事件消息。例如，在添加、更新或删除事物时，由注册表生成事件。每个事件将导致发送单个事件消息。事件消息通过 MQTT 随 JSON 负载一起发布。负载的内容取决于事件的类型。

Note

保证事件消息发布一次。事件消息可能会多次发布。事件消息的顺序无法保证。

接收事件消息的策略

要接收事件消息，您的设备必须使用合适的策略，该策略允许设备连接到 Amazon IoT 设备网关并订阅 MQTT 事件主题。您还必须订阅合适的主题筛选条件。

以下是接收生命周期事件时需要的策略示例：

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource":[
            "arn:aws:iot:region:account:/$aws/events/*"
        ]
    }]
}
```

启用 Amazon IoT 事件

在保留主题的订阅者能够接收消息之前，必须启用来自 Amazon Web Services Management Console 的消息或使用 API 或 CLI。有关不同选项管理的事件消息的信息，请参阅 Amazon IoT 事件配置设置表 (p. 1089)。

- 要启用事件消息，请转到 Amazon IoT 控制台的设置选项卡，然后，在基于事件的消息部分选择管理事件。您可以指定要管理的事件。
- 要通过 API 或 CLI 控制发布的事件类型，调用 UpdateEventConfigurationsAPI 或使用update-event-configurations CLI 命令。例如：

```
aws iot update-event-configurations --event-configurations "{\"THING\":{\"Enabled\":
 true}}"
```

Note

使用反斜杠 (\) 转义所有双引号 (")。

您可以通过调用 DescribeEventConfigurationsAPI 或使用describe-event-configurations CLI 命令来获取当前事件配置。例如：。

```
aws iot describe-event-configurations
```

Amazon IoT 事件配置设置表

事件类别 （Amazon IoT 控制台：设置：基于事件的消息）	**eventConfigurations** 键值 (Amazon CLI/API)	事件消息主题
（只能使用 Amazon CLI/API 进行配置）	CA_CERTIFICATE	$aws/events/certificates/registered/*caCertificateId*
（只能使用 Amazon CLI/API 进行配置）	CERTIFICATE	$aws/events/presence/connected/*clientId*
（只能使用 Amazon CLI/API 进行配置）	CERTIFICATE	$aws/events/presence/disconnected/*clientId*
（只能使用 Amazon CLI/API 进行配置）	CERTIFICATE	$aws/events/subscriptions/subscribed/*clientId*
（只能使用 Amazon CLI/API 进行配置）	CERTIFICATE	$aws/events/subscriptions/unsubscribed/*clientId*
任务已完成、已取消	JOB	$aws/events/job/*jobID*/canceled
任务已完成、已取消	JOB	$aws/events/job/*jobID*/cancellation_in_progress
任务已完成、已取消	JOB	$aws/events/job/*jobID*/completed
任务已完成、已取消	JOB	$aws/events/job/*jobID*/deleted
任务已完成、已取消	JOB	$aws/events/job/*jobID*/deletion_in_progress
任务执行：成功、失败、已被拒绝、已取消、已删除	JOB_EXECUTION	$aws/events/jobExecution/*jobID*/canceled
任务执行：成功、失败、已被拒绝、已取消、已删除	JOB_EXECUTION	$aws/events/jobExecution/*jobID*/deleted

事件类别 （Amazon IoT 控制台：设置：基于事件的消息）	**eventConfigurations** 键值 (Amazon CLI/API)	事件消息主题
任务执行：成功、失败、已被拒绝、已取消、已删除	JOB_EXECUTION	$aws/events/jobExecution/*jobID*/failed
任务执行：成功、失败、已被拒绝、已取消、已删除	JOB_EXECUTION	$aws/events/jobExecution/*jobID*/rejected
任务执行：成功、失败、已被拒绝、已取消、已删除	JOB_EXECUTION	$aws/events/jobExecution/*jobID*/removed
任务执行：成功、失败、已被拒绝、已取消、已删除	JOB_EXECUTION	$aws/events/jobExecution/*jobID*/succeeded
任务执行：成功、失败、已被拒绝、已取消、已删除	JOB_EXECUTION	$aws/events/jobExecution/*jobID*/timed_out
事物：已创建、已更新、已删除	THING	$aws/events/thing/*thingName*/created
事物：已创建、已更新、已删除	THING	$aws/events/thing/*thingName*/updated
事物：已创建、已更新、已删除	THING	$aws/events/thing/*thingName*/deleted
事物组：已添加、已删除	THING_GROUP	$aws/events/thingGroup/*thingGroupName*/created
事物组：已添加、已删除	THING_GROUP	$aws/events/thingGroup/*thingGroupName*/updated
事物组：已添加、已删除	THING_GROUP	$aws/events/thingGroup/*thingGroupName*/deleted
事物组层次结构：已添加、已删除	THING_GROUP_HIERARCHY	$aws/events/thingGroupHierarchy/thingGroup/*parentThingGroupName*/childThingGroup/*childThingGroupName*/added

事件类别 （Amazon IoT 控制台：设置：基于事件的消息）	**eventConfigurations** 键值 (Amazon CLI/API)	事件消息主题
事物组层次结构：已添加、已删除	THING_GROUP_HIERARCHY	$aws/events/thingGroupHierarchy/thingGroup/*parentThingGroupName*/childThingGroup/*childThingGroupName*/removed
事物组成员资格：已添加、已删除	THING_GROUP_MEMBERSHIP	$aws/events/thingGroupMembership/thingGroup/*thingGroupName*/thing/*thingName*/added
事物组成员资格：已添加、已删除	THING_GROUP_MEMBERSHIP	$aws/events/thingGroupMembership/thingGroup/*thingGroupName*/thing/*thingName*/removed
事物类型：已创建、已更新、已删除	THING_TYPE	$aws/events/thingType/*thingTypeName*/created
事物类型：已创建、已更新、已删除	THING_TYPE	$aws/events/thingType/*thingTypeName*/updated
事物类型：已创建、已更新、已删除	THING_TYPE	$aws/events/thingType/*thingTypeName*/deleted
事物类型关联：已添加、已删除	THING_TYPE_ASSOCIATION	$aws/events/thingTypeAssociation/thing/*thingName*/thingType/*thingTypeName*/added $aws/events/thingTypeAssociation/thing/*thingName*/thingType/*thingTypeName*/removed

注册表事件

在创建、更新或删除了事物、事物类型和事物组时，注册表将发布事件消息。但是，预设情况下，这些事件不可用。有关如何启用这些事件的更多信息，请参阅 启用 Amazon IoT 事件 (p. 1088)。

注册表可以提供以下事件类型：

- 事物事件 (p. 1092)

- 事物类型事件 (p. 1093)
- 事物组事件 (p. 1095)

事物事件

事物已创建/已更新/已删除

在创建、更新或删除了事物时，注册表将发布以下事件消息：

- $aws/events/thing/*thingName*/created
- $aws/events/thing/*thingName*/updated
- $aws/events/thing/*thingName*/deleted

消息包含以下示例负载：

```
{
    "eventType" : "THING_EVENT",
    "eventId" : "f5ae9b94-8b8e-4d8e-8c8f-b3266dd89853",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName" : "MyThing",
    "versionNumber" : 1,
    "thingTypeName" : null,
    "attributes": {
                "attribute3": "value3",
                "attribute1": "value1",
                "attribute2": "value2"
    }
}
```

负载包含以下属性：

eventType

设置为“THING_EVENT”。

eventId

唯一事件 ID (字符串)。

timestamp

事件发生的 UNIX 时间戳。

operation

触发事件的操作。有效值为：

- CREATED
- 已更新
- DELETED

accountId

您的 Amazon Web Services 账户 ID。

thingId

要创建、更新或删除的事物的 ID。

thingName

要创建、更新或删除的事物的名称。

versionNumber

要创建、更新或删除的事物的版本。在创建事物时，此值设置为 1。每次更新事物时，此值增加 1。

thingTypeName

与事物关联的事物类型 (如果存在)。否则为 null。

个属性

与事物关联的名称/值对的集合。

事物类型事件

事物类型相关事件：

- 事物类型已创建/已弃用/已取消弃用/已删除 (p. 1093)
- 事物类型已与某个事物关联或取消关联 (p. 1094)

事物类型已创建/已弃用/已取消弃用/已删除

在创建、弃用、取消弃用或删除了事物类型时，注册表将发布以下事件消息：

- $aws/events/thingType/*thingTypeName*/created
- $aws/events/thingType/*thingTypeName*/updated
- $aws/events/thingType/*thingTypeName*/deleted

消息包含以下示例负载：

```
{
    "eventType" : "THING_TYPE_EVENT",
    "eventId" : "8827376c-4b05-49a3-9b3b-733729df7ed5",
    "timestamp" : 1234567890123,
    "operation" : "CREATED|UPDATED|DELETED",
    "accountId" : "123456789012",
    "thingTypeId" : "c530ae83-32aa-4592-94d3-da29879d1aac",
    "thingTypeName" : "MyThingType",
    "isDeprecated" : false|true,
    "deprecationDate" : null,
    "searchableAttributes" : [ "attribute1", "attribute2", "attribute3" ],
    "description" : "My thing type"
}
```

负载包含以下属性：

eventType

设置为“THING_TYPE_EVENT”。

eventId

唯一事件 ID (字符串)。

timestamp

事件发生的 UNIX 时间戳。

operation

触发事件的操作。有效值为：

- CREATED
- 已更新
- DELETED

accountId

您的 Amazon Web Services 账户 ID。

thingTypeId

要创建、弃用或删除的事物类型的 ID。

thingTypeName

要创建、弃用或删除的事物类型的名称。

isDeprecated

如果事物类型已弃用，则为 `true`。否则为 `false`。

deprecationDate

弃用事物类型的 UNIX 时间戳。

searchableAttributes

与可用于搜索的事物类型关联的名称/值对集合。

description

事物类型的描述。

事物类型已与某个事物关联或取消关联

当事物类型与某个事物关联或取消关联时，注册表将发布以下事件消息。

- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/added`
- `$aws/events/thingTypeAssociation/thing/`*`thingName`*`/thingType/`*`typeName`*`/removed`

以下是 `added` 负载的示例。`removed` 消息的有效负载类似。

```
{
    "eventId" : "87f8e095-531c-47b3-aab5-5171364d138d",
    "eventType" : "THING_TYPE_ASSOCIATION_EVENT",
    "operation" : "ADDED",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "thingName": "myThing",
    "thingTypeName" : "MyThingType",
    "timestamp" : 1234567890123,
}
```

负载包含以下属性：

eventId

唯一事件 ID (字符串)。

eventType

设置为“THING_TYPE_ASSOCIATION_EVENT”。

operation

触发事件的操作。有效值为：

- 已添加
- REMOVED

thingId

已更改其类型关联的事物的 ID。

thingName

已更改其类型关联的事物的名称。

thingTypeName

已与事物关联或不再与事物关联的事物类型。

timestamp

事件发生的 UNIX 时间戳。

事物组事件

事物组相关事件：

- 事物组已创建/已更新/已删除 (p. 1095)
- 事物已添加到事物组或从事物组中删除 (p. 1097)
- 事物组已添加到事物组或从事物组中删除 (p. 1098)

事物组已创建/已更新/已删除

在创建、更新或删除了事物组时，注册表将发布以下事件消息。

- `$aws/events/thingGroup/groupName/created`
- `$aws/events/thingGroup/groupName/updated`
- `$aws/events/thingGroup/groupName/deleted`

以下是 `updated` 负载的示例。`created` 和 `deleted` 消息的负载类似。

```
{
  "eventType": "THING_GROUP_EVENT",
  "eventId": "8b9ea8626aeaa1e42100f3f32b975899",
  "timestamp": 1603995417409,
  "operation": "UPDATED",
  "accountId": "571EXAMPLE833",
  "thingGroupId": "8757eec8-bb37-4cca-a6fa-403b003d139f",
  "thingGroupName": "Tg_level5",
  "versionNumber": 3,
  "parentGroupName": "Tg_level4",
  "parentGroupId": "5fce366a-7875-4c0e-870b-79d8d1dce119",
  "description": "New description for Tg_level5",
  "rootToParentThingGroups": [
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/TgTopLevel",
      "groupId": "36aa0482-f80d-4e13-9bff-1c0a75c055f6"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level1",
```

```
      "groupId": "bc1643e1-5a85-4eac-b45a-92509cbe2a77"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level2",
      "groupId": "0476f3d2-9beb-48bb-ae2c-ea8bd6458158"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level3",
      "groupId": "1d9d4ffe-a6b0-48d6-9de6-2e54d1eae78f"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level4",
      "groupId": "5fce366a-7875-4c0e-870b-79d8d1dce119"
    }
  ],
  "attributes": {
    "attribute1": "value1",
    "attribute3": "value3",
    "attribute2": "value2"
  },
  "dynamicGroupMappingId": null
}
```

负载包含以下属性：

eventType

设置为“THING_GROUP_EVENT”。

eventId

唯一事件 ID (字符串)。

timestamp

事件发生的 UNIX 时间戳。

operation

触发事件的操作。有效值为：

- CREATED
- 已更新
- DELETED

accountId

您的 Amazon Web Services 账户 ID。

thingGroupId

要创建、更新或删除的事物组的 ID。

thingGroupName

要创建、更新或删除的事物组的名称。

versionNumber

事物组的版本。在创建事物组时，此值设置为 1。每次更新事物组时，此值增加 1。

parentGroupName

父事物组的名称（如果存在）。

parentGroupId

父事物组的 ID（如果存在）。

description

事物组的描述。

rootToParentThingGroups

有关父事物组的信息数组。每个父事物组都有一个条目，从根事物组开始，继续直至达到父事物组。每个条目均包含事物组的 groupArn 和 groupId。

个属性

与事物组关联的名称/值对的集合。

事物已添加到事物组或从事物组中删除

当事物已添加到事物组或从事物组中删除时，注册表将发布以下事件消息。

- $aws/events/thingGroupMembership/thingGroup/*thingGroupName*/thing/*thingName*/added
- $aws/events/thingGroupMembership/thingGroup/*thingGroupName*/thing/*thingName*/removed

消息包含以下示例负载：

```
{
    "eventType" : "THING_GROUP_MEMBERSHIP_EVENT",
    "eventId" : "d684bd5f-6f6e-48e1-950c-766ac7f02fd1",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "groupArn" : "arn:aws:iot:ap-northeast-2:123456789012:thinggroup/MyChildThingGroup",
    "groupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "thingArn" : "arn:aws:iot:ap-northeast-2:123456789012:thing/MyThing",
    "thingId" : "b604f69c-aa9a-4d4a-829e-c480e958a0b5",
    "membershipId" : "8505ebf8-4d32-4286-80e9-c23a4a16bbd8"
}
```

负载包含以下属性：

eventType

设置为“THING_GROUP_MEMBERSHIP_EVENT”。

eventId

事件 ID。

timestamp

事件发生的 UNIX 时间戳。

operation

当事物添加到事物组时为 ADDED。当事物从事物组中删除时为 REMOVED。

accountId

您的 Amazon Web Services 账户 ID。

groupArn

事物组的 ARN。

groupId

组的 ID。

thingArn

在事物组中添加或删除的事物的 ARN。

thingId

在事物组中添加或删除的事物的 ID。

membershipId

表示事物与事物组之间关系的 ID。将事物添加到事物组时生成此值。

事物组已添加到事物组或从事物组中删除

当某个事物组已添加到另一个事物组或从另一个事物组中删除时，注册表将发布以下事件消息。

- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/added`
- `$aws/events/thingGroupHierarchy/thingGroup/`*`parentThingGroupName`*`/childThingGroup/`*`childThingGroupName`*`/removed`

消息包含以下示例负载：

```
{
    "eventType" : "THING_GROUP_HIERARCHY_EVENT",
    "eventId" : "264192c7-b573-46ef-ab7b-489fcd47da41",
    "timestamp" : 1234567890123,
    "operation" : "ADDED|REMOVED",
    "accountId" : "123456789012",
    "thingGroupId" : "8f82a106-6b1d-4331-8984-a84db5f6f8cb",
    "thingGroupName" : "MyRootThingGroup",
    "childGroupId" : "06838589-373f-4312-b1f2-53f2192291c4",
    "childGroupName" : "MyChildThingGroup"
}
```

负载包含以下属性：

eventType

设置为“THING_GROUP_HIERARCHY_EVENT”。

eventId

事件 ID。

timestamp

事件发生的 UNIX 时间戳。

operation

当事物添加到事物组时为 `ADDED`。当事物从事物组中删除时为 `REMOVED`。

accountId

您的 Amazon Web Services 账户 ID。

thingGroupId

父事物组的 ID。

thingGroupName

父事物组的名称。

childGroupId

子事物组的 ID。

childGroupName

子事物组的名称。

任务事件

当任务为待处理、已完成或已取消状态时，或者设备在运行任务时报告成功或失败，Amazon IoT Jobs 服务会发布到 MQTT 协议上的预留主题。设备或管理和监控应用程序可以通过订阅这些主题来跟踪任务的状态。

如何启用任务事件

来自 Amazon IoT Jobs 服务的响应消息不会通过消息代理，也无法由其它客户端或规则订阅。要订阅与任务活动相关的消息，请使用 `notify` 和 `notify-next` 主题。有关任务主题的更多信息，请参阅 任务主题 (p. 101)。

要收到任务更新的通知，请使用 Amazon Web Services Management Console 或 API 或 CLI启用这些任务。有关更多信息，请参阅启用 Amazon IoT 事件 (p. 1088)：

任务事件的工作原理

由于取消或删除任务可能需要一些时间，所以发送两条消息来指示请求的开始和结束。例如，当取消请求开始时，系统会向 `$aws/events/job/jobID/cancellation_in_progress` 主题发送一条消息。当取消请求完成时，系统会向 `$aws/events/job/jobID/canceled` 主题发送一条消息。

任务删除请求的过程与之相似。管理和监控应用程序可以订阅这些主题来跟踪任务的状态。有关发布和订阅 MQTT 主题的更多信息，请参阅 the section called “设备通信协议” (p. 75)。

任务事件类型

不同的任务事件类型如下：

任务已完成/已取消/已删除

当任务已完成、已取消、已删除或正在取消或删除时，Amazon IoT Jobs 服务会在 MQTT 主题上发布消息：

- `$aws/events/job/`*`jobID`*`/completed`
- `$aws/events/job/`*`jobID`*`/canceled`
- `$aws/events/job/`*`jobID`*`/deleted`
- `$aws/events/job/`*`jobID`*`/cancellation_in_progress`
- `$aws/events/job/`*`jobID`*`/deletion_in_progress`

`completed` 消息包含以下示例负载：

```
{
  "eventType": "JOB",
  "eventId": "7364ffd1-8b65-4824-85d5-6c14686c97c6",
  "timestamp": 1234567890,
  "operation": "completed",
  "jobId": "27450507-bf6f-4012-92af-bb8a1c8c4484",
  "status": "COMPLETED",
```

```
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/a39f6f91-70cf-4bd2-a381-9c66df1a80d0",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/2fc4c0a4-6e45-4525-
a238-0fe8d3dd21bb"
  ],
  "description": "My Job Description",
  "completedAt": 1234567890123,
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123,
  "jobProcessDetails": {
    "numberOfCanceledThings": 0,
    "numberOfRejectedThings": 0,
    "numberOfFailedThings": 0,
    "numberOfRemovedThings": 0,
    "numberOfSucceededThings": 3
  }
}
```

canceled 消息包含以下示例负载。

```
{
  "eventType": "JOB",
  "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
  "timestamp": 1234567890,
  "operation": "canceled",
  "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
  "status": "CANCELED",
  "targetSelection": "SNAPSHOT|CONTINUOUS",
  "targets": [
    "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
    "arn:aws:iot:us-east-1:123456789012:thinggroup/ThingGroup1-95c644d5-1621-41a6-9aa5-
ad2de581d18f"
  ],
  "description": "My job description",
  "createdAt": 1234567890123,
  "lastUpdatedAt": 1234567890123
}
```

deleted 消息包含以下示例负载。

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "deleted",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETED",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

cancellation_in_progress 消息包含以下示例负载：

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "cancellation_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "CANCELLATION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

deletion_in_progress 消息包含以下示例负载：

```
{
      "eventType": "JOB",
      "eventId": "568d2ade-2e9c-46e6-a115-18afa1286b06",
      "timestamp": 1234567890,
      "operation": "deletion_in_progress",
      "jobId": "4d2a531a-da2e-47bb-8b9e-ff5adcd53ef0",
      "status": "DELETION_IN_PROGRESS",
      "targetSelection": "SNAPSHOT|CONTINUOUS",
      "targets": [
        "arn:aws:iot:us-east-1:123456789012:thing/Thing0-947b9c0c-ff10-4a80-b4b3-
cd33d0145a0f",
        "arn:aws:iot:us-east-1:123456789012:thinggroup/
ThingGroup1-95c644d5-1621-41a6-9aa5-ad2de581d18f"
      ],
      "description": "My job description",
      "createdAt": 1234567890123,
      "lastUpdatedAt": 1234567890123,
      "comment": "Comment for this operation"
    }
```

任务执行最终状态

当设备将任务执行更新为最终状态时，Amazon IoT Jobs 服务会发布消息：

- $aws/events/jobExecution/*jobID*/succeeded
- $aws/events/jobExecution/*jobID*/failed
- $aws/events/jobExecution/*jobID*/rejected
- $aws/events/jobExecution/*jobID*/canceled
- $aws/events/jobExecution/*jobID*/timed_out
- $aws/events/jobExecution/*jobID*/removed
- $aws/events/jobExecution/*jobID*/deleted

消息包含以下示例负载：

```
{
  "eventType": "JOB_EXECUTION",
  "eventId": "cca89fa5-8a7f-4ced-8c20-5e653afb3572",
  "timestamp": 1234567890,
```

```
  "operation": "succeeded|failed|rejected|canceled|removed|timed_out",
  "jobId": "154b39e5-60b0-48a4-9b73-f6f8dd032d27",
  "thingArn": "arn:aws:iot:us-east-1:123456789012:myThing/6d639fbc-8f85-4a90-924d-
a2867f8366a7",
  "status": "SUCCEEDED|FAILED|REJECTED|CANCELED|REMOVED|TIMED_OUT",
  "statusDetails": {
    "key": "value"
  }
}
```

生命周期事件

Amazon IoT 可以在 MQTT 主题下发布生命周期事件。这些事件在默认情况下可用，且不能被禁用。

Note

生命周期消息可能不会按顺序发送。您可能会收到重复的消息。

连接/断开连接事件

Note

借助 Amazon IoT Device Management 实例集索引，您可以搜索事物，运行聚合查询，并根据事物连接/断开连接事件创建动态组。有关更多信息，请参阅实例集索引。

Amazon IoT 在客户端建立连接或断开连接时将消息发布到以下 MQTT 主题：

- `$aws/events/presence/connected/`*`clientId`* — 连接到消息代理的客户端。
- `$aws/events/presence/disconnected/`*`clientId`* — 与消息代理断开连接的客户端。

下面是一系列 JSON 元素，发布到 `$aws/events/presence/connected/`*`clientId`* 主题的连接/断开连接消息中包含这些元素。

clientId

建立连接或断开连接的客户端的 ID。

Note

包含 # 或 + 的客户端 ID 不接收生命周期事件。

clientInitiatedDisconnect

如果客户端启动断开连接，则为“True”。否则为 false。只能在断开连接消息中找到。

disconnectReason

客户端断开连接的原因。只能在断开连接消息中找到。下表包含有效值。

断开连接原因	描述
`AUTH_ERROR`	客户端无法进行身份验证或授权失败。
`CLIENT_INITIATED_DISCONNECT`	客户端指示它将断开连接。客户端可以通过发送 MQTT`DISCONNECT` 控制数据包来实现此目的，`Close frame`如果客户端正在使用 WebSocket 连接，则可以发送。

断开连接原因	描述
CLIENT_ERROR	客户端出错，导致它断开连接。例如，如果客户端在同一连接上发送多个 MQTT CONNECT 数据包，或者客户端尝试使用超过负载限制的负载进行发布，则将断开连接。
CONNECTION_LOST	客户端-服务器连接被切断。在高网络延迟期间或互联网连接断开时，可能会发生此情况。
DUPLICATE_CLIENTID	客户端使用的是已在使用的客户端 ID。在这种情况下，已连接的客户端将因这一断开连接原因而断开连接。
FORBIDDEN_ACCESS	不允许连接客户端。例如，IP 地址被拒绝的客户端将无法连接。
MQTT_KEEP_ALIVE_TIMEOUT	如果在 1.5 倍的客户端保持连接时间内没有客户端-服务器通信，则客户端将断开连接。
SERVER_ERROR	由于意外的服务器问题而断开连接。
SERVER_INITIATED_DISCONNECT	服务器出于操作原因而故意断开与客户端的连接。
THROTTLED	客户端由于超出限制而断开连接。
WEBSOCKET_TTL_EXPIRATION	客户端已断开连接，因为 a 的连接时间超过 WebSocket 了其 time-to-live值。

eventType

事件类型。有效值为 connected 或 disconnected。

ipAddress

连接客户端的 IP 地址。该地址可以采用 IPv4 或 IPv6 格式。只能在连接消息中找到。

principalIdentifier

用于执行身份验证的凭证。对于 TLS 双向身份验证证书，这是证书 ID。对于其它连接，这是 IAM 凭证。

sessionIdentifier

Amazon IoT 中的全局唯一标识符，在会话持续时间内存在。

timestamp

大致的事件发生时间，使用自 Unix 纪元时间以来的毫秒数表示。时间戳的准确度是 +/- 2 分钟。

versionNumber

生命周期事件的版本号。对于每个客户端 ID 连接，这是一个单调递增的长整数值。订阅者可以使用版本号来推断生命周期事件的顺序。

> **Note**
>
> 客户端连接的连接和断开消息具有相同的版本号。
> 版本号可能会跳过值，而不保证对于每个事件始终增加 1。
> 如果客户端约一小时未连接，则版本号将重置为 0。对于持续会话，当客户端断开连接的时间超过为永久会话配置的 time-to-live (TTL) 时，版本号将重置为 0。

连接消息具有以下结构。

```
 {
    "clientId": "186b5",
    "timestamp": 1573002230757,
    "eventType": "connected",
    "sessionIdentifier": "a4666d2a7d844ae4ac5d7b38c9cb7967",
    "principalIdentifier": "123456789012345678901234567890123456",
    "ipAddress": "192.0.2.0",
    "versionNumber": 0
}
```

断开连接消息具有以下结构。

```
{
    "clientId": "186b5",
    "timestamp": 1573002340451,
    "eventType": "disconnected",
    "sessionIdentifier": "a4666d2a7d844ae4ac5d7b38c9cb7967",
    "principalIdentifier": "123456789012345678901234567890123456",
    "clientInitiatedDisconnect": true,
    "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
    "versionNumber": 0
}
```

处理客户端断开连接

最佳实践是始终为生命周期事件实现等待状态，包括 Last Will and Testament (LWT) 消息。收到断开连接消息后，您的代码应等待一段时间，并在采取措施之前验证设备是否仍处于脱机状态。实现此目标的一种方法是使用 SQS 延迟队列。当客户端收到 LWT 或生命周期事件时，您可以将消息列入队列进行排队（例如 5 秒）。当消息可用并被处理（通过 Lambda 或其它服务）时，您可以先检查设备是否仍处于离线状态，然后再采取进一步操作。

订阅/取消订阅事件

当客户端订阅或取消订阅 MQTT 主题时，Amazon IoT 将向以下 MQTT 主题发布消息：

```
$aws/events/subscriptions/subscribed/clientId
```

or

```
$aws/events/subscriptions/unsubscribed/clientId
```

其中 `clientId` 是连接到 Amazon IoT 消息代理的 MQTT 客户端 ID。

发布到该主题的消息具有以下结构：

```
{
    "clientId": "186b5",
    "timestamp": 1460065214626,
    "eventType": "subscribed" | "unsubscribed",
    "sessionIdentifier": "00000000-0000-0000-0000-000000000000",
    "principalIdentifier": "000000000000/ABCDEFGHIJKLMNOPQRSTU:some-user/
ABCDEFGHIJKLMNOPQRSTU:some-user",
    "topics" : ["foo/bar","device/data","dog/cat"]
```

```
}
```

下面是一系列 JSON 元素，发布到 `$aws/events/subscriptions/subscribed/`*`clientId`* 和 `$aws/events/subscriptions/unsubscribed/`*`clientId`* 主题的已订阅/未订阅消息中包含这些元素。

clientId

订阅或取消订阅的客户端的 ID。

Note

包含 # 或 + 的客户端 ID 不接收生命周期事件。

eventType

事件类型。有效值为 `subscribed` 或 `unsubscribed`。

principalIdentifier

用于执行身份验证的凭证。对于 TLS 双向身份验证证书，这是证书 ID。对于其它连接，这是 IAM 凭证。

sessionIdentifier

Amazon IoT 中的全局唯一标识符，在会话持续时间内存在。

timestamp

大致的事件发生时间，使用自 Unix 纪元时间以来的毫秒数表示。时间戳的准确度是 +/- 2 分钟。

topics

客户端已订阅的一系列 MQTT 主题。

Note

生命周期消息可能不会按顺序发送。您可能会收到重复的消息。

适用于 LoRaWAN 的 Amazon IoT Core

适用于 LoRaWAN 的 Amazon IoT Core 是一个完全托管的 LoRaWAN 网络服务器 (LNS)，它使用配置和更新服务器 (CUPS) 和无线固件更新 (FUOTA) 功能提供网关管理。您可以用私有 LNS 替换适用于 LoRaWAN 的 Amazon IoT Core 并将远距离广域网 (LoRaWAN) 设备和网关连接到 Amazon IoT Core。通过这样做，您将减少维护、运营成本、设置时间和开销成本。

Note

Amazon IoT Core for LoRaWAN 仅支持 IPv4 地址格式。它不支持 IPv6 或双堆栈配置（IPv4 和 IPv6）。有关更多信息，请参阅支持 IPv6 的 Amazon Web Service。

介绍

LoRaWAN 设备是远程、低功耗、电池供电的设备，使用 LoRaWAN 协议在免许可的无线电频谱中运行。LoRaWAN 是基于 LoRa 构建的低功耗广域网 (LPWAN) 通信协议。LoRa 是实现设备之间低功耗、广域通信的物理层协议。

你可以像将其他 IoT 设备加载到 Amazon IoT 一样加载 LoRaWAN 设备. 要将 LoRaWAN 设备连接到 Amazon IoT，您必须使用 loraWAN 网关。网关充当桥梁，将您的设备连接到适用于 LoRaWAN 的 Amazon IoT Core 并交换消息。适用于 LoRaWAN 的 Amazon IoT Core 使用 Amazon IoT 规则引擎将消息从您的 LoRaWAN 设备路由到 Amazon IoT 其他服务。

为了减少开发工作并快速将设备搭载到适用于 LoRaWAN 的 Amazon IoT Core，我们建议您使用 LoRaWAN 认证的终端设备。有关更多信息，请参阅适用于 LoRaWAN 的 Amazon IoT Core 产品概述页面。有关获得设备 LoRaWAN 认证的信息，请参阅认证 LoRaWAN 产品.

如何使用适用于 LoRaWAN 的 Amazon IoT Core

通过使用控制台或 Amazon IoT 无线 API，您可以将 LoRaWAN 设备和网关快速搭载到适用于 LoRaWAN 的 Amazon IoT Core。

使用控制台

要通过使用 Amazon Web Services Management Console 搭载 LoRaWAN 设备和网关，登录到 Amazon Web Services Management Console 然后导航到 Amazon IoT 控制台中适用于 LoRaWAN 的 Amazon IoT Core 页面。然后，您可以使用介绍部分将网关和设备添加到适用于 LoRaWAN 的 Amazon IoT Core。有关更多信息，请参阅使用控制将您的设备和网关连接到 Amazon IoT Core for LoRaWAN (p. 1110)。

使用 API 或 CLI

您可以通过使用 Amazon IoT 无线 API 同时搭载 LoRaWAN 和 Sidewalk 设备。构建适用于 LoRaWAN 的 Amazon IoT Core 的 Amazon IoT Wireless API 由 Amazon SDK 提供支持。有关更多信息，请参阅 Amazon SDK 和工具包。

您可以使用 Amazon CLI 运行用于搭载和管理 LoRaWAN 网关和设备的命令。有关更多信息，请参阅 Amazon IoT 无线 CLI 参考。

适用于 LoRaWAN 的 Amazon IoT Core 区域和端点

适用于 LoRaWAN 的 Amazon IoT Core 为特定于您 Amazon Web Services 区域 的控制层面和数据层面 API 端点提供支持。数据层面 API 端点特定于您的 Amazon Web Services 账户 和 Amazon Web Services 区域。有关适用于 LoRaWAN 的 Amazon IoT Core 端点的更多信息，请参阅 Amazon IoT Core 一般参考中的适用于 LoRaWAN 的 Amazon 端点。

为了在设备和 Amazon IoT 之间进行更安全的通信，您可以在 Virtual Private Cloud (VPC) 中通过 Amazon PrivateLink 将设备连接到 Amazon IoT Core for LoRaWAN，而不是通过公有互联网连接。有关更多信息，请参阅通过 VPC 接口端点连接到 Amazon IoT Core for LoRaWAN (p. 1136)。

适用于 LoRaWAN 的 Amazon IoT Core 具有适用于在设备之间传输的设备数据的配额和 Amazon IoT 无线 API 操作的最大 TPS。有关更多信息，请参阅《Amazon 一般参考》中的适用于 LoRaWAN 的 Amazon IoT Core 配额。

适用于 LoRaWAN 的 Amazon IoT Core 定价

注册 Amazon 后，您可以通过 Amazon 免费套餐开始免费使用适用于 LoRaWAN 的 Amazon IoT Core。

有关一般产品概述和定价的更多信息，请参阅 Amazon IoT Core定价。

什么是适用于 LoRaWAN 的 Amazon IoT Core？

通过将您的 LoRaWAN 设备和网关连接到 Amazon，适用于 LoRaWAN 的 Amazon IoT Core 替换私有 LoRaWAN 网络服务器 (LNS)。使用 Amazon IoT 规则引擎，您可以路由从 LoRaWAN 设备接收的消息，这些消息可以在那里格式化并发送到其他 Amazon IoT 服务。为了保护与 Amazon IoT 的设备通信，适用于 LoRaWAN 的 Amazon IoT Core 使用 X.509 证书。

适用于 LoRaWAN 的 Amazon IoT Core 管理 Amazon IoT Core 需要用于管理 LoRaWAN 网关和设备并与之通信的服务和设备策略。适用于 LoRaWAN 的 Amazon IoT Core 还管理描述将设备数据发送到其它服务的 Amazon IoT 规则。

使用适用于 LoRaWAN 的 Amazon IoT Core，您可以：

- 加载 LoRaWAN 设备和网关并将其连接到 Amazon IoT，无需设置和管理私有 LNS。
- 连接符合 1.0.x 或 1.1 标准或 LoRa Alliance 标准化的 LoRaWAN 规范的 LoRaWAN 设备。这些设备可以在 A 类、B 类或 C 类模式下运行。
- 使用支持 LoRa 基本站点版本 2.0.4 或更高版本的 LoRaWAN网关。符合适用于 LoRaWAN 的 Amazon IoT Core 所有网关运行 LoRa 基本站点兼容版本。
- 使用适用于 LoRaWAN 的 Amazon IoT Core 自适应数据速率监控信号强度、带宽和扩展因子，并根据需要优化数据速率。
- 使用 CUPS 服务更新 LoRaWAN 网关的固件，并使用无线固件更新 (FUOTA) 更新 LoRaWAN 设备的固件。

主题

- 什么是 LoRaWAN？ (p. 1108)
- 适用于 LoRaWAN 的 Amazon IoT Core 工作原理 (p. 1108)

什么是 LoRaWAN？

LoRa Alliance 将 LoRaWAN 描述为“一种低功耗、广域 (LPWA) 网络协议，旨在将电池供电的“事物”无线连接到区域、国家或全球网络中的互联网，并满足物联网 (IoT) 的关键要求，如双向通信、端到端安全、移动性和本地化服务。"。

LoRa 和 LoRaWAN

LoRaWAN 协议是在 LoRA 上运行的低功耗广域网 (LPWAN) 通信协议。LoRaWAN 规范是开放的，因此任何人都可以设置和操作 LoRa 网络。

LoRa 是一种无线音频技术，在无许可证的射频频谱中运行。LoRa 是一种物理层协议，使用扩频调制并支持远程通信，但以窄带宽为代价。它使用具有中央频率的窄带波形发送数据，从而使其能够抵御干扰。

LoRaWAN 技术的特点

- 远距离通信可达 10 英里的视线
- 电池续航时间长达 10 年。为了延长电池续航时间，您可以在 A 类或 B 类模式下操作设备，这需要增加下行链路延迟。
- 设备和维护成本低。
- 无许可证的无线电频谱，但适用于特定区域的规定。
- 低功耗，但有限的负载大小为 51 字节到 241 字节，具体取决于数据速率。数据速率可以是 0.3 Kbit/s — 27 Kbit/s 的数据速率，最大有效负载大小为 222。

了解有关 LoRaWAN 的更多信息

以下链接包含有关 LoRaWAN 技术和 LoRa Basics Station 的有用信息，这是在 LoRaWAN 网关上运行的软件，用于将终端设备连接到适用于 LoRaWAN 的 Amazon IoT Core。

- LoRaWAN 的事物基础

  LoRaWAN 的事物基础知识包含了一个介绍视频，涵盖了 LoRaWAN 的基础知识，以及一系列章节内容，可帮助您了解 LoRa 和 LoRaWAN。
- 什么是 LoRaWAN

  LoRa Alliance 提供了 LoRa 和 LoRaWAN 的技术概览，包括不同区域的 LoRaWAN 规范摘要。
- LoRa Basics Station

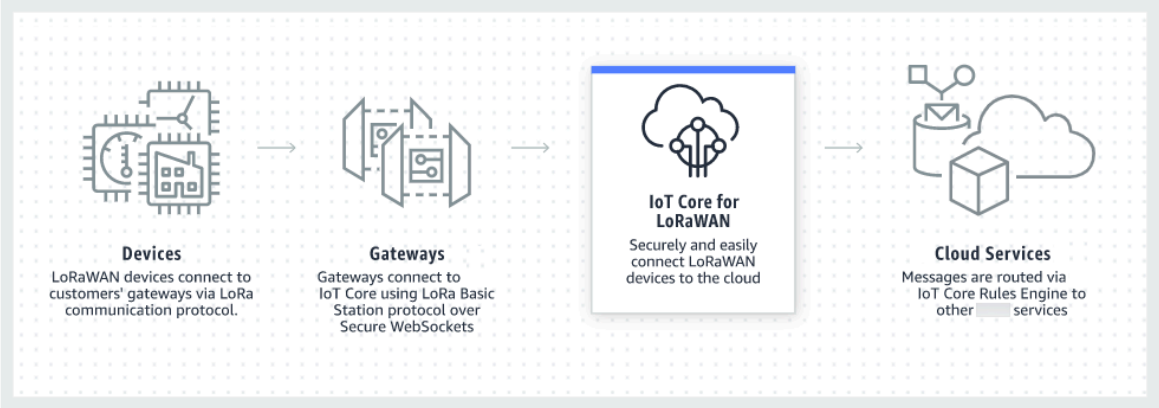
  Semtech Corporation 为网关和终端节点提供了有关 LoRa 基础知识的有用概念。LoRa Basics Station 是一种在您的 LoRaWAN 网关上运行的开源软件，通过 Semtech Corporation 的 GitHub 存储库进行维护和分发。您还可以了解描述如何交换 LoRaWAN 数据和执行配置更新的 LNS 和 CUPS 协议。

适用于 LoRaWAN 的 Amazon IoT Core 工作原理

LoRaWAN 网络架构部署在 星形拓扑中，网关在终端设备和 LoRaWAN 网络服务器 (LNS) 之间中继信息。

适用于 LoRaWAN 的 Amazon IoT Core 将帮助您连接和管理无线 LoRaWAN（低功耗远距离广域网）设备，并取代了开发和操作 LNS 的需求。远距离 WAN (LoRaWAN) 设备和网关可以使用适用于 LoRaWAN 的 Amazon IoT Core 连接到 Amazon IoT Core。

下面显示了 LoRaWAN 设备如何与适用于 LoRaWAN 的 Amazon IoT Core 交互。还显示适用于 LoRaWAN 的 Amazon IoT Core 如何取代 LNS 并与 Amazon Web Services 云 中的其他 Amazon Web Service 进行通信。



LoRaWAN 设备通过 LoRaWAN 网关与 Amazon IoT Core 进行通信。适用于 LoRaWAN 的 Amazon IoT Core 管理需要管理 Amazon IoT Core 和与LoRaWAN网关和设备通信的服务和设备策略。适用于 LoRaWAN 的 Amazon IoT Core 还管理描述将设备数据发送到其他服务的 Amazon IoT 规则的目标。

开始使用适用于 LoRaWAN 的 Amazon IoT Core

1. 选择您需要的无线设备和 LoRaWAN 网关

   Amazon 合作伙伴设备目录包含网关和开发人员工具包，这些网关和开发人员工具包可与适用于 LoRaWAN 的 Amazon IoT Core 搭配使用。有关更多信息，请参阅使用来自Amazon Partner Device Catalog 的合格网关 (p. 1146)。

2. 将您的无线设备和 LoRaWAN 网关添加到适用于 LoRaWAN 的 Amazon IoT Core

   将网关和设备连接到 Amazon IoT Core for LoRaWAN (p. 1110) 为您提供有关如何描述资源以及如何将无线设备和 LoRaWAN 网关添加到适用于 LoRaWAN 的 Amazon IoT Core 的信息。您还将了解如何配置其他适用于 LoRaWAN 的 Amazon IoT Core 资源，以便您用于管理这些设备并将其数据发送到 Amazon 服务。

3. 完成您的适用于 LoRaWAN 的 Amazon IoT Core 解决方案

   从我们的适用于 LoRaWAN 的 Amazon IoT Core 示例解决开始操作并使它为你所用。

适用于 LoRaWAN 的 Amazon IoT Core 资源

以下资源将帮助您熟悉 LoRaWAN 技术和适用于 LoRaWAN 的 Amazon IoT Core。

- 适用于 LoRaWAN 的 Amazon IoT Core 入门

  下面的视频介绍了适用于 LoRaWAN 的 Amazon IoT Core 的工作原理，并指导您完成从 Amazon Web Services Management Console 添加 LoRaWAN 网关的整个过程。

- 适用于 LoRaWAN 的 Amazon IoT Core 工作站

  该工作站涵盖 LoRaWAN 技术的基础知识及其对适用于 LoRaWAN 的 Amazon IoT Core 进行的操作。您还可以使用工作站演示如何将网关和设备连接到适用于 LoRaWAN 的 Amazon IoT Core，以构建一个示例 IoT 解决方案。

将网关和设备连接到 Amazon IoT Core for LoRaWAN

Amazon IoT Core for LoRaWAN将帮助您连接和管理无线 LoRa WAN（低功耗远距离广域网）设备，并取代了开发和操作 LNS 的需求。远距离 WALoRa N (WAN) 设备和网关可以Amazon IoT Core使用连接到 Amazon IoT Core for LoRaWAN。

设备、网关、配置文件和目标的命名约定

在您开始使用 Amazon IoT Core for LoRaWAN 并创建资源之前，请考虑设备、网关和目标的命名约定。

Amazon IoT Core for LoRaWAN 会为您为无线设备、网关和配置文件创建的资源分配唯一 ID；但是，您也可以为资源指定更具描述性的名称，以便更轻松地识别它们。在将设备、网关、配置文件和目标添加到 Amazon IoT Core for LoRaWAN 时，请考虑如何命名以便更轻松地管理。

您可以将标签添加到创建的资源中。在添加 LoRa WAN 设备之前，请考虑如何使用标签来识别和管理 Amazon IoT Core for LoRaWAN资源。添加标签后，可以对其进行修改。

有关命名和标记资源的更多信息，请参阅 描述您的 foAmazon IoT Core r LoRa WAN 资源 (p. 1111)。

将设备数据映射到服务数据

LoRaWAN 无线设备的数据通常被编码以优化带宽。这些编码的消息将以不容易被其它Amazon服务使用的格式抵达 Amazon IoT Core for LoRaWAN。Amazon IoT Core for LoRaWAN 将使用 Amazon IoT 规则，可以借助 Amazon Lambda 函数来处理设备消息并将其解码为其他Amazon服务可以使用的格式。

转换设备数据并将其发送到其它Amazon服务，您需要知道：

- 无线设备发送的数据格式和内容。
- 您想要向其发送数据的服务。
- 服务所需的格式。

使用该信息，您可以创建 Amazon IoT 规则，该规则执行转换并将转换后的数据发送到将使用该数据的Amazon服务。

使用控制将您的设备和网关连接到 Amazon IoT Core for LoRaWAN

您可以使用控制台界面或 API 添加 LoRa WAN 网关和设备。如果您第一次使用的是 Amazon IoT Core for LoRaWAN，我们建议您使用控制台。控制台界面在一次性管理多个 Amazon IoT Core for LoRaWAN 资源时最为实用。当管理大量 Amazon IoT Core for LoRaWAN 资源时，请考虑使用 Amazon IoT Wireless API 创建更为自动化的解决方案。

配置Amazon IoT Core for LoRaWAN资源时输入的大部分数据由设备供应商提供，并且特定于他们支持的 LoRa WAN 规范。以下主题将介绍如何描述Amazon IoT Core适用于 LoRa WAN 的资源，并使用控制台或 API 添加您的网关和设备。

主题

- 描述您的 foAmazon IoT Core r LoRa WAN 资源 (p. 1111)
- 将您的网关搭载到 Amazon IoT Core for LoRaWAN (p. 1112)

- 将您的设备搭载到 Amazon IoT Core for LoRaWAN 上 (p. 1119)

描述您的 foAmazon IoT Core r LoRa WAN 资源

如果您是Amazon IoT Core for LoRaWAN首次使用，您可以借助使用Amazon IoT控制台中Amazon IoT Core for LoRaWAN简介页面来添加您的第一个 LoRa WAN 网关和设备。

在开始创建资源之前，请考虑设备、网关和目标的命名约定。Amazon IoT Core for LoRaWAN 提供了多种选项来标识您创建的资源。尽管 Amazon IoT Core for LoRaWAN 资源会在创建时提供一个唯一的 ID，但这个 ID 不是描述性的，也不能在创建资源后进行更改。您还可以指定名称、添加描述以及将标签和标签值附加到大多数 Amazon IoT Core for LoRaWAN 资源上，从而能够更方便地选择、识别和管理 Amazon IoT Core for LoRaWAN 资源。

- 资源名称 (p. 1111)

  对于网关、设备和配置文件，资源名称是一个可选字段，您可以在创建资源后更改该字段。该名称将出现在资源中心页面上显示的列表中。

  对于目标，您需要在您的Amazon账户和Amazon Web Services 区域中提供一个唯一的名称。创建目标资源之后无法更改目标名称。

  虽然名称最多可包含 256 个字符，但资源中心中的显示空间有限。如果可能，请确保名称的前 20 到 30 个字符即可区分不同名称。
- 资源标签 (p. 1112)

  标签是能够附加到 Amazon 资源上的键/值对（元数据）。您可以同时选择标签键及其相应的值。

  网关、目标和配置文件最多可以附加 50 个标签。设备不支持附加标签。

资源名称

对名称的 Amazon IoT Core for LoRaWAN 资源支持

资源	名称字段支持	
目标	名称是资源的唯一 ID，无法更改。	
设备	名称是资源的可选描述符，可以更改。	
网关	名称是资源的可选描述符，可以更改。	
Profile	名称是资源的可选描述符，可以更改。	

名称字段显示在资源中心的资源列表中；但是，由于空间有限，因此只能显示名称的前 15-30 个字符。

为资源选择名称时，请考虑您希望它们如何识别资源以及如何在控制台中显示。

描述

目标、设备和网关资源还支持描述字段，该字段最多可接受 2048 个字符。描述字段仅显示在单个资源的详细信息页面中。虽然描述字段可以保存大量信息，但由于它只显示在资源的详细信息页面中，因此在多个资源的上下文中扫描并不方便。

资源标签

对 Amazon 标签的 Amazon IoT Core for LoRaWAN 资源支持

资源	Amazon 标签支持	
目标位置	可向资源中最多添加 50 个 Amazon标签。	
设备	此资源不支持Amazon标签。	
网关	可向资源中最多添加 50 个 Amazon标签。	
Profile	可向资源中最多添加 50 个 Amazon标签。	

标签是可作为元数据标识和组织Amazon资源的词或短语。您可以将标签键视为信息类别，将标签值视为该类别中的特定值。

例如，您的标记值可能为 color，然后对于一些资源的这个标签，您可以给予 blue 值，对于其它资源，则给予 red 值。有了这个，您可以使用Amazon控制台中的 Tag Editor 来查找 color 标签值为 blue 的资源。

有关标记和标记策略的更多信息，请参阅 Tag Editor。

将您的网关搭载到 Amazon IoT Core for LoRaWAN

如果您是Amazon IoT Core for LoRaWAN首次使用，您可以使用控制台添加您的第一个 LoRa WAN 网关和设备。

搭载网关之前

将您的网关搭载到 Amazon IoT Core for LoRaWAN 之前，我们建议您：

- 使用能够与 Amazon IoT Core for LoRaWAN 搭配使用的网关。这些网关将连接到，Amazon IoT Core而不会有任何其它配置设置，并且会有兼容版本的 LoRa Basics Station 软件在上面运行。有关更多信息，请参阅使用 for LoRa WANAmazon IoT Core 管理网关 (p. 1145)：
- 请考虑您创建的资源的命名约定，以便更轻松地管理这些资源。有关更多信息，请参阅描述您的 foAmazon IoT Core r LoRa WAN 资源 (p. 1111)：
- 预先准备好每个网关独有的配置参数，使数据输入控制台更加顺畅。Amazon IoT需要用来与网关进行通信和管理的无线网关配置参数包括网关的 EUI 及其 LoRa 频带。

如要搭载您的网关以与 Amazon IoT Core for LoRaWAN 通信：

- 考虑频段选择并添加必要的 IAM 角色 (p. 1112)
- 将网关添加到 Amazon IoT Core for LoRaWAN (p. 1114)
- Connect 您的 LoRa WAN 网关并验证其连接状态 (p. 1118)

考虑频段选择并添加必要的 IAM 角色

在将网关添加到 Amazon IoT Core for LoRaWAN 之前，我们建议您考虑网关运行的频段，并添加必要的 IAM 角色，以便将网关连接到 Amazon IoT Core for LoRaWAN。

> **Note**
>
> 如果您使用控制台添加网关，请在控制台中单击 Create role（创建角色）创建必要的 IAM 角色，以便跳过这些步骤。仅当您使用 CLI 创建网关时，才需要执行这些步骤。

为您的网关和设备连接选择 LoRa 频段

Amazon IoT Core for LoRaWAN 支持 EU863-870、US902-928、AU915 和 AS923-1 频段，您可以使用这些频段连接实际存在于支持这些频段的频率范围和特性的国家/地区的网关和设备。EU863-870 频段和 US902-928 频段通常分别用于欧洲和北美。AS923-1 频段常用于澳大利亚、新西兰、日本和新加坡等国家。AU915 用于澳大利亚和阿根廷等国家。有关在您所在地区或国家/地区使用哪个频段的详细信息，请参阅 LoRaWAN® 区域参数。

LoRa Alliance 发布 LoRa WAN 规范和区域参数文档，可从 A LoRa lliance 网站下载。A LoRa lliance 的区域参数可帮助公司决定在其所在地区或国家使用哪个频段。 Amazon IoT Core for LoRaWAN的频段执行操作遵循了区域参数规范文档中的建议。这些区域参数被分组为一组无线电参数，以及适应工业、科学和医疗 (ISM) 波段的频率分配。我们建议您与合规团队合作，以确保您符合任何适用的法规要求。

添加 IAM 角色以允许 Configuration and Update Server (CUPS) 管理网关凭证

此流程介绍如何添加允许 Configuration and Update Server (CUPS) 管理网关凭证的 IAM 角色。请确保在 LoRa WAN 网关尝试连接到前执行此流程Amazon IoT Core for LoRaWAN；不过，该操作只需执行一次。

添加 IAM 角色以允许 Configuration and Update Server (CUPS) 管理网关凭证

1. 开启 IAM 控制台的角色中心，然后选择 Create role（创建角色）。
2. 如果您认为您可能已经添加了 IoTWirelessGatewayCertManagerRole 角色，请在搜索栏中输入**`IoTWirelessGatewayCertManagerRole`**。

   如果您在搜索结果中看到了 IoTWirelessGatewayCertManagerRole 角色，则您拥有了必要的 IAM 角色。您现在可以离开流程。

   如果搜索结果为空，则您还没有必要的 IAM 角色。继续执行流程以添加该角色。
3. 在 Select type of trusted entity（选择受信任实体的类型）下，选择 Another Amazon Web Services 账户（另一个亚马逊云科技账户）。
4. 在 Account ID（账户 ID），请输入您的Amazon Web Services 账户 ID，然后选择 Next: Permission（下一步：权限）。
5. 在搜索框中，输入 **`AWSIoTWirelessGatewayCertManager`**。
6. 在搜索结果中，选择名为的策略AWSIoTWirelessGatewayCertManager。
7. 依次选择 Next: Tags（下一步：标签）和 Next: Review（下一步：查看）。
8. 对于 Role name（角色名称），请输入 **`IoTWirelessGatewayCertManagerRole`**，然后选择 Create role（创建角色）。
9. 要编辑新角色，请在确认消息中，选择 IoTWirelessGatewayCertManagerRole。
10. 在 Summary（摘要）页面上，选择 Trust relationships（信任关系），然后选择 Edit trust relationship（编辑信任关系）。
11. 在 Policy Document（策略文档）中，更改 `Principal` 属性以使其类似于此示例。

```
"Principal": {
    "Service": "iotwireless.amazonaws.com"
},
```

在您更改 `Principal` 属性后，完整的策略文档应该如此示例所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iotwireless.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

12. 要保存更改并退出，请选择 Update Trust Policy（更新信任策略）。

您现在已经创建了 IoTWirelessGatewayCertManagerRole。今后便无需再次执行此操作了。

如果您在添加网关时执行了此流程，则可以关闭此窗口和 IAM 控制台，然后返回 Amazon IoT 控制台以完成网关的添加。

将网关添加到 Amazon IoT Core for LoRaWAN

您可以私用控制台或 CLI 将网关添加到 Amazon IoT Core for LoRaWAN。

添加您的网关之前，我们建议您考虑 将您的网关搭载到 Amazon IoT Core for LoRaWAN (p. 1112) 的挂载网关之前的注意事项部分中提到的各项因素。

如果您是第一次添加网关，我们建议您使用控制台。如果要改为使用 CLI 添加网关，您必须已创建必要的 IAM 角色，以便网关可以连接到 Amazon IoT Core for LoRaWAN。有关如何创建此角色的信息，请参阅 添加 IAM 角色以允许 Configuration and Update Server (CUPS) 管理网关凭证 (p. 1113)。

使用控制台添加网关

导航到 Amazon IoT 控制台的 Amazon IoT Core for LoRaWAN 简介页面并选择 Get started（开始使用），然后选择 Add gateway（添加网关）。如果您已添加网关，请选择 View gateway（查看网关）以查看您添加的网关。如果要添加更多网关，请选择 Add gateway（添加网关）。

1. 提供网关详细信息和频带信息

   使用 Gateway details（网关详细信息）部分提供有关设备配置数据（如网关的 EUI 和频带配置）的信息。

   - 网关的 EUI

     单个网关设备的 EUI（扩展唯一标识符）。EUI 是一个 16 位的字母数字代码，例如`c0ee40ffff29df10`，它可以唯一标识 LoRa WAN 网络中的网关。此信息特定于您的网关型号，您可以在网关设备或其用户说明手册中找到。

     > **Note**
     >
     > 网关的 EUI 与您可能看到印在网关设备上的 Wi-Fi MAC 地址不同。EUI 遵循 EUI-64 标准，该标准可唯一标识您的网关，因此不能在其它 Amazon Web Services 账户 和区域中重复使用。

   - 频段 (RFRegion)

     网关的频带。您可以从 US915、EU868、AU915 或者 AS923-1 中进行选择，具体取决于您的网关支持的标准以及网关从哪个国家或地区进行实际连接。有关频段的更多信息，请参阅 为您的网关和设备连接选择 LoRa 频段 (p. 1113)。

2. 指定您的无线网关配置数据（可选）

   这些字段是可选字段，您可以使用它们提供有关网关及其配置的其它信息。

   - 网关的名称、描述和标签

     这些可选字段中的信息来自于您如何组织和描述无线系统中的元素。您可以将 Name（名称）添加到网关，使用 Description（说明）字段提供有关网关的信息，并使用 Tag（标签）添加有关网关的元数据键值对。有关命名和描述资源的更多信息，请参阅 描述您的 foAmazon IoT Core r LoRa WAN 资源 (p. 1111)。

- LoRa使用子带和滤波器的 WAN 配置

  此外，您还可以指定 LoRa WAN 配置数据，例如要使用的子带和可控制流量的滤波器。在本教程中，您可以跳过这些字段。有关更多信息，请参阅使用 Amazon IoT Core for LoRaWAN 配置无线资源的位置 (p. 1129)：
- 位置信息和目标

  （可选）您还可以指定位置信息和用于描述处理设备位置数据以供 Amazon IoT Core for LoRaWAN 使用的 Amazon IoT 规则的目标。有关更多信息，请参阅使用 Amazon IoT Core for LoRaWAN 配置无线资源的位置 (p. 1129)：

  Note

  位置信息功能位于 Amazon IoT Core for LoRaWAN 的预览版中，随时可能更改。

3. 将 Amazon IoT 事物与网关关联

   指定是否创建 Amazon IoT 事物，并将其与网关联。Amazon IoT 中的事物可以让您更轻松地搜索和管理您的设备。将某个事物与您的网关关联将允许网关访问其它 Amazon IoT Core 功能。
4. 创建并下载网关证书

   对您的网关进行身份验证，以便它能够安全地与通信Amazon IoT，您的 LoRa WAN 网关必须向提供私有秘钥和证书Amazon IoT Core for LoRaWAN。创建网关证书，从而使 Amazon IoT 能够使用 X.509 标准验证网关的身份。

   单击 Create certificate（创建证书）按钮并下载证书文件。稍后您将使用它们来配置网关。
5. 复制 CUPS 和 LNS 终端节点并下载证书

   在建立与之间的连接时， LoRaWAN 网关必须连接到 CUPS 或 LNS 终端节点Amazon IoT Core for LoRaWAN。建议您使用 CUPS 终端节点，因为它还提供配置管理。要验证的Amazon IoT Core for LoRaWAN终端节点的身份，您的网关将为每个 CUPS 和 LNS 终端节点使用信任证书。

   单击 Copy（复制）按钮复制 CUPS 和 LNS 终端节点。稍后您将需要此信息来配置您的网关。然后单击 Download server trust certificates（下载服务器信任证书）按钮下载 CUPS 和 LNS 终端节点的信任证书。
6. 为网关权限创建 IAM 角色

   您需要添加 IAM 角色，以允许 Configuration and Update Server (CUPS) 管理网关凭证。您必须在 LoRa WAN 网关尝试连接到前执行此操作Amazon IoT Core for LoRaWAN；不过，该操作只需执行一次。

   要为您的账户创建WirelessGatewayCertManager I oT IAM 角色，请单击 Create role（创建角色）按钮。如果该角色已存在，请从下拉列表中选择该角色。

   单击 Submit（提交）以完成网关创建。

使用 API 添加网关

Note

如果您是首次使用 API 或 CLI 添加网关，则必须将WirelessGatewayCertManager I oT IAM 角色，以便网关可以与连接Amazon IoT Core for LoRaWAN。有关如何创建角色的信息，请参阅以下部分添加 IAM 角色以允许 Configuration and Update Server (CUPS) 管理网关凭证 (p. 1113)。

以下部分介绍如何使用Amazon IoT Wireless API 操作或添加网关Amazon CLI。您首先添加您的网关，然后将证书与该网关关联。您还可以使用其他 API 操作，例如更新现有网关。

主题

- 如何添加您的网关 (p. 1116)

- 将证书与您的网关关联 (p. 1116)
- 其他 API 操作 (p. 1118)

如何添加您的网关

您可以使用创建无线网关Amazon CLI，方法是利用 CreateWirelessGatewayAPI 操作或 create-wireless-gatewayCLI 命令来添加您的无线网关。

> Note
>
> 如果您的网关正在与 B 类 LoRa WAN 设备通信，还可以在使用`CreateWirelessGateway` API 或`create-wireless-gateway` CLI 命令添加网关时指定某些信标参数。有关更多信息，请参阅将网关配置为向 B 类设备发送信标 (p. 1146)：

下面的示例将展示如何创建无线 LoRa WAN 设备网关。您也可以提供 `input.json` 文件，该文件将包含其它详细信息，例如网关证书和预调配凭证。

> Note
>
> 您也可以使用 API 执行此流程，即使用 Amazon API 中与此处显示的 CLI 命令对应的方法。

```
aws iotwireless create-wireless-gateway \
    --lorawan GatewayEui="a1b2c3d4567890ab",RfRegion="US915" \
    --name "myFirstLoRaWANGateway" \
    --description "Using my first LoRaWAN gateway"
    --cli-input-json file://input.json
```

将证书与您的网关关联

将网关添加到后Amazon IoT Wireless，必须将其与证书关联才能连接到 CUPS 终端节点。要连接到终端节点，运行 LoRa Basics Station 的网关需要以下文件：

- `cups.crt`-网关的 CUPS 证书，用于连接到 CUPS 端点。
- `cups.key`-与证书对应的私钥。
- `cups.trust`-CUPS 端点的信任证书。
- `cups.uri`-CUPS 端点 URI。

以下步骤演示如何生成证书并将其与网关关联。

主题

- 步骤 1：生成网关证书 (p. 1116)
- 步骤 2：获取服务器信任证书和 CUPS 端点 (p. 1117)
- 步骤 3：将证书与您的网关关关联 (p. 1117)

步骤 1：生成网关证书

要为您的网关生成证书，请使用Amazon IoT API 参考 API 操作或Amazon CLI命令 create-keys-and-certificateCLI 命令。`CreateKeysAndCertificate`

以下命令显示了生成证书`cups.crt`、和私钥的示例`cups.key`。

```
aws iot create-keys-and-certificate \
    --set-as-active --certificate-pem-outfile "cups.crt" \
    --private-key-outfile "cups.key"
```

运行此命令会生成证书和私钥以及证书 ID。下面的示例显示运行此命令的输出。

```
{
    "certificateArn": "arn:aws:iot:us-
east-1:123456789012:cert/abc1234d55ef32101a34434bb123cba2a011b2cdefa6bb5cee1a221b4567ab12",
    "certificateId": "abc1234d55ef32101a34434bb123cba2a011b2cdefa6bb5cee1a221b4567ab12",
    "certificatePem": "-----BEGIN CERTIFICATE-----\n..\n-----END CERTIFICATE-----\n,
          "KeyPair": {
              "PublicKey": "-----BEGIN PUBLIC KEY -----\n..\n----END PUBLIC KEY----\n",
              "PrivateKey": "----BEGIN RSA PRIVATE KEY----\n..\nEND RSA PRIVATE KEY----\n"
    }
}
```

临时存储证书 ID，因为它将在后续步骤中用于将您的证书与网关关联。

Note

您必须安全地存储私钥，cups.key。如果您放错了私钥，请重新运行create-keys-and-certificate命令以生成另一个证书。

步骤 2：获取服务器信任证书和 CUPS 端点

现在您已经生成了证书和私钥，请使用 GetServiceEndpointAPI 操作或 get-service-endpointCLI 命令来获取服务器信任证书cups.trust和端点 URIcups.uri。

以下命令显示获取服务器信任证书和端点 URI 的示例。运行命令时，将service-type参数设置为CUPS。

```
aws iotwireless get-service-endpoint --service-type CUPS
```

下面的内容显示运行此命令的输出。

```
{
    "ServiceType": "CUPS",
    "ServiceEndpoint": "https://ABCDEFGHIJKLMN.cups.lorawan.us-east-1.amazonaws.com:443",
    "ServerTrust": "-----BEGIN CERTIFICATE-----\n..\n-----END CERTIFICATE-----\n"
}
```

从响应中ServiceEndpoint获得的对应于 CUPS 端点cups.uri。

Note

将ServerTrust证书存储在.pem文件中，用新行\n替换。

步骤 3：将证书与您的网关关联

您必须将生成的网关证书与您添加的网关相关联。 Amazon IoT Core for LoRaWAN将使用此信息来识别网关将用于连接到 CUPS 端点的证书。

要将证书与您的网关关联，请使用 AssociateWirelessGatewaywithCertificateAPI 操作或 associate-wireless-gateway-with-certificateCLI 命令。

以下命令显示将证书与网关关联的示例。

```
aws iotwireless associate-wireless-gateway-with-certificate \
    --id <WirelessGatewayId> \
    --iot-certificate-id <CertificateId>
```

运行此命令将返回IotCertificateId，这是您与网关关联的证书的 ID。
以下显示了运行命令的输出，其中IotCertificateId是证书的 ID，例如*abc1234d55ef32101a34434bb123cba2a011b2cdefa6bb5cee1a221b4567ab12*。

```
{
    "IotCertificateId": "<CertificateId>"
}
```

其他 API 操作

您可以使用以下 API 操作来执行与添加、更新或删除 LoRa WAN 网关相关任务。

Amazon IoT Core for LoRaWAN 网关的 Amazon IoT Wireless API 操作

- GetWirelessGateway
- ListWirelessGateways
- UpdateWirelessGateway
- DeleteWirelessGateway

有关可用于创建和管理 Amazon IoT Core for LoRaWAN 资源的操作和数据类型的完整列表，请参阅 Amazon IoT Wireless API 参考。

有关您可以使用的 CLI 的信息，请参阅 Amazon CLI 参考。

Connect 您的 LoRa WAN 网关并验证其连接状态

在检查网关连接状态之前，您必须已添加网关并将其连接到 Amazon IoT Core for LoRaWAN。有关如何添加网关的信息，请参阅 将网关添加到 Amazon IoT Core for LoRaWAN (p. 1114)。

将网关连接到 Amazon IoT Core for LoRaWAN

添加网关后，连接到网关的配置接口以输入配置信息和信任证书。

将网关的信息添加到 Amazon IoT Core for LoRaWAN 之后，请向网关设备添加一些 Amazon IoT Core for LoRaWAN 信息。网关供应商提供的文档中描述的过程应告知您如何将证书文件上载到网关以及配置网关设备以与 Amazon IoT Core for LoRaWAN 通信。

可以与 Amazon IoT Core for LoRaWAN 搭配使用的网关

有关如何配置 W LoRa AN 网关的说明，请参阅Amazon IoT Core for LoRaWAN工作室的配置网关设备部分。在这里，您可以找到相关说明，指导您连接能够与 Amazon IoT Core for LoRaWAN 搭配使用的网关。

支持 CUPS 协议的网关

以下说明介绍如何连接支持 CUPS 协议的网关。

1. 上载您在添加网关时获得的以下文件。
   - 网关设备证书和私有密钥文件。
   - CUPS 终端节点的信任证书文件，`cups.trust`。
2. 指定您之前获得的 CUPS 终端节点 URL。终端节点的格式为 *prefix*`.cups.lorawan.`*region*`.amazonaws.com:443`。

有关如何获取此信息的详细信息，请参阅 将网关添加到 Amazon IoT Core for LoRaWAN (p. 1114)。

支持 LNS 协议的网关

以下说明介绍如何连接支持 LNS 协议的网关。

1. 上载您在添加网关时获得的以下文件。

- 网关设备证书和私有密钥文件。
- LNS 终端节点的信任证书文件，`lns.trust`。

2. 指定您之前获得的 LNS 终端节点 URL。终端节点的格式为 *prefix*`.lns.lorawan.`*region*`.amazonaws.com:443`。

有关如何获取此信息的详细信息，请参阅 将网关添加到 Amazon IoT Core for LoRaWAN (p. 1114)。

在您将您的网关连接到 Amazon IoT Core for LoRaWAN 后，您可以使用控制台或 API 检查连接状态并获取有关何时接收上一个上行链路的信息。

使用控制台检查网关连接状态

要使用控制台检查连接状态，请导航到 Amazon IoT 控制台中的 Gateway（网关）页面，然后选择已添加的网关。在 Gatetway（网关）详细信息页面的LoRa WAN specific details（广域网）部分，您将看到连接状态以及上次接收上行链路的日期和时间。

使用 API 检查网关连接状态

要使用 API 检查连接状态，请使用 `GetWirelessGatewayStatistics` API。此 API 没有请求正文，只包含一个响应正文，用于显示网关是否已连接以及上一个上行链路于何时收到。

> **Note**
>
> 收到最后一条上行链路的时间或 `LastUplinkReceivedAt` 值仅在 3 个月内有效。

```
HTTP/1.1 200
Content-type: application/json

{
  "ConnectionStatus": "Connected",
  "LastUplinkReceivedAt": "2021-03-24T23:13:08.476015749Z",
  "WirelessGatewayId": "30cbdcf3-86de-4291-bfab-5bfa2b12bad5"
}
```

将您的设备搭载到 Amazon IoT Core for LoRaWAN 上

在您将网关挂载到 Amazon IoT Core for LoRaWAN 并验证其连接状态后，您就可以挂载无线设备了。有关如何挂载网关的信息，请参阅 将您的网关搭载到 Amazon IoT Core for LoRaWAN (p. 1112)。

LoRaWAN 设备使用 LoRa WAN 协议与云托管应用程序交换数据。 Amazon IoT Core for LoRaWAN支持符合 Al LoRa liance 标准化的 1.0.x 或 1.1 LoRa WAN 规范的设备。

LoRaWAN 设备通常包含一个或多个传感器和参与者。这些设备通过 LoRa WAN 网关将上行链路遥测数据发送到Amazon IoT Core for LoRaWAN。云托管应用程序可以通过 LoRa WAN 网关向 LoRa WAN 设备发送下行链路命令来控制传感器。

在搭载无线设备之前

在您将无线设备搭载至 Amazon IoT Core for LoRaWAN 前，您需要提前准备好以下信息：

> **Note**
>
> （可选）如果您具有可用设备的位置信息，则可以在搭载设备时指定此信息。有关更多信息，请参阅使用 Amazon IoT Core for LoRaWAN 配置无线资源的位置 (p. 1129)：

- LoRaWAN 规范和无线设备配置

提前准备好每台设备独有的配置参数，使数据输入控制台更加顺畅。您需要输入的具体参数取决于设备使用的 LoRa WAN 规范。有关其规格和配置参数的完整列表，请参阅每个设备的文档。

- 设备名称和描述（可选）

  这些可选字段中的信息来自于您如何组织和描述无线系统中的元素。有关命名和描述资源的更多信息，请参阅 描述您的 foAmazon IoT Core r LoRa WAN 资源 (p. 1111)。

- 设备和服务配置文件

  准备好一些无线设备配置参数，这些参数由许多设备共享，并且可以存储在 Amazon IoT Core for LoRaWAN 中作为设备和服务配置文件。配置参数可在设备的文档中或设备本身找到。在添加设备之前，您需要确定与设备配置参数匹配的设备配置文件，或创建一个配置文件（如有必要）。有关更多信息，请参阅将配置文件添加到 Amazon IoT Core for LoRaWAN (p. 1122)：

- Amazon IoT Core for LoRaWAN 目标

  必须将每个设备分配到一个目标，该目标将处理其消息以发送到 Amazon IoT 和其它服务。负责处理和发送设备消息的 Amazon IoT 规则特定于设备的消息格式。要处理来自设备的消息并将其发送到正确的服务，请确定您将创建与设备消息一起使用的目标，并将其分配给设备。

要将您的无线设备搭载到 Amazon IoT Core for LoRaWAN

- 将您的无线设备添加到 Amazon IoT Core for LoRaWAN (p. 1120)
- 将配置文件添加到 Amazon IoT Core for LoRaWAN (p. 1122)
- Amazon IoT Core为 LoRa WAN 添加目的地 (p. 1124)
- 创建规则以处理 LoRa WAN 设备消息 (p. 1126)
- Connect 您的 LoRa WAN 设备并验证其连接状态 (p. 1128)

将您的无线设备添加到 Amazon IoT Core for LoRaWAN

如果您是首次添加无线设备，我们建议您使用控制台。在 Amazon IoT 控制台中导航到 Amazon IoT Core for LoRaWAN 简介页面，选择 Get started（开始使用），然后选择 Add device（添加设备）。如果您已添加设备，请选择 View device（查看设备）以查看您添加的网关。如果要添加更多设备，请选择 Add device（添加设备）。

此外，您还可以从 Amazon IoT 控制台的 Devices（设备）页面添加无线设备。

> **Note**
>
> 如果从 Devices（设备）页面添加设备，您还可以为设备指定可选位置信息。此信息包含静态位置坐标、您要用于计算设备位置的任何地理位置解算器，以及用于描述处理设备位置数据以供 Amazon IoT Core for LoRaWAN 使用的 Amazon IoT 规则的目标。有关更多信息，请参阅使用 Amazon IoT Core for LoRaWAN 配置无线资源的位置 (p. 1129)：

使用控制台将您的无线设备规范添加到 Amazon IoT Core for LoRaWAN

根据您的激活方法和 LoRa WAN 版本选择 Wireless devices（无线设备规范）。选择后，您的数据将使用 Amazon拥有并为您管理的密钥加密。

OTAA 和 ABP 激活模式

在 LoRa WAN 设备能够发送上行链路数据之前，您必须完成 activation 或者 join p roc edure 流程。要激活您的设备，您可以使用 OTAA（空中激活）或 ABP（通过个性化激活）。

ABP 不需要联接流程，并使用静态密钥。当您使用 OTAA 时， LoRaWAN 设备会发送联接请求，网络服务器将允许该请求。我们建议您使用 OTAA 激活您的设备，因为每次激活都会生成新的会话密钥，这样更加安全。

LoRa广域网版本

当您使用 OTAA 时，您的 LoRa WAN 设备和云托管应用程序将共享根密钥。这些根密钥取决于您使用的是版本 v1.0.x 还是 v1.1。v1.0.x 只有一个根密钥，AppKey（应用程序密钥）；而 v1.1 有两个根密钥，AppKey（应用程序密钥）和 NwkKey（网络密钥）。会话密钥是根据每次激活的根密钥派生的。NwkKey和AppKey都是无线供应商提供的 32 位十六进制值。

无线设备 EUI

在您选择无线设备规范时，您会看到控制台上显示无线设备的 EUI（扩展唯一标识符）参数。您可以从设备或无线供应商的文档中找到此信息。

- DevEUI：设备标签或文档中找到的 16 位六进制值，为您的设备所专有。
- AppEUI：在设备文档中找到的联接服务器的 16 位六进制值，为此服务器所专有。在 LoRa WAN v1.1 版本中，A ppEUI 被称为 JoinEUI。

有关唯一标识符、会话密钥和根密钥的详细信息，请参阅 LoRa Alliance 文档。

使用 API 将您的无线设备规范添加到 Amazon IoT Core for LoRaWAN

如果要使用 API 添加无线设备，则必须先创建设备配置文件和服务配置文件，然后再创建无线设备。创建无线设备时，您将用到设备配置文件和服务配置文件 ID。有关如何使用 API 创建这些配置文件的信息，请参阅使用 API 添加设备配置文件 (p. 1122)。

以下列表描述了执行与添加、更新或删除服务配置文件相关任务的 API 操作。

服务配置文件的 Amazon IoT Wireless API 操作

- CreateWirelessDevice
- GetWirelessDevice
- ListWirelessDevices
- UpdateWirelessDevice
- DeleteWirelessDevice

有关可用于创建和管理 Amazon IoT Core for LoRaWAN 资源的操作和数据类型的完整列表，请参阅 Amazon IoT Wireless API 参考。

如何使用 Amazon CLI 创建无线设备

您可以使用create-wireless-device命令Amazon CLI来创建无线设备。以下示例通过使用输入 input.json 文件来输入参数以创建无线设备。

> **Note**
>
> 您也可以使用 API 执行此流程，即使用 Amazon API 中与此处显示的 CLI 命令对应的方法。

input.json 的内容

```
{
    "Description": "My LoRaWAN wireless device",
    "DestinationName": "IoTWirelessDestination",
    "LoRaWAN": {
        "DeviceProfileId": "ab0c23d3-b001-45ef-6a01-2bc3de4f5333",
        "ServiceProfileId": "fe98dc76-cd12-001e-2d34-5550432da100",
        "OtaaV1_1": {
            "AppKey": "3f4ca100e2fc675ea123f4eb12c4a012",
            "JoinEui": "b4c231a359bc2e3d",
```

```
            "NwkKey": "01c3f004a2d6efffe32c4eda14bcd2b4"
        },
        "DevEui": "ac12efc654d23fc2"
    },
    "Name": "SampleIoTWirelessThing",
    "Type": LoRaWAN
}
```

您可以将此文件作为 `create-wireless-device` 命令的输入。

```
aws iotwireless create-wireless-device \
    --cli-input-json file://input.json
```

有关您可以使用的 CLI 的信息，请参阅 Amazon CLI 参考

将配置文件添加到 Amazon IoT Core for LoRaWAN

可以定义设备和服务配置文件来描述常见设备配置。这些配置文件描述了设备共享的配置参数，以便更轻松地添加这些设备。 Amazon IoT Core for LoRaWAN 支持设备配置文件和服务配置文件。

要输入到这些配置文件的配置参数和值由设备制造商提供。

添加设备配置文件

设备配置文件定义网络服务器用于设置 LoRa WAN 无线电访问服务的设备功能和引导参数。它包括选择参数，例如 LoRa 频段、 LoRa 区域参数版本和设备的 MAC 版本。要了解不同频段的信息，请参阅 为您的网关和设备连接选择 LoRa 频段 (p. 1113)。

使用控制台添加设备配置文件

如果您按 使用控制台将您的无线设备规范添加到 Amazon IoT Core for LoRaWAN (p. 1120) 中所述使用控制台添加无线设备，则在添加无线设备规范后，您可以添加设备配置文件。此外，您还可以在 LoRaWAN 选项卡上的Amazon IoT控制台的 Profiles（ 配置文件 ）的页面上添加无线设备。

您可以从默认设备配置文件中进行选择或创建新的设备配置文件。建议您使用默认设备配置文件。如果您的应用程序要求您创建设备配置文件，请提供设备配置文件名称，选择 Frequency ban (RfRegion)（ 频段（ ），并将其它设置保持为默认值，除非在设备文档中另有指定。

使用 API 添加设备配置文件

如果要使用 API 添加无线设备，则必须在创建无线设备之前创建设备配置文件。

以下列表描述了执行与添加、更新或删除服务配置文件相关任务的 API 操作。

服务配置文件的 Amazon IoT Wireless API 操作

- CreateDeviceProfile
- GetDeviceProfile
- ListDeviceProfiles
- UpdateDeviceProfile
- DeleteDeviceProfile

有关可用于创建和管理 Amazon IoT Core for LoRaWAN 资源的操作和数据类型的完整列表，请参阅 Amazon IoT Wireless API 参考。

如何使用 Amazon CLI 创建设备配置文件

您可以使用create-device-profile命令Amazon CLI来创建设备配置文件。以下示例创建了一个配置文件。

```
aws iotwireless create-device-profile
```

运行此命令会自动创建一个带有 ID 的设备配置文件，您可以在创建无线设备时使用该 ID。现在，您可以使用以下 API 创建服务配置文件，然后使用设备和服务配置文件创建无线设备。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-
a1b2-3c45-67d8-e90fa1b2c34d",
    "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
}
```

有关您可以使用的 CLI 的信息，请参阅 Amazon CLI 参考

添加服务配置文件

服务配置文件描述了设备与应用程序服务器通信所需的通信参数。

使用控制台添加服务配置文件

如果您正在按照 使用控制台将您的无线设备规范添加到 Amazon IoT Core for LoRaWAN (p. 1120) 中所述使用控制台添加无线设备，则在添加设备配置文件后，您可以添加服务配置文件。此外，您还可以在LoRaWAN 选项卡上的Amazon IoT控制台的 Profiles（配置文件）的页面上添加无线设备。

建议您将设置保留MetaData启用 AddGW，从而您将接收到每个负载的其它网关元数据，例如用于数据传输的 RSSI 和 SNR。

使用 API 添加服务配置文件

如果要使用 API 添加无线设备，则必须先创建服务配置文件，然后再创建无线设备。

以下列表描述了执行与添加、更新或删除服务配置文件相关任务的 API 操作。

服务配置文件的 Amazon IoT Wireless API 操作

- CreateServiceProfile
- GetServiceProfile
- ListServiceProfiles
- DeleteServiceProfile

有关可用于创建和管理 Amazon IoT Core for LoRaWAN 资源的操作和数据类型的完整列表，请参阅 Amazon IoT Wireless API 参考。

如何使用 Amazon CLI 创建服务配置文件

您可以使用create-service-profile命令Amazon CLI来创建服务。以下示例创建了一个服务配置文件。

```
aws iotwireless create-service-profile
```

运行此命令会自动创建一个带有可在创建无线设备时使用的 ID 的服务配置文件。现在，您可以使用设备和服务配置文件创建无线设备。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:ServiceProfile/12345678-
a1b2-3c45-67d8-e90fa1b2c34d",
```

```
    "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
}
```

Amazon IoT Core为 LoRa WAN 添加目的地

Amazon IoT Core对于 LoRa WAN 目的地，描述处理设备数据供Amazon服务使用的Amazon IoT规则。

因为 LoRa大多数 WAN 设备不会将数据以能够使用的格式发送到Amazon IoT Core适用于 LoRa WANAmazon 的，因为Amazon IoT规则必须先行处理。Amazon IoT 规则包含解释设备数据的 SQL 语句和主题规则操作，这些操作会将 SQL 语句的结果发送到将要使用它的服务。

如果您是首次添加目标，我们建议您使用控制台。

使用控制台添加一个目标

如果您正在按照 使用控制台将您的无线设备规范添加到 Amazon IoT Core for LoRaWAN (p. 1120) 中所述使用控制台添加无线设备，则当您将无线设备规范和配置文件按前述操作添加到 Amazon IoT Core for LoRaWAN 之后，您可以继续添加目标。

您还可以从 Amazon IoT 控制台的 Destinations（目标）页面添加一个 Amazon IoT Core for LoRaWAN 目标。

要处理设备的数据，请在创建Amazon IoT Core适用于 LoRa WAN 的目标时指定以下字段，然后选择 Add destination（添加目标）。

- 目标详细信息

  输入 Destination name（目标名称）以及您的目标描述（可选）。
- Rule name（规则名称）

  被配置为评估设备发送的消息并处理设备数据的 Amazon IoT 规则。规则名称将映射到您的目标。目标要求规则来处理收到的消息。您可以通过调用 Amazon IoT 规则或通过发布到 Amazon IoT 消息代理。
  - 如果选择 Enter a rule name（输入规则名称），请输入名称，然后选择 Copy（复制）以复制创建 Amazon IoT 规则时要输入的规则名称。您可以选择 Create rule（创建规则）以立即创建规则，或导航到 Amazon IoT 控制台的 Rule（规则）中心并使用该名称创建规则。

    您也可以输入规则并使用高级设置以指定主题名称。主题名称是在规则调用期间提供的，可通过使用 `topic` 规则中的表达式访问。有关Amazon IoT规则的更多信息，请参阅Amazon IoT 规则 (p. 442)。
  - 如果选择 Publish to Amazon IoT message broker（发布到消息代理），输入主题名称。然后，您可以复制 MQTT 主题名称，多个订阅者可以订阅此主题以接收发布到该主题的消息。有关更多信息，请参阅MQTT 主题 (p. 94)：

  有关目标的 Amazon IoT 规则的更多信息，请参阅 创建规则以处理 LoRa WAN 设备消息 (p. 1126)。
- 角色名称

  IAM 角色，该角色授予设备数据权限以访问在 Rule name（规则名称）中命名的规则。在控制台中，您可以创建新的服务角色或选择现有的服务角色。如果要创建新的服务角色，可以输入角色名称（例如，**`IoTWirelessDestinationRole`**），或将其留白 Amazon IoT Core for LoRaWAN 以生成新的角色名称。然后 Amazon IoT Core for LoRaWAN 将代表您自动创建具有适当权限的 IAM 角色。

  有关 IAM 角色的更多信息，请参阅使用 IAM 角色。

使用 API 添加目标

如果要改为使用 CLI 添加目标，必须先为目标创建规则和 IAM 角色。有关角色中目标所需的详细信息，请参阅 为您的目标创建 IAM 角色 (p. 1125)。

以下列表包括执行与添加、更新或删除目标相关任务的 API 操作。

Amazon IoT Wireless目标的 API 操作

- CreateDestination
- GetDestination
- ListDestinations
- UpdateDestination
- DeleteDestination

有关可用于创建和管理 Amazon IoT Core for LoRaWAN 资源的操作和数据类型的完整列表，请参阅 Amazon IoT Wireless API 参考。

如何使用 Amazon CLI 添加一个目标

您可以使用 Amazon CLI 添加目标，方法是使用 create-destination 命令。下面的示例演示如何通过输入规则名称创建目标，并将 `RuleName` 用作 `expression-type` 参数的值。如果要指定用于发布或订阅消息代理的主题名称，请将 `expression-type` 参数值更改为`MqttTopic`。

```
aws iotwireless create-destination \
    --name IoTWirelessDestination \
    --expression-type RuleName \
    --expression IoTWirelessRule \
    --role-arn arn:aws:iam::123456789012:role/IoTWirelessDestinationRole
```

运行此命令可创建带有指定目标名称、规则名称和角色名称的目标。有关目标的规则和角色名称的信息，请参阅 创建规则以处理 LoRa WAN 设备消息 (p. 1126) 和 为您的目标创建 IAM 角色 (p. 1125)。

有关您可以使用的 CLI 的信息，请参阅 Amazon CLI 参考。

为您的目标创建 IAM 角色

Amazon IoT Core for LoRaWAN 目标需要 IAM 角色为 Amazon IoT Core for LoRaWAN 提供将数据发送到 Amazon IoT 规则所需的权限。如果尚未定义此类角色，必须对其进行定义，以便出现在角色列表中。

如本主题之前所述，当您使用控制台添加目标时，Amazon IoT Core for LoRaWAN 自动为您创建 IAM 角色。使用 API 或 CLI 添加目标时，必须为目标创建 IAM 角色。

要为您的 Amazon IoT Core for LoRaWAN 目标角色创建 IAM policy

1. 打开 IAM 控制台中的 Policies（策略）中心。
2. 选择 Create policy（创建策略），然后选择 JSON 选项卡。
3. 在编辑器中，删除编辑器中的所有内容并粘贴此策略文档。

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

4. 选择 Review policy（查看策略），并在 Name（名称）中输入此策略的名称。在下一流程中，您将需要使用此名称。

   如有需要，您还可以在 Description（说明）中描述此策略。
5. 选择 Create policy（创建策略）。

要为 Amazon IoT Core for LoRaWAN 目标创建 IAM 角色

1. 开启 IAM 控制台的角色中心，然后选择 Create role（创建角色）。
2. 在 Select type of trusted entity（选择受信任实体的类型）下，选择 Another Amazon Web Services 账户（另一个亚马逊云科技账户）。
3. 在 Account ID（账户 ID），请输入您的Amazon Web Services 账户 ID，然后选择 Next: Permission（下一步：权限）。
4. 在搜索框中，输入您在上一流程中创建的 IAM policy 名称。
5. 在搜索结果中，检查您在上一个流程中创建的 IAM policy。
6. 依次选择 Next: Tags（下一步：标签）和 Next: Review（下一步：查看）。
7. 在 Role name（角色名称），输入此角色的名称，然后选择 Create role（创建角色）。
8. 在确认消息中，选择您创建的角色的名称以编辑新角色。
9. 在 Summary（摘要）页面上，选择 Trust relationships（信任关系），然后选择 Edit trust relationship（编辑信任关系）。
10. 在 Policy Document（策略文档）中，更改 `Principal` 属性以使其类似于此示例。

```
"Principal": {
    "Service": "iotwireless.amazonaws.com"
},
```

在您更改 `Principal` 属性后，完整的策略文档应该如此示例所示。

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

11. 要保存更改并退出，请选择 Update Trust Policy（更新信任策略）。

定义了此角色后，您可以在配置 Amazon IoT Core for LoRaWAN 目标时在角色列表中找到它。

创建规则以处理 LoRa WAN 设备消息

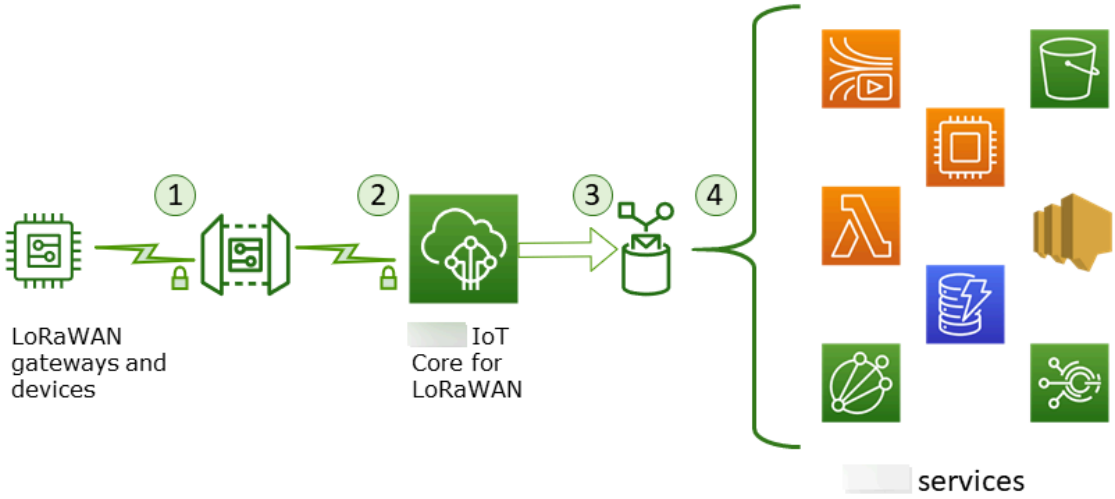
Amazon IoT规则将设备消息发送到其他服务。 Amazon IoT规则还可以处理从 LoRa WAN 设备接收的二进制消息，以便将消息转换为其它格式，从而更加便于其它服务使用。

Amazon IoT Core for LoRaWAN 目标 (p. 1124)将无线设备与规则相关联，该规则将处理要发送到其它服务的设备消息数据。该规则会在 Amazon IoT Core for LoRaWAN 收到数据时立即开始处理设备数据。Amazon IoT Core for LoRaWAN 目标 (p. 1124)可由消息具有相同数据格式并将数据发送到同一服务的所有设备共享。

Amazon IoT 规则如何处理设备消息

Amazon IoT 规则处理设备消息数据的方式取决于将接收数据的服务、设备消息数据的格式以及该服务所需的数据格式。通常，该规则会调用 Amazon Lambda 函数将设备的消息数据转换为服务所需的格式，然后将结果发送到服务。

以下示意图显示了在消息数据在从无线设备移动到Amazon服务的过程中是如何确保安全以及如何处理的。



1. LoRaWAN 无线设备在传输二进制消息之前会使用 AES128 CTR 模式对其进行加密。
2. Amazon IoT Core for LoRaWAN 会对二进制消息进行解密，并将解密的二进制消息负载编码为 base64 字符串。
3. 生成的 base64 编码消息将作为二进制消息负载（未格式化为 JSON 文档的消息负载）发送到分配给设备的任务中所描述的 Amazon IoT 规则。
4. Amazon IoT 规则将消息数据定向到规则配置中描述的服务。

从无线设备接收的加密二进制负载不会被 Amazon IoT Core for LoRaWAN 修改或解释。解密的二进制消息负载仅编码为 base64 字符串。要使服务访问二进制消息负载中的数据元素，必须通过规则调用的函数从负载中解析数据元素。base64 编码的消息负载是一个 ASCII 字符串，因此可以将其存储以便稍后进行解析。

为 LoRa WAN 创建规则

Amazon IoT Core for LoRaWAN 使用 Amazon IoT 规则来安全地将设备消息直接发送到其它Amazon服务，而无需使用消息代理。通过从摄取路径中删除消息代理，可以降低成本并优化数据流。

Amazon IoT Core for LoRaWAN 规则如要将设备消息发送到其它 Amazon 服务，则需要 Amazon IoT Core for LoRaWAN 目标，并要将 Amazon IoT 规则分配给该目标。Amazon IoT 规则必须包含 SQL 查询语句和至少一个规则操作。

通常，Amazon IoT 规则查询语句由以下内容组成：

- 一个 SQL SELECT 子句，用于从消息负载中选择数据并设置其格式
- 标识要使用的消息的主题筛选条件（规则查询语句中的 FROM 对象）
- 可选条件语句（SQL WHERE 子句），用于指定执行操作的条件

以下是规则查询语句示例：

```
SELECT temperature FROM iot/topic' WHERE temperature > 50
```

构建Amazon IoT规则以处理来自 LoRa WAN 设备的负载时，则不必将 FROM 子句指定为规则查询对象的一部分。规则查询语句必须具有 SQL SELECT 子句，并且可以包含或不包含 WHERE 子句。如果查询语句使用 FROM 子句，该子句会被忽略。

以下是一个可以处理来自 LoRa WAN 设备的负载的规则查询语句示例：

```
SELECT WirelessDeviceId, WirelessMetadata.LoRaWAN.FPort as FPort,
       WirelessMetadata.LoRaWAN.DevEui as DevEui,
       PayloadData
```

在此示例中，是由您PayloadData的 LoRa WAN 设备发送的 base64 编码二进制负载。

下面是一个示例规则查询语句，它可以对传入的有效负载执行二进制解码，并将其转换为不同的格式，如 JSON：

```
SELECT WirelessDeviceId, WirelessMetadata.LoRaWAN.FPort as FPort,
       WirelessMetadata.LoRaWAN.DevEui as DevEui,
       aws_lambda("arn:aws:lambda:<region>:<account>:function:<name>",

                  {
                   "PayloadData":PayloadData,
                   "Fport": WirelessMetadata.LoRaWAN.FPort
                  }) as decodingoutput
```

有关使用 SELECT AND WHERE 子句的更多信息，请参阅 Amazon IoT SQL 参考 (p. 530)

有关 Amazon IoT 规则以及如何创建和使用该规则的信息，请参阅 Amazon IoT 规则 (p. 442) 和 创建将设备数据路由到其他服务的 Amazon IoT规则 (p. 175)。

有关创建和使用 Amazon IoT Core for LoRaWAN 目标的信息，请参阅 Amazon IoT Core为 LoRa WAN 添加目的地 (p. 1124)。

有关在规则中使用二进制消息负载的信息，请参阅 使用二进制负载 (p. 587)。

有关用于在传输过程中保护消息负载的数据安全和加密技术的详细信息，请参阅 Amazon IoT Core 中的数据保护 (p. 345)。

有关显示 IoT 规则二进制解码和实现示例的参考体系架构，请参阅上的Amazon IoT Core for LoRaWAN解决方案示例 GitHub。

Connect 您的 LoRa WAN 设备并验证其连接状态

在检查设备连接状态之前，您必须已经添加设备并将其连接到 Amazon IoT Core for LoRaWAN。有关如何添加设备的信息，请参阅 将您的无线设备添加到 Amazon IoT Core for LoRaWAN (p. 1120)。

添加设备后，请参阅设备的用户手册，了解如何从 LoRa WAN 设备初始发送上行链路消息。

使用控制台检查设备连接状态

要使用控制台检查连接状态，请导航到 Amazon IoT 控制台的 Devices（设备）的页面，然后选择已添加的设备。在无线设备详细信息页面的 Details（详细信息）部分，您将看到上次接收上行链路的日期和时间。

使用 API 检查设备连接状态

要使用 API 检查连接状态，请使用 `GetWirelessDeviceStatistics` API。此 API 没有请求体，只包含显示最后一个上行链路何时收到的响应体。

> Note
>
> 收到最后一条上行链路的时间或 `LastUplinkReceivedAt` 值仅在 3 个月内有效。

```
HTTP/1.1 200
Content-type: application/json

{
  "LastUplinkReceivedAt": "2021-03-24T23:13:08.476015749Z",
  "LoRaWAN": {
        "DataRate": 5,
        "DevEui": "647fda0000006420",
        "Frequency": 868100000,
        "Gateways": [
         {
            "GatewayEui": "c0ee40ffff29df10",
            "Rssi": -67,
            "Snr": 9.75
         }
      ],
  "WirelessDeviceId": "30cbdcf3-86de-4291-bfab-5bfa2b12bad5"
}
```

后续步骤

现在，您已连接设备并验证了连接状态，您可以使用 Amazon IoT 控制台 Test（测试）页面的 MQTT 测试客户端观察从设备接收的上行链路元数据格式。有关更多信息，请参阅 查看从 LoRa WAN 设备发送的上行链路消息的格式 (p. 1162)。

使用 Amazon IoT Core for LoRaWAN 配置无线资源的位置

在使用此功能之前，请注意，用于解析 LoRaWAN 设备位置信息的所选第三方提供商依赖于国际 GNSS 服务 (IGS)、通过 NASA 的 EarthData 或其他第三方提供或维护的数据源和数据集。这些数据源和数据集是第三方内容（定义见客户协议），并按原样提供。有关更多信息，请参阅 Amazon 服务条款。

可以使用 Amazon IoT Core for LoRaWAN 指定您的静态位置数据，或使用第三方解算器激活定位以实时识别设备的位置。您可以添加或更新 LoRaWAN 设备和/或网关的位置信息。

在将设备或网关添加到 Amazon IoT Core for LoRaWAN 或编辑设备或网关的配置详细信息时，可以指定位置信息。位置信息被指定为 GeoJSON 有效负载。GeoJSON 格式是一种用于对地理数据结构进行编码的格式。此有效负载中包含设备位置的纬度和经度坐标，这些坐标基于世界大地坐标系统 (WGS84)。

求解器计算资源的位置后，如果您具有 Amazon Location Service，可以激活 Amazon Location 地图，其中将显示您的资源的位置。使用位置数据，您可以：

- 激活定位以识别和获取 LoRaWAN 设备的位置。
- 跟踪和监控网关和设备的位置。
- 定义用于处理对位置数据的任何更新并将其路由到其他 Amazon Web Service 的 Amazon IoT 规则。有关规则操作的列表，请参阅 Amazon IoT 规则操作 (p. 449)。
- 使用位置数据和 Amazon SNS 创建提醒，并在发生任何异常活动时向设备接收通知。

LoRaWAN 设备定位的工作原理

可以使用第三方 Wi-Fi 和 GNSS 求解器激活定位以识别设备的位置。可以使用此信息跟踪和监控设备。以下步骤向您展示如何激活定位和查看 LoRaWAN 设备的位置信息。

Note

第三方求解器只能与具有 LoRa Edge 芯片的 LoRaWAN 设备一起使用。不能与 LoRaWAN 网关一起使用。对于网关，您仍然可以指定静态位置信息，并在 Amazon Location 地图上识别该位置。

1. 添加设备

   在激活定位之前，先将设备添加到 Amazon IoT Core for LoRaWAN。LoRaWAN 设备必须具有 LoRa Edge 芯片，LoRa Edge 是一个超低功耗平台，集成了针对地理定位应用的远程 LoRa 收发器、多星座 GNSS 扫描仪和无源 Wi-Fi MAC 扫描仪。
2. 激活定位

   要获取设备的实时位置，请激活定位。当您的 LoRaWAN 设备发送上行链路消息时，会使用地理定位帧端口将消息中包含的 Wi-Fi 和 GNSS 扫描数据发送到 Amazon IoT Core for LoRaWAN。
3. 检索位置信息

   从基于来自收发器的扫描结果计算的求解器中检索估计的设备位置。如果位置信息是使用 Wi-Fi 和 GNSS 扫描结果计算出来的，Amazon IoT Core for LoRaWAN 会选择准确性较高的估计位置。
4. 查看位置信息

   解算器在计算位置信息后，还将提供准确性信息，该信息指示解算器计算的位置与您输入的静态位置信息之间的差异。您还可以在 Amazon Location 地图上查看设备位置。

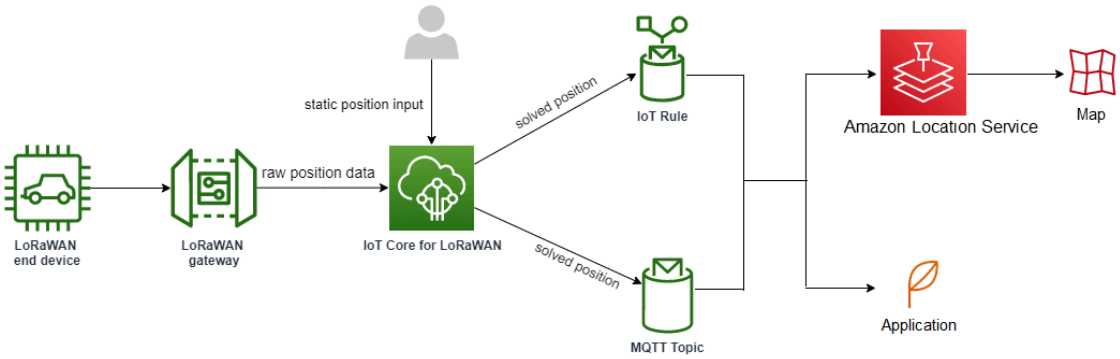
   Note

   由于解算器不能用于 LoRaWAN 网关，因此准确性信息将报告为 `0.0`。

有关用于定位解算器的上行链路消息格式和频率端口的更多信息，请参阅 从 Amazon IoT Core for LoRaWAN 到规则引擎的上行链路消息 (p. 1133)。

定位工作流概述

下图显示了 Amazon IoT Core for LoRaWAN 如何存储和更新设备和网关的位置信息。



1. 指定资源的静态位置

   使用纬度和经度坐标指定设备或网关的静态位置信息，作为 GeoJSON 有效负载。您还可以指定可选的高度坐标。这些坐标基于 WGS84 坐标系。有关更多信息，请参阅 World Geodetic System (WGS84) [世界大地测量系统（WGS84）]。
2. 激活设备定位

   如果您使用的是具有 LoRa Edge 芯片的 LoRaWAN 设备，则可以选择激活定位来实时跟踪您的设备位置。当您的设备发送上行链路消息时，会使用地理定位帧端口将 GNSS 和 Wi-Fi 扫描数据发送到 Amazon IoT Core for LoRaWAN。然后，求解器使用此信息来解析设备位置。

3. 添加目的地以路由位置数据

   您可以添加一个目标来描述用于处理设备数据的 IoT 规则，并将更新的位置信息路由到 Amazon IoT Core for LoRaWAN。您还可以在 Amazon Location 地图上查看资源的上次已知位置。

配置资源位置

您可以使用 Amazon Web Services Management Console、Amazon IoT Wireless API 或 Amazon CLI 配置资源的位置。

如果您的设备具有 LoRa Edge 芯片，您可以激活定位来计算实时位置信息。对于您的网关，您仍然可以输入静态位置坐标并使用 Amazon Location 在 Amazon Location 地图上跟踪网关位置。

配置 LoRaWAN 网关的位置

将网关添加到 Amazon IoT Core for LoRaWAN 时，您可以指定静态位置数据。如果您激活了 Amazon Location Service 地图，位置数据将显示在 Amazon Location 地图上。

> Note
>
> 第三方求解器不能与 LoRaWAN 网关一起使用。对于网关，您仍然可以指定静态位置坐标。如果不使用解算器来计算位置（例如，在网关的情况下），准确度信息将报告为 `0.0`。

您可以使用 Amazon Web Services Management Console、Amazon IoT Wireless API 或 Amazon CLI 配置网关位置。

使用控制台配置网关的位置

要使用 Amazon Web Services Management Console 配置网关资源的位置，请先登录控制台，然后转到 Amazon IoT 控制台的 Gateways（网关）中心页面。

添加位置信息

为网关添加位置配置

1. 在 Gateways（网关）中心页面中，选择 Add gateway（添加网关）。
2. 输入网关的 EUI、频段（RFRegion）以及任何其他网关详细信息和 LoRaWAN 配置信息。有关更多信息，请参阅使用控制台添加网关 (p. 1114)。
3. 转到 Position information - Optional（位置信息 - 可选），使用纬度和经度坐标以及可选的海拔坐标输入网关的位置信息。位置信息基于 WGS84 坐标系。

查看网关的位置

配置网关的位置后，Amazon IoT Core for LoRaWAN 会创建一个称为 `iotwireless.map` 的 Amazon Location 地图。您可以在网关详细信息页面中的 Position（位置）选项卡上查看此地图。根据您指定的位置坐标，网关的位置将在地图上显示为标记。您可以放大或缩小以在地图上清晰地查看网关的位置。在 Position（位置）选项卡上，您还将看到准确性信息，以及确定网关位置的时间戳。

> Note
>
> 如果没有安装 Amazon Location Service 地图，您将看到一条消息，指示您必须使用 Amazon Location Service 才能访问地图并查看网关位置。使用 Amazon Location Service 地图可能会对您的 Amazon Web Services 账户 额外计费。有关更多信息，请参阅 Amazon IoT Core 定价。

地图 `iotwireless.map` 用作可使用 `Get` API 操作（`GetMapTile`）访问的地图数据的来源。有关与地图一起使用的 `Get` API 的信息，请参阅 Amazon Location Service API 参考。

要获取有关此地图的其他详细信息，请转到 Amazon Location Service 控制台，选择 maps（地图），然后选择 iotwireless.map。有关更多信息，请参阅《Amazon Location Service 开发人员指南》中的地图。

更新网关的位置配置

要更改网关的位置配置，请在网关详细信息页面中选择 Edit（编辑），然后更新位置信息和目标。

Note

有关历史位置数据的信息不可用。当您更新网关的位置坐标时，它将覆盖以前报告的位置数据。更新位置后，在网关详细信息的 Position（位置）选项卡中，您将看到新的位置信息。时间戳的更改表示它对应于网关的上次已知位置。

使用 API 配置网关的位置

您可以指定位置信息，并使用 Amazon IoT Wireless API 或 Amazon CLI 配置网关位置。

Important

不再支持 API 操作 UpdatePosition、GetPosition、PutPositionConfiguration、GetPositionConfiguration 和 ListPositionConfigurations。更新和检索位置信息的调用应改用 GetResourcePosition 和 UpdateResourcePosition API 操作。

添加位置信息

要添加给定无线网关的静态位置信息，请使用 UpdateResourcePosition API 操作或 update-resource-position CLI 命令指定坐标。将 `WirelessGateway` 指定为 `ResourceType`，将要更新的无线网关 ID 指定为 `ResourceIdentifier`，并将位置信息指定为 GeoJSON 有效负载。

```
aws iotwireless update-resource-position \
    --resource-type WirelessGateway \
    --resource-id "12345678-a1b2-3c45-67d8-e90fa1b2c34d" \
    --cli-input-json file://gatewayposition.json
```

下面显示的是 *gatewayposition.json* 文件的内容。

gatewayposition.json 的内容

```
{
     "type": "Point",
     "coordinates": [33.3318, -22.2155, 13.123],
     "properties": {
          "timestamp": "2018-11-30T18:35:24Z"
      }
}
```

运行此命令不会生成任何输出。要查看您指定的位置信息，请使用 `GetResourcePosition` API 操作。

获取位置信息

要获取给定无线网关的位置信息，请使用 GetResourcePosition API 操作或 get-resource-position CLI 命令。将 `WirelessGateway` 指定为 `resourceType`，并提供无线网关的 ID 作为 `resourceIdentifier`。

```
aws iotwireless get-resource-position \
    --resource-type WirelessGateway \
```

```
    --resource-id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

运行此命令会显示无线网关的位置信息，作为 GeoJSON 有效负载。您将看到有关位置坐标、位置信息类型和其他属性（例如与网关的上次已知位置相对应的时间戳）的信息。

```
{
    {
    "type": "Point",
     "coordinates": [33.3318,-22.2155,13.123],
     "properties": {
         "timestamp": "2018-11-30T18:35:24Z"
         }
    }
}
```

配置 LoRaWAN 设备的位置

将设备添加到 Amazon IoT Core for LoRaWAN 时，您可以指定静态位置信息，也可以激活定位，并指定目标。目标描述用于处理设备位置信息并将更新的位置路由到 Amazon Location Service 的 IoT 规则。配置设备位置后，位置数据将显示在 Amazon Location 地图上，其中包含准确性信息和您指定的目标。

您可以使用 Amazon Web Services Management Console、Amazon IoT Wireless API 或 Amazon CLI 配置设备的位置。

上行链路消息的帧端口和格式

如果激活定位，则必须指定用于将 Wi-Fi 和 GNSS 扫描数据从设备传输到 Amazon IoT Core for LoRaWAN 的地理定位帧端口。系统会使用此帧端口将此位置信息传输给 Amazon IoT Core for LoRaWAN。

LoRaWAN 规范提供了数据传输字段（FRMPayload）和端口字段（FPort）来区分不同类型的消息。要传输位置信息，您可以为帧端口指定一个介于 1 到 223 之间的任意值。FPort 0 为 MAC 消息预留，FPort 224 为 MAC 合规性测试预留，端口 225-255 为未来标准化应用扩展预留。

从 Amazon IoT Core for LoRaWAN 到规则引擎的上行链路消息

当您添加目标时，它会创建一个 Amazon IoT 规则，以使用规则引擎将数据路由到 Amazon Location Service。然后，更新的位置信息将显示在 Amazon Location 地图上。如果您尚未激活，则在您更新设备的静态位置坐标时，目标将路由位置数据。

以下代码显示了从 Amazon IoT Core for LoRaWAN 发送的上行链路消息的格式，其中包含位置信息、准确度、解算器配置和无线元数据。下面突出显示的字段是可选的。如果没有垂直准确性信息，则此值为 `null`。

```
{
    // Position configuration parameters for given wireless device
    "WirelessDeviceId": "5b58245e-146c-4c30-9703-0ca942e3ff35",

    // Position information for a device in GeoJSON format. Altitude
    // is optional. If no vertical accuracy information is available
    // or positioning isn't activated, the value is set to null.
    // The position information coordinates are listed in the order
    // [longitude, latitude, altitude].
    "coordinates": [33.33000183105469, -22.219999313354492, 99.0],
    "type": "Point",
    "properties": {
         "horizontalAccuracy": number,
         "verticalAccuracy": number",
         "timestamp": "2022-08-19T03:08:35.061Z"
```

```
    },

    //Parameters controlled by IoT Core for LoRaWAN
    "WirelessMetadata":
    {
        "LoRaWAN":
        {
            "ADR": false,
            "Bandwidth": 125,
            "ClassB": false,
            "CodeRate": "4/5",
            "DataRate": "0",
            "DevAddr": "00b96cd4",
            "DevEui": "58a0cb000202c99",
            "FOptLen": 2,
            "FCnt": 1,
            "Fport": 136,
            "Frequency": "868100000",
            "Gateways": [
             {
                    "GatewayEui": "80029cfffe5cf1cc",
                    "Snr": -29,
                    "Rssi": 9.75
             }
             ],
            "MIC": "7255cb07",
            "MType": "UnconfirmedDataUp",
            "Major": "LoRaWANR1",
            "Modulation": "LORA",
            "PolarizationInversion": false,
            "SpreadingFactor": 12,
            "Timestamp": "2021-05-03T03:24:29Z"

        }
    }
}
```

使用控制台配置设备的位置

要使用 Amazon Web Services Management Console配置和管理设备位置，请先登录控制台，然后转到 Amazon IoT 控制台的 Devices（ 设备 ）中心页面。

添加位置信息

添加设备的位置信息：

1. 在 Devices（设备）中心页面中，选择 Add wireless device（添加无线设备）。
2. 输入无线设备规格、设备和服务配置文件以及用于定义将数据路由到其他 Amazon Web Service 的 IoT 规则的目标。有关更多信息，请参阅将您的设备搭载到 Amazon IoT Core for LoRaWAN 上 (p. 1119)。
3. 输入位置信息，可以选择激活地理位置，并指定要用于路由消息的位置数据目标。
   - 位置信息

     使用纬度和经度坐标以及可选的高度坐标指定设备的位置数据。位置信息基于 WGS84 坐标系。
   - 地理位置

     如果您希望 Amazon IoT Core for LoRaWAN 使用地理位置来计算设备位置，请激活定位。它使用第三方 GNSS 和 Wi-Fi 解算器实时识别设备位置。

     要输入地理位置信息，请选择 Activate positioning（激活定位），然后输入用于向 Amazon IoT Core for LoRaWAN 传输 GNSS 和 Wi-Fi 扫描数据的地理位置帧端口。您将看到填充的默认 FPort 供您参考。但是，您可以选择 1 到 223 之间的其他任意值。

- 位置数据目标

  选择一个目标以描述用于处理设备位置数据并将此数据转发到 Amazon IoT Core for LoRaWAN 的 Amazon IoT 规则。此目标仅用于路由位置数据。它必须不同于您用于将设备数据路由到其他 Amazon Web Service 的目标。

查看设备的位置配置

配置设备的位置后，Amazon IoT Core for LoRaWAN 会创建一个称为 `iotwireless.map` 的 Amazon Location 地图。您可以在设备详细信息页面中的 Position（位置）选项卡上查看此地图。根据您指定的位置坐标或第三方求解器计算的位置，设备的位置将在地图上显示为标记。您可以放大或缩小以在地图上清晰地查看设备的位置。在设备详细信息页面中的 Position（位置）选项卡上，您还将看到准确度信息、确定设备位置的时间戳以及您指定的位置数据目标。

Note

如果尚未激活 Amazon Location Service 地图，您将看到一条消息，指示您必须使用 Amazon Location Service 才能访问地图并查看位置。使用 Amazon Location Service 地图可能会对您的 Amazon Web Services 账户 额外计费。有关更多信息，请参阅 Amazon IoT Core 定价。

地图 `iotwireless.map` 用作可使用 `Get` API 操作（`GetMapTile`）访问的地图数据的来源。有关与地图一起使用的 `Get` API 的信息，请参阅 Amazon Location Service API 参考。

要获取有关此地图的其他详细信息，请转到 Amazon Location Service 控制台，选择 maps（地图），然后选择 iotwireless.map。有关更多信息，请参阅《Amazon Location Service 开发人员指南》中的地图。

更新设备的位置配置

要更改设备的位置配置，请在设备详细信息页面中选择 Edit（编辑），然后更新位置信息、任何地理位置设置和目标。

Note

有关历史位置数据的信息不可用。当您更新设备的位置坐标时，它将覆盖以前报告的位置数据。更新位置后，在设备详细信息的 Position（位置）选项卡中，您将看到新的位置信息。时间戳的更改表示它对应于设备的上次已知位置。

使用 API 配置设备位置

您可以使用 Amazon IoT Wireless API 或 Amazon CLI 指定位置信息、配置设备位置，以及激活可选的地理位置。

Important

不再支持 API 操作 UpdatePosition、GetPosition、PutPositionConfiguration、GetPositionConfiguration 和 ListPositionConfigurations。更新和检索位置信息的调用应改用 GetResourcePosition 和 UpdateResourcePosition API 操作。

添加位置信息和配置

要添加给定无线设备的位置信息，请使用 UpdateResourcePosition API 操作或 update-resource-position CLI 命令指定坐标。将 `WirelessDevice` 指定为 `ResourceType`，将要更新的无线设备 ID 指定为 `ResourceIdentifier` 和位置信息。

```
aws iotwireless update-resource-position \
    --resource-type WirelessDevice \
```

```
    --resource-id "1ffd32c8-8130-4194-96df-622f072a315f" \
    --position [33.33, -33.33, 10.0]
```

下面显示的是 *deviceposition.json* 文件的内容。要指定用于发送地理位置数据的 FPort 值，请将定位对象与 CreateWirelessDevice 和 UpdateWirelessDevice API 操作结合使用。

deviceposition.json 的内容

```
{
     "type": "Point",
     "coordinates": [33.3318, -22.2155, 13.123],
     "properties": {
          "verticalAccuracy": 707,
          "horizontalAccuracy":
          "timestamp": "2018-11-30T18:35:24Z"
      }
}
```

运行此命令不会生成任何输出。要查看您指定的位置信息，请使用 `GetResourcePosition` API 操作。

获取位置信息和配置

要获取给定无线设备的位置信息，请使用 GetResourcePosition API 或 get-resource-position CLI 命令。将 `WirelessDevice` 指定为 `resourceType`，并提供无线设备的 ID 作为 `resourceIdentifier`。

```
aws iotwireless get-resource-position \
    --resource-type WirelessDevice \
    --resource-id "1ffd32c8-8130-4194-96df-622f072a315f"
```

运行此命令会将无线设备的位置信息显示为 GeoJSON 有效负载。您将看到有关位置坐标、位置类型和属性的信息，这些信息可能包括准确性信息和对应于设备最后一个已知位置的时间戳。

```
{
     "type": "Point",
     "coordinates": [33.3318, -22.2155, 13.123],
     "properties": {
          "verticalAccuracy": 707,
          "horizontalAccuracy": 389,
          "horizontalConfidenceLevel": 0.68,
          "verticalConfidenceLevel": 0.68,
          "timestamp": "2018-11-30T18:35:24Z"
      }
}
```

通过 VPC 接口端点连接到 Amazon IoT Core for LoRaWAN

您可以通过 Virtual Private Cloud (VPC) 中的 接口 VPC 终端节点 (Amazon PrivateLink) 直接连接到 Amazon IoT Core for LoRaWAN，而不是通过公共互联网进行连接。当您使用 VPC 接口端点时，VPC 与 Amazon IoT Core for LoRaWAN 之间的通信完全在 Amazon 网络内安全进行。

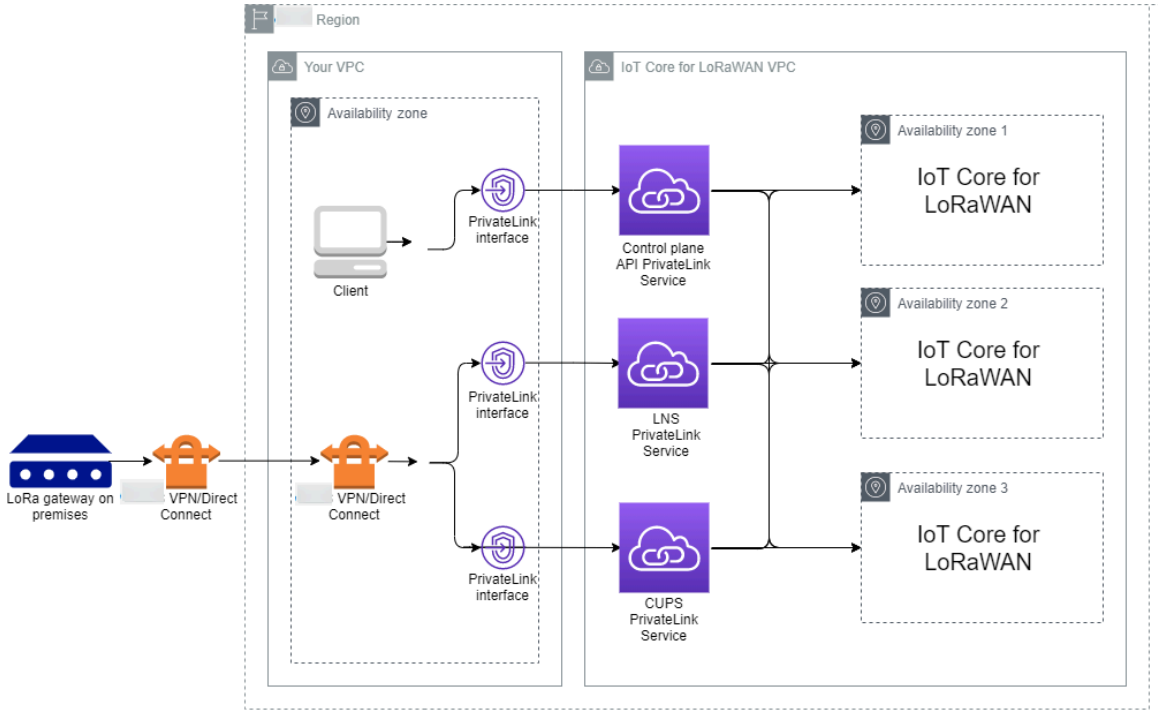
Amazon IoT Core for LoRaWAN 支持 Amazon Virtual Private Cloud 接口端点，这些端点由 Amazon PrivateLink 提供支持。每个 VPC 终端节点都由您的 VPC 子网中一个或多个使用私有 IP 地址的弹性网络接口 (ENI) 代表。

有关 VPC 和端点的更多信息，请参阅什么是 Amazon VPC。

有关 Amazon PrivateLink 的更多信息，请参阅 Amazon PrivateLink 和 VPC 终端节点。

Amazon IoT Core for LoRaWAN 私有链接架构

下图显示了 Amazon IoT Core for LoRaWAN 私有链路架构 该架构使用 Transit Gateway 和 Route 53 Resolver 在您的 VPC、Amazon IoT Core for LoRaWAN VPC 以及本地部署环境之间共享 Amazon PrivateLink 接口端点。在设置与 VPC 接口端点的连接时，您将找到更详细的架构图。



Amazon IoT Core for LoRaWAN 端点

Amazon IoT Core for LoRaWAN 有三个公有端点。每个公有端点都有相应的 VPC 接口端点。公有端点可分为控制层面和数据层面端点。有关这些端点的信息，请参阅 Amazon IoT Core for LoRaWAN API 端点。

> **Note**
>
> Amazon PrivateLink 对端点的支持仅在美国东部（弗吉尼亚北部）和欧洲（爱尔兰）提供。

- 控制层面 API 端点

  您可以使用控制层面 API 端点与 Amazon IoT Wireless API 交互。这些端点可以使用 Amazon PrivateLink 从托管在 Amazon VPC 中的客户端进行访问。
- 数据层面 API 端点

  数据面板 API 端点是 LoRaWAN Network Server（LNS）和 Configuration and Update Server（CUPS）端点，您可以使用它们与 Amazon IoT Core for LoRaWAN LNS 和 CUPS 端点进行交互。这些端点可以使用 Amazon VPN 或 Amazon Direct Connect 从本地部署的 LoRa 网关访问。您可以在将网关搭载到 Amazon IoT Core for LoRaWAN 时获取这些端点。有关更多信息，请参阅将网关添加到 Amazon IoT Core for LoRaWAN (p. 1114)。

以下主题介绍如何搭载这些端点。

主题

- 搭载 Amazon IoT Core for LoRaWAN 控制层面 API 端点 (p. 1138)
- 搭载 Amazon IoT Core for LoRaWAN 数据层面 API 端点 (p. 1140)

搭载 Amazon IoT Core for LoRaWAN 控制层面 API 端点

您可以使用 Amazon IoT Core for LoRaWAN 控制层面 API 端点与 Amazon IoT Wireless API 交互。例如，您可以使用此端点运行 SendDataToWirelessDevice API 以将数据从 Amazon IoT 发送到 LoRaWAN 设备。有关更多信息，请参阅Amazon IoT Core for LoRaWAN 控制层面 API 端点。

您可以使用 Amazon VPC 中托管的客户端访问由 Amazon PrivateLink 提供支持的控制层面端点。您可以通过 Virtual Private Cloud (VPC) 中的接口端点直接连接到 Amazon IoT Wireless API，而不是通过公共互联网进行连接。

要搭载控制层面端点：

- 创建您的 Amazon VPC 和子网 (p. 1138)
- 将 Amazon EC2 实例启动到您的子网中 (p. 1138)
- 创建 Amazon VPC 接口端点 (p. 1139)
- 测试与接口端点的连接 (p. 1139)

创建您的 Amazon VPC 和子网

您必须先创建 VPC 和子网，然后才能连接到接口端点。然后，您将在子网中启动 EC2 实例，您可以使用该实例连接到接口端点。

要创建您的 VPC：

1. 导航 Amazon VPC 控制台的 VPC 页面上，然后选择 Create VPC（创建 VPC）。
2. 在 Create VPC（创建 VPC）页面：
   - 为 VPC Name tag - optional（VPC 名称标签 — 可选）输入名称（例如，**VPC-A**）。
   - 在 IPv4 CIDR 块中输入 VPC 的 IPv4 地址范围（例如 **10.100.0.0/16**）。
3. 保留其它字段的默认值，然后选择 Create VPC（创建 VPC）。

要创建子网：

1. 导航 Amazon VPC 控制台的 Subnets（子网）页面上，然后选择 Create subnet（创建子网）。
2. 在 Create subnet（创建子网）页面中：
   - 对于 VPC ID，请选择之前创建的 VPC（例如，VPC-A）。
   - 在 Subnet name（子网名称）中输入名称（例如，**Private subnet**）。
   - 为您的子网选择可用区。
   - 在 IPv4 CIDR 块中输入子网的 IP 地址数据块（例如，**10.100.0.0/24**）。
3. 要创建子网并将其添加到 VPC 中，请选择 Create subnet（创建子网）。

有关更多信息，请参阅使用 VPC 和子网。

将 Amazon EC2 实例启动到您的子网中

要启动 EC2 实例：

1. 导航到 Amazon EC2 控制台并选择 Launch Instance（启动实例）。
2. 对于 AMI，请选择 Amazon Linux 2 AMI (HVM)，SSD Volume Type，然后选择 t2 micro 实例类型。要配置实例详情，请选择 Next（下一步）。
3. 在 Configure Instance Details（配置实例详细信息）页面：
   - 对于 Network（网络），请选择您之前创建的 VPC（例如：`VPC-A`）。
   - 对于 Subnet（子网），请选择您之前创建的子网（例如，**`Private subnet`**）。
   - 对于 IAM role（IAM 角色），请选择角色 AWSIoTWirelessFullAccess 以授予 Amazon IoT Core for LoRaWAN 完全访问策略。有关更多信息，请参阅 `AWSIoTWirelessFullAccess` 策略摘要。
   - 对于假定私有 IP，请使用 IP 地址，例如 10.100.0.42。
4. 选择 Next: Add Storage（下一步：添加存储），然后选择 Next: Add Tags（下一步：添加标签）。您可以选择添加要与 EC2 实例关联的任何标签。选择 Next: Configure Security Group。
5. 在 Configure Security Group（配置安全组）页面上，将安全组配置为允许执行以下操作：
   - 对源 `10.200.0.0/16` 打开 All TCP。
   - 对源 `10.200.0.0/16` 打开 All ICMP - IPV4。
6. 要查看实例详细信息并启动 EC2 实例，请选择 Review and Launch（审核和启动）。

有关更多信息，请参阅 Amazon EC2 Linux 实例入门。

创建 Amazon VPC 接口端点

您可以为您的 VPC 创建 VPC 终端节点，然后可以通过 EC2 API 访问该端点。要创建端点：

1. 导航到 VPC 终端节点控制台并选择 Create Endpoint（创建端点）。
2. 在 Create Endpoint（创建终端节点）页面上，指定以下信息。
   - 为 Service category（服务类别）选择Amazon Web Service。
   - 对于 Service Name（服务名称），通过输入关键字 **`iotwireless`** 进行搜索。在显示的 `iotwireless` 服务列表中，请为您的区域选择控制层面 API 端点。端点的格式为 `com.amazonaws.`*`region`*`.iotwireless.api`。
   - 对于 VPC 和 Subnets（子网），选择要在其中创建端点的 VPC 和要在其中创建端点网络的可用区 (AZ)。

     Note

     并非所有可用区都支持 `iotwireless` 服务。
   - 对于 Enable DNS Name（启用 DNS 名称），请选择 Enable for this endpoint（为此端点启用）。

     选择此选项将自动解析 DNS 并在 Amazon Route 53 Public Data Plane 创建一个路由，以便稍后用于测试连接的 API 能够通过私密链路端点。
   - 对于 Security group（安全组），选择要与端点网络接口关联的安全组。
   - 您可以选择添加或删除标签。标签是用于与端点关联的名称-值对。
3. 要创建 VPC 终端节点，请选择 Create endpoint（创建端点）。

测试与接口端点的连接

您可以使用 SSH 访问您的 Amazon EC2 实例，然后使用 Amazon CLI 连接到私有链路接口端点。

在连接到接口端点之前，请根据在 Linux 上安装、更新和卸载 Amazon CLI 版本 2 中所描述的方法下载最新的 Amazon CLI 版本。

以下示例介绍如何使用 CLI 测试到接口端点的连接。

```
aws iotwireless create-service-profile \
    --endpoint-url https://api.iotwireless.region.amazonaws.com  \
    --name='test-privatelink'
```

以下是运行命令的示例。

```
Response:
{
 "Arn": "arn:aws:iotwireless:region:acct_number:ServiceProfile/1a2345ba-4c5d-67b0-ab67-
e0c8342f2857",
 "Id": "1a2345ba-4c5d-67b0-ab67-e0c8342f2857"
}
```

同样，您可以运行以下命令来获取服务配置文件信息或列出所有服务配置文件。

```
aws iotwireless get-service-profile \
    --endpoint-url https://api.iotwireless.region.amazonaws.com
    --id="1a2345ba-4c5d-67b0-ab67-e0c8342f2857"
```

以下是 list-device-profiles 命令的示例。

```
aws iotwireless list-device-profiles \
    --endpoint-url https://api.iotwireless.region.amazonaws.com
```

搭载 Amazon IoT Core for LoRaWAN 数据层面 API 端点

Amazon IoT Core for LoRaWAN 数据层面端点由以下端点组成。将网关添加到 Amazon IoT Core for LoRaWAN 时，您将获得这些端点。有关更多信息，请参阅将网关添加到 Amazon IoT Core for LoRaWAN (p. 1114)。

- LoRaWAN 网络服务器 (LNS) 端点

  LNS 端点的格式为 *account-specific-prefix*.lns.lorawan.*region*.amazonaws.com。您可以使用此端点建立用于交换 LoRa 上行链路和下行链路消息的连接。
- Configuration and Update Server (CUPS) 端点

  CUPS 端点的格式为 *account-specific-prefix*.cups.lorawan.*region*.amazonaws.com。您可以将此端点用于网关的凭证管理、远程配置和固件更新。

有关更多信息，请参阅使用 CUPS 和 LNS 协议 (p. 1146)。

要查找适用于您的 Amazon Web Services 账户 和区域的数据层面 API 端点，请使用在此显示的 get-service-endpoint CLI 命令，或是 GetServiceEndpoint REST API。有关更多信息，请参阅Amazon IoT Core for LoRaWAN 数据层面 API 端点。

您可以在本地部署连接 LoRaWAN 网关，以便与 Amazon IoT Core for LoRaWAN 端点通信。要建立此连接，请首先使用 VPN 连接将本地部署网关连接到您 VPC 中的Amazon Web Services 账户。然后，您可以与 Amazon IoT Core for LoRaWAN VPC 中的数据层面接口端点通信，该 VPC 由私有链路提供支持。

以下是如何搭载这些端点的演示。

- 创建 VPC 接口端点和私有托管区域 (p. 1141)
- 使用 VPN 以将 LoRa 网关连接到Amazon Web Services 账户。 (p. 1143)

创建 VPC 接口端点和私有托管区域

Amazon IoT Core for LoRaWAN 有两个数据层面端点： Configuration and Update Server (CUPS) 端点和 LoRaWAN Network Server (LNS) 端点。建立到两个端点的私有链路连接的设置过程是相同的，因此我们可以使用 LNS 端点进行说明。

对于数据层面端点，LoRa 网关首先连接到您在 Amazon VPC 中的 Amazon Web Services 账户，然后它会连接到 Amazon IoT Core for LoRaWAN VPC 中的 VPC 终端节点。

连接到终端节点时，DNS 名称可以在一个 VPC 内解析，但无法跨多个 VPC 解析。要在创建端点时禁用私有 DNS，请禁用 Enable DNS name（启用 DNS 名称）设置。您可以使用私有托管区域提供有关您希望 Route 53 响应 VPC 的 DNS 查询的信息。要与本地部署环境共享您的 VPC，您可以使用 Route 53 Resolver 来支持混合 DNS。

要完成本流程，请执行以下步骤。

- 创建 Amazon VPC 和子网 (p. 1141)
- 创建 Amazon VPC 接口端点 (p. 1141)
- 配置私有托管区域 (p. 1142)
- 配置 Route 53 入站解析程序 (p. 1142)
- 后续步骤 (p. 1143)

创建 Amazon VPC 和子网

您可以重复使用您在搭载控制层面端点时创建的 Amazon VPC 和子网。有关信息，请参阅 创建您的 Amazon VPC 和子网 (p. 1138)。

创建 Amazon VPC 接口端点

您可以为 VPC 创建 VPC 终端节点，这与为控制层面终端节点创建 VPC 终端节点的方式类似。

1. 导航到 VPC 终端节点控制台并选择 Create Endpoint（创建端点）。
2. 在 Create Endpoint（创建终端节点）页面上，指定以下信息。
   - 为 Service category（服务类别）选择Amazon Web Service。
   - 对于 Service Name（服务名称），通过输入关键字 **lns** 进行搜索。在显示的 `lns` 服务列表中，请选择您所在区域的 LNS 数据层面 API 端点。端点的格式为 `com.amazonaws.`*`region`*`.lorawan.lns`。

     Note

     如果您正在为 CUPS 端点执行此流程，请搜索 `cups`。端点的格式为 `com.amazonaws.`*`region`*`.lorawan.cups`。
   - 对于 VPC 和 Subnets（子网），选择要在其中创建端点的 VPC 和要在其中创建端点网络的可用区 (AZ)。

     Note

     并非所有可用区都支持 `iotwireless` 服务。
   - 对于 Enable DNS name（启用 DNS 名称），请确保未选择 Enable for this endpoint（为此终端节点启用）。

     如果不选择此选项，您可以禁用 VPC 终端节点的私有 DNS，并改为使用私有托管区域。
   - 对于 Security group（安全组），选择要与端点网络接口关联的安全组。
   - 您可以选择添加或删除标签。标签是用于与端点关联的名称-值对。
3. 要创建 VPC 终端节点，请选择 Create endpoint（创建端点）。

配置私有托管区域

创建私有链路端点后，在端点的 Details（详细信息）选项卡下，您将看到 DNS 名称列表。您可以使用这些 DNS 名称来配置私有托管区域。DNS 名称将采用 `vpce-xxxx.lns.lorawan.region.vpce.amazonaws.com` 格式。

创建私有托管区域

要创建私有托管区域：

1. 导航到 Route 53 托管区域控制台并选择 Create hosted zone（创建托管区域）。
2. 在 Create hosted zone（创建托管区域）页面上，指定以下信息。
   - 对于 Domain name（域名），输入 LNS 端点的完整服务名称，**`lns.lorawan.region.amazonaws.com`**。

     Note

     如果您正在为 CUPS 端点执行此流程，请输入 **`cups.lorawan.region.amazonaws.com`**。
   - 对于 Type（类型），选择 Private hosted zone（私有托管区域）。
   - 或者，您可以添加或删除要与托管区域关联的标签。
3. 要创建您的私有托管区域，请选择 Create hosted zone（创建托管区域）。

有关更多信息，请参阅创建私有托管区域。

创建私有托管区域后，您可以创建一个记录，告诉 DNS 如何将流量路由到该域。

创建记录

创建私有托管区域后，您可以创建一个记录，告诉 DNS 如何将流量路由到该域。要创建记录：

1. 在显示的托管区域列表中，选择您之前创建的私有托管区域，然后选择 Create record（创建记录）。
2. 使用向导提供的方法创建记录。如果控制台为您提供了 Quick create（快速创建）的方法，选择 Switch to wizard（切换到向导）。
3. 在 Routing policy（路由策略）中选择 Simple Routing（简单路由），然后选择 Next（下一步）。
4. 在 Configure records（配置记录）下，选择 Define simple record（定义简单记录）。
5. 在 Define simple record（定义简单记录）页面:
   - 对于 Record name（记录名称），输入您的Amazon Web Services 账户账号别名。您可以在搭载网关时获得此值，或者通过使用 `GetServiceEndpoint` REST API 获取此值。
   - 对于 Record type（记录类型），请将该值保持为 `A - Routes traffic to an IPv4 address and some Amazon resources`。
   - 对于 Value/Route traffic to（值/流量路由至），选择 Alias to VPC endpoint（向 VPC 添加别名）。然后选择您的 Region（区域），接着从显示的端点列表中选择您之前创建的端点，如 创建 Amazon VPC 接口端点 (p. 1141) 中所述。
6. 选择 Define simple record（定义简单记录）以创建您的记录。

配置 Route 53 入站解析程序

若要将 VPC 终端节点共享到本地部署环境，可以使用 Route 53 Resolver 来支持混合 DNS。入站解析程序将允许您将流量从本地部署网络路由到数据层面端点，而无需通过公共互联网。要为服务返回私有 IP 地址值，请在 VPC 终端节点所在的 VPC 中创建 Route 53 Resolver。

创建入站解析程序时，您只需指定您之前在可用区 (AZ) 中创建的 VPC 以及子网即可。Route 53 Resolver 使用此信息自动分配 IP 地址，将流量路由到每个子网。

要创建入站解析程序：

1. 导航到 Route 53 Inbound endpoint（Route 53 入站端点）控制台并选择 Create inbound endpoint（创建入站端点）。

   Note

   请确保您使用的是与您在创建端点和私有托管区域时相同的Amazon Web Services 区域。
2. 在 Create inbound endpoint（创建入站端点）页面，请指定以下信息。
   - 为 Endpoint name（端点名称）输入名称（例如，**`VPC_A_Test`**）。
   - 对于 VPC in the region（该区域中的 VPC），选择您在创建 VPC 终端节点时使用的相同 VPC。
   - 配置 Security group for this endpoint（适用于此端点的安全组）以允许来自本地部署网络的传入流量。
   - 对于 IP 地址，选择 Use an IP adress that is selected automatically（使用自动选择的 IP 地址）。
3. 选择 Submit（提交）以创建您的入站解析程序。

在这个例子中，让我们假设 IP 地址 `10.100.0.145` 和 `10.100.192.10` 被分配给了路由流量的入站 Route 53 Resolver。

后续步骤

您已创建私有托管区域和入站解析程序，以路由 DNS 条目的流量。现在，您可以使用 Site-to-Site VPN 或 Client VPN 端点。有关更多信息，请参阅使用 VPN 以将 LoRa 网关连接到Amazon Web Services 账户。 (p. 1143)。

使用 VPN 以将 LoRa 网关连接到Amazon Web Services 账户。

要将本地部署网关连接到您的Amazon Web Services 账户，可以使用 Site-to-Site VPN 连接或 Client VPN 端点。

在连接本地部署网关之前，您必须已创建 VPC 终端节点，并配置私有托管区域和入站解析程序，以避免来自网关的流量通过公共互联网。有关更多信息，请参阅创建 VPC 接口端点和私有托管区域 (p. 1141)。

Site-to-Site VPN 端点

如果您没有网关硬件或希望使用其它Amazon Web Services 账户测试 VPN 连接，则可以使用 Site-to-Site VPN 连接。您可以使用 Site-to-Site VPN 连接来自同一个Amazon Web Services 账户的 VPC 终端节点；或如果您打算在不同Amazon Web Services 区域中使用，则可以选择来自其它Amazon Web Services 账户的 VPC 终端节点。

Note

如果您拥有网关硬件并希望建立 VPN 连接，我们建议您改用 Client VPN。有关说明，请参阅 Client VPN 终端节点 (p. 1144)。

要设置 Site-to-Site VPN：

1. 在要从中设置连接的站点中创建另一个 VPC。对于 `VPC-A`，您可以重复使用之前创建的 VPC。要创建另一个 VPC（例如，`VPC-B`），请使用与您之前创建的 VPC 的 CIDR 块不重叠的 CIDR 块。

   有关设置 VPC 的信息，请按照Amazon设置 Site-to-Site VPN 连接中所述操作执行。

   Note

   文档中描述的 Site-to-Site VPN VPN 方法使用 OpenSWAN 进行 VPN 连接，该连接仅支持一个 VPN 隧道。如果您为 VPN 使用了不同的商业软件，则可在站点之间设置两个通道。
2. 设置 VPN 连接后，请添加您的Amazon Web Services 账户中的入站解析程序 IP 地址，以更新 `/etc/resolv.conf` 文件。您将为名称服务器使用此 IP 地址。有关如何获取此 IP 地址的信息，请参阅 配置

Route 53 入站解析程序 (p. 1142)。在这个示例中，我们可以使用您在创建 Route 53 Resolver 时分配的 IP 地址 10.100.0.145。

```
options timeout:2 attempts:5
; generated by /usr/sbin/dhclient-script
search region.compute.internal
nameserver 10.100.0.145
```

3. 现在，我们可以测试 VPN 连接是否使用了 Amazon PrivateLink 端点，而无需使用 nslookup 命令经由公共互联网进行测试。以下是运行命令的示例。

```
nslookup account-specific-prefix.lns.lorawan.region.amazonaws.com
```

下面显示了运行命令的示例输出，该输出显示了一个私有 IP 地址，表示与 Amazon PrivateLink LNS 端点的连接已建立。

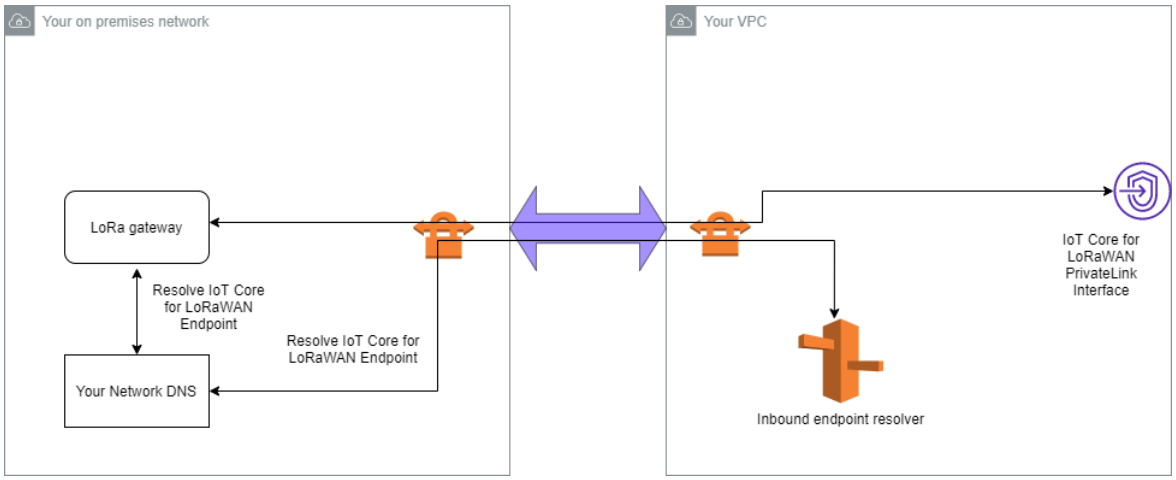
```
Server: 10.100.0.145
Address: 10.100.0.145

Non-authoritative answer:
Name: https://xxxxx.lns.lorawan.region.amazonaws.com
Address: 10.100.0.204
```

有关使用 Site -to-Site VPN 连接的信息，请参阅 Site-to-Site VPN 的工作原理。

Client VPN 终端节点

Amazon Client VPN 是一种基于客户端的托管 VPN 服务，让您能够安全地访问 Amazon 资源和本地网络中的资源。下面显示了 Client VPN 服务的架构。



要建立到 Client VPN 端点的 VPN 连接：

1. 按照 Amazon Client VPN 入门中描述的说明创建 Client VPN 端点。
2. 使用该路由器的访问 URL（例如 192.168.1.1），登录到本地网络（例如，Wi-Fi 路由器），然后查找根名称和密码。
3. 按照网关文档中的说明设置 LoRaWAN 网关，然后将您的网关添加到 Amazon IoT Core for LoRaWAN。有关如何添加网关的信息，请参阅 将您的网关搭载到 Amazon IoT Core for LoRaWAN (p. 1112)。
4. 检查您的网关固件是否为最新版本。如果固件过期，您可以按照本地部署网络中提供的说明更新网关的固件。有关更多信息，请参阅使用 Amazon IoT Core for LoRaWAN 的 CUPS 服务更新网关固件 (p. 1150)。

5. 检查 OpenVPN 是否已启用。如果已启用，请跳到下一步以配置本地部署网络中的 OpenVPN 客户端。如果尚未启用，请遵循为 OpenWrt 安装 OpenVPN 的指南。

   Note

   在此示例中，我们使用 OpenVPN。您可以使用其它 VPN 客户端，例如 Amazon VPN 或者 Amazon Direct Connect 以设置 Client VPN 连接。
6. 根据客户端配置中的信息配置 OpenVPN 客户端，并了解您可以如何使用使用 LuCi 的 OpenVPN 客户端。
7. 将入站解析程序的 IP 地址添加到Amazon Web Services 账户 (`10.100.0.145`)，从而 SSH 连接到您的本地部署网络，并更新`/etc/resolv.conf`文件。
8. 对于要使用Amazon PrivateLink连接到端点的网关流量，请将网关的第一个 DNS 条目替换为入站解析程序的 IP 地址。

有关使用 Site-to-Site VPN 连接的信息，请参阅 Client VPN 入门。

连接到 LNS 和 CUPS VPC 终端节点

下面显示了如何测试与 LNS 和 CUPS VPC 终端节点的连接。

测试 CUPS 端点

要测试您的 LoRa 网关与 CUPS 端点的 Amazon PrivateLink 连接，请运行以下命令：

```
curl -k -v -X POST https://xxxx.cups.region.iotwireless.iot:443/update-info
     --cacert cups.trust --cert cups.crt --key cups.key --header "Content-Type:
 application/json"
     --data '{
              "router": "xxxxxxxxxxxx",
              "cupsUri": "https://xxxx.cups.lorawan.region.amazonaws.com:443",
              "cupsCredCrc":1234, "tcCredCrc":552384314
             }'
      –output cups.out
```

测试 LNS 端点

要测试您的 LNS 端点，请首先预置一个可与您的无线网关配合使用的 LoRaWAN 设备。然后，您可以添加您的设备并执行加入流程，随后您便可以开始发送上行链路消息了。

使用 for LoRa WANAmazon IoT Core 管理网关

网关充当桥梁，通常借助高带宽网络（如 Wi-Fi、Ethernet 或蜂窝移动网络）同 Network Server 传输 W LoRa AN 设备数据。 LoRaWAN 网关将无线设备连接到Amazon IoT Core for LoRaWAN。

以下是将网关与 Amazon IoT Core for LoRaWAN 搭配使用时的一些重要注意事项。有关如何向 Amazon IoT Core for LoRaWAN 添加网关的信息，请参阅 将您的网关搭载到 Amazon IoT Core for LoRaWAN (p. 1112)。

LoRa 基础站软件要求

如要连接到Amazon IoT Core for LoRaWAN，您的 LoRa WAN 网关上必须运行名为 LoRa Basics St ation 的软件。 LoRa 基础站是一个开源软件，由 Semtech Corporation 维护，并通过 Semtech Corporation 的GitHub存储 Amazon IoT Core for LoRaWAN支持 LoRa Basics Station 2.0.4 及更高版本。

使用来自Amazon Partner Device Catalog 的合格网关

Amazon 合作伙伴设备目录包含网关和开发人员工具包，这些网关和开发人员工具包可与 Amazon IoT Core for LoRaWAN 搭配使用。我们建议您使用这些合格的网关，因为您不必修改嵌入软件才能将网关连接到 Amazon IoT Core。这些网关已经具有一个版本的 BasicStation软件，兼容Amazon IoT Core for LoRaWAN。

> Note
>
> 如果您的网关未在 Partner Catalog 中列为可与Amazon IoT Core for LoRaWAN搭配使用的合格网关，则如果网关正在运行 2.0.4 及更高版本的 LoRa Basics Station 软件，那么您仍然能够使用它。确保使用 TLS 服务器和客户端身份验证来验证您的 LoRa WAN 网关。

使用 CUPS 和 LNS 协议

LoRa 基础站软件包含两个子协议，用于将网关连接到网络服务器，即 LoRa WAN Network Server (LNS) 和 Configuration and Update Server (CUPS) 协议。

LNS 协议在 B LoRa asics Station 兼容网关和网络服务器之间建立了数据连接。 LoRa 上行链路和下行链路消息将借助此数据连接通过安全的方式进行交换 WebSockets。

CUPS 协议支持凭证管理以及网关远程配置和固件更新。 Amazon IoT Core for LoRaWAN提供 LNS 和 CUPS 端点，分别用于 LoRa WAN 数据摄取和远程网关管理。

有关更多信息，请参阅 LNS 协议和 CUPS 协议。

配置 LoRa WAN 网关的信标和筛选功能

在使用 LoRa WAN 设备时，您可以为 W LoRa AN 网关配置某些可选参数。这些参数包括：

- 信标

  您可以为充当 B 类 LoRa WAN 设备的网桥的 W LoRa AN 网关配置信标参数。这些设备在预定时隙接收下行链路消息，因此，您必须为网关配置信标参数，才能传输这些时间同步信标。
- 过滤

  您可以为 LoRa WAN 网关配置`NetID`和`JoinEUI`参数来筛选设备数据流量。筛选流量有助于节省带宽用量并减少网关与 LNS 之间的流量。
- 子频段

  您可以为网关配置子频段，以指定要使用的特定子频段。对于无法在不同子频带之间跳转的无线设备，您可以使用此功能与仅使用该特定子频带中的频率通道的设备通信。

以下主题包含有关这些参数及其配置方法的更多信息。信标参数在 Amazon Web Services Management Console中不可用，只能使用 Amazon IoT Wireless API 或 Amazon CLI 指定。

主题

- 将网关配置为向 B 类设备发送信标 (p. 1146)
- 配置网关的子频带和筛选功能 (p. 1148)

将网关配置为向 B 类设备发送信标

如果您将 B 类无线设备搭载到 Amazon IoT Core for LoRaWAN，这些设备会在预定时隙内接收下行链路消息。这些设备根据网关传输的时间同步信标打开这些时隙。为了让网关传输这些时间同步信标，可以使用 Amazon IoT Core for LoRaWAN 为网关配置某些信标相关参数。

要配置这些信标参数，您的网关必须运行 B LoRa asics Station 软件版本 2.0.6。请参阅使用来自Amazon Partner Device Catalog 的合格网关 (p. 1146)。

如何配置信标参数

Note

只有在网关与 B 类无线设备通信时，才需要为网关配置信标参数。

使用 CreateWirelessGateway API 操作将网关添加到 Amazon IoT Core for LoRaWAN 时，可以配置信标参数。调用 API 操作时，通过为网关使用 Beaconing 对象指定以下参数。配置参数后，网关将以 128 秒的间隔向您的设备发送信标。

- DataRate：传输信标的网关的数据速率。
- Frequencies：网关传输信标的频率列表。

下面的示例显示如何为网关配置这些参数。input.json 文件将包含其他详细信息，如网关证书和预调配凭证。有关使用 CreateWirelessGateway API 操作将网关添加到 Amazon IoT Core for LoRaWAN 的更多信息，请参阅使用 API 添加网关 (p. 1115)。

Note

当您使用 Amazon IoT 控制台将网关添加到 Amazon IoT Core for LoRaWAN 时，信标参数不可用。

```
aws iotwireless create-wireless-gateway \
    --name "myLoRaWANGateway" \
    --cli-input-json file://input.json
```

下面显示的是 input.json 文件的内容。

input.json 的内容

```
{
    "Description": "My LoRaWAN gateway",
    "LoRaWAN": {
        "Beaconing": {
          "DataRate": 8,
          "Frequencies": ["923300000","923900000"]
        },
        "GatewayEui": "a1b2c3d4567890ab",
        "RfRegion": US915,
        "JoinEuiFilters": [
         ["0000000000000001", "00000000000000ff"],
         ["000000000000ff00", "000000000000ffff"]
         ],
        "NetIdFilters": ["000000", "000001"],
        "RfRegion": "US915",
        "SubBands": [2]
    }
}
```

下面的代码显示运行此命令的输出示例。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:400232685877aa:WirelessGateway/a01b2c34-
d44e-567f-abcd-0123e445663a",
    "Id": a01b2c34-d44e-567f-abcd-0123e445663a"
```

```
}
```

获取有关信标参数的信息

您可以使用 GetWirelessGateway API 操作获取有关网关信标参数的信息。

> Note
>
> 如果已经搭载网关，则不能使用 UpdateWirelessGateway API 操作配置信标参数。要配置这些参数，您必须删除网关，然后在使用 CreateWirelessGateway API 操作添加网关时指定这些参数。

```
aws iotwireless get-wireless-gateway \
    --identifier "12345678-a1b2-3c45-67d8-e90fa1b2c34d" \
    --identifier-type WirelessGatewayId
```

运行此命令会返回有关您的网关和信标参数的信息。

配置网关的子频带和筛选功能

LoRaWAN 网关运行 LoRa Basics St ation 软件，使网关能够连接到Amazon IoT Core for LoRaWAN。为连接到Amazon IoT Core for LoRaWAN，您的 LoRa 网关首先查询 CUPS 服务器的 LNS 端点，然后建立与该端点 WebSockets 的数据连接。建立连接后，上行链路和下行链路帧便可以通过该连接进行交换。

筛选网关接收 LoRa 的数据帧

在您的 LoRa WAN 网关与终端节点建立连接后，使用一条router_config消息进行Amazon IoT Core for LoRaWAN响应，该消息为 LoRa 网关的配置指定了一组参数，包括过滤参数NetID和JoinEui。有关router_config以及如何与 LoRa WAN Network Server (LNS) 建立连接，请参阅 LNS 协议。

```
{
"msgtype"    : "router_config"
"NetID"      : [ INT, .. ]
"JoinEui"    : [ [INT,INT], .. ] // ranges: beg,end inclusive
"region"     : STRING             // e.g. "EU863", "US902", ..
"hwspec"     : STRING
"freq_range" : [ INT, INT ]       // min, max (hz)
"DRs"        : [ [INT,INT,INT], .. ]   // sf,bw,dnonly
"sx1301_conf": [ SX1301CONF, .. ]
"nocca"      : BOOL
"nodc"       : BOOL
"nodwell"    : BOOL
}
```

这些网关通常借助高带宽网络（如 Wi-Fi、Ethernet 或蜂窝移动网络）同 LNS 传输 W LoRa AN 设备数据。网关通常会接收所有消息，然后将收到的流量传送到 Amazon IoT Core for LoRaWAN。但是，您可以配置网关以筛选某些设备数据流量，这有助于节省带宽用量并减少网关与 LNS 之间的流量流。

要将 LoRa 网关配置为筛选数据帧，可以使用router_config消息JoinEui中的参数NetID和。 NetID是可接受的 netId 值的列表。任何 LoRa 带有除所列内容以外数据帧的数据帧都将被删除。 JoinEui是编码 JoinEui 值范围的整数值对列表。联接请求帧将被网关删除，除非消息JoinEui中的字段处于 [BegEui,EndEui] 范围内。

频率通道和子频段

对于 US915 和 AU915 射频区域，无线设备可选择 64 个 125 KHz 和 8 个 500kHz 上行链路通道，以便使用 LoRa网关访问 LoRa WAN 网络。上行链路频率通道分为 8 个子频段，每个子频段有 8 个 125KHz 通道和一个 500KqHz 通道。对于 AU915 区域中的每个常规网关，将支持一个或多个子频段。

某些无线设备无法在子频段之间跳远，且在连接到 Amazon IoT Core for LoRaWAN 时仅在一个子频段中使用频率通道。对于要传输且来自这些设备的上行链路数据包，请配置 LoRa 网关以使用该特定子频段。对于其它射频区域（如 EU868）的网关，不需要此配置。

使用控制台配置您的网关以使用筛选条件和子频段

您可以配置网关以使用特定的子频段，还可以启用筛选 LoRa 数据帧的功能。要使用控制台指定这些参数：

1. 导航到 Amazon IoT 控制台的 Amazon IoT Core for LoRaWAN Gateways（网关）页面并选择 Add gateway（添加网关）。
2. 指定网关详细信息，例如 Gateway's Eui（网关的 Eui）、Frequency band (RFRegion)（频段（射频区域））和可选内容，如 Name（名称）和 Description（说明），然后选择是否将 Amazon IoT 事物与您的网关关联。有关如何添加网关的更多信息，请参阅 使用控制台添加网关 (p. 1114)。
3. 在 LoRaWAN configuration（广域网配置）部分中，您可以指定子频段和筛选信息。
   - SubBands：要添加子频段，请选择 Add（添加） SubBand并指定一个整数值列表，这些值将说明网关支持哪些子频段。SubBands 参数只能在 RfRegion US915 和 AU915 上进行配置，并且在其中支持的区域之一中拥有处于范围 [1,8] 内的值。
   - NetIdFilters：要筛选上行链路帧，请选择 Add（添加） NetId并指定网关使用的字符串值列表。来自无线设备的传入上行链路帧的 NetID 必须与至少一个列出的值匹配，否则该帧将被丢弃。
   - JoinEuiFilters：选择 Add ange（添加 JoinEui范围），并指定网关用于筛选 LoRa 帧的字符串值对列表。作为来自无线设备的联接请求的一部分指定的 JoinEui JoinEuiRange 值必须至少在一部分以 [BegEui, EndEui] 对的形式列示，否则该帧将被删除。
4. 然后，您可以按照 使用控制台添加网关 (p. 1114) 中所述的指示继续配置网关。

添加网关后，如果您选择已添加的Amazon IoT Core for LoRaWANAmazon IoT网关，则可以在网关详细信息页面的 LoRaWAN 特定详细信息部分查看SubBandsNetIdFilters和JoinEuiFilters筛选条件。

使用 API 配置您的网关以使用筛选条件和子频段。

您可以使用您用来创建网关CreateWirelessGateway的 API，来配置要使用的子频段并启用筛选功能。借助 CreateWirelessGateway API，您可以指定子频段和筛选条件，将其作为您使用 LoRaWAN 字段所提供的网关配置信息的一部分。下面显示包含此信息的请求令牌。

```
POST /wireless-gateways HTTP/1.1
Content-type: application/json

{
    "Arn": "arn:aws:iotwireless:us-east-1:400232685877aa:WirelessGateway/a11e3d21-
e44c-471c-afca-6716c228336a",
    "Description": "Using my first LoRaWAN gateway",
    "LoRaWAN": {
        "GatewayEui": "a1b2c3d4567890ab",
        "JoinEuiFilters": [
        ["0000000000000001", "00000000000000ff"],
        ["0000000000000ff00", "0000000000000ffff"]
      ],
      "NetIdFilters": ["000000", "000001"],
      "RfRegion": "US915",
      "SubBands": [2]
   },
   "Name": "myFirstLoRaWANGateway"
   "ThingArn": null,
   "ThingName": null
}
```

您也可以使用 UpdateWirelessGatewayAPI 来更新筛选条件，但不更新子频段。如果 JoinEuiFilters 和 NetIdfilters 值为空，这意味着字段没有更新。如果值不为空且包含空列表，则应用更新。若要获取您指定的字段的值，请使用 GetWirelessGatewayAPI。

使用 Amazon IoT Core for LoRaWAN 的 CUPS 服务更新网关固件

您的网关上运行的 LoRa Basics St ation 软件使用 Configuration and Update Server (CUPS) 协议提供凭证管理和固件更新接口。CUPS 协议借助 ECDSA 签名提供安全的固件更新交付。

您必须经常更新网关的固件。您可以使用 Amazon IoT Core for LoRaWAN 的 CUPS 服务来向能够签署更新的网关提供固件更新。要更新网关固件，您可以使用 SDK 或 CLI，但不要使用控制台。

该升级过程可能最多需要 45 分钟才能完成。如果您是首次设置网关来连接到 Amazon IoT Core for LoRaWAN，则可能需要更长的时间。网关制造商通常会提供自己的固件更新文件和签名，因此您可以使用它们并前进至 将固件文件上载到 S3 存储桶并添加 IAM 角色 (p. 1153)。

如果您没有固件更新文件，请参阅 生成固件更新文件和签名 (p. 1150) 作为示例，您可以使用该示例来修改您的应用程序。

要执行网关固件更新：

- 生成固件更新文件和签名 (p. 1150)
- 将固件文件上载到 S3 存储桶并添加 IAM 角色 (p. 1153)
- 使用任务定义计划并运行固件更新 (p. 1156)

生成固件更新文件和签名

本流程的步骤是可选的，取决于您使用的网关。网关制造商会以更新文件或脚本的形式提供自己的固件更新，Basics Station 会在后台运行此脚本。在这种情况下，您很可能会在所使用网关的发布说明中找到固件更新文件。然后，您可以改为使用该更新文件或脚本，然后前进至 将固件文件上载到 S3 存储桶并添加 IAM 角色 (p. 1153)。

如果您没有此脚本，下面显示了用于生成固件更新文件的命令。您还可以签署更新，以确保代码未被更改或损坏，并且设备仅运行由受信任的作者发布的代码。

在此流程中，您将：

- 生成固件更新文件 (p. 1150)
- 为固件更新生成签名 (p. 1152)
- 查看后续步骤 (p. 1153)

生成固件更新文件

在网关上运行的 LoRa Basics Station 软件能够接收 CUPS 响应中的固件更新。如果您没有制造商提供的脚本，请参阅以下固件更新脚本，该脚本是为基于 Raspberry Pi 的 RAKWireless Gateway 编写的。我们有一个基本脚本，而新工作站二进制文件、版本文件和 `station.conf` 将被附加其上。

> **Note**
>
> 该脚本特定于 RAKWireless Gateway ，因此您必须根据您使用的网关对其进行调整，以使其适用于您的应用程序。

基本脚本

下面显示了基于 Raspberry Pi 的 RAKWireless Gateway 的基本脚本示例。您可以将以下命令保存在文件 `base.sh` 中，然后在 Raspberry Pi 的 Web 浏览器终端中运行脚本。

```
*#!/bin/bash*
execution_folder=/home/pi/Documents/basicstation/examples/aws_lorawan
```

```
station_path="$execution_folder/station"
version_path="$execution_folder/version.txt"
station_conf_path="$execution_folder/station_conf"

# Function to find the Basics Station binary at the end of this script
# and store it in the station path
function prepare_station()
{
 match=$(grep --text --line-number '^STATION:$' $0 | cut -d ':' -f 1)
 payload_start=$((match + 1))
 match_end=$(grep --text --line-number '^END_STATION:$' $0 | cut -d ':' -f 1)
 payload_end=$((match_end - 1))
 lines=$(($payload_end-$payload_start+1))
 head -n $payload_end $0 | tail -n $lines  > $station_path
}

# Function to find the version.txt at the end of this script
# and store it in the location for version.txt
function prepare_version()
{
  match=$(grep --text --line-number '^VERSION:$' $0 | cut -d ':' -f 1)
  payload_start=$((match + 1))
  match_end=$(grep --text --line-number '^END_VERSION:$' $0 | cut -d ':' -f 1)
  payload_end=$((match_end - 1))
  lines=$(($payload_end-$payload_start+1))
  head -n $payload_end $0 | tail -n $lines  > $version_path
}

# Function to find the version.txt at the end of this script
# and store it in the location for version.txt
function prepare_station_conf()
{
 match=$(grep --text --line-number '^CONF:$' $0 | cut -d ':' -f 1)
 payload_start=$((match + 1))
 match_end=$(grep --text --line-number '^END_CONF:$' $0 | cut -d ':' -f 1)
 payload_end=$((match_end - 1))
 lines=$(($payload_end-$payload_start+1))
 head -n $payload_end $0 | tail -n $lines  > $station_conf_path
}

# Stop the currently running Basics station so that it can be overwritten
# by the new one
killall station

# Store the different files
prepare_station
prepare_versionp
prepare_station_conf

# Provide execute permission for Basics station binary
chmod +x $station_path

# Remove update.bin so that it is not read again next time Basics station starts
rm -f /tmp/update.bin

# Exit so that rest of this script which has binaries attached does not get executed
exit 0
```

添加负载脚本

在基本脚本中，我们会附加 Basics Station 二进制文件、标识要更新版本的 version.txt 和名为 `addpayload.sh` 脚本中的 `station.conf`。然后，运行此脚本。

```
*#!/bin/bash
```

```
*
base.sh > fwstation

# Add station
echo "STATION:" >> fwstation
cat $1 >> fwstation
echo "" >> fwstation
echo "END_STATION:" >> fwstation

# Add version.txt
echo "VERSION:" >> fwstation
cat $2 >> fwstation
echo "" >> fwstation
echo "END_VERSION:" >> fwstation

# Add station.conf
echo "CONF:" >> fwstation
cat $3 >> fwstation
echo "END_CONF:" >> fwstation

# executable
chmod +x fwstation
```

运行这些脚本后，您可以在终端中运行以下命令来生成固件更新文件，`fwstation`。

```
$ ./addpayload.sh station version.txt station.conf
```

为固件更新生成签名

LoRa 基础站软件提供带有 ECDSA 签名的已签名固件更新。要支持已签名的更新，您需要：

- 必须由 ECDSA 私有秘钥生成且少于 128 字节的签名。
- 用于签名的私有密钥且必须存储在网关中，其文件名格式应为 `sig-%d.key`。建议使用 `sig-0.key` 作为文件名。
- 私有密钥上的 32 位 CRC。

签名和 CRC 将传递给 Amazon IoT Core for LoRaWAN API。要生成以前的文件，可以使用以下脚本`gen.sh`，该脚本是根据 b asicstaion 示例编写 GitHub 的。

```
*#!/bin/bash

*function ecdsaKey() {
    # Key not password protected for simplicity
    openssl ecparam -name prime256v1 -genkey | openssl ec -out $1
}

# Generate ECDSA key
ecdsaKey sig-0.prime256v1.pem

# Generate public key
openssl ec -in sig-0.prime256v1.pem -pubout -out sig-0.prime256v1.pub

# Generate signature private key
openssl ec -in sig-0.prime256v1.pub -inform PEM -outform DER -pubin | tail -c 64 >
 sig-0.key

# Generate signature
openssl dgst -sha512 -sign sig-0.prime256v1.pem $1 > sig-0.signature

# Convert signature to base64
```

```
openssl enc -base64 -in sig-0.signature -out sig-0.signature.base64

# Print the crc
crc_res=$(crc32 sig-0.key)printf "The crc for the private key=%d\n" $((16#$crc_res))

# Remove the generated files which won't be needed later
rm -rf sig-0.prime256v1.pem sig-0.signature sig-0.prime256v1.pub
```

脚本生成的私有秘钥应保存到网关中。密钥文件采用二进制格式。

```
./gen_sig.sh fwstation
read EC key
writing EC key
read EC key
writing EC key
read EC key
writing EC key
The crc for the private key=3434210794

$ cat sig-0.signature.base64
MEQCIDPY/p2ssgXIPNCOgZr+NzeTLpX+WfBo5tYWbh5pQWN3AiBROen+XlIdMScv
AsfVfU/ZScJCalkVNZh4esyS8mNIgA==

$ ls sig-0.key
sig-0.key

$ scp sig-0.key pi@192.168.1.11:/home/pi/Documents/basicstation/examples/iotwireless
```

查看后续步骤

现在您已经生成了固件和签名，请转到下一个主题，将固件文件 `fwstation` 上载到 Amazon S3 存储桶。存储桶是将固件更新文件作为对象存储的容器。您可以添加 IAM 角色，该角色将授予 CUPS 服务器读取 S3 存储桶中的固件更新文件的权限。

将固件文件上载到 S3 存储桶并添加 IAM 角色

您可以使用 Amazon S3 创建存储桶，这是一个可以存储固件更新文件的容器。您可以将文件上载到 S3 存储桶，并添加允许 CUPS 服务器从存储桶读取更新文件的 IAM 角色。有关 Amazon S3 的更多信息，请参阅 Amazon S3 入门。

要上载的固件更新文件取决于您使用的网关。如果您遵循类似于 生成固件更新文件和签名 (p. 1150) 中所述的流程，则您将上载通过运行脚本所生成的 `fwstation` 文件。

完成本流程需要大约 20 分钟。

要上载您的固件文件：

- 创建 Amazon S3 存储桶并上载更新文件 (p. 1153)
- 创建具有读取 S3 存储桶权限的 IAM 角色 (p. 1154)
- 查看后续步骤 (p. 1156)

创建 Amazon S3 存储桶并上载更新文件

您将使用 Amazon Web Services Management Console 创建 Amazon S3 存储桶，然后将固件更新文件上载到存储桶中。

创建 S3 存储桶

要创建 S3 存储桶，请打开 Amazon S3 控制台。登录（如果尚未登录），然后执行以下步骤：

1. 选择创建桶。
2. 输入唯一且有意义的名称作为 Bucket name（存储桶名称），例如 `iotwirelessfwupdate`。有关存储桶的建议命名约定，请参阅 https://docs.amazonaws.cn/AmazonS3/latest/userguide/bucketnamingrules.html。
3. 确保您选择了来创建 LoRa WAN 网关和设备，并且Amazon Web Services 区域选中 Block all public access（阻止所有公有访问）设置，以便您的存储桶使用默认权限。
4. 为 Bucket versioning（存储桶版本控制）选择 Enable（启用），这将帮助您将固件更新文件的多个版本保留在同一存储桶中。
5. 确认将 Server-side encryption（服务器端加密）设置为 Disable（禁用），然后选择 Create bucket（创建存储桶）。

上载固件更新文件

现在，您可以在显示在 Amazon Web Services Management Console 的存储桶列表中看到您的存储桶了。选择您的存储桶并完成以下步骤以上载您的文件。

1. 选择您的存储桶，然后选择 Upload（上载）。
2. 选择 Add file（添加文件），然后上载固件更新文件。如果您按照 生成固件更新文件和签名 (p. 1150) 中所述的流程执行，您应上载 `fwstation` 文件，否则请上载网关制造商提供的文件。
3. 确保所有设置都保留为默认值。请确保 Predefined ACL（预定义 ACL）设置为 private（私有），然后选择 Upload（上载）以上载您的文件。
4. 复制您所上载文件的 S3 URI。选择您的存储桶，您将看到上载的文件显示在 Objects（对象）列表中。选择您的文件，然后选择 Copy S3 URI（复制 S3 URI）。如果您将存储桶命名为类似于前述示例 (`fwstation`)，则 URI 将类似于：`s3://iotwirelessfwupdate/fwstation`。在创建 IAM 角色时，您将使用 Amazon S3 URI。

创建具有读取 S3 存储桶权限的 IAM 角色

现在，您将创建 IAM 角色和策略，该角色和策略将授予 CUPS 从 S3 存储桶读取固件更新文件的权限。

为您的角色创建 IAM 策略

要为 Amazon IoT Core for LoRaWAN 目标角色创建 IAM 策略，请打开 IAM 控制台的策略中心，然后完成以下步骤：

1. 选择 Create policy（创建策略），然后选择 JSON 选项卡。
2. 从编辑器中删除所有内容并粘贴此策略文档。策略提供访问 `iotwireless` 存储桶的权限，固件更新文件 `fwstation` 存储在对象内。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::iotwirelessfwupdate/fwstation",
                "arn:aws:s3:::iotwirelessfwupdate"
            ]
        }
```

```
    ]
}
```

3. 选择 Review policy（查看策略），在 Name（名称）中，输入此策略的名称（例如，`IoTWirelessFwUpdatePolicy`）。在下一流程中，您将需要使用此名称。
4. 选择 Create policy（创建策略）。

使用附加的策略创建 IAM 角色。

现在，您将创建一个 IAM 角色并附加以前为访问 S3 存储桶而创建的策略。打开 IAM 控制台的角色中心并完成以下步骤：

1. 选择 Create role（创建角色）。
2. 在 Select type of trusted entity（选择受信任实体的类型）下，选择 Another Amazon Web Services 账户（另一个亚马逊云科技账户）。
3. 在 Account ID（账户 ID），请输入您的Amazon Web Services 账户 ID，然后选择 Next: Permission（下一步：权限）。
4. 在搜索框中，输入您在上一流程中创建的 IAM 策略名称。在搜索结果中检查您之前创建的 IAM 策略（例如 `IoTWirelessFwUpdatePolicy`），然后选择该策略。
5. 依次选择 Next: Tags（下一步：标签）和 Next: Review（下一步：查看）。
6. 对于 Role name（角色名称），输入此角色的名称（例如 `IoTWirelessFwUpdateRole`），然后选择 Create role（创建角色）。

编辑 IAM 角色的信任关系

在运行上一步后显示的确认信息中，选择您创建的角色的名称进行编辑。您将编辑角色以添加以下信任关系。

1. 在刚刚创建的角色的 Summary（摘要）页面上，选择 Trust relationship（信任关系）选项卡，然后选择 Edit trust relationship（编辑信任关系）。
2. 在 Policy Document（策略文档）中，更改 `Principal` 属性以使其类似于此示例。

```
"Principal": {
    "Service": "iotwireless.amazonaws.com"
},
```

在您更改 `Principal` 属性后，完整的策略文档应该如此示例所示。

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

3. 要保存更改并退出，请选择 Update Trust Policy（更新信任策略）。
4. 获取角色的 ARN。选择在“摘要”部分您的 IAM 角色，您将看到 Role ARN（角色 ARN），例如 `arn:aws:iam::123456789012:role/IoTWirelessFwUpdateRole`。复制此 Role ARN（角色 ARN）。

查看后续步骤

现在，您已创建 S3 存储桶和允许 CUPS 服务器读取 S3 存储桶的 IAM 角色，请转到下一个主题以安排并运行固件更新。保留您之前复制的 S3 URI 和 Role ARN（角色 ARN），以便您可以输入这些信息以创建将运行的任务定义，以执行固件更新。

使用任务定义计划并运行固件更新

您可以使用任务定义包含有关固件更新的详细信息并定义更新。Amazon IoT Core for LoRaWAN 将根据与网关关联的以下三个字段中的信息提供固件更新。

- Station

  Basics Station 软件的版本和构建时间。要识别此信息，您还可以使用网关运行的 Basics Station 软件生成该信息（例如，`2.0.5(rpi/std) 2021-03-09 03:45:09`）。
- PackageVersion

  固件版本，由网关中的文件 `version.txt` 指定。虽然网关中可能不存在此信息，但我们建议您使用此方法定义固件版本（例如，`1.0.0`）。
- 模型

  网关正在使用的平台或模型（例如，Linux）。

此流程需要 20 分钟时间完成。

要完成此流程：

- 获取网关上运行的当前版本 (p. 1156)
- 创建无限网关任务定义 (p. 1157)
- 运行固件更新任务并跟踪进度 (p. 1158)

获取网关上运行的当前版本

要确定网关是否有资格进行固件更新，CUPS 服务器将检查所有三个字段：`Station`、`PackageVersion` 和 `Model`，网关在 CUPS 请求期间提供这些信息时，服务器将进行匹配。使用任务定义时，这些字段将作为 `CurrentVersion` 字段的一部分予以存储。

您可以使用 Amazon IoT Core for LoRaWAN API 或 Amazon CLI 为您的网关获取 `CurrentVersion`。以下命令演示如何使用 CLI 获取此信息。

1. 如果您已经预置了网关，则可以使用get-wireless-gateway命令获取有关网关的信息。

```
aws iotwireless get-wireless-gateway \
    --identifier 5a11b0a85a11b0a8 \
        --identifier-type GatewayEui
```

下面显示了此命令的示例输出。

```
{
    "Name": "Raspberry pi",
    "Id": "1352172b-0602-4b40-896f-54da9ed16b57",
    "Description": "Raspberry pi",
    "LoRaWAN": {
        "GatewayEui": "5a11b0a85a11b0a8",
        "RfRegion": "US915"
    },
```

```
    "Arn": "arn:aws:iotwireless:us-
east-1:231894231068:WirelessGateway/1352172b-0602-4b40-896f-54da9ed16b57"
}
```

2. 使用get-wireless-gateway命令报告的无线网关 ID，您可以使用 get-wireless-gateway-firmware-information 命令获取CurrentVersion。

```
aws iotwireless get-wireless-gateway-firmware-information \
    --id "3039b406-5cc9-4307-925b-9948c63da25b"
```

下面显示了该命令的示例输出，其中包含 CurrentVersion 显示的所有三个字段的信息。

```
{
    "LoRaWAN": {
        "CurrentVersion": {
            "PackageVersion": "1.0.0",
            "Model": "rpi",
            "Station": "2.0.5(rpi/std) 2021-03-09 03:45:09"
        }
    }
}
```

创建无限网关任务定义

在创建任务定义时，我们建议您使用 AutoCreateTasks 参数指定自动创建任务。AutoCreateTasks 适用于与前面提到的所有三个参数匹配的任何网关。如果禁用此参数，则必须手动将参数分配给网关。

您可以使用 Amazon IoT Core for LoRaWAN API 或 Amazon CLI 创建无限网关任务定义。以下命令演示如何使用 CLI 创建任务定义。

1. 创建一个文件，input.json，它将包含要传递给 CreateWirelessGatewayTaskDefinition API 的信息。在 input.json 文件中，提供之前获取的以下信息：
   - UpdateDataSource

     提供指向对象的链接，该对象中包含您上载到 S3 存储桶的固件更新文件（例如，s3://iotwirelessfwupdate/fwstation）。
   - UpdateDataRole

     提供指向您创建的 IAM 角色的角色 ARN 的链接，该角色提供读取 S3 存储桶的权限（例如，arn:aws:iam::123456789012:role/IoTWirelessFwUpdateRole）。
   - SigKeyCRC 和 UpdateSignature

     此信息可能由您的网关制造商提供，但如果您遵循 生成固件更新文件和签名 (p. 1150) 中所述的流程，则您将在生成签名时找到此信息。
   - CurrentVersion

     提供您之前通过运行 get-wireless-gateway-firmware-information 命令获得的 CurrentVersion 输出。

   ```
   cat input.json
   ```

   下面显示的是 input.json 文件的内容。

   ```
   {
       "AutoCreateTasks": true,
       "Name": "FirmwareUpdate",
   ```

```
    "Update":
    {
        "UpdateDataSource" : "s3://iotwirelessfwupdate/fwstation",
        "UpdateDataRole" : "arn:aws:iam::123456789012:role/IoTWirelessFwUpdateRole",
        "LoRaWAN" :
        {
            "SigKeyCrc": 3434210794,
            "UpdateSignature": "MEQCIDPY/p2ssgXIPNCOgZr+NzeTLpX+WfBo5tYWbh5pQWN3AiBROen
+XlIdMScvAsfVfU/ZScJCalkVNZh4esyS8mNIgA==",
            "CurrentVersion" :
            {
            "PackageVersion": "1.0.0",
            "Model": "rpi",
            "Station": "2.0.5(rpi/std) 2021-03-09 03:45:09"
            }
        }
    }
}
```

2. 将input.json文件传递给 create-wireless-gateway-task-defin ition 命令以创建任务定义。

```
aws iotwireless create-wireless-gateway-task-definition \
    --cli-input-json file://input.json
```

以下显示了命令的输出。

```
{
    "Id": "4ac46ff4-efc5-44fd-9def-e8517077bb12",
    "Arn": "arn:aws:iotwireless:us-
east-1:231894231068:WirelessGatewayTaskDefinition/4ac46ff4-efc5-44fd-9def-e8517077bb12"
}
```

运行固件更新任务并跟踪进度

网关已准备好接收固件更新，一旦打开电源，就会连接到 CUPS 服务器。当 CUPS 服务器在网关版本中找到匹配项时，它会计划固件更新。

任务是正在进行的任务定义。当您通过将 AutoCreateTasks 设置为 True 指定自动任务创建时，一旦找到匹配的网关，固件更新任务就会立即启动。

您可以使用 GetWirelessGatewayTask API 跟踪任务的进度。当您首次运行该get-wireless-gateway-task命令时，它会将任务状态显示为IN_PROGRESS。

```
aws iotwireless get-wireless-gateway-task \
    --id 1352172b-0602-4b40-896f-54da9ed16b57
```

以下显示了命令的输出。

```
{
    "WirelessGatewayId": "1352172b-0602-4b40-896f-54da9ed16b57",
    "WirelessGatewayTaskDefinitionId": "ec11f9e7-b037-4fcc-aa60-a43b839f5de3",
    "LastUplinkReceivedAt": "2021-03-12T09:56:12.047Z",
    "TaskCreatedAt": "2021-03-12T09:56:12.047Z",
    "Status": "IN_PROGRESS"
}
```

下次运行命令时，如果固件更新生效，它将显示更新后的字段，Package、Version 和 Model，而任务状态将变为 COMPLETED。

```
aws iotwireless get-wireless-gateway-task \
    --id 1352172b-0602-4b40-896f-54da9ed16b57
```

以下内容显示命令的输出。

```
{
    "WirelessGatewayId": "1352172b-0602-4b40-896f-54da9ed16b57",
    "WirelessGatewayTaskDefinitionId": "ec11f9e7-b037-4fcc-aa60-a43b839f5de3",
    "LastUplinkReceivedAt": "2021-03-12T09:56:12.047Z",
    "TaskCreatedAt": "2021-03-12T09:56:12.047Z",
    "Status": "COMPLETED"
}
```

在此示例中，我们向您展示了使用基于 Raspberry Pi 的 RAKWireless 网关的固件更新。固件更新脚本 BasicStation 会停止运行 BasicStation 以存储更Package新的和Model字段，因此必须重新启动。Version

```
2021-03-12 09:56:13.108 [CUP:INFO] CUPS provided update.bin
2021-03-12 09:56:13.108 [CUP:INFO] CUPS provided signature len=70 keycrc=37316C36
2021-03-12 09:56:13.148 [CUP:INFO] ECDSA key#0 -> VERIFIED
2021-03-12 09:56:13.148 [CUP:INFO] Running update.bin as background process
2021-03-12 09:56:13.149 [SYS:VERB] /tmp/update.bin: Forked, waiting...
2021-03-12 09:56:13.151 [SYS:INFO] Process /tmp/update.bin (pid=6873) completed
2021-03-12 09:56:13.152 [CUP:INFO] Interaction with CUPS done - next regular check in 10s
```

如果固件更新失败，您将从 CUPS server 看到 FIRST_RETRY 的状态，而网关会发送相同的请求。如果 SECOND_RETRY 后 CUPS 服务器无法连接到网关，它将显示 FAILED 的状态。

在上一任务为COMPLETED或之后FAILED，使用delete-wireless-gateway-task命令删除旧任务，然后再启动一个新任务。

```
aws iotwireless delete-wireless-gateway-task \
    --id 1352172b-0602-4b40-896f-54da9ed16b57
```

选择网关以接收 LoRa WAN 下行链路数据流量

当您从 Amazon IoT Core for LoRaWAN 向设备发送下行链路消息时，您可以选择要用于下行链路数据流量的网关。您可以指定单个网关或从网关列表中进行选择，以接收下行链路流量。

如何指定网关列表

当使用 SendDataToWirelessDevice API 操作从 Amazon IoT Core for LoRaWAN 向设备发送下行链路消息时，您可以指定要使用的单个网关或网关列表。调用 API 操作时，通过为网关使用 ParticipatingGateways 对象指定以下参数。

> **Note**
>
> 您要使用的网关列表在 Amazon IoT 控制台中不可用。仅当使用 SendDataToWirelessDevice API 操作或 CLI 时，才可以指定要使用的这一网关列表。

- DownlinkMode：指示是以顺序模式还是并发模式发送下行链路消息。对于 A 类设备，请指定 UsingUplinkGateway，以仅使用先前上行链路消息传输中选定的网关。
- GatewayList：您要用于发送下行链路数据流量的网关列表。下行链路负载将以指定的频率发送到指定的网关。这是使用由 GatewayId 和 DownlinkFrequency 对组成的 GatewayListItem 对象的列表来表示的。
- TransmissionInterval：Amazon IoT Core for LoRaWAN 在将负载传输到下一个网关之前将等待的时间。

Note

您可以将此网关列表指定为仅在向 B 类或 C 类无线设备发送下行链路消息时使用。如果您使用 A 类设备，则在向设备发送下行链路消息时，将使用您在发送上行链路消息时选择的网关。

下面的示例显示如何为网关指定这些参数。input.json 文件将包含其他详细信息。有关使用 SendDataToWirelessDevice API 操作发送下行链路消息的更多信息，请参阅使用 API 执行下行链路队列操作 (p. 1166)。

Note

使用 Amazon IoT 控制台从 Amazon IoT Core for LoRaWAN 发送下行链路消息时，用于指定参与网关列表的参数不可用。

```
aws iotwireless send-data-to-wireless-device \
    --id "11aa5eae-2f56-4b8e-a023-b28d98494e49" \
    --transmit-mode "1" \
    --payload-data "SGVsbG8gVG8gRGV2c2lt" \
    --cli-input-json file://input.json
```

下面显示的是 input.json 文件的内容。

input.json 的内容

```
{
    "WirelessMetadata": {
        "LoRaWAN": {
            "FPort": "1",
            "ParticipatingGateways": {
                "DownlinkMode": "SEQUENTIAL",
                "TransmissionInterval": 1200,
                "GatewayList": [
                    {
                        "DownlinkFrequency": 100000000,
                        "GatewayID": a01b2c34-d44e-567f-abcd-0123e445663a
                    },
                    {
                        "DownlinkFrequency": 100000101,
                        "GatewayID": 12345678-a1b2-3c45-67d8-e90fa1b2c34d
                    }
                ]
            }
        }
    }
}
```

运行此命令的输出会生成下行链路消息的 MessageId。在某些情况下，即使收到 MessageId，数据包也可能丢失。有关如何解决错误的更多信息，请参阅 排查下行链路消息队列错误 (p. 1166)。

```
{
    MessageId: "6011dd36-0043d6eb-0072-0008"
}
```

获取有关参与网关的列表的信息

您可以通过列出下行链路队列中的消息，获取有关参与接收下行链路消息的网关列表的信息。要列出消息，请使用 ListQueuedMessages API。

```
aws iotwireless list-queued-messages \
```

```
    --wireless-device-type "LoRaWAN"
```

运行此命令会返回有关队列中的消息及其参数的信息。

使用 Amazon IoT Core for LoRaWAN 管理设备

LoRaWAN 设备Amazon IoT Core for LoRaWAN通过 W LoRa AN 网关与之通信。将设备添加到 Amazon IoT Core for LoRaWAN 使得 Amazon IoT 能够处理从设备接收的消息，以供 Amazon IoT 和其它服务使用。

以下是将您的设备与 Amazon IoT Core for LoRaWAN 搭配使用时需要注意的重要事项。有关如何将设备添加到 Amazon IoT Core for LoRaWAN 的信息，请参阅 将您的设备搭载到 Amazon IoT Core for LoRaWAN 上 (p. 1119)。

设备注意事项

选择要用于与 Amazon IoT Core for LoRaWAN 通信的设备时，请考虑以下事项：

- 可用传感器
- 电池容量
- 能源消耗
- 费用
- 天线类型和传输范围

使用具有可与 Amazon IoT Core for LoRaWAN 搭配使用的网关的设备

您使用的设备可以与能够同 Amazon IoT Core for LoRaWAN 搭配使用的无限网关配对。这些网关和开发人员工具包可在 Amazon 合作伙伴设备目录中找到。我们还建议您考虑将这些设备靠近您的网关。有关更多信息，请参阅使用来自Amazon Partner Device Catalog 的合格网关 (p. 1146)：

LoRa广域网版本

Amazon IoT Core for LoRaWAN支持所有符合 Al LoRa liance 标准化的 1.0.x 或 1.1 LoRa WAN 规范的设备。

激活模式

在 LoRa WAN 设备能够发送上行链路数据之前，您必须完成 activation 或者 join p roc ed ure 流程。要激活您的设备，您可以使用 OTAA（空中激活）或 ABP（通过个性化激活）。我们建议您使用 OTAA 激活您的设备，因为每次激活都会生成新的会话密钥，这样更加安全。

您的无线设备规范基于 LoRa WAN 版本和激活模式，该模式决定了为每次激活生成的根密钥和会话密钥。有关更多信息，请参阅使用控制台将您的无线设备规范添加到 Amazon IoT Core for LoRaWAN (p. 1120)：

设备类

LoRaWAN 设备可以随时发送上行链路消息。侦听下行链路消息会消耗电池容量并缩短电池持续时间。LoRaWAN 协议指定了三类 LoRa WAN 设备。

- A 类设备大部分时间都处于睡眠状态，并且仅在短时间内侦听下行链路消息。这些设备主要是电池供电的传感器，电池寿命长达 10 年。
- B 类设备可以在计划的下行链路插槽中接收消息。这些设备主要是电池供电的执行器。
- C 类设备永远不会睡眠，并会持续侦听传入的消息，因此接收消息时没有太多延迟。这些器件主要是电源驱动器。

有关这些无线设备注意事项的更多信息，请参阅 了解有关 LoRaWAN 的更多信息 (p. 1108) 中提及的资源。

管理 LoRa WAN 设备与之间的通信Amazon IoT

将 LoRa WAN 设备连接到之后Amazon IoT Core for LoRaWAN，您的设备可以开始向云发送消息。上行链路消息是从您设备发送并由 Amazon IoT Core for LoRaWAN 接收的消息。您的 LoRa WAN 设备可以随时发送上行链路消息，随后会转发至其他Amazon Web Service和云托管应用程序。从 Amazon IoT Core for LoRaWAN 以及其他 Amazon Web Service 和应用程序发送到设备的消息称为下行链路消息。

下面演示了如何查看和管理在设备与云之间发送的上行链路和下行链路消息。您可以维护下行链路消息队列，然后按照这些消息的队列添加顺序将消息发送到设备。

主题

- 查看从 LoRa WAN 设备发送的上行链路消息的格式 (p. 1162)
- 管理要发送到 LoRa WAN 设备的下行链路消息队列 (p. 1164)

查看从 LoRa WAN 设备发送的上行链路消息的格式

将 LoRa WAN 设备连接到之后Amazon IoT Core for LoRaWAN，您可以观察将从无线设备接收的上行链路消息的格式。

在您观察上行链路消息之前

您必须已搭载无线设备并将设备连接到 Amazon IoT 以便它能够传输和接收数据。有关将您的设备搭载到 Amazon IoT Core for LoRaWAN 的更多消息，请参阅 将您的设备搭载到 Amazon IoT Core for LoRaWAN 上 (p. 1119)。

上行链路消息包含什么内容？

LoRaWAN 设备使用 LoRa WAN 网关连接到。Amazon IoT Core for LoRaWAN您从设备接收的上行链路消息将包含以下信息。

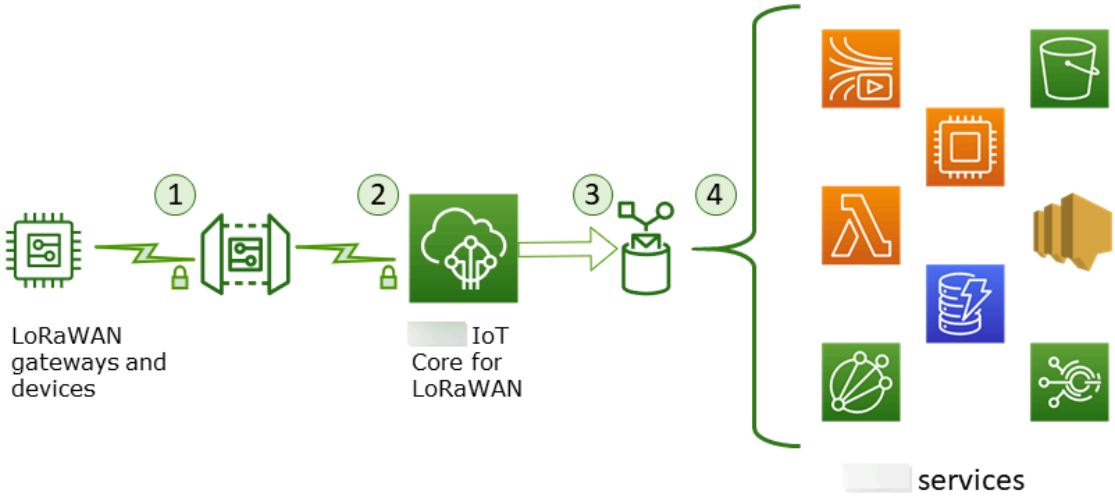
- 对应从无线设备发送的加密负载消息的负载数据。
- 无线元数据，包括：
  - 设备信息 DevEui，例如数据速率和设备运行所在的频率通道。
  - 连接到设备的网关的可选附加参数和网关信息。网关参数包括网关的 EUI、SNR 和 RSSi。

  通过使用无线元数据，您可以获取有关无线设备的有用信息以及在设备和 Amazon IoT 之间传输的数据。例如，您可以使用 `AckedMessageId` 参数来检查设备是否已收到上次确认的下行链路消息。或者，如果您选择包含网关信息，则可以确定是否要切换到更靠近设备的更强网关通道。

如何观察上行链路消息？

搭载设备后，您可以使用 Amazon IoT 控制台中 Test（测试）页面的 MQTT 测试客户端订阅创建目标时指定的主题。连接设备并开始发送负载数据后，您将开始看到消息。

此图表标识了连接到的 LoRa WAN 系统中的密钥元素Amazon IoT Core for LoRaWAN，其中显示了主数据层面以及数据在系统中流动的方式。



当无线设备开始发送上行链路数据时，Amazon IoT Core for LoRaWAN 将无线元数据信息与负载打包，然后将其发送到 Amazon 应用程序。

上行链路消息示例

以下示例显示了从您的设备接收的上行链路消息的格式。

Note

如果您的设备发送不带 Fport 值的上行链路消息，则 Amazon IoT Core for LoRaWAN 会将值 225 添加到接收的上行链路消息中的 Fport。

```
{
    "WirelessDeviceId": "5b58245e-146c-4c30-9703-0ca942e3ff35",
    "PayloadData": "Cc48AAAAAAAAAAA=",
    "WirelessMetadata":
    {
        "LoRaWAN":
        {
            "ADR": false,
            "Bandwidth": 125,
            "ClassB": false,
            "CodeRate": "4/5",
            "DataRate": "0",
            "DevAddr": "00b96cd4",
            "DevEui": "58a0cb000202c99",
            "FOptLen": 2,
            "FCnt": 1,
            "Fport": 136,
            "Frequency": "868100000",
            "Gateways": [
             {
                    "GatewayEui": "80029cfffe5cf1cc",
                    "Snr": -29,
                    "Rssi": 9.75
             }
             ],
            "MIC": "7255cb07",
            "MType": "UnconfirmedDataUp",
            "Major": "LoRaWANR1",
            "Modulation": "LORA",
            "PolarizationInversion": false,
            "SpreadingFactor": 12,
```

```
            "Timestamp": "2021-04-29T04:19:43Z"

        }
    }
}
```

下表显示了上行链路元数据中使用的字段的说明：

LoRaWAN 上行链路消息字段

参数	描述	类型	必填
WirelessDeviceID	发送数据的无线设备的 ID。	字符串	是
PayloadData	从设备收到的二进制消息，以 base64 编码。	字符串	是
WirelessMetadata	关于 LoRa WAN 设备和消息请求的元数据。这包括设备标识符、数据和代码速率、消息时间戳、是否启用 ADR（自适应数据速率）以及网关元数据等信息。	枚举	否

从上行链路元数据中排除网关元数据

如果要从上行链路元数据中排除网关元数据信息，请在您创建服务配置文件时禁用该AddGwMetadata参数。有关禁用此参数的信息，请参阅 添加服务配置文件 (p. 1123)。

在这种情况下，您将不会在上行链路元数据中看到 Gateways 部分，如以下示例所示。

```
{
    "WirelessDeviceId": "0d9a439b-e77a-4573-a791-49d5c0f4db95",
    "PayloadData": "AAAAAAAA//8=",
    "WirelessMetadata": {
        "LoRaWAN": {
            "ClassB": false,
            "CodeRate": "4/5",
            "DataRate": "1",
            "DevAddr": "01920f27",
            "DevEui": "ffffff10000163b0",
            "FCnt": 1,
            "FPort": 5,
            "Timestamp": "2021-04-29T04:19:43Z"
    }
  }
}
```

管理要发送到 LoRa WAN 设备的下行链路消息队列

云托管应用程序和其他 Amazon Web Service 可以向无线设备发送下行链路消息。下行链路消息是从 Amazon IoT Core for LoRaWAN 发送到无线设备的消息。您可以为已搭载到 Amazon IoT Core for LoRaWAN 的每台设备调度和发送下行链路消息。

如果要向多台设备发送下行链路消息，可以使用多播组。多播组中的设备共享同一个多播地址，将该地址分发到整组接收设备。有关更多信息，请参阅创建多播组向多台设备发送下行链路有效负载 (p. 1167)：

下行链路消息队列的工作原理

LoRaWAN 设备的设备类别决定了队列中的消息如何发送到设备。A 类设备将上行链路消息发送到 Amazon IoT Core for LoRaWAN，表明设备可以接收下行链路消息。B 类设备可在常规的下行链路槽接收消息。C 类设备可以随时接收下行链路消息。有关设备类别的更多信息，请参阅 设备类 (p. 1161)。

下面演示了如何形成消息队列以及如何将消息发送到 A 类设备。

1. Amazon IoT Core for LoRaWAN 借助您通过 Amazon IoT 控制台或 Amazon IoT Wireless API 指定的帧端口、有效负载数据和确认模式参数，对添加到队列中的下行链路消息进行缓冲。
2. 您的 LoRa WAN 设备会发送上行链路消息，表明其处于线上状态且可以开始接收下行链路消息。
3. 如果已将多条下行链路消息添加到队列中，Amazon IoT Core for LoRaWAN 会将队列中的第一条下行链路消息发送到已设置确认 (ACK) 标志的设备。
4. 设备可立即向 Amazon IoT Core for LoRaWAN 发送上行链路消息，也可休眠直至出现下一条上行链路消息且消息中包含 ACK 标志。
5. Amazon IoT Core for LoRaWAN 收到带有 ACK 标志的上行链路消息后，会清除队列中的下行链路消息，表示设备已成功收到下行链路消息。如果三次检查后，上行链路消息中仍缺少 ACK 标志，则会丢弃该消息。

使用控制台执行下行链路队列操作

您可以使用 Amazon Web Services Management Console 管理下行链路消息队列，以及根据需要清除单条消息或整个队列。如果为 A 类设备，收到从设备发送的表明设备处于线上状态的上行链路消息后，将队列消息发送到该设备。发送消息后，系统会自动从队列中清除该消息。

管理下行链路消息队列

创建下行链路消息队列

1. 转到 Amazon IoT 控制台的设备中心，然后选择要管理下行链路消息队列的设备。
2. 在设备详细信息页面的 Downlink messages（下行链路消息）部分，选择 Queue downlink messages（管理下行链路消息队列）。
3. 要配置下行链路消息，请指定以下参数：
   - FPort：选择设备与 Amazon IoT Core for LoRaWAN 通信的帧端口。
   - Payload（有效负载）：指定要发送到设备的有效负载消息。最大有效负载大小为 242 字节。如果已启用自适应数据速率 (ADR)，Amazon IoT Core for LoRaWAN 会用其选择有效负载大小的租价数据速率。您可以根据需要进一步优化数据速率。
   - Acknowledge mode（确认模式）：确认设备是否收到下行链路消息。如果消息需要此模式，数据流会显示一条带有 ACK 标志的上行链路消息，并且该消息将从队列中清除。
4. 要将下行链路消息添加到队列中，请选择 Submit（提交）。

下行链路消息现已添加到队列中。如果没有显示消息或收到错误，您可以如 排查下行链路消息队列错误 (p. 1166) 中所述排查该错误。

> **Note**
>
> 将下行链路消息添加到队列后，便无法继续编辑 FPort 参数、Payload（有效负载）和 Acknowledge mode（确认模式）。要为这些参数发送具有不同值的下行链路消息，您可以删除此消息，再对具有更新参数值的新下行链路消息进行队列管理。

队列会列出已添加的下行链路消息。若要查看设备与 Amazon IoT Core for LoRaWAN 之间交换的上行链路和下行链路消息的有效负载，您可以使用网络分析器。有关更多信息，请参阅使用网络分析器实时监控无线资源机群 (p. 1184)：

列出下行链路消息队列

已将创建的下行链路消息添加到队列中。后续每条下行链路消息均会在此消息之后添加到队列中。您可以在设备详细信息页面的 Downlink messages（下行链路消息）部分查看下行链路消息列表。收到上行链路消息后，消息就会发送到设备。设备收到下行链路消息后，该消息会从队列中删除。然后，下一条消息会在队列中向前移动，等待发送到设备。

删除单条下行链路消息或清除整个队列

发送到设备后，每条下行链路消息都会自动从队列中清除。您还可以删除单条消息或清除整个下行链路队列。这些操作无法撤消。

- 如果发现队列中有不想发送的消息，请选中相应消息并选择 Delete（删除）。
- 如果不想将任何消息从队列发送到设备，请选择 Clear downlink queue（清除下行链路队列）来清除整个队列。

使用 API 执行下行链路队列操作

您可以使用 Amazon IoT Wireless API 管理下行链路消息队列，以及根据需要清除单条消息或整个队列。

管理下行链路消息队列

若要创建下行链路消息队列，请使用 SendDataToWirelessDevice API 操作或 send-data-to-wireless-device CLI 命令。

> Note
>
> 使用 SendDataToWirelessDevice API 发送下行链路消息时，您可以选择要用于下行链路数据流量的网关。有关更多信息，请参阅选择网关以接收 LoRa WAN 下行链路数据流量 (p. 1159)：

```
aws iotwireless send-data-to-wireless-device \
    --id "11aa5eae-2f56-4b8e-a023-b28d98494e49" \
    --transmit-mode "1" \
    --payload-data "SGVsbG8gVG8gRGV2c2lt" \
    --wireless-metadata LoRaWAN={FPort=1}
```

运行此命令的输出会生成下行链路消息的 MessageId。在某些情况下，即使收到 MessageId，数据包也可能丢失。有关如何解决错误的更多信息，请参阅 排查下行链路消息队列错误 (p. 1166)。

```
{
    MessageId: "6011dd36-0043d6eb-0072-0008"
}
```

列出队列中的下行链路消息

若要列出队列中的所有下行链路消息，请使用 ListQueuedMessages API 操作或 list-queued-messages CLI 命令。

```
aws iotwireless list-queued-messages
```

默认情况下，运行此命令时最多会显示 10 条下行链路消息。

删除单条下行链路消息或清除整个队列

若要从队列中删除单条消息或清除整个队列，请使用 DeleteQueuedMessages API 操作或 delete-queued-messages CLI 命令。

- 若要删除单条消息，请针对无线设备（由 wirelessDeviceId 指定）提供待删除消息的 messageID。
- 若要清除整个下行链路消息队列，请针对无线设备（由 wirelessDeviceId 指定）将 messageID 指定为 *。

排查下行链路消息队列错误

如果没有看到期望的结果，可检查以下事项：

- Amazon IoT 控制台中未显示下行链路消息

如果如 使用控制台执行下行链路队列操作 (p. 1165) 中所述添加下行链路消息后，但队列中并未显示下行链路消息，可能是因为设备尚未完成名为 activation（激活）或 join（联接）过程。当设备搭载到 Amazon IoT Core for LoRaWAN，此过程才算完成。有关更多信息，请参阅使用控制台将您的无线设备规范添加到 Amazon IoT Core for LoRaWAN (p. 1120)：

将设备搭载到后Amazon IoT Core for LoRaWAN，您可以使用网络分析器或者 Amazon 监控设备来检查联接和重新联接是否成功 CloudWatch。有关更多信息，请参阅监控和记录Amazon IoT Wireless使用亚马逊的情况 CloudWatch (p. 1248)：

- 使用 API 时缺少下行链路消息包

  使用 `SendDataToWirelessDevice` API 操作时，API 会返回唯一的 `MessageId`。不过，其无法确认 LoRa WAN 设备是否收到下行链路消息。在设备尚未完成联接过程等情况下，下行链路数据包可能遭到丢弃。有关如何解决此错误的更多信息，请参阅上一节的内容。

- 发送下行链路消息时缺少 ARN 错误

  从队列向设备发送下行链路消息时，可能会出现缺少 Amazon Resource Name (ARN) 错误。出现此错误可能是因为尚未为接收下行链路消息的设备准确指定目标。若要解决此错误，请检查设备的目标详细信息。

创建多播组向多台设备发送下行链路有效负载

要向多台设备发送下行链路有效负载，请创建一个多播组。借助多播，源可以将数据发送到单个多播地址，然后分发到整组接收设备。

多播组中的设备共享同一个多播地址、会话密钥和帧计数器。通过使用相同的会话密钥，多播组中的设备可以在启动下行链路传输时解密消息。多播组仅支持下行链路。它不能确认设备是否接收了下行链路有效负载。

使用 Amazon IoT Core for LoRaWAN 多播组，您可以：

- 使用设备配置文件、RFRegion 或设备类别筛选您设备列表，然后将这些设备添加到多播组中。
- 在 48 小时的分发窗口内，调度一条或多条下行负载消息并向多播组中的设备发送一条或多条下行负载消息。
- 让设备在组播会话开始时临时切换到 B 类或 C 类模式接收下行链路消息。
- 监控多播组的设置及其设备的状态，并对发现的问题进行故障排除。
- 使用无线固件更新 (FUOTA) 将固件更新安全地部署到多播组中的设备。

Amazon IoT Core for LoRaWAN对 FUOTA 和多播组的支持基于LoRa 联盟的以下规范：

- LoRaWAN 远程组播设置规范，TS005-2.0.0
- LoRaWAN 碎片化数据块传输规范，TS004-2.0.0
- LoRaWAN 应用层时钟同步规范，TS003-2.0.0

> **Note**
>
> Amazon IoT Core for LoRaWAN根据A LoRa lliance 规范自动执行设备的时钟同步。使用函数`AppTimeReq`，运用 ClockSync 信令将服务器端时间回复到请求时间的设备。

下面介绍了如何创建多播组并调度下行链路消息。

主题

- 为多播和 FUOTA 配置准备设备 (p. 1168)
- 创建多播组并将设备添加到组 (p. 1170)

- 监控多播组和组中设备的状态并对进行故障排除 (p. 1173)
- 调度向多播组中的设备发送下行链路消息 (p. 1174)

为多播和 FUOTA 配置准备设备

将无线设备添加到 Amazon IoT Core for LoRaWAN 时，您可以使用控制台或 CLI 准备无线设备进行多播设置和 FUOTA 配置。如果您是第一次执行这一配置，我们建议您使用控制台。要管理多播组并从组中添加或删除多个设备，我们建议使用 CLI 管理大量资源。

GenAppKey 和 FPort

添加无线设备时，请先配置下列参数，然后再将您的设备添加到多播组或执行 FUOTA 更新。在配置这些参数之前，请确保您的设备支持 FUOTA 和组播，并且无线设备规格为 OTAA v1.1 或 OTAAv1.0.x。

- GenAppKey：对于支持 LoRa WAN 1.0.x 版本的设备以及要使用多播组的设备，GenAppKey是具体设备的根密钥，从中派生多播组的会话密钥。

  Note

  对于使用无线规范OTAA v1.1AppKey的 LoRa WAN 设备，的用途与GenAppKey。

  要设置参数来启动数据传输，Amazon IoT Core for LoRaWAN 与终端设备一起分发会话密钥。有关 LoRa WAN 版本的更多信息，请参阅LoRa广域网版本 (p. 1161)。

  Note

  Amazon IoT Core for LoRaWAN 存储了您以加密格式提供的 GenAppKey 信息。
- FPorts：根据 FUOTA 和多播组的 WAmazon IoT Core for LoRaWAN A LoRa N 规范，为FPorts参数的以下字段分配原定设置值。如果您已经分配了下列任意 FPort 值，您可以选择一个不同的可用值，从 1 到 223。
  - Multicast：200

    该 FPort 值用于多播组。
  - FUOTA：201

    该 FPort 值用于 FUOTA。
  - ClockSync：202

    该 FPort 值用于时钟同步。

多播和 FUOTA 的设备配置文件

在多播会话开始时，使用 B 类或 C 类分发窗口向组中的设备发送下行链路消息。为多播和 FUOTA 添加的设备必须支持 B 类或 C 类操作模式。根据您的设备支持的设备类别，选择启用 B 类或 C 类模式之一或两种模式的设备配置文件。

有关设备配置文件的更多信息，请参阅 将配置文件添加到 Amazon IoT Core for LoRaWAN (p. 1122)。

使用控制台为多播和 FUOTA 做好设备准备

要使用控制台指定多播设置的 FPort 和 GenAppKey 参数和 FUOTA，请执行以下操作：

1. 导航到 Amazon IoT 控制台的设备中心然后选择添加无线设备。
2. 选择无线设备规格。您的设备必须使用 OTAA 进行激活。选择 OTAA v1.0.x 或 OTAA v1.1 时，出现 FUOTA 配置-可选。
3. 输入无线设备的 EUI（扩展唯一标识符）参数。

4. 展开FUOTA 配置-可选部分然后选择该设备支持无线固件更新 (FUOTA)。现在，您可以输入多播、FUOTA 和时钟同步的 FPort 值。如果OTAA v1.0.x针对无线设备规格您选择了，请输入GenAppKey。
5. 通过选择您的个人资料和路由消息目的地，将您的设备添加到 Amazon IoT Core for LoRaWAN。对于链接到设备的设备配置文件，请确保选择支持 B 类或支持 C 类模式，或选择两种模式。

   Note

   要指定 FUOTA 配置参数，您必须使用 Amazon IoT 控制台的设备中心。如果您使用介绍的页面上的 Amazon IoT 控制台，不会出现这些参数。

有关无线设备规格和启动设备的更多信息，请参阅 将您的无线设备添加到 Amazon IoT Core for LoRaWAN (p. 1120)。

Note

您只能在创建无线设备时指定这些参数。更新现有设备时无法更改或指定参数。

使用 API 操作为多播和 FUOTA 准备设备

要使用多播组或执行 FUOTA 更新，请使用 `CreateWirelessDevice` API 操作或 `create-wireless-device`CLI 命令确认这些参数。除了指定应用程序密钥和 FPort 参数之外，请确保链接到设备的设备配置文件支持 B 类或 C 类模式或两者都支持。

您可以将 `input.json` 文件作为 `create-wireless-device` 命令的输入。

```
aws iotwireless create-wireless-device \
    --cli-input-json file://input.json
```

其中：

input.json 的内容

```
{
    "Description": "My LoRaWAN wireless device"
    "DestinationName": "IoTWirelessDestination"
    "LoRaWAN": {
        "DeviceProfileId": "ab0c23d3-b001-45ef-6a01-2bc3de4f5333",
        "ServiceProfileId": "fe98dc76-cd12-001e-2d34-5550432da100",
        "FPorts": {
            "ClockSync": 202,
            "Fuota": 201,
            "Multicast": 200
      },
        "OtaaV1_0_x": {
            "AppKey": "3f4ca100e2fc675ea123f4eb12c4a012",
            "AppEui": "b4c231a359bc2e3d",
            "GenAppKey": "01c3f004a2d6efffe32c4eda14bcd2b4"
        },
        "DevEui": "ac12efc654d23fc2"
    },
    "Name": "SampleIoTWirelessThing"
    "Type": LoRaWAN
}
```

有关您可以使用的 CLI 的信息，请参阅 Amazon CLI 参考。

Note

指定这些参数的值之后，无法通过 `UpdateWirelessDevice` API 操作进行更新。相反，您可以使用参数 `GenAppKey` 和 `FPorts` 的值创建新的设备。

要获取这些参数指定值的信息，可以使用 GetWirelessDevice API 操作或 get-wireless-device CLI 命令。

后续步骤

配置参数后，您可以创建多播组和 FUOTA 任务来发送下行链路有效负载或更新 LoRa WAN 设备的固件。

- 有关创建多播组的更多信息，请参阅 创建多播组并将设备添加到组 (p. 1170)。
- 有关创建 FUOTA 任务的更多信息，请参阅 创建 FUOTA 任务并提供固件映像 (p. 1178)。

创建多播组并将设备添加到组

您可以使用控制台或 CLI 创建多播组。如果您是第一次创建多播组，我们建议您使用控制台添加多播组。如果要管理多播组并在组中添加或删除设备时，可以使用 CLI。

在与添加的终端设备交换信令之后，Amazon IoT Core for LoRaWAN 与终端设备建立共享密钥并设置数据传输的参数。

先决条件

在创建多播组并将设备添加到组之前，请执行以下操作：

- 通过指定 FUOTA 配置参数 GenAppKey 和 FPorts，为设备多播和 FUOTA 设置做好准备。有关更多信息，请参阅为多播和 FUOTA 配置准备设备 (p. 1168)：
- 检查设备是否支持 B 类或 C 类操作模式。根据您的设备支持的设备类别，选择支持 B 类或支持 C 类或两者都支持的的设备配置文件。有关设备配置文件的更多信息，请参阅 将配置文件添加到 Amazon IoT Core for LoRaWAN (p. 1122)。

  在多播会话开始时，使用 B 类或 C 类分发窗口向组中的设备发送下行链路消息。

使用控制台查创建多播组

要使用控制台创建多播组，请转至 Amazon IoT 控制台的 Multicast groups（多播组）页面并选择 Create multicast group（创建多播组）。

1. 创建多播组

   要创建您的多播组，请为组指定多播属性和标签。

   1. 指定多播属性

      要指定多播属性，请为多播组输入下列信息。
      - 名称：为您的多播组输入唯一的名称。名称只能包含字母、数字、连字符和下划线。它不能包含空格。
      - 说明：您可以为多播组提供可选描述。描述的长度最多为 2,048 个字符。

   2. 多播组的标签

      您可以选择提供任何键值作为多播组的 标签。要继续创建多播组，请选择 Next（下一步）。
2. 将设备添加到多播组

   您可以将单个设备或一组设备添加到多播组中。要添加设备：

   1. 指定 RFRegion

      为您的多播组指定RfRegion或频段。的您的多播组RfRegion必须匹配您在多播组中添加设备的 RfRegion。有关 RfRegion 的更多信息，请参阅 为您的网关和设备连接选择 LoRa 频段 (p. 1113)。

2. 选择多播设备类别

   在多播会话开始时，选择您希望多播组中的设备切换到 B 类还是 C 类模式。B 类会话可以在常规的下行链路槽接收下行链路消息，C 类会话可以随时接收下行链路消息。

3. 选择您想要添加在组中的设备。

   选择您要在多播组中单独添加设备还是批量添加设备。

   - 要单独添加设备，请输入要添加到组的每台设备的无线设备 ID。
   - 要批量添加设备，您可以按设备配置文件或标签筛选要添加的设备。对于设备配置文件，您可以添加带支持 B 类、C 类或支持两者的配置文件的设备。

4. 要创建多播组，请选择 Create（创建）。

   组 中出现多播组详细信息和您添加的设备 。有关多播组和设备的状态以及对任何问题的故障排除信息，请参阅 监控多播组和组中设备的状态并对进行故障排除 (p. 1173)。

创建多播组后，您可以选择 Action（操作）编辑、删除或将设备添加到多播组中。添加设备后，您可以调度会话，将下行链路负载发送到组中的设备。

使用 API 创建多播组

要使用 API 创建多播组并将设备添加到组中，请执行以下操作：

1. 创建多播组

   要创建多播组，请使用CreateMulticastGroup API 操作或 create-multicast-groupCLI 命令。您可以将 input.json 文件作为 create-multicast-group 命令的输入。

```
aws iotwireless create-multicast-group \
    --cli-input-json file://input.json
```

   其中：

   input.json 的内容

```
{
   "Description": "Multicast group to send downlink payload and perform FUOTA
 updates.",
   "LoRaWAN": {
      "DlClass": "ClassB",
      "RfRegion": "US915"
   },
   "Name": "MC_group_FUOTA"
}
```

   创建多播组后，您可以使用下列 API 操作或 CLI 命令来更新、删除或获取有关多播组的信息。

   - UpdateMulticastGroup 或者 update-multicast-group
   - GetMulticastGroup 或者 get-multicast-group
   - ListMulticastGroups 或者 list-multicast-groups
   - DeleteMulticastGroup 或者 delete-multicast-group

2. 将设备添加到多播组

   您可以将设备单独添加或批量添加到多播组。

   - 要将设备批量添加到多播组，请使用 StartBulkAssociateWirelessDeviceWithMulticastGroup API 操作或 start-bulk-

associate-wireless-device-with-multicast-groupCLI 命令。要筛选批量添加到多播组的设备，请提供查询字符串。下面说明了如何通过相关的特定 ID 添加一组具有配置文件的设备。

```
aws iotwireless start-bulk-associate-wireless-device-with-multicast-group \
    --id "12abd34e-5f67-89c2-9293-593b1bd862e0" \
    --cli-input-json file://input.json
```

其中：

input.json 的内容

```
{
     "QueryString": "DeviceProfileName: MyWirelessDevice AND DeviceProfileId:
 d6d8ef8e-7045-496d-b3f4-ebcaa1d564bf",
     "Tags": [
        {
            "Key": "Multicast",
            "Value": "ClassB"
        }
    ]
}
```

在这里，multicast-groups/d6d8ef8e-7045-496d-b3f4-ebcaa1d564bf/bulk 是用于将设备与组进行关联的 URL。

- 要将设备单独添加到多播组中，请使用 AssociateWirelessDeviceWithMulticastGroup API 操作或 associate-wireless-device-with-multicast-group CLI。为要添加到组的每个设备提供无线设备 ID。

```
aws iotwireless associate-wireless-device-with-multicast-group \
    --id "12abd34e-5f67-89c2-9293-593b1bd862e0" \
    --wireless-device-id "ab0c23d3-b001-45ef-6a01-2bc3de4f5333"
```

创建多播组后，您可以使用以下 API 操作或 CLI 命令获取有关多播组的信息或取消设备的关联。

- DisassociateWirelessDeviceFromMulticastGroup 或者 disassociate-wireless-device-from-multicast-group
- StartBulkDisassociateWirelessDeviceFromMulticastGroup 或者 start-bulk-disassociate-wireless-device-from-multicast-group
- ListWirelessDevices 或者 list-wireless-devices

  Note

  ListWirelessDevices API 操作可用于列出一般的无线设备，以及与多播组或 FUOTA 任务关联的无线设备。

  - 要列出与多播组关联的无线设备，请将 ListWirelessDevices API 操作和 MulticastGroupID 用作筛选器。
  - 要列出与 FUOTA 任务关联的无线设备，请将 ListWirelessDevices API 操作和 FuotaTaskID 用作筛选器。

后续步骤

创建多播组并添加设备后，您可以继续添加设备并监控多播组和设备的状态。如果您的设备已成功添加到组中，可以配置并调度向设备发送下行链路消息。在发送下行链路消息之前，设备的状态必须为多播设置已就绪。调度下行链路消息后，状态将变为尝试会话。有关更多信息，请参阅调度向多播组中的设备发送下行链路消息 (p. 1174)：

如果要更新多播组中设备的固件，可以通过以下方式用 Amazon IoT Core for LoRaWAN 执行无线固件更新 (FUOTA)。有关更多信息，请参阅Amazon IoT Core for LoRaWAN 设备的无线固件更新 (FUOTA) (p. 1176)：

如果没有添加设备，或如果在多播组或设备状态中看到错误，可以将鼠标悬停在错误上获取更多信息并解决问题。如果错误仍然存在，有关如何排查和解决问题的信息，请参阅 监控多播组和组中设备的状态并对进行故障排除 (p. 1173)。

监控多播组和组中设备的状态并对进行故障排除

添加设备并创建多播组后，请打开 Amazon Web Services Management Console。导航到 Amazon IoT 控制台Multicast groups（多播组）的页面并选择您创建的多播组查看详细信息。您将看到有关多播组的信息、已添加的设备数量以及 设备 状态的详细信息。您可以使用状态信息跟踪多播会话的进度并对任何错误进行故障排除。

多播组状态

您的多播组可以在 Amazon Web Services Management Console 中显示如下状态消息。

- Pending (待处理)

  此状态表示您已经创建了多播组，但还没有多播会话。创建群组后，您将看到显示此状态消息。在此期间，您可以更新多播组，并将设备与组关联或取消关联。在状态从 Pending（待定）更改后，其他设备不能添加到组中。
- 尝试会话

  您的设备成功添加到多播组之后，如果您的组有已调度的多播会话，您将看到显示此状态消息。在此期间，您无法更新或将设备添加到您的多播组。如果取消了多播会话，组状态将更改为 Pending（待定）。
- 在会话中

  出现最早多播会话 时间时，您将看到显示此状态消息。当多播组与 FUOTA 任务（正在进行的固件更新会话）相关联时，多播组会保持在这个状态。

  如果会话中没有关联的 FUOTA 任务，且由于会话时间超时而取消多播会话或您取消了多播会话，则组状态将更改为 Pending（待定）。
- 等待删除

  如果删除多播组，组状态将更改为 Delete waiting（等待删除）。删除操作是永久性的，无法撤消。此操作可能需要时间，在多播组删除之前，组的状态是 Delete_Waiting（等待删除）。多播组进入此状态后，无法转换到其他状态。

多播组中设备的状态

多播组中的设备可以在 Amazon Web Services Management Console 中显示下列状态消息。您可以将鼠标悬停在每条状态消息上获取表示内容的更多信息

- 尝试程序包

  设备与多播组关联后，设备状态为尝试程序包。这种状态表明 Amazon IoT Core for LoRaWAN 尚未确认设备是否支持多播设置和操作。
- 不支持的程序包

  在您的设备与多播组关联后，Amazon IoT Core for LoRaWAN 检查设备的固件是否能够进行多播设置和操作。如果您的设备没有支持的多播软件包，其状态为不支持程序包。要纠正错误，请检查设备的固件是否能够进行多播设置和操作。
- 尝试多播设置

如果与多播组关联的设备能够进行多播设置和操作，状态为尝试多播设置。此状态表示设备尚未完成组播设置。

- 多播设置就绪

  您的设备已完成组播设置并已添加到多播组中。此状态表示设备已准备好进行多播会话，并且可以向这些设备发送下行链路消息。状态还表示您可以使用 FUOTA 更新组中设备固件的时间。

- 尝试会话

  已为多播组中的设备调度了多播会话。在多播组会话开始时，设备状态为 Session attempting（尝试会话），并且请求已发送，以了解是否可以为会话启动 B 类或 C 类分发窗口。如果设置多播会话所需的时间超时或取消了多播会话，状态将更改为多播设置完成。

- 在会话中

  此状态表示 B 类或 C 类分发窗口已启动，您的设备有正在进行的多播会话。在此期间，Amazon IoT Core for LoRaWAN 中的下行链路消息可以发送到多播组中的设备。如果更新会话时间，会覆盖当前会话，状态更改为尝试会话。当会话时间结束或取消多播会话时，状态将更改为多播设置就绪。

后续步骤

现在，您已经了解了多播组和组中设备的不同状态，以及如何解决问题（例如设备无法设置多播时），您可以计划向设备发送下行链路消息，然后您的多播组将处于会话中的状态。有关调度下行链路消息的信息，请参阅 调度向多播组中的设备发送下行链路消息 (p. 1174)。

调度向多播组中的设备发送下行链路消息

成功将设备添加到多播组后，您可以启动多播会话并配置要发送到这些设备的下行链路消息。必须在 48 小时之内 调度下行链路消息，组播的开始时间必须 至少比当前时间晚 30 分钟。

> Note
>
> 多播组中的设备无法确认收到下行链路消息的时间。

先决条件

在发送下行链路消息之前，您必须已创建多播组并成功将设备添加到要发送下行链路消息的组中。在为多播会话调度了开始时间后，您无法添加更多的设备。有关更多信息，请参阅创建多播组并将设备添加到组 (p. 1170)：

如果没有成功添加任何设备，多播组和设备状态将包含帮助您纠正错误的信息。如果错误仍然存在，有关纠正这些错误的信息，请参阅 监控多播组和组中设备的状态并对进行故障排除 (p. 1173)。

使用控制台调度下行链路消息

要使用控制台发送下行链路消息，请转至 Amazon IoT 控制台的多播组的页面，然后选择您创建的多播组。在多播组详细信息页面中，选择调度下行链路消息然后选择调度下行会话。

1. 调度下行链路消息窗口

   您可以设置一个时间窗口，以便向多播组中的设备发送下行链路消息。必须在 48 小时内调度下行链路消息。

   要调度多播会话，请指定以下参数：

   - Start date（开始日期）和 Start time（开始时间）：开始日期和时间必须至少为当前时间之后的 30 分钟和当前时间之前的 48 小时。

Note

您指定的时间是 UTC，因此在调度下行链路窗口时，请考虑检查与时区的时差。

- 会话超时：如果没有收到下行链路消息，您希望多播会话超时之后的时间。最短超时时间为 60 秒。B 类多播组的最长超时时间为 2 天，C 类多播组的最长超时时间为 18 小时。

2. 配置下行链路消息

   要配置下行链路消息，请指定以下参数：

   - 数据速率：为下行链路消息选择数据速率。数据速率取决于 RFRegion 和有效载荷大小。US915 区域的原定设置数据速率为 8，EU868 区域的原定设置数据速率为 0。
   - 频率：选择发送下行链路消息的频率。为避免消息冲突，请根据 RFRegion 选择可用频率。
   - FPort：选择可用的频率端口将下行链路消息发送到设备。
   - 有效负载：根据数据速率指定的最大有效负载。使用原定设置数据速率，您可以在 US915 RfRegion 中拥有 33 个字节的最大有效负，在 EU868 中为 51 个字节 RfRegion。使用较大的数据速率，您可以传输最多 242 字节的有效负载。

   要调度下行链路消息，请选择 Schedule（调度）。

使用 API 调度下行链路消息

要使用 API 调度下行链路消息，请使用 `StartMulticastGroupSession` API 操作或 `start-multicast-group-session`CLI 命令。您可以使用 `GetMulticastGroupSession` API 操作或 `get-multicast-group-session` CLI 命令来获取有关组播组的信息。

要在会话启动后向多播组发送数据，请使用`SendDataToMulticastGroup` API 操作或 `send-data-to-multicast-group`CLI 命令。

后续步骤

将下行链路消息配置为发送到设备后，该消息将在会话开始时发送。多播组中的设备无法确认是否已收到消息。

配置其他下行链路消息

您还可以配置其他下行链路消息发送到多播组中的设备：

- 要从控制台配置其他下行链路消息：
  1. 转至 Amazon IoT 控制台的 Multicast groups（多播组）的页面，然后选择您创建的多播组。
  2. 在多播组详细信息页面中，选择 Schedule downlink message（调度下行链路消息）然后选择 Configure additional downlink message（配置额外的下行链路消息）。
  3. 指定 Data rate（数据速率）、Frequency（频率）、FPort 和 Payload（有效负载）等参数，类似于为第一条下行链路消息配置这些参数的方式。
- 要使用 API 或 CLI 配置其他下行链路消息，请调用 `SendDataToMulticastGroup` API 操作或 `send-data-to-multicast-group`每条附加下行链路消息的 CLI 命令。

更新会话调度

您还可以更新会话调度，为多播会话使用新的开始日期和时间。新的会话调度将覆盖之前调度的会话。

Note

仅在需要时更新您的多播会话。这些更新可能会导致一组设备长时间唤醒并耗尽电池。

- 要从控制台更新会话调度：
  1. 转至 Amazon IoT 控制台的 Multicast groups（多播组）的页面，选择您创建的多播组。
  2. 在多播组详细信息页面中，选择 Schedule downlink message（安排下行链路消息）然后选择Update session schedule（更新会话调度）。
  3. 指定状态日期、开始时间和会话超时等参数，类似于您为第一条下行链路消息指定这些参数的方式。
- 要在 API 或 CLI 更新会话调度，请使用`StartMulticastGroupSession` API 操作或 `start-multicast-group-session`CLI 命令。

Amazon IoT Core for LoRaWAN 设备的无线固件更新 (FUOTA)

使用无线固件更新 (FUOTA) 将固件更新部署到 Amazon IoT Core for LoRaWAN 设备。

借助 FUOTA，您可以将固件更新发送到单个设备或一组设备。您还可以通过创建多播组向多台设备发送固件更新。首先将设备添加到多播组，然后将固件更新映像发送到所有这些设备。我们建议您对固件映像进行数字签名，以便接收映像的设备可以验证它们来自正确的来源。

借助 Amazon IoT Core for LoRaWAN 的 FUOTA 更新，您可以：

- 将新的固件映像或delta 映像部署到单个设备、一组设备中。
- 在新固件部署到设备之后，验证其真实性和完整性。
- 监控部署进度并在部署失败的情况下调试问题。

Amazon IoT Core for LoRaWAN对 FUOTA 和多播组的支持基于LoRa 联盟的以下规范：

- LoRaWAN 远程组播设置规范，TS005-2.0.0
- LoRaWAN 碎片化数据块传输规范，TS004-2.0.0
- LoRaWAN 应用层时钟同步规范，TS003-2.0.0

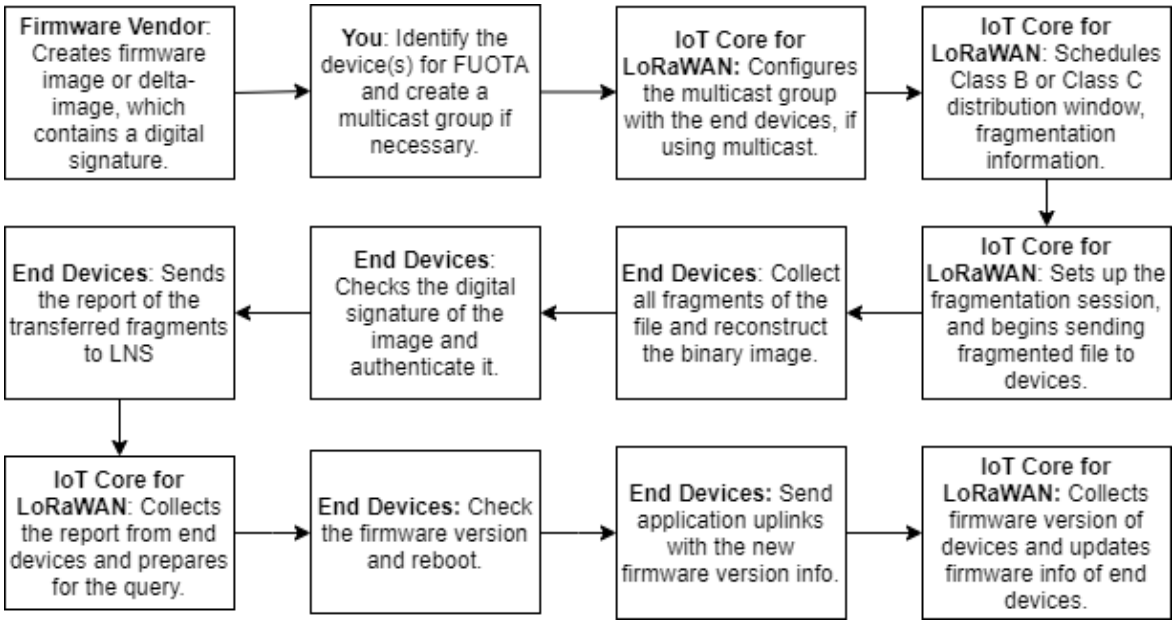
> **Note**
>
> Amazon IoT Core for LoRaWAN根据 LoRa 联盟规范自动执行时钟同步。它使用该函数`AppTimeReq`将服务器端时间回复使用 ClockSync信令请求时间的设备。

下面演示了如何执行 FUOTA 更新。

- FUOTA 过程概述 (p. 1176)
- 创建 FUOTA 任务并提供固件映像 (p. 1178)
- 将设备和多播组添加到 FUOTA 任务并调度 FUOTA 会话 (p. 1180)
- 监控 FUOTA 任务和添加到该任务的设备的状态并进行故障排除 (p. 1182)

FUOTA 过程概述

下图说明了 Amazon IoT Core for LoRaWAN 如何为终端设备执行 FUOTA 流程。如果要向 FUOTA 会话添加单个设备，可以跳过创建和配置多播组的步骤。您可以直接将设备添加到 FUOTA 会话中，然后 Amazon IoT Core for LoRaWAN 将启动固件更新过程。



要为设备执行 FUOTA 更新，请首先创建数字签名固件映像，然后配置要添加到 FUOTA 任务中的设备和多播组。启动 FUOTA 会话后，您的终端设备会收集所有碎片，从碎片中重建映像，将状态报告至 Amazon IoT Core for LoRaWAN，然后应用新的固件映像。

下面说明了 FUOTA 流程的不同步骤：

1. 使用数字签名创建固件映像或增量映像

   对于Amazon IoT Core for LoRaWAN要为 LoRa WAN 设备执行的 FUOTA 更新，我们建议您通过无线发送固件更新时对固件映像或增量映像进行数字签名。然后，接收映像的设备可以验证映像来自正确的来源。

   固件映像的大小不得超过 1 兆字节。固件越大，更新过程完成所需的时间就越长。要更快地传输数据，或如果您的新映像大于 1 兆字节，请使用增量映像，这是新映像的一部分，即新固件映像和上一个映像之间的增量。

   > Note
   >
   > Amazon IoT Core for LoRaWAN 不提供数字签名生成工具和固件版本管理系统。您可以使用任何第三方工具为固件映像生成数字签名。我们建议使用数字签名工具，例如嵌入在 ARM mbed GitHub 存储库的工具，其中还包括用于生成增量映像和设备使用该映像的工具。

2. 识别和配置 FUOTA 的设备

   识别 FUOTA 的设备后，将固件更新发送到单个或多个设备。

   - 要将固件更新发送到多台设备，请创建多播组，然后使用终端设备配置多播组。有关更多信息，请参阅创建多播组向多台设备发送下行链路有效负载 (p. 1167)：
   - 要向各个设备发送固件更新，请将这些设备添加到 FUOTA 会话中，然后执行固件更新。

3. 调度分发窗口并设置碎片会话

   如果创建了多播组，可以指定 B 类或 C 类分发窗口，确定设备可以从 Amazon IoT Core for LoRaWAN 接收碎片的时间。在切换到 B 类或 C 类模式之前，您的设备可能在 A 类中运行。您还必须指定会话的开始时间。

   B 类或 C 类设备会在指定的分发窗口唤醒，然后开始接收下行链路数据包。在 C 类模式下运行的设备可以消耗比 B 类设备更多的电量。有关更多信息，请参阅设备类 (p. 1161)：

4. 终端设备将状态报告至 Amazon IoT Core for LoRaWAN 并更新固件映像

   设置碎片会话后，您的终端设备和 Amazon IoT Core for LoRaWAN 执行以下步骤以更新设备的固件。

   1. 由于 LoRa WAN 设备的数据速率较低，要启动 FUOTA 流程，Amazon IoT Core for LoRaWAN设置碎片会话以对固件映像进行碎片化。 然后 会将这些碎片发送到终端设备。
   2. Amazon IoT Core for LoRaWAN发送映像碎片后，W LoRa AN 终端设备执行以下任务。
      a. 收集碎片，然后从这些碎片中重建二进制图像。
      b. 检查重建后映像的数字签名以验证映像并验证来自正确的来源。
      c. 比较来自的固件版本 Amazon IoT Core for LoRaWAN 转至当前版本。
      d. 报告转移到 Amazon IoT Core for LoRaWAN 的碎片映像状态，然后应用新的固件映像。

      Note

      在某些情况下，在检查固件映像的数字签名之前，终端设备报告传输到 Amazon IoT Core for LoRaWAN 的碎片映像的状态。

现在您已经了解了 FUOTA 流程，您可以创建 FUOTA 任务并将设备添加到任务中以更新固件。有关更多信息，请参阅创建 FUOTA 任务并提供固件映像 (p. 1178)：

创建 FUOTA 任务并提供固件映像

要更新 LoRa WAN 设备的固件，首先创建一个 FUOTA 任务并提供要用于更新的数字签名固件映像。然后，您可以将设备和多播组添加到任务中并调度 FUOTA 会话。会话开始时，Amazon IoT Core for LoRaWAN 设置碎片会话，最终设备收集碎片、重建映像并应用新固件。有关 FUOTA 过程的更多信息，请参阅 FUOTA 过程概述 (p. 1176)。

下面说明了如何创建 FUOTA 任务并上传要存储在 S3 存储桶中的固件映像或增量映像。

先决条件

在执行 FUOTA 更新之前，必须对固件映像进行数字签名，这样您的终端设备可以在应用映像时验证映像的真实性。您可以使用任何第三方工具为固件映像生成数字签名。我们建议使用数字签名工具，例如嵌入在 ARM mbed GitHub 存储库的工具，其中还包括用于生成增量映像和设备使用该映像的工具。

使用控制台创建 FUOTA 任务并上传固件映像

要创建 FUOTA 任务并使用控制台上传固件映像，请转到控制台的 FUOTA 任务 的选项卡，然后选择 Create FUOTA task（创建 FUOTA 任务）。

1. 创建 FUOTA 任务

   要创建 FUOTA 任务，请指定任务属性和标签。

   1. 指定 FUOTA 任务属性

      要指定 FUOTA 任务属性，请为 FUOTA 任务输入以下信息。

      - 名称：为 FUOTA 任务输入唯一的名称。名称只能包含字母、数字、连字符和下划线。它不能包含空格。
      - 说明：您可以为多播组提供可选描述。说明字段最多为 2,048 个字符。
      - RFRegion：为 FUOTA 任务设置频段。频段必须与您用于配置无线设备或多播组的频段相匹配。

   2. FUOTA 任务的标签

      您可以选择提供任何键值对作为的 FUOTA 任务的标签。要继续创建您的映像，选择 Next(下一步)。

2. 上传固件映像

   选择要用来更新添加到 FUOTA 任务设备固件的固件映像文件。固件映像文件存储在 S3 存储桶中。您可以为 Amazon IoT Core for LoRaWAN 提供代表您访问固件映像的权限。我们建议您对固件映像进行数字签名，以便在执行固件更新时验证其真实性。

   1. 选择固件映像文件

      您可以将新的固件映像文件上传到 S3 存储桶，也可以选择已上传到 S3 存储桶的现有映像。

      Note

      固件映像文件不得超过 1 兆字节。固件越大，更新过程完成所需的时间就越长。

      - 要使用现有映像，选择 Select an existing firmware image（选择现有的固件映像），选择 Browse S3（浏览 S3），然后选择要使用的固件映像文件。

        Amazon IoT Core for LoRaWAN 填充 S3 URL，该 URL 是 S3 存储桶中固件映像文件的路径。`s3://bucket_name/file_name` 路径的格式为：要在 Amazon Simple Storage Service控制台中查看文件，选择 View（查看）。
      - 要上传新的固件映像。
        a. 选择 Upload a new firmware image（上载新的固件映像），然后上载固件映像。映像文件不得大于 1 兆字节。
        b. 要创建 S3 存储桶并输入要存储固件映像文件的Bucket 名称，请选择 Create S3 bucket（创建 S3 存储桶）。
   2. 要访问存储桶的权限

      您可以创建新的服务角色或选择现有角色，让 Amazon IoT Core for LoRaWAN 代表您访问 S3 存储桶中的固件映像文件。选择下一步。

      要创建新角色，您可以输入角色名称或将其留白以便自动生成随机名称。要查看授予对 S3 存储桶访问权限的策略权限，选择 View policy permissions（查看策略权限）。

      Note

      如果将 S3 存储桶加密指定为`SSE-KMS`，则必须将 FUOTA 任务使用的服务角色添加为此 KMS 密钥的用户。此服务角色要么由 FUOTA 任务生成，要么在创建任务时分配给 FUOTA 任务。

   有关使用 S3 存储桶存储映像和授予 Amazon IoT Core for LoRaWAN 访问权限的更多信息，请参阅 将固件文件上载到 S3 存储桶并添加 IAM 角色 (p. 1153)。
3. 审核和创建

   要创建 FUOTA 任务，请查看您指定的 FUOTA 任务和配置详细信息，然后选择 Create task(创建任务)。

使用 API 创建 FUOTA 任务并上传固件映像

要使用 API创建 FUOTA 任务并指定固件映像文件，请使用 `CreateFuotaTask` API 操作或 `create-fuota-task`CLI 命令。您可以将 `input.json` 文件作为 `create-fuota-task` 命令的输入。使用 API 或 CLI 时，作为输入提供的固件映像文件必须已上传到 S3 存储桶。您还可以指定让 Amazon IoT Core for LoRaWAN 访问S3 存储桶中固件映像的 IAM 角色。

```
aws iotwireless create-fuota-task \
    --cli-input-json file://input.json
```

其中：

input.json 的内容

```
{
   "Description": "FUOTA task to update firmware of devices in multicast group.",
   "FirmwareUpdateImage": "S3:/firmware_bucket/firmware_image
   "FirmwareUpdateRole": "arn:aws:iam::123456789012:role/service-role/ACF1zBEI"
   "LoRaWAN": {
      "RfRegion": "US915"
   },
   "Name": "FUOTA_Task_MC"
}
```

创建 FUOTA 任务后，您可以使用以下 API 操作或 CLI 命令来更新、删除或获取有关 FUOTA 任务的信息。

- UpdateFuotaTask 或者 update-fuota-task
- GetFuotaTask 或者 get-fuota-task
- ListFuotaTasks 或者 list-fuota-tasks
- DeleteFuotaTask 或者 delete-fuota-task

后续步骤

现在，您已创建了 FUOTA 任务并提供了固件映像，您可以将设备添加到任务中更新固件。您可以将单个设备或多播组添加到任务中。有关更多信息，请参阅将设备和多播组添加到 FUOTA 任务并调度 FUOTA 会话 (p. 1180)：

将设备和多播组添加到 FUOTA 任务并调度 FUOTA 会话

创建 FUOTA 任务后，您可以将设备添加到要更新固件的任务中。将设备成功添加到 FUOTA 任务后，您可以调度 FUOTA 会话更新设备固件。

- 如果您只有少数设备，可以将这些设备直接添加到 FUOTA 任务中。
- 如果有大量设备 要更新固件 ，您可以将这些设备添加到多播组中，然后将多播组添加到 FUOTA 任务中。有关创建和使用多播组的信息，请参阅 创建多播组向多台设备发送下行链路有效负载 (p. 1167)。

> **Note**
>
> 您可以将单个设备或多播组添加到 FUOTA 任务中。您不能将设备和多播组添加到任务中。

添加设备或多播组后，您可以启动固件更新会话。Amazon IoT Core for LoRaWAN 收集固件映像，对映像进行碎片化，然后以加密格式存储碎片。您的终端设备收集碎片并应用新的固件映像。固件更新所需的时间取决于映像的大小和碎片方式。固件更新完成后，Amazon IoT Core for LoRaWAN 存储的固件映像的加密片段已删除。您仍然可以在 S3 存储桶中找到固件映像。

先决条件

在将设备或多播组添加到 FUOTA 任务之前，请执行以下操作。

- 您必须已创建 FUOTA 任务并提供固件映像。有关更多信息，请参阅创建 FUOTA 任务并提供固件映像 (p. 1178)：
- 配置要更新设备固件的无线设备。有关您的设备搭载的更多消息，请参阅 将您的设备搭载到 Amazon IoT Core for LoRaWAN 上 (p. 1119)。
- 要更新多台设备的固件，您可以将它们添加到多播组。有关更多信息，请参阅创建多播组向多台设备发送下行链路有效负载 (p. 1167)：

- 您将设备搭载到 Amazon IoT Core for LoRaWAN 上时，指定 FUOTA 配置参数 `FPorts`。如果使用的是 LoRa WAN 1.0.x 设备，您还必须指定`GenAppKey`。有关 FUOTA 配置参数的更多信息，请参阅 为多播和 FUOTA 配置准备设备 (p. 1168)。

将设备添加到 FUOTA 任务并使用控制台调度 FUOTA 会话

要使用控制台添加设备或多播组并调度 FUOTA 会话，请转到控制台的 FUOTA 任务选项卡。然后，选择要向其添加设备的 FUOTA 任务并执行固件更新。

添加设备和多播组

1. 您可以将单个设备或多播组添加到 FUOTA 任务中。但是，您不能将单个设备和多播组添加到同一个 FUOTA 任务中。要使用控制台添加设备，请执行以下操作。

   1. 在FUOTA 任务详细信息中，选择 Add device（添加设备）。
   2. 选择您添加到任务中的设备频段或 RFRegion。此值必须与您为 FUOTA 任务选择的RFRegion 匹配。
   3. 选择是否要将单个设备或多播组添加到任务。
      - 要添加单个设备，请选择 Add individual devices （添加单个设备），然后输入要添加到 FUOTA 任务中的设备 ID。
      - 要添加多播组，请选择 Add multicast groups（添加多播组），然后将多播组添加到任务中。您可以使用设备配置文件或标签筛选要添加到任务中的多播组。按设备配置文件进行筛选时，可以选择带配置文件设备的多播组，已启用支持 B 类或支持 C 类。
2. 调度 FUOTA 会话

   成功添加设备或多播组后，您可以调度 FUOTA 会话。要调度会话，请执行以下操作。

   1. 选择要更新其设备固件的 FUOTA 任务，然后选择 Schedule FUOTA session.（调度 FUOTA 会话）
   2. 为您的 FUOTA 会话指定开始日期和开始时间。确保开始时间是当前时间 30 分钟之后。

将设备添加到 FUOTA 任务并使用 API 调度 FUOTA 会话

您可以使用 Amazon IoT Wireless API 或 CLI 将无线设备或多播组添加到 FUOTA 任务中。然后，您可以调度 FUOTA 会话。

1. 添加设备和多播组

   您可以将无线设备或多播组与 FUOTA 任务关联起来。

   - 要将各单台设备与 FUOTA 任务关联，请使用 `AssociateWirelessDeviceWithFuotaTask` API 操作或 `associate-wireless-device-with-fuota-task` CLI 命令，然后将 `WirelessDeviceID` 作为输入。

     ```
     aws iotwireless associate-wireless-device-with-fuota-task \
         --id "01a23cde-5678-4a5b-ab1d-33456808ecb2"
         --wireless-device-id "ab0c23d3-b001-45ef-6a01-2bc3de4f5333"
     ```

   - 要将多播组与 FUOTA 任务关联，请使用 `AssociateMulticastGroupWithFuotaTask` API 操作或 `associate-multicast-group-with-fuota-task` CLI 命令，然后将 `MulticastGroupID` 作为输入。

     ```
     aws iotwireless associate-multicast-group-with-FUOTA-task \
         --id 01a23cde-5678-4a5b-ab1d-33456808ecb2"
         --multicast-group-id
     ```

将无线设备或多播组与 FUOTA 任务关联后，请使用以下 API 操作或 CLI 命令列出您的设备或多播组或将它们与任务取消关联。

- DisassociateWirelessDeviceFromFuotaTask 或者 disassociate-wireless-device-from-fuota-task
- DisassociateMulticastGroupFromFuotaTask 或者 disassociate-multicast-group-from-fuota-task
- ListWirelessDevices 或者 list-wireless-devices
- ListMulticastGroups 或者 list-multicast-groups-by-fuota-task

  Note

  API：

  - MulticastGroupID 作为筛选条件时，ListWirelessDevices 可以列出一般无线设备以及与多播组关联的设备。FuotaTaskID 作为筛选条件时，API 列出了与 FUOTA 任务关联的无线设备。
  - FuotaTaskID 作为筛选条件时，ListMulticastGroups 可以在下列情况下列出一般多播组和与 FUOTA 任务关联的多播组。

2. 调度 FUOTA 会话

   设备或多播组成功添加到 FUOTA 任务后，您可以启动 FUOTA 会话更新设备固件。开始时间离当前时间必须至少有 30 分钟。要使用 API 或 CLI 调度 FUOTA 会话，请使用StartFuotaTask API 操作或 start-fuota-taskCLI 命令。

   启动 FUOTA 会话后，您无法再向任务添加设备或多播组。您可以使用 GetFuotaTask API 操作或 get-fuota-taskCLI 命令来获取有关 FUOTA 会话状态的信息。

监控 FUOTA 任务和添加到该任务的设备的状态并进行故障排除

在配置无线设备并创建了可能要使用的任何多播组之后，可以通过执行以下步骤启动 FUOTA 会话。

FUOTA 任务状态

您的 FUOTA 任务可以在 Amazon Web Services Management Console 中显示下列状态消息。

- Pending (待处理)

  此状态表示您已经创建了 FUOTA 任务，但还没有固件更新会话。创建任务后，您将看到显示此状态消息。在此期间，您可以更新 FUOTA 任务， 关联 或者取消设备或多播组与任务的关联。在状态从Pending（待定）更改后，无法将其他设备添加到任务中。

- 等待 FUOTA 会话

  您的设备已经成功 添加 到 FUOTA 任务，当您的任务有已调度固件更新会话时，您将看到显示此状态消息。在此期间，您无法更新或将设备添加到 FUOTA 会话。如果您取消 FUOTA 会话，群组状态将更改为 Pending（待定。）

- 在 FUOTA 会话中

  当 FUOTA 会话开始时，您将看到显示此状态消息。碎片会话开始，您的终端设备收集碎片、重建固件映像、将新固件版本与原始版本进行比较，然后应用新映像。

- Fuota 已完成

  在您的终端设备向 Amazon IoT Core for LoRaWAN 报告已应用新的固件映像，或会话超时时，FUOTA 会话被标记为已完成，您将看到显示此状态。

在以下任何情况下，您还会显示看到此状态，因此请务必检查固件更新是否已正确应用于设备。

- 当 FUOTA 任务状态为等待 FUOTA 会话，并且存在 S3 存储桶错误时，例如指向 S3 存储桶中映像文件的链接不正确或 Amazon IoT Core for LoRaWAN 没有足够的权限来访问存储桶中的文件。
- 当 FUOTA 任务状态为等待 FUOTA 会话，并且有启动 FUOTA 会话的请求时，但是没有从 FUOTA 任务中的设备或多播组收到响应。
- 当 FUOTA 任务状态为 FUOTA 会话中时，并且设备或多播组在一段时间内没有发送任何碎片，这导致会话超时。

- 等待删除

  如果删除处于任何其他状态的 FUOTA 任务，会显示此状态。这是永久性操作，无法撤消。此操作可能需要时间，在 FUOTA 任务删除之前，任务状态为 Delete waiting （删除等待）。FUOTA 任务进入此状态后，无法转换到其他状态。

FUOTA 任务中设备的状态

FUOTA 任务中的设备可以在 Amazon Web Services Management Console 中显示下列状态消息。您可以将鼠标悬停在每条状态消息上获取表示内容的更多信息

- 初次

  在 FUOTA 会话的开始时间，Amazon IoT Core for LoRaWAN 检查您的设备是否有固件更新支持的软件包。如果您的设备有支持的软件包，该设备的 FUOTA 会话将启动。固件映像已碎片化，碎片将发送到您的设备。看到显示此状态时，表示设备的 FUOTA 会话尚未启动。

- 不支持的程序包

  如果设备不支持 FUOTA 软件包，您将看到显示此状态。如果不支持固件更新包，设备的 FUOTA 会话将无法启动。要纠正此错误，请检查设备的固件是否可以使用 FUOTA 接收固件更新。

- 不受支持碎片算法

  FUOTA 会话开始时，Amazon IoT Core for LoRaWAN 为设备设置碎片会话。如果您看到显示此状态，表示使用的碎片算法类型无法应用于设备的固件更新。出现错误的原因是您的设备没有支持的 FUOTA 软件包。要纠正此错误，请检查设备的固件是否可以使用 FUOTA 接收固件更新。

- 内存不足

  Amazon IoT Core for LoRaWAN 发送映像碎片之后，您的终端设备收集映像碎片并从这些碎片中重建二进制映像。当设备没有足够的内存来组装固件映像的传入碎片时，会显示此状态，这可能会导致固件更新会话过早结束。要纠正错误，请检查设备的硬件是否可以接收此更新。如果您的设备无法接收此更新，请使用增量映像更新固件。

- 不受支持碎片索引

  碎片索引标识了四个同时可能出现的碎片会话中的一个。如果您的设备不支持指定的碎片索引值，会显示此状态。要纠正这个错误，可以执行下列操作。

  - 为设备启动新的 FUOTA 任务。
  - 如果错误仍然存在，请将从单播模式切换到多播模式。
  - 如果错误仍未纠正，请检查设备固件。

- 内存错误

  此状态表示您的设备在接收来自 Amazon IoT Core for LoRaWAN 传入的片段时出现了内存错误。如果发生此错误，您的设备可能无法接收此更新。要纠正错误，请检查设备的硬件是否可以接收此更新。如果需要，请使用增量映像更新设备固件。

- 错误描述符

  您的设备不支持指定的描述符。描述符是描述在碎片会话期间传输的文件的字段。如果您看到此错误，请联系 Amazon Web Services Support 中心。

- 会话计数重播

  此状态表示您的设备之前使用过此会话计数。要纠正该错误，请启动设备的 FUOTA 新任务。

- 缺少碎片

  当您的设备从 Amazon IoT Core for LoRaWAN 中收集图像碎片时，它从独立的编码碎片中重建新的固件映像。如果您的设备尚未收到所有碎片，无法重建新映像，您将看到此状态。要纠正该错误，请启动设备的 FUOTA 新任务。

- MIC 错误

  当您的设备从收集的碎片中重建新固件映像时，会执行 MIC（消息完整性检查）来验证映像的真实性以及是否来自正确的来源。如果您的设备在重新组装碎片后检测到麦克风中不匹配，会显示此状态。要纠正此错误，请启动设备的 FUOTA 新任务。

- 成功

  您的设备 FUOTA 会话成功。

  Note

  尽管此状态消息表示设备已从碎片中重建映像并进行了验证，但设备将状态报告到时，设备固件可能尚未更新到 Amazon IoT Core for LoRaWAN。检查您的设备固件是否已更新。

后续步骤

您已经了解了 FUOTA 任务及其设备的不同状态以及如何解决问题。有关每个状态的更多信息，请参阅 LoRaWAN 碎片化数据块传输规范，TS004-1.0.0。

使用网络分析器实时监控无线资源机群

网络分析器使用原定设置 WebSocket 连接接收无线连接资源的实时跟踪消息日志。通过使用网络分析器，您可以添加要监控的资源、激活跟踪消息传递会话以及开始实时接收跟踪消息。

要监控您的资源，还可以使用 Amazon CloudWatch。要使用 CloudWatch，您需要设置 IAM 角色来配置日志记录，然后等待日志条目在控制台中显示。网络分析器显著缩短了建立连接和开始接收跟踪消息所需的时间，为您的资源机群提供了即时日志信息。有关使用 CloudWatch 监控的信息，请参阅 监控和记录Amazon IoT Wireless使用亚马逊的情况 CloudWatch (p. 1248)。

通过缩短设置时间并使用跟踪消息中的信息，您可以更有效地监控资源、获得有意义的见解并排查错误。您可以监控 LoRaWAN 设备和 LoRaWAN 网关。例如，在搭载其中一台 LoRaWAN 设备时，您可以快速识别加入错误。要调试错误，请使用提供的跟踪消息日志中的信息。

如何使用网络分析器

要监控资源机群并开始接收跟踪消息，请执行以下步骤

1. 创建网络分析器配置并添加资源

   在激活跟踪消息收发之前，请创建网络分析器配置并向配置添加资源。首先，指定配置设置，其中包括日志级别和无线设备帧信息。然后，通过使用无线网关和无线设备标识符添加要监控的无线资源。

2. 使用 WebSockets 流式传输跟踪消息

   您可以使用 IAM 角色的凭据生成预签名的请求 URL，使用 WebSocket 协议流式传输网络分析器跟踪消息。

3. 激活跟踪消息收发会话并监控跟踪消息

要开始接收跟踪消息，请激活跟踪消息会话。为了避免产生额外费用，您可以停用或关闭网络分析器跟踪消息会话。

下面介绍了如何创建配置、添加资源和激活跟踪消息收发会话。

主题

- 为网络分析器添加必要的 IAM 角色 (p. 1185)
- 创建网络分析器配置并添加资源 (p. 1186)
- 使用 WebSockets 串流网络分析器跟踪消息 (p. 1191)
- 实时查看和监控网络分析器跟踪消息日志 (p. 1199)

为网络分析器添加必要的 IAM 角色

使用网络分析器时，必须授予用户使用 API 操作 UpdateNetworkAnalyzerConfiguration 和 GetNetworkAnalyzerConfiguration 访问网络分析器资源的权限。下面显示了您用于授予权限的 IAM 策略。

网络分析器的 IAM 策略

请使用以下任一项：

- 完全访问无线策略

  通过将策略 AWSIoTWirelessFullAccess 附加到您的角色，授予 Amazon IoT Core for LoRaWAN 完全访问策略。有关更多信息，请参阅 AWSIoTWirelessFullAccess 策略摘要。
- 用于获取和更新 API 的限定范围的 IAM 策略

  通过转到 IAM 控制台的 Create policy（创建策略）页面，并在 Visual editor（可视化编辑器）选项卡上创建以下 IAM 策略：

  1. 对于 Service（服务），选择 IoTWireless。
  2. 在 Access level（访问级别）下，展开 Read（读取）并选择 GetNetworkAnalyzerConfiguration，然后展开 Write（写入）并选择 UpdateNetworkAnalyzerConfiguration。
  3. 选择 Next:Tags（下一步：标签），然后输入策略的 Name（名称），例如 IoTWirelessNetworkAnalyzerPolicy。选择 Create policy（创建策略）。

  以下内容显示您创建的策略 IoTWirelessNetworkAnalyzerPolicy。有关创建策略的更多信息，请参阅创建 IAM 策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iotwireless:GetNetworkAnalyzerConfiguration",
                "iotwireless:UpdateNetworkAnalyzerConfiguration"
            ],
            "Resource": "*"
        }
    ]
}
```

用于访问特定资源的限定范围的策略

要配置更精细的访问控制，您必须将无线网关和设备添加到 Resource（资源）字段中。以下策略使用通配符 ARN 授予对所有网关和设备的访问权限。您可以使用 `WirelessGatewayId` 和 `WirelessDeviceId` 来控制对特定网关和设备的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iotwireless:GetNetworkAnalyzerConfiguration",
                "iotwireless:UpdateNetworkAnalyzerConfiguration"
            ],
            "Resource": [
                "arn:aws:iotwireless:*:{accountId}:WirelessDevice/*",
                "arn:aws:iotwireless:*:{accountId}:WirelessGateway/*",
                "arn:aws:iotwireless:*:{accountId}:NetworkAnalyzerConfiguration/*"
            ]
        }
    ]
}
```

要授予用户使用网络分析器但不使用任何无线网关或设备的权限，请使用以下策略。除非指定，否则隐式拒绝使用资源的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iotwireless:GetNetworkAnalyzerConfiguration",
                "iotwireless:UpdateNetworkAnalyzerConfiguration"
            ],
            "Resource": [
                "arn:aws:iotwireless:*:{accountId}:NetworkAnalyzerConfiguration/*"
            ]
        }
    ]
}
```

后续步骤

现在您已创建策略，您可以将资源添加到网络分析器配置中，并接收这些资源的跟踪消息收发信息。有关更多信息，请参阅创建网络分析器配置并添加资源 (p. 1186)。

创建网络分析器配置并添加资源

在流式传输跟踪消息之前，请创建网络分析器配置并将要监控的资源添加到此配置中。创建配置时，您可以：

- 指定配置名称和可选说明。
- 自定义配置设置，例如帧信息和日志消息的详细级别。
- 确定您要监控的资源。资源可以是无线网关和/或无线设备。

您指定的配置设置将决定您将针对添加到配置的资源收到的跟踪消息收发信息。您可能还需要根据监控用例创建多个配置。

下面的内容说明了如何创建配置和添加资源。

主题

- 创建网络分析器配置 (p. 1187)
- 添加资源并更新网络分析器配置 (p. 1190)

创建网络分析器配置

必须先创建网络分析器配置，然后才能监控无线网关或无线设备。创建配置时，只需指定配置名称。即使在创建配置之后，您也可以自定义配置设置并将希望监控的资源添加到配置中。配置设置决定了您将针对这些资源收到的跟踪消息收发信息。

根据要监控的资源以及希望为其接收的信息级别，您可能需要创建多个配置。例如，您可以创建一个仅显示您的 Amazon Web Services 账户中一组网关的错误信息的配置。您还可以创建一个配置，以显示有关要监控的无线设备的所有信息。

以下各节介绍了各种配置设置以及如何创建配置。

配置设置

在创建或更新网络分析器配置时，还可以自定义以下参数以筛选日志流信息。

- 帧信息

  此设置是跟踪消息的无线设备资源的帧信息。帧信息可用于调试网络服务器与终端设备之间的通信。该功能默认已启用。
- 日志级别

  您可以查看信息或错误日志，也可以关闭日志记录。
  - Info

    日志级别为 Info（信息）的日志更详细，其中包含错误日志流和信息日志流。信息日志可用于查看设备或网关状态的变化。

    Note

    收集更多详细的日志流可能会产生额外的成本。有关定价的更多信息，请参阅 Amazon IoT Core 定价。
  - 错误

    日志级别为 Error 的日志不那么详细，只显示错误信息。当应用程序出错（例如设备连接错误）时，可以使用这些日志。通过使用日志流中的信息，您可以识别机群中资源的错误并对其进行故障排除。

使用控制台创建配置

通过使用 Amazon IoT 控制台或 Amazon IoT 无线 API，您可以创建网络分析器配置和自定义可选参数。还可以创建多个配置，之后删除任何不再使用的配置。

创建网络分析器配置

1. 打开 Amazon IoT 控制台的网络分析器中心并选择 Create configuration（创建配置）。
2. 指定配置设置。
   - 名称、描述和标签

     指定一个唯一的 Configuration name（配置名称），其中仅包含字母、数字、连字符或下划线。使用可选的 Description（说明）字段以提供有关配置的信息，并使用 Tags（标签）字段以添加有关配置的元

数据的键值对。有关命名和描述资源的更多信息，请参阅 描述您的 foAmazon IoT Core r LoRa WAN 资源 (p. 1111)。

- 配置设置

  选择是否禁用帧信息，并使用 Select log levels（选择日志级别）以选择希望用来跟踪消息日志的日志级别。选择 Next（下一步）。

3. 向配置添加资源。您可以立即添加资源，也可以选择 Create（创建）并在以后添加您的资源。要以后添加资源，请选择 Create（创建）。

   在 Network Analyzer hub page（网络分析器中心页面）中，您将看到创建的配置及其设置。要查看新配置的详细信息，请选择配置名称。

删除网络分析器配置

您可以创建多个网络分析器配置，具体取决于要监视的资源以及希望针对这些资源接收的跟踪消息收发信息的级别。

从控制台中删除配置

1. 转至 Amazon IoT 控制台的网络分析器中心，然后选择要删除的配置。
2. 选择 Actions，然后选择 Delete。

使用 API 创建控制台

要使用 API 创建网络分析器配置，请使用 CreateNetworkAnalyzerConfiguration API 操作或 create-network-analyzer-configuration CLI 命令。

创建配置时，只需指定配置名称。还可以使用此 API 操作在创建配置时指定配置设置和添加资源。

> Note
>
> 使用 CreateNetworkAnalyzerConfiguration API 操作添加资源时，对于每个 API 请求，您最多只能指定 99 个无线资源。单个网络分析器配置最多可以组合 250 个无线设备和 250 个无线网关。要添加其他资源，请使用 Amazon IoT 控制台，如上一节所述。

- 创建配置

  创建配置时，必须指定名称。例如，以下命令通过仅提供名称和可选描述来创建配置。原定设置情况下，配置激活了帧信息，并使用 INFO 日志级别。

```
aws iotwireless create-network-analyzer-configuration \
    --configuration-name My_Network_Analyzer_Config \
    --description "My first network analyzer configuration"
```

  运行此命令将显示网络分析器配置的 ARN 和 ID。

```
{
    "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:NetworkAnalyzerConfiguration/12345678-a1b2-3c45-67d8-e90fa1b2c34d",
    "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
}
```

- 使用资源创建配置

  要自定义这些配置设置，请使用 trace-content 参数。要添加资源，请使用 WirelessDevices 和 WirelessGateways 参数指定要添加到配置中的网关和/或设备。例如，以下命令自定义配置设置并将无线资源（由其 WirelessGatewayID 和 WirelessDeviceID 指定）添加到配置中。

```
aws iotwireless create-network-analyzer-configuration \
    --configuration-name My_NetworkAnalyzer_Config \
    --trace-content WirelessDeviceFrameInfo=DISABLED,LogLevel="ERROR" \
    --wireless-gateways "12345678-a1b2-3c45-67d8-e90fa1b2c34d" "90123456-de1f-2b3b-4c5c-
bb1112223cd1"
    --wireless-devices "1ffd32c8-8130-4194-96df-622f072a315f"
```

下面的示例显示了运行此命令的输出：

```
{
    "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:NetworkAnalyzerConfiguration/12345678-a1b2-3c45-67d8-e90fa1b2c34d",
    "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
}
```

列出网络分析器配置

您可以创建多个网络分析器配置，具体取决于要监视的资源以及要针对这些资源接收的跟踪消息收发信息的详细程度。创建这些配置后，可以使用 ListNetworkAnalyzerConfigurations API 操作或 list-network-analyzer-configuration CLI 命令以获取这些配置的列表。

```
aws iotwireless list-network-analyzer-configurations
```

运行此命令会显示您的 Amazon Web Services 账户 中的所有网络分析器配置。也可以使用 `max-results` 参数来指定要显示多少个配置。下面的内容显示运行此命令的输出。

```
{
   "NetworkAnalyzerConfigurationList": [
      {
         "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:NetworkAnalyzerConfiguration/12345678-a1b2-3c45-67d8-e90fa1b2c34d",
         "Name": "My_Network_Analyzer_Config1"
      },
      {
         "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:NetworkAnalyzerConfiguration/90123456-a1a2-9a87-65b4-c12bf3c2d09a",
         "Name": "My_Network_Analyzer_Config2"
      }
   ]
}
```

删除网络分析器配置

您可以通过 DeleteNetworkAnalyzerConfiguration API 操作或 delete-network-analyzer-configuration CLI 命令删除不再使用的配置。

```
aws iotwireless delete-network-analyzer-configuration \
    --configuration-name My_NetworkAnalyzer_Config
```

运行此命令不会生成任何输出。要查看可用配置，可以使用 `ListNetworkAnalyzerConfigurations` API 操作。

后续步骤

现在您已创建网络分析器配置，可以向配置中添加资源或更新配置设置。有关更多信息，请参阅添加资源并更新网络分析器配置 (p. 1190)。

添加资源并更新网络分析器配置

在激活跟踪消息收发之前，将要监控的资源添加到网络分析器配置中。资源可以是 LoRaWAN 设备和 LoRaWAN 网关之一，也可以是两者。

先决条件

在添加资源之前：

1. 您必须已将要监控的网关和设备引导到 Amazon IoT Core for LoRaWAN。有关更多信息，请参阅将网关和设备连接到 Amazon IoT Core for LoRaWAN (p. 1110)。
2. 您必须已经创建了将为其添加资源的网络分析器配置。有关更多信息，请参阅创建网络分析器配置 (p. 1187)。

使用控制台添加资源并更新配置设置

通过使用Amazon IoT控制台或 Amazon IoT无线 API，您可以添加资源和自定义可选参数。除了资源之外，您还可以编辑配置设置并保存更新的配置。

向配置添加资源

1. 打开 Amazon IoT 控制台的网络分析器中心，然后选择要为其添加资源的配置。
2. 选择 Actions（操作），然后选择 Add resources（添加资源）。
3. 使用无线网关和无线设备标识符添加要监控的资源。您可以选择多个资源，最多可添加 250 个无线网关或无线设备。
4. 在添加所有资源后，请选择Add（添加）。

   您将在网络分析器中心页面中看到添加的网关和设备的数量。在激活跟踪消息收发会话之前，您可以继续添加和删除资源。激活会话后，要添加资源，您必须停用会话。

更新配置设置

1. 打开 Amazon IoT 控制台的网络分析器中心，然后选择要更新其设置的配置。
2. 选择 Actions，然后选择 Edit。
3. 选择是否禁用帧信息并使用选择日志级别选择希望用来跟踪消息日志的日志级别。选择 Save（保存）。

   您将看到在网络分析器配置的详细信息页面中指定的配置设置。

使用 API 添加资源并更新配置设置

要添加资源或更新配置设置，请使用 UpdateNetworkAnalyzerConfiguration API 或 update-network-analyzer-configuration CLI。

> **Note**
>
> 使用 `UpdateNetworkAnalyzerConfiguration` API 操作添加或更新资源时，对于每个 API 请求，您最多只能指定 99 个无线资源。单个网络分析器配置最多可以组合 250 个无线设备和 250 个无线网关。要添加其他资源，请使用 Amazon IoT 控制台，如上一节所述。

- 更新配置设置

  要更新配置设置，请使用 `TraceContent` 参数来指定日志级别以及是否启用帧信息。例如，以下命令通过禁用帧信息并将日志级别设置为 `ERROR` 来更新配置设置。

```
aws iotwireless update-network-analyzer-configuration \
    --configuration-name NetworkAnalyzerConfig_Default \
    --trace-content WirelessDeviceFrameInfo=DISABLED,LogLevel="ERROR"
```

- 添加资源

  要添加资源，请使用 `WirelessDevicesToAdd` 和 `WirelessGatewaysToAdd` 参数指定要添加到配置中的网关和/或设备。例如，以下命令更新配置设置并将无线资源（由其 `WirelessGatewayID` 和 `WirelessDeviceID` 指定）添加到配置中。

```
aws iotwireless update-network-analyzer-configuration \
    --configuration-name NetworkAnalyzerConfig_Default \
    --trace-content WirelessDeviceFrameInfo=DISABLED,LogLevel="ERROR" \
    --wireless-gateways-to-add "12345678-a1b2-3c45-67d8-e90fa1b2c34d" "90123456-
de1f-2b3b-4c5c-bb1112223cd1"
    --wireless-devices-to-add "1ffd32c8-8130-4194-96df-622f072a315f"
```

  要删除设备或网关，请使用 API 的 `WirelessDevicesToRemove` 和 `WirelessGatewaysToRemove` 参数。

获取有关配置的信息

运行 `UpdateNetworkAnalyzerConfiguration` API 不会生成任何输出。要查看您的配置设置以及已添加的网关或设备，请使用 GetNetworkAnalyzerConfiguration API 操作或 get-network-analyzer-configuration 命令。提供网络分析器配置的名称作为输入。

```
aws iotwireless get-network-analyzer-configuration \
    --configuration-name NetworkAnalyzerConfig_Default
```

运行此命令将生成以下输出。

```
{
    "TraceContent": {
        "WirelessDeviceFrameInfo": "DISABLED",
        "LogLevel": "ERROR"
    },
    "WirelessDevices": [],
    "WirelessGateways": [
        "a0dd70e5-8f15-41a5-89cf-310284e691a1",
        "41682155-de4f-4f8f-84bf-bb5557221fc8"
    ]
}
```

后续步骤

现在，您已经添加了资源并为配置指定了任何可选的配置设置，您可以使用 WebSocket 协议与适用于 LoRaWAN 的 Amazon IoT Core建立连接来使用网络分析器。然后，您可以激活跟踪消息并开始接收资源的跟踪消息。有关更多信息，请参阅 使用 WebSockets 串流网络分析器跟踪消息 (p. 1191)。

使用 WebSockets 串流网络分析器跟踪消息

当使用 WebSocket 协议时，您可以实时流式传输网络分析器跟踪消息。当您发送请求时，该服务会使用 JSON 结构响应。激活跟踪消息后，您可以使用消息日志获取有关资源的信息并排查错误。有关更多信息，请参阅WebSocket Numbers。

下面展示了如何使用 WebSockets 流式传输网络分析器跟踪消息。

主题

- 使用 WebSocket 库生成预签名请求 (p. 1192)
- 用于生成预签名 URL 的示例 Python 代码 (p. 1195)
- WebSocket 消息和状态码 (p. 1198)

使用 WebSocket 库生成预签名请求

下面显示如何生成预签名请求，这样，您可以使用 WebSocket 库向服务发送请求。

将 WebSocket 请求的策略添加到您的 IAM 角色

要使用 WebSocket 协议调用网络分析器，您需要将以下策略附加到发出请求的 Amazon Identity and Access Management(IAM) 角色。

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

创建预签名 URL

为 WebSocket 请求构造一个 URL，该请求中包含在应用程序和网络分析器 之间建立通信所需的信息。要验证请求的标识，WebSocket 流式处理使用 Amazon 签名版本 4 流程处理签名请求。有关签名版本4的更多信息，请参阅 Amazon Web Services 一般参考中的签名 AmazonAPI 请求。

要调用网络分析器，请使用 `StartNetworkAnalyzerStream`请求 URL。使用之前提到的 IAM 角色凭据对请求进行签名。URL的格式如下，出于可读性目的，添加了换行符。您必须在行 `&X-Amz-SignedHeaders=host` 之下添加配置名称。任何其他参数都应按字母顺序添加到此行之下。

以下示例演示了如何将此请求 URL 与配置名称 *`NaConfig`* 一起使用：

```
wss://api.iotwireless.<region>.amazonaws.com/start-network-analyzer-stream?configuration-
name=NaConfig
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Date=20220427T001057Z&X-Amz-SignedHeaders=host
    &X-Amz-Expires=300
    &X-Amz-Credential=credential_number/account/region/iotwireless/aws4_request
    &X-Amz-Signature=c123456789098765a012c3a45d6789dd01234af5678bba9bbc0dbc112a3334d
```

> **Note**
>
> 如果您的 URL 不包含配置名称，Amazon IoT Core for LoRaWAN 将包含网络分析器配置的原定设置名称 `NetworkAnalyzerConfig_Default`。

```
GET wss://api.iotwireless.<region>.amazonaws.com/start-network-analyzer-stream?X-Amz-
Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=Signature Version 4 credential scope
   &X-Amz-Date=date
   &X-Amz-Expires=time in seconds until expiration
   &X-Amz-Security-Token=security-token
   &X-Amz-Signature=Signature Version 4 signature
```

```
&X-Amz-SignedHeaders=host
```

对于签名版本 4 参数，请使用以下值：

- X-Amz-Algorithm – 您在签名过程中使用的算法。唯一有效值为 AWS4-HMAC-SHA256。
- X-Amz-Credential – 以斜杠（“/”）分隔的字符串，它通过将您的访问密钥 ID 和凭证范围组件串联起来而形成。凭证范围包括采用 YYYYMMDD 格式的日期、Amazon 区域、服务名称和终止字符串 (aws4_request)。
- X-Amz-Date – 创建签名的日期和时间。根据 Amazon Web Services 一般参考中签名版本 4 中处理日期的说明生成日期和时间。
- X-Amz-Expires – 凭证到期之前的时间长度（以秒为单位）。最大值为 300 秒（5 分钟）。
- X-Amz-Security-Token – 临时凭证的签名版本 4 令牌。如果您指定此参数，则将其包含在规范请求中。更多信息，请参阅 AmazonIdentity and Access Management 中的请求临时安全凭据。
- X-Amz-Signature – 您为请求生成的签名版本 4 签名。
- X-Amz-SignedHeaders – 在为请求创建签名时签名的标头。唯一有效值为 host。

构建请求 URL 并创建签名版本 4 签名

要构造请求的 URL 并创建签名版本 4 签名，请使用以下步骤。

> **Note**
>
> 此节中的示例采用伪代码。有关说明如何创建签名的示例 python 代码，请参阅用于生成预签名 URL 的示例 Python 代码 (p. 1195)。

任务 1：创建规范请求

创建一个字符串，其中包含来自标准格式的请求的信息。这可确保 Amazon在收到请求时，计算出的签名与您在 任务 3：计算签名 (p. 1195)中计算出的签名相同。有关更多信息，请参阅 Amazon Web Services 一般参考中的 创建签名版本 4 的规范请求。

1. 为您应用程序中的请求定义变量。

```
# HTTP verb
method = "GET"

# Service name
service = "iotwireless"

# Amazon Web Services ##
region = "Amazon Web Services ##"

# Service streaming endpoint
endpoint = "wss://api.iotwireless.region.amazonaws.com"

# Host
host = "api.iotwireless.<region>.amazonaws.com"

# Date and time of request
amz-date = YYYYMMDD'T'HHMMSS'Z'

# Date without time for credential scope
datestamp = YYYYMMDD
```

2. 创建规范 URI（统一资源标识符）。规范 URI 是域与查询字符串之间的 URI 部分。

```
canonical_uri = "/start-network-analyzer-stream"
```

3. 创建规范标头和签名标头。请注意规范标头尾部的 \n。

- 追加小写标头名称，后跟冒号。
- 追加该标头的值的逗号分隔列表。请勿对有多个值的标头进行值排序。
- 追加一个新行(\n)。

```
canonical_headers = "host:" + host + "\n"
signed_headers = "host"
```

4. 将算法与哈希算法匹配。您必须使用 SHA-256。

```
algorithm = "AWS4-HMAC-SHA256"
```

5. 创建凭证范围，该范围将派生密钥范围限定为发出请求的日期以及将请求发送到的区域和服务。

```
credential_scope = datestamp + "/" + region + "/" + service + "/" + "aws4_request"
```

6. 创建规范查询字符串。查询字符串值必须是 URL 编码，并且按名称排序。

- 按字符代码点以升序顺序对参数名称进行排序。具有重复名称的参数应按值进行排序。例如，以大写字母 F 开头的参数名称排在以小写字母 b 开头的参数名称之前。
- 请勿对 RFC 3986 定义的任何非预留字符进行 URI 编码，这些字符包括：A-Z、a-z、0-9、连字符 (-)、下划线 (_)、句点 (.) 和波形符 (~)。
- 使用 %XY 对所有其他字符进行百分比编码，其中“X”和“Y”为十六进制字符（0-9 和大写字母 A-F）。例如，空格字符必须编码为 %20（不像某些编码方案那样使用“+”），扩展 UTF-8 字符必须采用格式 %XY%ZA%BC。
- 对参数值中的任何等于 (=) 字符进行双重编码。

```
canonical_querystring  = "X-Amz-Algorithm=" + algorithm
canonical_querystring += "&X-Amz-Credential="+ URI-encode(access key + "/" +
 credential_scope)
canonical_querystring += "&X-Amz-Date=" + amz_date
canonical_querystring += "&X-Amz-Expires=300"
canonical_querystring += "&X-Amz-Security-Token=" + token
canonical_querystring += "&X-Amz-SignedHeaders=" + signed_headers
```

7. 创建负载的哈希。对于 GET 请求，负载为空字符串。

```
payload_hash = HashSHA256(("").Encode("utf-8")).HexDigest()
```

8. 组合所有元素以创建规范请求。

```
canonical_request = method + '\n'
   + canonical_uri + '\n'
   + canonical_querystring + '\n'
   + canonical_headers + '\n'
   + signed_headers + '\n'
   + payload_hash
```

任务 2：创建待签字符串

待签字符串包含您的请求的元信息。在计算请求签名时，您可以使用该字符串登录下一步。有关更多信息，请参阅 Amazon Web Services 一般参考中的创建签名版本 4 签名字符串。

```
string_to_sign=algorithm + "\n"
```

```
  + amz_date + "\n"
  + credential_scope + "\n"
  + HashSHA256(canonical_request.Encode("utf-8")).HexDigest()
```

任务 3：计算签名

您可以从您的 Amazon秘密访问密钥派生签名密钥。为了实现更高程度的保护，派生密钥特定于日期、服务和 Amazon区域。您可以使用派生密钥签名请求。有关更多信息，请参阅 Amazon Web Services 一般参考中的为 Amazon签名版本 4 计算签名。

该代码假定您实施了函数 GetSignatureKey来派生签名密钥。有关更多信息，请参阅 Amazon Web Services 一般参考中的如何为签名版本 4 派生签名密钥的示例。

函数 HMAC(key, data)表示以二进制格式返回结果的 HMAC-SHA256 函数。

```
#Create the signing key
signing_key = GetSignatureKey(secret_key, datestamp, region, service)

# Sign the string_to_sign using the signing key
signature = HMAC.new(signing_key, (string_to_sign).Encode("utf-8"), Sha256()).HexDigest
```

任务 4：将签名信息添加到请求中并创建请求 URL

在计算签名之后，将它添加到查询字符串。有关更多信息，请参阅 Amazon Web Services 一般参考中的向请求添加签名。

然后，您可以使用 WebSocket 库来请求预签名 URL。有关用于 Python 的 WebSocket 客户端示例，请参阅 websocket-client 1.4.1。

```
#Add the authentication information to the query string
canonical_querystring += "&X-Amz-Signature=" + signature

# Sign the string_to_sign using the signing key
request_url = endpoint + canonical_uri + "?" + canonical_querystring
```

后续步骤

现在，您可以在 WebSocket 库中使用请求 URL 向服务发出请求并观察消息。有关说明如何生成预签名 URL 的示例 Python 代码，请参阅用于生成预签名 URL 的示例 Python 代码 (p. 1195)。

用于生成预签名 URL 的示例 Python 代码

以下代码显示了一个使用 Python 作为编程语言来生成预签名 URL 的示例。

先决条件

要使用 Python 编程语言来生成请求，您必须：

- 已在您的计算机上安装 Python。您可以运行以下命令，也可以下载并运行 Python 安装程序。

```
sudo apt install python3
```

- Python 请求库。您可以运行以下命令或下载请求库，该库在示例脚本中用于发出 Web 请求。

```
pip install requests
```

- 一个访问密钥，由环境变量中名为 AWS_ACCESS_KEY_ID 和 AWS_SECRET_ACCESS_KEY 的访问密钥 ID 和秘密访问密钥构成。或者，您也可以在凭证文件中保存这些值，然后从这些文件中读取。

Note

作为最佳实践，我们建议您不要在代码中嵌入凭证。有关更多信息，请参阅《Amazon Account Management 参考指南》中的 Amazon 账户最佳实践。

```
$ export AWS_ACCESS_KEY_ID=My_Access_Key
$ export AWS_SECRET_ACCESS_KEY=My_Secret_Key

# Session token is required only if you use temporary access key starting with "ASIA"
$ export AWS_SESSION_TOKEN=My_Session_token
```

- 在您的账户中创建了网络分析器配置。要运行此脚本，请将网络分析器配置的名称指定为变量。此示例指的是名为 *My_Network_Analyzer_Config* 的配置。

  Note

  如果不指定配置名称，将使用默认配置 NetworkAnalyzerConfig_Default。

示例 Python 代码

Python 代码生成预签名 URL，WebSocket 库可使用该 URL 向服务发送请求。此函数创建一个规范请求，再创建用于计算签名的待签名字符串，然后将签名添加到 HTTP 请求以创建预签名 URL。之后，您可以使用 WebSocket 库来请求预签名 URL。

要运行脚本 generate_presigned_url.py，请运行以下命令（如果要从脚本所在的同一路径运行该脚本）。

```
python generate_presigned_url.py
```

下面显示的是 generate_presigned_url.py 脚本的内容。

generate_presigned_url.py 的内容

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

#  Version 4 signing example

"""
    Sample Python code to generate the pre-signed URL. You can
    change the parameters in this code to your own values, such
    as the variables that are required for the request URL, the
    network analyzer configuration name, and Region.
"""

# ************* Step 1. Import the required libraries and define the functions
# sign and getSignatureKey that will be used to derive a signing key.
# *************
import sys, os, base64, datetime, hashlib, hmac, urllib.pars
import requests     # pip install requests

def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()

def getSignatureKey(key, dateStamp, regionName, serviceName):
    kDate = sign(("AWS4" + key).encode("utf-8"), dateStamp)
    kRegion = sign(kDate, regionName)
    kService = sign(kRegion, serviceName)
    kSigning = sign(kService, "aws4_request")
```

```
    return kSigning

# ---------------------------------------------------------------------
# Step 2. Define the variables required for the request URL. Replace
# values for the variables, such as region, with your own values.
# ---------------------------------------------------------------------
method = "GET"
service = "iotwireless"
region = "us-east-1"

# Host and endpoint information.
host = "api.iotwireless." + region + ".amazonaws.com"
endpoint = "wss://" + host

# Create a date for headers and the credential string.
t = datetime.datetime.utcnow()
amz_date = t.strftime("%Y%m%dT%H%M%SZ")

# For date stamp, the date without time is used in credential scope.
datestamp = t.strftime("%Y%m%d")

# ---------------------------------------------------------------------------
# Step 3. Create the canonical URI and canonical headers for the request.
# ---------------------------------------------------------------------------
canonical_uri = "/start-network-analyzer-stream"
configuration_name = "My_Network_Analyzer_Config"

canonical_headers = "host:" + host + "\n"
signed_headers = "host"
algorithm = "AWS4-HMAC-SHA256"
credential_scope = datestamp + "/" + region + "/" + service + "/" + "aws4_request"

# ---------------------------------------------------------------------------
# Step 4. Read the  credentials that are required for the request
# from environment variables or configuration file.
# ---------------------------------------------------------------------------

# IMPORTANT: Best practice is NOT to embed credentials in code.

access_key = os.environ.get("AWS_ACCESS_KEY_ID")
secret_key = os.environ.get("AWS_SECRET_ACCESS_KEY")
token = os.environ.get("AWS_SESSION_TOKEN")

if access_key is None or secret_key is None:
    print("No access key is available.")
    sys.exit()

if access_key.startswith("ASIA") and token is None:
    print("Detected temporary credentials. You must specify a token.")
    sys.exit()

# ---------------------------------------------------------------------------
# Step 5. Create the canonical query string. Query string values must be
# URI-encoded and sorted by name. Query headers must in alphabetical order.
# ---------------------------------------------------------------------------
    canonical_querystring  = "X-Amz-Algorithm=" + algorithm

    canonical_querystring += "&X-Amz-Credential=" + \
    urllib.parse.quote(access_key + "/" + credential_scope, safe="-_.~")

    canonical_querystring += "&X-Amz-Date=" + amz_date
    canonical_querystring += "&X-Amz-Expires=300"

    if access_key.startswith("ASIA"):
        # percent encode the token and double encode "="
        canonical_querystring += "&X-Amz-Security-Token=" + \
```

```
        urllib.parse.quote(token, safe="-_.~").replace("=", "%253D")

    canonical_querystring += "&X-Amz-SignedHeaders=" + signed_headers
    canonical_querystring += "&configuration-name=" + configuration_name

# ************* TASK 6: CREATE A HASH OF THE PAYLOAD *************
# ----------------------------------------------------------------------
payload_hash = hashlib.sha256(("").encode("utf-8")).hexdigest()

# ************* TASK 7: COMBINE THE ELEMENTS, WHICH INCLUDES THE QUERY STRING, THE
# HEADERS, AND THE PAYLOAD HASH, TO FORM THE CANONICAL REQUEST. *************
# ----------------------------------------------------------------------
canonical_request = method + "\n" + canonical_uri + "\n" + canonical_querystring \
+ "\n" + canonical_headers + "\n" + signed_headers + "\n" + payload_hash

# ************* TASK 8: CREATE THE METADATA STRING TO STORE THE INFORMATION REQUIRED TO
# CALCULATE THE SIGNATURE IN THE FOLLOWING STEP. *************
# ----------------------------------------------------------------------
string_to_sign = algorithm + "\n" + amz_date + "\n" + \
credential_scope + "\n" + hashlib.sha256(canonical_request.encode("utf-8")).hexdigest()

# ************* TASK 9: CALCULATE THE SIGNATURE BY USING A SIGNING KEY THAT"S OBTAINED
# FROM YOUR SECRET KEY. *************
# ----------------------------------------------------------------------
# Create the signing key from your  secret key.
signing_key = getSignatureKey(secret_key, datestamp, region, service)

# Sign the string_to_sign using the signing key.
signature = hmac.new(signing_key, (string_to_sign).encode("utf-8"),
 hashlib.sha256).hexdigest()

# ************* TASK 10: CREATE THE REQUEST URL USING THE CALCULATED SIGNATURE AND BY
# COMBINING IT WITH THE CANONICAL URI AND THE QUERY STRING. *************
# ----------------------------------------------------------------------
canonical_querystring += "&X-Amz-Signature=" + signature

request_url = endpoint + canonical_uri + "?" + canonical_querystring

print("\n-----------PRESIGNED URL-----------")
print(request_url)
```

后续步骤

现在，您可以在 WebSocket 库中使用请求 URL 向服务发出请求并观察消息。

要安装与 Python一起使用的 WebSocket 库，请运行以下命令。有关如何将 WebSocket 客户端与 Python 结合使用的信息，请参阅带低级别 API 选项的适用于 Python 的 WebSocket 客户端。

```
pip install websocket-client
```

安装客户端并提出请求后，您将看到指示请求状态的消息和状态代码。有关更多信息，请参阅WebSocket 消息和状态码 (p. 1198)。

WebSocket 消息和状态码

创建预签名请求后，您可以将请求 URL 与 WebSocket 库或适合您的编程语言的库一起使用，向服务发出请求。有关如何生成此预签请求的更多信息，请参阅 使用 WebSocket 库生成预签名请求 (p. 1192)。

WebSocket 消息

可以使用 WebSocket 协议建立双向连接 消息可以从客户端传输到服务器，从服务器传输到客户端。但是，网络分析器仅支持从服务器发送到客户端的消息。从客户端收到的任何消息都是意外的，如果从客户端收到消息，服务器将自动关闭 WebSocket 连接。

收到请求并启动跟踪消息会话时，服务器将使用 JSON 结构（即有效负载）进行响应。有关有效负载以及如何从 Amazon Web Services Management Console激活跟踪消息的更多信息，请参阅 实时查看和监控网络分析器跟踪消息日志 (p. 1199)。

WebSocket 状态代码

下面显示了从服务器到客户端的通信的 WebSocket 状态码。WebSocket 状态码遵循 RFC 正常关闭连接标准。

下面显示了支持的状态代码：

- 1000

  此状态代码表示正常关闭，这意味着 WebSocket 连接已建立并且请求已完成。会话处于空闲状态时，可以观察到此状态，从而导致连接超时。
- 1002

  此状态代码表示端点由于协议错误而终止连接。
- 1003

  此状态代码表示错误状态，其中端点以无法接受的格式接收数据而终止连接。端点仅支持文本数据，如果它收到来自客户端的二进制消息或使用不受支持的格式的消息，可能会显示此状态代码。
- 1008

  此状态代码表示错误状态，端点因收到违反其策略的消息而终止连接。此状态为通用状态，当其他状态代码（例如 1003 或 1009）不适用时显示。如果需要隐藏策略或授权失败（例如签名过期），您还会看到此状态显示。
- 1011

  此状态代码表示错误状态，其中服务器因遇到意外情况或内部错误而导致无法完成请求而终止连接。

后续步骤

现在您已经学会了如何生成预签名请求以及如何使用 WebSocket 连接观察来自服务器的消息，现在您可以激活跟踪消息并开始接收无线网关和无线设备资源的消息日志。有关更多信息，请参阅 实时查看和监控网络分析器跟踪消息日志 (p. 1199)。

实时查看和监控网络分析器跟踪消息日志

如果已将资源添加到网络分析器配置中，可以激活跟踪消息开始接收资源的跟踪消息。您可以使用 Amazon Web Services Management Console，Amazon IoT 无线 API 或 Amazon CLI。

先决条件

在使用网络分析器激活跟踪消息传送之前，您必须：

- 将希望监控的资源添加到原定设置网络分析器配置中。有关更多信息，请参阅 添加资源并更新网络分析器配置 (p. 1190)。
- 通过使用 `StartNetworkAnalyzerStream`请求 URL 生成预签请求。将使用发出此请求的 Amazon Identity and Access Management角色的凭据对请求进行签名。有关更多信息，请参阅 创建预签名 URL (p. 1192)。

使用控制台激活跟踪消息

要激活跟踪消息

1. 打开 Amazon IoT控制台的网络分析器中心然后选择网络分析器配置，Network Analyszerconfig_DEFAULT。
2. 在网络分析仪配置的详细信息页面中，选择Activate trace messaging（激活跟踪消息），然后选择Activate（激活）。

   您将开始接收跟踪消息，其中最新的跟踪消息首先出现在控制台中。

   > Note
   >
   > 消息传递会话开始后，在您停用会话或退出跟踪会话之前，接收跟踪消息可能会产生额外费用。有关定价的更多信息，请参阅 Amazon IoT Core定价。

查看和监控跟踪消息

激活跟踪消息后，将建立 WebSocket 连接，并且跟踪消息开始实时出现，首先是最新消息。您可以自定义首选项以指定要在每个页面中显示的跟踪消息的数量，并仅显示每条消息的相关字段。例如，您可以自定义跟踪消息日志以仅显示具有日志级别设置为 `ERROR`，以便您可以快速识别和调试网关的错误。跟踪消息包含以下信息。

- 消息编号：显示首先收到的最后一条消息的唯一编号。
- 资源 ID：资源的无线网关或无线设备 ID。
- 时间戳：收到消息的时间。
- 消息 ID：标识符 Amazon IoT Core对于 LoRaWAN 分配给每条收到的消息。
- FPort：使用 WebSocket 连接与设备通信的频率端口。
- DevEui：无线设备的扩展唯一标识符 (EUI)。
- 资源：受监控的资源是无线设备还是无线网关。
- 事件：无线设备的日志消息的事件，可以是 Join、Rejoin、Uplink_Data、Downlink_Data 或 Registration。
- 日志级别: 有关 `INFO`或 `ERROR`设备的日志流的信息。

网络分析器 JSON 日志消息

您还可以一次选择一条跟踪消息来查看该消息的 JSON 有效负载。根据在跟踪消息日志中选择的消息，您将在 JSON 负载中看到指示包含两部分的信息：CustomerLog 和 LoRaFrame。

CustomerLog

JSON 的客户日志部分显示接收消息的资源的类型和标识符、日志级别和消息内容。以下示例显示了 CustomerLog 日志消息。您可以使用 JSON 中的 `message`字段来获取有关错误以及如何解决错误的更多信息。

LoRaFrame

JSON 的 lorRaFrame 部分有消息 ID 并包含有关设备的物理负载和无线元数据的信息。

以下示例显示了追踪消息的结构。

> Note
>
> 如果您的设备发送不带 `Fport` 值的上行链路消息，则 LoRaWAN 网络分析器的 Amazon IoT Core 将显示收到的跟踪消息中的 `fPort` 值 225。

```
export type TraceMessage = {
  ResourceId: string;
  Timestamp: string;
  LoRaFrame:
  {
    MessageId: string;
    PhysicalPayload: any;
    WirelessMetadata:
    {
      fPort: number;
      dataRate: number;
      devEui: string;
      frequency: number,
      timestamp: string;
    },
  }
  CustomerLog:
  {
    resource: string;
    wirelessDeviceId: string;
    wirelessDeviceType: string;
    event: string;
    logLevel: string;
    messageId: string;
    message: string;
  },
};
```

回顾和后续步骤

在本节中，您查看了跟踪消息，并了解了如何使用这些信息调试错误。查看所有消息后，您可以：

- 停用跟踪消息

  为避免产生任何额外费用，您可以停用跟踪消息传送会话。取消激活会话会断开 WebSocket 连接的连接，因此您将不会收到任何其他跟踪消息。您仍然可以继续在控制台中查看现有消息。

- 编辑配置的帧信息

  您可以编辑网络分析器配置，然后选择是否停用帧信息并选择消息的日志级别。在更新配置之前，请考虑停用跟踪消息传送会话。要进行这些编辑，请打开 Amazon IoT控制台中的网络分析器详细信息页面，然后选择 Edit（编辑）。然后，您可以使用新的配置设置更新配置，并激活跟踪消息查看更新的消息。

- 向配置添加资源

  您还可以向网络分析器配置添加更多资源并实时监控。您最多可以合计 250 个无线网关和无线设备资源。要添加资源，请在 Amazon IoT控制台的网络分析器详细信息页面上，选择 Resources（资源）选项卡，然后选择 Add resources（添加资源）。然后，您可以使用新资源更新配置并激活跟踪消息查看其他资源的更新消息。

有关通过编辑配置设置和添加资源来更新网络分析器配置的详细信息，请参阅 添加资源并更新网络分析器配置 (p. 1190)。

适用于 LoRaWAN 的 Amazon IoT Core 数据安全性

两种方法保护来自您的适用于 LoRaWAN 的 Amazon IoT Core 设备的数据安全：

- 无线设备用于与网关通信的安全性实践。

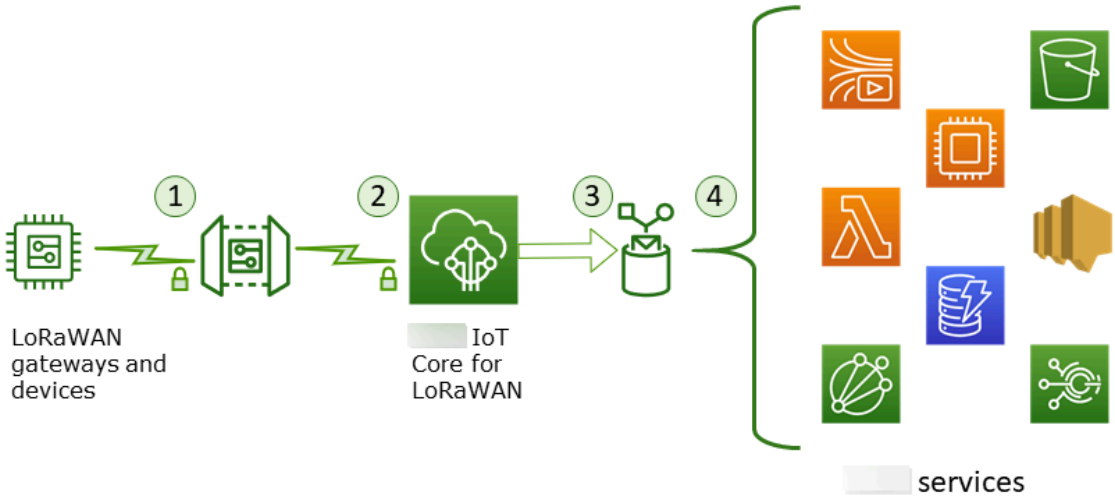
LoRaWAN 设备遵循 LoRAWAN™ 安全性：Gemalto、Actility 和 Semtech 为 LoRa Alliance™ 编制的白皮书中所述的安全实践来与网关进行通信。

- Amazon IoT Core 用于将网关连接到适用于 LoRaWAN 的 Amazon IoT Core 并将数据发送给其他 Amazon 服务的安全性实践。

  Amazon IoT Core 安全性实践已在 Amazon IoT Core 中的数据保护 (p. 345) 中说明。

如何在整个系统中保护数据

此图表标识了 LoRaWAN 系统中的关键元素，该元素用于连接到适用于 LoRaWAN 的 Amazon IoT Core 以确定数据在整个过程中是如何受保护的。



1. LoRaWAN 无线设备在传输二进制消息之前会使用 AES128 CTR 模式对其进行加密。
2. 网关与适用于 LoRaWAN 的 Amazon IoT Core 的连接通过 Amazon IoT 中的传输安全 (p. 346) 中所述的 TLS 予以保护。适用于 LoRaWAN 的 Amazon IoT Core 对二进制消息进行解密，并将解密的二进制消息负载编码为 base64 字符串。
3. 生成的 base64 编码消息将作为消息负载发送到分配给设备的目标中所述的 Amazon IoT 规则中。Amazon 内的数据使用 Amazon 拥有的密钥进行加密。
4. Amazon IoT 规则将消息数据定向到规则配置中描述的服务。Amazon 内的数据使用 Amazon 拥有的密钥进行加密。

LoRaWAN 设备和网关传输安全性

LoRaWAN 设备和适用于 LoRaWAN 的 Amazon IoT Core 存储预共享的根密钥。会话密钥由 LoRaWAN 设备和适用于 LoRaWAN 的 Amazon IoT Core 根据这些协议衍生得出。对称会话密钥用于标准 AES-128 CTR 模式下的加密和解密。还使用 4 字节的消息完整性代码 (MIC) 来检查遵循标准 AES-128 CMAC 算法的数据完整性。可以使用加入/重新加入进程更新会话密钥。

LoRa 网关的安全实践在 LoRaWAN 规范中有所说明。LoRa 网关使用 `Basics Station` 通过 Web 套接字连接到 Amazon IoT Core for LoRaWAN。Amazon IoT Core for LoRaWAN 仅支持 `Basics Station` 版本 2.0.4 及更高版本。

在建立 Web 套接字连接之前，适用于 LoRaWAN 的 Amazon IoT Core 使用 TLS 服务器和客户端身份验证模式 (p. 346)对网关进行身份验证。适用于 LoRaWAN 的 Amazon IoT Core 同时也维护 Configuration and Update Server (CUPS)，用于配置和更新用于 TLS 身份验证的证书和密钥。

Amazon IoT Core适用于 Amazon Side

Amazon IoT Corefor Amazon Sidewalk 提供云服务，您可以使用这些服务将Amazon Web Service Sidewalk 终端设备连接到Amazon Web Services 云并使用其他设备。

Amazon Sidewalk 是一个安全的共享网络，它使您社区中的设备能够连接并保持连接。Amazon Sidewalk 在人行道终端设备和人行道网关之间以及人行道网关和人行道云之间传输数据。

如何Amazon IoT Core用于亚马逊人行道？

您可以使用控制台或Amazon IoT Wireless API 操作Amazon IoT将 Sidewalk 终端设备搭载到上。设备接入后，其消息将发送至Amazon IoT Core。然后，您可以开始在Amazon云上开发您的业务应用程序，将使用来自您的 Amazon Sidewalk 终端设备中的数据。

使用控制台

要登录 Sidewalk 终端设备，请登录Amazon Web Services Management Console并导航到Amazon IoT主机上的 “设备” 页面。设备接入后，您可以在 IoT 控制台的此页面上查看和管理它们。

使用 API 或 CLI

您可以使用 Amazon IoTWireless API 操作搭载 Sidewalk 和 LoRa WAN 设备。构建 Amazon IoT Core 的 Amazon IoT Wireless API 由 Amazon SDK 提供支持。有关更多信息，请参阅 Amazon SDK 和工具包。

您可以使用运行Amazon CLI用于搭载和管理 Sidewalk 终端设备的命令。有关更多信息，请参阅 Amazon IoT Wireless CLI 参考。

Amazon IoT Core适用于亚马逊人行道区域和终端节点

亚马逊人行道仅在`us-east-1`Amazon Web Services 区域. Amazon IoT Core适用于 Amazon Sidewalk 的为该区域的控制面板和数据层面 API 终端节点提供支持。数据层面 API 终端节点特定于您的Amazon Web Services 账户。有关更多信息，请参阅《Amazon一般参考》中的Amazon IoT无线服务端点。

Amazon IoT Core适用于 Amazon Sidewalk 的具有适用于在设备和之间传输的Amazon Web Services 云设备数据的配额和Amazon IoT无线 API 操作的最大 TPS。有关更多信息，请参阅《Amazon一般参考》中的Amazon IoT无线配额。

Amazon IoT Core获取亚马逊 Sidewalk 定价

注册后Amazon，您可以通过Amazon免费套餐开始免费使用Amazon IoT Core适用于 Amazon Sidewalk 的。

有关一般产品概述和定价的更多信息，请参阅 Amazon IoT Core定价。

亚马逊人行道有什么Amazon IoT Core用？

借Amazon IoT Core助 For Amazon Sidewalk，您可以登录 Amazon Sidewalk 终端设备，Amazon IoT并对其进行管理和监控。它还管理向其他人发送设备数据的目的Amazon Web Service地。

亚马逊人行道的特点Amazon IoT Core

使用亚马逊Amazon IoT Core Sidewalk，您可以：

- 登录 Sidewalk 终端设备以Amazon IoT使用Amazon IoT控制台、执行Amazon IoT Core Amazon Sidewalk API 操作或Amazon CLI命令。
- 利用提供的功能Amazon Web Services 云。
- 创建一个使用Amazon IoT规则处理传入的负载消息并与其他人交互的目的地Amazon Web Services。
- 启用事件通知以接收有关事件的消息，例如您的 Sidewalk 终端设备何时配置或注册，或者下行链路消息是否已成功传送到您的设备。
- 实时记录和监控您的 Sidewalk 终端设备，获得有用的见解，并识别和解决错误。
- 将你的 Sidewalk 终端设备与某Amazon IoT件事物关联起来，这可以帮助你在云端存储设备的示意图。中的事物可以Amazon IoT让您更轻松地搜索和管理您的功能，以及访问您的Amazon IoT Core功能。

主题

- 什么是 Amazon Sidewalk？ (p. 1204)
- 亚马逊Amazon IoT Core Sidewalk 的工作原理 (p. 1205)

什么是 Amazon Sidewalk？

Amazon Sidewalk 是一个安全的社区网络，它使用亚马逊 Sidewalk Bridges（例如兼容的亚马逊 Echo 和 Ring 设备）为IoT 设备提供云连接。Amazon Sidewalk 使用 Bluetooth LE 进行短距离通信，使用 900MHz 频率的 FSK 无线电协议覆盖更远的距离，在家中及其他地方实现低带宽 LoRa 和远程连接。

Amazon Sidewalk 启用后，这个网络可以支持您社群中的其它 Sidewalk 终端设备，并可用于应用程序。Amazon Sidewalk 可帮助您的设备连接并保持连接。

Amazon Sidewalk 功能

以下是 Amazon Sidewalk 的功能。

- Amazon Sidewalk 使用包括 Ring 和精选 Echo 设备的 Sidewalk 网关创建低带宽网络。使用网关，您可以共享部分互联网带宽，然后将其用于将终端设备连接到网络。
- Amazon Sidewalk 提供了一种具有多层加密和安全性的安全联网机制。
- Amazon Sidewalk 提供了一种启用或禁用参与 Sidewalk 的简单机制。

Amazon Sidewal

以下是 Amazon Sidewalk 的一些关键概念。

人行道网关

人行道网关或 Amazon Sidewalk bridges 在您的 Sidewalk 终端设备和云端之间路由数据。网关是亚马逊设备，例如 Echo 设备或 Ring Floodlight Cam，它们支持 Subg-CSS（异步、LDR）、subg-FSK（同步、HDR）或蓝牙 LE 进行人行道通信。Sidewalk网关与Sidewalk社区共享您的部分互联网带宽，以提供与一组支持人行道的设备的连接。

人行道终端设备

人行道终端设备通过连接到人行道网关在亚马逊人行道上漫游。终端设备是低带宽、低功耗的智能产品，例如支持人行道的灯或门锁。

Note

某些 Sidewalk 网关也可以用作终端设备。

人行道网络服务器

由亚马逊运营的 Sidewalk Network Server 验证传入的数据包并将上行链路和下行链路消息路由到所需的目的地，同时保持 Sidewalk 网络的时间同步。

了解有关更多信息更多信息更多信息更多信息更多信息更多信息更多信息更多信息更多信息更多信息更多信息更多信息

有关 Amazon Sidewalk 的更多信息，请参阅以下网页：

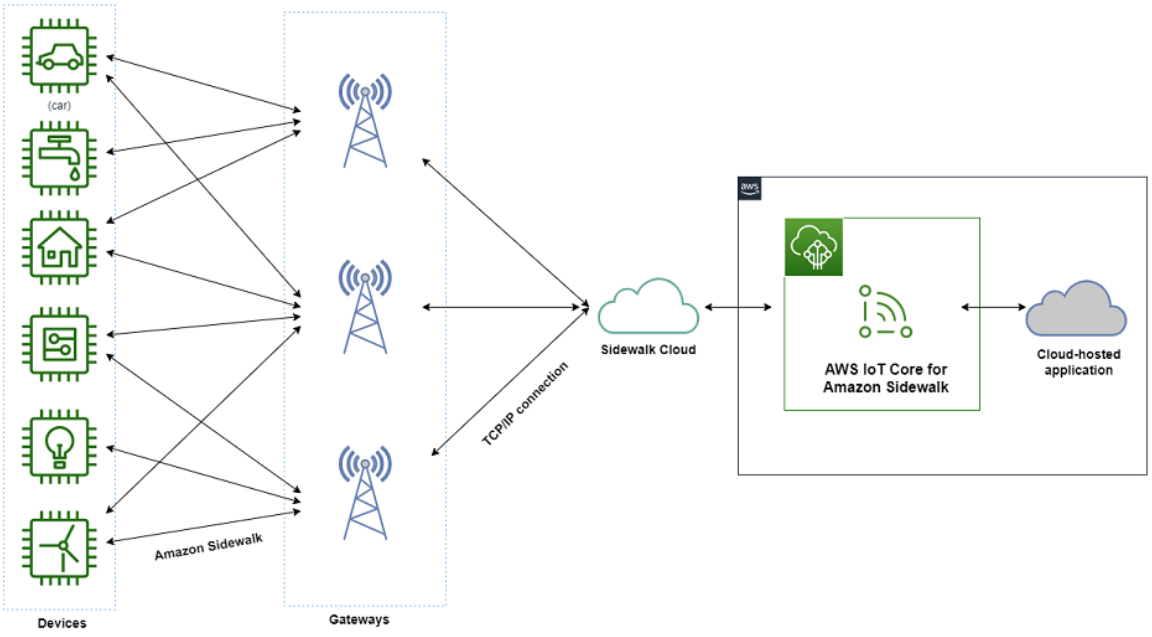
- 亚马逊人行道
- 亚马逊人行道文档
- Amazon IoT Core适用于亚马逊人行道

亚马逊Amazon IoT Core Sidewalk 的工作原理

借Amazon IoT Core助 For Amazon Sidewalk，您可以登录 Amazon Sidewalk 终端设备，Amazon IoT并对其进行管理和监控。它还管理将设备数据发送到其他Amazon Web Service目的地

Amazon IoT Corefor Amazon Sidewalk 提供云服务，您可以使用这些服务将Amazon Web Service Sidewalk 终端设备连接到Amazon Web Services 云并使用其他设备。您还可以使用Amazon IoT Core Amazon Sidewalk 管理您的 Sidewalk 设备，并在其上监控和构建应用程序。

人行道终端设备Amazon IoT Core通过人行道网关与之通信。 Amazon IoT Corefor Amazon Sidewalk 管理管理和与 Sidewalk 终端设备和网关通信Amazon IoT Core所需的服务和设备策略。它还管理向其他人发送设备数据的目的Amazon Web Service地。



开始使用亚马逊Amazon IoT Core Sidewalk

您可以使用Amazon IoT控制台、for Amazon Sidewalk API 或 Sidewalk 终端设备Amazon CLI来创建和加载 Sidewalk 终端设备并将它们连接到 Sidewalk 网络。Amazon IoT Core有关 Amazon Sidewalk 入门的信息 Amazon IoT，请参阅以下主题。

- 亚马逊 SidAmazon IoT Core ewalk 入门 (p. 1206)

  本主题介绍了入门 Sidewalk 终端设备的先决条件，说明了使用传感器监控应用程序的工作流程，并概述了如何使用Amazon CLI命令加载设备。
- 将 Sidewalk 设备连接到亚马逊Amazon IoT Core Sidewalk (p. 1212)

  本节介绍入门工作流程介绍中的不同步骤，并介绍如何使用控制台引导终端设备和 API 操作。您还将连接您的设备并查看您的设备与 Amazon Sidewalk 之间交换Amazon IoT Core的消息。
- 用Amazon IoT Core于 Amazon Sidewalk 的批量配置设备 (p. 1225)

  本节提供了有关使用Amazon IoT Core亚马逊 Sidewalk 批量配置 Sidewalk 终端设备的详细 step-by-step 教程。您将学习批量配置工作流程，以及如何加载大量 Sidewalk 设备。

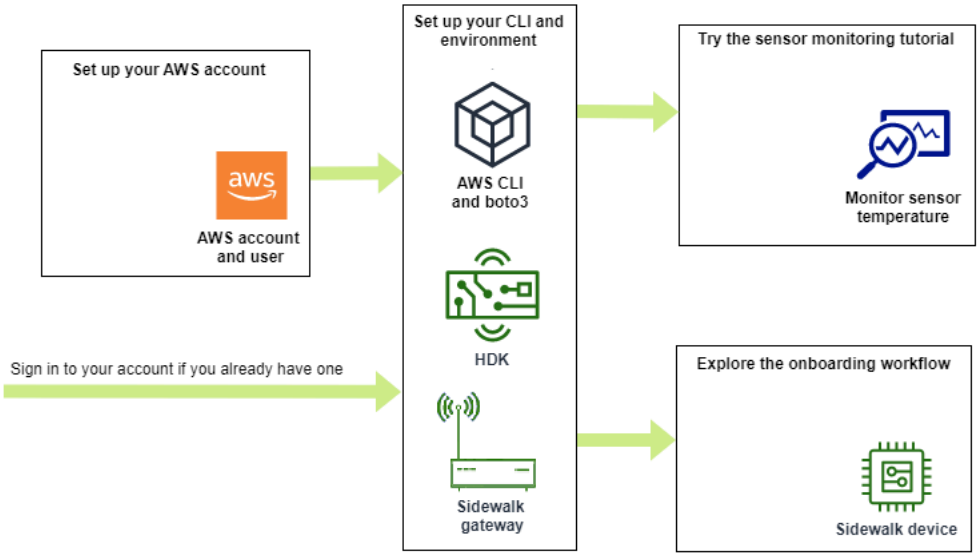
了解有关 Amazon SidewalkAmazon IoT Core 的更多信息

有关 AmazonAmazon IoT Core Sidewalk 的更多信息，请参阅以下网页：

- 亚马逊人行道
- 亚马逊人行道文档
- Amazon IoT Core适用于亚马逊人行道

亚马逊 SidAmazon IoT Core ewalk 入门

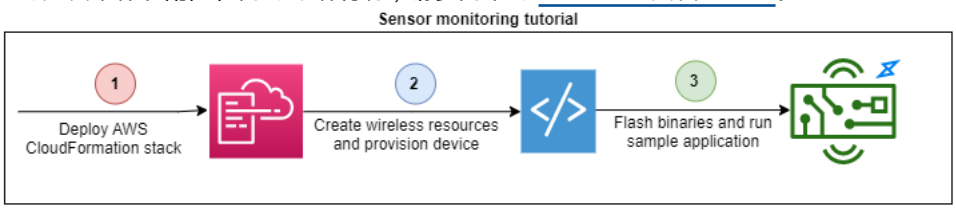
本部分向您展示如何开始将 Sidewalk 终端设备连接到 AmazonAmazon IoT Core Sidewalk。它解释了如何将终端设备连接到 Amazon Sidewalk 并在它们之间传递消息。您还将了解 Sidewalk 示例应用程序，并概述如何使用 Amazon SidewalAmazon IoT Core k 执行传感器监控。示例应用程序为您提供了一个仪表板，用于查看和监视传感器温度的变化。



试试传感器监控教程

本节概述了上面的 Amazon Sidewalk 示例应用程序 GitHub，该应用程序向您展示了如何监控传感器的温度。在本教程中，您将使用脚本以编程方式创建所需的无线资源，配置终端设备并刷新二进制文件，然后将终端设备连接到应用程序。使用Amazon CLI和 Python 命令的脚本创建Amazon CloudFormation堆栈和无线资源，然后刷新二进制文件并将应用程序部署到您的硬件开发套件 (HDK)。

下图显示了运行示例应用程序并将您的 Sidewalk 终端设备连接到该应用程序时所涉及的步骤。有关包括本教程的先决条件和配置在内的详细说明，请参阅中的 README 文档GitHub。



本节中执行以下内容：

主题

- 设置为Amazon IoT Core用于亚马逊 Sidewalk (p. 1207)
- 描述您的Sidewalk 资源 (p. 1208)
- 入门使用 Sidewalk 设备简介 (p. 1210)

设置为Amazon IoT Core用于亚马逊 Sidewalk

在连接 Sidewalk 终端设备之前，您必须设置Amazon Web Services 账户并配置Amazon CLI。

> Important
>
> 要执行配置和注册 Sidewalk 终端设备的整个入门工作流程，还必须设置 Sidewalk 网关和 HDK。有关说明，请参阅 Amazon Sidewal k 文档中的设置硬件开发套件 (HDK) 和设置 Sidewal k 网关。

主题

- 安装 Python 和 Python3-pip (p. 1207)
- 设置账户，以及Amazon CLI (p. 1208)

安装 Python 和 Python3-pip

要使用账户Amazon CLI，您必须使用 Python 3.6 或更高版本，您必须使用 Python 3.6 或更高版本。如果您想使用Amazon IoT控制台加载终端设备，可以跳过本节继续设置Amazon Web Services 账户。要检查你是否已经安装了 Python 和 Python3-pip，请运行以下命令。如果运行这些命令返回版本，则表示 Python 和 Python3-Pip 已正确安装。

```
python3 -V
pip3 --version
```

如果此命令返回错误，则可能是因为未安装 Python，或者您的操作系统将 Python v3.x 可执行文件调用为 Python3。在这种情况下，请在运行命令`python3`时将的所有`python`实例替换为。如果仍然出现错误，请下载并运行 Python 安装程序，或者根据您的操作系统安装 Python，如下所述。

Windows

在你的 Windows 计算机上，从 Pyt hon 网站下载 Python，然后运行安装程序在你的计算机上安装 Python。

Linux

在你的 Ubuntu 计算机上，运行以下sudo命令来安装 Python。

```
sudo apt install python3
sudo apt install python3-pip
```

macOS

在 Mac 计算机上，使用 Homebrew 安装 Python。Homebrew 还会安装 pip，然后它指向已安装的 Python3 版本。

```
$ brew install python
```

设置账户，以及Amazon CLI

以下步骤向您展示如何配置Amazon Web Services 账户、Amazon CLI和 boto3（Amazon适用于 Python 的 SDK）。

1. 设置您的Amazon Web Services 账户

   要使用Amazon IoT Core Amazon Sidewalk，请注册Amazon Web Services 账户并创建管理用户。有关说明，请参阅 设置您的Amazon Web Services 账户 (p. 16)。
2. 安装和配置 Amazon CLI

   您可以使用Amazon CLI以编程方式将您的 Sidewalk 终端设备加载到Amazon IoT Core Amazon Sidewalk。如果要使用账户，Amazon IoT可以跳过此部分。打开主Amazon IoT Core机，然后继续进入下一部分，开始将您的设备连接到 Amazon Sidewalk。Amazon IoT Core有关配置的说明Amazon CLI，请参阅安装和配置Amazon CLI。
3. 安装 boto3（Amazon适用于 Python 的开发工具包）

   以下命令向您展示如何安装 boto3（Amazon适用于 Python 的 SDK）和Amazon CLI. 你还将安装 botocore，这是运行 boto3 所必需的。有关详细说明，请参阅《Boto3 文档指南》中的 “安装 Boto3”。

   Note

   awscli版本1.26.6需要 3.10 或更高版本的 PyYAML 版本，但不高于 5.5。

   ```
   python3 -m pip install botocore-version-py3-none-any.whl
   python3 -m pip install boto3-version-py3-none-any.whl
   ```
4. 配置凭证和默认区域

   在和~/.aws/config文件中配置您的证书~/.aws/credentials和默认区域。boto3 库使用这些证书来识别Amazon Web Services 账户和授权 API 调用。有关配置说明，请参阅：

   - Boto3 文档指南中的@@ 配置
   - Amazon CLI文档指南中的配置和凭证文件设置

描述您的Sidewalk 资源

在您开始使用资源并创建资源之前，建议您考虑资源的命名约定。您的无线资源可以包括 Sidewalk 终端设备、设备配置文件或目的地。 Amazon IoT Corefor Amazon Sidewalk 会为您创建的资源分配唯一标识符。但是，您可以为它们提供更具描述性的名称、添加描述或添加可选标签以帮助识别和管理它们。

以下各节介绍了各种Sidewalk 资源及其长度限制。

主题

- 资源名称和描述 (p. 1209)
- 资源标签 (p. 1209)

资源名称和描述

Sidewalk 资源的名称最多可包括 256 个字符。对于Sidewalk 终端设备和配置文件，资源名称是可选的，资源名称是可选的，可在创建资源资源的资源后更改资源名称，资源名称是可选的，可在创建资源 对于目的地，请提供您的Amazon Web Services 账户和独有的名称Amazon Web Services 区域。目标名称的名称在创建后将无法更改。我们建议您使用简短而有意义的名称来帮助识别资源。例如，如果您的 Sidewalk 终端设备用于对象定位，则可以指定一个名称，例如*Sidewalk_Object_Locator_Device*。

在Amazon IoT控制台中，名称字段出现在资源中心的资源列表中。为资源选择名称时，请考虑您希望如何在控制台中显示资源的名称时，请考虑您希望如何在控制台中显示资源的名称时，请考虑您希望如何在控制台中显示 由于控制台中的空间有限，可能只能看到前 15-30 个字符。

人行道名称字段约束

资源名称	名称字段约束
目标位置	名称是唯一的 ID，无法更改 ID，因此无法更改 ID。
Sidewalk 端点设备	名称是一个可选的描述符，资源的名称是一个可选的描述符，资源的名称是一个可选的描述符，资源的名称是一个可选的
设备配置文件	名称是一个可选的描述符，资源的名称是一个可选的描述符，资源的名称是一个可选的描述符，资源的名称是一个可选的

目标、终端设备和配置文件还支持描述字段，该字段最多可包含 2048 个字符。虽然描述字段可以保存大量信息，但它只显示在Amazon IoT控制台中资源的详细信息页面中，因此在多个资源的上下文中扫描可能并不方便。为了帮助您识别和管理您的资源，我们建议您提供简短而有意义的描述。例如，如果您的目的地将传感器数据重新发布到 IoT 主题，则可以指定描述，例如*Sidewalk destination to republish sensor data*。

资源标签

标签是可作为元数据标识和组织Amazon资源的词或短语。标签可帮助您分类资源。您可以选择标签键及其相应的值。

目标和配置文件最多可以附加 50 个Amazon标签。Sidewalk 设备不支持附加标签。例如，您可以为一组配置文件定义一组标签，这些配置文件与一组特定 Sidewalk 终端设备相关联的配置文件，您可以给予 Sidewalk 终端设备相关的一组配置文件定义一组标签 为了更轻松地管理资源，我们建议您创建一组一致的标签密钥，以满足您的各种资源的需要，我们建议您创建一组一致的标签密钥，以更轻松地管理资源，我们建议您创建一组一致的标签密钥 下表显示您的Sidewalk 资源支持的标签，您可以给予 Sidewalk 资源支持的

Sidewalk 标签限制

资源名称	标签限制
目标位置	每个资源最多可以添加 50 个Amazon标签。
Sidewalk 设备	此资源不支持标签。
设备配置文件	每个资源最多可以添加 50 个Amazon标签。

您可以将标签键视为信息类别，将标签值视为该类别中的特定值。例如，您的标记值可能为，对于一些资源的这个标签，您可以给予，对于其它资源的*color*这个标签，您可以给予值，对于其它资源的这个标签，您可以给予值，*blue*对于其它资源的*red*这个标签，您的标记值可能为 有了这个，您可以使用中的标签编辑器来查找 color 标签值为的资源Amazon Web Services Management Console来查找 color 标签值为的资源，您可以给予*blue*。

有关标记和标记策略的更多信息，请参阅 Tag Editor。

入门使用 Sidewalk 设备简介

本部分介绍如何将 Sidewalk 终端设备登录 AmazonAmazon IoT Core Sidewalk。要加载您的设备，请先添加您的 Sidewalk 设备，然后预置和注册您的设备，然后将硬件连接到云应用程序。在运行本教程之前，请查看并完成设置为Amazon IoT Core用于亚马逊 Sidewalk (p. 1207)。

以下步骤向您展示如何登录 Sidewalk 终端设备并将其连接到 Amazon Sidewalk。Amazon IoT Core如果您想使用接入设备Amazon CLI，可以参考本节提供的示例命令。有关使用Amazon IoT控制台载入设备的信息，请参阅将 Sidewalk 设备连接到亚马逊Amazon IoT Core Sidewalk (p. 1212)。

> Important
>
> 要执行整个入门工作流程，您还需要预置和注册终端设备，并连接硬件开发套件 (HDK)。有关更多信息，请参阅 Amazon Sidewalk 文档中的配置和注册您的终端设备。

主题

- 第 1 步：将你的 Sidewalk 设备添加到Amazon IoT Core Amazon Sidewalk (p. 1210)
- 步骤 2：为您的 Sidewalk 终端设备创建目标 (p. 1211)
- 步骤 3：配置和注册终端设备 (p. 1211)
- 步骤 4：Connect Sidewalk 终端设备并交换消息 (p. 1212)

第 1 步：将你的 Sidewalk 设备添加到Amazon IoT Core Amazon Sidewalk

以下是将您的 Sidewalk 终端设备添加到 Amazon Sidewalk 时要执行Amazon IoT Core的步骤的概述。存储您获得的有关设备配置文件和您创建的无线设备的信息。您将使用这些信息来配置和注册终端设备。有关这些步骤的更多信息，请参阅将您的设备添加到亚马逊Amazon IoT Core Sidewalk (p. 1213)。

1. 创建设备配置文件

   创建包含您的 Sidewalk 设备共享配置的设备配置文件。创建配置文件时，将配置文件指定*name*为字母数字字符串。要创建配置文件，请转到Amazon IoT控制台配置文件中心的 Sidewalk 选项卡并选择创建配置文件，或者使用 `CreateDeviceProfile`API 操作或 `create-device-profile`CLI 命令，如本示例所示。

   ```
   // Add your device profile using a name and the sidewalk object.
   aws iotwireless create-device-profile --name sidewalk_profile --sidewalk {}
   ```

2. 创建你的 Sidewalk 终端设备

   使用 Amazon Sidewalk 创建你Amazon IoT Core的 Sidewalk 终端设备。指定目标名称和从上一步获取的设备配置文件的 ID。要添加设备，请转到Amazon IoT控制台中设备中心的 Sidewalk 选项卡并选择配置设备，或者使用 `CreateWirelessDevice`API 操作或 `create-wireless-device`CLI 命令，如本示例所示。

   > Note
   >
   > 为您的目的地指定一个专属于您的Amazon Web Services 账户和的名称Amazon Web Services 区域。将目的地添加到 Amazon Sidewalk 时，Amazon IoT Core您将使用相同的目的地名称。

```
// Add your Sidewalk device by using the device profile ID.
aws iotwireless create-wireless-device --type "Sidewalk" --name sidewalk_device \
    --destination-name SidewalkDestination \
    --sidewalk DeviceProfileId="12345678-234a-45bc-67de-e8901234f0a1"
```

3. 获取设备配置文件和无线设备信息

   以 JSON 格式获取设备配置文件和无线设备信息。JSON 将包含有关设备详细信息、设备证书、私钥和 Sidewalk 制造序列号 (SMSN) 的信息。DeviceTypeId

   - 如果你使用的是Amazon IoT主机，你可以使用设备中心的 Sidewalk 选项卡为你的 Sidewalk 终端设备下载合并 JSON 文件。
   - 如果您使用的是 API 操作，请将从 API 操作GetDeviceProfile中获得的响应存储GetWirelessDevice为单独的 JSON 文件，例如*device_profile.json*和*wireless_device.json*。

   ```
   // Store device profile information as a JSON file.
   aws iotwireless get-device-profile \
       --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d" > device_profile.json

   // Store wireless device information as a JSON file.
   aws iotwireless get-wireless-device --identifier-type WirelessDeviceId \
       --identifier "23456789-abcd-0123-bcde-fabc012345678" > wireless_device.json
   ```

步骤 2：为您的 Sidewalk 终端设备创建目标

以下是您将为Amazon Sidewalk添加目的地所要执行Amazon IoT Core的步骤的概述。使用Amazon Web Services Management Console、或Amazon IoT无线 API 操作或Amazon CLI，您可以运行以下步骤来创建Amazon IoT规则和目标。然后，您可以连接到硬件平台，查看和交换消息。有关用于本节示例的Amazon CLI示例 IAM 角色和Amazon IoT规则，请参阅为您的目标创建 IAM 角色和 IoT 规则 (p. 1221)。

1. 创建 IAM 角色

   创建一个 IAM 角色，向 Amazon Sidewalk 授Amazon IoT Core予向Amazon IoT规则发送数据的权限。要创建角色，请使用 CreateRoleAPI 操作或 create-roleCLI 命令。您可以将角色命名为*SidewalkRole*。

   ```
   aws iam create-role --role-name lambda-ex \
       --assume-role-policy-document file://lambda-trust-policy.json
   ```

2. 为目的地创建规则

   创建处理设备数据的Amazon IoT规则，并指定要向其发布消息的主题。连接到硬件平台后，您将看到有关此主题的消息。使用Amazon IoT Core API 操作或Amazon CLI命令为目的地创建规则。CreateTopicRulecreate-topic-rule

   ```
   aws iot create-topic-rule --rule-name Sidewalkrule \
       --topic-rule-payload file://myrule.json
   ```

3. 创建目标

   创建一个目的地，将您的 Sidewalk 设备与处理该设备的 IoT 规则关联起来，以便与其他设备一起使用 Amazon Web Services。您可以使用Amazon IoT控制台的目的地中心、CreateDestinationAPI 操作或 create-destinationCLI 命令添加目的地。

   ```
   aws iotwireless create-destination --name SidewalkDestination \
   ```

```
    --expression-type RuleName --expression SidewalkRule \
    --role-arn arn:aws:iam::123456789012:role/SidewalkRole
```

步骤 3：配置和注册终端设备

使用 Python 命令，您可以预置和注册您的终端设备。配置脚本使用您获得的设备 JSON 数据生成制造二进制映像，然后在硬件板上刷新该映像。然后，您可以注册您的终端设备以连接到硬件平台。有关更多信息，请参阅 Amazon Sidewalk 文档中的配置和注册您的终端设备。

Note

注册您的 Sidewalk 终端设备时，您的网关必须选择加入 Amazon Sidewalk，并且您的网关和设备必须处于彼此的覆盖范围内。

步骤 4：Connect Sidewalk 终端设备并交换消息

注册终端设备后，您可以连接终端设备并开始交换消息和设备数据。

1. Connect Sidewalk 终端设备

   将 HDK 连接到您的计算机，然后按照供应商文档提供的说明Connect 到 HDK。有关更多信息，请参阅 Amazon Sidewalk 文档中的配置和注册您的终端设备。
2. 查看和交换消息

   使用 MQTT 客户端订阅规则中指定的主题并查看收到的消息。您也可以使用 SendDataToWirelessDeviceAPI 操作或 send-data-to-wireless-deviceCLI 命令向您的设备发送下行链路消息并验证连接状态。

   （可选）您可以启用消息传送状态事件来检查下行链路消息是否已成功接收。

```
aws iotwireless send-data-to-wireless-device \
    --id "<Wireless_Device_ID>" \
    --payload-data "SGVsbG8gVG8gRGV2c2lt" \
    --wireless-metadata Sidewalk={Seq=1,AckModeRetryDurationSecs=10}
```

将 Sidewalk 设备连接到亚马逊Amazon IoT Core Sidewalk

本节介绍如何登录 Sidewalk 终端设备，然后将设备连接到 Sidewalk 网络。它描述了您在入门教程中执行的步骤，如中所述入门使用 Sidewalk 设备简介 (p. 1210)。您将学习如何使用Amazon IoT控制台和进行Amazon SidewalAmazon IoT Core k API 操作来加载设备。您还将了解执行这些操作的Amazon CLI命令。

先决条件

要将您的终端设备和目的地添加到Amazon IoT Core Amazon Sidewalk，您必须设置Amazon Web Services 账户. 要使用Amazon IoT无线 API 或Amazon CLI命令执行这些操作，还必须设置Amazon CLI。有关先决条件和设置的更多信息，以了解有关先决条件和设置的更多信息设置为Amazon IoT Core用于亚马逊 Sidewalk (p. 1207)。

Note

要执行配置和注册终端设备以及连接到硬件开发套件 (HDK) 的整个入门工作流程，还必须设置 Sidewalk 网关和 HDK。有关更多信息，请参阅 Amazon Sidewal k 文档中的设置硬件开发套件 (HDK) 和设置 Sidewal k 网关。

描述您的 Sidewalk 资源

在您开始并创建资源之前，我们建议您考虑 Sidewalk 终端设备、设备配置文件和目标的命名约定。 Amazon IoT Corefor Amazon Sidewalk 会为您创建的资源分配唯一标识符。但是，您可以为它们提供更具描述性的名称、添加描述或添加可选标签以帮助识别和管理它们。

> Note
>
> 目标名称在创建后将无法更改。使用您的Amazon Web Services 账户和独有的名称Amazon Web Services 区域。

有关更多信息，请参阅描述您的Sidewalk 资源 (p. 1208)：

以下主题向您展示如何将 Sidewalk 设备安装到 AmazonAmazon IoT Core Sidewalk

主题

- 将您的设备添加到亚马逊Amazon IoT Core Sidewalk (p. 1213)
- 为您的 Sidewalk 终端设备添加目标 (p. 1219)
- 连接您的 Sidewalk 设备并查看上行链路元数据格式 (p. 1223)

将您的设备添加到亚马逊Amazon IoT Core Sidewalk

在创建无线设备之前，请先创建设备配置文件。设备配置文件定义了 Sidewalk 设备的设备功能和其他参数。单个设备配置文件可以与多个设备相关联。

创建设备配置文件后，当您检索有关该配置文件的信息时，它会返回`DeviceTypeId`。在预置终端设备时，您将使用此 ID、设备证书、应用程序服务器公钥和 SMSN。

如何创建和添加您的设备

1. 为您的 Sidewalk 终端设备创建设备配置文件。以字母数字字符串形式指定要用于 Sidewalk 设备的配置文件名称。配置文件将有助于识别与之关联的设备。
   - （控制台）添加您的 Sidewalk 设备时，您还可以创建新的配置文件。这可以帮助您快速将设备添加到 Amazon IoT Core Amazon Sidewalk 并将其与个人资料相关联。
   - (API) 通过指定配置文件名称和 Sidewalk 对象来使用`CreateDeviceProfile` API 操作`sidewalk {}`。API 响应将包含配置文件 ID 和 ARN（亚马逊资源名称）。
2. 将您的无线设备添加到Amazon IoT Core适用于 Amazon 的 Sidewalk 上的。指定目标名称，然后选择在先前步骤中创建的设备配置文件。
   - （控制台）添加您的 Sidewalk 设备时，输入目标名称，然后选择您创建的配置文件。
   - (API) 使用`CreateWirelessDevice` API 操作。指定目标名称和先前获得的设备配置文件的 ID。

   无线设备参数

参数	描述	注意
目标名称	目的地的名称，用于描述其他 Amazon Web Service人将使用的设备数据的处理Amazon IoT规则。	如果尚未创建目标，可以提供任何字符串值。Amazon IoT Corefor Amazon Sidewalk 将在创建设备时创建一个空目的地，然后你可以在添加目的地时对其进行更新。
设备配置文件	您之前创建的设备配置文件。	–
3. 获取包含配置终端设备所需信息的 JSON 文件。

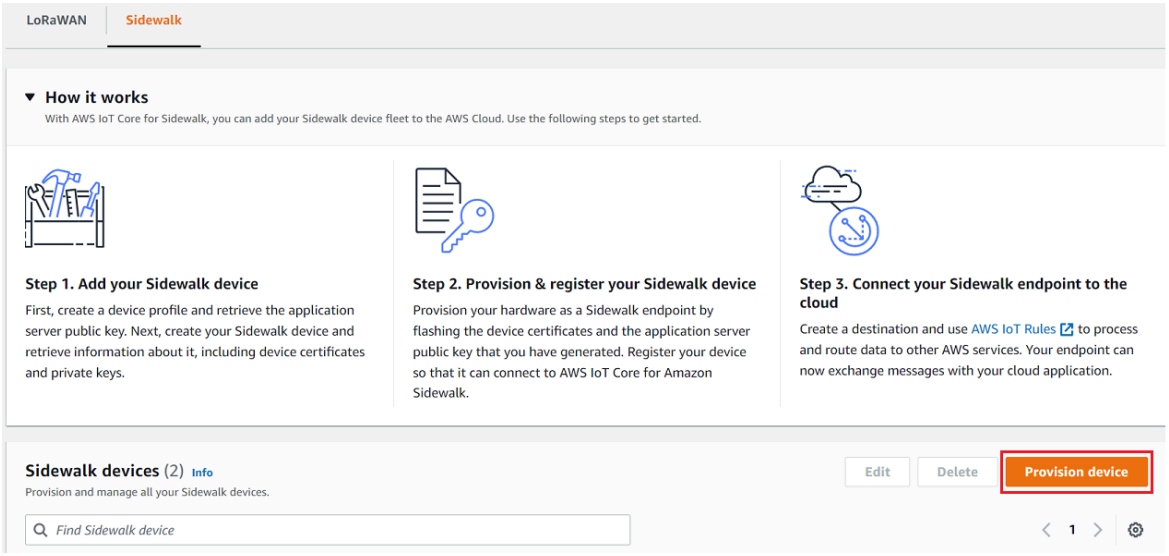
- （控制台）从您创建的 Sidewalk 设备的详细信息页面下载此文件。
- (API) 使用`GetDeviceProfile`和`GetWirelessDevice` API 操作检索有关您的设备配置文件和无线设备的信息。将 API 响应信息存储为 JSON 文件，例如*`device_profile.json`*和*`wireless_device.json`*。

添加您的设备配置文件和 Sidewalk 终端设备

本节介绍如何创建设备配置文件。它还显示了如何使用Amazon IoT控制台和将你的 Sidewalk 终端设备添加Amazon CLI到Amazon IoT Core Amazon Sidewalk 上。

添加你的 Sidewalk 设备（主机）

要使用Amazon IoT控制台添加您的 Sidewalk 设备，请转到设备中心的 Sidewalk 选项卡，选择配置设备，然后执行以下步骤。



1. 指定设备详细信息

   为您的 Sidewalk 设备指定配置信息。您还可以创建新的设备配置文件，或为 Sidewalk 设备选择现有配置文件。

   a. 指定设备名称和可选说明。描述长度最多为 2,048 个字符。创建设备后，可以编辑这些字段。
   b. 选择要与您的 Sidewalk 设备关联的设备配置文件。如果您有任何现有的设备配置文件，则可以选择您的配置文件。要创建新的配置文件，请选择“创建新配置文件”，然后输入配置文件的名称。

      Note

      要为设备配置文件添加标签，请在创建个人资料后，转到配置文件中心，然后编辑您的配置文件以添加此信息。

   c. 指定您的目的地的名称，该目的地将消息从您的设备路由到其他设备Amazon Web Services。如果您尚未创建目的地，请前往目的地中心创建目的地。然后您可以为您的 Sidewalk 设备选择目标。有关更多信息，请参阅为您的 Sidewalk 终端设备添加目标 (p. 1219)：
   d. 选择“下一步”继续添加您的 Sidewalk 设备。
2. 将人行道设备与Amazon IoT事物关联（可选）

   你可以选择将你的 Sidewalk 设备与某Amazon IoT件事物关联起来。IoT 事物是 Amazon IoT 设备注册表中的条目。这样可以让您更轻松地搜索和管理您的设备。将某个事物与您的设备关联将允许您的设备访问其它Amazon IoT Core功能。

要将您的设备与事物关联，请选择自动事物注册。

a. 为要关联您的 Sidewalk 设备的 IoT 事物输入一个唯一名称。事物名称区分大小写，并且在Amazon Web Services 账户和中必须是唯一的Amazon Web Services 区域。
b. 为您的 IoT 事物提供任何其他配置，例如使用事物类型或可用于从事物列表中筛选的可搜索属性。
c. 选择 “下一步” 并验证有关您的 Sidewalk 设备的信息，然后选择 “创建”。

添加你的 Sidewalk 设备 (CLI)

要添加您的 Sidewalk 设备并下载用于配置 Sidewalk 设备的 JSON 文件，请执行以下 API 操作。

主题

- 步骤 1：创建设备配置文件 (p. 1215)
- 步骤 2：添加您的 Sidewalk 设备 (p. 1215)

步骤 1：创建设备配置文件

要在中创建设备配置文件Amazon Web Services 账户，请使用 `CreateDeviceProfile`API 操作或 `create-device-profile`CLI 命令。创建设备配置文件时，指定名称并以名称/值对的形式提供任何可选标签。

例如，以下命令为您的 Sidewalk 设备创建设备配置文件：

```
aws iotwireless create-device-profile \
    --name sidewalk_profile --sidewalk {}
```

运行此命令将返回Amazon Resource Name (ARN) 和设备配置文件的 ID 作为输出。

```
{
    "DeviceProfileArn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-a1b2-3c45-67d8-e90fa1b2c34d",
    "DeviceProfileId": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
}
```

步骤 2：添加您的 Sidewalk 设备

要将您的 Sidewalk 设备添加到您的 Amazon Sidewalk 账户，请使用 `CreateWirelessDevice`API 操作或 `create-wireless-device`CLI 命令。Amazon IoT Core创建设备时，除了 Sidewalk 设备的可选名称和描述外，还要指定以下参数。

Note

如果你想将 Sidewalk 设备与某Amazon IoT件事关联起来，请使用 `AssociateWirelessDeviceWithThing`API 操作或 `associate-wireless-device-with-thing`CLI 命令。有关更多信息，请参阅将你中的 Sidewalk 终端设备关联Amazon Web Services 账户到 IoT 设备 (p. 1240)：

以下命令显示创建 Sidewalk 设备的示例：

```
aws iotwireless create-wireless-device \
     --cli-input-json "file://device.json"
```

下面显示的是 `device.json` 文件的内容。

device.json 的内容

```
{
  "Type": "Sidewalk",
  "Name": "SidewalkDevice",
  "DestinationName": "SidewalkDestination",
  "Sidewalk": {
    "DeviceProfileId": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
    }
}
```

运行此命令将返回设备 ID 和 Amazon Resource Name (ARN) 作为输出。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:WirelessDevice/23456789-abcd-0123-bcde-fabc012345678",
    "Id": "23456789-abcd-0123-bcde-fabc012345678"
}
```

获取用于配置的设备 JSON 文件

将 Sidewalk 设备添加到Amazon IoT Core Amazon Sidewalk 版后，下载包含配置终端设备所需信息的 JSON 文件。您可以使用Amazon IoT控制台或检索此信息Amazon CLI。有关如何配置设备的更多信息，请参阅 Amazon Sidewalk 文档中的配置和注册您的终端设备。

获取 JSON 文件（控制台）

要获取用于配置 Sidewalk 设备的 JSON 文件，请执行以下操作：

1. 前往 S idewalk 设备中心。
2. 选择您Amazon IoT Core为 Amazon Sidewalk 添加的设备以查看其详细信息。
3. 在您添加的设备的详细信息页面中选择下载设备 JSON 文件，获取 JSON 文件。

   将下载一个包含配置终端设备所需信息的certificate.json文件。以下显示了 JSN 文件示例。它包含设备证书、私钥、Sidewalk 制造序列号 (SMSN) 和DeviceTypeID.

   ```
   {
     "p256R1": "grg8izXoVvQ86cPVm0GMyWuZYHEBbbH ... DANKkOKoNT3bUGz+/f/pyTE+xMRdIUBZ1Bw==",
     "eD25519": "grg8izXoVvQ86cPVm0GMyWuZYHEBbbHD ... UiZmntHiUr1GfkTOFMYqRB+Aw==",
     "metadata": {
       "devicetypeid": "fe98",
       "applicationDeviceArn": "arn:aws:iotwireless:us-east-1:123456789012:WirelessDevice/897ce68e-3ca2-4ed0-85a2-30b0666c4052",
       "applicationDeviceId": "897ce68e-3ca2-4ed0-85a2-30b0666c4052",
       "smsn": "82B83C8B35E856F43CE9C3D59B418CC96B996071016DB1C3BE5901F0F3071A4A",
       "devicePrivKeyP256R1":
    "3e704bf8d319b3a475179f1d68c60737b28c708f845d0198f2d00d00c88ee018",
       "devicePrivKeyEd25519":
    "17dacb3a46ad9a42d5c520ca5f47f0167f59ce54d740aa13918465faf533b8d0"
     },
     "applicationServerPublicKey":
    "5ce29b89c2e3ce6183b41e75fe54e45f61b8bb320efbdd2abd7aefa5957a316b"
   }
   ```

在您的 Sidewalk 设备的详细信息页面中，您还将看到有关以下内容的信息：

- 设备 ID、其 Amazon Resource Name (ARN) 以及与设备相关的所有Amazon IoT事物的详细信息。

- 设备配置文件和目标详细信息。
- 从设备接收到最后一条上行链路消息的时间。
- 该状态表明您的设备已配置或已注册。

获取 JSON 文件 (CLI)

要使用 for Amazon Sidewalk API 或获取Amazon IoT Core用于配置 Sidewalk 终端设备的 JSON 文件，请将检索设备配置文件和无线设备相关信息时的 API 响应保存为 JSON 文件，例如*wireless_device.json*和*device_profile.json*临时文件。Amazon CLI你将使用它们来配置你的 Sidewalk 设备。

下面显示如何检索 JSN 文件。

主题

- 步骤 1：以 JSON 文件形式获取设备配置文件信息 (p. 1217)
- 第 2 步：以 JSON 文件形式获取 Sidewalk 设备信息 (p. 1218)

步骤 1：以 JSON 文件形式获取设备配置文件信息

使用 GetDeviceProfileAPI 操作或 get-device-profileCLI 命令获取有关您已添加到 Amazon Sidewalk 账户Amazon IoT Core的设备配置文件的信息。要检索有关您的设备配置文件的信息，请指定配置文件 ID。

然后，API 将返回有关与指定标识符和设备 ID 相匹配的设备配置文件的信息。您可以将此响应信息另存为文件，并为其指定一个名称，例如*device_profile.json*。

以下显示了 CLI 命令的示例：

```
aws iotwireless get-device-profile \
    --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d" > device_profile.json
```

运行此命令会返回设备配置文件的参数、应用程序服务器公钥和DeviceTypeID。以下显示了一个 JSON 文件，其中包含来自 API 的示例响应信息。有关 API 响应中参数的更多信息 GetDeviceProfile，

GetDeviceProfileAPI 响应（的内容***device_profile.json***）

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-
a1b2-3c45-67d8-e90fa1b2c34d",
    "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",
    "Name": "Sidewalk_profile",
    "LoRaWAN": null,
    "Sidewalk":
    {
        "ApplicationServerPublicKey":
 "a123b45c6d78e9f012a34cd5e6a7890b12c3d45e6f78a1b234c56d7e890a1234",
        "DAKCertificateMetadata": [
            {
                "DeviceTypeId: "fe98",
                "CertificateId": "43564A6D2D50524F544F54595045",
                "FactorySupport": false,
                "MaxAllowedSignature": 1000
            }
        ],
        "QualificationStatus": false
    }
}
```

第 2 步：以 JSON 文件形式获取 Sidewalk 设备信息

使用 GetWirelessDeviceAPI 操作或 get-wireless-deviceCLI 命令获取有关您添加到 Amazon Sidewalk 账户的 SideAmazon IoT Core walk 设备的信息。要获取有关您的终端设备的信息，请提供您在添加设备时获得的无线设备的标识符。

然后，API 将返回有关与指定标识符和设备 ID 相匹配的设备的信息。将此响应信息保存为 JSN 文件。为文件提供一个有意义的名称，例如*wireless_device.json*。

以下是使用 CLI 运行命令的示例：

```
aws iotwireless get-wireless-device --identifier-type WirelessDeviceId \
    --identifier "23456789-abcd-0123-bcde-fabc012345678" > wireless_device.json
```

运行此命令会返回设备详细信息、设备证书、私钥和 Sidewalk 制造序列号 (SMSN)。以下显示运行此命令的输出示例。有关 API 响应中参数的更多信息，请参阅GetWirelessDevice。

GetWirelessDeviceAPI 响应（的内容***wireless_device.json***）

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:WirelessDevice/23456789-abcd-0123-
bcde-fabc012345678",
    "Id": "23456789-abcd-0123-bcde-fabc012345678",
    "DestinationName": "SidewalkDestination",
    "Type": "Sidewalk",
    "Sidewalk": {
        "CertificateId": "4C7438772D50524F544F54595045",
        "DeviceCertificates": [
            {
                "SigningAlg": "Ed25519",

 "Value": "hDdkJw9L2uMCORjImjMHqzNR6nYYh6QKncSl5GthQNl7NKe4ounb5UMQtLjnm7zOUPYOqghCeVOLCBUiQe2ZiMBEW18J
F+GeltcafZcFKhS+O5NPcVNR/fHYaf/cn5iUbRwlz/T
+ODXvGdwkBkgDyFgoUJgn7JdzFjaneE5qzTWXUbL79i1sXToGGjP8hiD9jJhidPWhIswleydAWgO1OZGA4CjzIaSGVM1VtaLB0VDphA
uMMBfgAeL8Tdv5LkFIPIB3ZX9zt8zzmAuFRzI4MuNjWfIDnOF6AKu37WWU6/
QYhZoQrW9D/wndiCcsRGl+ANn367r/HE02Re4DOiCfs9f2rjc4LT1LKt7g/KW2ii+W
+9HYvvY0bBAI+AHx6Cx4j+djabTsvrgW2k6NU2zUSM7bdDP3z2a2+Z4WzBji/jYwt/
OP8rpsy5Ee4ywXUfCsfQ0rKOr0zay6yh27p3I3MZle2oCO4JIlqK0VbIQqsXzSSyp6XXS0lhmuGugZ1AAADGz
+gFBeX/ZNN8VJwnsNfgzj4me1HgVJdUo4W9kvx9cr2jHWkC3Oj/bdBTh1+yBjOC53yHlQK/
l1GHrEWiWPPnE434LRxnWkwr8EHD4oieJxC8fkIxkQfj+gHhU79Z
+oAAYAAAzsnf9SDIZPoDXF0TdC9POqTgld0oXDl2XPaVD4CvvLearrOSlFv+lsNbC4rgZn23MtIBM/7YQmJwmQ
+FXRup6Tkubg1hpz04J/09dxg8UiZmntHiUr1GfkTOFMYqRB+Aw=="
            },
            {
                "SigningAlg": "P256r1",
                "Value": "hDdkJw9L2uMCORjImjMHqzNR6nYYh6QKncSl5GthQNmHmGU8a
+SOqDXWwDNt3VSntpbTTQl7cMIusqweQo+JPXXWElbGh7eaxPGz4ZeF5yM2cqVNUrQr1lX/6lZ
+OLuycrFrLzzB9APi0NIMLqV/Rt7XJssHQs2RPcT1ul/2XVpa6ztULJeQi2JwhTb/k48wbh/EvafG/ibrIBIx9v7/
dwGRAPKHq7Uwb9hHnhpa8qNOUtjeUdIwJNh9vCBFX9s22t4PdortoFxbXo9C149PDDD4wqUHJGYlCsVX/
Sqqjf7Aug3h5dwdYN6cDgsuuiOm0+aBcXBGpkh7OxVxlwXkIP
+11dt23TkrSUKd0B01sc9Mc/0yEBCzx5RutKBwsefzyOl4vQX3AHgV7oD/XV73THMgGiDxQ55CPaaxN/
pm791VkQ76BSZaBeF+Su6tg0k/
eQneklt8Du5uqkyBHVxy8MvxsBIMZ73vIFwUrLHjDeq3+nOOyQqSBMnrHKU2mAwN3zb2LolwjPkKNOh1+NNnv99L2pBcNCnhnoBULWm
+BgewzYNdWrXyKkp4O3ZDa4f+5SVWvbY5eyDDXcohvz/
OcCtuRjAkzKBCvIjBDnCv1McjVdCO3+utizGntfhAo1RZstnOoRkgVF2WuMT9IrUmzYximuTXUmWtjyFSTqgNBZwHWUTlMmjlpLCVzZ
csC4HPTKr3dazdvEkhwGAAAIFByCjSp/5WHc4AhsyjMvKCsZQiKgiI8ECwjfXBaSZdY4zYsRlO3FC428H1atrFChFCZT0Bqt5LPXD38
+vAUJiP8XqiEdXeqf2mYMJ5ykoDpwkve/cUQfPpjzFQlQfvwjBwiJDANKkOKoNT3bUGz+/f/pyTE+xMRdIUBZ1Bw=="
            }
        ],
        "DeviceProfileId": "0ff5b0c6-f149-4498-af34-21993acd52a7",
        "PrivateKeys": [
            {
```

```
                "SigningAlg": "Ed25519",
                "Value": "2c24d4572327f23b9bef38097137c29224a9e979081b3d90124ac9dfa477934e"
            },
            {
                "SigningAlg": "P256r1",
                "Value": "38d526f29cfaf142f596deca187bd809ef71bc13435eedc885b63bb825d63def"
            }
        ],

 "SidewalkManufacturingSn": "843764270F4BDAE3023918C89A3307AB3351EA761887A40A9DC4A5E46B6140D9",
        "Status": "PROVISIONED"
    },

    ...

}
```

后续步骤

存储 JSON 文件，*wireless_device.json*并在下一步中*device_profile.json*临时使用它们来预置和注册您的终端设备以连接到硬件平台。有关更多信息，请参阅 Amazon Sidewalk 文档中的配置和注册您的终端设备。

为您的 Sidewalk 终端设备添加目标

使用Amazon IoT规则处理数据和设备消息并将其路由到其他服务。您还可以定义规则以处理从设备接收的二进制消息，并将消息转换为其它格式，从而更加便于其它服务使用。目标将您的 Sidewalk 终端设备与规则相关联，该规则将处理要发送到其它Amazon Web Service设备的设备数据。

如何创建和使用目的地

1. 为目的地创建Amazon IoT规则和 IAM 角色。该Amazon IoT规则指定了处理设备数据并将其路由给其他 Amazon Web Service人和您的应用程序使用的规则。IAM 角色授予访问规则的权限。
2. 使用`CreateDestination` API 操作为您的 Sidewalk 设备创建目的地。指定目标名称、规则名称、角色名称和任何可选参数。API 将返回目的地的唯一标识符，您可以在向 Amazon Sidewalk 添加终端设备时指定该标识符。Amazon IoT Core

以下显示了如何创建目的地以及目的地的Amazon IoT规则和 IAM 角色。

主题

- 为您的 Sidewalk 设备创建目标 (p. 1219)
- 为您的目标创建 IAM 角色和 IoT 规则 (p. 1221)

为您的 Sidewalk 设备创建目标

您可以使用目的地中心或使用 Amazon Sidewalk 的账户添加目的地`CreateDestination`。Amazon IoT Core创建目的地时，请指定：

- 目的地的唯一名称，用于您的 Sidewalk 终端设备。

  Note

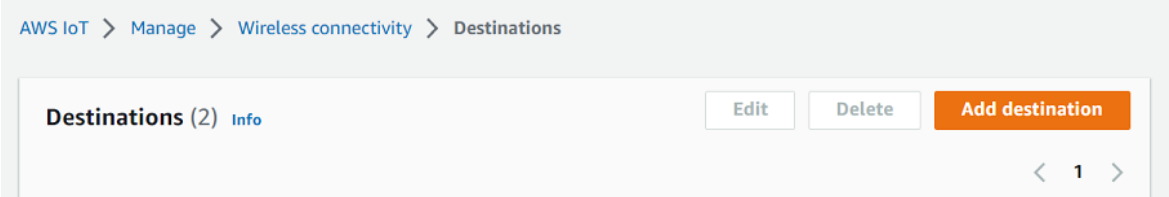
  如果您已使用目的地名称添加设备，则在创建目的地时必须使用该名称。有关更多信息，请参阅步骤 2：添加您的 Sidewalk 设备 (p. 1215)：
- 将处理设备数据的Amazon IoT规则的名称，以及发布消息的主题。

- 一个 IAM 角色，该角色授予设备数据权限以访问规则。

以下部分介绍了如何为您的目标创建Amazon IoT规则和 IAM 角色。

创建目标（控制台）

要使用Amazon IoT控制台创建目的地，请转到目标中心并选择添加目的地。



要处理设备的数据，请在创建目标时指定以下字段，然后选择 Add destination（添加目标）。

- 目标详细信息

  输入 Destination name（目标名称）以及您的目标描述（可选）。
- Rule name（规则名称）

  被配置为评估设备发送的消息并处理设备数据的 Amazon IoT 规则。规则名称将映射到您的目标。目标要求规则来处理收到的消息。您可以通过调用 Amazon IoT 规则或通过发布到 Amazon IoT 消息代理。
  - 如果选择 Enter a rule name（输入规则名称），请输入名称，然后选择 Copy（复制）以复制创建 Amazon IoT 规则时要输入的规则名称。您可以选择 Create rule（创建规则）以立即创建规则，或导航到 Amazon IoT 控制台的 Rule（规则）中心并使用该名称创建规则。

    您也可以输入规则并使用高级设置以指定主题名称。主题名称是在规则调用期间提供的，可通过使用 `topic` 规则中的表达式访问。有关Amazon IoT规则的更多信息，请参阅Amazon IoT规则。
  - 如果选择 Publish to Amazon IoT message broker（发布到消息代理），输入主题名称。然后，您可以复制 MQTT 主题名称，多个订阅者可以订阅此主题以接收发布到该主题的消息。有关更多信息，请参阅 MQTT 主题。

  有关目标的Amazon IoT规则的更多信息，以创建规则以处理 LoRa WAN 设备消息的消息。
- 角色名称

  IAM 角色，该角色授予设备数据权限以访问在 Rule name（规则名称）中命名的规则。在控制台中，您可以创建新的服务角色或选择现有的服务角色。如果要创建新的服务角色，可以输入角色名称（例如**`SidewalkDestinationRole`**），或将其留白以便 LoRa WAN 生成新的角色名称。Amazon IoT Core Amazon IoT Core然后， LoRafor WAN 将自动代表您创建具有适当权限的 IAM 角色。

创建目标 (CLI)

要创建目的地，请使用 `CreateDestination`API 操作或 `create-destination`CLI 命令。例如，以下命令为您的 Sidewalk 终端设备创建目的地：

```
aws iotwireless create-destination --name SidewalkDestination \
    --expression-type RuleName --expression SidewalkRule \
    --role-arn arn:aws:iam::123456789012:role/SidewalkRole
```

运行此命令将返回目标详细信息，包括Amazon Resource Name (ARN) 和目标名称（ARN）和目标名称。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:Destination/SidewalkDestination",
```

```
    "Name": "SidewalkDestination"
}
```

有关创建目标的更多信息，请参阅创建处理 LoRa WAN 设备消息的规则。

为您的目标创建 IAM 角色和 IoT 规则

Amazon IoT规则将设备消息发送到其他服务。 Amazon IoT规则还可以处理从 Sidewalk 终端设备收到的二进制消息，以供其他服务使用。 Amazon IoT Core适用于 Amazon Sidewalk 的目标将无线设备与规则相关联，该规则将处理要发送到其它服务的设备消息数据的消息数据。一旦 Amazon Sidewalk 收到设备的数据，Amazon IoT Core该规则就会立即对其生效。对于将数据发送到同一服务的所有设备，您可以创建可由所有设备共享的目的地。您还必须创建一个 IAM 角色来授予向规则发送数据的权限。

为您的目标创建 IAM 角色

创建一个 IAM 角色，向 Amazon Sidewalk 授Amazon IoT Core予向Amazon IoT规则发送数据的权限。要创建角色，请使用 CreateRoleAPI 操作或 create-roleCLI 命令。您可以将角色命名为*SidewalkRole*。

```
aws iam create-role --role-name SidewalkRole \
    --assume-role-policy-document '{"Version": "2012-10-17","Statement": [{ "Effect":
 "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action": "sts:AssumeRole"}]}'
```

您还可以使用 JSON 文件为角色定义信任策略。

```
aws iam create-role --role-name SidewalkRole \
    --assume-role-policy-document file://trust-policy.json
```

以下显示的是 JSN 文件的内容。

trust-policy.json 的内容

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

为您的目的地创建规则

使用Amazon IoT Core API 操作或Amazon CLI命令来创建规则。CreateTopicRulecreate-topic-rule您的目的地将使用主题规则将从您的 Sidewalk 终端设备收到的数据路由到其他设备Amazon Web Services。例如，您可以创建一个规则操作，向 Lambda 函数发送消息。您可以定义 Lambda 函数，使其从您的设备接收应用程序数据，并使用 base64 对负载数据进行解码，以便其他应用程序可以使用。

以下步骤显示如何创建 Lambda 函数，然后创建向该函数发送消息的主题规则。

1. 创建执行角色和策略

   创建一个 IAM 角色，该角色授予您的函数访问Amazon资源的权限。您还可以使用 JSON 文件为角色定义信任策略。

```
aws iam create-role --role-name lambda-ex \
    --assume-role-policy-document file://lambda-trust-policy.json
```

以下显示的是 JSN 文件的内容。

lambda-trust-policy.json 的内容

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

2. 创建和测试 Lambda 函数

   执行以下步骤以创建 base64 对负载数据进行解码的Amazon Lambda函数。

   a. 编写用于解码有效载荷数据的代码。例如，您可以使用以下 Python 代码示例。为脚本指定名称，例如*base64_decode.py*。

      base64_decode.py 的内容

      ```
      // ---------------------------------------------------------
      // ----- Python script to decode incoming binary payload -----
      // ---------------------------------------------------------
      import json
      import base64

      def lambda_handler(event, context):

          message = json.dumps(event)
          print (message)

          payload_data = base64.b64decode(event["PayloadData"])
          print(payload_data)
          print(int(payload_data,16))
      ```

   b. 将部署包创建为包含 Python 文件的 zip 文件并将其命名为*base64_decode.zip*。使用CreateFunction API 或create-function CLI 命令为示例代码创建 Lambda 函数*base64_decode.py*。

   c.
      ```
      aws lambda create-function --function-name my-function \
      --zip-file fileb://base64_decode.zip --handler index.handler \
      --runtime python3.9 --role arn:aws:iam::123456789012:role/lambda-ex
      ```

      您应当看到如下输出。创建主题规则时，您将使用输出中的亚马逊资源名称 (ARN) 值。FunctionArn

      ```
      {
          "FunctionName": "my-function",
          "FunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
          "Runtime": "python3.9",
          "Role": "arn:aws:iam::123456789012:role/lambda-ex",
      ```

```
    "Handler": "index.handler",
    "CodeSha256": "FpFMvUhayLkOoVBpNuNiIVML/tuGv2iJQ7t0yWVTU8c=",
    "Version": "$LATEST",
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "RevisionId": "88ebe1e1-bfdf-4dc3-84de-3017268fa1ff",
    ...
}
```

d. 要从命令行获取调用的日志，请在invoke命令中使用--log-type选项。响应包含一个 LogResult 字段，该字段包含来自调用的多达 4KB 的 base64 编码日志。

```
aws lambda invoke --function-name my-function out --log-type Tail
```

您应该会收到一个为 200StatusCode 的响应。有关在中创建和使用 Lambda 函数的更多信息 Amazon CLI，请参阅将 Lambda 与一起使用Amazon CLI。

3. 创建主题规则

使用CreateTopicRule API 或create-topic-rule CLI 命令创建向此 Lambda 函数发送消息的主题规则。您还可以添加重新发布到Amazon IoT主题的第二个规则操作。将此主题规则命名为*Sidewalkrule*。

```
aws iot create-topic-rule --rule-name Sidewalkrule \
    --topic-rule-payload file://myrule.json
```

您可以使用该myrule.json文件指定有关该规则的更多详细信息。例如，以下 JSON 文件显示如何重新发布到某个Amazon IoT主题以及如何向 Lambda 函数发送消息。

```
{
    "sql": "SELECT * ",
    "actions": [
       {
            // You obtained this functionArn when creating the Lambda function using
 the
            // create-function command.
            "lambda": {
                "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:my-
function"
             }
        },
        {
            // This topic can be used to observe messages exchanged between the device
 and
            // Amazon IoT Core for Amazon Sidewalk after the device is connected.
             "republish": {
                 "roleArn": "arn:aws:iam::123456789012:role/service-
role/SidewalkRepublishRole",
                 "topic": "project/sensor/observed"
             }
        }
    ],
}
```

连接您的 Sidewalk 设备并查看上行链路元数据格式

在本教程中，您将使用 MQTT 测试客户端测试连接并查看终端设备与... 之间交换的消息Amazon Web Services 云。要接收消息，请在 MQTT 测试客户端中订阅为目的地创建 IoT 规则时指定的主题。您也可以使

用SendDataToWirelessDevice API 操作将来自Amazon IoT Core Amazon Sidewalk 的下行链路消息发送到您的设备。您可以通过启用邮件传送状态事件通知来验证邮件是否已送达。

Note

有关连接和设置硬件平台的信息，请参阅 Amazon Sidewalk 文档中的配置和注册终端设备和设置硬件开发套件 (HDK)。

向您的终端设备发送下行链路消息

使用 SendDataToWirelessDeviceAPI 操作或 send-data-to-wireless-deviceCLI 命令将来自 Amazon SidewalkAmazon IoT Core 的下行链路消息发送到您的 Sidewalk 终端设备。下面显示了如何运行此命令的示例。有效负载数据是要发送的二进制文件，以 base64 编码。

```
aws iotwireless send-data-to-wireless-device \
    --id "<Wireless_Device_ID>" \
    --payload-data "SGVsbG8gVG8gRGV2c2lt" \
    --wireless-metadata Sidewalk={Seq=1,AckModeRetryDurationSecs=10}
```

以下显示了运行此命令的输出示例，该命令是发送到设备的下行链路消息的 ID。

```
{
    MessageId: "6011dd36-0043d6eb-0072-0008"
}
```

Note

SendDataToWirelessDeviceAPI 可以返回消息 ID，但消息可能无法成功传送。要检查发送到设备的消息的状态，您可以为 Sidewalk 帐户和设备启用消息传送状态事件。有关如何启用此事件的更多信息为 Sidewalk 设备启用事件 (p. 1246)。有关此事件类型的更多信息，请参阅消息传送事件。

查看来自设备的上行链路消息的格式

在连接设备之后，您可以订阅在创建目标规则时指定的主题（例如 *project/sensor/observed*），以观察来自设备的上行链路消息。

如果您在创建目的地时指定了主题名称，则可以订阅该主题以监控来自终端设备的上行链路消息。转到Amazon IoT控制台测试页面上的 MQTT 测试客户端，输入主题名称（例如，*project/sensor/observed*），然后选择订阅。

下面的示例显示了从 Sidewalk 设备发送到 Amazon IoT 的上行链路消息格式。WirelessMetadata 包含有关消息请求的元数据。

```
{
   "PayloadData":"ZjRlNjY1ZWNlNw==",
   "WirelessDeviceId":"wireless_device_id",
   "WirelessMetadata":{
      "Sidewalk":{
         "CmdExStatus":"Cmd",
         "SidewalkId":"device_id",
         "Seq":0,
         "MessageType":"messageType"
      }
    }
}
```

下表显示了上行链路元数据中不同参数的定义。*device-id*是无线设备的 ID（例如*ABCDEF1234*和）*messageType*是从设备接收的上行链路消息的类型。

Sidewalk 上行链路元数据参数

参数	描述	类型	必填
PayloadData	从无线设备发送的消息有效负载。	字符串	是
WirelessDeviceID	发送数据的无线设备的标识符	字符串	是
Sidewalk.CmdExStatus	命令运行时状态。响应类型的消息应包含状态代码，COMMAND_EXEC_STATUS_SUCCESS。但是，通知可能不包含状态代码。	枚举	否
Sidewalk.NackExStatus	响应 nack 状态，可以为 RADIO_TX_ERROR 或者 MEMORY_ERROR。	字符串数组	否

用Amazon IoT Core于 Amazon Sidewalk 的批量配置设备

您可以使用批量配置将大量终端设备批量加载到Amazon IoT Core Amazon Sidewalk。批量配置非常有用，尤其是在您在工厂制造大量设备并希望将这些设备装载到时Amazon IoT。有关制造设备的更多信息，请参阅亚马逊 Sidewalk 文档中的制造 Amazon Sidewalk 设备。

以下主题介绍批量配置的工作原理。

- 亚马逊 Sidewalk 批量配置工作流程 (p. 1225)

  本主题向您展示了批量配置的一些关键概念及其工作原理。它还显示了为将您的 Sidewalk 设备导入 Amazon SidewalAmazon IoT Core k 而必须执行的步骤。
- 在出厂支持下创建设备配置文件 (p. 1228)

  本主题介绍如何创建设备配置文件并获取对其的出厂支持。您还将学习如何在设备制造完成后检索 YubiHSM 密钥并将其发送给制造商以获取控制日志。
- 使用导入任务配置 Sidewalk 设备 (p. 1231)

  本主题向您展示如何通过创建和使用导入任务来批量配置 Sidewalk 设备。您还将学习如何更新或删除导入任务，以及如何查看任务中导入任务和设备的状态。

主题

- 亚马逊 Sidewalk 批量配置工作流程 (p. 1225)
- 在出厂支持下创建设备配置文件 (p. 1228)
- 使用导入任务配置 Sidewalk 设备 (p. 1231)

亚马逊 Sidewalk 批量配置工作流程

以下部分向您展示了批量配置的关键概念及其工作原理。批量配置涉及的步骤包括：

1. 使用 Amazon SideAmazon IoT Core walk 创建设备配置文件。
2. 向 Amazon Sidewalk 团队索取 YubiHSM 密钥，并在出厂支持下更新您的设备配置文件。

3. 将 YubiHSM 密钥发送给您的制造商，Amazon IoT Core这样 Amazon Sidewalk 就可以在设备制造完成后获得控制日志。
4. 为 Amazon Sidewalk 创建导入任务并提供要加载的设备的序列号 (SMSN)。Amazon IoT Core

批量配置的组件

以下概念向您展示了批量配置的一些关键组件，以及如何在批量配置 Sidewalk 设备时使用它们。

yubiHSM 密钥

亚马逊为您的每个 Sidewalk 产品创建一个或多个 HSM（硬件安全模块）。每个 HSM 都有一个唯一的序列号，称为 YubiHSM 密钥，该序列号打印在硬件模块上。此密钥可以从 Yubico 网页购买。

该密钥是每个 HSM 所独有的，并与您为 Amazon Sidewalk 创建Amazon IoT Core的每个设备配置文件相关联。要获取 YubiHSM 密钥，请联系亚马逊 Sidewalk 团队。如果您在出厂制造 Sidewalk 设备后，将 YubiHSM 密钥发送给制造商，Amazon Sidewalk 将收到一个包含设备序列号的控制日志文件。Amazon IoT Core然后，它将此信息与您的输入 CSV 文件进行比较，以便将设备加载到Amazon IoT。

设备认证密钥 (DAK)

当 Sidewalk 终端设备加入 Sidewalk 网络时，必须为其配置 Sidewalk 设备证书。用于设置设备的证书包括私有设备专用证书和公共设备证书，后者对应于 Sidewalk 证书链。当你的 Sidewalk 设备制造完成后，YubiHSM 会签署设备证书。

以下显示了一个包含设备证书和私钥的 JSON 文件示例。有关更多信息，请参阅获取用于配置的设备 JSON 文件 (p. 1216)：

```
{
  "p256R1": "grg8izXoVvQ86cPVm0GMyWuZYHEBbbH ... DANKkOKoNT3bUGz+/f/pyTE+xMRdIUBZ1Bw==",
  "eD25519": "grg8izXoVvQ86cPVm0GMyWuZYHEBbbHD ... UiZmntHiUr1GfkTOFMYqRB+Aw==",
  "metadata": {
    "devicetypeid": "fe98",

    ...

    "devicePrivKeyP256R1":
 "3e704bf8d319b3a475179f1d68c60737b28c708f845d0198f2d00d00c88ee018",
    "devicePrivKeyEd25519":
 "17dacb3a46ad9a42d5c520ca5f47f0167f59ce54d740aa13918465faf533b8d0"
  },
  "applicationServerPublicKey":
 "5ce29b89c2e3ce6183b41e75fe54e45f61b8bb320efbdd2abd7aefa5957a316b"
}
```

设备认证密钥 (DAK) 是您在创建设备配置文件时获得的私钥。它对应于产品证书，这是颁发给每个 Sidewalk 产品的唯一证书。当您联系亚马逊 Sidewalk 团队时，您将收到 Sidewalk 证书链、YubiHSM 密钥和使用产品设备认证密钥 (DAK) 配置的 HSM。

您的设备配置文件也会使用新的设备认证密钥 (DAK) 进行更新，并启用了出厂支持。设备配置文件的 DAK 元数据信息提供诸如 DAK 名称、证书 ID、 ApId（广告产品 ID）、是否启用出厂支持以及 DAK 可以签名的最大签名数等详细信息。

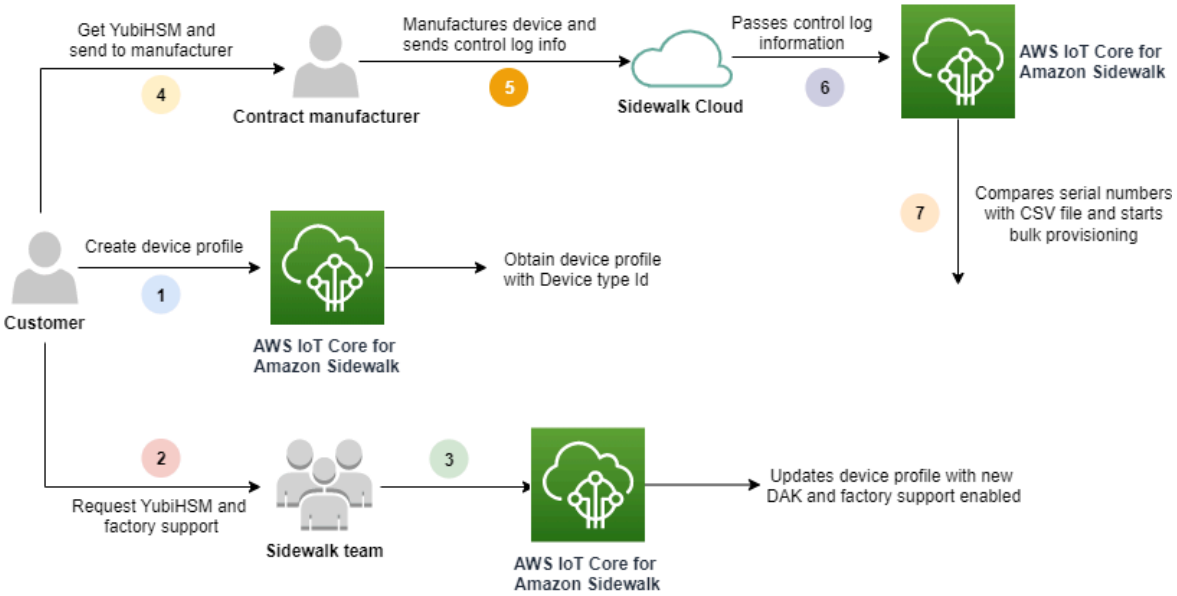
广告的产品 ID (ApId)

该ApId参数是一个由字母数字组成的字符串，用于标识宣传的产品。当您要为批量配置的 Sidewalk 设备使用给定的设备配置文件时，必须指定此字段。 Amazon IoT Core然后，Amazon Sidewalk 会生成 DAK，并通过 YubiHSM 密钥将其提供给你。相关的 DAK 信息将显示在设备配置文件中。

要获取ApId，在检索有关您创建的设备配置文件的信息后，请联系 Amazon Sidewalk Support 团队。您可以从Amazon IoT控制台、GetDeviceProfileAPI 操作或 get-device-profileCLI 命令获取设备配置文件信息。

批量配置的工作原理

此流程图显示了 Amazon SidewalkAmazon IoT Core 的批量配置是如何运作的。



以下过程说明了批量配置过程的不同步骤。

1. 为 Sidewalk 设备创建设备配置文件

   在将终端设备带回出厂之前，请先创建设备配置文件。您可以使用此配置文件配置单个设备，如中所述添加您的设备配置文件和 Sidewalk 终端设备 (p. 1214)。
2. 为您的个人资料申请出厂支持

   当你准备好将终端设备带回工厂时，请向 Amazon Sidewalk 团队索取 YubiHSM 密钥以及设备配置文件的出厂支持。
3. 获取 DAK 和出厂支持的配置文件

   然后，Amazon Sidewalk Support 团队将使用产品设备认证密钥 (DAK) 和出厂支持更新您的设备配置文件。您的设备配置文件将使用广告的产品 ID (apID) 以及新的 DAK 和证书信息（例如证书 ID）自动更新。使用此配置文件的 Sidewalk 设备有资格用于批量配置。
4. 将 YubiHSM 密钥发送给制造商 (CM)

   您的终端设备现已合格，因此您可以将 YubiHSM 密钥发送给合同制造商 (CM) 以开始制造过程。有关更多信息，请参阅亚马逊 Sidewalk 文档中的制造亚马逊 Sidewalk 设备。
5. 制造设备并发送控制日志和序列号

   CM 制造设备并生成控制日志。CM 还为您提供一个 CSV 文件，其中包含待生产设备列表及其人行道制造序列号 (SMSN)。以下代码显示示例控制日志。它包含设备的序列号、APID 和公共设备证书。

```
{
    "controlLogs": [
    {
        "version": "4-0-1",
        "device":
```

```
        {
            "serialNumber": "device1",
            "productIdentifier": {
                "advertisedProductId": "abCD"
             },
             "sidewalkData": {
                "SidewalkED25519CertificateChain": "...",
                "SidewalkP256R1CertificateChain": "..."
             }
         }
      }
   ]
}
```

6. 将控制日志信息传递给 AmazonAmazon IoT Core Sidewalk

   Amazon Sidewalk 云从制造商那里检索控制日志信息，并将这些信息传递给 Amazon Sidewalk。Amazon IoT Core然后可以创建设备及其序列号。
7. 检查序列号是否匹配并开始批量配置

   当您使用Amazon IoT控制台或进行 Amazon Sidewalk API 操作`StartWirelessDeviceImportTask`时，Amazon Sidewalk 会将从 Amazon Sidewalk 获得的每台设备的 Sidewalk 制造序列号 (SMSN) 与 CSV 文件中相应的序列号进行比较。Amazon IoT CoreAmazon IoT Core如果此信息匹配，它将启动批量配置过程并为 Amazon Sidewalk 创建要导入Amazon IoT Core的设备。

在出厂支持下创建设备配置文件

在批量配置 Amazon Sidewalk 设备之前，您必须创建设备配置文件，然后联系 Amazon Sidewalk 支持团队为其申请工厂支持。然后，Amazon Sidewalk 团队将使用新的设备认证密钥 (DAK) 更新您的设备配置文件，并为其添加出厂支持。然后，使用此配置文件的 Sidewalk 设备就有资格用Amazon IoT Core于 Amazon Sidewalk，并且可以入网进行批量配置。

以下步骤说明如何创建出厂支持的设备配置文件。

1. 创建设备配置文件

   首先创建设备配置文件。创建配置文件时，将名称和可选标签指定为名称/值对。有关所需参数以及创建和使用配置文件的更多信息，请参阅如何创建和添加您的设备 (p. 1213)。
2. 获取对配置文件的出厂支持

   然后为您的设备配置文件获取出厂支持，以便对使用此配置文件的设备进行认证。要获得资格，请与亚马逊 Sidewalk 团队一起创建票证。团队确认后，您将收到 ApId（广告中的产品编号），并且您的个人资料将使用出厂签发的 DAK 进行更新。使用此配置文件的 Sidewalk 终端设备将获得认证。

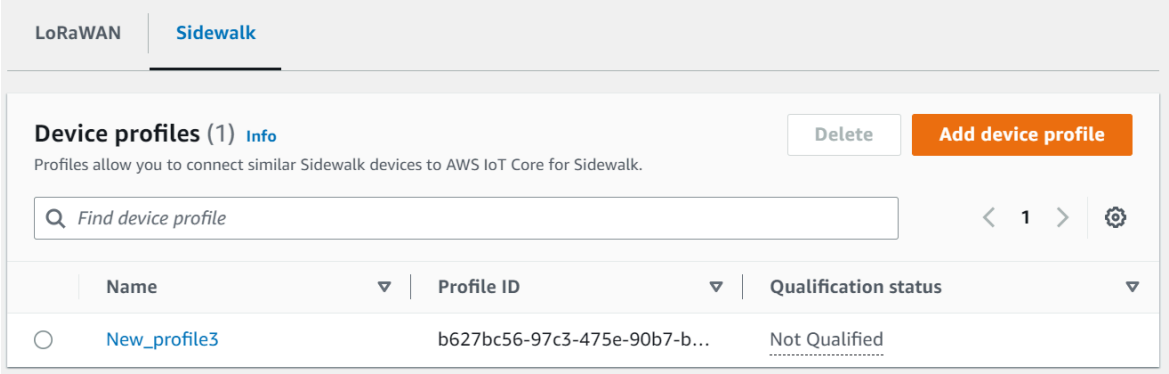
您可以使用Amazon IoT控制台、用Amazon IoT Core于 Amazon Sidewalk API 操作或Amazon CLI.

主题

- 创建配置文件（控制台）(p. 1228)
- 创建个人资料 (CLI) (p. 1229)
- 后续步骤 (p. 1231)

创建配置文件（控制台）

要使用Amazon IoT控制台创建设备配置文件，请转到配置文件中心的 Sidewalk 选项卡，然后选择创建配置文件。



要创建个人资料，请指定以下字段，然后选择提交。

- 名称

  输入个人资料的名称。
- 标签

  以名称/值对的形式输入可选标签，以帮助您更轻松地识别您的个人资料。标签还使跟踪账单费用变得更加容易。

查看个人资料信息和资格资料

您将在配置文件中心看到您创建的个人资料。选择配置文件以查看其详细信息。您将看到有关以下内容的信息：

- 设备配置文件名称和唯一标识符，以及您指定为名称值对的任何可选标签。
- 配置文件的应用程序服务器公钥和设备类型 ID。
- 资格状态，表明您使用的设备配置文件不支持出厂设置。要验证您的设备配置文件以使其在出厂时获得 Support，请联系 Amazon Sidewalk 支持人员。
- 设备认证密钥 (DAK) 信息。一旦您的设备配置文件合格，将颁发一个新的 DAK，并使用新的 DAK 信息自动更新您的配置文件。

创建个人资料 (CLI)

要创建设备配置文件，请使用 CreateDeviceProfileAPI 操作或 create-device-profileCLI 命令。例如，以下命令为您的 Sidewalk 终端设备创建配置文件。

```
aws iotwireless create-device-profile \
    --name sidewalk_device_profile --sidewalk {}
```

运行此命令会返回配置文件详细信息，其中包括配置文件的 Amazon 资源名称 (ARN) 和配置文件的 ID。

```
{
    "DeviceProfileArn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-a1b2-3c45-67d8-e90fa1b2c34d",
    "DeviceProfileId": "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
}
```

查看个人资料信息和资格资料

使用 GetDeviceProfileAPI 操作或 get-device-profileCLI 命令获取有关您已添加到 Amazon Sidewalk 账户Amazon IoT Core的设备配置文件的信息。要检索有关您的设备配置文件的信息，请指定配置文件 ID。然后，API 将返回有关与指定标识符匹配的设备配置文件的信息。

下面是示例 CLI 命令：

```
aws iotwireless get-device-profile \
    --id "12345678-234a-45bc-67de-e8901234f0a1" > device_profile.json
```

运行此命令会返回设备配置文件的参数、应用程序服务器公钥DeviceTypeId、ApId、、资格状态和DAKCertificate信息。

在此示例中，资格状态和 DAK 信息表明您的设备配置文件不合格。要验证您的个人资料，请联系 Amazon Sidewalk Support 人员，您的个人资料将获得一个没有设备限制的新 DAK。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-
a1b2-3c45-67d8-e90fa1b2c34d",
    "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",
    "Name": "Sidewalk_profile",
    "LoRaWAN": null,
    "Sidewalk":
    {
        "ApplicationServerPublicKey":
 "a123b45c6d78e9f012a34cd5e6a7890b12c3d45e6f78a1b234c56d7e890a1234",
        "DAKCertificateMetadata": [
            {
                "DeviceTypeId": "fe98",
                "CertificateId": "43564A6D2D50524F544F54595045",
                "FactorySupport": false,
                "MaxAllowedSignature": 1000
            }
        ],
        "QualificationStatus": false
    }
}
```

Amazon Sidewalk Support 团队确认此信息后，您将收到 APID 和出厂支持的 DAK，如以下示例所示。

Note

MaxAllowedSignature的-1表示 DAK 没有任何设备限制。有关 DAK 参数的信息，请参阅 DAKCertificateMetadata。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-
a1b2-3c45-67d8-e90fa1b2c34d",
    "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",
    "Name": "Sidewalk_profile",
    "LoRaWAN": null,
    "Sidewalk":
    {
        "ApplicationServerPublicKey":
 "a123b45c6d78e9f012a34cd5e6a7890b12c3d45e6f78a1b234c56d7e890a1234",
        "DAKCertificateMetadata": [
            {
                "ApId": "GZBd",
                "CertificateId": "43564A6D2D50524F544F54595045",
                "FactorySupport": true,
                "MaxAllowedSignature": -1
            }
        ],
        "QualificationStatus": true
```

```
    }
}
```

后续步骤

现在，您已经创建了具有出厂支持的 DAK 的设备配置文件，请将您从团队那里获得的 YubiHSM 密钥提供给制造商。然后，您的设备将在工厂制造，然后控制日志信息将传递到 Amazon Sidewalk，其中包含设备的序列号 (SMSN)。有关此工作流程的更多信息，请参阅亚马逊 Sidewalk 文档中的制造 Amazon Sidewalk 设备。

然后，您可以通过向 Amazon SidewalkAmazon IoT Core 提供待装设备的序列号来批量配置您的 Sidewalk 设备。当Amazon IoT Core Amazon Sidewalk 收到控制日志时，它会将控制日志中的序列号与您提供的序列号进行比较。如果序列号匹配，则导入任务将开始将您的设备导入 AmazonAmazon IoT Core Sidewalk。有关更多信息，请参阅使用导入任务配置 Sidewalk 设备 (p. 1231)：

使用导入任务配置 Sidewalk 设备

本部分介绍如何使用控制台、用Amazon IoT Core于 Amazon Sidewalk API 操作的Amazon IoT控制台批量配置 Sidewalk 设备，或者Amazon CLI。以下部分介绍如何批量配置您的 Sidewalk 设备。

主题

- Sidewalk 批量配置的工作原理 (p. 1231)
- Sidewalk 批量配置的关键注意事项 (p. 1231)
- CSV 文件格式 (p. 1232)
- 如何使用 Sidewalk 批量配置 (p. 1232)
- 批量配置人行道设备 (p. 1233)
- 查看导入任务和设备启动状态 (p. 1236)

Sidewalk 批量配置的工作原理

以下步骤说明了批量配置的工作原理。

1. 启动无线设备导入任务

   要批量配置 Sidewalk 设备，您必须创建导入任务并提供 Amazon Sidewalk 要加载的设备的 Sidewalk 制造序列号 (SMSN)。Amazon IoT Core制造商将控制日志上传到 Amazon Sidewalk 后，您在电子邮件中以 CSV 文件形式获得了设备的 Sidewalk 制造序列号 (SMSN)。有关工作流程以及如何获取控制日志的更多信息，请参阅 Amazon Sidewalk 文档中的制造 Amazon Sidewalk 设备。
2. 在后台运行导入进程

   当 fAmazon IoT Core or Amazon Sidewalk 收到导入任务请求时，它会开始设置并启动频繁轮询系统的后台进程。一旦后台进程收到导入任务指令，它就会开始读取 CSV 文件。 Amazon IoT Corefor Amazon Sidewalk 会同时检查是否已从亚马逊 Sidewalk 收到控制日志。
3. 创建无线设备记录

   收到来自 Amazon Sidewalk 的控制日志后，Amazon IoT CoreAmazon Sidewalk 会检查控制日志中的序列号是否与 CSV 文件中的 SMSN 值相匹配。如果序列号匹配，Amazon IoT CoreAmazon Sidewalk 将开始为与这些序列号对应的 Sidewalk 设备创建无线设备记录。所有设备都加载后，导入任务将标记为已完成。

Sidewalk 批量配置的关键注意事项

将您的 Sidewalk 设备批量配置到Amazon IoT Core Amazon Sidewalk 时，以下是一些关键注意事项。

- 您必须在创建设备配置文件的同一Amazon Web Services 账户位置使用Amazon IoT控制台或 Amazon Sidewalk API 操作执行批量配置。Amazon IoT Core
- 在批量配置 Sidewalk 设备之前，您的设备配置文件必须已经包含表示出厂支持的 DAK 信息。否则，使用 Amazon IoT控制台进行批量配置或批量配置 API 操作可能会失败。
- 开始导入任务后，处理 CSV 文件、导入无线设备并将其加载到 Amazon Sidewalk 可能需要至Amazon IoT Core少 10 分钟或更长时间。
- 无线设备导入任务启动后将运行 90 天。在此期间，它会检查是否已收到来自 Amazon Sidewalk 的控制日志。如果在 90 天之前未收到来自 Amazon Sidewalk 的控制日志，则当您查看任务详细信息时，该任务将被标记为已完成，并显示一条消息，表明该任务已过期。导入任务中等待控制日志的设备的启动状态将标记为失败。
- 当您尝试更新已创建的导入任务时，只能向该任务添加其他设备。在创建导入任务之后，在已添加到导入任务的设备上启动任务之前，您可以随时添加新设备。如果更新文件包含原始导入任务中已存在的设备的序列号，则这些序列号将被忽略。
- 当您请求更新操作时，将假定与您在创建导入任务时使用的相同 IAM 角色访问 Amazon S3 存储桶中的 CSV 文件。
- 仅当导入任务已成功完成或任务更新失败时，才能删除该任务。在提供不正确的 IAM 角色或找不到 Amazon S3 存储桶文件等情况下，任务可能无法更新。如果导入任务`PENDING`处于状态，则无法更新或删除该任务。
- 您导入到任务的 CSV 文件必须使用下一节中描述的格式。

CSV 文件格式

您为导入任务指定的 Amazon S3 存储桶中的 CSV 文件必须使用以下格式：

- 第 1 行必须使用关键字`smsn`，这表示要导入的 CSV 文件包含要导入的设备的 SMSN。
- 第 2 行及之后的行必须包含要载入的设备的 SMSN。设备 SMSN 必须采用 64 十六进制字符格式。

此 JSON 文件显示 CSV 文件格式示例。

```
smsn
1C1A10B0AC0A200C012BBAC2CBB1B21CB12C0CA2AC1C1BB22CAA01C1B0B01122
B122C2B1121BACA2221001AC1B22012AAC11112C11C2A100C1C2B012A1100C10
02B222C110B0A210B0A0C2C112CCCAC21C1C0B0AA1221AB1022A2CC11B1B1122
C2C021CA1C111CCAB1221C0021C1C2AAA0AA1A2A01ABC10CBAACCA2A0121022A
0CB22C01BBC2CA2C0B11001121ACB2ABB0BB0121C2BA101C012CC2B20C011AC0
```

如何使用 Sidewalk 批量配置

以下步骤介绍如何使用 Amazon Sidewalk 批量配置。

1. 提供设备序列号

   要配置您的 Sidewalk 设备，您必须提供要加载的设备的序列号。您可以使用以下方法之一配置您的设备。

   - 使用每台设备的 Sidewalk 制造序列号 (SMSN) 单独调配每台设备。当您想要测试工作流程并更快地启动设备时，此方法很有用，无需上传具有相应 IAM 角色的 CSV 文件，也无需等待设备准备好开始执行任务。
   - 通过提供一个 Amazon S3 存储桶 URL 来批量配置设备，该网址包含要在 CSV 文件中配置的设备的 SMSN。在大量设备要上载时，此方法特别有用。在这种情况下，单独启动每台设备可能很乏味。相反，您只需要提供已上传到 Amazon S3 存储桶的 CSV 文件的路径以及访问该文件的 IAM 角色即可。

2. 获取导入任务和设备入门状态

   对于您创建的每个导入任务，您可以检索有关任务加载状态和添加到该任务的设备的入门状态的信息。您还可以查看其他状态信息，例如任务或设备启动失败的原因。有关更多信息，请参阅

3. （可选）更新或删除导入任务

   您可以更新或删除您已经创建的导入任务。

   - 在已添加的设备上启动导入任务之前，您可以随时更新导入任务并向该任务添加其他设备。 Amazon IoT Corefor Amazon Sidewalk 承担的 IAM 角色与您在创建导入任务时使用的角色相同。创建任务时，指定新的 CSV 文件，其中包含要添加到任务中的设备的序列号。

     Note

     更新现有导入任务时，只能向任务中添加设备。 Amazon IoT Corefor Amazon Sidewalk 在已在导入任务中的设备与您尝试添加到任务中的设备之间执行合并操作。如果新文件包含导入任务中已存在的设备的序列号，则这些序列号将被忽略。

   - 您可以删除已成功完成的导入任务，或者在 IAM 角色信息不正确或 S3 存储段文件在创建或更新任务时不可用等情况下更新失败的导入任务。

主题

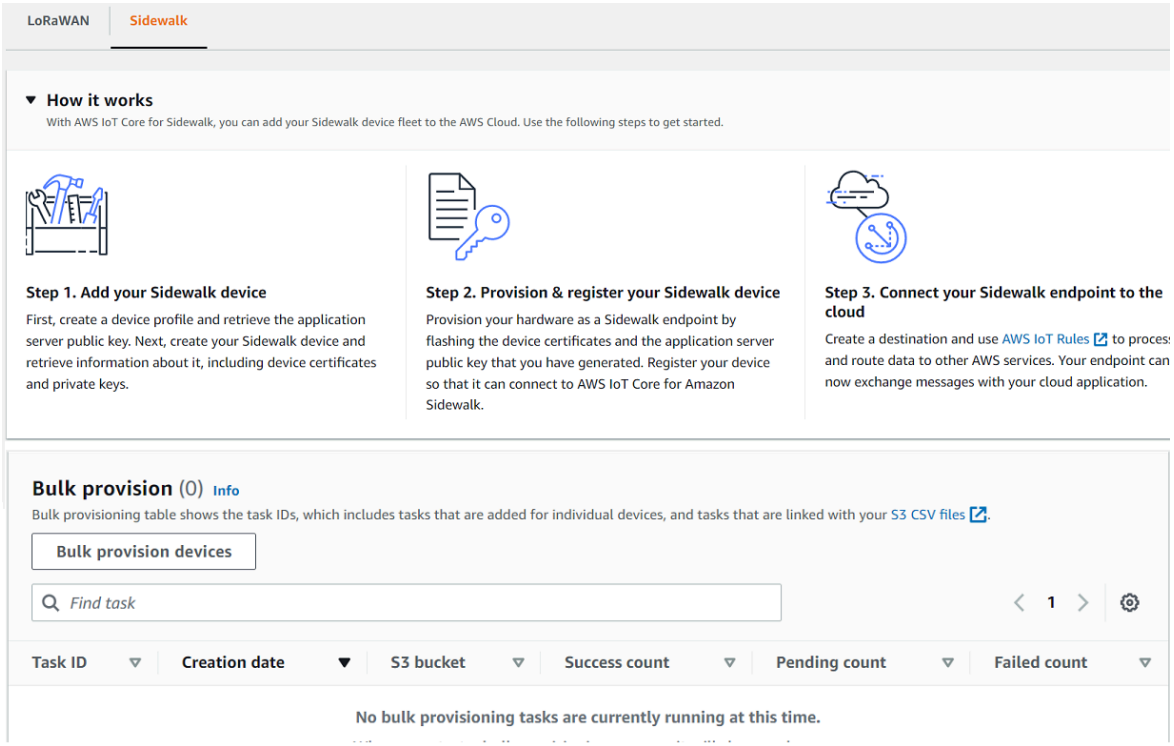
- 批量配置人行道设备 (p. 1233)
- 查看导入任务和设备启动状态 (p. 1236)

批量配置人行道设备

本部分介绍如何使用Amazon IoT控制台和 Amazon Sidewalk 向 Amazon Sidewalk 批量配置 Sidewalk 设备 Amazon CLI。Amazon IoT Core

批量配置人行道设备（控制台）

要使用Amazon IoT控制台添加您的 Sidewalk 设备，请转到设备中心的 Sidewalk 选项卡，选择批量配置设备，然后执行以下步骤。



1. 选择导入方法

   指定您希望如何将要批量载入的设备导入到 Amazon Sidewalk。Amazon IoT Core

   - 要使用其 SMSN 配置单个设备，请选择 “配置单个出厂支持的设备”。
   - 要通过提供包含设备及其 SMS 列表的 CSV 文件来批量配置设备，请选择 Use S3 存储桶。

2. 指定要载入的设备

   根据您选择的设备载入方法，添加设备信息及其序列号。

   a. 如果您选择配置出厂支持的单个设备，请指定以下信息：

      i. 要装载的每台设备的名称。该名称在您的Amazon Web Services 账户和中必须是唯一的 Amazon Web Services 区域。

      ii. 他们的 Sidewalk 制造序列号 (SMSN) 在 “输入 SMSN” 字段中。

      iii. 一个目的地，描述了将消息从设备路由到其他设备的 IoT 规则Amazon Web Services。

   b. 如果您选择使用 S3 存储桶：

      i. 提供 S3 存储桶目标信息，其中包含 S3 URL 信息。要提供 CSV 文件，请选择 Browse S3，然后选择要使用的 CSV 文件。

      Amazon IoT CoreFor Amazon Sidewalk 自动填充 S3 URL，该 URL 是 S3 存储桶中CSV 文件的路径。`s3://bucket_name/file_name` 路径的格式为：要在 Amazon Simple Storage Service控制台中查看文件，选择 View（查看）。

      ii. 提供 S3 配置角色，让Amazon IoT Core Amazon Sidewalk 代表您访问 S3 存储桶中的 CSV 文件。您可以创建新的服务角色或选择现有角色。

      要创建新角色，您可以提供角色名称或将其留白以便自动生成随机名称。

      iii. 提供描述IoT 规则的目的地，将消息从设备路由到其他Amazon Web Services设备。

3. 启动导入任务

以名称/值对的形式提供任何可选标签，然后选择 S ub mit 开始您的无线设备导入任务。

批量配置人行道设备 (CLI)

要将您的 Sidewalk 设备加入您的 Amazon Sidewalk 账户，请使用以下任何 API 操作，具体取决于您是要单独添加设备还是要提供 S3 存储桶中包含的 CSV 文件。Amazon IoT Core

- 使用 S3 CSV 文件批量上传设备

  要通过在 S3 存储桶中提供 CSV 文件来批量上传设备，请使用 StartWirelessDeviceImportTaskAPI 操作或start-wireless-device-import-taskAmazon CLI命令。创建任务时，指定 Amazon S3 存储桶中 CSV 文件的路径以及授予Amazon IoT Core Amazon Sidewalk 访问 CSV 文件权限的 IAM 角色。

  任务开始运行后，Amazon Sidewalk 将开始读取 CSV 文件，并将文件中的序列号 (SMSN) 与从 Amazon Sidewalk 收到的控制日志中的相应信息进行比较。Amazon IoT Core当序列号匹配时，它将开始创建与这些序列号对应的无线设备记录。

  以下命令显示创建导入任务的示例：

```
aws iotwireless start-wireless-device-import-task \
    --cli-input-json "file://task.json"
```

  下面显示的是 task.json 文件的内容。

  task.json 的内容

```
{
    "DestinationName": "Sidewalk_Destination",
    "Sidewalk": {
        "DeviceCreationFile": "s3://import_task_bucket/import_file1",
        "Role": "arn:aws:iam::123456789012:role/service-role/ACF1zBEI"
    }
}
```

  运行此命令会返回导入任务的 ID 和 ARN。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:ImportTask/a1b234c5-67ef-21a2-a1b2-3cd4e5f6789a"
    "Id": "a1b234c5-67ef-21a2-a1b2-3cd4e5f6789a"
}
```

- 使用 SMSN 单独配置设备

  要使用 SMSN 单独配置设备，请使用 StartSingleWirelessDeviceImportTaskAPI 操作或start-single-wireless-device-import-taskAmazon CLI命令。创建任务时，请指定 Sidewalk 目的地和要装载的设备的序列号。

  当序列号与从 Amazon Sidewalk 收到的控制日志中的相应信息相匹配时，任务将运行并创建无线设备记录。

  以下命令显示创建导入任务的示例：

```
aws iotwireless start-single-wireless-device-import-task \
    --destination-name sidewalk_destination \
```

```
    --sidewalk
 '{"SidewalkManufacturingSn": "82B83C8B35E856F43CE9C3D59B418CC96B996071016DB1C3BE5901F0F3071A4A"}'
```

运行此命令会返回导入任务的 ID 和 ARN。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:ImportTask/e2a5995e-743b-41f2-
a1e4-3ca6a5c5249f"
    "Id": "e2a5995e-743b-41f2-a1e4-3ca6a5c5249f"
}
```

更新或删除导入任务

如果要向导入任务添加其他设备，可以更新该任务。如果您不再需要某项任务或该任务失败，也可以将其删除。有关何时更新或删除任务的信息，请参阅如何使用 Sidewalk 批量配置 (p. 1232)。

> Warning
>
> 删除是永久性操作，无法撤消。删除已成功完成的导入任务不会移除已经使用该任务载入的终端设备。

要更新或删除导入任务，请执行以下操作：

- 使用 Amazon IoT 控制台

  以下步骤说明如何使用Amazon IoT控制台更新或删除导入任务。

  要更新导入任务，请执行以下操作：

  1. 前往Amazon IoT控制台的 Sidewalk 设备中心。
  2. 选择要更新的导入任务，然后选择 “编辑”。
  3. 提供另一个 S3 文件，其中包含要添加到任务中的设备序列号，然后选择 S ub mit。

  要删除导入任务，请执行以下操作：

  1. 前往Amazon IoT控制台的 Sidewalk 设备中心。
  2. 选择要删除的任务，然后选择 Delete（删除）。
- 使用Amazon IoT无线 API 或Amazon CLI

  使用以下Amazon IoT无线 API 操作或 CLI 命令更新或删除您的导入任务。

  - **UpdateWirelessDeviceImportTask**API 或 **update-wireless-device-import-task**CLI

    此 API 操作将 Amazon S3 CSV 文件的内容附加到现有导入任务中。您只能添加以前未包含在任务中的设备的序列号。有关更多信息，请参阅将设备添加到导入任务 (p. 1244)：
  - **DeleteWirelessDeviceImportTask**API 或 **delete-wireless-device-import-task**CLI

    此 API 操作删除使用导入任务 ID 标记为删除的导入任务。有关更多信息，请参阅从您的导入任务中删除Amazon Web Services 账户 (p. 1245)：

查看导入任务和设备启动状态

您的无线设备导入任务和您添加到任务的 Sidewalk 设备可以在中显示下列状态消息。您将在Amazon IoT控制台中看到这些消息，或者当您使用任何Amazon IoT Wireless API 操作或Amazon CLI命令检索有关这些任务及其设备的信息时。

查看导入任务状态信息

创建导入任务后，您可以查看您创建的导入任务以及添加到该任务的设备的入门状态。入门状态表示待登机的设备数量、成功入网的设备数量以及无法入网的设备数量。

刚刚创建导入任务时，待处理计数将显示一个与添加的设备数量相对应的值。任务启动并读取 CSV 文件以创建无线设备记录后，待处理计数将减少，成功次数将随着设备成功加入而增加。如果任何设备无法加载，失败次数将增加。

要查看导入任务和设备启动状态，请执行以下操作：

- 使用 Amazon IoT 控制台

  在Amazon IoT控制台的 Sidewalk 设备中心中，您可以看到您创建的导入任务以及设备入门状态信息摘要的计数。如果您查看所创建的任何导入任务的详细信息，则可以看到有关设备入门状态的其他信息。
- 使用Amazon IoT无线 API 或Amazon CLI

  要查看设备接入状态，请使用以下任一Amazon IoT无线 API 操作或相应的Amazon CLI命令。有关说明如何使用它们的更多信息和示例，请参阅Amazon IoT Core用于批量配置的亚马逊 Sidewalk API 操作 (p. 1242)。
  - **`ListWirelessDeviceImportTasks`**API 或 **`list-wireless-device-import-tasks`**CLI

    此 API 操作返回有关已添加到您的Amazon IoT无线账户的所有导入任务及其状态的信息。它还会返回这些任务中 Sidewalk 设备的入门状态摘要计数。
  - **`ListDevicesForWirelessDeviceImportTask`**API 或 **`list-devices-for-wireless-device-import-task`**CLI

    此 API 操作返回有关指定导入任务及其状态的信息，以及有关已添加到导入任务的所有 Sidewalk 设备及其入门状态信息的信息。
  - **`GetWirelessDeviceImportTask`**API 或 **`get-wireless-device-import-task`**CLI

    此 API 操作返回有关指定导入任务及其状态的信息，以及该任务中 Sidewalk 设备的入门状态摘要计数。

导入任务状态

您在中创建的导入任务Amazon Web Services 账户可以在中显示下列状态消息。状态表明您的导入任务是已开始处理、已完成还是失败。您还可以使用任何Amazon IoT无线 API 操作的Amazon IoT控制台或`StatusReason`参数来检索其他状态详细信息。

- 初始化

  Amazon IoT Corefor Amazon Sidewalk 已收到无线设备导入任务请求，正在设置该任务。
- 初始化

  Amazon IoT Corefor Amazon Sidewalk 已完成导入任务的设置，正在等待控制日志到达，这样它就可以使用设备的序列号 (SMSN) 导入设备并继续处理该任务。
- 待处理

  导入任务正在队列中等待进行处理。 Amazon IoT Core因为 Amazon Sidewalk 正在评估处理队列中的其他任务。
- 完成

  导入任务已处理并完成。
- FAILED

  导入任务或设备任务失败。您可以使用`StatusReason`参数来确定导入任务失败的原因，例如验证异常。

- DELETING

  导入任务已被标记为删除，正在删除。

设备入门状态

您添加到导入任务的 Sidewalk 设备可以在中显示下列状态消息。状态表明您的设备是否已准备好加载、已装机或无法加载。您还可以使用Amazon IoT控制台或Amazon IoT Wireless API 操作的`OnboardingStatusReason`参数来检索其他状态详细信息。`ListDevicesForWirelessDeviceImportTask`

- 初始化

  Amazon IoT Corefor Amazon Sidewalk 已完成导入任务的设置，正在等待控制日志到达，这样它就可以使用设备的序列号 (SMSN) 导入设备并继续处理该任务。
- 待处理

  导入任务正在队列中等待处理并开始让您的设备加入该任务。 Amazon IoT Core因为 Amazon Sidewalk 正在评估处理队列中的其他任务。
- 入职

  Sidewalk 设备已成功加入导入任务。
- FAILED

  导入任务或设备任务失败，Sidewalk 设备无法加入该任务。您可以使用`OnboardingStatusReason`参数检索有关设备上线失败原因的更多详细信息。

Amazon IoT Core用于亚马逊 Sidewalk API 操作

在加载 Sidewalk 终端设备或创建批量配置 Sidewalk 终端设备的导入任务时，您可以执行以下其他 API 操作。

以下部分包含有关这些 API 操作的更多信息。

主题

- Amazon IoT Core用于对设备配置文件执行 Amazon Sidewalk API 操作 (p. 1238)
- Amazon IoT Core用于 Sidewalk 终端设备的亚马逊 Sidewalk API 操作 (p. 1239)
- Amazon IoT Core用于人行道终端设备目的地的 Amazon Sidewalk API 操作 (p. 1241)
- Amazon IoT Core用于批量配置的亚马逊 Sidewalk API 操作 (p. 1242)

Amazon IoT Core用于对设备配置文件执行 Amazon Sidewalk API 操作

您可以为 Sidewalk 设备配置文件执行以下 API 操作：

- `CreateDeviceProfile`API 或 C `create-device-profile`LI
- `GetDeviceProfile`API 或 C `get-device-profile`LI
- `ListDeviceProfiles`API 或 C `list-device-profiles`LI
- `DeleteDeviceProfile`API 或 C `delete-device-profile`LI

以下部分介绍如何列出和删除个人资料。有关创建和检索设备配置文件的更多信息，请参阅步骤 1：创建设备配置文件 (p. 1215)。

列出您的设备配置文件Amazon Web Services 账户

您可以使用 ListDeviceProfilesAPI 操作列出您为亚马逊 SidAmazon Web Services 账户 ewalk 添加Amazon IoT Core的设备配置文件。您可以使用此信息来识别要将此配置文件关联到的设备。

要筛选列表以仅显示 Sidewalk 设备配置文件，Type请在运行 APISidewalk 时设置为。以下显示了 CLI 命令的示例：

```
aws iotwireless list-device-profiles --wireless-device-type "Sidewalk"
```

运行此命令会返回您添加的设备配置文件列表，包括其配置文件标识符和亚马逊资源名称 (ARN)。要检索有关特定配置文件的更多详细信息，请使用GetDeviceProfile API。

```
{
    "DeviceProfileList": [
        {
            "Name": "SidewalkDeviceProfile1",
            "Id": "12345678-a1b2-3c45-67d8-e90fa1b2c34d",
            "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/12345678-
a1b2-3c45-67d8-e90fa1b2c34d"
        },
        {
            "Name": "SidewalkDeviceProfile2",
            "Id": "a1b2c3d4-5678-90ab-cdef-12ab345c67de",
            "Arn": "arn:aws:iotwireless:us-east-1:123456789012:DeviceProfile/
a1b2c3d4-5678-90ab-cdef-12ab345c67de"
        }
    ]
}
```

从您的设备中删除设备配置文件Amazon Web Services 账户

您可以使用 DeleteDeviceProfileAPI 操作删除您的设备配置文件。下面显示了示例命令：

> Warning
>
> 删除操作无法撤消。设备配置文件将从您的设备中永久删除Amazon Web Services 账户。

```
aws iotwireless delete-device-profile --name "SidewalkProfile"
```

此命令不会生成任何输出。您可以使用GetDeviceProfile API 或 APListDeviceProfiles I 操作来验证配置文件是否已从您的账户中删除。

Amazon IoT Core用于 Sidewalk 终端设备的亚马逊 Sidewalk API 操作

您可以为 Sidewalk 设备执行以下 API 操作：

- CreateWirelessDeviceAPI 或 C create-wireless-deviceLI
- GetWirelessDeviceAPI 或 C get-wireless-deviceLI
- ListWirelessDevicesAPI 或 C list-wireless-devicesLI
- DeleteWirelessDeviceAPI 或 C delete-wireless-deviceLI

- UpdateWirelessDeviceAPI 或 C update-wireless-deviceLI
- AssociateWirelessDeviceWithThingAPI 或 C associate-wireless-device-with-thingLI
- DisassociateWirelessDeviceFromThingAPI 或 C disassociate-wireless-device-from-thingLI

以下部分介绍如何列出和删除设备。有关创建 Sidewalk 终端设备和检索设备信息的信息，请参阅步骤 2：添加您的 Sidewalk 设备 (p. 1215)

将你中的 Sidewalk 终端设备关联Amazon Web Services 账户到 IoT 设备

要将你的 Sidewalk 设备与某Amazon IoT件事物关联起来，请使用 AssociateWirelessDeviceWithThingAPI 操作。

中的事物Amazon IoT让您更轻松地搜索和管理您的设备。将某个事物与您的设备关联将允许设备访问其它Amazon IoT Core功能。有关使用此 API 的更多信息，请参阅 API 参考文档。

下面显示运行此命令的示例。运行此命令不会生成任何输出。

```
aws iotwireless associate-wireless-device-with-thing \
    --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d" \
    --thing-arn "arn:aws:iot:us-east-1:123456789012:thing/MySidewalkThing"
```

要取消您的 Sidewalk 设备与Amazon IoT事物的关联，请使用 DisassociateWirelessDeviceFromThingAPI 操作，如以下示例所示。

```
aws iotwireless disassociate-wireless-device-from-thing \
    --id "12345678-a1b2-3c45-67d8-e90fa1b2c34d"
```

列出您的 Sidewalk 终端设备Amazon Web Services 账户

要列出您Amazon IoT Core为亚马逊 SideAmazon Web Services 账户 walk 添加的 Sidewalk 设备，请使用 ListWirelessDevicesAPI 操作。要筛选列表以仅返回 Sidewalk 设备，WirelessDeviceType请将设置为Sidewalk。

下面显示运行此命令的示例：

```
aws iotwireless list-wireless-devices --wireless-device-type Sidewalk
```

运行此命令会返回您添加的设备列表，包括其配置文件标识符和亚马逊资源名称 (ARN)。要检索有关特定设备的其他详细信息，请使用 GetWirelessDeviceAPI 操作。

```
{
    "WirelessDeviceList": [
        {
            "Name": "mySidewalkDevice",
            "DestinationName": "SidewalkDestination",
            "Id": "1ffd32c8-8130-4194-96df-622f072a315f",
            "Type": "Sidewalk",
            "Sidewalk": {
                "SidewalkId": "1234567890123456"
            },
            "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:WirelessDevice/1ffd32c8-8130-4194-96df-622f072a315f"
        }
```

```
    ]
}
```

从您的 Sidewalk 设备中删除Amazon Web Services 账户

要删除您的 Sidewalk 设备，请将要删除WirelessDeviceID的设备传递给 DeleteWirelessDeviceAPI 操作。

下面显示示例命令：

```
aws iotwireless delete-wireless-device --id "23456789-abcd-0123-bcde-fabc012345678"
```

此命令不会生成任何输出。您可以使用GetWirelessDevice API 或ListWirelessDevices API 操作来验证设备是否已从您的账户中删除。

Amazon IoT Core用于人行道终端设备目的地的 Amazon Sidewalk API 操作

您可以对 Sidewalk 终端设备的目的地执行以下 API 操作：

- CreateDestinationAPI 或 C create-destinationLI
- GetDestinationAPI 或 C get-destinationLI
- UpdateDestinationAPI 或 C update-destinationLI
- ListDestinationsAPI 或 C list-destinationsLI
- DeleteDestinationAPI 或 C delete-destinationLI

以下部分介绍如何获取、列出、更新和删除目的地。有关创建目标的更多信息，请参阅为您的 Sidewalk 终端设备添加目标 (p. 1219)。

获取有关您的目的地的信息

您可以使用 GetDestinationAPI 操作获取有关您添加到 Amazon Sidewalk 账户Amazon IoT Core的目的地的信息。提供目标名称作为 API 的输入。API 将返回有关与指定标识符匹配的目的地的信息。

下面显示了示例命令：

```
aws iotwireless get-destination --name SidewalkDestination
```

运行此命令将返回目的地的参数。

```
{
    "Arn": "arn:aws:iotwireless:us-east-1:123456789012:Destination/IoTWirelessDestination",
    "Name": "SidewalkDestination",
    "Expression": "IoTWirelessRule",
    "ExpressionType": "RuleName",
    "RoleArn": "arn:aws:iam::123456789012:role/IoTWirelessDestinationRole"
}
```

更新目的地的属性

使用 UpdateDestinationAPI 操作更新您添加到 Amazon Sidewalk 账户Amazon IoT Core的目的地的属性。下面显示更新描述属性的命令的示例：

```
aws iotwireless update-destination --name SidewalkDestination \
    --description "Destination for messages processed using IoTWirelessRule"
```

列出您的目的地Amazon Web Services 账户

使用 ListDestinationsAPI 操作列出您添加的Amazon Web Services 账户 Amazon SideAmazon IoT Core walk 目的地。要筛选列表以仅返回 Sidewalk 终端设备的目的地，请使用WirelessDeviceType参数。

下面显示了示例命令：

```
aws iotwireless list-destinations --wireless-device-type "Sidewalk"
```

运行此命令将返回您添加的 Amazon Resource Name Name (ARN)。要检索有关特定目的地的更多详细信息，请使用GetDestination API。

```
{
    "DestinationList": [
        {
            "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:Destination/IoTWirelessDestination",
            "Name": "IoTWirelessDestination",
            "Expression": "IoTWirelessRule",
            "Description": "Destination for messages processed using IoTWirelessRule",
            "RoleArn": "arn:aws:iam::123456789012:role/IoTWirelessDestinationRole"
        },
        {
            "Arn": "arn:aws:iotwireless:us-
east-1:123456789012:Destination/IoTWirelessDestination2",
            "Name": "IoTWirelessDestination2",
            "Expression": "IoTWirelessRule2",
            "RoleArn": "arn:aws:iam::123456789012:role/IoTWirelessDestinationRole"
        }
    ]
}
```

从您的目的地中删除目的地Amazon Web Services 账户

要删除您的目的地，请将要删除的目的地的名称作为输入传递给 DeleteDestinationAPI 操作。下面显示了示例命令：

> **Warning**
>
> 删除操作无法撤消。该目的地将从您的帐户中永久删除Amazon Web Services 账户。

```
aws iotwireless delete-destination --name "SidewalkDestination"
```

此命令不会生成任何输出。您可以使用GetDestination API 或 APListDestinations I 操作来验证目的地是否已从您的账户中删除。

Amazon IoT Core用于批量配置的亚马逊 Sidewalk API 操作

您可以执行以下 API 操作来批量配置 Sidewalk 终端设备：

- StartWirelessDeviceImportTaskAPI 或 C start-wireless-device-import-taskLI

- StartSingleWirelessDeviceImportTaskAPI 或 C start-single-wireless-device-import-taskLI
- ListWirelessDeviceImportTasksAPI 或 C list-wireless-device-import-tasksLI
- ListDevicesForWirelessDeviceImportTaskAPI 或 C list-devices-for-wireless-device-import-taskLI
- GetWirelessDeviceImportTaskAPI 或 C get-wireless-device-import-taskLI
- UpdateWirelessDeviceImportTaskAPI 或 C update-wireless-device-import-taskLI
- DeleteWirelessDeviceImportTaskAPI 或 C delete-wireless-device-import-taskLI

以下部分介绍如何获取、列出、更新和删除导入任务。有关创建导入任务的更多信息，请参阅Amazon IoT Core用于批量配置的亚马逊 Sidewalk API 操作 (p. 1242)。

获取有关您的导入任务的信息

您可以使用 ListDevicesForWirelessDeviceImportTaskAPI 操作检索有关特定导入任务以及该任务中设备的入门状态的信息。作为 API 操作的输入，指定您从StartWirelessDeviceImportTask或StartSingleWirelessDeviceImportTask API 操作中获得的导入任务 ID。然后，API 将返回有关与指定标识符匹配的导入任务的信息。

下面显示了示例命令：

```
aws iotwireless list-devices-for-wireless-device-import-task --id e2a5995e-743b-41f2-a1e4-3ca6a5c5249f
```

运行此命令会返回您的导入任务信息和设备启动状态。

```
{
   "DestinationName": "SidewalkDestination",
   "ImportedWirelessDeviceList": [
      {
         "Sidewalk": {
            "OnboardingStatus": "ONBOARDED",
            "LastUpdateTime": "2023-02021T06:11:09.151Z",
            "SidewalkManufacturingSn":
 "82B83C8B35E856F43CE9C3D59B418CC96B996071016DB1C3BE5901F0F3071A4A"
         },
         "Sidewalk": {
             "OnboardingStatus": "PENDING",
             "LastUpdateTime": "2023-02021T06:22:12.061Z",
             "SidewalkManufacturingSn":
 "12345ABCDE6789FABDESBDEF123456789012345FEABC0123679AFEBC01234EF"
         },
      }
   ]
}
```

获取导入任务设备摘要

要获取您添加到特定导入任务的设备入门状态的摘要信息，请使用 GetWirelessDeviceImportTaskAPI 操作。下面显示了示例命令。

```
aws iotwireless get-wireless-device-import-task --Id "e2a5995e-743b-41f2-a1e4-3ca6a5c5249f"
```

以下代码显示命令的示例命令。

```
{
```

```
   "NumberOfFailedImportedDevices": 2,
   "NumberOfOnboardedImportedDevices": 4,
   "NumberOfPendingImportedDevices": 1
}
```

将设备添加到导入任务

使用UpdateWirelessDeviceImportTask API 操作将设备添加到您添加的现有导入任务中。您可以使用此 API 操作添加以前未包含在使用StartWirelessDeviceImportTask API 操作创建的任务中的设备的序列号 (SMSN)。

要将设备附加到导入任务，作为 API 请求的一部分，请在 Amazon S3 存储桶中指定一个新的 CSV 文件，其中包含要添加的设备的序列号。只有在当前处于导入任务中的设备的入门流程尚未开始时，才会接受该请求。如果入门流程已经开始，则UpdateWirelessDeviceImportTask API 请求将失败。

如果您仍想将设备附加到导入任务，则可以再次执行UpdateWirelessDeviceImportTask API 操作。在执行此 API 操作之前，第一个UpdateWirelessDeviceImportTask API 请求必须已完成 S3 存储段中的 CSV 文件的处理。

> **Note**
>
> 当您执行ListImportedWirelessDeviceTasks API 请求时，当前不返回使用UpdateWirelessDeviceImportTask API 操作指定的新 CSV 文件的 S3 URL。相反，API 操作会返回最初使用 API 请求发送的请求StartWirelessDeviceImportTask的 S3 网址。

下面显示了示例命令。

```
aws iotwireless update-wireless-device-import task \
    --Id "e2a5995e-743b-41f2-a1e4-3ca6a5c5249f" \
    --sidewalk '{"FileForCreateDevices": "s3://import_task_bucket/import_file3"}'
```

列出您的导入任务Amazon Web Services 账户

使用ListWirelessDeviceImportTasks API 或list-imported-wireless-device-tasks CLI 命令在您的列表中列出导入任务Amazon Web Services 账户。下面显示了示例命令。

```
aws iotwireless list-wireless-device-import-tasks
```

运行此命令会返回您创建的导入任务列表。该列表包括他们的 Amazon S3 CSV 文件和指定的 IAM 角色、导入任务 ID 和设备入门状态的摘要信息。

```
{
   "ImportWirelessDeviceTaskList": [
      {
         "FileForCreateDevices": "s3://import_task_bucket/import_file1",
         "ImportTaskId": "e2a5995e-743b-41f2-a1e4-3ca6a5c5249f",
         "NumberOfFailedImportedDevices": 1,
         "NumberOfOnboardedImportedDevices": 3,
         "NumberOfPendingImportedDevices": 2,
         "Role": "arn:aws:iam::123456789012:role/service-role/ACF1zBEI",
         "TimeStamp": "1012202218:23:55"
      },
      {
         "FileForCreateDevices": "s3://import_task_bucket/import_file2",
         "ImportTaskId": "a1b234c5-67ef-21a2-a1b2-3cd4e5f6789a",
         "NumberOfFailedImportedDevices": 2,
         "NumberOfOnboardedImportedDevices": 4,
         "NumberOfPendingImportedDevices": 1,
         "Role": "arn:aws:iam::123456789012:role/service-role/CDEFaBC1",
```

```
        "TimeStamp": "1201202210:12:20"
      }
   ]
}
```

从您的导入任务中删除Amazon Web Services 账户

要删除导入任务，请将导入任务 ID 传递给DeleteWirelessDeviceImportTask API 操作或delete-wireless-device-import-task CLI 命令。

> Warning
>
> 删除操作无法撤消。导入任务将从您的系统中永久删除Amazon Web Services 账户。

当您执行DeleteWirelessDeviceImportTask API 请求时，后台进程开始删除导入任务。请求正在进行时，正在删除导入任务中设备的序列号 (SMSN)。只有在删除完成后，您才能使用ListImportedWirelessDeviceTasks或GetImportedWirelessDeviceTasks API 操作查看此信息。

如果导入任务仍包含等待加载的设备，则只有在导入任务中的所有设备都已加载或无法加载之后，才会处理DeleteWirelessDeviceImportTask API 请求。导入任务将在 90 天后过期，任务过期后，可以将其从您的账户中删除。但是，使用导入任务成功加载的设备将不会被删除。

> Note
>
> 如果您尝试使用DeleteWirelessDeviceImportTask API 请求创建另一个包含待删除设备的序列号的导入任务，则StartWirelessDeviceImportTask API 操作将返回错误。

下面显示了示例命令：

```
aws iotwireless delete-import-task --Id "e2a5995e-743b-41f2-a1e4-3ca6a5c5249f"
```

此命令不会生成任何输出。删除任务后，要验证导入任务是否已从您的账户中删除，您可以使用GetWirelessDeviceImportTask API 操作或ListWirelessDeviceImportTasks API 操作。

亚马逊 SidewalAmazon IoT Core k 的监控和事件

您可以配置事件并使用 CloudWatch 日志来监视您的 Sidewalk 设备并解决任何问题。

Sidewalk 设备活动

Amazon IoT Core适用于 Amazon Sidewalk 的用于发布消息，该消息将通知您有关事件的消息，例如您账户中的 Sidewalk 设备已配置或注册的时间。事件通过 MQTT 随 JSON 负载一起发布，该负载的内容取决于事件类型。

Sidewalk 设备的事件类型

Sidewalk 资源的活动包括：

- 设备注册状态事件

  这些事件在设备注册状态发生更改时发布通知，例如在 Sidewalk 设备已配置或注册时。
- 近似事件

  当从 Sidewalk 设备收到信标时，这些事件会发布通知。当您的 Sidewalk 设备接近 Amazon Sidewalk 时，信标会定期发送。

- 消息传输状态事件

  这些事件发布有关在 Amazon Sidewalk 和 Sidewalk 设备之间Amazon IoT Core交换的消息状态的通知。例如，即使SendDataToWirelessDevice API 返回消息 ID，它也会发布一个事件来指示消息何时无法传送。

为 Sidewalk 设备启用事件

要接收事件，您的设备必须使用合适的策略，该策略允许设备连接到Amazon IoT设备网关并订阅 MQTT 事件主题，如下面的示例所示。

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource":[
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/join/*",
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/connection_status/
*",
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/
device_registration_state/*",
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/proximity/*",
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/
message_delivery_status/*"
        ]
    }]
}
```

您可以使用 for AmazonAmazon IoT Core Sidewalk API 启用事件Amazon CLI，或者 Amazon Web Services Management Console。要为所有 Sidewalk 设备配置事件，请使用UpdateEventConfigurationByResourceTypes API 或update-event-configuration-by-resource-types CLI 命令。

```
aws iotwireless update-event-configuration-by-resource-types \
   --device-registration-state Sidewalk={WirelessDeviceEventTopic="Enabled"}
```

要为单个设备配置事件，请使用UpdateResourceEventConfiguration API 或update-resource-event-configuration CLI 命令。

```
aws iotwireless update-resource-event-configuration --identifier-type WirelessDeviceID \
   --identifer "1ffd32c8-8130-4194-96df-622f072a315f" \
   --message-delivery-status Sidewalk={WirelessDeviceIdEventTopic="Enabled"}
```

有关配置事件的详细信息，请参阅 Sidewalk 事件。

监控Sidewalk 设备

您可以使用 Amazon 监控 Sidewalk 设备和实时运行的应用程序 CloudWatch。 CloudWatch 用于收集和跟踪指标、查看包含有关 Sidewalk 设备状态的信息的日志、筛选日志以仅查看错误或获取有用的见解。

使用亚马逊时 CloudWatch，请先配置日志记录角色，然后使用日志指标或 CloudWatch 见解来监控您的设备。发生的任何错误都将被记录并发送到您的 CloudWatch 日志组。然后，您可以详细了解该错误并解决问题以解决这些问题。

以下步骤说明如何记录和监控 Sidewalk 设备：

1. 创建日志记录角色来记录 Sidewalk 设备，如下面的示例策略所示。有关更多信息，请参阅为Amazon IoT无线创建日志记录角色和策略。

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

2. 要获得较少的详细日志（例如仅查看错误信息），请将日志配置为日志。有关更多信息，请参阅配置Amazon IoT无线资源的日志记录。
3. 通过在日志控制台中查看日志来监控您的 Sidewalk 设备，创建简单的过滤器并查看日志组中的日志条目。 CloudWatch 有关更多信息，请参阅查看 CloudWatch Amazon IoT无线日志条目。
4. 根据您要获取的日志信息创建高级筛选表达式，然后前往 CloudWatch Insights 筛选日志查询并获得更多见解。有关更多信息，请参阅 View CloudWatch Insights 以筛选Amazon IoT无线日志。

有关详细说明，请参阅Amazon IoT无线监控和记录。

监控和记录Amazon IoT Wireless使用亚马逊的情况 CloudWatch

您可以监控Amazon IoT Wireless资源和使用 Amazon 实时运行的应用程序 CloudWatch。您可以监控已使用 Amazon 引导的 LoRa WAN 和 Sidewalk 设备的状态 CloudWatch。

- 有关加载 LoRa WAN 设备的信息，请参阅将网关和设备连接到 Amazon IoT Core for LoRaWAN (p. 1110)。
- 有关将 Amazon Sidewalk 设备搭载到 Amazon IoT Core 上的信息，请参阅 亚马逊 SidAmazon IoT Core ewalk 入门 (p. 1206)。

CloudWatch 用于收集和跟踪指标，这些指标是您可衡量的相关资源和应用程序的变量。有关使用监控好处的更多信息，请参阅 监控 Amazon IoT (p. 394)。

如果要从 LoRa WAN 设备获取更多实时日志信息，请使用网络分析器。有关更多信息，请参阅使用网络分析器实时监控无线资源机群 (p. 1184)：

如何监控无线资源

要记录和监控无线资源，请执行以下步骤。

1. 创建日志记录角色来记录 Amazon IoT Wireless 资源，如 为 Amazon IoT Wireless 创建日志记录角色和策略 (p. 1249) 中所述。
2. Lo CloudWatch gs 控制台中日志消息的默认日志级别为ERROR，这些内容较为简略，仅包含错误信息。如果您要查看更详细的消息，我们建议您先使用 CLI 配置日志记录，如 为 Amazon IoT Wireless 资源配置日志记录 (p. 1251) 中所述。
3. 接下来，您可以通过在 Log CloudWatch s 控制台中查看日志条目来监控资源。有关更多信息，请参阅查看 CloudWatch Amazon IoT Wireless日志条目 (p. 1258)：
4. 您可以使用 Logs groups（日志组）来创建筛选条件表达式，但我们建议您首先创建简单的筛选条件，并查看日志组中的日志条目，然后转到 CloudWatch Insights 创建查询，以根据您所监控的资源或事件筛选日志条目。有关更多信息，请参阅使用 CloudWatch Insights 为筛选日志Amazon IoT Wireless (p. 1263)：

以下主题介绍如何为配置日志记录，Amazon IoT Wireless并从中收集指标 CloudWatch。除了 LoRa WAN 设备之外，您还可以使用这些主题为您添加到账户的任何 Sidewalk 设备配置日志记录并加以监控。有关如何添加这些设备的信息，请参阅 Amazon IoT Core适用于 Amazon Side (p. 1203)。

主题

- 为 Amazon IoT Wireless 配置日志记录 (p. 1248)
- Amazon IoT Wireless使用 CloudWatch 日志进行监控 (p. 1258)

为 Amazon IoT Wireless 配置日志记录

在可以监控和日志记录 Amazon IoT 活动前，首先使用 CLI 或 API 启用 Amazon IoT Wireless 资源的日志记录。

在考虑如何配置 Amazon IoT Wireless 日志记录时，除非另有指定，否则原定设置日志记录配置将确定如何记录 Amazon IoT 活动。首先，您可能要获取默认日志级别为 INFO 的详细日志。

查看初始日志后，您可以将默认日志级别更改为较低的详细程度级别 ERROR，并对可能需要更多关注的资源设置更详细的资源特定日志级别。日志级别可随时更改。

以下主题介绍了如何为 Amazon IoT Wireless 资源配置日志记录。

主题

- 为 Amazon IoT Wireless 创建日志记录角色和策略 (p. 1249)
- 为 Amazon IoT Wireless 资源配置日志记录 (p. 1251)

为 Amazon IoT Wireless 创建日志记录角色和策略

以下演示了如何仅为 Amazon IoT Wireless 资源创建日志记录角色。如果您还想为 Amazon IoT Core 创建日志记录角色，请参阅 创建日志记录角色 (p. 395)。

为 Amazon IoT Wireless 创建日志记录角色

在可以启用日志记录之前，您必须创建一个 IAM 角色和一个策略，用于向 Amazon 授予代表您监控 Amazon IoT Wireless 活动的权限。

创建 IAM 角色以进行日志记录

要为 Amazon IoT Wireless 创建日志记录角色，请打开 Roles hub of the IAM console（IAM 控制台的角色中心）并选择 Create role（创建角色）。

1. 在 Select type of trusted entity（选择受信任实体的类型）下，选择 Another Amazon 账户（其它亚马逊云科技账户）。
2. 在 Account ID（账户 ID）中，输入您的Amazon账户 ID，然后选择 Next: Permissions（下一步：权限）。
3. 在搜索框中，输入 **AWSIoTWirelessLogging**。
4. 选择策略名称旁边的框 AWSIoTWirelessLogging，然后选择下一步:标签。
5. 选择 Next: Review（下一步: 审核）。
6. 对于 Role name（角色名称），请输入 **IoTWirelessLogsRole**，然后选择 Create role（创建角色）。

编辑 IAM 角色的信任关系

在运行上一步后显示的确认信息中，选择您创建的角色的名称，IoTWirelessLogsRole。接下来，您将编辑角色以添加以下信任关系。

1. 在角色 IoT 的 “摘要” 部分中WirelessLogsRole，选择 “信任关系” 选项卡，然后选择 “编辑信任关系”。
2. 在 Policy Document（策略文档）中，更改 `Principal` 属性以使其类似于此示例。

```
"Principal": {
    "Service": "iotwireless.amazonaws.com"
},
```

在您更改 `Principal` 属性后，完整的策略文档应该如此示例所示。

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

```
      "Principal": {
        "Service": "iotwireless.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

3. 要保存更改并退出，请选择 Update Trust Policy（更新信任策略）。

Amazon IoT Wireless 的日志记录策略

以下策略文档提供了角色策略和信任策略，借助这些策略，可代表您Amazon IoT Wireless向 Sights CloudWatch 提交日志条目。

> Note
>
> 该Amazon由托管的策略文档是在您创建日志记录角色 IoT 时自动创建的WirelessLogsRole。

角色策略

以下显示了角色策略文档。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/iotwireless*"
        }
    ]
}
```

信任策略仅记录 Amazon IoT Wireless 活动：

下面显示了仅记录 Amazon IoT Wireless 活动的信任策略。

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

如果您创建了 IAM 角色以同时录入 Amazon IoT Core 活动，则策略文档将允许您记录这两个活动。有关为 Amazon IoT Core 创建的日志记录角色的信息，请参阅 创建日志记录角色 (p. 395)。

后续步骤

您已了解如何创建记录角色来录入您的 Amazon IoT Wireless 资源。预设情况下，日志的日志级别为 ERROR，因此如果您只想查看错误信息，请转到 查看 CloudWatch Amazon IoT Wireless日志条目 (p. 1258) 通过查看日志条目来监控您的无线资源。

如果需要在日志条目中获得更多信息，可以为资源或不同事件类型配置默认日志级别，例如将日志级别设置为 INFO。有关为您的资源配置日志记录的信息，请参阅 为 Amazon IoT Wireless 资源配置日志记录 (p. 1251)。

为 Amazon IoT Wireless 资源配置日志记录

要为 Amazon IoT Wireless 资源配置日志记录，您可以使用 API 或 CLI。开始监控 Amazon IoT Wireless 资源时，您可以使用原定设置配置。若要执行该操作，您可以跳过此主题，然后继续执行 Amazon IoT Wireless使用 CloudWatch 日志进行监控 (p. 1258) 来监控您的日志。

开始监控日志后，您可以使用 CLI 将日志级别更改为更详细的选项，例如提供 INFO 和 ERROR 信息并为更多资源启用日志记录。

Amazon IoT Wireless 资源和日志级别

在使用 API 或 CLI 之前，请使用下表了解可以为其配置日志记录的不同日志级别和资源。该表显示了您在监控资源时能在 CloudWatch 日志中看到的参数。如何为资源配置日志记录将决定您会在控制台中看到的日志。

有关示例 CloudWatch 日志以及如何使用这些参数记录有关Amazon IoT Wireless资源有关有用信息的内容，请参阅查看 CloudWatch Amazon IoT Wireless日志条目 (p. 1258)。

日志级别和资源

名称	可能的值	描述
logLevel	INFO、ERROR 或 DISABLED	• ERROR：显示导致操作失败的任何错误。日志中仅包含 ERROR 信息。 • INFO：提供有关事物流的高级别信息。日志包括 INFO 和 ERROR 信息。 • DISABLED：禁用所有日志记录。
resource	WirelessGateway 或者 WirelessDevice	资源类型，可以是 WirelessGateway 或 WirelessDevice。
wirelessGatewayType	LoRaWAN	无线网关的类型，如果resource是WirelessGateway，则始终为 LoRa WAN。
wirelessDeviceType	LoRaWAN 或者 Sidewalk	无线设备的类型，当 resource 为 WirelessDevice 时，可以是 LoRaWAN 或者 Sidewalk。
wirelessGatewayId	-	无线网关的标识符，当 resource 为 WirelessGateway 时。
wirelessDeviceId	-	无线设备的标识符，当 resource 为 WirelessDevice 时。

名称	可能的值	描述
event	Join、Rejoin、Registration、Uplink_Data、Downlink_Data、Uplink_Message、Downlink_Message、Location_Update、Ack、Discard_Uplink_Data、Discard_Uplink_Message、Discard_Downlink_Data、Discard_Downlink_Message、Discard_Registration、Discard_Location_Update、Discard_Ack、Discard_Join、Discard_Rejoin、Discard_Downlink_Request、Downlink_Request、Uplink_Request、Discard_Uplink_Request、Downlink_Queued、Downlink_Sent、Downlink_Retry、Downlink_Failed、Downlink_Scheduled、Downlink_Duty_Cycle_Delayed、Downlink_Rejected、Uplink_Retry、Uplink_Duplicate、Uplink_Failed、Uplink_Rejected、Uplink_Forwarded_To_NS、Uplink_Forwarded_To_AS、Uplink_Delivered、Uplink_Queued、Uplink_Dequeued、Uplink_Duty_Cycle_Delayed、Uplink_Rejected_By_Device、Downlink_Dequeued、Downlink_Forwarded_To_NS、Downlink_Forwarded_To_AS、Downlink_Delivered、Downlink_Rejected_By_Device、Device_Status、Mac_Command、Mac_Answer、Join_Request、Join_Accept、Rejoin_Request、Rejoin_Accept、Data_Up、Data_Down、Proprietary、Relay_Tx_Wor_Activate、Relay_Tx_Wor_Activate_Accept、Relay_Tx_Wor_Activate_Reject、Relay_Tx_Wor_Deactivate、Relay_Tx_Wor_Deactivate_Accept、Relay_Tx_Wor_Deactivate_Reject、Relay_Rx_Wor_Activate、Relay_Rx_Wor_Activate_Accept、Relay_Rx_Wor_Activate_Reject、Relay_Rx_Wor_Deactivate、Relay_Rx_Wor_Deactivate_Accept、Relay_Rx_Wor_Deactivate_Reject、Relay_Ed_Activate、Relay_Ed_Activate_Accept、Relay_Ed_Activate_Reject、Relay_Ed_Deactivate、Relay_Ed_Deactivate_Accept、Relay_Ed_Deactivate_Reject、Relay_Update_Uplink_List、Relay_Update_Uplink_List_Ack、Relay_Config_Uplink_List、Relay_Config_Uplink_List_Ack、Relay_Ctrl_Uplink_List、Relay_Ctrl_Uplink_List_Ack、Relay_Config_Cad、Relay_Config_Cad_Ack、Relay_Ctrl_Cad、Relay_Ctrl_Cad_Ack、Relay_Config_Second_Channel、Relay_Config_Second_Channel_Ack、Relay_Ctrl_Second_Channel、Relay_Ctrl_Second_Channel_Ack、Relay_Config_Default_Channel、Relay_Config_Default_Channel_Ack、Relay_Ctrl_Default_Channel、Relay_Ctrl_Default_Channel_Ack、Relay_Config_Extended_Channel、Relay_Config_Extended_Channel_Ack、Relay_Ctrl_Extended_Channel、Relay_Ctrl_Extended_Channel_Ack、Relay_Config_Join_Request、Relay_Config_Join_Request_Ack、Relay_Ctrl_Join_Request、Relay_Ctrl_Join_Request_Ack、Relay_Config_Udd、Relay_Config_Udd_Ack、Relay_Ctrl_Udd、Relay_Ctrl_Udd_Ack、Relay_Config_Max_Child、Relay_Config_Max_Child_Ack、Relay_Ctrl_Max_Child、Relay_Ctrl_Max_Child_Ack、Relay_Config_Mode、Relay_Config_Mode_Ack、Relay_Ctrl_Mode、Relay_Ctrl_Mode_Ack、Relay_Config_Sync、Relay_Config_Sync_Ack、Relay_Ctrl_Sync、Relay_Ctrl_Sync_Ack、Relay_Config_Qos、Relay_Config_Qos_Ack、Relay_Ctrl_Qos、Relay_Ctrl_Qos_Ack、Relay_Config_Low_Duty_Cycle、Relay_Config_Low_Duty_Cycle_Ack、Relay_Ctrl_Low_Duty_Cycle、Relay_Ctrl_Low_Duty_Cycle_Ack、Relay_Config_Uplink_Limit、Relay_Config_Uplink_Limit_Ack、Relay_Ctrl_Uplink_Limit、Relay_Ctrl_Uplink_Limit_Ack、Relay_Config_Backoff、Relay_Config_Backoff_Ack、Relay_Ctrl_Backoff、Relay_Ctrl_Backoff_Ack、Relay_Config_Second_Channel_Ack_Offset、Relay_Config_Second_Channel_Ack_Offset_Ack、Relay_Ctrl_Second_Channel_Ack_Offset、Relay_Ctrl_Second_Channel_Ack_Offset_Ack、Relay_Config_Udd_Ack、Relay_Config_Udd_Ack_Ack、Relay_Ctrl_Udd_Ack、Relay_Ctrl_Udd_Ack_Ack、Relay_Config_Join_Eui_Filter、Relay_Config_Join_Eui_Filter_Ack、Relay_Ctrl_Join_Eui_Filter、Relay_Ctrl_Join_Eui_Filter_Ack、Relay_Config_Nwk_Id_Filter、Relay_Config_Nwk_Id_Filter_Ack、Relay_Ctrl_Nwk_Id_Filter、Relay_Ctrl_Nwk_Id_Filter_Ack、Relay_Config_Dev_Addr_Filter、Relay_Config_Dev_Addr_Filter_Ack、Relay_Ctrl_Dev_Addr_Filter、Relay_Ctrl_Dev_Addr_Filter_Ack、Relay_Config_Child_Eui_Filter、Relay_Config_Child_Eui_Filter_Ack、Relay_Ctrl_Child_Eui_Filter、Relay_Ctrl_Child_Eui_Filter_Ack、Relay_Config_Parent_Eui_Filter、Relay_Config_Parent_Eui_Filter_Ack、Relay_Ctrl_Parent_Eui_Filter、Relay_Ctrl_Parent_Eui_Filter_Ack、Relay_Config_Group_Id_Filter、Relay_Config_Group_Id_Filter_Ack、Relay_Ctrl_Group_Id_Filter、Relay_Ctrl_Group_Id_Filter_Ack、Relay_Config_Dev_Addr_Mask_Filter、Relay_Config_Dev_Addr_Mask_Filter_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack、Relay_Config_Child_Eui_Mask_Filter、Relay_Config_Child_Eui_Mask_Filter_Ack、Relay_Ctrl_Child_Eui_Mask_Filter、Relay_Ctrl_Child_Eui_Mask_Filter_Ack、Relay_Config_Parent_Eui_Mask_Filter、Relay_Config_Parent_Eui_Mask_Filter_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack、Relay_Config_Group_Id_Mask_Filter、Relay_Config_Group_Id_Mask_Filter_Ack、Relay_Ctrl_Group_Id_Mask_Filter、Relay_Ctrl_Group_Id_Mask_Filter_Ack、Relay_Config_Filter_Mode、Relay_Config_Filter_Mode_Ack、Relay_Ctrl_Filter_Mode、Relay_Ctrl_Filter_Mode_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Offset、Relay_Config_Child_Eui_Mask_Filter_Ack_Offset_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Offset、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Offset_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack、Relay_Config_Group_Id_Filter_Ack、Relay_Config_Group_Id_Filter_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack、Relay_Config_Filter_Mode_Ack、Relay_Config_Filter_Mode_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack、Relay_Ctrl_Filter_Mode_Ack_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Offset、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Offset_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Offset、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Offset_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack_Ack、Relay_Config_Group_Id_Filter_Ack_Ack、Relay_Config_Group_Id_Filter_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack_Ack、Relay_Config_Filter_Mode_Ack_Ack、Relay_Config_Filter_Mode_Ack_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack_Ack_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Offset、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Offset_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Offset、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Offset_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Filter_Ack_Ack_Ack、Relay_Config_Group_Id_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack_Ack_Ack、Relay_Config_Filter_Mode_Ack_Ack_Ack、Relay_Config_Filter_Mode_Ack_Ack_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Offset、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Offset_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Offset、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Offset_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Filter_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Filter_Mode_Ack_Ack_Ack_Ack、Relay_Config_Filter_Mode_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack_Ack_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Offset、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Offset_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Offset、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Offset_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Mask_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Join_Eui_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Join_Eui_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Nwk_Id_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Nwk_Id_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Dev_Addr_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Dev_Addr_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Child_Eui_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Child_Eui_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Parent_Eui_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Parent_Eui_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Config_Group_Id_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Group_Id_Filter_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Config_Filter_Mode_Ack_Ack_Ack_Ack_Ack、Relay_Config_Filter_Mode_Ack_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack_Ack_Ack_Ack_Ack、Relay_Ctrl_Filter_Mode_Ack

API 名称	描述	IAM 策略示例
		}] }

PutResourceLogLevel	为给定资源标识符和资源类型设置日志级别覆盖。资源可以是无线网关或无线设备。
	Note
	此 API 每个账户限制 200 个日志级别覆盖。
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Effect": "Allow",
		"Action": [
		"iotwireless:PutResourceLogLevel"
		],
		"Resource": [
		"arn:aws:iotwireless:us-east-1:123456789012:WirelessDevice/012bc5ab12-cd3a-d00e-1f0e20c1204a",
		]
		}
		]
		}

ResetAllResourceLogLevels	删除所有资源（包括无线网关和无线设备）的日志级别覆盖。
	Note
	此 API 不会影响使用 UpdateLogLevelsByResourceTypes API 设置的日志级别。
		{
		"Version": "2012-10-17",
		"Statement": [
		{
		"Effect": "Allow",
		"Action": [
		"iotwireless:ResetAllResourceLogLevels"
		],
		"Resource": [
		"arn:aws:iotwireless:us-east-1:123456789012:WirelessDevice/*",
		"arn:aws:iotwireless:us-east-1:123456789012:WirelessGateway/*
		]
		}
		]
		}

API 名称	描述	IAM 策略示例

ResetResourceLogLevel	删除给定资源标识符和资源类型的日志级别覆盖。资源可以是无线网关或无线设备。	<pre>{
		    "Version": "2012-10-17",
		
		    "Statement": [
		        {
		            "Effect":
		"Allow",
		            "Action": [
		
		"iotwireless:ResetResourceLogLevel"
		
		            ],
		            "Resource": [
		
		"arn:aws:iotwireless:us-
		east-1:123456789012:WirelessDevice/012bc5
		ab12-cd3a-
		d00e-1f0e20c1204a",
		
		            ]
		        }
		    ]
		}</pre>

UpdateLogLevelsByResourceTypes	设置默认日志级别，或按资源类型设置日志级别。您可以将此 API 用于无线设备或无线网关的日志选项，并控制中显示的日志消息 CloudWatch。
	
	Note
	
	事件是可选的，事件类型与资源类型相关联。有关更多信息，请参阅事件和资源类型 (p. 1259)：
		<pre>{
		    "Version": "2012-10-17",
		
		    "Statement": [
		        {
		            "Effect":
		"Allow",
		            "Action": [
		
		"iotwireless:UpdateLogLevelsByResourceTy
		
		            ],
		            "Resource": [
		
		                "*"
		
		            ]
		        }
		    ]
		}</pre>

使用 CLI 配置资源的日志级别

本节介绍了如何使用 API 或 Amazon CLI 配置 Amazon IoT Wireless 资源的日志级别。

使用 CLI 之前：

- 请确保为要为其运行 CLI 命令的 API 创建了 IAM 策略，如前所述。
- 您需要提供要使用角色的 Amazon Resource Name (ARN)。如果需要创建用于日志记录的角色，请参阅为 Amazon IoT Wireless 创建日志记录角色和策略 (p. 1249)。

为什么要使用 Amazon CLI

默认情况下，如果您创建 IAM 角色`IoTWirelessLogsRole`，如中所述为 Amazon IoT Wireless 创建日志记录角色和策略 (p. 1249)，您将在中Amazon Web Services Management Console看到默认日志级别为的日 CloudWatch 志`ERROR`。要更改所有资源或特定资源的原定设置日志级别，请使用 Amazon IoT Wireless 无线日志记录 API 或 CLI。

如何使用 Amazon CLI

API 操作可分为以下类型，具体取决于您是要为所有资源配置日志级别还是为特定资源配置日志级别：

- API 操作，`GetLogLevelsByResourceTypes`并`UpdateLogLevelsByResourceTypes`可以为您账户中特定类型的所有资源检索和更新日志级别，例如无线网关或 LoRa WAN 或 Sidewalk 设备。
- API 操作 `GetResourceLogLevel`、`PutResourceLogLevel` 和 `ResetResourceLogLevel` 可以检索、更新和重置您使用资源标识符指定的各个资源的日志级别。
- 对于您使用 `PutResourceLogLevel` API 指定日志级别覆盖操作的所有资源， API 操作 `ResetAllResourceLogLevels` 会将日志级别覆盖重置为 `null`。

使用 CLI 为 Amazon IoT 配置资源特定的日志记录

> **Note**
>
> 您也可以使用 API 执行此流程，即使用 Amazon API 中与此处显示的 CLI 命令对应的方法。

1. 预设情况下，所有资源的日志级别都设置为 `ERROR`。要为账户中的所有资源设置默认日志级别或按资源类型设置日志级别，请使用 update-log-levels-by-resource-types 命令。以下示例显示如何创建 JSON 文件 `Input.json`，并将其作为 CLI 命令的输入提供。您可以使用此命令有选择地禁用特定类型的资源和时间的日志记录或覆盖默认日志级别。

```
{
    "DefaultLogLevel": "INFO",
    "WirelessDeviceLogOptions":
     [
        {
         "Type": "Sidewalk",
         "LogLevel": "INFO",
         "Events":
          [
            {
              "Event": "Registration",
              "LogLevel": "DISABLED"
            }
          ]
        },
        {
         "Type": "LoRaWAN",
         "LogLevel": "INFO",
         "Events":
          [
            {
             "Event": "Join",
             "LogLevel": "DISABLED"
            },
            {
             "Event": "Rejoin",
             "LogLevel": "ERROR"
            }
          ]
        }
     ],
     "WirelessGatewayLogOptions":
```

```
      [
        {
         "Type": "LoRaWAN",
         "LogLevel": "INFO",
         "Events":
          [
            {
             "Event": "CUPS_Request",
             "LogLevel": "DISABLED"
            },
            {
              "Event": "Certificate",
              "LogLevel": "ERROR"
            }
          ]
        }
      ]
}
```

其中：

WirelessDeviceLogOptions

无线设备的日志选项列表。每个日志选项都包括无线设备类型（Sidewal LoRa k 或 WAN），以及无线设备事件日志选项列表。每个无线设备事件日志选项都可以选择是否包括事件类型及其日志级别。

WirelessGatewayLogOptions

无线网关的日志选项列表。每个日志选项都包括无线网关类型 (LoRaWAN) 和无线网关事件日志选项列表。每个无线网关事件日志选项都可以选择是否包括事件类型及其日志级别。

DefaultLogLevel

用于所有资源的日志级别。有效值包括 ERROR、INFO 和 DISABLED。默认值为 INFO。

LogLevel

要用于单个资源类型和事件的日志级别。这些日志级别会覆盖默认日志级别，例如 LoRa WAN 网关的日志级INFO别，DISABLED以及ERROR适用于两种事件类型的日志级别。

运行以下命令以向命令提供 Input.json 文件作为输入。此命令不会生成任何输出。

```
aws iotwireless update-log-levels-by-resource-types \
    --cli-input-json file://input.json
```

如果要删除无线设备和无线网关的日志选项，请运行以下命令。

```
{
    "DefaultLogLevel":"DISABLED",
    "WirelessDeviceLogOptions": [],
    "WirelessGatewayLogOptions":[]
}
```

2. update-log-levels-by-resource-types 命令不返回任何输出。使用 get-log-levels-by-resource-types 命令检索资源特定的日志记录信息。该命令返回默认日志级别以及无线设备和无线网关日志选项。

   Note

   该get-log-levels-by-resource-types命令无法直接在 CloudWatch 控制台中检索日志级别。您可以使用 get-log-levels-by-resource-types 命令获取您使用 update-log-levels-by-resource-types 命令为资源指定的最新日志级别信息。

```
aws iotwireless get-log-levels-by-resource-types
```

当您运行以下命令时，它会返回您使用 update-log-levels-by-resource-types 指定的最新日志记录信息。例如，如果删除无线设备日志选项，则运行 get-log-levels-by-resource-types 将返回此值，为 null。

```
{
    "DefaultLogLevel": "INFO",
    "WirelessDeviceLogOptions": null,
     "WirelessGatewayLogOptions":
      [
        {
         "Type": "LoRaWAN",
         "LogLevel": "INFO",
         "Events":
          [
            {
             "Event": "CUPS_Request",
             "LogLevel": "DISABLED"
            },
            {
              "Event": "Certificate",
              "LogLevel": "ERROR"
            }
          ]
        }
      ]
}
```

3. 要控制单个无线网关或无线设备资源的日志级别，请使用以下 CLI 命令：

   - put-resource-log-level
   - get-resource-log-level
   - reset-resource-log-level

   有关何时使用这些 CLI 的示例，假设您的账户中有大量正在记录的无线设备或网关。如果您只想排除某些无线设备的故障，可以通过将 DefaultLogLevel 设置为 DISABLED 禁用所有无线设备的日志记录，然后针对您账户中的设备，使用 put-resource-log-level 将 LogLevel 设置为 ERROR。

   ```
   aws iotwireless put-resource-log-level \
       --resource-identifier
       --resource-type WirelessDevice
       --log-level ERROR
   ```

   在本例中，命令仅将特定无线设备资源的日志级别设置为 ERROR，而所有其它资源的日志均已禁用。此命令不会生成任何输出。要检索此信息并验证是否已设置日志级别，请使用 get-resource-log-level 命令。

4. 在上一步中，调试问题并解决错误后，您可以运行 reset-resource-log-level 命令将该资源的日志级别重置为 null。如果您使用 put-resource-log-level 命令为多个无线设备或网关资源设置日志级别覆盖（例如，为多个设备排除故障），您可以使用 reset-all-resource-log-levels 命令为所有这些资源将日志级别覆盖重置为 null。

   ```
   aws iotwireless reset-all-resource-log-levels
   ```

   此命令不会生成任何输出。要检索资源的日志记录信息，请运行 get-resource-log-level 命令。

后续步骤

您已了解如何创建日志记录角色并使用 Amazon IoT Wireless API 用于配置 Amazon IoT Wireless 资源的日志记录。接下来，要了解有关监控日志条目的信息，请转到 Amazon IoT Wireless使用 CloudWatch 日志进行监控 (p. 1258)。

Amazon IoT Wireless使用 CloudWatch 日志进行监控

Amazon IoT Wireless默认情况下，有 50 多个启用 CloudWatch 的日志条目。每个日志条目都描述了事件类型、日志级别和资源类型。有关更多信息，请参阅Amazon IoT Wireless 资源和日志级别 (p. 1251)：

如何监控 Amazon IoT Wireless 资源

如果为 Amazon IoT Wireless 启用日志记录，当消息在您的设备与 Amazon IoT 之间来回传递时， Amazon IoT Wireless 会发送关于每条消息的进度事件。预设情况下， Amazon IoT Wireless 日志条目具有原定设置的错误日志级别。如果如中所述启用日志记录为 Amazon IoT Wireless 创建日志记录角色和策略 (p. 1249)，您将在 CloudWatch 控制台中看到具有默认日志级别ERROR。使用此日志级别，消息将仅显示您正在使用的所有无线设备和网关资源的错误信息。

如果希望日志显示其它信息，例如日志级别为 INFO 的信息，或者想要禁用某些设备的日志并仅显示某些设备的日志消息，则可以使用 Amazon IoT Wireless 日志记录 API。有关更多信息，请参阅使用 CLI 配置资源的日志级别 (p. 1254)：

您还可以创建筛选条件表达式以仅显示所需消息。

在您可以在控制台中查看 Amazon IoT Wireless 日志前

要使 /aws/iotwireless 日志组显示在 CloudWatch控制台中，则必须执行以下操作。

- Amazon IoT Wireless 中已启用日志记录。有关如何启用在 Amazon IoT Wireless 中启用日志记录的更多信息，请参阅 为 Amazon IoT Wireless 配置日志记录 (p. 1248)。
- 通过执行 Amazon IoT Wireless 操作，写入部分日志条目。

要更有效地创建和使用筛选条件表达式，我们建议您按照以下主题中所述尝试使用 CloudWatch Insights。我们还建议您按照主题在此处显示的顺序进行操作。这将首先帮助您使用 CloudWatch Log groups（日志组），以了解可用于在控制台中查看日志条目的不同类型的资源、其事件类型和日志级别。然后，您可以了解如何通过使用 CloudWatch Insights 来创建筛选条件表达式，从资源中获取更多有用的信息。

主题

- 查看 CloudWatch Amazon IoT Wireless日志条目 (p. 1258)
- 使用 CloudWatch Insights 为筛选日志Amazon IoT Wireless (p. 1263)

查看 CloudWatch Amazon IoT Wireless日志条目

在根据中所述Amazon IoT Wireless为配置日志记录为 Amazon IoT Wireless 创建日志记录角色和策略 (p. 1249)并写入部分日志条目时，您可以执行以下步骤以在 CloudWatch 控制台中查看日志条目。

在Amazon IoT CloudWatch 日志组控制台中查看日志

在CloudWatch 控制台中， CloudWatch 日志显示在名为 /aws/iotwireless 的日志组中。有关 CloudWatch 日志的更多信息，请参阅CloudWatch 日志。

在 CloudWatch 控制台中查看您的Amazon IoT日志

导航到CloudWatch控制台，然后选择导航窗格中的 Log groups（日志组）。

1. 在 Filter（筛选条件）文本框中，输入 **/aws/iotwireless**，然后选择 /aws/iotwireless 日志组。
2. 要查看 Amazon IoT Wireless 为您的账户生成的完整列表，请选择 Search all（搜索全部）。要查看单个日志流，请选择展开图标。
3. 要筛选日志流，您也可以在 Filter events（筛选事件）文本框中输入一条查询。以下是可以尝试输入的查询：

   - `{ $.logLevel = "ERROR" }`

     使用此筛选条件可以查找所有日志级别为 ERROR 的日志，您可以展开各个错误流以读取错误消息，这将帮助您解决这些错误。

   - `{ $.resource = "WirelessGateway" }`

     查找适用于 WirelessGateway 资源的所有日志，无论日志级别为何。

   - `{ $.event = "CUPS_Request" && $.logLevel = "ERROR" }`

     查找所有事件状态为 CUPS_Request 且日志级别为 ERROR 的日志。

事件和资源类型

下表显示了不同类型的事件，而您将看到其日志条目。事件类型还取决于资源类型是无线设备还是无线网关。您可以对资源和事件类型使用默认日志级别，也可以通过为每个资源和事件类型指定日志级别来覆盖默认日志级别。

基于使用资源的事件类型

资源	资源类型	事件类型	
无线网关	LoRa广域网	• CUPS_Request • 证书	
无线设备	LoRa广域网	• Join • 重新加入 • Uplink_Data • Downlink_Data	
无线设备	Sidewalk	• 注册 • Uplink_Data • Downlink_Data	

以下主题包含有关这些事件类型以及无线网关和无线设备日志条目的详细信息。

主题

- 无线网关和无线设备资源的日志条目 (p. 1259)

无线网关和无线设备资源的日志条目

启用日志记录后，您可以查看无线网关和无线设备的日志条目。以下部分介绍基于您的资源和事件类型的各种日志条目。

无线网关日志条目

此部分显示了无线网关资源的一些示例日志条目，您可在CloudWatch 控制台中看到这些资源。这些日志消息的事件类型可以为 CUPS_Request 或者 Certificate，并且可以配置为在资源级别或事件级别显示 INFO、ERROR 或者 DISABLED 的日志级别。如果您只想查看错误信息，请将日志级别设置为 ERROR。ERROR 日志条目中的消息将包含有关失败原因的信息。

无线网关资源的日志条目可根据以下事件类型进行分类：

- CUPS_Request

  网关上运行的 LoRa Base Station 会定期向 Configuration and Updates Server (CUPS) 发送更新请求。对于此事件类型，如果在为您的无线网关资源配置 CLI 时将日志级别设置为 INFO，则在日志中：

  - 如果事件成功，您将看到 logLevel 为 INFO 的日志消息。这些消息将包括有关发送到您的网关的 CUPS 响应的详细信息和网关详细信息。以下是此日志条目的一个示例。有关 logLevel 和日志条目中其它字段的更多信息，请参阅 Amazon IoT Wireless 资源和日志级别 (p. 1251)。

    ```
    {
        "timestamp": "2021-05-13T16:56:08.853Z",
        "resource": "WirelessGateway",
        "wirelessGatewayId": "5da85cc8-3361-4c79-8be3-3360fb87abda",
        "wirelessGatewayType": "LoRaWAN",
        "gatewayEui": "fefffff00000000e2",
        "event": "CUPS_Request",
        "logLevel": "INFO",
        "message": "Sending CUPS response of total length 3213 to GatewayEui:
     fefffff00000000e2 with  TC Credentials,"
    }
    ```

  - 如果出现错误，您将看到 logLevel 为 ERROR 的日志条目，且消息将包含有关错误的详细信息。CUPS_Request 事件可能出现的错误包括：缺少 CUPS CRC、网关的 TC Uri 与 Amazon IoT Wireless 不匹配、缺少 IoTWirelessGatewayCertManagerRole，或者无法获取无线网关记录。以下示例显示了一个丢失的 CRC 日志条目。要解决此错误，请检查您的网关设置以验证您输入的 CUPS CRC 是否正确。

    ```
    {
        "timestamp": "2021-05-13T16:56:08.853Z",
        "resource": "WirelessGateway",
        "wirelessGatewayId": "5da85cc8-3361-4c79-8be3-3360fb87abda",
        "wirelessGatewayType": "LoRaWAN",
        "gatewayEui": "fefffff00000000e2",
        "event": "CUPS_Request",
        "logLevel": "ERROR",
        "message": "The CUPS CRC is missing from the request. Check your gateway setup and
     enter the CUPS CRC,"
    }
    ```

- 证书

  这些日志条目将帮助您检查您的无线网关是否提供了正确的证书，用于对与 Amazon IoT 的连接进行身份验证。对于此事件类型，如果在为您的无线网关资源配置 CLI 时将日志级别设置为 INFO，则在日志中：

  - 如果事件成功，您将看到 logLevel 为 INFO 的日志消息。这些消息将包括有关证书 ID 和无线网关标识符的详细信息。以下是此日志条目的一个示例。有关 logLevel 和日志条目中其它字段的更多信息，请参阅 Amazon IoT Wireless 资源和日志级别 (p. 1251)。

    ```
    {
        "resource": "WirelessGateway",
        "wirelessGatewayId": "5da85cc8-3361-4c79-8be3-3360fb87abda",
        "wirelessGatewayType": "LoRaWAN",
        "event": "Certificate",
    ```

```
    "logLevel": "INFO",
    "message": "Gateway connection authenticated.
    (CertificateId: b5942a7aee973eda24314e416889227a5e0aa5ed87e6eb89239a83f515dea17c,
 WirelessGatewayId: 5da85cc8-3361-4c79-8be3-3360fb87abda)"
}
```

- 如果出现错误，您将看到 `logLevel` 为 `ERROR` 的日志条目，且消息将包含有关错误的详细信息。`Certificate` 事件可能出现错误的示例包括无效的证书 ID、无线网关标识符，或无线网关标识符与证书 ID 之间不匹配。以下示例显示了由于无线网关标识符无效导致的 `ERROR`。要纠正该错误，请检查网关标识符。

```
{
    "resource": "WirelessGateway",
    "wirelessGatewayId": "5da85cc8-3361-4c79-8be3-3360fb87abda",
    "wirelessGatewayType": "LoRaWAN",
    "event": "Certificate",
    "logLevel": "INFO",
    "message": "The gateway connection couldn't be authenticated because a provisioned
 gateway associated with the certificate couldn't be found.
    (CertificateId: 729828e264810f6fc7134daf68056e8fd848afc32bfe8082beeb44116d709d9e)"
}
```

无线设备日志条目

此部分显示了无线设备资源的一些示例日志条目，您可在CloudWatch 控制台中看到这些资源。这些日志消息的事件类型取决于您使用的是 LoRa WAN 还是 Sidewalk 设备。每种无线设备资源或事件类型都可以配置为显示 `INFO`、`ERROR` 或者 `DISABLED` 日志级别。

> Note
>
> 您的请求不得同时包含 LoRa WAN 和 Sidewalk 无线元数据。要避免 `WANERROR` 或 Sidewalk 无线数据，请指定 LoRa WAN 或 Sidewalk 无线数据。

LoRaWAN 设备日志条目

您的 LoRa WAN 无线设备的日志条目可根据以下事件类型进行分类：

- **`Join` 和 `Rejoin`**

  当您添加 LoRa WAN 设备并将其连接到时Amazon IoT Wireless，在您的设备可以发送上行链路数据之前，您必须完成名为 `oactivation r` 的流程`join procedure`。有关更多信息，请参阅将您的无线设备添加到 Amazon IoT Core for LoRaWAN (p. 1120)：

  对于此事件类型，如果在为您的无线网关资源配置 CLI 时将日志级别设置为 `INFO`，则在日志中：

  - 如果事件成功，您将看到 `logLevel` 为 `INFO` 的日志消息。这些消息将包含有关您的加入或重新加入请求状态的详细信息。以下是此日志条目的一个示例。有关 `logLevel` 和日志条目中其它字段的更多信息，请参阅 Amazon IoT Wireless 资源和日志级别 (p. 1251)。

```
{
    "timestamp": "2021-05-13T16:56:08.853Z",
    "resource": "WirelessDevice",
    "wirelessDeviceType": "LoRaWAN",
    "WirelessDeviceId": "5da85cc8-3361-4c79-8be3-3360fb87abda",
    "devEui": "fefffff00000000e2",
    "event": "Rejoin",
    "logLevel": "INFO",
    "message": "Rejoin succeeded"
}
```

- 如果出现错误，您将看到 `logLevel` 为 `ERROR` 的日志条目，且消息将包含有关错误的详细信息。`Join`和`Rejoin`事件可能出现错误的示例包括无效的 LoRa WAN 区域设置或无效的 Message Integrity Code (MIC) 检查。以下示例显示了由于 MIC 检查引起的加入错误。要解决此错误，请检查您是否输入了正确的根密钥。

```
{
    "timestamp": "2020-11-24T01:46:50.883481989Z",
    "resource": "WirelessDevice",
    "wirelessDeviceType": "LoRaWAN",
    "WirelessDeviceId": "cb4c087c-1be5-4990-8654-ccf543ee9fff",
    "devEui": "58a0cb000020255c",
    "event": "Join",
    "logLevel": "ERROR",
    "message": "invalid MIC. It's most likely caused by wrong root keys."
}
```

- Uplink_Data and Downlink_Data

  当负载从无线设备发送至 Amazon IoT 时，事件类型 `Uplink_Data` 用于由 Amazon IoT Wireless 生成的消息。事件类型 `Downlink_Data` 用于与下行链接消息相关的消息，这些消息是从 Amazon IoT 发送到无线设备的。

  Note

  活动`Uplink_Data`并`Downlink_Data`适用于 LoRa WAN 和 Sidewalk 设备。

  对于此事件类型，如果在为您的无线设备配置 CLI 时将日志级别设置为 `INFO`，那么在日志中，您将看到：

  - 如果事件成功，您将看到 `logLevel` 为 `INFO` 的日志消息。这些消息将包含有关已发送的上行链路或下行链路消息的状态以及无线设备标识符的详细信息。下面显示了 Sidewalk 设备的此日志条目示例。有关 `logLevel` 和日志条目中其它字段的更多信息，请参阅 Amazon IoT Wireless 资源和日志级别 (p. 1251)。

```
{
    "resource": "WirelessDevice",
    "wirelessDeviceId": "5371db88-d63d-481a-868a-e54b6431845d",
    "wirelessDeviceType": "Sidewalk",
    "event": "Downlink_Data",
    "logLevel": "INFO",
    "messageId": "8da04fa8-037d-4ae9-bf67-35c4bb33da71",
    "message": "Message delivery succeeded.  MessageId: 8da04fa8-037d-4ae9-
bf67-35c4bb33da71. AWS IoT Core: {\"message\":\"OK\",\"traceId\":\"038b5b05-a340-
d18a-150d-d5a578233b09\"}"
}
```

  - 如果出现错误，您将看到 `logLevel` 为 `ERROR`，并且消息将包含有关错误的详细信息，这将帮助您解决错误。`Registration` 事件可能会出现错误的示例包括：身份验证问题、请求无效或过多、无法加密或解密负载，或者无法使用指定 ID 找到无线设备。以下示例显示了处理消息时遇到的权限错误。

```
{
    "resource": "WirelessDevice",
    "wirelessDeviceId": "cb4c087c-1be5-4990-8654-ccf543ee9fff",
    "wirelessDeviceType": "LoRaWAN",
    "event": "Uplink_Data",
    "logLevel": "ERROR",
    "message": "Cannot assume role MessageId: ef38877f-3454-4c99-96ed-5088c1cd8dee.
    Access denied: User: arn:aws:sts::005196538709:assumed-role/
DataRoutingServiceRole/6368b35fd48c445c9a14781b5d5890ed is not authorized to perform:
 sts:AssumeRole on resource: arn:aws:iam::400232685877:role/ExecuteRules_Role\tstatus
 code: 403, request id: 471c3e35-f8f3-4e94-b734-c862f63f4edb"
```

```
}
```

Sidewalk 设备日志条目

您的 Sidewalk 设备的日志条目可以根据以下事件类型进行分类：

- **`Registration`**

  这些日志条目将帮助您监控您使用 Amazon IoT Wireless 注册的任何 Sidewalk 设备的状态。对于此事件类型，如果在为您的无线设备资源配置 CLI 时将日志级别设置为 `INFO`，那么在日志中，您将看到 `logLevel` 为 `INFO` 和 `ERROR` 的日志消息。这些消息将包括有关整个注册进度的详细信息。`ERROR` 日志消息将包含有关如何解决注册设备问题的信息。

  下面显示了日志消息的示例，其日志级别为 `INFO`。有关 `logLevel` 和日志条目中其它字段的更多信息，请参阅 Amazon IoT Wireless 资源和日志级别 (p. 1251)。

```
{
    "resource": "WirelessDevice",
    "wirelessDeviceId": "8d0b2775-e19b-4b2a-a351-cb8a2734a504",
    "wirelessDeviceType": "Sidewalk",
    "event": "Registration",
    "logLevel": "INFO",
    "message": "Successfully completed device registration. Amazon SidewalkId =
 2000000002"
}
```

- Uplink_Data and Downlink_Data

  Sidewalk 设备的事`Downlink_Data`件类型`Uplink_Data`和 WAN 设备的事件类型类似于 LoRa WAN 设备的相应事件类型。有关更多信息，请参阅之前 LoRa WAN 设备日志条目中所述的 Uplink_Data 和 Downlink_Data 部分。

后续步骤

您已经学会了如何查看资源的日志条目以及在启用日志记录后可以在 CloudWatch 控制台中查看的不同日志条目Amazon IoT Wireless。虽然您可以使用 Log groups（日志组）来创建筛选条件流，但我们建议您使用 CloudWatch Insights 创建和使用筛选条件流。有关更多信息，请参阅使用 CloudWatch Insights 为筛选日志Amazon IoT Wireless (p. 1263)：

使用 CloudWatch Insights 为筛选日志Amazon IoT Wireless

虽然您可以使用 CloudWatch Logs 创建筛选条件表达式，但我们建议您使用 CloudWatch Insights 更有效地根据您的应用程序创建和使用筛选条件表达式。

建议您首先使用 CloudWatch Log groups（日志组），了解可用于在控制台中查看日志条目的不同类型的资源、其事件类型和日志级别。然后，您可以使用本页上的某些筛选条件表达式示例作为参考，为您的 Amazon IoT Wireless 资源创建自己的筛选条件。

在Amazon IoT日志洞察控制台中查看 CloudWatch 日志

在CloudWatch 控制台中，CloudWatch 日志显示在名为 /aws/iotwireless 的日志组中。有关 CloudWatch 日志的更多信息，请参阅CloudWatch 日志。

在 CloudWatch 控制台中查看您的Amazon IoT日志

导航到CloudWatch 控制台，然后在导航窗格中选择 Logs Insights。

1. 在 Filter（筛选条件）文本框中，输入 **/aws/iotwireless**，然后选择 /aws/iotwireless Logs Insights。
2. 要查看日志组的完整列表，请选择 Select log group(s)（选择日志组）。要查看 Amazon IoT Wireless 日志组，请选择 /aws/iotwireless。

现在，您可以开始输入查询来筛选日志组了。以下部分包含一些有用的查询，可帮助您获得有关资源指标的洞察。

创建有用的查询以筛选和获取对 Amazon IoT Wireless 的洞察

您可以使用筛选条件表达式通过 CloudWatch Insights 显示其它有用的日志信息。以下是一些示例查询：

仅显示特定资源类型的日志

您可以创建一个查询，以帮助您仅显示特定资源类型的日志，例如 LoRa WAN 网关或 Sidewalk 设备。例如，要筛选日志以仅显示 Sidewalk 设备的消息，您可以输入以下查询并选择 Run query（运行查询）。要保存此查询，请选择 Save（保存）。

```
fields @message
| filter @message like /Sidewalk/
```

在运行查询后，您将在 Logs（日志）选项卡中看到结果，该选项卡显示与您账户中的 Sidewalk 设备相关的日志时间戳。您还将看到一个条形图，该条形图将显示事件发生的时间（如果以前发生了与您的 Sidewalk 设备相关的事件）。如果您在 Logs（日志）选项卡中展开其中一个结果，将显示如下所示的内容。或者，如果要排除与 Sidewalk 设备相关的故障，可以添加另一个筛选条件，将日志级别设置为 ERROR 并仅显示错误信息。

```
Field           Value
@ingestionTime      1623894967640
@log              954314929104:/aws/iotwireless
@logStream    WirelessDevice-
Downlink_Data-715adccfb34170214ec2f6667ddfa13cb5af2c3ddfc52fbeee0e554a2e780bed
@message      {
                    "resource": "WirelessDevice",
                    "wirelessDeviceId": "3b058d05-4e84-4e1a-b026-4932bddf978d",
                    "wirelessDeviceType": "Sidewalk",
                    "devEui": "feffff000000011a",
                    "event": "Downlink_Data",
                    "logLevel": "INFO",
                    "messageId": "7e752a10-28f5-45a5-923f-6fa7133fedda",
                    "message": "Successfully sent downlink message. Amazon SidewalkId =
 2000000006, Sequence number = 0"
                    }
@timestamp          1623894967640
devEui            feffff000000011a
event         Downlink_Data
logLevel            INFO
message           Successfully sent downlink message. Amazon SidewalkId = 2000000006,
 Sequence number = 0
messageId     7e752a10-28f5-45a5-923f-6fa7133fedda
resource      WirelessDevice
wirelessDeviceId    3b058d05-4e84-4e1a-b026-4932bddf978d
wirelessDeviceType  Sidewalk
```

显示特定消息或事件

您可以创建一个查询，以帮助您显示特定消息并观察事件发生时间。例如，如果要查看从 LoRa WAN 无线设备发送下行链路消息的时间，可以输入以下查询并选择 Run query（运行查询）。要保存此查询，请选择 Save（保存）。

```
filter @message like /Downlink message sent/
```

在查询运行后，您将在 Logs（日志）选项卡下看到结果，该选项卡显示下行链路消息成功发送到您的无线设备时的时间戳。您还将看到一个条形图，该条形图将显示下行链路消息的发送时间（如果以前有下行链接消息已发送到您的无线设备）。如果您在 Logs（日志）选项卡中展开其中一个结果，将显示如下所示的内容。或者，如果没有发送下行链接消息，则可以修改查询，从而仅显示消息未发送时的结果，以便您可以调试问题。

```
Field           Value
@ingestionTime      1623884043676
@log              954314929104:/aws/iotwireless
@logStream    WirelessDevice-
Downlink_Data-42d0e6d09ba4d7015f4e9756fcdc616d401cd85fe3ac19854d9fbd866153c872
@message      {
                    "timestamp": "2021-06-16T22:54:00.770493863Z",
                    "resource": "WirelessDevice",
                    "wirelessDeviceId": "3b058d05-4e84-4e1a-b026-4932bddf978d",
                    "wirelessDeviceType": "LoRaWAN",
                    "devEui": "fefffff000000011a",
                    "event": "Downlink_Data",
                    "logLevel": "INFO",
                    "messageId": "7e752a10-28f5-45a5-923f-6fa7133fedda",
                    "message": "Downlink message sent. MessageId:
 7e752a10-28f5-45a5-923f-6fa7133fedda"
                    }
@timestamp          1623884040858
devEui            fefffff000000011a
event         Downlink_Data
logLevel            INFO
message           Downlink message sent. MessageId: 7e752a10-28f5-45a5-923f-6fa7133fedda
messageId     7e752a10-28f5-45a5-923f-6fa7133fedda
resource      WirelessDevice
timestamp     2021-06-16T22:54:00.770493863Z
wirelessDeviceId    3b058d05-4e84-4e1a-b026-4932bddf978d
wirelessDeviceType  LoRaWAN
```

后续步骤

您已经学会了如何使用 CloudWatch Insights 通过创建查询来筛选日志消息以获取更多有用的信息。您可以组合前面描述的一些筛选条件，并根据所监控的资源设计自己的过滤条件。有关使用 CloudWatch Insights 的更多信息，请参阅使用 Ins CloudWatch ights 分析日志数据。

使用 CloudWatch Insights 创建查询后，如果已保存查询，则可以根据需要加载并运行已保存的查询。或者，如果单击 CloudWatch Logs Insights 控制台中的 History（历史记录）按钮，您可以查看先前运行的查询并根据需要重新运行它们，或者通过创建其它查询进一步修改这些查询。

Amazon IoT Wireless 的事件通知

Amazon IoT Wireless可以发布消息，通知您引导到上的 LoRa WAN 和 Sidewalk 设备的事件Amazon IoT Core。例如，您可以收到有关事件的通知，例如您账户中的 Sidewalk 设备已配置或注册的时间。

如何通知资源有关事件

发生特定事件时，会发布事件通知。例如，在预调配 Sidewalk 设备时生成事件。每个事件将导致发送单个事件通知。事件通知通过 MQTT 随 JSON 负载一起发布。负载的内容取决于事件的类型。

Note

至少发布一次事件通知。事件消息可能会多次发布。事件通知的顺序无法保证。

事件和资源类型

下表显示您将收到通知的不同类型的事件。事件类型取决于资源类型是无线设备、无线网关还是 Sidewalk 账户。您还可以在资源级别为资源启用事件，这适用于特定类型的所有资源或选定的资源，如下一节所述。有关不同事件类型的更多信息，请参阅 LoRaWAN 资源的事件通知 (p. 1272) 和 Sidewalk 资源的事件通知 (p. 1276)。

基于资源的事件类型

资源	资源类型	事件类型	
无线设备	LoRa广域网	Join	
	Sidewalk	• 设备注册状态 • 近距离 • 消息传输状态	
无线网关	LoRa广域网	连接状态	
Sidewalk 账户	Sidewalk	• 设备注册状态 • 近距离 • 消息传输状态	

用于接收无线事件通知的策略

要接收事件通知，您的设备必须使用合适的策略，该策略允许设备连接到 Amazon IoT 设备网关并订阅 MQTT 事件主题。您还必须订阅合适的主题筛选条件。

以下是接收各种无线事件的通知时需要的策略示例。

```
{
    "Version":"2012-10-17",
    "Statement":[{
        "Effect":"Allow",
        "Action":[
            "iot:Subscribe",
            "iot:Receive"
        ],
        "Resource":[
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/join/*",
```

```
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/connection_status/
*",
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/
device_registration_state/*",
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/proximity/*",
            "arn:aws:iotwireless:region:account:/$aws/iotwireless/events/
message_delivery_status/*"
        ]
    }]
}
```

无线事件的 MQTT 主题的格式

为向您发送无线资源的事件通知，Amazon IoT 使用以美元 ($) 符号开头的 MQTT 保留主题。您可以订阅并发布这些预留主题 但是，您不能创建以美元符号开头的新主题。

> **Note**
>
> MQTT 主题是特定于您的 Amazon Web Services 账户 并使用格式 `arn:aws:iotwireless:aws-region:Amazon-account-ID:topic/Topic`。有关更多信息，请参阅MQTT 主题 (p. 94)：

无线设备的保留 MQTT 主题使用以下格式：

- 资源级主题

  这些主题适用于您的 Amazon Web Services 账户 中已引导到 Amazon IoT Wireless 的特定类型的所有资源。

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/{resourceType}/resources
  ```

- 标识符级主题

  这些主题适用于您的 Amazon Web Services 账户 中已引导到 Amazon IoT Wireless 的特定类型的选定资源（由资源标识符指定）。

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/{resourceType}/
  {resourceIdentifierType}/{resourceID}/{id}
  ```

有关资源级和标识符级的主题的更多信息，请参阅 事件配置 (p. 1269)。

下表显示了各种事件的 MQTT 主题示例：

事件和 MQTT 主题

事件	MQTT 主题	注意

Sidewalk 设备注册状态	• 资源级主题
	`$aws/iotwireless/events/device_registration_state/{eventType}/sidewalk/wireless_devices`
	• 标识符级主题
	`$aws/iotwireless/events/device_registration_state/`
		• `{eventType}` 可以是 `registered` 或 `provisioned`
		• `{resourceType}` 可以是 `sidewalk_accounts` 或 `wireless_devices`
		• `{resourceID}` 对于 `sidewalk_accounts` 为 `amazon_id`，对于 `wireless_devices` 为 `wireless_device_id`

事件	MQTT 主题	注意
	`{eventType}/sidewalk/{resourceType}/{resourceID}/{id}`	

Sidewalk 近似	• 资源级主题
	`$aws/iotwireless/events/proximity/{eventType}/sidewalk/wireless_devices`
	• 标识符级主题
	`$aws/iotwireless/events/proximity/{eventType}/sidewalk/{resourceType}/{resourceID}/{id}`
		• `{eventType}` 可以是 `beacon_discovered` 或 `beacon_lost`
		• `{resourceType}` 可以是 `sidewalk_accounts` 或 `wireless_devices`
		• `{resourceID}` 对于 `sidewalk_accounts` 为 `amazon_id`，对于 `wireless_devices` 为 `wireless_device_id`

Sidewalk 消息传输状态	• 资源级主题
	`$aws/iotwireless/events/message_delivery_status/{eventType}/sidewalk/wireless_devices`
	• 标识符级主题
	`$aws/iotwireless/events/message_delivery_status/{eventType}/sidewalk/{resourceType}/{resourceID}/{id}`
		• `{eventType}` 可以是 `success` 或 `error`
		• `{resourceType}` 可以是 `sidewalk_accounts` 或 `wireless_devices`
		• `{resourceID}` 对于 `sidewalk_accounts` 为 `amazon_id`，对于 `wireless_devices` 为 `wireless_device_id`

LoRaWAN 加入	• 资源级主题
	`$aws/iotwireless/events/join/{eventType}/lorawan/wireless_devices`
	• 标识符级主题
	`$aws/iotwireless/events/join/{eventType}/lorawan/wireless_devices/{resourceID}/{id}`
		• `{eventType}` 可以是 `join_req_0_received` 或 `join_req_2_received` 或 `join_accepted`
		• `{resourceID}` 可以是 `wireless_device_id` 或 `dev_eui`

事件	MQTT 主题	注意

LoRaWAN 网关连接状态	• 资源级主题
	`$aws/iotwireless/events/join/{eventType}/lorawan/wireless_gateways`
	• 标识符级主题
	`$aws/iotwireless/events/join/{eventType}/lorawan/wireless_gateways/{resourceID}/{id}`
		• `{eventType}` 可以是 `connected` 或 `disconnected`
		• `{resourceID}` 可以是 `wireless_gateway_id` 或 `gateway_eui`

有关不同事件的更多信息，请参阅 LoRaWAN 资源的事件通知 (p. 1272) 和 Sidewalk 资源的事件通知 (p. 1276)。

如果您订阅了这些主题，当消息发布到其中一个事件通知主题时，您将收到通知。有关更多信息，请参阅保留的主题 (p. 96)：

无线事件的定价

有关订阅事件和接收通知的定价的信息，请参阅 Amazon IoT Core 定价。

启用无线资源的事件

在保留主题的订阅者能够接收消息之前，您必须启用事件通知。要做到这一点，您可以使用 Amazon Web Services Management Console，或者 Amazon IoT Wireless API 或 Amazon CLI。

事件配置

您可以将事件配置为向属于特定类型的所有资源或针对单独的无线资源发送通知。资源类型可以是无线网关、Sidewalk 合作伙伴账户或无线设备（可以是 LoRa WAN 或 Sidewalk 设备）。有关可以为无线设备启用的事件类型的信息，请参阅 LoRaWAN 资源的事件类型 (p. 1272) 和 Sidewalk 资源的事件类型 (p. 1276)。

所有资源

您可以启用事件，以便您的 Amazon Web Services 账户 中属于特定资源类型的所有资源收到通知。例如，您可以启用一个事件，通知您引导到的所有 LoRa WAN 网关的连接状态所发生的更改Amazon IoT Core for LoRaWAN。监控这些事件将帮助您在如下情况下获得通知：资源实例集中的某些 LoRa WAN 网关断开连接或者您中的一些 Sidewalk 设备的信标丢失Amazon Web Services 账户。

单独的资源

您还可以将单独的 LoRa WAN 和 Sidewalk 资源添加到事件配置中，并为它们启用通知。这将帮助您监控特定类型的单独资源。例如，您可以将选定的 LoRa WAN 和 Sidewalk 设备添加到配置中，并接收这些资源的加入或设备注册状态事件的通知。

先决条件

您的 LoRa WAN 或 Sidewalk 资源必须具有允许其接收事件通知的适当策略。有关更多信息，请参阅用于接收无线事件通知的策略 (p. 1266)：

使用 Amazon Web Services Management Console 启用通知

要从控制台启用事件消息，请转到控制台的 Settings（设置）选项卡，然后转至 LoRaWAN and Sidewalk 事件通知部分。Amazon IoT

您可以为您的 Amazon Web Services 账户 中属于特定资源类型的所有资源启用通知，并对其进行监视。

针对所有资源启用通知

1. 在 LoRaWAN 和 Sidewalk 事件通知部分中，转到所有资源选项卡，选择操作，然后选择管理事件。
2. 启用要监控的事件，然后选择 Update events（更新事件）。如果您不再要监视某些事件，请依次选择 Action（操作）和 Manage events（管理事件），然后禁用这些事件。

您还可以为您的 Amazon Web Services 账户 中属于特定资源类型的单独资源启用通知，并对其进行监视。

针对单独的资源启用通知

1. 在 LoRaWAN 和 Sidewalk 事件通知部分中，选择操作，然后选择添加资源。
2. 选择要接收通知的资源和事件：
   a. 选择是否要监视 LoRaWAN 资源或 Sidewalk 资源或 Sidewalk 资源的事件。
   b. 根据资源类型，您可以选择要为资源启用的事件。然后，您可以订阅这些事件并接收通知。如果您选择...
      - LoRaWAN 资源：您可以为 LoRa WAN 设备启用加入事件或为 WA LoRa N 网关启用连接状态事件。
      - Sidewalk resources（Sidewalk 资源）：您可以为您的 Sidewalk 合作伙伴账户和 Sidewalk 设备启用 device registration state（设备注册状态）和/或 proximity（邻近）事件。

        Note

        Sidewalk 消息传输状态事件配置在控制台中不可用。只能使用 Amazon IoT Wireless API 或 Amazon CLI 启用此功能。
3. 根据您选择的资源类型和事件，选择要监控的无线设备或网关。对于所有组合的资源，您可以选择最多 250 个资源。
4. 选择 Submit（提交）以添加您的资源。

您添加的资源将与它们的 MQQTT 主题一起出现在控制台的 LoRaWAN and Sidewalk 事件通知部分中对应于资源类型的选项卡中。

- LoRaWAN join（Loin 加入）事件和 Sidewalk 设备的事件将出现在控制台的 Wireless 设备部分。
- LoRaWAN 网关的@@ 连接状态事件将出现在无线网关的连接状态事件将出现在无线网关的连接状态事件中。
- Sidewalk 账户的 Device registration state（设备注册状态）和 proximity（邻近）事件将出现在 Sidewalk accounts（Sidewalk 账户）选项卡中。

使用 MQTT 客户端订阅主题

根据您是针对所有资源还是针对单独资源类型启用了事件，启用的事件将显示在控制台中，其 MQTT 主题将显示在 All resources（所有资源）选项卡或指定资源类型的选项卡中。

- 如果您选择其中一个 MQTT 主题，您可以前往 MQTT 客户端订阅这些主题并接收消息。

- 如果您添加了多个事件，则可以订阅多个事件主题并接收有关它们的通知。要订阅多个主题，请选择主题，然后依次选择 Action（操作）和 Subscribe（订阅）。

使用 Amazon CLI 启用通知

您可以使用 Amazon IoT Wireless API 或 Amazon CLI 配置事件并将资源添加到配置中。

针对所有资源启用通知

您可以为您的中属于特定资源类型的所有资源启用通知Amazon Web Services 账户，并使用 UpdateEventConfigurationByResourceTypesAPI 或 update-event-configuration-by-resource-typesCLI 命令对其进行监视。例如：

```
aws iotwireless update-event-configuration-by-resource-types \
   --cli-input-json input.json
```

input.json 的内容

```
{
   "DeviceRegistrationState": {
      "Sidewalk": {
         "AmazonIdEventTopic": "Enabled"
      }
   },
   "ConnectionStatus": {
      "LoRaWAN": {
         "WirelessGatewayEventTopic": "Enabled"
      }
   }
}
```

> **Note**
>
> 使用反斜杠 (\) 转义所有双引号 (")。

您可以通过调用 GetEventConfigurationByResourceTypesAPI 或使用 get-event-configuration-by-resource-typesCLI 命令来获取当前事件配置。例如：

```
aws iotwireless get-event-configuration-by-resource-types
```

针对单独的资源启用通知

要将单独的资源添加到事件配置中并控制使用 API 或 CLI 发布的事件，调用 API 或使用 API 或使用 UpdateResourceEventConfigurationAPI 或使用 AP update-resource-event-configurationI 命令。例如：

```
aws iotwireless update-resource-event-configuration \
   --identifer 1ffd32c8-8130-4194-96df-622f072a315f \
   --identifier-type WirelessDeviceId \
   --cli-input-json input.json
```

input.json 的内容

```
{
   "Join": {
      "LoRaWAN": {
         "DevEuiEventTopic": "Disabled"
      },
```

```
      "WirelessDeviceIdEventTopic": "Enabled"
   }
}
```

Note

使用反斜杠 (\) 转义所有双引号 (")。

您可以通过调用 GetResourceEventConfigurationAPI 或使用 `get-resource-event-configuration`CLI 命令来获取当前事件配置。例如：

```
aws iotwireless get-resource-event-configuration \
    --identifier-type WirelessDeviceId \
    --identifier 1ffd32c8-8130-4194-96df-622f072a315f
```

列出事件配置

您也可以使用 Amazon IoT Wireless API 或 Amazon CLI 列出至少启用了一个事件主题的事件配置。要列出配置，请使用 ListEventConfigurationsAPI 操作或使用 `list-event-configurations`CLI 命令。例如：

```
aws iotwireless list-event-configurations --resource-type WirelessDevice
```

LoRaWAN 资源的事件通知

您可以使用Amazon Web Services Management Console或Amazon IoT Wireless API 操作通知您 LoRa WAN 设备和网关的事件。有关事件通知以及如何启用它们的信息，请参阅 Amazon IoT Wireless 的事件通知 (p. 1266)和 启用无线资源的事件 (p. 1269)。

LoRaWAN 资源的事件类型

您可以为 LoRa WAN 资源启用的事件包括：

- 向您通知 LoRa WAN 设备加入事件的加入事件。当设备加入 Amazon IoT Core for LoRaWAN 时，或者收到类型为 0 或 2 的重新加入请求时，您将收到通知。
- 当 LoRa WAN 网关的连接状态更改为已连接或断开连接时向您发出通知的连接状态事件。

以下部分包含有关 LoRa WAN 资源的事件的更多信息：

主题

- LoRaWAN 加入事件 (p. 1272)
- 连接状态事件 (p. 1274)

LoRaWAN 加入事件

Amazon IoT Core for LoRaWAN可以发布消息，通知您引导到上的 LoRa WAN 设备的加入事件Amazon IoT。当收到类型为 0 或 2 的加入或重新加入请求并且设备已加入 Amazon IoT Core for LoRaWAN 时，加入事件会通知您。

加入事件的工作原理

当您将 LoRa WAN 设备引导到时Amazon IoT Core for LoRaWAN，Amazon IoT Core for LoRaWAN执行将您的设备加入过程Amazon IoT Core for LoRaWAN。然后，您的设备将被激活以供使用，并可以发送上

行消息以指示其可用。设备加入后，可以在设备与 Amazon IoT Core for LoRaWAN 之间交换上行链路和下行链路消息。有关搭载您的设备的更多消息，请参阅 将您的设备搭载到 Amazon IoT Core for LoRaWAN 上 (p. 1119)。

您可以启用事件以在设备加入 Amazon IoT Core for LoRaWAN 时通知您。如果加入事件失败，收到类型为 0 或 2 的重新加入请求以及加入事件被接受时，您也会收到通知。

启用 LoRa WAN 加入活动

在 LoRa WAN 加入保留主题的订阅方可以接收消息之前，必须从或使用 API 或 CLI 为订阅方启用事件通知。Amazon Web Services Management Console您可以为中的所有 LoRa WAN 资源Amazon Web Services 账户或所选资源启用这些事件。有关如何启用这些事件的更多信息，请参阅 启用无线资源的事件 (p. 1269)。

LoRaWAN 事件的 MQQTT 主题格式

LoRaWAN 设备的保留 MQQTT 主题使用以下格式。如果您已订阅了这些主题，那么注册到您的的所有 LoRa WAN 设备Amazon Web Services 账户都可以接收通知：

- 资源级主题

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/lorawan/wireless_devices
  ```
- 标识符主题

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/lorawan/wireless_devices/
  {resourceID}/{id}
  ```

其中：

{eventName}

{eventName} 必须是 `join`。

{eventType}

{eventType} 可以是：

- `join_req_received`
- `rejoin_req_0_received`
- `rejoin_req_2_received`
- `join_accepted`

{resourceID}

{resourceID} 可以是 `dev_eui` 或 `wireless_device_id`。

例如，您可以订阅以下主题，以便在 Amazon IoT Core for LoRaWAN 已接受来自设备的加入请求时接收事件通知。

```
$aws/iotwireless/events/join/join_accepted/lorawan/wireless_devices/
wireless_device_id/{id}
```

您还可以使用 + 通配符同时订阅多个主题。+ 通配符匹配级别中包含字符的任何字符串，例如以下主题：

```
$aws/iotwireless/events/join/join_req_received/lorawan/wireless_devices/
wireless_device_id/+
```

> **Note**
>
> 您不能使用通配符 # 订阅保留主题。有关主题筛选条件的更多信息，请参阅 主题筛选条件 (p. 95)。

有关订阅主题时使用 + 通配符的更多信息，请参阅主题筛选条件 (p. 95)。

LoRaWAN 加入事件的消息有效负载

下面显示了 LoRa WAN 加入事件的消息有效负载。

```
{
 // General fields
    "eventId": "string",
    "eventType": "join_req_received|rejoin_req_0_received|rejoin_req_2_received|
join_accepted",
    "WirelessDeviceId": "string",
    "timestamp": "timestamp",

 // Event-specific fields
    "LoRaWAN": {
        "DevEui": "string",

        // The fields below are optional indicating that it can be a null value.
        "DevAddr": "string",
        "JoinEui": "string",
        "AppEui": "string",
    }
}
```

负载包含以下属性：

eventId

Amazon IoT Core for LoRaWAN 生成的唯一事件 ID（字符串）。

eventType

发生的事件类型。可以是以下任一值：

- `join_req_received`：此字段将显示 EUI 参数 `JoinEui` 或 `AppEui`
- `rejoin_req_0_received`
- `rejoin_req_2_received`
- `join_accepted`：此字段将显示 `NetId` 和 `DevAddr`。

wirelessDeviceId

LoRa广域网设备的 ID。

timestamp

事件发生时的 Unix 时间戳。

DevEui

在设备标注或设备文档中找到的设备的唯一标识符。

DevAddr 和 EUI（可选）

这些字段是可选的设备地址和 EUI 参数 `JoinEUI` 或 `AppEUI`。

连接状态事件

Amazon IoT Core for LoRaWAN可以发布消息，通知您引导到上的 LoRa WAN 网关的连接状态事件Amazon IoT。当 LoRa WAN 网关的连接状态更改为已连接或断开连接时，连接状态事件会向您发出通知。

连接状态事件的工作原理

在将网关引导到 Amazon IoT Core for LoRaWAN 之后，您可以将网关连接到 Amazon IoT Core for LoRaWAN 并验证其连接状态。当网关连接状态更改为已连接或断开连接时，此事件会通知您。有关将您的网关引导并连接到 Amazon IoT Core for LoRaWAN 的更多信息，请参阅将您的网关搭载到 Amazon IoT Core for LoRaWAN (p. 1112)和Connect 您的 LoRa WAN 网关并验证其连接状态 (p. 1118)。

LoRaWAN 网关的 MQTT 主题格式

LoRaWAN 网关的保留 MQQTT 主题使用以下格式。如果您已订阅了这些主题，那么注册到您的的所有 LoRa WAN 网关Amazon Web Services 账户都可以接收通知：

- 对于资源级主题：

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/lorawan/wireless_gateways
  ```
- 对于标识符主题：

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/lorawan/wireless_gateways/
  {resourceID}/{id}
  ```

其中：

{eventName}

{eventName} 必须是 `connection_status`。

{eventType}

{eventType} 可以是 `connected` 或 `disconnected`。

{resourceID}

{resourceID} 可以是 `gateway_eui` 或 `wireless_gateway_id`。

例如，您可以订阅以下主题以便在所有网关连接到 Amazon IoT Core for LoRaWAN 时接收事件通知：

```
$aws/iotwireless/events/connection_status/connected/lorawan/wireless_gateways/
wireless_gateway_id/{id}
```

您还可以使用 + 通配符同时订阅多个主题。+ 通配符匹配级别中包含字符的任何字符串，例如以下主题：

```
$aws/iotwireless/events/connection_status/connected/lorawan/wireless_gateways/
wireless_gateway_id/+
```

Note

您不能使用通配符 # 订阅保留主题。有关主题筛选条件的更多信息，请参阅 主题筛选条件 (p. 95)。

有关订阅主题时使用 + 通配符的更多信息，请参阅主题筛选条件 (p. 95)。

连接状态事件的消息有效负载

下面显示了连接状态事件的消息有效负载。

```
{
 // General fields
    "eventId": "string",
    "eventType": "connected|disconnected",
```

```
    "WirelessGatewayId": "string",
    "timestamp": "timestamp",

 // Event-specific fields
    "LoRaWAN": {
        "GatewayEui": "string"
    }
}
```

负载包含以下属性：

eventId

Amazon IoT Core for LoRaWAN 生成的唯一事件 ID（字符串）。

eventType

发生的事件类型。可以是 `connected` 或 `disconnected`。

wirelessGatewayId

LoRa广域网网关的 ID。

timestamp

事件发生时的 Unix 时间戳。

GatewayEui

网关标签或网关文档中找到的网关的唯一标识符。

Sidewalk 资源的事件通知

您可以使用 Amazon Web Services Management Console 或 Amazon IoT Wireless API 操作就 Sidewalk 设备和合作伙伴账户的事件向您发出通知。有关事件通知以及如何启用它们的信息，请参阅 Amazon IoT Wireless 的事件通知 (p. 1266) 和 启用无线资源的事件 (p. 1269)。

Sidewalk 资源的事件类型

您可以为 Sidewalk 资源启用的事件包括：

- 通知您 Sidewalk 设备状态更改的设备事件，例如设备已注册并准备使用的时间。
- 接近事件，当 Amazon IoT Wireless 收到来自 Amazon Sidewalk 的已收到或丢失信标的通知时通知您。

以下各部分包含有关 Sidewalk 资源的事件的更多信息：

主题

- 设备注册状态事件 (p. 1276)
- 近似事件 (p. 1278)
- 消息传输状态事件 (p. 1280)

设备注册状态事件

设备注册状态事件在设备注册状态发生更改时发布事件通知，例如在 Sidewalk 设备已设置或注册时。这些事件为您提供有关设备从预调配到注册的不同状态的信息。

设备注册状态事件的工作原理

当您使用 Amazon Sidewalk 和 Amazon IoT Wireless 引导 Sidewalk 设备时，Amazon IoT Wireless 将执行一项 `create` 操作并将 Sidewalk 设备添加到 Amazon Web Services 账户 中。然后，您的设备进入预调配状态，然后 `eventType`变成 `provisioned`。有关搭载您的设备的更多消息，请参阅 亚马逊 SidAmazon IoT Core ewalk 入门 (p. 1206)。

对设备进行 `provisioned` 后，Amazon Sidewalk 将执行 `register` 操作，向 Amazon IoT Wireless 注册您的 Sidewalk 设备。注册过程开始，使用 Amazon IoT 设置加密和会话密钥。注册设备后，`eventType` 变成 `registered`，并且您的设备已能够使用。

对设备进行 `registered` 后，Sidewalk 可以发送请求以 `deregister` 您的设备。然后，Amazon IoT Wireless 执行请求并将设备状态更改回到 `provisioned`。有关设备状态的更多信息，请参阅DeviceState。

启用设备注册状态事件的通知

在设备注册状态保留主题的订阅方可以接收消息之前，必须从 Amazon Web Services Management Console 或使用 API 或 CLI 为其启用事件通知。您可以为 Amazon Web Services 账户 中的所有 Sidewalk 资源或所选资源启用这些事件。有关如何启用这些事件的更多信息，请参阅 启用无线资源的事件 (p. 1269)。

设备注册状态事件的 MQTT 主题格式

要通知设备注册状态事件，您可以订阅以美元 ($) 符号开头的 MQTT 保留主题。有关更多信息，请参阅MQTT 主题 (p. 94)：

Sidewalk 设备注册状态事件的保留 MQTT 主题使用以下格式：

- 对于资源级主题：

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/sidewalk/wireless_devices
  ```
- 对于标识符主题：

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/sidewalk/{resourceType}/
  {resourceID}/{id}
  ```

其中：

{eventName}

{eventName} 必须是 `device_registation_state`。

{eventType}

{eventType} 可以是 `provisioned` 或 `registered`。

{resourceType}

{resourceType} 可以是 `sidewalk_accounts` 或 `wireless_devices`。

{resourceID}

{resourceID} 对于 {resourceType} `sidewalk_accounts` 为 `amazon_id`，对于 {resourceType} `wireless_devices` 为 `wireless_device_id`。

您还可以使用 + 通配符同时订阅多个主题。+ 通配符匹配级别中包含字符的任何字符串。例如，如果您希望收到所有可能的事件类型（`provisioned` 和 `registered`）的通知以及注册到特定 Amazon ID 的所有设备通知，可以使用以下主题筛选条件：

```
$aws/iotwireless/events/device_registration_state/+/sidewalk/sidewalk_accounts/
amazon_id/+
```

Note

您不能使用通配符 # 订阅保留主题。有关主题筛选条件的更多信息，请参阅 主题筛选条件 (p. 95)。

设备注册状态事件的消息负载

启用设备注册状态事件的通知后，事件通知将通过 MQTT 随 JSON 负载一起发布。事件包含以下负载示例：

```
{
    "eventId": "string",
    "eventType": "provisioned|registered",
    "WirelessDeviceId": "string",
    "timestamp": "timestamp",

    // Event-specific fields
    "operation": "create|deregister|register",
    "Sidewalk": {
        "AmazonId": "string",
        "SidewalkManufacturingSn": "string"
    }
}
```

负载包含以下属性：

eventId

唯一事件 ID (字符串)。

eventType

发生的事件类型。可以是 `provisioned` 或 `registered`。

wirelessDeviceId

无线设备的标识符。

timestamp

事件发生时的 Unix 时间戳。

操作

触发事件的操作。有效值包括 `create`、`register` 和 `deregister`。

sidewalk

您要接收其事件通知的 Sidewalk Amazon ID 或 `SidewalkManufacturingSn`。

近似事件

Amazon IoT 从 Sidewalk 设备接收信标时近似事件发布事件通知。当您的 Sidewalk 设备接近 Amazon Sidewalk 时，Amazon Sidewalk 会按定期间隔筛选从您的设备发送的信标，并通过 Amazon IoT Wireless 接收信标。然后，在收到信标时，Amazon IoT Wireless 通知您这些事件。

近似事件的工作原理

近似事件会在 Amazon IoT 收到信标时通知您，您的 Sidewalk 设备可以随时发射信标。当您的设备靠近 Amazon Sidewalk 时，Sidewalk 将接收信标，并按定期时间间隔将信标转发给 Amazon IoT Wireless。Amazon Sidewalk 已将此时间间隔配置为 10 分钟。当 Amazon IoT Wireless 从 Sidewalk 接收到信标时，您将收到事件通知。

近似事件将在发现信标或信标丢失时通知您。您可以配置近似事件通知的时间间隔。

启用近似事件的通知

在 Sidewalk 近似保留主题的订阅方可以接收消息之前，必须从 Amazon Web Services Management Console 或使用 API 或 CLI 为订阅方启用事件通知。您可以为 Amazon Web Services 账户 中的所有 Sidewalk 资源或所选资源启用这些事件。有关如何启用这些事件的更多信息，请参阅 启用无线资源的事件 (p. 1269)。

近似事件的 MQTT 主题的格式

要通知近似事件，您可以订阅以美元 ($) 符号开头的 MQTT 保留主题。有关更多信息，请参阅MQTT 主题 (p. 94)：

Sidewalk 近似事件的保留 MQTT 主题使用以下格式：

- 对于资源级主题：

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/sidewalk/wireless_devices
  ```

- 对于标识符主题：

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/sidewalk/{resourceType}/
  {resourceID}/{id}
  ```

其中：

{eventName}

{eventName} 必须是 `proximity`。

{eventType}

{eventType} 可以是 `beacon_discovered` 或 `beacon_lost`。

{resourceType}

{resourceType} 可以是 `sidewalk_accounts` 或 `wireless_devices`。

{resourceID}

{resourceID} 对于 {resourceType} `sidewalk_accounts` 为 `amazon_id`，对于 {resourceType} `wireless_devices` 为 `wireless_device_id`。

您还可以使用 + 通配符同时订阅多个主题。+ 通配符匹配级别中包含字符的任何字符串。例如，如果您希望收到所有可能的事件类型（`beacon_discovered` 和 `beacon_lost`）的通知以及注册到特定 Amazon ID 的所有设备通知，可以使用以下主题筛选条件：

```
$aws/iotwireless/events/proximity/+/sidewalk/sidewalk_accounts/amazon_id/+
```

> **Note**
>
> 您不能使用通配符 # 订阅保留主题。有关主题筛选条件的更多信息，请参阅 主题筛选条件 (p. 95)。

近似事件的消息负载

启用近似事件通知后，将通过 MQTT 随 JSON 负载一起发布事件消息。事件包含以下负载示例：

```
{
    "eventId": "string",
```

```
    "eventType": "beacon_discovered|beacon_lost",
    "WirelessDeviceId": "string",
    "timestamp": "1234567890123",

    // Event-specific fields
    "Sidewalk": {
        "AmazonId": "string",
        "SidewalkManufacturingSn": "string"
    }
}
```

负载包含以下属性：

eventId

唯一的事件 ID，字符串。

eventType

发生的事件类型。可以是 `beacon_discovered` 或 `beacon_lost`。

WirelessDeviceId

无线设备的标识符。

timestamp

事件发生时的 Unix 时间戳。

sidewalk

您要接收其事件通知的 Sidewalk Amazon ID 或 `SidewalkManufacturingSn`。

消息传输状态事件

消息传输状态事件发布有关在 Sidewalk 设备和 Amazon IoT Wireless 之间交换的消息状态的事件通知。将针对从 Amazon IoT Wireless 发送到 Sidewalk 设备的下行链路消息和从您的设备发送到 Amazon IoT Wireless 的上行链路消息发布事件通知。

消息传输状态事件的工作原理

在您将 Sidewalk 设备登入 Amazon IoT Wireless 并连接您的设备后，可以在您的设备和 Amazon IoT Wireless 之间交换消息。这些事件发布有关消息传输状态的通知，指示这些消息是否已成功传输到您的设备或 Amazon IoT Wireless。

例如，如果从设备收到带有确认（ACK）标志的上行链路消息，则会发布一条通知，指示消息已成功传输。当您将下行链路消息从 Amazon IoT Wireless 发送到 Sidewalk 设备时，即使数据包丢失或消息未传递，`SendDataToWirelessDevice` API 也会为下行链路返回 `MessageId`。在这种情况下，消息传输状态事件会返回一个错误，指示消息未能传输到设备。

为消息传输状态事件启用通知

在 Sidewalk 消息传输状态保留主题的订阅方可以接收消息之前，必须使用 Amazon IoT Wireless API 或 Amazon CLI 为订阅方启用事件通知。您可以为 Amazon Web Services 账户 中的所有 Sidewalk 资源或所选资源启用这些事件。

Note

Sidewalk 消息传输状态事件配置在控制台中不可用。

有关如何启用这些事件的更多信息，请参阅 使用 Amazon CLI 启用通知 (p. 1271)。

用于消息传输状态事件的 MQTT 主题的格式

要接收有关消息传输状态事件的通知，您可以订阅以美元（$）符号开头的 MQTT 保留主题。有关更多信息，请参阅MQTT 主题 (p. 94)：

Sidewalk 近似事件的保留 MQTT 主题使用以下格式：

- 对于资源级主题：

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/sidewalk/wireless_devices
  ```
- 对于标识符主题：

  ```
  $aws/iotwireless/events/{eventName}/{eventType}/sidewalk/{resourceType}/
  {resourceID}/{id}
  ```

其中：

{eventName}

{eventName} 必须是 `message_delivery_status`。

{eventType}

{eventType} 可以是 `success` 或 `error`。

{resourceType}

{resourceType} 可以是 `sidewalk_accounts` 或 `wireless_devices`。

{resourceID}

{resourceID} 对于 {resourceType} `sidewalk_accounts` 为 `amazon_id`，对于 {resourceType} `wireless_devices` 为 `wireless_device_id`。

您还可以使用 + 通配符同时订阅多个主题。+ 通配符匹配级别中包含字符的任何字符串。例如，如果您希望收到所有可能的事件类型（`success` 和 `error`）的通知以及注册到特定 Amazon ID 的所有设备通知，可以使用以下主题筛选条件：

```
$aws/iotwireless/events/message_delivery_status/+/sidewalk/sidewalk_accounts/
amazon_id/+
```

Note

您不能使用通配符 # 订阅保留主题。有关主题筛选条件的更多信息，请参阅 主题筛选条件 (p. 95)。

消息传输状态事件的消息有效负载

为消息传输状态事件启用通知后，将通过 MQTT 随 JSON 有效负载一起发布事件消息。这些事件包含以下示例有效负载，具体取决于事件是否成功，表示设备已成功接收消息或出现错误。

成功事件

下图显示了事件为成功时的有效负载格式。

```
{
    "eventId": "string",
    "eventType": "success",
    "WirelessDeviceId": "string",
    "timestamp": "timestamp",
    "Sidewalk": {
```

```
        "Seq": "Integer",
        "MsgType": "CUSTOM_COMMAND_ID_RESP",
        "CmdExStatus": "COMMAND_EXEC_STATUS_SUCCESS"
    }
}
```

负载包含以下属性：

eventId

唯一的事件 ID，字符串。

eventType

发生的事件类型。可以是 `success` 或 `error`。在本例中，`eventType` 为 `error`。

WirelessDeviceId

无线设备的标识符。

timestamp

事件发生时的 Unix 时间戳。

sidewalk

包含成功消息的状态代码、消息的序列号和消息类型的 Sidewalk 包装程序。

错误事件

下图显示了事件指示发生错误时的有效负载格式。

```
{
    "eventId": "string",
    "eventType": "error" ,
    "WirelessDeviceId": "string",
    "timestamp": "timestamp",
    "Sidewalk": {
        "Seq": "Integer",
        "Status": "DeviceNotReachable" | "RADIO_TX_ERROR" | "MEMORY_ERROR"
    }
}
```

有效负载包含与 `eventType` 为 `success` 时类似的属性。以下是一些差别或附加属性：

eventType

发生的事件类型。在本例中，`eventType` 为 `error`。

sidewalk

包含序列号和状态代码的 Sidewalk 包装程序，状态代码指示下行链路消息未成功发送的原因。

Amazon IoT Device SDK、Mobile SDK 和 Amazon IoT Device Client

本页总结了 Amazon IoT Device SDK、开源库、开发人员指南、示例应用程序和移植指南的内容，可帮助您使用 Amazon IoT 和您选择的硬件平台构建创新的 IoT 解决方案。

这些 SDK 供您在 IoT 设备上使用。如果您正在开发用于移动设备的 IoT 应用，请参阅 AmazonMobile SDK (p. 1285)。如果您正在开发 IoT 应用程序或服务器端程序，请参阅 Amazon SDK (p. 67)。

Amazon IoT Device SDK

Amazon IoT Device SDK 包括开源库、开发人员指南（含示例）和移植指南，便于您在自己选择的硬件平台上构建富有创新精神的 IoT 产品或解决方案。

这些 SDK 可帮助您使用 MQTT 和 WSS 协议将 IoT 设备连接到 Amazon IoT。

C++

Amazon IoT C++ 设备 SDK

Amazon IoT C++ Device SDK 使得开发人员可以使用Amazon和 Amazon IoT API 构建互连应用程序。特别是，此 SDK 面向没有资源限制且需要高级功能（例如，消息队列、多线程支持和最新的语言功能）的设备而设计。有关更多信息，请参阅下列内容：

- GitHub 上的 Amazon IoT C++ v2 设备 SDK
- Amazon IoT C++ v2 设备 SDK 自述文件
- Amazon IoT C++ v2 设备 SDK 示例
- Amazon IoT C++ v2 设备 SDK API 文档

Python

适用于 Python 的 Amazon IoT 设备 SDK

借助 Amazon IoT Device SDK for Python，开发人员可以编写 Python 脚本，以使用其设备通过 MQTT 或基于 WebSocket 的 MQTT 协议来访问 Amazon IoT 平台。通过将设备连接到 Amazon IoT，用户可以安全地使用由 Amazon IoT 提供的消息代理、规则和影子，以及 Amazon Lambda、Kinesis 和 Amazon S3 等其它 Amazon 服务。

- GitHub 上适用于 Python v2 的 Amazon IoT 设备 SDK
- 适用于 Python 的 Amazon IoT 设备 SDK v2 自述文件
- 适用于 Python 的 Amazon IoT 设备 SDK v2 示例
- Amazon IoT Python v2 设备 SDK API 文档

JavaScript

适用于 JavaScript 的 Amazon IoT 设备 SDK

借助 aws-iot-device-sdk.js 软件包，开发人员可以编写 JavaScript 应用程序，这些应用程序可使用 MQTT 或基于 WebSocket 的 MQTT 协议来访问 Amazon IoT。它可用于 Node.js 环境和浏览器应用程序。有关更多信息，请参阅下列内容：

- GitHub 上适用于 JavaScript v2 的 Amazon IoT 设备 SDK
- 适用于 JavaScript 的 Amazon IoT 设备 SDK v2 自述文件
- 适用于 JavaScript 的 Amazon IoT 设备 SDK v2 示例
- 适用于 JavaScript 的 Amazon IoT 设备 SDK v2 API 文档

Java

适用于 Java 的 Amazon IoT 设备 SDK

借助 Amazon IoT Device SDK for Java，Java 开发人员能够通过 MQTT 或基于 WebSocket 协议的 MQTT 来访问 Amazon IoT 平台。该 SDK 内置有影子支持。您可以使用 HTTP 方法 (包括 GET、UPDATE 和 DELETE) 访问影子。该 SDK 还支持简化的影子访问模型，开发人员只需要使用 getter 和 setter 方法即可与影子交换数据，而不必对任何 JSON 文档进行序列化或反序列化。有关更多信息，请参阅下列内容：

- GitHub 上适用于 Java v2 的 Amazon IoT 设备 SDK
- 适用于 Java 的 Amazon IoT 设备 SDK v2 自述文件
- 适用于 Java 的 Amazon IoT 设备 SDK v2 示例
- Amazon IoT Java v2 设备 SDK API 文档

Amazon IoT Device SDK for Embedded C

Note

该 SDK 供经验丰富的嵌入式软件开发人员使用。

Amazon IoT Device SDK for Embedded C (C-SDK) 是 MIT 开源许可证下多个 C 源文件的集合，可在嵌入式应用程序中使用，以安全地将 IoT 设备连接到 Amazon IoT Core。其中包括 MQTT 客户端、JSON Parser、Amazon IoT Device Shadow、Amazon IoT Jobs、Amazon IoT Fleet Provisioning 和 Amazon IoT Device Defender 库。该开发工具包以源代码形式分发，可构建到客户固件和应用程序代码、其他库以及您选择的操作系统 (OS) 中。

Amazon IoT Device SDK for Embedded C通常面向需要优化的 C 语言运行时的资源受限设备。您可以在任何操作系统上使用此 SDK，并将其托管在任何类型的处理器（例如 MCU 和 MPU）上。

有关更多信息，请参阅下列内容：

- GitHub 上适用于嵌入式 C 的 Amazon IoT 设备 SDK
- 适用于嵌入式 C 的 Amazon IoT 设备 SDK 自述文件
- 适用于嵌入式 C 的 Amazon IoT 设备 SDK 示例

早期 Amazon IoT Device SDK 版本

这些早期版本的 Amazon IoT Device SDK 已被上面列出的较新版本替换。这些 SDK 仅接收维护和安全更新。它们不会更新以获取新功能，也不应用于新项目。

- GitHub 上的 Amazon IoT C++ 设备开发工具包
- Amazon IoT C++ 设备软件开发工具包自述文件
- GitHub 上适用于 Python v1 的 Amazon IoT 设备开发工具包
- 适用于 Python v1 的 Amazon IoT 设备开发工具包自述文件
- GitHub 上适用于 Java 的 Amazon IoT 设备软件开发工具包
- 适用于 Java 的 Amazon IoT 设备软件开发工具包自述文件

- GitHub 上适用于 JavaScript 的 Amazon IoT 设备软件开发工具包
- 适用于 JavaScript 的 Amazon IoT 设备软件开发工具包自述文件
- GitHub 上的 Arduino Yún 软件开发工具包
- Arduino Yún 软件开发工具包自述文件

AmazonMobile SDK

Amazon Mobile SDK 为移动应用程序开发人员提供特定于平台的支持，支持 Amazon IoT Core 服务的 API、使用 MQTT 的 IoT 设备通信以及其它 Amazon 服务的 API。

Android

Amazon Mobile SDK for Android

Amazon Mobile SDK for Android 包含为开发人员提供的库、示例和文档，以便于他们使用Amazon构建联网的移动应用程序。此 SDK 还包括对 MQTT 设备通信和调用 Amazon IoT Core 服务 API 的支持。有关更多信息，请参阅下列内容：

- GitHub 上的 Amazon Mobile SDK for Android
- Amazon Mobile SDK for Android 自述文件
- Amazon Mobile SDK for Android 示例
- Amazon Mobile SDK for Android API 参考
- AWSIoTClient 类参考文档

iOS

Amazon Mobile SDK for iOS

Amazon Mobile SDK for iOS 是开源的软件开发套件，依据 Apache 开源许可分发。Amazon Mobile SDK for iOS 为开发人员提供库、代码示例和文档，以便于他们使用Amazon构建联网的移动应用程序。此 SDK 还包括对 MQTT 设备通信和调用 Amazon IoT Core 服务 API 的支持。有关更多信息，请参阅下列内容：

- GitHub 上的 Amazon Mobile SDK for iOS
- Amazon Mobile SDK for iOS 自述文件
- Amazon Mobile SDK for iOS 示例
- AWSIoT 类参考文档位于Amazon Mobile SDK for iOS

Amazon IoT Device Client

Amazon IoT Device Client 提供代码以帮助您的设备连接到 Amazon IoT、执行机群预置任务、支持设备安全策略、使用安全隧道进行连接以及处理设备上的任务。您可以在设备上安装此软件来处理这些常规设备任务，以便专注于特定的解决方案。

Note

Amazon IoT Device Client 与基于微处理器且搭载 x86_64 或 ARM 处理器和常见 Linux 操作系统的 IoT 设备配合使用。

C++

Amazon IoT 设备客户端

有关以 C++ 编写的 Amazon IoT Device Client 的更多信息，请参阅以下内容：

- Amazon IoTGitHub 上以 C++ 源代码编写的 Device Client
- Amazon IoT以 C++ 编写的 Device Client 自述文件

排除 Amazon IoT 的故障

帮助我们改进此主题

告诉我们如何优化内容

以下信息可帮助您处理 Amazon IoT 中的常见问题。

任务

- 诊断连接问题 (p. 1287)
- 诊断规则问题 (p. 1289)
- 诊断影子问题 (p. 1291)
- 诊断 Salesforce IoT 输入流操作问题 (p. 1291)
- 机群索引排除指南 (p. 1292)
- 解决“您的Amazon账户已超过流限制”的问题 (p. 1294)
- Amazon IoT Device Defender 故障排除指南 (p. 1294)
- Amazon IoT Device Advisor 故障排除指南 (p. 1297)
- 排除设备机群断开连接的故障 (p. 1298)
- Amazon IoT 错误 (p. 1299)

诊断连接问题

帮助我们改进此主题

告诉我们如何优化内容

成功连接到 Amazon IoT 需要：

- 有效的连接
- 有效且已激活的证书
- 允许所需连接和操作的策略

连接

如何找到正确的终端节点？

- `aws iot describe-endpoint --endpoint-type iot:Data-ATS` 返回的 `endpointAddress`

  或

- `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"` 返回的 `domainName`

如何找到正确的 Server Name Indication (SNI) 值？

正确的 SNI 值是由 describe-endpoint 或者 describe-domain-configuration 命令返回的 `endpointAddress`。它是与上一步中的终端节点相同的地址。

我该如何解决持续存在的连接问题？

您可以使用 Amazon Device Advisor 帮助进行故障排除。Device Advisor 的预构建测试可帮助您根据最佳实践验证您的设备软件，以便使用 TLS、MQTT、Amazon IoT 设备影子和 Amazon IoT Jobs。

以下是现有 Device Advisor 内容的链接。

身份验证

设备必须经过身份验证 (p. 267)以连接到 Amazon IoT 终端节点。对于使用 X.509 客户端证书 (p. 267) 进行身份验证的设备，则必须将证书注册到 Amazon IoT 并使其处于活动状态。

我的设备如何对 Amazon IoT 终端节点进行身份验证？

将 Amazon IoT CA 证书添加到客户端的信任存储。请参阅 Amazon IoT Core 中的服务器身份验证上的文档，然后通过链接下载适当的 CA 证书。

当设备连接到 Amazon IoT 时需要检查什么？

当设备尝试连接到 Amazon IoT 时：

1. Amazon IoT 检查有效的证书和 Server Name Indication (SNI) 值。
2. Amazon IoT 检查所使用的证书是否已注册到 Amazon IoT 账户，并且是否已被激活。
3. 当设备尝试在 Amazon IoT 中执行任何操作时（例如订阅或发布消息），则会检查附加到用于连接的证书策略，以确认设备是否有权执行该操作。

如何验证证书的配置是否正确？

请使用 OpenSSL `s_client` 命令测试与 Amazon IoT 终端节点的连接：

```
openssl s_client -connect custom_endpoint.iot.aws-region.amazonaws.com:8443 -
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

有关如何使用 `openssl s_client` 的更多信息，请参阅 OpenSSL s_client 文档。

如何检查证书的状态？

- 列出证书

  如果您不知道该证书 ID，则可以使用 `aws iot list-certificates` 命令查看所有证书的状态。

- 显示证书的详细信息

  如果您知道证书的 ID，此命令将显示有关证书的更详细信息。

  ```
  aws iot describe-certificate --certificate-id "certificateId"
  ```

- 在 Amazon IoT 控制台中查看证书

  在 Amazon IoT控制台，在左侧菜单中，选择 Secure（安全），然后选择 Certificates（证书）。

  从列表中选择用于连接的证书以打开其详细信息页面。

  在证书的详细信息页面中，您可以查看其当前状态。

  证书的状态可以通过使用详细信息页面右上角的 Actions（操作）菜单来更改。

授权

Amazon IoT 资源使用 Amazon IoT Core 策略 (p. 299) 授权这些资源执行操作 (p. 300)。对于要授权的操作，指定的 Amazon IoT 资源必须附加一个策略文档，该文档将授予执行该操作的权限。

我收到了代理发送的 PUBNACK 或 SUBNACK 回复。我应该怎么办？

请确保当前有策略附加到您用于调用 Amazon IoT 的证书。预设情况下，所有的发布/订阅操作均将被拒绝。

确保附加的策略授权您尝试执行的操作 (p. 300)。

确保附加的策略授权正在尝试执行授权操作的资源 (p. 302)。

我的日志中有一个 AUTHORIZATION_FAILURE 的条目。

请确保当前有策略附加到您用于调用 Amazon IoT 的证书。预设情况下，所有的发布/订阅操作均将被拒绝。

确保附加的策略授权您尝试执行的操作 (p. 300)。

确保附加的策略授权正在尝试执行授权操作的资源 (p. 302)。

如何检查策略授权的内容？

在 Amazon IoT控制台，在左侧菜单中，选择 Secure（安全），然后选择 Certificates（证书）。

从列表中选择用于连接的证书以打开其详细信息页面。

在证书的详细信息页面中，您可以查看其当前状态。

在证书详细信息页面的左侧菜单中，选择 Policies（策略）查看附加到该证书的策略。

选择所需策略以查看其详细信息页面。

在策略的详细信息页面中，查看策略的策略文档以查看它授权的内容。

选择 Edit policy document（编辑策略文档）对策略文档进行更改。

安全和身份

当您为Amazon IoT自定义域配置提供服务器证书时，证书最多有四个域名。

有关更多信息，请参阅 Amazon IoT Core 终端节点和配额。

诊断规则问题

帮助我们改进此主题

告诉我们如何优化内容

本部分介绍了在遇到规则问题时要检查的一些事项。

配置 CloudWatch Logs 以进行故障排除

对规则所存在的问题进行调试的最佳方法是使用 CloudWatch Logs。为 Amazon IoT 启用 CloudWatch Logs 后，您可了解哪些规则被触发以及操作是成功还是失败。您还可以了解 WHERE 子句条件是否匹配。有关更多信息，请参阅Amazon IoT使用 CloudWatch 日志进行监控 (p. 412)。

最常见的规则问题是授权问题。可通过日志了解您的角色是否获得对资源执行 AssumeRole 操作所需的权限。下面是一个精细日志记录 (p. 398)功能生成的示例日志：

```
{
```

```
    "timestamp": "2017-12-09 22:49:17.954",
    "logLevel": "ERROR",
    "traceId": "ff563525-6469-506a-e141-78d40375fc4e",
    "accountId": "123456789012",
    "status": "Failure",
    "eventType": "RuleExecution",
    "clientId": "iotconsole-123456789012-3",
    "topicName": "test-topic",
    "ruleName": "rule1",
    "ruleAction": "DynamoAction",
    "resources": {
        "ItemHashKeyField": "id",
        "Table": "trashbin",
        "Operation": "Insert",
        "ItemHashKeyValue": "id",
        "IsPayloadJSON": "true"
    },
    "principalId": "ABCDEFG1234567ABCD890:outis",
    "details": "User: arn:aws:sts::123456789012:assumed-role/dynamo-testbin/5aUMInJH
 is not authorized to perform: dynamodb:PutItem on resource: arn:aws:dynamodb:us-
east-1:123456789012:table/testbin (Service: AmazonDynamoDBv2; Status Code: 400; Error Code:
 AccessDeniedException; Request ID: AKQJ987654321AKQJ123456789AKQJ987654321AKQJ987654321)"
}
```

下面是一个全局日志记录 (p. 397)功能生成的类似示例日志：

```
2017-12-09 22:49:17.954 TRACEID:ff562535-6964-506a-e141-78d40375fc4e
PRINCIPALID:ABCDEFG1234567ABCD890:outis [ERROR] EVENT:DynamoActionFailure
TOPICNAME:test-topic CLIENTID:iotconsole-123456789012-3
MESSAGE:Dynamo Insert record failed. The error received was User:
 arn:aws:sts::123456789012:assumed-role/dynamo-testbin/5aUMInJI is not authorized to
 perform: dynamodb:PutItem on resource: arn:aws:dynamodb:us-east-1:123456789012:table/
testbin
(Service: AmazonDynamoDBv2; Status Code: 400; Error Code: AccessDeniedException; Request
 ID: AKQJ987654321AKQJ987654321AKQJ987654321AKQJ987654321).
Message arrived on: test-topic, Action: dynamo, Table: trashbin, HashKeyField: id,
 HashKeyValue: id, RangeKeyField: None, RangeKeyValue: 123456789012
No newer events found at the moment. Retry.
```

有关更多信息，请参阅the section called “在 CloudWatch 控制台中查看Amazon IoT日志” (p. 413)。

诊断外部服务

外部服务由最终用户控制。在执行规则之前，请确保已设置链接到规则的外部服务，并且具有足够的应用程序吞吐量和容量单位。

诊断 SQL 问题

如果您的 SQL 查询没有返回您期望的数据：

- 查看日志中的错误消息。
- 确认您的 SQL 语法与消息中的 JSON 文档匹配。

  查看查询中使用的对象和属性名称，以及主题消息负载的 JSON 文档中使用的对象名和属性名称。有关 SQL 查询中 JSON 格式的更多信息，请参阅 JSON 扩展 (p. 583)。
- 检查 JSON 对象或属性名称是否包含预留字符或数字字符。

  有关 SQL 查询中 JSON 对象引用中预留字符的更多信息，请参阅 JSON 扩展 (p. 583)。

诊断影子问题

帮助我们改进此主题

告诉我们如何优化内容

诊断影子

问题	故障排除指南
设备的影子文档被拒绝，并显示 `Invalid JSON document`。	如果您不熟悉 JSON，请修改本指南中提供的示例，以供您自己使用。有关更多信息，请参阅影子文档示例 (p. 628)。
我提交了正确的 JSON，但设备的影子文档中没有存储其中的任何内容或仅存储了一部分。	请确保您遵循了以下 JSON 格式指南。仅存储 `desired` 和 `reported` 部分中的 JSON 字段。这些部分以外的 JSON 内容 (即使格式正确) 将被忽略。
我收到一条错误消息，称设备的影子超出了允许的大小。	设备的影子仅支持 8KB 数据。请尝试缩短您的 JSON 文档中的字段名称，也可以通过创建更多事物来创建更多影子。一个设备可拥有无限数量的事物/与之关联的影子。唯一的要求是，您账户中的每个事物名称都必须是唯一的。
我收到了一个超过 8KB 的设备的影子。这是怎么回事？	在您收到设备的影子之后，Amazon IoT 服务会向其添加元数据。该服务将此类数据添加到其响应中，但此类数据不会计入 8KB 的限制中。只有发送到设备的影子的状态文档中有关 `desired` 状态和 `reported` 状态的数据才会计入到此限制中。
我的请求由于版本错误而被拒绝了。我应该怎么办？	执行 GET 操作，以同步至最新的状态文档版本。使用 MQTT 时，请订阅 ./update/accepted 主题，这样您便会收到有关状态更改的通知以及最新版本的 JSON 文档。
时间戳关闭了几秒钟。	单个字段和整个 JSON 文档的时间戳会在 Amazon IoT 服务收到文档或状态文档发布到 ./update/accepted 和 ./update/delta 消息后更新。消息可能会由于网络发生延迟，从而导致时间戳关闭几秒钟。
我的设备可以在相应的影子主题下发布消息并订阅这些主题，但当我尝试通过 HTTP REST API 更新影子文档时，我收到了 HTTP 403 错误消息。	请确保您已在 IAM 中创建了相应的策略，能够访问这些主题并针对您使用的凭证执行相应的操作 (UPDATE/GET/DELETE)。IAM policy 和证书策略是相互独立的。
其它问题。	Device Shadow 服务会将错误记录到 CloudWatch Logs。要找出设备和配置问题，请启用 CloudWatch Logs 并查看日志，以获取调试信息。

诊断 Salesforce IoT 输入流操作问题

帮助我们改进此主题

告诉我们如何优化内容

执行跟踪

如何查看 Salesforce 操作的执行跟踪？

请参阅 Amazon IoT使用 CloudWatch 日志进行监控 (p. 412) 部分。激活日志后，您可以查看 Salesforce 操作的执行跟踪。

操作成功和失败

如何检查是否已成功将消息发送到 Salesforce IoT 输入流？

查看在 CloudWatch Logs 中执行 Salesforce 操作所生成的日志。如果您看到 `Action executed successfully`，则表示 Amazon IoT 规则引擎已收到来自 Salesforce IoT 的确认信息，表明消息已成功推送到目标输入流。

如果您在使用 Salesforce IoT 平台时遇到问题，请联系 Salesforce IoT 支持。

如果消息未成功发送到 Salesforce IoT 输入流，该怎么办？

查看在 CloudWatch Logs 中执行 Salesforce 操作所生成的日志。您可以尝试以下操作，具体取决于日志条目：

`Failed to locate the host`

检查此操作的 `url` 参数是否正确，以及您的 Salesforce IoT 输入流是否存在。

`Received Internal Server Error from Salesforce`

重试。如果问题仍然存在，请联系 Salesforce IoT 支持。

`Received Bad Request Exception from Salesforce`

检查您正在发送的负载是否存在错误。

`Received Unsupported Media Type Exception from Salesforce`

Salesforce IoT 目前不支持二进制负载。检查您是否正在发送 JSON 负载。

`Received Unauthorized Exception from Salesforce`

检查此操作的 `token` 参数是否正确，以及您的令牌是否仍有效。

`Received Not Found Exception from Salesforce`

检查此操作的 `url` 参数是否正确，以及您的 Salesforce IoT 输入流是否存在。

如果您收到此处未列出的错误，请联系 Amazon IoT Support。

机群索引排除指南

帮助我们改进此主题

告诉我们如何优化内容

队列索引服务的聚合查询疑难解答

如果您遇到类型不匹配错误，则可以使用 CloudWatch Logs 解决问题。Fleet Indexing 服务写入日志之前，必须启用 CloudWatch Logs。有关更多信息，请参阅Amazon IoT使用 CloudWatch 日志进行监控 (p. 412)。

在非托管字段上进行聚合查询时，只能指定您在传递给 UpdateIndexingConfiguration 或 update-indexing-configuration 的 customFields 参数中定义的字段。如果字段值与配置的字段数据类型不一致，则在执行聚合查询时忽略此值。

当字段由于类型不匹配而无法建立索引时，Fleet Indexing 服务会向 CloudWatch Logs 发出错误日志。错误日志包含字段名称、无法转换的值以及设备的事物名称。下面是一个错误日志示例：

```
{
  "timestamp": "2017-02-20 20:31:22.932",
  "logLevel": "ERROR",
  "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
  "accountId": "000000000000",
  "status": "SucceededWithIssues",
  "eventType": "IndexingCustomFieldFailed",
  "thingName": "thing0",
  "failedCustomFields": [
    {
      "Name": "attributeName1",
      "Value": "apple",
      "ExpectedType": "String"
    },
    {
      "Name": "attributeName2",
      "Value": "2",
      "ExpectedType": "Boolean"
    }
  ]
}
```

如果设备已断开连接大约一小时，则连接状态 timestamp 值可能会缺失。对于持久性会话，在客户端断开连接的时间超过为持久性会话配置的生存时间 (TTL) 后，该值可能会缺失。仅为客户端 ID 具有匹配事物名称的连接，对连接状态数据建立索引。（客户端 ID 是用于将设备到连接 Amazon IoT Core 的值。）

机群故障排除指标

无法创建机群指标

不支持通过更新机群索引配置来降级数据源。

如果您尝试使用降级的数据源创建机群指标（例如，以前的数据源是注册表数据、影子数据和设备连接数据，现在的数据源是注册表数据和影子数据，没有设备连接数据），则您将看到错误并且无法创建机群指标。

不支持修改现有机群指标使用的自定义字段。

在 CloudWatch 中看不到数据点

如果您能够创建机群指标，但看不到 CloudWatch 中的数据点，可能没有符合查询字符串条件的内容。

请参阅以下示例命令，了解如何创建机群指标：

```
aws iot create-fleet-metric --metric-name "example_FM" --query-string
 "thingName:TempSensor* AND attributes.temperature>80" --period 60 --aggregation-field
 "attributes.temperature" --aggregation-type name=Statistics,values=count
```

如果没有符合查询字符串条件 `--query-string "thingName:TempSensor* AND attributes.temperature>80"` 的事物：

- 使用 values=count，您将能够创建机群指标，然后将有数据点显示在 CloudWatch 中。该值 count 的数据点始终为0。

- 采用 `values` 而不是 `count`，您能够创建机群指标，但是在 CloudWatch 中看不到机群指标，也不会在 CloudWatch 中显示任何数据点。

解决“您的Amazon账户已超过流限制”的问题

帮助我们改进此主题

告诉我们如何优化内容

如果您看到 `"Error: You have exceeded the limit for the number of streams in your Amazon account."`，您可以清除账户中未使用的流，而不是请求提高限制。

要使用 Amazon CLI 或 SDK 清理您创建的未使用流：

```
aws iot delete-stream -stream-id value
```

有关更多详细信息，请参阅 delete-stream。

Note

您可以使用 `list-streams` 命令查找流 ID。

Amazon IoT Device Defender 故障排除指南

帮助我们改进此主题

告诉我们如何优化内容

常规

问：使用 Amazon IoT Device Defender 是否需要满足任何先决条件？

答：如果要使用设备报告的指标，必须首先在连接 Amazon IoT 的设备或设备网关上部署代理。设备必须提供一致的客户端标识符或事物名称。

审核

问：我启用了检查，审核持续显示“In-Progress”已经很长时间了。是哪里出错了吗？ 何时可以看到结果？

答：检查一经启用，数据收集就会立即开始。不过，如果账户具有大量数据（例如证书、事物或策略）要收集，您可能要在启用检查后等候一段时间才能看到检查结果。

Detect

问：如何知道在 Amazon IoT Device Defender 安全配置文件行为中设置的阈值？

答：首先创建低阈值的安全配置文件行为，并将其附加到包含相应设备组的事物组。您可以使用 Amazon IoT Device Defender 查看当前指标，然后根据使用案例调整设备行为阈值。

问：我创建了一个行为，但它没有在预计的时间点触发违规。应该如何修复此问题？

答：在定义行为时，您指定了预期的设备的正常行为方式。例如，您有一个监控摄像头，只在 TCP 端口 8888 上连接一个中央服务器，您不希望它进行任何其它连接。要想在摄像头连接其它端口时收到提醒，您可以定义一个类似如下的行为：

```
{
```

```
  "name": "Listening TCP Ports",
  "metric": "aws:listening-tcp-ports",
  "criteria": {
    "comparisonOperator": "in-port-set",
    "value": {
      "ports": [ 8888 ]
    }
  }
}
```

如果摄像头在 TCP 端口 443 上建立 TCP 连接，则违反了设备行为并将触发提醒。

问：我的一个或多个行为违规。如何清除违规？

答：警报将在设备恢复预期行为后解除，如行为配置文件中所定义。系统在收到设备指标数据后即会评估行为配置文件。如果设备超过两天时间未发布任何指标，则违规事件将自动设置为 `alarm-invalidated`。

问：我删除了违规行为，但如何停止提醒？

答：删除行为会停止针对该行为的所有未来的违规和提醒。必须从通知机制中清除早期的提醒。行为被删除后，该行为的违规记录将同账户中的其它违规一样保留相同的时间。

设备指标

问：我在提交已知违反行为的指标报告，但没有触发任何违规。这是怎么回事？

答：请订阅以下 MQTT 主题以核对您的指标报告是否被接受：

```
$aws/things/THING_NAME/defender/metrics/FORMAT/rejected
$aws/things/THING_NAME/defender/metrics/FORMAT/accepted
```

其中，`THING_NAME` 是报告指标的事物的名称，`FORMAT` 是“JSON”或“CBOR ”，具体取决于事物提交的指标报告的格式。

订阅后，对于提交的每个指标报告，您应该会收到关于这些主题的消息。`rejected` 消息表明解析指标报告时出现了问题。消息负载中会包含一条错误消息，以帮助您更正指标报告中的任何错误。`accepted` 消息表明指标报告已正确解析。

问：如果在指标报告中发送空指标会怎样？

答：空的端口列表或 IP 地址列表始终被视为符合对应行为。如果对应行为违规，则违规将被清除。

问：为什么我的设备指标报告中包含不在 Amazon IoT 注册表中的设备的消息？

如果您有一个或多个安全配置文件附加到所有事物或所有未注册的事物，那么 Amazon IoT Device Defender 将包括未注册事物的指标。如果您要排除未注册事物的指标，可以将配置文件附加到所有已注册的设备，而不是所有设备。

问：即使我将安全配置文件附加到所有未注册的设备或所有设备，我也没有看到来自一个或多个未注册设备的消息。如何修复此问题？

请确认您正使用受支持的格式发送格式正确的指标报告。有关信息，请参阅 设备指标文档规范 (p. 973)。验证未注册的设备是否使用的是一致的客户端标识符或事物名称。如果事物名称包含控制字符，或长度超过 128 字节的 UTF-8 编码字符，设备报告的消息将遭到拒绝。

问：如果将未注册的设备添加到注册表，或已注册的设备变为未注册，会发生什么情况？

答：如果在注册表中添加或删除设备：

- 如果它持续发布违规指标，您可以看到设备的两项单独的违规（一个在其注册的事物名称下，一个在其未注册的身份下）。旧身份的活动违规会在两天后不再显示，但在违规历史记录存在最长可达 14 天。

问：我应该在设备指标报告的报告 ID 字段提供什么值？

答：对每个指标报告使用唯一值，用正整数表示。常见的做法是使用 Unix 纪元时间戳。

问：我是否应该为 Amazon IoT Device Defender 指标创建专用 MQTT 连接？

答：不需要使用单独的 MQTT 连接。

问：在连接以发布设备指标时，我应该使用什么客户端 ID？

对于 Amazon IoT 注册表中的设备（事物），可使用注册的事物名称。对于不在 Amazon IoT 注册表中的设备，在连接到 Amazon IoT 时可使用一致的标识符。此做法有助于将违规与事物名称匹配。

问：我能否使用不同的客户端 ID 发布设备指标？

代表另一项事物发布指标是可以实现的。要实现此目的，您可以将指标发布到 Amazon IoT Device Defender 为该设备预留的主题。例如，`Thing-1` 要发布自己的指标，并代表 `Thing-2` 发布指标。`Thing-1` 收集自己的指标并将其发布到 MQTT 主题：

```
$aws/things/Thing-1/defender/metrics/json
```

`Thing-1` 随后从 `Thing-2` 获取指标，并将这些指标发布到 MQTT 主题：

```
$aws/things/Thing-2/defender/metrics/json
```

问：我的账户中能拥有多少个安全配置文件和行为？

答：请参阅 Amazon IoT Device Defender 终端节点和配额。

问：提醒目标的典型目标角色是什么样的？

答：允许 Amazon IoT Device Defender 针对提醒目标（SNS 主题）发布提醒的角色需要具备以下两个条件：

- 信任关系，将 iot.amazonaws.com 指定为可信实体。
- 附加的策略，授权 Amazon IoT 发布到指定的 SNS 主题。例如：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sns:Publish",
            "Resource": "<sns-topic-arn>"
        }
    ]
}
```

- 如果用于发布提示的 SNS 主题是加密主题，那么除了向 SNS 主题发布消息的权限外，Amazon IoT 还必须再授予两个权限。例如：

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "sns:Publish",
               "kms:Decrypt",
               "kms:GenerateDataKey"
           ],
           "Resource": "<sns-topic-arn>"
       }
```

```
    ]
}
```

问：带自定义指标类型 number 的指标报告提交失败，并显示错误消息 Malformed metrics report。这是怎么回事？

答：类型 number 只以单个指标值作为输入，但在 DeviceMetrics 报告中提交指标值时，必须将其作为具有单个值的数组传递。确保将指标值作为数组提交。

错误负载：

```
{"header":{"report_id":12334567,"version":"1.0"},"metrics":{"network_stats":
{"bytes_in":30680,"bytes_out":10652,"packets_in":113,"packets_out":118}},"custom_metrics":
{"my_custom_metric":{"number":0}}}
```

错误消息：

```
{"thingName":"myThing","status":"REJECTED","statusDetails":
{"ErrorCode":"InvalidPayload","ErrorMessage":"Malformed metrics
 report"},"timestamp":1635802047699}
```

无错误有效负载：

```
{"header":{"report_id":12334567,"version":"1.0"},"metrics":{"network_stats":
{"bytes_in":30680,"bytes_out":10652,"packets_in":113,"packets_out":118}},"custom_metrics":
{"my_custom_metric":[{"number":0}]}}
```

响应:

```
{"thingName":"myThing","12334567":1635800375,"status":"ACCEPTED","timestamp":1635801636023}
```

Amazon IoT Device Advisor 故障排除指南

帮助我们改进此主题

告诉我们如何优化内容

常规

问：我是否可以并行运行多个测试套件？

答：可以。Device Advisor 现在支持使用设备级终端节点在不同的设备上运行多个测试套件。如果使用账户级终端节点，则可以一次运行一个套件，因为每个账户有一个 Device Advisor 终端节点。有关更多信息，请参阅配置您的设备。

问：我从设备看到 TLS 连接被 Device Advisor 拒绝。这样是对的吗？

答：可以。Device Advisor 在每次测试运行之前和之后都会拒绝 TLS 连接。我们建议用户实施设备重试机制，以便使用 Device Advisor 获得全自动的测试体验。如果您执行具有多个测试用例的测试套件（例如 TLS 连接、MQTT 连接和 MQTT 发布），我们建议您为设备构建一个机制。该机制可以每 5 秒便尝试一次连接到测试终端节点，每次持续一到两分钟。通过这种方式，您能够以自动的方式按顺序运行多个测试用例。

问：我能否获得 Device Advisor 在我的账户上发起的所有 API 调用的历史记录，以便用于安全分析和运营方面的故障排除？

答：可以。要获取您账户上的 Device Avisor API 调用的历史记录，只需在 Amazon IoT 管理控制台开启 CloudTrail 并将事件源筛选为 iotdeviceadvisor.amazonaws.com 即可。

问：如何查看 CloudWatch 中的 Device Advisor 日志？

答：如果您将所需的策略（例如，CloudWatchFullAccess）添加到您的服务角色，则在测试套件运行过程中生成的日志将上载到 CloudWatch（请参阅 设置 (p. 1016)）。如果测试套件中至少有一个测试用例，则会使用两个日志流创建日志组“aws/iot/deviceadvisor/$testSuiteId”。其中一个日志流为“$testRunId”，包括在测试套件中执行测试用例之前和之后执行操作的日志，例如设置和清理步骤。另一个日志流是特定于测试套件运行的“$suiteRunId_$testRunId”。从设备发送的事件和 Amazon IoT Core 将被记录到此日志流中。

问：设备权限角色的目的是什么？

答：Device Advisor 位于您的测试设备和 Amazon IoT Core 之间来模拟测试场景。它接受来自测试设备的连接和消息，并通过承担您的设备权限角色并代表您启动连接来将它们转发到 Amazon IoT Core。请务必确保设备角色权限与用于运行测试的证书上的权限相同。Amazon IoT 证书策略不会在 Device Advisor 使用设备权限角色代表您启动与 Amazon IoT Core 的连接时强制执行。但是，您设置的设备权限角色中的权限将强制执行。

问：Device Advisor 在哪些区域受支持？

答：us-east-1、us-west-2、ap-northeast-1 和 eu-west-1 区域支持 Device Advisor。

问：为什么我看到不一致的结果？

答：结果不一致的主要原因之一是为测试的 `EXECUTION_TIMEOUT` 设置了过低的值。有关推荐和默认 `EXECUTION_TIMEOUT` 值的更多信息，请参阅 Device Advisor 测试案例。

问：Device Advisor 支持什么 MQTT 协议？

答：Device Advisor 支持使用 X509 客户端证书的 MQTT 3.1.1 版。

问：即使我尝试将设备连接到测试终端节点，如果我的测试用例失败并显示执行超时消息，该怎么办？

答：验证创建要用作设备角色的 IAM 角色下的所有步骤。如果测试仍然失败，则可能是设备没有发送正确的服务器名称指示 (SNI) 扩展，这是 Device Advisor 工作所必需的。正确的 SNI 值是在执行配置您的设备部分时返回的终端节点地址。此外，Amazon IoT 还要求设备将服务器名称指示 (SNI) 扩展发送到传输层安全 (TLS) 协议。有关更多信息，请参阅 Amazon IoT 中的传输安全。

问：我的 MQTT 连接失败并显示“libaws-c-mqtt: AWS_ERROR_MQTT_UNEXPECTED_HANGUP”错误，（或者）我的设备的 MQTT 连接正在自动断开与 Device Advisor 端点的连接。如何解决此错误？

答：此特定错误代码和意外断开连接可能是由许多不同的原因造成的，但很可能与附加到设备的设备角色相关。下面的检查点（按优先级顺序）将解决此问题。

- 附加到设备的设备角色必须具有运行测试所需的最低 IAM 权限。Device Advisor 将使用附加的设备角色来代表测试设备执行 Amazon IoT MQTT 操作。如果缺少所需权限，则当设备尝试连接到 Device Advisor 端点时，会看到 `AWS_ERROR_MQTT_UNEXPECTED_HANGUP` 错误或发生意外断开连接。例如，如果您已选择运行 MQTT 发布测试用例，则 Connect（连接）和 Publish（发布）操作都必须包含在具有相应 ClientId 和 Topic 的角色中（您可以通过使用逗号分隔值来提供多个值，也可以使用通配符（*）字符提供前缀值）。例如，要为任何以 `TestTopic` 开头的主题提供发布权限，您可以将“`TestTopic*`”作为资源值。下面是部分策略示例。
- 在设备角色中为资源类型定义的值与代码中使用的实际值不匹配。例如：角色中定义的 ClientId 与设备代码中使用的实际 ClientId 不匹配。ClientId、Topic 和 TopicFilter 之类的值在设备角色和代码中必须相同。
- 附加到设备的设备证书必须处于活跃状态，并且附加了具有资源所需操作权限的策略。请注意，设备证书策略授予或拒绝对 Amazon IoT 资源和 Amazon IoT Core 数据面板操作的访问权限。Device Advisor 要求您将活跃设备证书附加到您的设备，该证书可授予在测试用例期间使用的操作权限。

排除设备机群断开连接的故障

帮助我们改进此主题

告诉我们如何优化内容

Amazon IoT 设备机群断开连接可能由于多种原因导致的。本文介绍了如何诊断断开连接的原因，以及如何处理因定期维护 Amazon IoT 服务或节流限制而导致的连接断开。

诊断断开连接的原因

您可以检查 CloudWatch 中的 AWSIotLogsV2 日志组以在日志条目的 `disconnectReason` 字段中确定断开连接的原因。

您还可以使用 Amazon IoT 的生命周期事件功能来确定断开连接的原因。如果您已订阅生命周期断开连接事件 (`$aws/events/presence/disconnected/clientId`)，您将在断开连接的问题发生时收到 Amazon IoT 的通知。您可以在通知的 `disconnectReason` 字段找到断开连接的原因。

有关更多信息，请参阅 CloudWatch Amazon IoT 日志条目和生命周期事件。

排除由于 Amazon IoT 服务维护导致的断开连接故障

由于 Amazon IoT 的服务维护导致的断开连接将在 Amazon IoT 的生命周期事件和 CloudWatch 中记录为 `SERVER_INITIATED_DISCONNECT`。要处理这些断开连接的问题，请调整客户端设置，以确保您的设备可以自动重新连接到 Amazon IoT 平台。

排除由于节流限制而导致的断开连接故障

由节流限制引起的断开连接将在 Amazon IoT 的生命周期事件和 CloudWatch 中记录为 `THROTTLED`。要处理这些断开连接的问题，您可以在设备计数增长时请求消息代理限制增加。

有关更多信息，请参阅 Amazon IoT 核心消息代理。

Amazon IoT 错误

帮助我们改进此主题

告诉我们如何优化内容

此部分列出了 Amazon IoT 发送的错误代码。

消息代理错误代码

错误代码	错误描述
400	错误请求。
401	未授权。
403	禁止。
503	服务不可用。

身份和安全错误代码

错误代码	错误描述
401	未授权。

设备影子错误代码

错误代码	错误描述
400	错误请求。

错误代码	错误描述
401	未授权。
403	禁止.
404	未找到。
409	冲突。
413	请求太大。
422	无法处理请求。
429	过多请求。
500	内部错误。
503	服务不可用。

Amazon IoT 配额

您可以在《Amazon 一般参考》中查找有关 Amazon IoT 配额的信息。

- 有关 Amazon IoT Core 配额信息，请参阅 Amazon IoT Core 端点和配额。
- 有关 Amazon IoT Device Management 配额信息，请参阅 Amazon IoT Device Management 端点和配额。
- 有关 Amazon IoT Device Defender 配额信息，请参阅 Amazon IoT Device Defender 端点和配额。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。