



開發人員指南

# AWS IoT Core



# AWS IoT Core: 開發人員指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

# Table of Contents

什麼是 AWS IoT ? .....	1
您的裝置和應用程式存取方式 AWS IoT .....	2
什麼 AWS IoT 可以做 .....	2
產業中的 IoT .....	2
家庭自動化中的 IoT .....	3
如何 AWS IoT 工作 .....	4
IoT 的宇宙 .....	4
AWS IoT 服務概述 .....	6
AWS IoT Core 服務 .....	10
進一步了解 AWS IoT .....	13
培訓資源 AWS IoT .....	13
AWS IoT 資源和指南 .....	14
AWS IoT 在社交媒體 .....	14
AWS IoT Core 規則引擎使用的服務 .....	15
AWS IoT Core 支援的通訊協定 .....	16
AWS IoT 主控台的最新消息 .....	17
圖例 .....	19
使用 AWS 軟體開發套件 .....	20
開始使用 AWS IoT Core .....	22
將您的第一部裝置 Connect 到 AWS IoT Core .....	22
設定您的 AWS 帳戶 .....	24
註冊一個 AWS 帳戶 .....	24
建立具有管理權限的使用者 .....	24
開啟主 AWS IoT 控制台 .....	26
試用 AWS IoT Core 互動式教學 .....	26
連線 IoT 裝置 .....	27
儲存裝置離線狀態 .....	27
將裝置資料路由到服務 .....	28
嘗試 AWS IoT 快速連接 .....	29
步驟 1. 開始教學課程 .....	30
步驟 2. 建立物件 .....	30
步驟 3. 將檔案下載到您的裝置 .....	33
步驟 4. 執行範例 .....	35
步驟 5. 深入探索 .....	39

測試與裝置資料端點的連線 .....	39
在實作教學中探索 AWS IoT Core 服務 .....	45
哪種裝置選項最適合您？ .....	46
建立 AWS IoT 資源 .....	47
設定您的裝置 .....	50
使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT .....	86
在 MQTT 用戶端中檢視 MQTT 訊息 .....	87
從 MQTT 用戶端發佈 MQTT 訊息 .....	89
在 MQTT 用戶端中測試共享訂閱 .....	91
連接到 AWS IoT Core .....	93
AWS IoT Core：控制平面端點 .....	93
AWS IoT 裝置端點 .....	94
AWS IoT Core 適用於 LoRa WAN 閘道器和裝置 .....	95
連線至 AWS IoT Core 服務端點 .....	96
AWS CLI 對於 AWS IoT Core .....	97
AWS 開發套件 .....	97
AWS 行動開發套件 .....	103
AWS IoT Core 服務的其餘 API .....	104
將裝置連接至 AWS IoT .....	104
AWS IoT 裝置資料和服務端點 .....	105
AWS IoT 裝置開發套件 .....	106
裝置通訊協定 .....	109
MQTT 主題 .....	142
可設定端點 .....	161
連接至 AWS IoT FIPS 端點 .....	178
AWS IoT Core：控制平面端點 .....	179
AWS IoT Core：資料平面端點 .....	179
AWS IoT Device Management：任務資料端點 .....	179
AWS IoT Device Management：Fleet Hub 端點 .....	180
AWS IoT Device Management：安全通道端點 .....	180
AWS IoT 教學課程 .....	181
使用 AWS IoT 裝置用戶端建置示範 .....	181
使用 AWS IoT 裝置用戶端建置示範的先決條件 .....	182
準備用於 AWS IoT 裝置用戶端的裝置 .....	184
安裝並設定 AWS IoT 裝置用戶端 .....	196
示範 MQTT 訊息與 AWS IoT 裝置用戶端通訊 .....	208

使用 AWS IoT 裝置用戶端來示範遠端動作 (任務) .....	226
清除 .....	240
使用 AWS IoT 裝置 SDK 建置解決方案 .....	249
開始使用 AWS IoT 裝置 SDK 建置解決方案 .....	249
使用 AWS IoT 裝置 SDK 將裝置連接至 AWS IoT Core .....	249
建立 AWS IoT 規則以將裝置資料路由至其他服務 .....	270
在裝置離線時保留裝置狀態 .....	309
建立自訂授權者 AWS IoT Core .....	336
監測土壤水分 AWS IoT 與樹莓派 .....	351
管理裝置 AWS IoT .....	364
如何使用登錄檔管理物件 .....	364
建立物件 .....	365
列出物件 .....	365
說明物件 .....	367
更新物件 .....	368
刪除物件 .....	368
將主體連接至物件 .....	369
將主體與物件分離 .....	369
物件類型 .....	369
建立物件類型 .....	370
列出物件類型 .....	370
描述物件類型 .....	371
將物件類型與物件建立關聯 .....	371
棄用物件類型 .....	372
刪除物件類型 .....	373
靜態物件群組 .....	373
建立靜態物件群組 .....	375
描述物件群組 .....	376
新增一個物件到一個靜態的物件群組 .....	377
從靜態物件群組中移除一個物件 .....	377
列出物件群組中的物件 .....	378
列出物件群組 .....	378
列出物件的群組 .....	380
更新靜態物件群組 .....	381
刪除物件群組 .....	381
將政策附加到靜態物件群組 .....	382

將政策與靜態物件群組分離 .....	383
列出連線至靜態物件群組的政策 .....	383
列出政策所在的群組 .....	383
為物件取得有效政策 .....	384
MQTT 動作的測試授權 .....	385
動態物件群組 .....	386
動態物件群組的使用案例 .....	387
建立動態物件群組 .....	388
描述動態物件群組 .....	389
更新動態物件群組 .....	390
刪除動態物件群組 .....	390
動態與靜態物件群組限制 .....	391
動態物件群組限制 .....	391
標記您的 AWS IoT 資源 .....	394
標籤基本概念 .....	394
標籤的限制與上限 .....	395
搭配 IAM 政策使用標籤 .....	396
帳單群組 .....	398
檢視成本分配和用量資料 .....	399
安全 .....	401
中的安全性 AWS IoT .....	402
身分驗證 .....	403
AWS 培訓和認證 .....	403
X.509 憑證概觀 .....	403
伺服器驗證 .....	403
用戶端身分驗證 .....	407
自訂身分驗證和授權 .....	438
授權 .....	454
AWS 培訓和認證 .....	457
AWS IoT Core 政策 .....	457
使用 AWS IoT Core 認證提供者授權直接呼叫 AWS 服務 .....	523
利用 IAM 跨帳戶存取 .....	529
資料保護 .....	530
中的資料加密 AWS IoT .....	531
中的運輸安全 AWS IoT Core .....	532
資料加密 .....	537

身分與存取管理 .....	538
物件 .....	538
使用 IAM 身分來驗證 .....	539
使用政策管理存取權 .....	541
如何與 IAM AWS IoT 搭配使用 .....	543
身分型政策範例 .....	574
AWS 受管理政策 .....	577
故障診斷 .....	591
記錄和監控 .....	593
監控工具 .....	593
法規遵循驗證 .....	595
恢復能力 .....	596
AWS IoT Core 搭配 VPC 端點使用 .....	596
為 AWS IoT Core 資料平面建立 VPC 端點 .....	597
建立 AWS IoT Core 認證提供者的 VPC 端點 .....	598
建立 Amazon VPC 界面端點 .....	598
配置私有託管區域 .....	600
控制對 VPC 端點 AWS IoT Core 的存取 .....	601
限制 .....	603
使用擴展 VPC 端點 AWS IoT Core .....	603
搭配使用自訂網域與 VPC 端點 .....	604
下列項目的 VPC 端點的可用性 AWS IoT Core .....	604
基礎架構安全 .....	604
安全監控 .....	604
安全最佳實務 .....	605
保護 MQTT 連接 AWS IoT .....	605
讓裝置的時鐘保持同步 .....	607
驗證伺服器憑證 .....	608
使用每個裝置單一身分 .....	608
使用第二個 AWS 區域 作為備份 .....	608
使用及時佈建 .....	609
執行 AWS IoT 裝置建議程式測試的權限 .....	609
Device Advisor 跨服務預防混淆代理人 .....	610
AWS 培訓和認證 .....	611
監控 AWS IoT .....	612
設定 AWS IoT 記錄 .....	613

設定記錄角色和政策 .....	613
在 AWS IoT (主控台) 中設定預設記錄 .....	615
設定預設登入 AWS IoT (CLI) .....	617
在 AWS IoT 設定資源特定的記錄 (CLI) .....	618
日誌層級 .....	620
使用 Amazon 監控 AWS IoT 警示和指標 CloudWatch .....	621
使用 AWS IoT 指標 .....	621
在中建立 CloudWatch 警示 AWS IoT .....	622
AWS IoT 量度和維度 .....	626
AWS IoT 使用 CloudWatch 記錄監控 .....	643
在 CloudWatch 主控台中檢視 AWS IoT 記錄 .....	643
CloudWatch 記 AWS IoT 錄檔項目 .....	644
將設備端日誌上傳到 Amazon CloudWatch .....	677
運作方式 .....	678
使用 AWS IoT 規則上傳裝置端日誌 .....	678
使用記錄 AWS IoT API 呼叫 AWS CloudTrail .....	688
AWS IoT 中的資訊 CloudTrail .....	688
瞭解 AWS IoT 記錄檔項目 .....	689
規則 .....	691
授與 AWS IoT 規則所需的存取權 .....	692
傳遞角色權限 .....	694
建立規則 .....	695
建立規則 (主控台) .....	697
建立規則 (CLI) .....	697
標記您的規則 .....	702
檢視您的規則 .....	703
刪除規則 .....	703
AWS IoT 規則動作 .....	703
Apache Kafka .....	706
CloudWatch 警報 .....	717
CloudWatch 日誌 .....	718
CloudWatch 度量 .....	721
DynamoDB .....	723
DynamoDBv2 .....	726
Elasticsearch .....	728
HTTP .....	730



IoT Analytics .....	770
AWS IoT Events .....	772
AWS IoT SiteWise .....	774
Firehose .....	779
Kinesis Data Streams .....	782
Lambda .....	784
位置 .....	787
OpenSearch .....	790
Republish .....	793
S3 .....	796
Salesforce IoT .....	798
SNS .....	799
SQS .....	801
Step Functions .....	803
Timestream .....	805
規則疑難排解 .....	811
使用規則存取跨帳戶資 AWS IoT 源 .....	812
必要條件 .....	812
Amazon SQS 的跨帳戶設定 .....	812
Amazon SNS 的跨帳戶設定 .....	814
Amazon S3 的跨帳戶設定 .....	816
跨帳戶設定 AWS Lambda .....	818
錯誤處理 (錯誤動作) .....	820
錯誤動作訊息格式 .....	821
錯誤動作範例 .....	822
使用基本擷取減少簡訊費用 .....	823
使用基本擷取 .....	823
AWS IoT SQL 參考資料 .....	824
SELECT 子句 .....	825
FROM 子句 .....	827
WHERE 子句 .....	828
資料類型 .....	829
運算子 .....	833
函數 .....	842
文字 .....	906
案例陳述式 .....	907

JSON Extensions .....	908
替代範本 .....	910
巢狀物件查詢 .....	912
二進位承載 .....	914
SQL 版本 .....	919
Device Shadow 服務 .....	922
使用影子 .....	922
選擇使用已命名或未命名的影子 .....	922
存取影子 .....	923
在裝置、應用程式和其他雲端服務中使用影子 .....	924
訊息順序 .....	924
裁剪影子訊息 .....	926
在裝置中使用影子 .....	927
在第一次連接時初始化設備 AWS IoT .....	928
在設備連接到時處理消息 AWS IoT .....	929
裝置重新連線時處理訊息 AWS IoT .....	930
在應用程式和服務中使用影子 .....	930
在連接時初始化應用程序或服務 AWS IoT .....	931
在應用程式或服務連線時處理狀態變更 AWS IoT .....	932
偵測裝置已連線 .....	932
模擬 Device Shadow 服務通訊 .....	934
準備模擬 .....	934
初始化裝置 .....	934
從應用程式傳送更新 .....	938
回應裝置中的更新 .....	940
觀察應用程式中的更新 .....	945
超越模擬 .....	947
與影子互動 .....	947
通訊協定支援 .....	947
請求和報告狀態 .....	948
更新影子 .....	948
擷取影子文件 .....	952
刪除影子資料 .....	953
Device Shadow REST API .....	955
GetThingShadow .....	956
UpdateThingShadow .....	957

DeleteThingShadow .....	958
ListNamedShadowsForThing .....	959
Device Shadow MQTT 主題 .....	961
/get .....	962
/get/accepted .....	963
/get/rejected .....	964
/update .....	964
/update/delta .....	966
/update/accepted .....	967
/update/documents .....	968
/update/rejected .....	969
/delete .....	969
/delete/accepted .....	970
/delete/rejected .....	971
Device Shadow 服務文件 .....	972
影子文件範例 .....	972
文件屬性 .....	978
差量狀態 .....	979
版本控制影子文件 .....	982
影子文件中的用戶端字符 .....	982
空白影子文件屬性 .....	982
影子文件中的陣列值 .....	983
Device Shadow 錯誤訊息 .....	984
任務 .....	986
存取 AWS IoT 工作 .....	986
AWS IoT 工作區域和端點 .....	986
什麼是遠程操作？ .....	986
將工作用於遠端 AWS IoT Device Management 作業的好處 .....	987
什麼是 AWS IoT 工作？ .....	988
任務的重要概念 .....	989
任務和任務執行狀態 .....	992
管理任務 .....	997
任務的程式碼簽署 .....	997
任務文件 .....	997
預先簽章的 URL .....	997
使用主控台建立和管理任務 .....	1000

使用 CLI 建立和管理任務 .....	1002
任務範本 .....	1013
自訂和 AWS 受管理的範本 .....	1013
使用 AWS 受管理範本 .....	1013
建立自訂任務範本 .....	1031
任務 組態 .....	1039
任務組態的運作方式 .....	1039
指定其他組態 .....	1051
裝置和任務 .....	1059
設定裝置與任務合作 .....	1062
裝置工作流程 .....	1062
任務工作流程 .....	1064
任務通知 .....	1068
AWS IoT 任務 API 操作 .....	1076
任務管理、控制 API 和控制資料類型 .....	1078
任務裝置 MQTT、HTTPS API 操作和資料類型 .....	1097
為任務加密使用者和裝置 .....	1111
工作的必要原則 AWS IoT 類型 .....	1111
授權任務使用者和雲端服務 .....	1112
授權裝置使用任務 .....	1122
任務限制 .....	1126
作用中和並行任務限制 .....	1126
AWS IoT 安全隧道 .....	1129
什麼是安全通道？ .....	1129
安全通道概念 .....	1129
安全通道運作方式 .....	1130
安全通道生命週期 .....	1131
AWS IoT 安全隧道教程 .....	1132
本節中的教學課程 .....	1132
開啟通道並啟動遠端裝置的 SSH 工作階段 .....	1133
開啟遠端裝置的通道並使用瀏覽器型 SSH .....	1149
本機代理 .....	1152
如何使用本機代理 .....	1153
為使用 Web 代理的裝置配置本機代理 .....	1158
多工處理和同步 TCP 連線 .....	1165
多工處理多個資料串流 .....	1166

使用同時 TCP 連線 .....	1169
設定遠端裝置並使用 IoT 代理程式 .....	1171
IoT Agent Snippet .....	1172
控制 Kibana 的存取 .....	1174
通道存取先決條件 .....	1174
通道存取政策 .....	1174
解決安全通道連線問題 .....	1180
無效的用戶端存取字符錯誤 .....	1181
用戶端字符不相符錯誤 .....	1181
遠端裝置連線問題 .....	1183
裝置佈建 .....	1185
佈建 AWS IoT 中的裝置 .....	1186
機群佈建 API .....	1187
使用機群佈建來佈建沒有裝置憑證的裝置 .....	1187
透過要求佈建 .....	1188
由信任的使用者佈建 .....	1190
將預先佈建掛接與 AWS CLI 搭配使用 .....	1192
佈建具有裝置憑證的裝置 .....	1195
單一物件佈建 .....	1196
Just-in-time 佈建 .....	1196
大量註冊 .....	1202
佈建範本 .....	1203
參數部分 .....	1203
資源部分 .....	1204
大量註冊的範本範例 .....	1209
just-in-time 佈建的範本範例 (JITP) .....	1210
機群佈建 .....	1212
預先佈建掛接 .....	1215
預先佈建掛接輸入 .....	1216
預先佈建掛接傳回值 .....	1216
預先佈建掛接 Lambda 範例 .....	1217
使用憑證提供者自行管理 AWS IoT Core 憑證簽署 .....	1220
自我管理憑證簽署在叢集佈建中的運作方式 .....	1220
憑證提供程式 Lambda 函數 .....	1222
憑證提供者 Lambda 函數傳回值 .....	1222
Lambda 函數範例 .....	1223

用於叢集佈建的自我管理憑證簽署 .....	1224
AWS CLI 憑證提供者的指令 .....	1226
為安裝裝置的使用者建立 IAM 政策和角色 .....	1228
為安裝裝置的使用者建立 IAM 政策 .....	1228
為安裝裝置的使用者建立 IAM 角色 .....	1229
更新現有政策以授權新範本 .....	1230
裝置佈建 MQTT API .....	1231
CreateCertificateFromCsr .....	1232
CreateKeysAndCertificate .....	1234
RegisterThing .....	1236
機群索引 .....	1240
管理索引更新 .....	1240
跨資料來源搜尋 .....	1240
查詢彙總資料 .....	1240
使用機群指標監控彙總資料和建立警示 .....	1240
管理機群索引 .....	1241
物件索引 .....	1241
物件群組索引 .....	1242
受管欄位 .....	1242
自訂欄位 .....	1244
管理物件索引 .....	1245
管理物件群組索引 .....	1260
查詢彙總資料 .....	1262
GetStatistics .....	1262
GetCardinality .....	1265
GetPercentiles .....	1266
GetBuckets聚合 .....	1268
授權 .....	1269
查詢語法 .....	1269
支援的功能 .....	1269
不支援的功能 .....	1270
備註 .....	1270
範例物件查詢 .....	1271
範例物件群組查詢 .....	1274
索引位置資料 .....	1276
支援的資料格式 .....	1276

如何為位置資料建立索引 .....	1277
更新物件索引組態 .....	1277
地理查詢示例 .....	1280
入門教學課程 .....	1281
機群指標 .....	1285
入門教學課程 .....	1286
管理機群指標 .....	1292
MQTT 型檔案交付 .....	1299
什麼是串流？ .....	1299
管理 AWS 雲端中的串流 .....	1300
授與您的裝置許可 .....	1301
將您的設備 Connect 到 AWS IoT .....	1301
TagResource 用法 .....	1302
在裝置中使用以 AWS IoT MQTT 為基礎的檔案傳遞 .....	1303
用 DescribeStream 於獲取流數據 .....	1303
從串流檔案取得資料區塊 .....	1305
處理以 AWS IoT MQTT 為基礎的檔案傳送的錯誤 .....	1310
FreeRTOS OTA 中的範例使用案例 .....	1312
Device Advisor .....	1313
設定 .....	1314
建立 IoT 物件 .....	1314
建立要用作裝置角色的 IAM 角色 .....	1315
為要使用 Device Advisor 的 IAM 使用者建立自訂受管政策 .....	1318
建立要使用 Device Advisor 的 IAM 使用者 .....	1318
設定您的裝置 .....	1320
在主控台中開始使用 Device Advisor .....	1321
Device Advisor 工作流程 .....	1330
必要條件 .....	1330
建立測試套件定義 .....	1330
取得測試套件定義 .....	1333
取得測試端點 .....	1333
啟動測試套件執行 .....	1334
取得測試套件執行 .....	1335
停止測試套件執行 .....	1335
取得成功資格測試套件執行的資格報告 .....	1335
Device Advisor 詳細主控台工作流程 .....	1336

必要條件 .....	1336
建立測試套件定義 .....	1336
啟動測試套件執行 .....	1344
停止測試套件執行 (選用) .....	1346
檢視測試套件執行詳細資訊和記錄 .....	1347
下載 AWS IoT 資格報告 .....	1348
長時間測試主控台工作流程 .....	1349
Device Advisor VPC 端點 (AWS PrivateLink) .....	1357
AWS IoT Core Device Advisor VPC 端點的考量 .....	1357
為 AWS IoT Core Device Advisor 建立界面 VPC 端點 .....	1358
控制對 VPC 端點 AWS IoT Core Device Advisor 的存取 .....	1358
Device Advisor 測試案例 .....	1360
裝置顧問測試案例，以符合「AWS 裝置資格認證計劃」資格。 .....	1360
TLS .....	1361
MQTT .....	1367
影子 .....	1379
任務執行 .....	1382
許可和政策 .....	1384
長時間測試 .....	1384
軟體套件目錄 .....	1403
準備使用軟體套件目錄 .....	1403
.....	1403
套件版本生命週期 .....	1403
套件版本命名慣例 .....	1405
預設版本 .....	1405
版本屬性 .....	1406
啟用 AWS IoT 叢集索引 .....	1406
預留已命名影子 .....	1406
刪除軟體套件 .....	1408
準備安全性 .....	1408
以資源為基礎的身分驗證 .....	1408
AWS IoT 部署套件版本的 Job 權限 .....	1410
AWS IoT 更新保留的命名陰影的 Job 權限 .....	1411
AWS IoT 可從 Amazon S3 下載的任務許可 .....	1413
準備機群索引 .....	1413
將 \$package 影子設定為資料來源 .....	1413



控制台顯示的指標 .....	1414
查詢模式 .....	1414
透過 getBucketsAggregation 收集套件版本分佈 .....	1417
準備 AWS IoT 工作 .....	1417
AWS IoT 工作的替代參數 .....	1418
準備任務文件和套件版本以進行部署 .....	1420
部署時命名套件和版本 .....	1420
透過 AWS IoT 動態物件群組鎖定工作 .....	1420
預留已命名影子和套件版本 .....	1420
解除安裝軟體套件 .....	1421
開始使用 .....	1422
建立套件和版本 .....	1422
部署套件版本 .....	1423
將套件版本建立關聯 .....	1425
AWS IoT Core 裝置位置 .....	1427
測量類型和求解器 .....	1427
AWS IoT Core 裝置位置的運作方式 .....	1428
如何使用 AWS IoT Core 裝置位置 .....	1429
解析 IoT 裝置的位置 .....	1430
解析裝置位置 (主控台) .....	1430
解析裝置位置 (API) .....	1434
對解析位置時發生的錯誤進行疑難排解 .....	1435
使用 MQTT 主題解析裝置位置 .....	1436
裝置位置 MQTT 主題的格式 .....	1436
裝置位置 MQTT 主題的政策 .....	1437
裝置位置主題和承載 .....	1438
位置求解器和裝置承載 .....	1443
基於 Wi-Fi 的求解器 .....	1443
行動網路求解器 .....	1444
IP 反向查詢求解器 .....	1449
求解器 .....	1449
事件訊息 .....	1451
如何產生事件訊息 .....	1451
接收事件訊息的政策 .....	1451
啟用事件 AWS IoT .....	1452
登錄檔事件 .....	1457

物件事件 .....	1457
物件類型事件 .....	1458
物件群組事件 .....	1461
任務事件 .....	1467
生命週期事件 .....	1471
連線/中斷連線事件 .....	1472
訂閱/取消訂閱事件 .....	1475
疑難排解 .....	1478
AWS IoT Core 疑難排解指 .....	1478
診斷連線問題 .....	1479
診斷規則問題 .....	1482
診斷影子的問題 .....	1484
診斷 Salesforce 動作問題 .....	1485
診斷串流限制 .....	1486
對裝置機群中斷連線進行疑難排解 .....	1487
AWS IoT 裝置顧問疑難排解指 .....	1488
AWS IoT Device Management 疑難排解指 .....	1490
AWS IoT 工作疑難解 .....	1490
機群索引疑難排解指南 .....	1494
AWS IoT 錯誤 .....	1497
AWS IoT 裝置 SDK、行動 SDK 和 AWS IoT 裝置用戶端 .....	1499
AWS IoT 裝置開發套件 .....	1499
AWS IoT 適用於嵌入式 C 的裝置 SDK .....	1501
舊版 AWS IoT 裝置 SDK .....	1501
AWS 行動開發套件 .....	1502
AWS IoT 裝置用戶端 .....	1503
程式碼範例 .....	1504
動作 .....	1510
AttachThingPrincipal .....	1510
CreateKeysAndCertificate .....	1514
CreateThing .....	1519
CreateTopicRule .....	1523
DeleteCertificate .....	1528
DeleteThing .....	1531
DeleteTopicRule .....	1534
DescribeEndpoint .....	1535

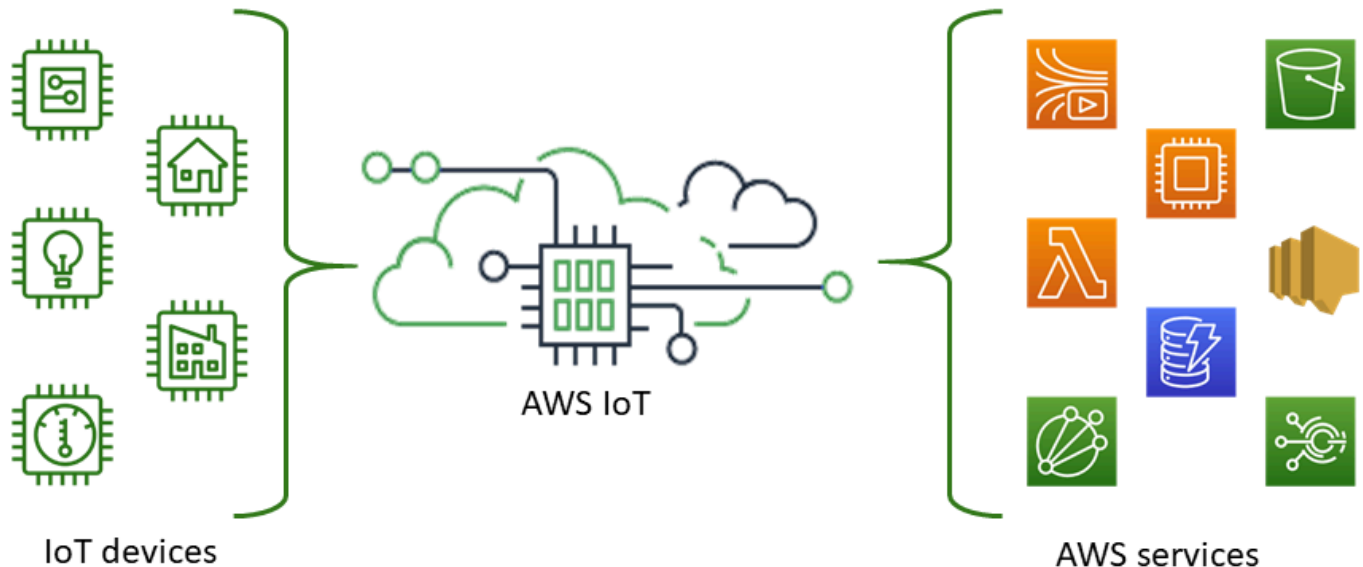
---

DescribeThing .....	1539
DetachThingPrincipal .....	1543
ListCertificates .....	1546
ListThings .....	1551
SearchIndex .....	1554
UpdateIndexingConfiguration .....	1559
UpdateThing .....	1561
案例 .....	1565
使用裝置管理使用案例 .....	1565
AWS IoT 配額 .....	1614
AWS IoT Core 定價 .....	1615
.....	mdcxvi

# 什麼是 AWS IoT ？

AWS IoT 提供將 IoT 裝置連接至其他裝置和 AWS 雲端服務的雲端服務。AWS IoT 提供可協助您將 IoT 裝置整合至 AWS IoT 基礎解決方案的裝置軟體。如果您的設備可以連接到 AWS IoT，則 AWS IoT 可以將它們連接到 AWS 提供的雲服務。

如需實際操作介紹 AWS IoT，請造訪 [開始使用 AWS IoT Core](#)。



AWS IoT 可讓您選擇最適合您解決方案的 up-to-date 技術。為了協助您在現場管理和支援 IoT 裝置，請 AWS IoT Core 支援下列通訊協定：

- [MQTT \(訊息佇列和遙測傳輸\)](#)
- [透過 WSS 的 MQTT \(Websocket 安全\)](#)
- [HTTPS \(超文字傳輸通訊協定：安全\)](#)
- [LoRaWAN \(長距離廣域網路\)](#)

AWS IoT Core 訊息代理程式支援透過 WSS 通訊協定使用 MQTT 和 MQTT 來發佈和訂閱訊息的裝置和用戶端。其也支援使用 HTTPS 通訊協定發佈訊息的裝置和用戶端。

AWS IoT Core for LoRa WAN 可協助您連線和管理無線 LoRa WAN (低功率遠距離廣域網路) 裝置。AWS IoT Core 對於 LoRa WAN，則取代了您開發和操作 LoRa WAN 網絡服務器 (LNS) 的需求。

如果您不需要裝置通訊、[規則](#)或[工作](#)等 AWS IoT 功能，請參閱[AWS 息](#)以取得其他可能更符合您需求之 AWS IoT 訊息服務的相關資訊。

## 您的裝置和應用程式存取方式 AWS IoT

AWS IoT 提供以下介面[AWS IoT 教學課程](#)：

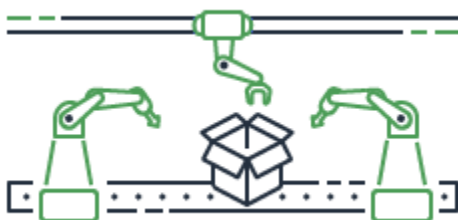
- AWS IoT 裝置 SDK：在您的裝置上建置可傳送訊息和接收訊息的應用程式。AWS IoT 如需詳細資訊，請參閱 [AWS IoT 裝置 SDK](#)、[行動 SDK](#) 和 [AWS IoT 裝置用戶端](#)。
- AWS IoT Core 適用 AWS IoT Core 於 LoRa WAN — [透過使用 WAN Connect 和管理遠端 LoRa WAN \(WAN\) 裝置和閘道。LoRa](#)
- AWS Command Line Interface (AWS CLI) — AWS IoT 在視窗、macOS 和 Linux 上執行指令。這些命令可讓您建立並管理物件物件、憑證、規則、任務和政策。若要開始使用，請參閱《[使用者指南 AWS Command Line Interface](#)》。如需有關的指令的詳細資訊 AWS IoT，請參閱《AWS CLI 命令參考》中的 [iot](#)。
- AWS IoT API：使用 HTTP 或 HTTPS 請求來建置您的 IoT 應用程式。這些 API 動作可讓您以程式設計方式建立並管理物件物件、憑證、規則和政策。如需的 API 動作的詳細資訊 AWS IoT，請參閱 AWS IoT API 參考中的 [動作](#)。
- AWS 開發套件：使用特定於語言的 API 建置 IoT 應用程式。這些 SDK 整合了 HTTP/HTTPS API，可讓您以任何支援的語言編寫程式。如需詳細資訊，請參閱[AWS 開發套件與工具](#)。

您也可以 AWS IoT 透過[AWS IoT 主控台進行存取](#)，[主控台](#)提供圖形化使用者介面 (GUI)，您可以透過此介面配置和管理 IoT 解決方案的物件物件、憑證、規則、工作、原則及其他元素。

## 什麼 AWS IoT 可以做

本主題說明您可能需要 AWS IoT 支援的一些解決方案。

### 產業中的 IoT



這些是[工業使用案例的一些 AWS IoT 解決方案範例](#)，這些解決方案應用 IoT 技術來提高工業流程的效能和生產力。

適用於工業使用案例的解決方案

- [用 AWS IoT 於在工業營運中建立預測性品質模型](#)

瞭解如 AWS IoT 何從工業營運收集和分析資料，以建立預測性品質模型。[進一步了解](#)

- [用 AWS IoT 於支援工業營運中的預測性維護](#)

瞭解如 AWS IoT 何協助規劃預防性維護，以減少意外停機時間。[進一步了解](#)

## 家庭自動化中的 IoT



這些是[家庭自動化使用 AWS IoT 案例的一些解決方案示例](#)，它們應用 IoT 技術來構建可擴展的 IoT 應用程式，從而使用互聯家庭設備自動化家庭

家庭自動化解決方案

- [AWS IoT 在您的連網家庭中使用](#)

了解如 AWS IoT 何提供整合式家庭自動化解決方案。

- [用 AWS IoT 於提供家庭安全和監控](#)

了解如 AWS IoT 何將機器學習和邊緣運算應用於您的家庭自動化解決方案。

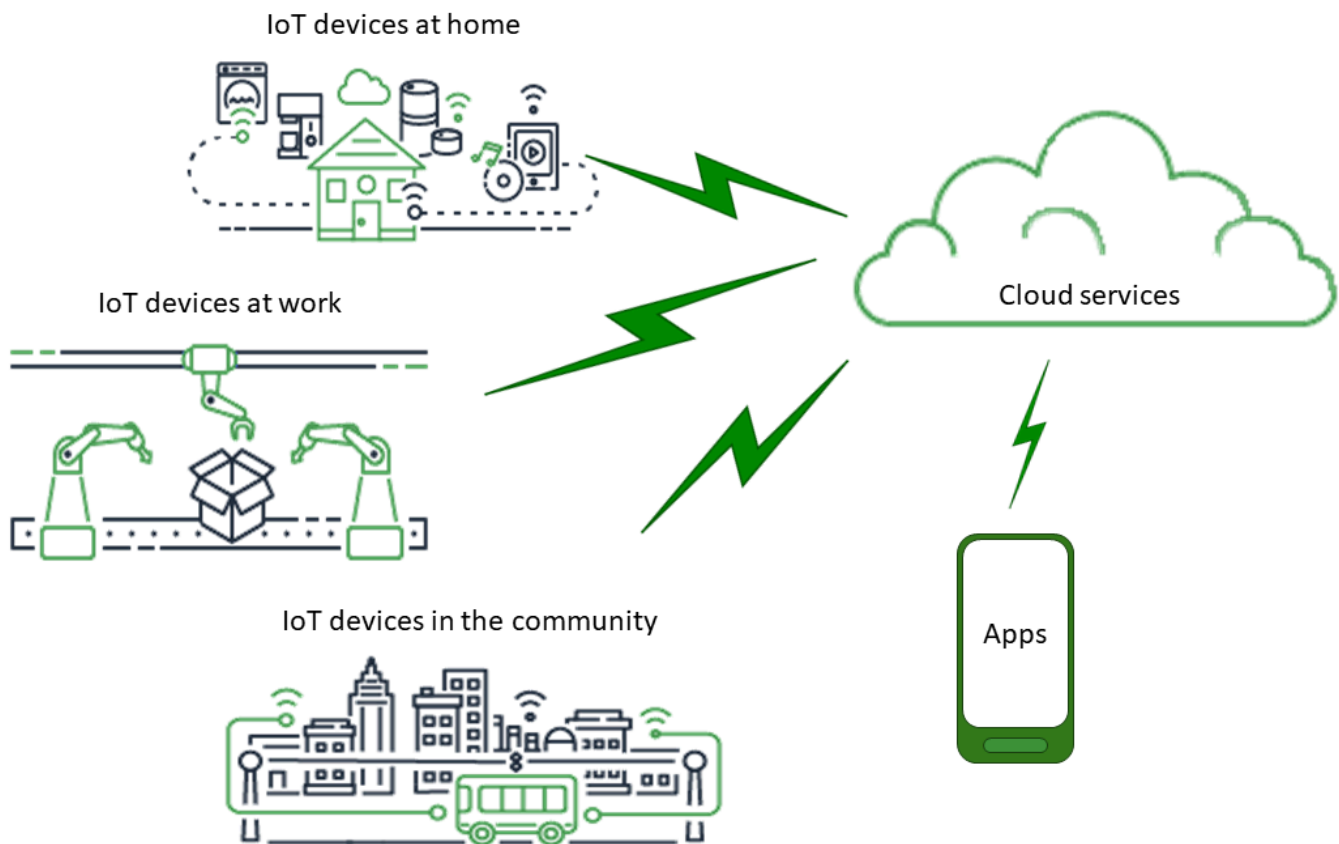
如需工業、消費者和商業使用案例的解決方案清單，請參閱 [AWS IoT 解決方案儲存庫](#)。

# 如何 AWS IoT 工作

AWS IoT 提供雲端服務和裝置支援，讓您用來實作 IoT 解決方案。AWS 提供許多雲端服務來支援物聯網應用程式。因此，為了幫助您了解從何處開始，本節提供了基本概念的圖表和定義，以便向您介紹 IoT 世界。

## IoT 的宇宙

一般而言，物聯網 (IoT) 是由此圖表所示的關鍵元件所組成。



## 應用程式

應用程式可讓最終使用者存取 IoT 裝置，以及這些裝置連接的雲端服務提供的功能。

## 雲端服務

雲端服務是連接到網際網路之分配式、大規模的資料儲存和處理服務。範例包括：

- IoT 連接和管理服務

AWS IoT 是 IoT 連線與管理服務的範例。

- 運算服務，例如 Amazon 彈性運算雲和 AWS Lambda
- 資料庫服務，例如 Amazon DynamoDB

## 通訊

裝置會使用各種技術和通訊協定與雲端服務進行通訊。範例包括：

- Wi-Fi/寬頻網際網路
- 寬頻行動數據
- 窄頻行動數據
- 遠距廣域網路 (LoRaWAN)
- 專屬 RF 通訊

## 裝置

裝置是一種管理介面和通訊的硬體類型。裝置通常位於所監控和控制的真實世界介面附近。裝置可包含運算和儲存資源，例如微控制器、CPU、記憶體。範例包括：

- Raspberry Pi
- Arduino
- 語音介面助理
- LoRa廣域網路和裝置
- Amazon Sidewalk 裝置
- 自訂 IoT 裝置

## 介面

介面是將裝置連接到實體世界的元件。

- 使用者介面

允許裝置和使用者的彼此通訊的元件。

- 文字輸入介面



讓使用者可以和裝置通訊

範例：鍵盤、按鈕

- 輸出介面

讓裝置可以和使用者通訊

範例：字母數字顯示器、圖形顯示器、指示燈、鬧鐘

- 感測器

測量或感測外部世界中的某些東西，並以裝置能理解的方式輸入的元件。範例包括：

- 溫度感測器 (將溫度轉換為類比或數位訊號)
- 濕度感測器 (將相對濕度轉換為類比或數位訊號)
- 類比數位轉換器 (將類比電壓轉換為數值)
- 超音波距離測量單元 (將距離轉換為數值)
- 光學感測器 (將光電位轉換為數值)
- 相機 (將影像資料轉換為數位資料)

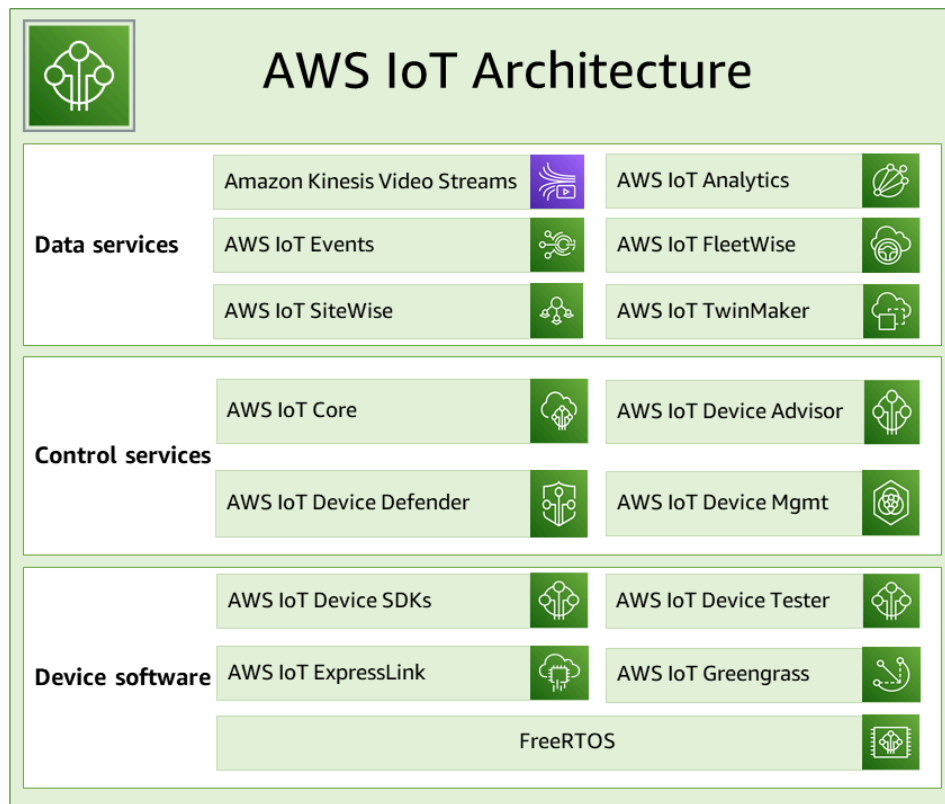
- 致動器

輸出元件，裝置可使用該元件控制外部世界的東西。範例包括：

- 步進馬達 (將電信號轉換為移動)
- 繼電器 (控制高電壓和電流)

## AWS IoT 服務概述

在 IoT 宇宙中，AWS IoT 提供支持與世界交互的設備以及在它們之間傳遞的數據的服務 AWS IoT。AWS IoT 由此圖例中顯示的服務組成，以支援您的 IoT 解決方案。



## AWS IoT 裝置軟體

AWS IoT 提供此軟體來支援您的 IoT 裝置。

### AWS IoT 裝置開發套件

[AWS IoT 裝置和行動 SDK](#) 可協助您有效率地將裝置連線到 AWS IoT。AWS IoT 裝置和行動 SDK 包含開放原始碼程式庫、含範例的開發人員指南，以及移植指南，讓您可以在自選的硬體平台上建置創新的 IoT 產品或解決方案。

### AWS IoT Device Tester

[AWS IoT Device Tester](#) 適用於 FreeRTOS，AWS IoT Greengrass 是用於微控制器的測試自動化工具。AWS IoT Device Tester 測試您的裝置，以判斷裝置是否會執行 FreeRTOS 或 AWS IoT Greengrass 與服務互通。AWS IoT

### AWS IoT ExpressLink

AWS IoT ExpressLink 為 [AWS 合作夥伴](#) 開發和提供的一系列硬件模塊提供支持。連線模組包含 AWS 經過驗證的軟體，可讓您更快、更輕鬆地將裝置安全地連接至雲端，並與各種 AWS 服務無縫整合。如需詳細資訊，請造訪 [AWS IoT ExpressLink 概觀頁面](#) 或參閱 [AWS IoT ExpressLink 式設計師指南](#)。

## AWS IoT Greengrass

[AWS IoT Greengrass](#) 延伸 AWS IoT 至邊緣裝置，以便它們可以在本機上根據其產生的資料採取行動、根據機器學習模型執行預測，以及篩選和彙總裝置資料。AWS IoT Greengrass 讓您的裝置能夠在更接近資料產生位置的位置收集和分析資料、自動回應本機事件，以及與區域網路上的其他裝置安全通訊。您可以使用預先建置的軟體模組 (稱為元件) AWS IoT Greengrass 來建置邊緣應用程式，這些模組可將邊緣裝置連接至 AWS 服務或協力廠商服務。

## FreeRTOS

[FreeRTOS](#) 是一個適用於微控制器的開放原始碼即時作業系統，可讓您將小型、低功耗邊緣裝置納入您的 IoT 解決方案。FreeRTOS 包含一個內核，以及一組不斷增長，並能支援許多應用程式的程式庫。FreeRTOS 系統可安全地將您的小型低功耗裝置連接至 [AWS IoT](#)，並支援執行 [AWS IoT Greengrass](#) 且功能更強大的邊緣裝置。

## AWS IoT 控制服務

Connect 至下列 AWS IoT 服務以管理 IoT 解決方案中的裝置。

### AWS IoT Core

[AWS IoT Core](#) 是一種託管雲服務，可讓連接的設備與雲應用程序和其他設備安全地進行交互。AWS IoT Core 可以支持許多設備和消息，並且可以處理這些消息並將其路由到 AWS IoT 端點和其他設備。使用時 AWS IoT Core，您的應用程式可以與您的所有裝置互動，即使它們未連線也是如此。

### AWS IoT Core 裝置顧問

[AWS IoT Core Device Advisor](#) 是一種以雲端為基礎的全受管測試功能，可在裝置軟體開發期間驗證 IoT 裝置。Device Advisor 提供預先建置的測試，您可以在將裝置部署到生產環境之前 AWS IoT Core，先驗證 IoT 裝置是否具有可靠且安全的連線能力。

### AWS IoT 設備後衛

[AWS IoT 裝置防禦者](#) 可協助您保護 IoT 裝置的安全。AWS IoT 裝置 Defender 會持續稽核您的 IoT 組態，以確保它們不會偏離安全性最佳做法。AWS IoT Device Defender 在偵測到可能造成安全風險的 IoT 組態中的任何間隙時傳送警示，例如在多個裝置上共用身分憑證或具有嘗試連線的已撤銷身分憑證的裝置。[AWS IoT Core](#)

### AWS IoT 設備管理

[AWS IoT 裝置管理](#) 服務可協助您追蹤、監控及管理組成裝置叢集的大量連線裝置。AWS IoT 裝置管理服務可協助您確保 IoT 裝置在部署完成後正常且安全地運作。它們還提供安全通道功能，以存

取您的裝置、監控其健全狀況、偵測和遠端進行問題疑難排解，以及管理裝置軟體和韌體更新的服務。

## AWS IoT 數據服務

分析 IoT 解決方案中裝置的資料，並使用下列 AWS IoT 服務採取適當的行動。

### Amazon Kinesis Video Streams

[Amazon Kinesis Video Streams](#) 可讓您將即時影片從裝置串流到 AWS 雲端，並在雲端持久存放、加密和編製索引，讓您透過 easy-to-use API 存取資料。您可以使用 Amazon Kinesis Video Streams 從數百萬個來源擷取大量即時影片資料，包括智慧型手機、監視攝影機、網路攝影機、汽車內建攝影機、空拍機及其他來源。Amazon Kinesis Video Streams 使您能夠播放影片，進行即時和隨選點播，並通過 Amazon Rekognition Video 和 ML 架構資料庫整合，快速利用視覺化運算和影片分析的方式建置應用程式。您也可以傳送非影片時間序列化資料，例如音訊資料、熱影像、深度資料、RADAR 資料等。

### Amazon Kinesis Video Streams with WebRTC

[Amazon Kinesis Video Streams with WebRTC](#) 以全受管功能提供符合標準的 WebRTC 實作。您可以使用 Amazon Kinesis Video Streams with WebRTC 安全地即時串流媒體，或在任何攝影機 IoT 裝置與符合 WebRTC 標準的行動或 Web 播放器之間，執行雙向音訊或視訊互動。因為是全受管功能，您不需要建置、執行或擴展任何與 WebRTC 相關的雲端基礎設施，例如訊號或媒體轉送伺服器，即可在應用程式和裝置之間安全地串流媒體。將 Amazon Kinesis Video Streams 與 WebRTC 搭配使用，您可以輕鬆建立即時 peer-to-peer 媒體串流的應用程式，或針對各種使用案例在攝影機 IoT 裝置、網頁瀏覽器和行動裝置之間進行即時音訊或視訊互動。

### AWS IoT 分析

[AWS IoT 分析](#) 可讓您針對大量非結構化 IoT 資料，有效率地執行和操作複雜的分析。AWS IoT Analytics 可自動化分析來自 IoT 裝置資料所需的每個困難步驟。AWS IoT Analytics 會篩選、轉換和豐富 IoT 資料，然後再將其儲存在時間序列資料存放區中進行分析。您可以使用內建的 SQL 查詢引擎或機器學習，執行一次性或排程查詢來分析資料。

### AWS IoT 活動

[AWS IoT 事件](#) 可偵測並回應來自 IoT 感應器和應用程式的事件。事件是識別比預期更複雜情況的資料模式，例如運動偵測器使用移動訊號來啟動燈光和監控攝影機。AWS IoT 事件會持續監控來自多個 IoT 感應器和應用程式的資料，並與 IoT AWS IoT Core SiteWise、DynamoDB 等其他服務整合，以提供早期偵測和獨特洞察。

## AWS IoT FleetWise

[AWS IoT FleetWise](#) 是一項託管服務，您可以使用它近乎即時地收集車輛資料並將其傳輸到雲端。使用 AWS IoT FleetWise，您可以輕鬆地從使用不同協議和數據格式的車輛中收集和組織數據。AWS IoT FleetWise 有助於將低級消息轉換為人類可讀的值，並將雲中的數據格式標準化以進行數據分析。您還可以定義資料蒐集的方案，控制要蒐集的車輛資料，以及何時將資料傳輸至雲端。

## AWS IoT SiteWise

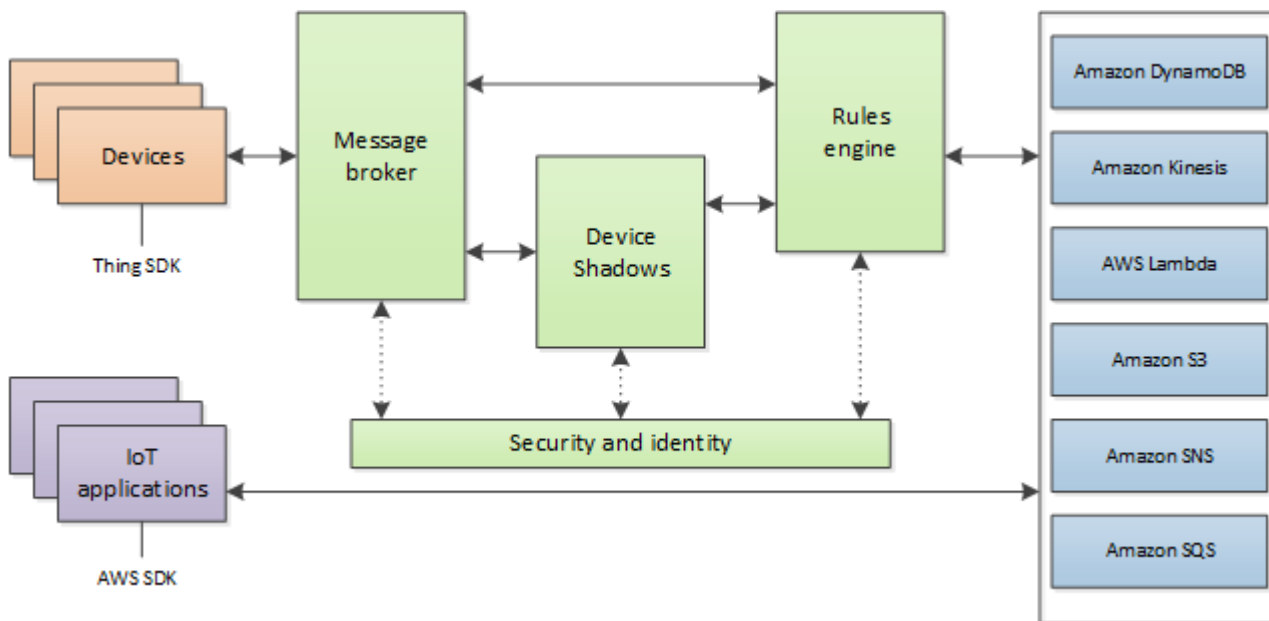
[AWS IoT SiteWise](#) 透過提供在設施閘道上執行的軟體，大規模收集、儲存、組織和監控 MQTT 訊息或 API 從工業設備傳送的資料。閘道安全地連接到您的內部部署資料伺服器，並自動化收集和組織資料並將其傳送到 AWS 雲端的程序。

## AWS IoT TwinMaker

[AWS IoT TwinMaker](#) 構建物理和數字系統的操作數字雙胞胎。AWS IoT TwinMaker 使用來自各種真實世界感測器、攝影機和企業應用程式的量測和分析來建立數位視覺化，協助您追蹤實體工廠、建築物或工業工廠。您可以使用實際資料監控作業、診斷和糾正錯誤以及優化作業。

## AWS IoT Core 服務

AWS IoT Core 提供將 IoT 裝置連接至雲端的服務，讓其他 AWS 雲端服務和應用程式可以與您的網際網路連線裝置互動。



下一節說明圖例中顯示的每個 AWS IoT Core 服務。

## AWS IoT Core 訊息服務

AWS IoT Core 連接服務提供與 IoT 設備的安全通信，並管理它們之間傳遞的消息 AWS IoT。

### 裝置閘道

讓裝置以安全有效的方式與 AWS IoT 通訊。裝置通訊由使用 X.509 憑證的安全通訊協定保護。

### 訊息代理程式

提供安全的機制，讓裝置和 AWS IoT 應用程式彼此發佈和接收訊息。您可以直接使用 MQTT 協議，也可以使用 MQTT WebSocket 來發布和訂閱。如需 AWS IoT 支援的通訊協定的詳細資訊，請參閱 [the section called “裝置通訊協定”](#)。裝置和用戶端也可使用 HTTP REST 介面，將資料發佈至訊息代理程式。

訊息代理程式會將裝置資料散佈到已訂閱裝置的裝置，以及其他 AWS IoT Core 服務 (例如 Device Shadow 服務和規則引擎)。

### AWS IoT Core 適用於 LoRa WAN

AWS IoT Core 對於 LoRa WAN，可以通過將 LoRa WAN 設備和閘道連接到設置私有 LoRa WAN 網絡，AWS 而無需開發和操作 LoRa WAN 網絡服務器 ( LNS )。從 LoRa WAN 裝置接收到的訊息會傳送至規則引擎，在其中可以格式化並傳送至其他 AWS IoT 服務。

### 規則引擎

規則引擎將訊息代理程式的資料連接至其他 AWS IoT 服務，以進行儲存和額外處理。例如，您可以插入、更新或查詢 DynamoDB 資料表，或根據您在規則引擎中定義的運算式呼叫 Lambda 函數。您可使用 SQL 型語言從訊息承載選取資料，然後處理資料並傳送至其他服務，例如 Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB 和 AWS Lambda。您還可建立將訊息重新發佈至訊息代理程式及其他訂閱者的規則。如需詳細資訊，請參閱 [的規則 AWS IoT](#)。

## AWS IoT Core 控制服務

AWS IoT Core 控制服務提供裝置安全性、管理和註冊功能。

### 自訂身分驗證服務

您可以定義自訂授權方，使用自訂身分驗證服務和 Lambda 函數，藉以管理自己的身分驗證和授權策略。自定義授權者允許使用承載令牌身份驗證和授權策略 AWS IoT 來驗證您的設備並授權操作。

自訂授權方可以實作各種身分驗證策略；例如 JSON Web 權杖驗證或 OAuth 提供者標註。它們必須傳回裝置閘道用來授權 MQTT 操作的政策文件。如需詳細資訊，請參閱 [自訂身分驗證和授權](#)。

## 裝置佈建服務

可讓您使用描述裝置所需資源 (物件物件、憑證、一或多項政策) 的範本，以佈建裝置。物件物件是登錄檔中的項目，內含描述裝置的屬性。裝置會使用憑證進行驗證 AWS IoT。政策決定裝置在 AWS IoT 可執行的操作。

範本包含了被字典 (對應) 之值取代的變數。只要在字典中為範本的變數傳入不同的值，您就可以使用相同的範本來佈建多個裝置。如需詳細資訊，請參閱 [裝置佈建](#)。

## 群組登錄檔

您可利用群組，將多個裝置分類至群組來同時管理。群組也可包含群組：您可建立群組階層。您對父群組執行的任何動作都會套用至其子群組。同樣的動作也適用於父群組中的所有裝置，以及子群組中的所有裝置。授予群組的許可也會套用至該群組及其所有子群組中的所有裝置。如需詳細資訊，請參閱 [管理裝置 AWS IoT](#)。

## 任務服務

可讓您定義一組遠端操作，這組操作會傳送到連接至 AWS IoT 的一個或多個裝置，並在其上執行。例如，您可以定義一個任務，指示一組裝置下載並安裝應用程式或韌體更新、重新啟動、輪換憑證，或者執行遠端故障排除操作。

欲建立任務，您必須指定要執行的遠端操作之描述，以及應執行操作的目標清單。目標可以是個別裝置、群組或兩者。如需詳細資訊，請參閱 [任務](#)。

## 登錄檔

整理 AWS 雲端中與每項裝置相關聯的資源。您可登錄您的裝置，且每項裝置最多可與三個自訂屬性相關聯。您也可以建立憑證和 MQTT 用戶端 ID 與每項裝置的關聯，提升您管理裝置及排解其問題的能力。如需詳細資訊，請參閱 [管理裝置 AWS IoT](#)。

## 安全與身分服務

對 AWS 雲端中的安全性承擔共同的責任。您的裝置必須保護其憑證，才可以將資料安全地傳送至訊息代理程式。訊息代理程式和規則引擎使用 AWS 安全功能將資料安全傳送至裝置或其他 AWS 服務。如需詳細資訊，請參閱 [身分驗證](#)。

## AWS IoT Core 數據服務

資 AWS IoT Core 料服務可協助您的 IoT 解決方案提供可靠的應用程式體驗，即使裝置並不總是連線。

## 裝置影子

用於存放及擷取裝置目前狀態資訊的 JSON 文件。

### Device Shadow 服務

Device Shadow 服務會維護裝置的狀態，無論裝置是否在線上，都能讓應用程式與裝置通訊。裝置離線時，Device Shadow 服務將為連線應用程式管理該裝置的資料。當裝置重新連線時，它會將自己的狀態與 Device Shadow 服務中的影子同步。您的裝置還可將其目前狀態發佈至影子，以供應用程式或其他可能並非隨時隨地連線的裝置使用。如需詳細資訊，請參閱 [AWS IoT Device Shadow 服務](#)。

## AWS IoT Core 支援服務

### Amazon 人行道集成 AWS IoT Core

[Amazon Sidewalk](#) 是個共享網路，可改善連線選項，協助裝置更好地共同運作。Amazon Sidewalk 支援範圍廣泛的客戶裝置，例如可找到寵物或貴重物品的裝置、提供智慧家庭安全和照明控制的裝置，及為設備和工具提供遠端診斷的裝置。Amazon 人行道整合可 AWS IoT Core 讓裝置製造商將其人行道裝置叢集新增至雲端 AWS IoT。

如需詳細資訊，請參閱 [AWS IoT Core for Amazon Sidewalk](#)。

## 進一步了解 AWS IoT

本主題可協助您熟悉的世界 AWS IoT。您可以取得有關如何在各種使用案例中應用 IoT 解決方案的一般資訊、訓練資源、社交媒體連結以 AWS IoT 及所有其他 AWS 服務，以及 AWS IoT 使用的服務和通訊協定清單。

### 培訓資源 AWS IoT

我們提供這些訓練課程，協助您瞭解以 AWS IoT 及如何將其應用於解決方案設計。

- [簡介到 AWS IoT](#)

視頻概述 AWS IoT 及其核心服務。

- [深入了解 AWS IoT 身份驗證和授權](#)

探索 AWS IoT 認證和授權概念的進階課程。您將學習如何驗證和授權用戶端存取 AWS IoT 控制平面和資料平面 API。



- [物聯網基礎系列](#)

IoT 線上學習模組的學習路徑，討論不同的 IoT 技術和功能。

## AWS IoT 資源和指南

這些是有關的特定方面的深入技術資源 AWS IoT。

- [IoT 鏡頭 — AWS IoT Well-Architected 的框架](#)

說明在 AWS 上架構 IoT 應用程式的最佳做法的文件。

- [為下列項目設計 MQTT 主題 AWS IoT Core](#)

白皮書，說明在 MQTT 中設計 MQTT 主題 AWS IoT Core 並利用 MQTT AWS IoT Core 功能的最佳做法。

- [摘要和介紹](#)

描述佈建大型裝置叢集的 AWS IoT 不同方式的 PDF 文件。

- [AWS IoT Core Device Advisor](#)

AWS IoT Core Device Advisor 提供預先建置的測試，您可以在將裝置部署到生產環境之前 AWS IoT Core，使用這些測試驗證 IoT 裝置是否具有可靠且安全的連線最佳實務。

- [AWS IoT 資源](#)

IoT 特定資源，例如技術指南、參考架構、電子書和彙整過的部落格文章，以可搜尋索引呈現。

- [物聯網圖集](#)

如何解決常見的 IoT 設計問題概述。IoT 圖集提供您在開發 IoT 解決方案時可能遇到的設計挑戰的深入探討。

- [AWS 白皮書與指南](#)

我們目前收集的白皮書和指南以 AWS IoT 及其他 AWS 技術。

## AWS IoT 在社交媒體

這些社交媒體渠道提供有關 AWS IoT 和 AWS 相關主題的信息。

- [物聯網上 AWS IoT — 官方博客](#)

- [AWS IoT Amazon Web Services 頻道中的視頻 YouTube](#)

這些社交媒體帳戶涵蓋所有 AWS 服務，包括 AWS IoT

- [Amazon Web Services 渠道 YouTube](#)
- [推特上的 Amazon Web Services](#)
- [臉書上的 Amazon Web Services](#)
- [Instagram 上的 Amazon Web Services](#)
- [Amazon Web Services LinkedIn](#)

## AWS IoT Core 規則引擎使用的服務

AWS IoT Core 規則引擎可以連線到這些 AWS 服務。

- [Amazon DynamoDB](#)

Amazon DynamoDB 是可擴展的 NoSQL 資料庫服務，提供快速且可預測的資料庫效能。

- [Amazon Kinesis](#)

Amazon Kinesis 可輕鬆收集、處理和分析即時串流資料，讓您及時取得洞見並快速回應新資訊。Amazon Kinesis 可擷取即時資料，例如影片、音訊、應用程式記錄、網站點擊流和 IoT 遙測資料，用於機器學習、分析及其他應用。

- [AWS Lambda](#)

AWS Lambda 可讓您執行程式碼，而無需佈建或管理伺服器。您可以將程式碼設定為從 AWS IoT 資料和事件自動觸發，或直接從 Web 或行動應用程式呼叫程式碼。

- [Amazon Simple Storage Service](#)

Amazon Simple Storage Service (Amazon S3) 可以隨時從 Web 上的任何位置存放和擷取任意數量的資料。AWS IoT 規則可以將資料傳送到 Amazon S3 以進行儲存。

- [Amazon Simple Notification Service](#)

Amazon Simple Notification Service (Amazon SNS) 是一種 Web 服務，可讓應用程式、最終使用者和裝置傳送和接收雲端的通知。

- [Amazon Simple Queue Service](#)

Amazon Simple Queue Service (Amazon SQS) 是一種訊息佇列服務，可解偶和擴展微服務、分配式系統和無伺服器應用程式。

- [Amazon OpenSearch 服務](#)

Amazon Ser OpenSearch vice (Ser OpenSearch vice) 是一種受管服務，可讓您輕鬆部署、操作和擴展 OpenSearch，這是一種熱門的開放原始碼搜尋和分析引擎。

- [Amazon SageMaker](#)

Amazon SageMaker 可以透過尋找 IoT 資料中的模式來建立機器學習 (ML) 模型。這個服務使用這些模型處理新資料，並為應用程式產生預測結果。

- [Amazon CloudWatch](#)

Amazon CloudWatch 提供可靠、可擴展且靈活的監控解決方案，以協助設定、管理和擴展您自己的監控系統和基礎設施。

## AWS IoT Core支援的通訊協定

這些主題提供 AWS IoT所使用通訊協定的詳細資訊。如需有關裝置 AWS IoT 和服務所使用之通訊協定的詳細資訊 AWS IoT，請參閱[連接到 AWS IoT Core](#)。

- [MQTT \(訊息佇列遙測傳輸\)](#)

在 MQTT.org 網站的首頁上，您可以找到 MQTT 通訊協定規格。若要取得有關如何 AWS IoT 支援 MQTT 的更多資訊，請參閱。[MQTT](#)

- [HTTPS \(超文字傳輸通訊協定：安全\)](#)

裝置和應用程式可以使用 HTTPS 存取 AWS IoT 服務。

- [LoRaWAN \(長距離廣域網路\)](#)

LoRaWAN 裝置和閘道可透 AWS IoT Core 過使 AWS IoT Core 用 LoRa WAN 連線到。

- [TLS \(傳輸層安全性\) 第 1.3 版](#)

TLS 第 1.3 版 (RFC 5246) 的規格。AWS IoT 使用 TLS v1.3 在裝置和 AWS IoT. 之間建立安全連線。

## AWS IoT 主控台的最新消息

我們正在更新 AWS IoT 主控台的使用者介面，以取得全新體驗。我們正在分階段更新使用者介面，因此主控台內的某些頁面將有新的體驗，有些可能同時具有原始和新的體驗，有些可能只有原始體驗。

此表格顯示截至 2022 年 1 月 27 日 AWS IoT 主控台使用者介面個別區域的狀態。

### AWS IoT 主控台使用者介面狀態

主控台頁面	原始體驗	全新體驗	說明
監控	不適用	可用性	
活動	不適用	可用性	
加入：開始使用	不適用	可用性	不適用於 CN 區域
加入：機群佈建範本	可用性	可用性	
管理：物件	可用性	可用性	
管理 - 類型	可用性	可用性	
管理：物件群組	可用性	可用性	
管理：帳單群組	可用性	可用性	
管理：任務	可用性	可用性	
管理：任務範本	不適用	可用性	
管理：通道	不適用	可用性	
Fleet Hub：開始使用	不適用	可用性	並非所有 AWS 區域都支援
Fleet Hub：應用程式	不適用	可用性	並非所有 AWS 區域都支援
Greengrass：開始使用	不適用	可用性	並非所有 AWS 區域都支援

主控台頁面	原始體驗	全新體驗	說明
Greengrass：核心裝置	不適用	可用性	並非所有 AWS 區域都支援
Greengrass：元件	不適用	可用性	並非所有 AWS 區域都支援
Greengrass：部署	不適用	可用性	並非所有 AWS 區域都支援
Greengrass：經典 (V1)	可用性	可用性	
無線連線：簡介	不適用	可用性	並非所有 AWS 區域都支援
無線連線：閘道	不適用	可用性	並非所有 AWS 區域都支援
無線連線：裝置	不適用	可用性	並非所有 AWS 區域都支援
無線連線：設定檔	不適用	可用性	並非所有 AWS 區域都支援
無線連線：目的地	不適用	可用性	並非所有 AWS 區域都支援
安全：憑證	可用性	可用性	
安全：政策	可用性	可用性	
安全：CA	可用性	可用性	
安全：角色別名	可用性	可用性	
安全：授權者	可用性	可用性	
防禦：簡介	不適用	可用性	

主控台頁面	原始體驗	全新體驗	說明
防禦：稽核	不適用	可用性	
防禦：偵測	不適用	可用性	
防禦：緩解動作	不適用	可用性	
防禦：設定	不適用	可用性	
動作：規則	可用性	可用性	
動作：目的地	可用性	可用性	
測試：Device Advisor	可用性	可用性	並非所有 AWS 區域都支援
測試：MQTT 測試用戶端	可用性	可用性	
軟體	可用性	可用性	
設定	不適用	可用性	
了解	可用性	尚無法使用	

## 圖例

### 狀態值

- 可用性

可使用此使用者介面體驗。

- 不適用

無法使用此使用者介面體驗。

- 尚無法使用

正在開發新的使用者介面體驗，但尚未準備好。

- In progress (正在進行)

新的使用者介面體驗正在更新中。但是，某些網頁可能仍保有原始的使用者體驗。

## 搭 AWS IoT 配 AWS SDK 使用

AWS 軟體開發套件 ( SDK ) 可用於許多流行的編程語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
<a href="#">AWS SDK for C++</a>	<a href="#">AWS SDK for C++ 程式碼範例</a>
<a href="#">AWS CLI</a>	<a href="#">AWS CLI 程式碼範例</a>
<a href="#">AWS SDK for Go</a>	<a href="#">AWS SDK for Go 程式碼範例</a>
<a href="#">AWS SDK for Java</a>	<a href="#">AWS SDK for Java 程式碼範例</a>
<a href="#">AWS SDK for JavaScript</a>	<a href="#">AWS SDK for JavaScript 程式碼範例</a>
<a href="#">適用於 Kotlin 的 AWS SDK</a>	<a href="#">適用於 Kotlin 的 AWS SDK 程式碼範例</a>
<a href="#">AWS SDK for .NET</a>	<a href="#">AWS SDK for .NET 程式碼範例</a>
<a href="#">AWS SDK for PHP</a>	<a href="#">AWS SDK for PHP 程式碼範例</a>
<a href="#">AWS Tools for PowerShell</a>	<a href="#">PowerShell 程式碼範例工具</a>
<a href="#">AWS SDK for Python (Boto3)</a>	<a href="#">AWS SDK for Python (Boto3) 程式碼範例</a>
<a href="#">AWS SDK for Ruby</a>	<a href="#">AWS SDK for Ruby 程式碼範例</a>
<a href="#">適用於 Rust 的 AWS SDK</a>	<a href="#">適用於 Rust 的 AWS SDK 程式碼範例</a>
<a href="#">適用於 SAP ABAP 的 AWS SDK</a>	<a href="#">適用於 SAP ABAP 的 AWS SDK 程式碼範例</a>
<a href="#">適用於 Swift 的 AWS SDK</a>	<a href="#">適用於 Swift 的 AWS SDK 程式碼範例</a>

**i** 可用性範例

找不到所需的內容嗎？請使用本頁面底部的提供意見回饋連結申請程式碼範例。



# 開始使用 AWS IoT Core

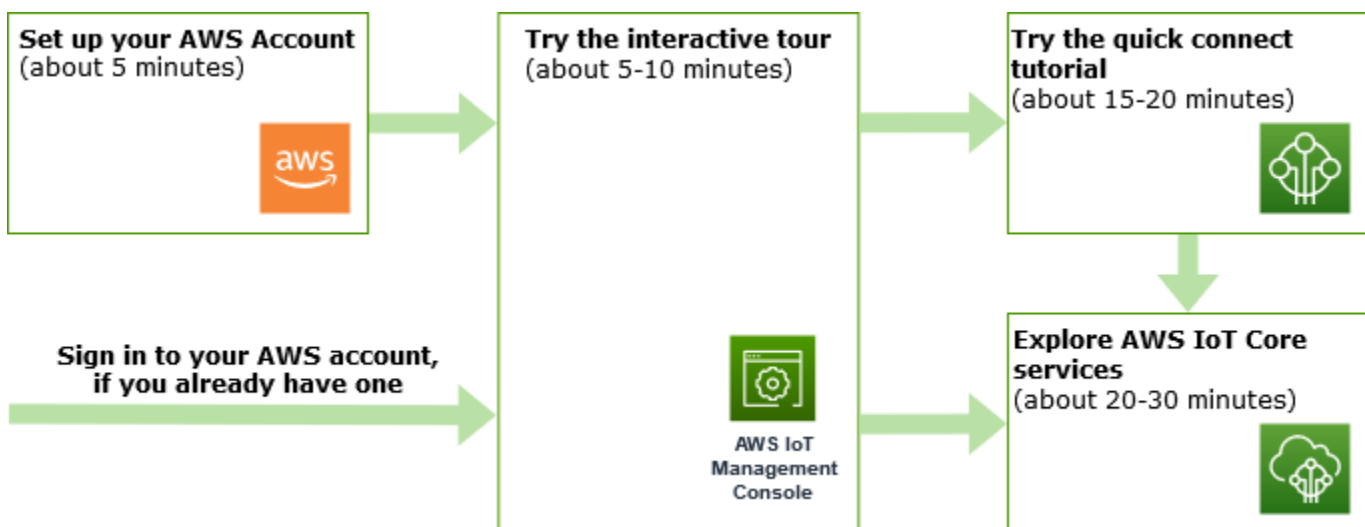
無論您是 IoT 新手還是擁有多年的經驗，這些資源都會提供可協助您開始使用的 AWS IoT 概念和術語 AWS IoT。

- 查看內部 AWS IoT 及其組件 [如何 AWS IoT 工作](#)。
- 從我們的培訓教材和影片集中 [進一步了解 AWS IoT](#)。本主題也包含 AWS IoT 可以連線的服務清單、社交媒體連結，以及通訊協定規格的連結。
- [the section called “將您的第一部裝置 Connect 到 AWS IoT Core”](#)。
- 透過 [連接到 AWS IoT Core](#) 並探索 [AWS IoT 教學課程](#) 來開發您的 IoT 解決方案。
- 使用 [Device Advisor](#) 來測試並驗證您的 IoT 裝置，以進行安全可靠的通訊。
- 使用 AWS IoT Core 管理服務 (例如 [機群索引](#)、[任務](#) 和 [AWS IoT Device Defender](#)) 來管理您的解決方案。
- 使用 [AWS IoT 數據服務](#) 來分析來自您裝置的資料。

## 將您的第一部裝置 Connect 到 AWS IoT Core

AWS IoT Core 服務可將 IoT 裝置連接至 AWS IoT 服務和其他 AWS 服務。AWS IoT Core 包括裝置閘道和訊息代理程式，可在 IoT 裝置和雲端之間連接和處理訊息。

以下是您可以開始使用 AWS IoT Core 和的方式 AWS IoT。



本節介紹了介紹其主 AWS IoT Core 要服務的導覽，並提供了如何將設備連接到設備 AWS IoT Core 並在它們之間傳遞消息的幾個示例。在裝置和雲端之間傳遞訊息是每個 IoT 解決方案的基礎，也是您的裝置與其他 AWS 服務互動的方式。

- [設定您的 AWS 帳戶](#)

在您可以使用 AWS IoT 服務之前，您必須先設定 AWS 帳戶。如果您已經有自己的 AWS 帳戶和 IAM 使用者，則可以使用它們並略過此步驟。

- [嘗試互動式教學課程](#)

如果您想在不連接設備或下載任何軟件的情況下查看基本 AWS IoT 解決方案可以做什麼，則此演示是最好的。互動式教學課程提供以 AWS IoT Core 服務為基礎的模擬解決方案，說明它們如何互動。

- [嘗試快速連接教學課程](#)


本教程是最好的，如果你想快速入門，看看它是如何在有限的情況下工作。AWS IoT 在本教程中，您將需要一個設備，並在其上安裝一些 AWS IoT 軟件。如果沒有 IoT 裝置，您可以使用 Windows、Linux 或 macOS 個人電腦，作為本教學課程的裝置。如果您想嘗試 AWS IoT，但沒有裝置，請嘗試下一個選項。

- [透過實作教學探索 AWS IoT Core 服務](#)

本教學課程最適合想要開始使用的開發人員，以 AWS IoT 便他們可以繼續探索其他 AWS IoT Core 功能，例如規則引擎和陰影。本教學課程遵循類似於快速連接教學課程的程序，但在每個步驟上提供更多詳細資訊，以便可以更順暢地轉換至其他進階教學課程。

- [使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)

了解如何使用 MQTT 測試用戶端，觀看您的第一部裝置將 MQTT 訊息發佈至 AWS IoT。MQTT 測試用戶端是監控和疑難排解裝置連線的實用工具。

 Note

如果想要嘗試其中一個以上的入門教學課程或重複相同的教學課程，您應該先刪除從先前教學課程建立的物件，然後再啟動另一個物件。如果未從先前的教學課程中刪除物件，您需要對後續教學課程使用不同的物件名稱。這是因為物件名稱在您的帳戶和 AWS 區域中必須是唯一的。

如需有關的詳細資訊 AWS IoT Core，請參閱[什麼是 AWS IoT Core？](#)

# 設定您的 AWS 帳戶

第一次使 AWS IoT Core 用前，請先完成下列工作：

## 主題

- [註冊一個 AWS 帳戶](#)
- [建立具有管理權限的使用者](#)
- [開啟主 AWS IoT 控制台](#)

## 註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 <https://portal.aws.amazon.com/billing/signup>。
2. 請遵循線上指示進行。

部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

當您註冊一個時 AWS 帳戶，將創建 AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

## 建立具有管理權限的使用者

註冊後，請保護 AWS 帳戶 AWS 帳戶根使用者、啟用和建立系統管理使用者 AWS IAM Identity Center，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶電子郵件地址，以帳戶擁有者身分登入。[AWS Management Console](#) 在下一頁中，輸入您的密碼。

如需使用根使用者登入的說明，請參閱 AWS 登入使用者指南中的[以根使用者身分登入](#)。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

如需指示，請參閱《IAM 使用者指南》中的[為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 \(主控台\)](#)。

### 建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

如需指示，請參閱 AWS IAM Identity Center 使用者指南中的[啟用 AWS IAM Identity Center](#)。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

[若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。](#)

### 以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

如需使用 IAM 身分中心使用者[登入的說明](#)，請參閱[使用AWS 登入 者指南中的登入 AWS 存取入口網站](#)。

### 指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[建立權限集](#)」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「[新增群組](#)」。

- [開啟主 AWS IoT 控制台](#)

如果您已經為自己擁有 AWS 帳戶 和用戶，則可以使用它們並跳到[the section called “開啟主 AWS IoT 控制台”](#)。

## 開啟主 AWS IoT 控制台

本節中大部分主控台導向的主題都是從主控台開始。AWS IoT 如果您尚未登入，請登入 AWS 帳戶，然後開啟[AWS IoT 主控台](#)並繼續下一個區段以繼續開始使用 AWS IoT。

## 試用 AWS IoT Core 互動式教學

互動式教學課程會顯示 AWS IoT 上建置之簡易 IoT 解決方案的元件。本教學課程的動畫顯示 IoT 裝置如何與 AWS IoT Core 服務互動。本主題提供 AWS IoT Core 互動式自學課程的預覽。主控台內的影像包含未出現在此教學課程中影像的動畫。

若要執行示範，您必須先 [the section called “設定您的 AWS 帳戶”](#)。但是，該教程不需要任何 AWS IoT 資源，其他軟件或任何編碼。

本示範預計大約需時 5-10 分鐘。請預留 10 分鐘，讓自己有充裕的時間來理解每一個步驟。

若要執行 AWS IoT Core 互動式教學課程

1. 在 AWS IoT 主控台中開啟[AWS IoT 首頁](#)。

在 AWS IoT 首頁上，於學習資源視窗窗格中選擇開始教學課程。

The screenshot shows the AWS IoT console interface. On the left is a navigation sidebar with a search bar and a list of menu items: Monitor, Connect (with sub-items: Connect one device, Connect many devices), Test (with sub-items: Device Advisor, MQTT test client, Device Location), Manage (with sub-items: All devices, Greengrass devices, LPWAN devices, Software packages, Remote actions, Message routing, Retained messages, Security, Fleet Hub), Device Software, Billing groups, Settings, Feature spotlight, and Documentation. Below the sidebar is a 'New console experience' button. The main content area has a dark header with the AWS IoT logo and the text 'Securely connect, test, and manage your IoT devices'. Below this is a 'How it works' section with three columns: 'Connect' (cloud icon), 'Test' (wrench icon), and 'Manage' (network icon). Each column has a brief description and a 'Learn more' link. Below that is a 'Watch it work' section with an 'Interactive tutorial' card. On the right side, there is a 'Learning resources' sidebar with a red box around the 'AWS IoT interactive tutorial' link, which is also pointed to by a red arrow. Other links in the sidebar include 'Get started with AWS IoT', 'Pricing', 'AWS IoT video resources', 'AWS IoT Developer Guide', and 'More resources'.

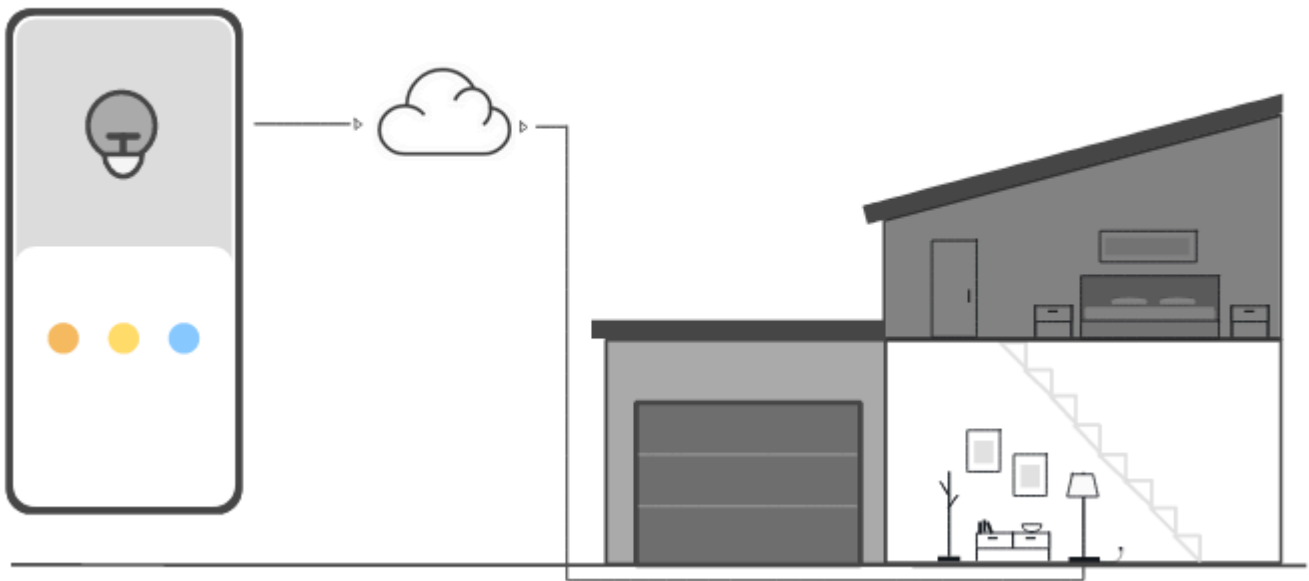
- 在AWS IoT 主控台教學課程頁面中，檢閱教學課程的各個章節，然後在您準備好繼續進行時，選擇開始章節。

下列各節說明「AWS IoT 主控台教學課程」如何 AWS IoT Core 提供這些功能：

- [連線 IoT 裝置](#)
- [儲存裝置離線狀態](#)
- [將裝置資料路由到服務](#)

## 連線 IoT 裝置

瞭解 IoT 裝置如何與之通訊 AWS IoT Core。

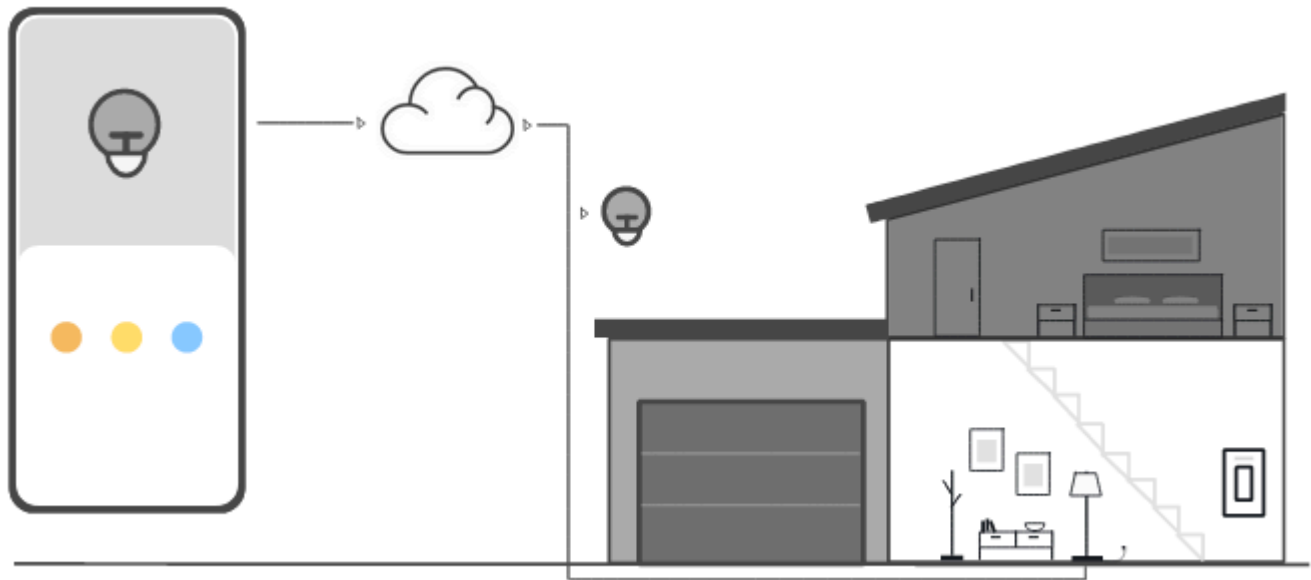


此步驟中的動畫顯示左側的控制器和右側房子中的智慧燈如何 AWS IoT Core 在雲端連接和通訊。該動畫顯示了與他們收到的消息進行通信 AWS IoT Core 和反應的設備。

如需將裝置連接至的詳細資訊 AWS IoT Core，請參閱[連接到 AWS IoT Core](#)。

## 儲存裝置離線狀態

瞭解裝置或應用程式離線時，如何 AWS IoT Core 儲存裝置狀態。



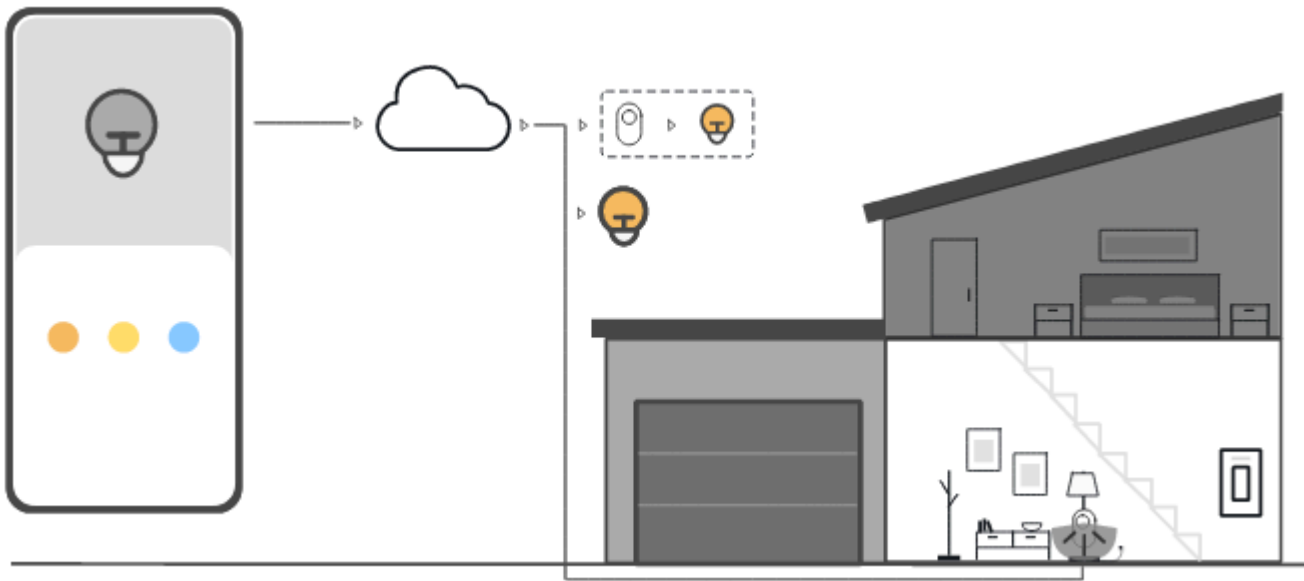
此步驟中的動畫顯示 Device Shadow 服務如何 AWS IoT Core 儲存控制裝置和智慧型指示燈的裝置狀態資訊。智慧燈離線時，裝置影子會儲存來自控制器的命令。

當智慧型指示燈重新連線至時 AWS IoT Core，會擷取這些指令。當控制器離線時，裝置影子會從智慧燈中儲存狀態信息。當控制器重新連接時，它會擷取智慧燈最新的狀態以更新其顯示內容。

如需裝置影子的詳細資訊，請參閱 [AWS IoT Device Shadow 服務](#)。

## 將裝置資料路由到服務

瞭解如何 AWS IoT Core 將裝置狀態傳送至其他 AWS 服務。



此步驟中的動畫顯示了如何使用 AWS IoT 規則 AWS IoT Core 將數據從設備發送到其他 AWS 服務。AWS IoT 規則會訂閱來自裝置的特定訊息、解譯這些訊息中的資料，並將解譯的資料路由到其他服務。在此範例中，AWS IoT 規則會解譯來自動作感應器的資料，並將指令傳送至 Device Shadow，然後將其傳送至智慧型燈泡。與先前的範例一樣，裝置影子存放控制器的裝置狀態資訊。

如需 AWS IoT 規則的詳細資訊，請參閱[規則 AWS IoT](#)。

## 嘗試 AWS IoT 快速連接

在本教學課程中，您將建立第一個物件、將裝置連接至其中，並觀看它傳送 MQTT 訊息。

您可以預期在本教學課程上花費 15-20 分鐘。

本教程最適合想要快速入門以了 AWS IoT 解其在有限場景下如何工作的人。如果您正在尋找讓您入門的範例，以便可以探索更多功能和服務，請嘗試 [在實作教學中探索 AWS IoT Core 服務](#)。

在本教學課程中，您將在連線至物件資源的裝置上下載並執行軟體，做 AWS IoT Core 為非常小型 IoT 解決方案的一部分。裝置可以是 IoT 裝置 (例如 Raspberry Pi)，也可以是正在執行 Linux、OS 和 OSX，或 Windows 的電腦。如果您想要將遠距離 WAN (LoRaWAN) 裝置連接到 AWS IoT，請參閱教學課程 [將裝置和閘道連接到 AWS IoT Core LoRa WAN](#)。

如果您的裝置支援可以執行 [AWS IoT 主控台](#) 的瀏覽器，我們建議您在該裝置上完成本教學課程。



**Note**

如果您的裝置沒有相容的瀏覽器，請在電腦上遵循本教學課程。當程序要求您下載檔案時，請將它下載到您的電腦，然後使用安全複製 (SCP) 或類似程序，將下載的檔案傳輸到您的裝置。

本教學課程需要您的 IoT 裝置，才能在 AWS 帳戶裝置的資料端點上與 8443 埠通訊。若要測試其是否可以存取該連接，請嘗試 [測試與裝置資料端點的連線](#) 中的操作程序。

## 步驟 1. 開始教學課程

如果可能，請在您的裝置上完成此程序；否則，請準備好稍後在此程序中將檔案傳輸到您的裝置。

若要開始進行教學課程，請登入 [AWS IoT 主控台](#)。在 AWS IoT 主機首頁的左側，選擇 [Connect 線]，然後選擇 [Connect 一個裝置]。

The screenshot displays the AWS IoT Core console interface. On the left is a navigation pane with the following sections:

- Monitor
- Connect**
  - Connect one device
  - ▶ Connect many devices
- Test
  - ▶ Device Advisor
  - MQTT test client
  - Device Location [New](#)
- Manage
  - ▶ All devices
  - ▶ Greengrass devices

The main content area is titled "How it works" and contains the following text:

Connect devices to AWS IoT so they can send and receive data. **Bold text** refers to an entry in the **Connect** menu of the navigation pane.

Two columns of information are shown:

- Connect one device**: The **Quick connect** wizard walks you through the steps to create the resources and download the software required to connect your IoT device to AWS IoT.
- Connect many devices**: **Fleet provisioning templates** define security policies and registry settings when a device connects to AWS IoT for the first time.

## 步驟 2. 建立物件

1. 在 Prepare your device (準備您的裝置) 區段中，依照畫面上的指示，準備您的裝置以連線至 AWS IoT。

**AWS IoT** ×

AWS IoT > Connect > Connect one device

Step 1  
**Prepare your device**

Step 2  
Register and secure your device

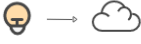


Step 3  
Choose platform and SDK

Step 4  
Download connection kit

Step 5  
Run connection kit

### Prepare your device [Info](#)

#### How it works

 →  → 

In this wizard, we'll be creating a thing resource in AWS IoT. A thing resource is a digital representation of a physical device or logical entity.

A thing resource uses certificates to secure communication between your device and AWS IoT. AWS IoT policies control access to the AWS IoT resources. This wizard creates the certificate and policy for your device.

When a device connects to AWS IoT, policies enable it to subscribe and publish MQTT messages with AWS IoT message broker.

#### Prepare your device

1. Turn on your device and make sure it's connected to the internet.
2. Choose how you want to load files onto your device.
  1. If your device supports a browser, open the AWS IoT console on your device and run this wizard. You can download the files directly to your device from the browser.
  2. If your device doesn't support a browser, choose the best way to transfer files from the computer with the browser to your device. Some options to transfer files include using the file transfer protocol (FTP) and using a USB memory stick.
3. Make sure that you can access a command-line interface on your device.
  1. If you're running this wizard on your IoT device, open a terminal window on your device to access a command-line interface.
  2. If you're not running this on your IoT device, open an SSH terminal window on this device and connect it to your IoT device.
4. From the terminal window, enter this command:

After you complete these steps and get a successful ping response, you're ready to continue and connect your device to AWS IoT.

Cancel **Next**

2. 在 Register and secure your device (註冊並保護您的裝置) 區段中，選擇 Create a new thing (建立新的物件) 或者 Choose an existing thing (選擇現有物件)。在 Thing name (物件名稱) 欄位中，為您的物件輸入名稱。在這個範例中使用的物件名稱為 **TutorialTestThing**。

#### **⚠ Important**

繼續之前，請仔細檢查您的物件名稱。

建立物件之後，無法變更物件名稱。如果想要變更物件名稱，您必須建立物件名稱正確的新物件，然後刪除名稱不正確的物件。

在 Additional configurations (其他組態) 區段中，使用列出的選用組態進一步自訂物件資源。

為物件提供名稱並選取任何其他組態後，請選擇 Next (下一步)。

3. 在 [選擇平台和 SDK] 區段中，選擇您要使用的 AWS IoT 裝置 SDK 的平台和語言。這個範例會使用 Linux/OSX 平台和 Python SDK。請確保在目標裝置上安裝了 python3 和 pip3，再繼續下一步。

#### Note

請務必在主控制台頁面底部檢查所選 SDK 所需的必要軟體清單。  
您必須在目標電腦上安裝必要的軟體，然後才能繼續下一個步驟。

在您選擇平台和裝置 SDK 語言之後，請選擇 Next (下一步)。

AWS IoT > Connect > Connect one device

Step 1  
Prepare your device

Step 2  
Register and secure your device


Step 3  
**Choose platform and SDK**

Step 4  
Download connection kit

Step 5  
Run connection kit

## Choose platform and SDK [Info](#)

### Choose the software for your device



This wizard helps you download a software development kit (SDK) to your device. AWS IoT supports Device SDKs that run on your device and include a sample program that publishes and subscribes to MQTT messages. AWS IoT supports Device SDKs in the languages shown below.

### Platform and SDK

Choose the platform OS and AWS IoT Device SDK that you want to use for your device.

**Device platform operating system**  
This is the operating system installed on the device that will connect to AWS.

- Linux / macOS**  
Linux version: any  
macOS version: 10.13+
- Windows**  
Version 10

**AWS IoT Device SDK**  
Choose a Device SDK that's in a language your device supports.

- Node.js**  
Version 10+  
Requires Node.js and npm to be installed
- Python**  
Version 3.6+  
Requires Python and Git to be installed
- Java**  
Version 8  
Requires Java JDK, Maven, and Git to be installed

Cancel Previous **Next**

## 步驟 3。將檔案下載到您的裝置

此頁面會在建立連線套件之後 AWS IoT 顯示，其中包含您的裝置所需的下列檔案和資源：

- 物件的憑證檔案，用來驗證裝置
- 政策資源，用來授權您的物件可與 AWS IoT 互動
- 下載 AWS 裝置 SDK 並在裝置上執行範例程式的指令碼

1. 當您準備好繼續時，請選擇 Download connection kit for (下載連線套件) 按鈕，為您先前選擇的平台下載適用的連線套件。

AWS IoT > Connect > Connect one device

Step 1  
Prepare your device

Step 2  
Register and secure your device



Step 3  
Choose platform and SDK

Step 4  
**Download connection kit**

Step 5  
Run connection kit

## Download connection kit Info

### Install the software on your device

 →  We created the AWS IoT resources that your device needs to connect to AWS IoT. We also created a connection kit that includes the resources in a zipped file that you need to install on your device. The resources in the connection kit are listed below. In this step, you'll install them on your device.


### Connection kit

Certificate TutorialTestThing.cert.pem	Private key TutorialTestThing.private.key	AWS IoT Device SDK Python
Script to send and receive messages start.sh	Policy TutorialTestThing-Policy <a href="#">View policy</a>	



### Download


If you are running this from a browser on the device, after you download the connection kit, it will be in the browser's download folder.

If you are not running this from a browser on your device, you'll need to transfer the connection kit from your browser's download folder to your device using the method you tested when you prepared your device in step 1.

 **Download connection kit**

### Unzip connection kit on your device

  After the connection kit is on your device, unzip it using this command:

 Copy

Cancel Previous **Next**

- 如果您是在裝置上執行此程序，請將連線套件檔案儲存到可以從中執行命令列命令的目錄。

如果您不是在裝置上執行此程序，請將連線套件檔案儲存至本機目錄，然後將檔案傳輸至您的裝置。

- 在 Unzip connection kit on your device (解壓縮裝置上的連線套件) 區段中，輸入連線套件檔案所在目錄中的 `unzip connect_device_package.zip`。

如果您使用的是 Windows PowerShell 命令視窗，但該 `unzip` 命令不起作用，請取代之為 `expand-archive`，然後再次嘗試命令列。

#### 4. 在裝置上具有連線套件檔案之後，請選擇 Next (下一步) 來繼續教學課程。

AWS IoT > Connect > Connect one device

Step 1  
Prepare your device

Step 2  
Register and secure your device

Step 3  
Choose platform and SDK

Step 4  
**Download connection kit**

Step 5  
Run connection kit

### Download connection kit [Info](#)

#### Install the software on your device

We created the AWS IoT resources that your device needs to connect to AWS IoT. We also created a connection kit that includes the resources in a zipped file that you need to install on your device. The resources in the connection kit are listed below. In this step, you'll install them on your device.

#### Connection kit

Certificate TutorialTestThing.cert.pem	Private key TutorialTestThing.private.key	AWS IoT Device SDK Python
Script to send and receive messages start.sh	Policy TutorialTestThing-Policy <a href="#">View policy</a>	

#### Download

If you are running this from a browser on the device, after you download the connection kit, it will be in the browser's download folder.

If you are not running this from a browser on your device, you'll need to transfer the connection kit from your browser's download folder to your device using the method you tested when you prepared your device in step 1.

[Download connection kit](#)

#### Unzip connection kit on your device

After the connection kit is on your device, unzip it using this command:

```
unzip connect_device_package.zip
```

[Copy](#)

Cancel Previous **Next**

## 步驟 4. 執行範例

您可以在裝置上的終端機或命令視窗中執行此程序，同時遵循主控台中顯示的指示。您主控台中看見的命令是適用於 [the section called “步驟 2. 建立物件”](#) 中所選作業系統的命令。這裡顯示的命令適用於 Linux/OSX 作業系統。

1. 在裝置的終端機或指令視窗中，在具有連線套件檔案的目錄中，執行 AWS IoT 主控台中顯示的步驟。

AWS IoT > Connect > Connect one device

Step 1  
Prepare your device

Step 2  
Register and secure your device

Step 3  
Choose platform and SDK

Step 4  
Download connection kit

Step 5  
**Run connection kit**

## Run connection kit Info

### How to display messages from your device

**Step 1: Add execution permissions**  
On the device, launch a terminal window to copy and paste the command to add execution permissions.

```
chmod +x start.sh
```

**Step 2: Run the start script**  
On the device, copy and paste the command to the terminal window and run the start script.

```
./start.sh
```

**Step 3: Return to this screen to view your device's messages**  
After running the start script, return to this screen to see the messages between your device and AWS IoT. The messages from your device appear in the following list.

Subscriptions	sdk/test/Python	Pause	Clear
sdk/test/Python	Waiting for messages		

Cancel Previous Continue

2. 在裝置的終端機或命令視窗中，於您輸入來自主控台中 Step 2 (步驟 2) 的命令之後，您應該會看到如下的輸出。此輸出來自程式傳送到 AWS IoT Core，然後從中收回的訊息。

```
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]"'
```

當範例程式正在執行時，也會出現測試訊息 Hello World!。測試訊息會顯示在終端機或裝置上的命令視窗中。

### Note

如需有關主題訂閱和發佈的詳細資訊，請參閱所選 SDK 的範例程式碼。

3. 您可以在此程序的主控台中重複來自 Step 2 (步驟 2) 的命令，以再次執行範例程式。
4. (選擇性) 如果您想要在主控台中查看來自 IoT 用戶端的訊息，請在 [AWS IoT 主控台](#) 的 [\[測試\] 頁面上開啟 MQTT 測試用戶端](#)。AWS IoT 如果您選擇 Python SDK，則在 MQTT test client (MQTT 測試用戶端) 的 Topic filter (主題篩選條件) 中，輸入主題，例如 **sdk/test/python** 以訂閱裝置上的訊息。主題篩選條件區分大小寫，且取決於 Step 1 (步驟 1) 中所選 SDK 的程式設計語言。如需有關主題訂閱和發佈的詳細資訊，請參閱所選 SDK 的程式碼範例。
5. 訂閱測試主題後，請在您的裝置上執行 `./start.sh`。如需詳細資訊，請參閱 [the section called “使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT”](#)。

執行 `./start.sh` 之後，訊息會顯示在 MQTT 用戶端中，內容與下列相似：

```
{
  "message": "Hello World!" [1]
}
```

每次收到新的 Hello World! 訊息，以 [] 括住的 sequence 號碼就會增加 1，並在您結束程式時停止。

6. 若要完成教學課程並查看摘要，請在 AWS IoT 主控台中選擇 [\[繼續\]](#)。



AWS IoT > Connect > Connect one device

Step 1  
Prepare your device

Step 2  
Register and secure your device

Step 3  
Choose platform and SDK

Step 4  
Download connection kit


Step 5  
**Run connection kit**


## Run connection kit Info

### How to display messages from your device

**Step 1: Add execution permissions**  
On the device, launch a terminal window to copy and paste the command to add execution permissions.


```
chmod +x start.sh
```

 Copy



**Step 2: Run the start script**  
On the device, copy and paste the command to the terminal window and run the start script.

```
./start.sh
```

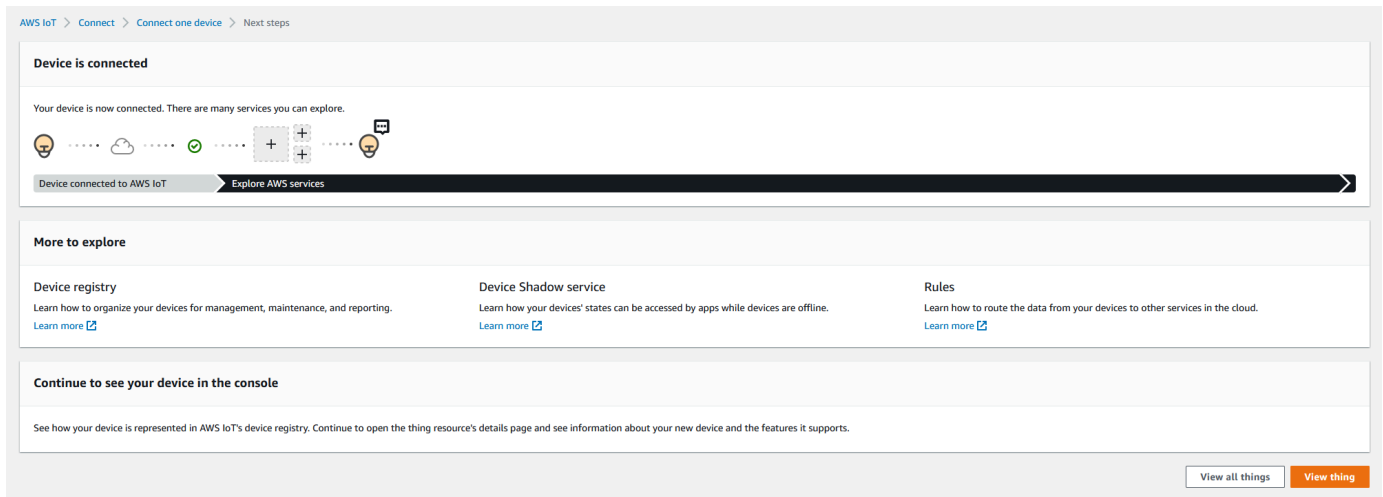
 Copy

**Step 3: Return to this screen to view your device's messages**  
After running the start script, return to this screen to see the messages between your device and AWS IoT. The messages from your device appear in the following list.

Subscriptions	sdk/test/Python	Resume	Clear
sdk/test/Python	<p>▼ sdk/test/Python <span style="float: right;">September 14, 2022, 10:47:44 (UTC-0700)</span></p> <p>"Hello World! [3]"</p>		
	<p>▼ sdk/test/Python <span style="float: right;">September 14, 2022, 10:47:43 (UTC-0700)</span></p> <p>"Hello World! [2]"</p>		
	<p>▼ sdk/test/Python <span style="float: right;">September 14, 2022, 10:47:42 (UTC-0700)</span></p> <p>"Hello World! [1]"</p>		

Cancel Previous Continue

7. 現在將顯示快 AWS IoT 速連接教程的摘要。



## 步驟 5. 深入探索

完成快速入門後，您可以 AWS IoT 進一步探索以下一些想法。

- [在 MQTT 測試用戶端中檢視 MQTT 訊息](#)

從 [AWS IoT 主控台中](#)，您可以在 AWS IoT 主控台的 Test (測試) 頁面上開啟 [MQTT 用戶端](#)。在 MQTT test client (MQTT 測試用戶端) 中，訂閱 #，然後在裝置上執行程式 `./start.sh`，如前一個步驟所述。如需詳細資訊，請參閱 [the section called “使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT”](#)。

- 在您的裝置上，透過 [Device Advisor](#) 執行測試

使用「裝置建議程式」來測試您的裝置是否可以安全可靠地連線到、以及與之互動 AWS IoT。

- [the section called “試用 AWS IoT Core 互動式教學”](#)

若要啟動互動式教學課程，請從 AWS IoT 主控台的 [學習] 頁面的 [查看 AWS IoT 運作方式] 動態磚中，選擇 [開始教學課程]。

- [準備好探索其他教學課程](#)

這個快速入門只會提供一個範例 AWS IoT。如果您想 AWS IoT 進一步探索並了解使其成為功能強大的 IoT 解決方案平台的功能，請開始準備您的開發平台 [在實作教學中探索 AWS IoT Core 服務](#)。

## 測試與裝置資料端點的連線

本主題會描述如何測試裝置與您帳戶的裝置資料端點連線，也就是您的 IoT 裝置用於連接到 AWS IoT 的端點。

在要測試的裝置上執行這些程序，或使用連結到要測試裝置的 SSH 終端階段作業。

若要測試裝置與您裝置資料端點的連線

- [尋找您的裝置資料端點](#)
- [快速測試連線](#)
- [取得應用程式，以測試與裝置資料端點和通訊埠的連線](#)
- [若要測試與裝置資料端點和通訊埠的連線](#)

## 尋找您的裝置資料端點

若要尋找您的裝置資料端點

1. 在[AWS IoT 主控台](#)的導覽窗格底部，選擇 Settings (設定)。
2. 在 Settings (設定) 頁面的 Device data endpoint (裝置資料端點) 容器中，找到 Endpoint (端點) 的值並進行複製。您的端點值是唯一的 AWS 帳戶，類似於以下範例：`a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`。
3. 儲存您的裝置資料端點，以便在下列程序中使用。

## 快速測試連線

此程序會測試與裝置資料端點的一般連線，但不測試您的裝置會使用的特定通訊埠。此測試會使用一般程式，通常足以知道您的裝置是否可以連線到 AWS IoT。

如果要測試與裝置將使用的特定通訊埠連線，請跳過此程序並繼續 [取得應用程式，以測試與裝置資料端點和通訊埠的連線](#)。

若要快速測試裝置資料端點

1. 在裝置上的終端或命令列視窗中，將範例裝置資料端點 (`a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com`) 替換為帳戶的裝置資料端點，然後輸入此命令。

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 顯示類似於以下內容的輸出，則其已成功連結到您的裝置資料端點。雖然它沒有 AWS IoT 直接與通信，但它確實找到了服務器，並且可能可以 AWS IoT 通過此端點獲得。

```
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
```

如果您對此結果滿意，可以在此處停止測試。

如果要測試與 AWS IoT 所使用之特定通訊埠的連線，請繼續進行 [取得應用程式，以測試與裝置資料端點和通訊埠的連線](#)。

3. 如果 ping 沒有傳回成功的輸出，請檢查終端值，以確保您具有正確的端點，並檢查裝置與網際網路的連線。

## 取得應用程式，以測試與裝置資料端點和通訊埠的連線

更徹底的連線測試可以通過使用 nmap 來執行。此程序會進行測試，以檢查 nmap 是否已安裝在您的裝置上。

### 若要檢查裝置上的 nmap

1. 在要測試的裝置上的終端或命令行視窗中，輸入此命令列，以查看 nmap 是否已安裝。

```
nmap --version
```

2. 如果您看到類似下列的輸出，代表 nmap 已安裝，且您可以繼續 [the section called “若要測試與裝置資料端點和通訊埠的連線”](#)。

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

3. 如果您沒有看到類似上述步驟中顯示的回覆，您必須在裝置上安裝 nmap。選擇適用於裝置作業系統的程序。

## Linux

此程序需要您在電腦上安裝軟體的權限。

若要在 Linux 電腦上安裝 nmap

1. 在裝置上的終端或命令列視窗中，輸入與其執行的 Linux 版本相對應的命令。
  - a. Debian 或 Ubuntu :

```
sudo apt install nmap
```

- b. CentOS 或 RHEL :

```
sudo yum install nmap
```

2. 使用以下命令測試安裝：

```
nmap --version
```

3. 如果您看到類似下列的輸出，代表已安裝 nmap，且您可以繼續 [the section called “若要測試與裝置資料端點和通訊埠的連線”](#)。

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

```
Available nsock engines: epoll poll select
```

## macOS

此程序需要您在電腦上安裝軟體的權限。

在 macOS 電腦上安裝 nmap

1. 在瀏覽器中，開啟 <https://nmap.org/download#macosx>，並下載最新穩定的安裝程式。

出現提示時，選取開啟方式 DiskImageInstaller。

2. 在安裝視窗中，將套件移動到 Applications (應用程式) 資料夾。
3. 在 Finder 中尋找 nmap-xxxx-mpkg 套件在 Applications (應用程式) 資料夾中的位置。在套件上 Ctrl-click，並選取 Open (開啟) 以開啟套件。
4. 檢閱安全對話方塊。如果您已準備好安裝 nmap，請選擇 Open (開啟) 以安裝 nmap。
5. 在 Terminal 中，使用此命令測試安裝。

```
nmap --version
```

6. 如果您看到類似下列的輸出，代表已安裝 nmap，且您可以繼續 [the section called “若要測試與裝置資料端點和通訊埠的連線”](#)。

```
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-libpcrc-7.6 nmap-libpcap-1.9.1 nmap-libdnet-1.12 ipv6 Compiled without:
Available nsock engines: kqueue poll select
```

## Windows

此程序需要您在電腦上安裝軟體的權限。

若要在 Windows 電腦上安裝 nmap

1. 在瀏覽器中，開啟 <https://nmap.org/download#windows> 並下載安裝程式的最新穩定版本。

出現提示時，選擇 Save file (儲存檔案)。下載檔案後，從下載文件夾中將其打開。

2. 安裝檔案下載完成後，打開下載的 nmap-xxxx-setup.exe 以安裝應用程式。

3. 程式安裝時，接受預設設定。

您不需要 Npcap 應用程式即可進行此測試。如果您不想安裝，可以取消選擇此選項。

4. 在 Command 中，使用此命令測試安裝。

```
nmap --version
```

5. 如果您看到類似下列的輸出，代表已安裝 nmap，且您可以繼續 [the section called “若要測試與裝置資料端點和通訊埠的連線”](#)。

```
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-libpcrc-7.6 Npcap-1.50 nmap-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: iocp poll select
```

## 若要測試與裝置資料端點和通訊埠的連線

### 若要測試裝置資料端點和通訊埠

1. 在裝置上的終端或命令列視窗中，將範例裝置資料端點 (*a3qEXAMPLEsffp-ats.iot.eu-west-1.amazonaws.com*) 替換為帳戶的裝置資料端點，然後輸入此命令。

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

2. 如果 nmap 會顯示類似下列內容的輸出，nmap 能夠在選定的通訊埠成功連結到您的裝置資料端點。

```
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
```

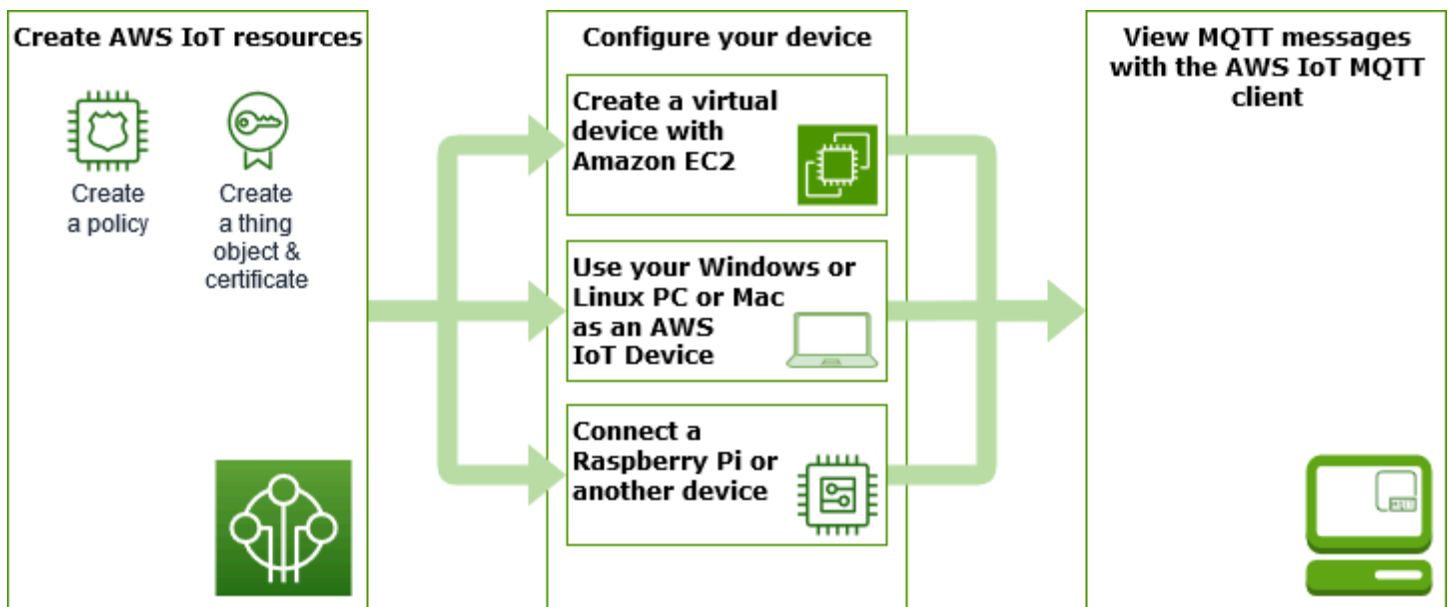
3. 如果 nmap 沒有傳回成功的輸出，請檢查端點值，以確保您的端點正確，並檢查裝置與網際網路的連線。

您可以測試裝置資料端點上的其他通訊埠，例如 443 通訊埠 (主要的 HTTPS 通訊埠)，只要將步驟 1 (8443) 中使用的通訊埠替換為您要測試的通訊埠即可。

## 在實作教學中探索 AWS IoT Core 服務

在本教程中，您將安裝軟件並創建連接設備所需的 AWS IoT 資源，以 AWS IoT Core 便它可以被用來發送和接收 MQTT 消息。AWS IoT Core 您會在主控台的 MQTT 用戶端中看到訊息。AWS IoT

您可以預期在本教學課程上花費 20-30 分鐘。當您使用 IoT 裝置或 Raspberry Pi 時，例如，如果您需要安裝作業系統並設定裝置，本教學課程可能需要更長的時間。



本教學課程最適合想要開始使用的開發人員，以 AWS IoT Core 便他們可以繼續探索更進階的功能，例如 [規則引擎](#) 和 [陰影](#)。本自學課程會比 [快速入門教學課程](#) 更詳細地說明這些步驟，讓您能夠繼續學習它與其他 AWS 服務的 [互動](#) 方式。AWS IoT Core 如果您只是尋找快速的 Hello World 體驗，請嘗試 [嘗試 AWS IoT 快速連接](#)。

設定 AWS 帳戶 和 AWS IoT 主機後，您將按照以下步驟瞭解如何連接裝置並讓裝置傳送訊息至 AWS IoT Core。



## 後續步驟

- [選擇最適合您的裝置選項](#)
- 如果您不打算使用 Amazon EC2 建立虛擬裝置，請 [the section called “建立 AWS IoT 資源”](#)。
- [the section called “設定您的裝置”](#)
- [the section called “使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT”](#)

如需有關的詳細資訊 AWS IoT Core，請參閱[什麼是 AWS IoT Core](#)？

## 哪種裝置選項最適合您？

如果您不確定要挑選哪個選項，請使用下列各選項的優缺點清單，協助您決定哪個選項最適合您。

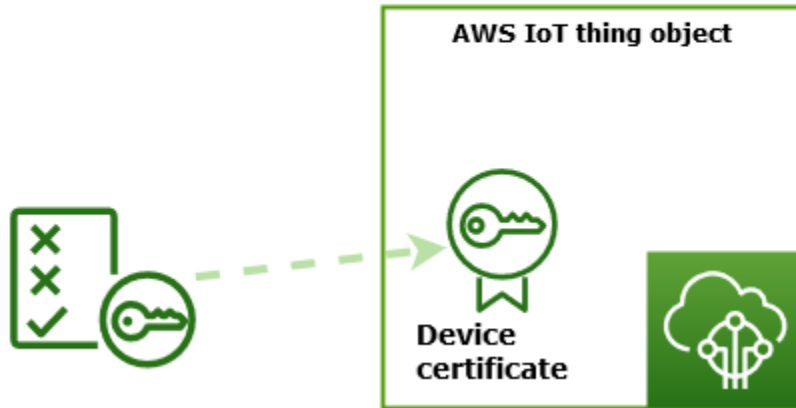
選項	這可能是一個好選項，如果：	這可能不是一個好選項，如果：
<a href="#">the section called “使用 Amazon EC2 建立虛擬裝置”</a>	<ul style="list-style-type: none"> <li>• 您沒有自己的裝置進行測試。</li> <li>• 您不想要在自己的系統上安裝任何軟體。</li> <li>• 您想要在 Linux 作業系統上進行測試。</li> </ul>	<ul style="list-style-type: none"> <li>• 您不習慣使用命令列命令。</li> <li>• 您不想要產生任何額外的 AWS 費用。</li> <li>• 您不想要在 Linux 作業系統上進行測試。</li> </ul>
<a href="#">the section called “使用您的視窗或 Linux 個人電腦或 Mac 作為 AWS IoT 裝置”</a>	<ul style="list-style-type: none"> <li>• 您不想要產生任何額外的 AWS 費用。</li> <li>• 您不想要設定任何額外裝置。</li> </ul>	<ul style="list-style-type: none"> <li>• 您不想要在個人電腦上安裝任何軟體。</li> <li>• 您想要一個更具代表性的測試平台。</li> </ul>
<a href="#">the section called “連接 Raspberry Pi 或其他裝置”</a>	<ul style="list-style-type: none"> <li>• 您想 AWS IoT 使用實際設備進行測試。</li> <li>• 您已有測試用的裝置。</li> <li>• 您具備將硬體整合至系統的經驗。</li> </ul>	<ul style="list-style-type: none"> <li>• 您不想要只為了試用而購買或設定裝置。</li> <li>• 您現在想要盡可能簡單地測試 AWS IoT。</li> </ul>

## 建立 AWS IoT 資源

在本教程中，您將創建設備連接 AWS IoT Core 和交換消息所需的 AWS IoT 資源。

### Create an AWS IoT Core policy

### Create a thing and its certificate



1. 建立 AWS IoT 政策文件，以授權您的裝置與 AWS IoT 服務互動。
2. 在中建立物物件 AWS IoT 及其 X.509 裝置憑證，然後附加原則文件。物件物件是您裝置在 AWS IoT 登錄中的虛擬表示。憑證會對您的裝置進行驗證 AWS IoT Core，而政策文件則授權您的裝置與之互動。AWS IoT

#### Note

若您打算 [the section called “使用 Amazon EC2 建立虛擬裝置”](#)，則可跳過本頁，並繼續 [the section called “設定您的裝置”](#)。建立您的虛擬物件時，您將建立這些資源。

本教學課程使用 AWS IoT 主控台來建立 AWS IoT 資源。如果您的裝置支援網頁瀏覽器，在裝置的網頁瀏覽器上執行此程序可能會比較容易，因為您可以直接將憑證檔案下載到您的裝置。如果您在其他電腦上執行此程序，則必須先將憑證檔案複製到您的裝置，然後範例應用程式才能使用這些檔案。

## 建立 AWS IoT 策略

裝置會使用 X.509 憑證來進行驗證。AWS IoT Core憑證已附加 AWS IoT 原則。這些政策決定了裝置可執行哪些 AWS IoT 操作 (例如訂購或發佈至 MQTT 主題)。您的裝置會在連線並傳送訊息至時顯示憑證 AWS IoT Core。

遵循這些步驟，以建立一個政策，讓您的裝置執行 AWS IoT 執行範例程式所需的作業。您必須先建立 AWS IoT 政策，才能將其附加到之後會建立的裝置憑證中。

## 若要建立 AWS IoT 策略

1. 在 [AWS IoT 主控台](#) 的左側選單中，選擇安全，然後選擇政策。
2. 請在 You don't have a policy yet (尚無任何政策) 頁面上，選擇 Create a policy (建立政策)。

如果您的帳戶已有現有政策，請選擇建立政策。

3. 在 Create policy (建立政策) 頁面上：

1. 在 Policy properties (政策內容) 區段的 Policy name (政策名稱) 欄位中，輸入政策名稱 (例如 **My\_Iot\_Policy**)。請勿在政策名稱中使用個人識別資訊。
2. 在 Policy document (政策文件) 區段中，建立可授予或拒絕授予 AWS IoT Core 作業資源存取權的政策陳述式。若要建立政策陳述式，授予所有用戶端執行 **iot:Connect**，請執行這些步驟：
  - 在 Policy effect (政策效果) 欄位中，選擇允許。這允許已將此政策附加至其憑證的所有用戶端執行 Policy action (政策動作) 欄位中列出的動作。
  - 在 Policy action (政策動作) 欄位中，選擇政策操作 (例如 **iot:Connect**)。政策動作是您的裝置從裝置 SDK 執行範例程式時，需要許可才能執行的動作。
  - 在 Policy resource (政策資源) 欄位中，輸入資源 Amazon Resource Name (ARN) 或 \*。一個 \* 以選擇任何用戶端 (裝置)。

若要建立 **iot:Receive**、**iot:Publish** 以及 **iot:Subscribe** 的政策陳述式，請選擇 Add new statement (新增陳述式) 並重複這些步驟。

<u>Policy effect</u>	<u>Policy action</u>	<u>Policy resource</u>	
Allow ▼	iot:Connect ▼	*	Remove
Allow ▼	iot:Receive ▼	*	Remove
Allow ▼	iot:Publish ▼	*	Remove
Allow ▼	iot:Subscribe ▼	*	Remove

### Note

在這個快速入門中，萬用字元 (\*) 字元是為了簡單起見而使用。如需更高的安全性，您應該指定用戶端 ARN，而非將萬用字元指定為資源，以限制哪些用戶端 (裝置) 可以連

接和發佈訊息。用戶端 ARN 遵循以下格式：`arn:aws:iot:your-region:your-aws-account:client/my-client-id`。

不過，您必須先建立資源 (例如用戶端裝置，或物件影子)，然後才能將其 ARN 指配給政策。如需詳細資訊，請參閱 [AWS IoT Core 動作資源](#)。

4. 政策相關資訊輸入完畢後，請選擇 Create (建立)。

如需詳細資訊，請參閱 [如何與 IAM AWS IoT 搭配使用](#)。

## 建立物件

連線到的裝置 AWS IoT Core 會由 AWS IoT 登錄中的物件物件表示。「物件」代表特定的裝置或邏輯實體。它可以是實體裝置或感應器 (例如燈泡或牆上的燈光開關)。它也可以是一個邏輯實體，例如應用程序的實體或實體實體，這些實體不連接 AWS IoT，但與其他設備有關 (例如，具有引擎傳感器或控制面板的汽車) 相關。

若要在 AWS IoT 主控台中建立物件

1. 在 [AWS IoT 主控台](#) 的左側選單中，選擇所有裝置，然後選擇物件。
2. 在 Things (物件) 頁面上，選擇 Create things (建立物件)。
3. 在 Create things (建立物件) 頁面上，選擇 Create a single thing (建立單一物件)，然後選擇 Next (下一步)。
4. 在 Specify thing properties (指定物件屬性) 頁面上，針對 Thing name (物件名稱)，輸入您的物件名稱，例如 **MyIotThing**。

請小心選擇物件名稱，因為您之後就無法變更物件名稱。

要變更物件的名稱，您必須建立一個新的物件並為它命名，然後刪除舊的物件。

### Note

請勿在物件名稱中使用個人識別資訊。物件名稱可以出現在未加密的通訊和報告中。

5. 將此頁面上的其餘欄位保持空白。選擇下一步。
6. 在 Configure device certificate - optional (設定裝置憑證 - 選用) 頁面上，選擇 Auto-generate a new certificate (recommended) (自動產生新憑證 (建議動作))。選擇下一步。
7. 在 Attach policies to certificate - optional (將策略連接到憑證 - 選用) 頁面上，選取您在上節所建立的政策。在該節中，政策已命名為 **My\_Iot\_Policy**。選擇 Create thing (建立物件)。

## 8. 在 Download certificates and keys (下載憑證和金鑰) 頁面上：

1. 下載每個憑證和金鑰檔案，並儲存它們以供稍後使用。您必須在裝置上安裝這些檔案。

當您儲存憑證檔案時，請給與它們下表中的名稱。這些是稍後範例中使用的檔案名稱。

### 憑證檔案名稱

檔案	檔案路徑
私有金鑰	private.pem.key
公有金鑰	(未在這些範例中使用)
裝置憑證	device.pem.crt
根 CA 憑證	Amazon-root-CA-1.pem

2. 若要下載這些檔案的根 CA 檔案，請選擇根 CA 憑證檔案的 Download (下載) 連結，此憑證檔案對應於資料端點類型以及您正在使用的密碼套件。在本教學課程中，選擇 RSA 2048 bit key: Amazon Root CA 1 (RSA 2048 位元金鑰：Amazon 根 CA 1) 右側的 Download (下載)，然後下載 RSA 2048 bit key: Amazon Root CA 1 (RSA 2048 位元金鑰：Amazon 根 CA 1) 憑證檔案。

#### Important

您必須先儲存憑證檔案，然後才能離開此頁面。在主控台中離開此頁面之後，您再也不能存取憑證檔案。

如果忘了下載在此步驟中所建立的憑證檔案，您必須結束此主控台畫面、移至主控台內的物件清單、刪除您建立的物件，然後從頭重新啟動此程序。

3. 選擇 Done (完成)。

完成此程序之後，您應該會在物件清單中看到新的物件。

## 設定您的裝置

本節描述如何設定您的裝置以連接至 AWS IoT Core。如果您想要開始使用，AWS IoT Core 但還沒有裝置，可以使用 Amazon EC2 建立虛擬裝置，也可以使用 Windows PC 或 Mac 做為 IoT 裝置。

選擇最佳設備選項供您嘗試 AWS IoT Core。當然，您可以嘗試它們全部，但一次只能嘗試一個。如果您不確定哪個裝置選項最適合您，請閱讀如何選擇[哪個是最好的裝置選項](#)，然後返回此頁面。

## 裝置選項

- [使用 Amazon EC2 建立虛擬裝置](#)
- [使用您的視窗或 Linux 個人電腦或 Mac 作為 AWS IoT 裝置](#)
- [連接 Raspberry Pi 或其他裝置](#)

## 使用 Amazon EC2 建立虛擬裝置

在本教學課程中，您將建立 Amazon EC2 執行個體，充當雲端中的虛擬裝置。

若要完成此自學課程，您需要 AWS 帳戶。若您沒有帳戶，請完成 [設定您的 AWS 帳戶](#) 所述的步驟，然後再繼續。

在本教學課程中，您將會執行下列動作：

- [設定 Amazon EC2 執行個體](#)
- [安裝 Git、Node.js，並設定 AWS CLI](#)
- [為您的虛擬裝置建立 AWS IoT 資源](#)
- [安裝下列項目的 AWS IoT 裝置 SDK JavaScript](#)
- [執行範例應用程式](#)
- [在 AWS IoT 主控台中檢視來自範例應用程式的訊息](#)

## 設定 Amazon EC2 執行個體


下列步驟顯示如何建立 Amazon EC2 執行個體，其將充當您的虛擬裝置取代實體裝置。

如果您是第一次建立 Amazon EC2 執行個體，[開始使用 Amazon EC2Linux 執行個體](#) 中的相關指示可能會更有幫助。

## 啟動執行個體

1. 在 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 從左側的主控台功能表展開 Instances (執行個體) 區段，然後選擇 Instances (執行個體)。在 Instances (執行個體) 儀表板中，選擇右側的 Launch instances (啟動執行個體) 以顯示基本組態清單。
3. 在 Name and tags (名稱和標籤) 區段中，輸入執行個體的名稱，並選擇性地新增標籤。

4. 在 Application in OS Images (Amazon Machine Image) (應用程式和作業系統映像) 區段中選擇執行個體的 AMI 範本，例如 Amazon Linux 2 AMI (HVM)。請注意，此 AMI 會標示為 "Free tier eligible." (符合免費方案。)
5. 在 Instance Type (執行個體類型) 頁面中，您可以選取您執行個體的硬體組態。選取 t2.micro 類型，其預設為選取。請注意，此執行個體類型符合免費方案資格。
6. 在 Key pair (login) (金鑰對 (登入)) 區段中，從下拉式清單中選擇金鑰對名稱，或選擇 Create a new key pair (建立新金鑰對) 以建立新的金鑰對。建立新金鑰對時，請務必下載私密金鑰檔案並將其儲存在安全的地方，因為這是您下載並儲存的唯一機會。每次您連線至執行個體來啟動執行個體與對應的私有金鑰時，都需要提供您的金鑰對名稱。

 Warning

不要選取 Proceed without a key pair (不使用金鑰對而繼續) 選項。如果不使用金鑰對而啟動執行個體，就無法與它連線。

7. 在 Network settings (網路設定) 區段和 Configure storage (設定儲存區) 區段中，您可以保留預設設定。當您就緒後，選擇 Launch Instances (啟動執行個體)。
8. 會有確認頁面讓您知道您的執行個體正在啟動。選擇 View Instances (檢視執行個體) 關閉確認頁面並返回主控台。
9. 您可以在 Instances (執行個體) 畫面中檢視啟動狀態。啟動執行個體無須費時。當您啟動執行個體時，其初始狀態是 pending。在執行個體啟動後，其狀態會變更為 running，並得到公有的 DNS 名稱。(如果隱藏 Public DNS (IPv4) (公有 DNS (IPv4)) 欄，請選擇頁面右上角的 Show/Hide Columns (顯示/隱藏欄) (齒輪狀圖示)，然後選取 Public DNS (IPv4) (公有 DNS (IPv4))。)
10. 執行個體可能需要幾分鐘的時間準備就緒讓您連線。確認您的執行個體是否已通過狀態檢查，您可以在 Status Checks (狀態檢查) 欄檢視此資訊。

在新的執行個體通過其狀態檢查之後，請繼續下一個程序並連接至該執行個體。

## 連結到您的執行個體

您可以從 Amazon EC2 主控台選取執行個體，並選擇使用 Amazon EC2 Instance Connect 來進行連線，以使用瀏覽器型用戶端連接至執行個體。Instance Connect 會處理許可並提供成功的連線。

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在左側選單中，選擇 Instances (執行個體)。
3. 選取執行個體，然後選取 Connect (連線)。

#### 4. 選擇 Amazon EC2 Instance Connect , Connect (連線)。

您現在應該具有已登入新 Amazon EC2 執行個體的 Amazon EC2 Instance Connect 視窗。

安裝 Git、Node.js , 並設定 AWS CLI

在本節中 , 您會在 Linux 執行個體上安裝 Git 和 Node.js。

##### 安裝 Git

1. 在您的 Amazon EC2 Instance Connect 視窗中 , 使用下列命令更新您的執行個體。

```
sudo yum update -y
```

2. 在您的 Amazon EC2 Instance Connect 視窗中 , 使用下列命令安裝 Git。

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

3. 若要檢查 Git 是否已安裝 , 以及 Git 的目前版本 , 請執行下列命令 :

```
git --version
```

##### 安裝 Node.js

1. 在您的 Amazon EC2 Instance Connect 視窗中 , 使用下列命令安裝 Node Version Manager (NVM)。

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

nvm 可以安裝多種 Node.js 版本且允許您在各版本間進行切換 , 因此我們會使用 nvm 安裝 Node.js。

2. 在您的 Amazon EC2 Instance Connect 視窗中 , 使用此命令啟用 nvm。

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

3. 在您的 Amazon EC2 Instance Connect 視窗中 , 使用 nvm 來透過使用此命令安裝最新版本的 Node.js。



```
nvm install 16
```

**Note**

這會安裝 Node.js 的最新 LTS 版本。

安裝 Node.js 時，系統會一併安裝節點套件管理工具 (npm)，您可以視需要安裝額外的模組。

4. 在您的 Amazon EC2 Instance Connect 視窗中，使用此命令來測試是否已安裝並正確地執行 Node.js。

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

本教學課程需要 Node v10.0 或更新版本。如需詳細資訊，請參閱 [Tutorial: Setting Up Node.js on an Amazon EC2 Instance](#) (教學課程：在 Amazon EC2 執行個體上設定 Node.js)。

## 若要設定 AWS CLI

您的 Amazon EC2 執行個體已預先載入 AWS CLI。但是，您必須完成您的 AWS CLI 個人資料。如需如何設定 CLI 的詳細資訊，請參閱 [設定 AWS CLI](#)。

1. 下列範例顯示範本值。將它們取代為您自己的值。您可以在 [AWS 主控台中於您在 My Security Credentials](#) (我的安全憑證) 下的帳戶資訊中找到這些值。

在您的 Amazon EC2 Instance Connect 視窗中，輸入此命令：

```
aws configure
```

然後在顯示的提示中輸入來自您帳戶的值。

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

2. 您可以使用以下命令測試您的 AWS CLI 配置：

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

如果您 AWS CLI 的設定正確，該命令應該從您的 AWS 帳戶。

## 為您的虛擬裝置建立 AWS IoT 資源

本節說明如何使用 AWS CLI 直接在虛擬裝置上建立物件及其憑證檔案。這是直接在裝置上完成，以避免從另一台電腦將它們複製到裝置時可能造成的潛在複雜性。在本節中，您將為虛擬裝置建立下列資源：

- 代表虛擬裝置的物件 AWS IoT。
- 要驗證虛擬裝置的憑證。
- 政策文件，用來授權虛擬裝置連接至 AWS IoT，以及用來發佈、接收和訂閱訊息。

## 若要在 Linux 執行 AWS IoT 個體中建立物件物件

連線到的裝置 AWS IoT 會由 AWS IoT 登錄中的物件物件表示。「物件」代表特定的裝置或邏輯實體。在此案例中，您的「物件」將代表您的虛擬裝置，即 Amazon EC2 執行個體。

1. 在您的 Amazon EC2 Instance Connect 視窗中，執行下列命令來建立您的物件。

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

2. JSON 回應應該如下所示：

```
{
  "thingArn": "arn:aws:iot:your-region:your-aws-account:thing/MyIotThing",
  "thingName": "MyIotThing",
  "thingId": "6cf922a8-d8ea-4136-f3401EXAMPLE"
}
```

## 在 Linux 執行個體中建立並附加 AWS IoT 金鑰和憑證

[create-keys-and-certificate](#) 命令會建立由 Amazon 根憑證授權機構簽署的用戶端憑證。此憑證用來驗證虛擬裝置的身分。

1. 在您的 Amazon EC2 Instance Connect 視窗中，建立目錄以存放您的憑證和金鑰檔案。

```
mkdir ~/certs
```

2. 在您的 Amazon EC2 Instance Connect 視窗中，使用此命令下載 Amazon 憑證授權機構 (CA) 憑證的副本。

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

3. 在您的 Amazon EC2 Instance Connect 視窗中，執行下列命令來建立您的私有金鑰、公有金鑰，以及 X.509 憑證檔案。此指令也會使用註冊和啟動憑證 AWS IoT。

```
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"
```

回傳的結果如下所示。儲存 `certificateArn`，以便您可以在後續命令中使用它。您需要它，將您的憑證連接到您的物件，並在稍後的步驟中將政策連接到憑證。

```
{
  "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
  "certificateId":
  "9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
  "certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMaKGA1UEBhMC
VVMx CzAJBgNVBAGEXAMPLEAwDgYDVQQHEwdTZWF0dGxLMQ8wDQYDVQQKEwZBbWF6
b24xFDASBgNVBAwTC0lBTSEXAMPLE2xLMRIwEAYDVQQDEwLUZXN0Q2lsYWxhZAd
BgkqhkiG9w0BCQEWEG5vb25lQGFTYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMaKGA1UEBhMCEXAMPLEJBgNVBAGTAldBMRAwDgYD
VQQHEwdTZWF0dGxLMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC0lBTSBDb25z
b2xLMRIwEAYDVQQDEwLUZXN0Q2lsYWxhZAdBgkqhkiG9w0BCQEXAMPLE25lQGFT
YXpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZEXAMPLEELG5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcvaEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEEyExzyLwaxLAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJIILJ00zbhNYS5f6GuoEDEXAMPLEBHjJnyp3780D8uTs7fLvJx79LjSTB
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
```

```

-----END CERTIFICATE-----\n",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC
KEY-----\nMIIBIjANBgkqhkEXAMPLEQEFAA0CAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h
\nMMEXAMPLEUuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/
gHr99VEEXAMPLE5VF13\n59VK7cEXAMPLE67GK+y+jikqX0gHh/xJTtwo
+sGpWEXAMPLEDz18x0d2ka4tCzuWEXAMPLEEahJbYkCPUBSU8opVkr7qkEXAMPLE1DR6sx2HocLi00Ltu6Fkw91swQW
\nGB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEVp9mQ0UXP6pLfgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEEcw+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 視窗中，將您的物件連接到您剛建立的憑證，方法是使用下列命令，以及來自前一個命令之回應中的 *certificateArn*。

```

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

```

如果成功，此命令不會顯示任何輸出。

## 建立並連接政策

1. 在您的 Amazon EC2 Instance Connect 視窗中，複製此策略文件並將其貼到名為 `~/policy.json` 的檔案來建立政策檔案。

如果您沒有最愛的 Linux 編輯器，則可以使用此命令開啟 nano。

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

將 `policy.json` 的政策文件貼入其中。輸入 `ctrl-x` 以結束 nano 編輯器並儲存檔案。

`policy.json` 的政策文件內容

```

{
  "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 視窗中，使用下列命令建立您的政策。

```

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

```

輸出：

```

{
  "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 視窗中，使用下列命令將政策連接至虛擬裝置的憑證。

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

如果成功，此命令不會顯示任何輸出。

## 安裝下列項目的 AWS IoT 裝置 SDK JavaScript

在本節中，您將安裝 AWS IoT Device SDK JavaScript，其中包含應用程式可用來與之通訊的程式碼以 AWS IoT 及範例程式。如需詳細資訊，請參閱 [JavaScript GitHub 儲存庫的 AWS IoT 裝置 SDK](#)。

若要 JavaScript 在您的 Linux 執行個體上安裝的裝 AWS IoT 置 SDK

1. 在 Amazon EC2 執行個體 Connect 視窗中，使用此命令將 JavaScript 儲存庫的 AWS IoT 裝置 SDK 複製到主目錄的目錄中。aws-iot-device-sdk-js-v2

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

2. 導覽至您在上述步驟中建立的 aws-iot-device-sdk-js-v2 目錄。

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

3. 使用 npm 來安裝 SDK。

```
npm install
```

## 執行範例應用程式

以下各節命令假設您的金鑰和憑證檔案存放在虛擬裝置上，如下表所示。

## 憑證檔案名稱

檔案	檔案路徑
私有金鑰	~/certs/private.pem.key
裝置憑證	~/certs/device.pem.crt
根 CA 憑證	~/certs/Amazon-root-CA-1.pem

在本節中，您將安裝並運行在 AWS IoT 設備 SDK 的 `aws-iot-device-sdk-js-v2/samples/node` 目錄中找到的 `pub-sub.js` 示例應用程序 JavaScript。此應用程式顯示裝置 (您的 Amazon EC2 執行個體) 如何使用 MQTT 程式庫來發佈和訂閱 MQTT 訊息。`pub-sub.js` 範例應用程式會訂閱主題 `topic_1`、將 10 則訊息發佈至該主題，以及在從訊息代理程式接收訊息時顯示這些訊息。

## 安裝並執行範例應用程式

1. 在您的 Amazon EC2 Instance Connect 視窗中，導覽至 SDK 已建立的 `aws-iot-device-sdk-js-v2/samples/node/pub_sub` 目錄，並使用這些命令安裝範例應用程式。

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

2. 在您的 Amazon EC2 執行個體 Connect 視窗 *your-iot-endpoint* 中，使 AWS IoT 用此命令進入。

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

3. 在 Amazon EC2 執行個體 Connect 視窗中，*your-iot-endpoint* 按指示插入並執行此命令。

```
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
```

## 範例應用程式：

1. 連接到您 AWS IoT Core 的帳戶。
2. 訂閱訊息主題 `topic_1`，並顯示其收到關於該主題的訊息。
3. 將 10 則訊息發佈至主題 `topic_1`。

#### 4. 顯示類似下列內容的輸出：

```
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}
```

若在執行範例應用程式時發生問題，請檢閱 [the section called “疑難排解範例應用程式的問題”](#)。

您也可以將 `--verbosity debug` 參數新增至命令列，以便範例應用程式顯示有關其正在做什麼的詳細訊息。這些資訊可能會提供您更正問題所需的協助。

在 AWS IoT 主控台中檢視來自範例應用程式的訊息

您可以在範例應用程式的訊息透過訊息代理程式傳遞時看到這些訊息，方法為使用 AWS IoT 主控台內的 MQTT test client (MQTT 測試用戶端)。

檢視範例應用程式所發佈的 MQTT 訊息

1. 檢閱 [使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)。這有助於您了解如何使用 AWS IoT 主控台內的 MQTT 測試用戶端，在 MQTT 訊息通過訊息代理程式時，檢視訊息。
2. 在 AWS IoT 主控台中開啟 MQTT 測試用戶端。
3. 在Subscribe to a topic (訂閱主題)中，訂閱主題，topic\_1。



4. 在您的 Amazon EC2 Instance Connect 視窗中，再次執行範例應用程式，並在 AWS IoT 主控台的 MQTT test client (MQTT 測試用戶端) 中觀看訊息。

```
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
```

[如需 MQTT 以及如何 AWS IoT Core 支援通訊協定的詳細資訊，請參閱 MQTT。](#)

## 使用您的視窗或 Linux 個人電腦或 Mac 作為 AWS IoT 裝置

在本教學課程中，您將配置個人電腦以搭配使用 AWS IoT。這些指示支援 Windows 和 Linux PC 以及 Mac。若要完成此目的，您需要在電腦上安裝一些軟體。如果您不想要在電腦上安裝軟體，則可以嘗試[使用 Amazon EC2 建立虛擬裝置](#)，它會在虛擬機器上安裝所有軟體。

在本教學課程中，您將會執行下列動作：

- [設定您的個人電腦](#)
- [安裝 Git、Python 和裝 AWS IoT 置開發套件](#)
- [設定政策和執行範例應用程式](#)
- [在 AWS IoT 主控台中檢視來自範例應用程式的訊息](#)
- [在 Python 中執行共享訂閱範例](#)

### 設定您的個人電腦

若要完成本教學課程，您需要一部 Windows 或 Linux PC，或是一部可連接到網際網路的 Mac。

繼續下一個步驟之前，請確定您可以在電腦上開啟命令列視窗。在 Windows PC 上使用 cmd.exe。在 Linux PC 或 Mac 上，使用 Terminal。

### 安裝 Git、Python 和裝 AWS IoT 置開發套件

在本節中，您將在計算機上安裝 Python 和適用於 Python 的 AWS IoT 設備 SDK。

#### 安裝最新版本的 Git 和 Python

在您的電腦上下載並安裝 Git 和 Python

1. 查看您的電腦上是否已安裝 Git。在命令列中輸入此命令。

```
git --version
```

如果命令顯示 Git 版本，表示已安裝 Git，而且您可以繼續進行下一個步驟。

如果命令顯示錯誤，請開啟 <https://git-scm.com/download> 並為您的電腦安裝 Git。

2. 查看您是否已安裝 Python。在命令列中輸入此命令。

```
python -V
```

#### Note

如果此命令提供錯誤：Python was not found，原因可能是您的作業系統呼叫 Python v3.x 可執行檔作為 Python3。在此情況下，將 python 的所有執行個體取代為 python3，並繼續進行本教學課程的其餘部分。

如果命令顯示 Python 版本，則表示已安裝 Python。本教學課程需要 Python v3.7 或更新版本。

3. 如果已安裝 Python，則可以跳過本節的其餘步驟。如果未安裝，請繼續。
4. 開啟 <https://www.python.org/downloads/>，並下載適用於您電腦的安裝程式。
5. 如果下載未自動開始安裝，請執行下載的程式來安裝 Python。
6. 驗證 Python 的安裝。

```
python -V
```

確認命令顯示 Python 版本。如果未顯示 Python 版本，請嘗試再次下載並安裝 Python。

## 安裝適用於 Python 的 AWS IoT 裝置開發套件

在您的電腦上安裝適用於 Python 的裝 AWS IoT 置開發套件

1. 安裝適用於 Python 的裝 AWS IoT 置開發套件的第 2 版。

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

2. 將適用於 Python 儲存庫的 AWS IoT 裝置 SDK 複製到主目錄的 aws-iot-device-sdk-python-v2 目錄中。此程序會將所安裝檔案的目錄稱為 *home*。

`home` 目錄的實際位置取決於您的作業系統。

## Linux/macOS

在 macOS 和 Linux 中，`home` 目錄為 `~`。

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

## Windows

在 Windows 中，您可以在 cmd 視窗中執行此命令，來尋找 `home` 目錄路徑。

```
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 Windows PowerShell 而不是 cmd.exe，請使用以下命令。

```
echo $home
```

如需詳細資訊，請參閱[適用於 Python GitHub 存放庫的 AWS IoT 裝置 SDK](#)。

## 準備執行範例應用程式

### 準備您的系統以執行範例應用程式

- 建立 `certs` 目錄。將下列項目複製至 `certs` 目錄：私有金鑰、裝置憑證，以及您在 [the section called “建立 AWS IoT 資源”](#) 建立和註冊物件時所儲存的根憑證授權機構憑證檔案。目的地目錄中每個檔案的檔案名稱應與表格中的檔案名稱相符。

下節命令假設您的金鑰和憑證檔案儲存在您的裝置上，如下表所示。

## Linux/macOS

執行此命令來建立 `certs` 子目錄，您會在執行範例應用程式時使用這個子目錄。

```
mkdir ~/certs
```

在新的子目錄中，將檔案複製到下表所示的目的地檔案路徑。

### 憑證檔案名稱

檔案	檔案路徑
私有金鑰	~/certs/private.pem.key
裝置憑證	~/certs/device.pem.crt
根 CA 憑證	~/certs/Amazon-root-CA-1.pem

執行此命令來列出 `certs` 目錄中的檔案，並將其與表格中列出的檔案進行比較。

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

## Windows

執行此命令來建立 `certs` 子目錄，您會在執行範例應用程式時使用這個子目錄。

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

在新的子目錄中，將檔案複製到下表所示的目的地檔案路徑。

### 憑證檔案名稱

檔案	檔案路徑
私有金鑰	%USERPROFILE%\certs\private.pem.key

檔案	檔案路徑
裝置憑證	%USERPROFILE%\certs\device.pem.crt
根 CA 憑證	%USERPROFILE%\certs\Amazon-root-CA-1.pem

執行此命令來列出 certs 目錄中的檔案，並將其與表格中列出的檔案進行比較。

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

## 設定政策和執行範例應用程式

在本節中，您會設定政策，並執行 pubsub.py 範例指令碼，其位於適用於 Python 的 AWS IoT Device SDK 的 aws-iot-device-sdk-python-v2/samples 目錄中。此指令碼顯示您的裝置如何使用 MQTT 程式庫來發佈和訂閱 MQTT 訊息。

pubsub.py 範例應用程式會訂閱主題 test/topic、將 10 則訊息發佈至該主題，以及在從訊息代理程式接收訊息時顯示這些訊息。

若要執行 pubsub.py 範例指令碼，您需要以下資訊：

### 應用程式參數值

參數	可在哪裡找到值
<i>your-iot-endpoint</i>	<ol style="list-style-type: none"> <li>在 <a href="#">AWS IoT 主控台</a> 的左側選單中，選擇 Settings (設定)。</li> <li>在 Settings (設定) 頁面上，您的端點會顯示在 Device data endpoint (裝置資料端點) 區段中。</li> </ol>

該 *your-iot-endpoint* 值的格式為：*endpoint\_id*-ats.iot.*region*.amazonaws.com，例如 a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com。

執行指令碼之前，請確定您物件的政策許可範例指令碼連接、訂閱、發佈和接收。

## 尋找並檢閱物件資源的政策文件

1. 在 [AWS IoT 主控台](#) 的 Things (物件) 清單中，尋找代表您裝置的資源。
2. 選擇代表您裝置物件資源的 Name (名稱) 連結，以開啟 Thing details (物件詳細資訊) 頁面。
3. 在 Thing details (物件詳細資訊) 頁面中的 Certificates (憑證) 索引標籤上，選擇與物件資源相連的憑證。清單中應該只有一個憑證。如果有一個以上的憑證，請選擇其檔案已安裝在裝置上且將用來連接至 AWS IoT Core 的憑證。

於 Certificate (憑證) 詳細資訊頁面的 Policies (政策) 索引標籤中，選擇與憑證相連的政策。其中應只有一個政策。如果有多個政策，請針對每個政策重複下一個步驟，以確保至少有一個政策授予必要的存取權。

4. 在 Policy (政策) 概觀頁面中，找到 JSON 編輯器並選擇 Edit policy document (編輯政策文件)，視需要檢閱和編輯政策文件。
5. 政策 JSON 會顯示在下列範例中。在 "Resource" 元素中，*region:account* 用您 AWS 帳戶的 AWS 區域 和替換每個 Resource 值。

```
{
  "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 上執行範本指令碼

1. 在命令列視窗中，導覽至 SDK 使用這些命令所建立的 `~/aws-iot-device-sdk-python-v2/samples/node/pub_sub` 目錄。

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

2. 在命令行窗口中，*your-iot-endpoint* 按照指示替換並運行此命令。

```
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

在 Windows PC 上執行範例應用程式

1. 在命令列視窗中，導覽至 SDK 所建立的 `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` 目錄，然後使用這些命令來安裝範例應用程式。

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

2. 在命令行窗口中，*your-iot-endpoint* 按照指示替換並運行此命令。

```
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
```

**範例指令碼：**

1. 連線至您帳戶的。AWS IoT Core
2. 訂閱訊息主題 test/topic，並顯示其收到關於該主題的訊息。
3. 將 10 則訊息發佈至主題 test/topic。
4. 顯示類似下列內容的輸出：

```
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!
```

若在執行範例應用程式時發生問題，請檢閱 [the section called “疑難排解範例應用程式的問題”](#)。

您也可以將 `--verbosity Debug` 參數新增至命令列，以便範例應用程式顯示有關其正在做什麼的詳細訊息。該資訊可能會協助您更正問題。



## 在 AWS IoT 主控台中檢視來自範例應用程式的訊息

您可以在範例應用程式的訊息透過訊息代理程式傳遞時看到這些訊息，方法為使用 AWS IoT 主控台中央的 MQTT test client (MQTT 測試用戶端)。

### 檢視範例應用程式所發佈的 MQTT 訊息

1. 檢閱 [使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)。這有助於您了解如何使用 AWS IoT 主控台中央的 MQTT 測試用戶端，在 MQTT 訊息通過訊息代理程式時，檢視訊息。
2. 在 AWS IoT 主控台中開啟 MQTT 測試用戶端。
3. 在 Subscribe to a topic (訂閱主題) 中，訂閱主題 test/topic。
4. 在您的命令列視窗中，再次執行範例應用程式，並在 AWS IoT 主控台的 MQTT client (MQTT 用戶端) 中觀看訊息。

### 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
```

[如需 MQTT 以及如何 AWS IoT Core 支援通訊協定的詳細資訊，請參閱 MQTT。](#)

### 在 Python 中執行共享訂閱範例

AWS IoT Core 支援 MQTT 3 和 MQTT 5 的[共用訂閱](#)。共享訂閱允許多個用戶端共享一個主題的訂閱，而且只有一個用戶端會使用隨機分佈接收發佈至該主題的訊息。若要使用共享訂閱，用戶端會訂閱共享訂閱的[主題篩選條件](#)：`$share/{ShareName}/{TopicFilter}`。

### 設定政策並執行共享訂閱範例

1. 若要執行共享訂閱範例，您必須依照 [MQTT 5 共享訂閱](#) 中所述來設定物件的政策。

- 若要執行共享訂閱，請執行下列命令。

## Linux/macOS

在 Linux/macOS 上執行範本指令碼

- 在命令列視窗中，導覽至 SDK 使用這些命令所建立的 `~/aws-iot-device-sdk-python-v2/samples` 目錄。

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

- 在命令行窗口中，*your-iot-endpoint* 按照指示替換並運行此命令。

```
python3 mqtt5_shared_subscription.py --endpoint your-iot-endpoint --ca_file  
~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/  
private.pem.key --group_identifier consumer
```

## Windows

在 Windows PC 上執行範例應用程式

- 在命令列視窗中，導覽至 SDK 所建立的 `%USERPROFILE%\aws-iot-device-sdk-python-v2\samples` 目錄，然後使用這些命令來安裝範例應用程式。

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

- 在命令行窗口中，*your-iot-endpoint* 按照指示替換並運行此命令。

```
python3 mqtt5_shared_subscription.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 --group_identifier  
consumer
```

### Note

執行範例 (例如 `--group_identifier consumer`) 時，您可以根據需求選擇性地指定群組識別碼。如果您沒有指定一個，則 `python-sample` 是預設群組識別碼。

- 此命令列的輸出如下所示：

```
Publisher]: Lifecycle Connection Success
[Publisher]: Connected
Subscriber One]: Lifecycle Connection Success
[Subscriber One]: Connected
Subscriber Two]: Lifecycle Connection Success
[Subscriber Two]: Connected
[Subscriber One]: Subscribed to topic 'test/topic' in shared subscription group
'consumer'.
[Subscriber One]: Full subscribed topic is: '$share/consumer/test/topic' with
SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Subscriber Two]: Subscribed to topic 'test/topic' in shared subscription group
'consumer'.
[Subscriber Two]: Full subscribed topic is: '$share/consumer/test/topic' with
SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [1]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [2]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [3]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [4]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [5]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [6]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [7]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
```

```
[Subscriber One] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [8]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [9]"'
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
    Publish received message on topic: test/topic
    Message: b'"Hello World! [10]"'
[Subscriber One]: Unsubscribed to topic 'test/topic' in shared subscription group
'consumer'.
[Subscriber One]: Full unsubscribed topic is: '$share/consumer/test/topic' with
UnsubAck code: [<UnsubackReasonCode.SUCCESS: 0>]
[Subscriber Two]: Unsubscribed to topic 'test/topic' in shared subscription group
'consumer'.
[Subscriber Two]: Full unsubscribed topic is: '$share/consumer/test/topic' with
UnsubAck code [<UnsubackReasonCode.SUCCESS: 0>]
Publisher]: Lifecycle Disconnected
[Publisher]: Lifecycle Stopped
[Publisher]: Fully stopped
Subscriber One]: Lifecycle Disconnected
[Subscriber One]: Lifecycle Stopped
[Subscriber One]: Fully stopped
Subscriber Two]: Lifecycle Disconnected
[Subscriber Two]: Lifecycle Stopped
[Subscriber Two]: Fully stopped
Complete!
```

4. 在 AWS IoT 主控台中開啟 MQTT 測試用戶端。在訂閱主題中，訂閱共享訂閱的主題，例如：`$share/consumer/test/topic`。執行範例 (例如 `--group_identifier consumer`) 時，您可以根據需求指定群組識別碼。如果您未指定群組識別碼，則預設值為 `python-sample`。如需詳細資訊，請參閱來自《AWS IoT Core 開發人員指南》中的 [MQTT 5 共享訂閱 Python 範例](#)和[共享訂閱](#)。

在您的命令列視窗中，再次執行範例應用程式，並在 AWS IoT 主控台的 MQTT 測試用戶端，以及命令列中觀看訊息的分佈。

**Subscribe to a topic** | **Publish to a topic**

Topic filter [Info](#)  
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

\$share/consumer/test/topic

▶ Additional configuration

**Subscribe**

---

**Subscriptions**

\$share/consumer/test/topic ×

Pause Clear Export Edit

Subscription	Created	Message
test/topic	April 21, 2023, 14:43:10 (UTC-0700)	"Hello World! [10]"
test/topic	April 21, 2023, 14:43:07 (UTC-0700)	"Hello World! [7]"
test/topic	April 21, 2023, 14:43:03 (UTC-0700)	"Hello World! [4]"
test/topic	April 21, 2023, 14:43:00 (UTC-0700)	"Hello World! [1]"

```

[Publisher]: Lifecycle Connection Success
[Publisher]: Connected
[Subscriber One]: Lifecycle Connection Success
[Subscriber One]: Connected
[Subscriber Two]: Lifecycle Connection Success
[Subscriber Two]: Connected
[Subscriber One]: Subscribed to topic 'test/topic' in shared subscription group 'consumer'.
[Subscriber One]: Full subscribed topic is: '$share/consumer/test/topic' with SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]
[Subscriber Two]: Subscribed to topic 'test/topic' in shared subscription group 'consumer'.
[Subscriber Two]: Full subscribed topic is: '$share/consumer/test/topic' with SubAck code: [<SubackReasonCode.GRANTED_QOS_1: 1>]

[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
  Publish received message on topic: test/topic
  Message: b"Hello World! [2]"
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
  Publish received message on topic: test/topic
  Message: b"Hello World! [3]"
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
  Publish received message on topic: test/topic
  Message: b"Hello World! [5]"
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
  Publish received message on topic: test/topic
  Message: b"Hello World! [6]"
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One] Received a publish
  Publish received message on topic: test/topic
  Message: b"Hello World! [8]"
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber Two] Received a publish
  Publish received message on topic: test/topic
  Message: b"Hello World! [9]"
[Publisher]: Sent publish and got PubAck code: <PubackReasonCode.SUCCESS: 0>
[Subscriber One]: Unsubscribed to topic 'test/topic' in shared subscription group 'consumer'.
[Subscriber Two]: Full unsubscribed topic is: '$share/consumer/test/topic' with UnsubAck code: [<UnsubackReasonCode.SUCCESS: 0>]
[Subscriber Two]: Unsubscribed to topic 'test/topic' in shared subscription group 'consumer'.
[Publisher]: Lifecycle Disconnected
[Publisher]: Lifecycle Stopped
[Publisher]: Fully stopped
[Subscriber One]: Lifecycle Disconnected
[Subscriber One]: Lifecycle Stopped
[Subscriber One]: Fully stopped
[Subscriber Two]: Lifecycle Disconnected
[Subscriber Two]: Lifecycle Stopped
[Subscriber Two]: Fully stopped
Complete!

```

## 連接 Raspberry Pi 或其他裝置

在本節中，我們將配置樹莓派以與一起使用 AWS IoT。如果您有另一個想要連接的裝置，Raspberry Pi 的指示會包括參考資料，其可以協助您調整這些指示以適用您的裝置。

這通常約需 20 分鐘，但是如果您的有許多要安裝的系統軟體升級，則所需時間會更長。

在本教學課程中，您將會執行下列動作：

- [設定您的裝置](#)
- [安裝 AWS IoT 裝置 SDK 所需的工具和程式庫](#)
- [安裝 AWS IoT 裝置 SDK](#)
- [安裝並執行範例應用程式](#)
- [在 AWS IoT 主控台中檢視來自範例應用程式的訊息](#)

### Important

調整這些指示以適用其他裝置和作業系統可能是一項艱鉅的挑戰。您必須充分了解您的裝置，才能解譯這些指示並將其套用至您的裝置。

如果您在設定裝置時遇到問題 AWS IoT，您可以嘗試其他裝置選項之一作為替代方案，例如 [使用 Amazon EC2 建立虛擬裝置](#) 或 [使用您的視窗或 Linux 個人電腦或 Mac 作為 AWS IoT 裝置](#)。

## 設定您的裝置

此步驟的目標就是收集您設定裝置所需的資訊，使其可以啟動作業系統 (OS)、連接至網際網路，並可讓您在命令列界面與其互動。

為了完成本教學，您需要以下項目：

- 一個 AWS 帳戶。若您沒有帳戶，請完成 [設定您的 AWS 帳戶](#) 所述的步驟，然後再繼續。
- [Raspberry Pi 3 代 B 型](#) 或更新的型號。這可能適用於較早版本的 Raspberry Pi，但尚未對其進行測試。
- [Raspberry Pi OS \(32 位元\)](#) 或更新版本。我們建議您使用最新版本的 Raspberry Pi 作業系統。較早版本的 OS 可能適用，但尚未對其進行測試。

若要執行此範例，您不需要安裝具有圖形使用者界面 (GUI) 的桌面；但是，如果您是初次使用 Raspberry Pi，而且您的 Raspberry Pi 支援它，使用具有 GUI 的桌面可能會更輕鬆。

- 乙太網路或 WiFi 連線。
- 裝置所需的鍵盤、滑鼠、顯示器、纜線、電源供應器和其他硬體。

### Important

在您繼續下一個步驟之前，您的裝置必須已安裝、設定和執行作業系統。裝置必須連接至網際網路，而且您必須能夠使用其命令列界面存取裝置。可以透過直接連接的鍵盤、滑鼠和顯示器，或透過 SSH 終端機遠端界面存取命令列。

如果您是在具有圖形使用者界面 (GUI) 的 Raspberry Pi 上執行作業系統，請在裝置上開啟終端機視窗，並在該視窗中執行下列指示。否則，如果您是使用遠端終端機 (例如 PuTTY) 連接到您的裝置，請開啟您裝置的遠端終端機並使用該終端機。

## 安裝裝 AWS IoT 置 SDK 所需的工具和程式庫

在安裝裝 AWS IoT 置 SDK 和範例程式碼之前，請確定您的系統為最新狀態，並具備安裝 SDK 所需的工具和程式庫。

### 1. 更新作業系統並安裝必要的程式庫

安裝裝 AWS IoT 置 SDK 之前，請在裝置的終端機視窗中執行這些命令，以更新作業系統並安裝所需的程式庫。

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

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

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

### 2. 安裝 Git

如果您的設備的操作系統沒有安裝 Git，則必須安裝它以安裝 JavaScript. AWS IoT

a. 執行此命令來測試看看是否已安裝 Git。

```
git --version
```

b. 如果上一個命令傳回 Git 版本，表示已安裝 Git，而且您可以跳到步驟 3。

c. 若在執行 git 命令時顯示錯誤，請執行此命令來安裝 Git。

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

d. 執行此命令來再次測試看看是否已安裝 Git。

```
git --version
```

e. 如果已安裝 Git，請繼續進行下一節。如果未安裝，請在繼續之前疑難排解並更正錯誤。您需要 Git 來安 AWS IoT 裝 JavaScript.

## 安裝 AWS IoT 裝置 SDK

安裝裝 AWS IoT 置 SDK。

### Python

在本節中，您將在設備上安裝 Python，其開發工具以及適用於 Python 的設 AWS IoT 備 SDK。這些指示適用於執行最新 Raspberry Pi OS 的 Raspberry Pi。若有其他裝置或正在使用其他作業系統，您可能需要調整這些指示以適用您的裝置。

#### 1. 安裝 Python 及其開發工具

對於 Python 的 AWS IoT 設備 SDK 要求 Python v3.5 或更高版本被安裝在樹莓派。

在裝置的終端機視窗中，執行這些命令。

##### 1. 執行此命令以判定您裝置上安裝的 Python 版本。

```
python3 --version
```

如果已安裝 Python，會顯示其版本。

2. 如果顯示的版本為 Python 3.5 或更新版本，您可以跳到步驟 2。
3. 如果顯示的版本小於 Python 3.5，您可以執行此命令來安裝正確的版本。

```
sudo apt install python3
```

##### 4. 執行此命令以確認現在已安裝正確版本的 Python。

```
python3 --version
```

#### 2. 測試 pip3

在裝置的終端機視窗中，執行這些命令。

##### 1. 執行此命令以查看是否已安裝 pip3。

```
pip3 --version
```

2. 如果命令傳回版本編號，表示已安裝 pip3，而且您可以跳到步驟 3。
3. 如果上一個命令傳回錯誤，請執行此命令來安裝 pip3。



```
sudo apt install python3-pip
```

4. 執行此命令以查看是否已安裝 pip3。

```
pip3 --version
```

3. 安裝目前的 Python AWS IoT 裝置開發套件

安裝適用於 Python 的 AWS IoT 裝置 SDK，並將範例應用程式下載到您的裝置。

在您的裝置上，執行這些命令。

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

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

## JavaScript

在本節中，您將在設備 JavaScript 上安裝 Node.js，npm 軟件包管理器和設 AWS IoT 備 SDK。這些指示適用於執行 Raspberry Pi OS 的 Raspberry Pi。若有其他裝置或正在使用其他作業系統，您可能需要調整這些指示以適用您的裝置。

1. 安裝最新版本的 Node.js

設 AWS IoT 備 SDK JavaScript 需要 Node.js 和 NPM 包管理器要安裝在您的樹莓派。

- a. 輸入此命令，下載最新版本的 Node 儲存庫。

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

- b. 安裝 Node 和 npm。

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

- c. 驗證 Node 的安裝。

```
node -v
```

確認命令顯示 Node 版本。本教學課程需要 Node v10.0 或更新版本。如果未顯示 Node 版本，請嘗試再次下載 Node 儲存庫。

- d. 驗證 npm 的安裝。

```
npm -v
```

確認命令顯示 npm 版本。如果未顯示 npm 版本，請嘗試再次安裝 Node 和 npm。

- e. 重新啟動裝置。

```
sudo shutdown -r 0
```

在裝置重新啟動之後繼續。

## 2. 安裝下列項目的 AWS IoT 裝置 SDK JavaScript

安裝設 AWS IoT 備 SDK 為您 JavaScript 的樹莓派。

- a. 將 JavaScript 儲存庫的 AWS IoT 裝置 SDK 複製到 `#aws-iot-device-sdk-js-v2` 目錄的目錄中。在 Raspberry Pi 上，`home` 目錄是 `~/`，用作下列命令中的 `home` 目錄。如果您的裝置對 `home` 目錄使用不同的路徑，您必須在下列命令中將 `~/` 取代為您裝置的正確路徑。

這些命令會建立 `~/aws-iot-device-sdk-js-v2` 目錄並將 SDK 程式碼複製到其中。

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

- b. 變更為您在上述步驟中建立的 `aws-iot-device-sdk-js-v2` 目錄，並執行 `npm install` 以安裝 SDK。命令 `npm install` 將會呼叫 `aws-crt` 程式庫建置，可能需要幾分鐘時間才能完成。

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

## 安裝並執行範例應用程式

在本節中，您將安裝並運行在 AWS IoT 設備 SDK 中找到的 pubsub 示例應用程序。此應用程式顯示您的裝置如何使用 MQTT 程式庫來發佈和訂閱 MQTT 訊息。範例應用程式會訂閱主題 `topic_1`、將 10 則訊息發佈至該主題，以及在從訊息代理程式接收訊息時顯示這些訊息。

### 安裝憑證檔案

範例應用程式需要驗證裝置是否安裝在裝置上的憑證檔案。

#### 安裝範例應用程式的裝置憑證檔案

1. 執行這些命令，在您的 *home* 目錄中建立 `certs` 子目錄。

```
cd ~  
mkdir certs
```

2. 將私有金鑰、裝置憑證，以及您先前在 [the section called “建立 AWS IoT 資源”](#) 中建立的根憑證授權機構憑證複製到 `~/certs` 目錄中。

將憑證檔案複製到裝置的方式取決於裝置和作業系統，此處不會加以描述。不過，如果您的裝置支援圖形使用者界面 (GUI) 且具有網頁瀏覽器，您可以從裝置的網頁瀏覽器執行 [the section called “建立 AWS IoT 資源”](#) 中所述的程序，將產生的檔案直接下載到您的裝置。

下節命令假設您的金鑰和憑證檔案儲存在裝置上，如下表所示。

#### 憑證檔案名稱

檔案	檔案路徑
根 CA 憑證	<code>~/certs/Amazon-root-CA-1.pem</code>
裝置憑證	<code>~/certs/device.pem.crt</code>
私有金鑰	<code>~/certs/private.pem.key</code>

若要執行範例應用程式，您需要以下資訊：

## 應用程式參數值

參數	可在哪裡找到值
<i>your-iot-endpoint</i>	<p>於 <a href="#">AWS IoT 主控台</a> 中，依序選擇 Manage (管理) 和 Things (物件)。</p> <p>在 AWS IoT 功能表中的 [設定] 頁面上。您的端點會顯示在 Device data endpoint (裝置資料端點) 區段中。</p>

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

## Python

### 安裝並執行範例應用程式

1. 導覽至範例應用程式目錄。

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

2. 在命令行窗口中，按 *your-iot-endpoint* 照指示替換並運行此命令。

```
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
```

3. 觀察範例應用程式：

1. 連線至您帳戶的 AWS IoT 服務。
2. 訂閱訊息主題 `topic_1`，並顯示其收到關於該主題的訊息。
3. 將 10 則訊息發佈至主題 `topic_1`。
4. 顯示類似下列內容的輸出：

```
Connecting to a3qEXAMPEffp-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!
```

若在執行範例應用程式時發生問題，請檢閱 [the section called “疑難排解範例應用程式的問題”](#)。

您也可以將 `--verbosity Debug` 參數新增至命令列，以便範例應用程式顯示有關其正在做什麼的詳細訊息。這些資訊可能會提供您更正問題所需的協助。

## JavaScript

### 安裝並執行範例應用程式

1. 在命令列視窗中，導覽至 SDK 所建立的 `~/aws-iot-device-sdk-js-v2/samples/node/pub_sub` 目錄，然後使用這些命令來安裝範例應用程式。命令 `npm install` 將會呼叫 `aws-crt` 程式庫建置，可能需要幾分鐘時間才能完成。

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

```
npm install
```

2. 在命令行窗口中，按`your-iot-endpoint`照指示替換並運行此命令。

```
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
```

3. 觀察範例應用程式：

1. 連線至您帳戶的 AWS IoT 服務。
2. 訂閱訊息主題 `topic_1`，並顯示其收到關於該主題的訊息。
3. 將 10 則訊息發佈至主題 `topic_1`。
4. 顯示類似下列內容的輸出：

```
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 }
```

若在執行範例應用程式時發生問題，請檢閱 [the section called “疑難排解範例應用程式的問題”](#)。

您也可以將 `--verbosity Debug` 參數新增至命令列，以便範例應用程式顯示有關其正在做什麼的詳細訊息。這些資訊可能會提供您更正問題所需的協助。

在 AWS IoT 主控台中檢視來自範例應用程式的訊息

您可以在範例應用程式的訊息透過訊息代理程式傳遞時看到這些訊息，方法為使用 AWS IoT 主控台內的 MQTT test client (MQTT 測試用戶端)。

檢視範例應用程式所發佈的 MQTT 訊息

1. 檢閱 [使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)。這有助於您了解如何使用 AWS IoT 主控台內的 MQTT 測試用戶端，在 MQTT 訊息通過訊息代理程式時，檢視訊息。
2. 在 AWS IoT 主控台中開啟 MQTT 測試用戶端。
3. 訂閱主題 `topic_1`。
4. 在您的命令列視窗中，再次執行範例應用程式，並在 AWS IoT 主控台的 MQTT client (MQTT 用戶端) 中觀看訊息。

## 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/device.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/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-
endpoint
```

## 疑難排解範例應用程式的問題

如果您在嘗試執行範例應用程式時遇到錯誤，以下是一些需要檢查的事項。

## 檢查憑證

如果憑證未處於作用中狀態，AWS IoT 將不接受任何使用憑證進行授權的連線嘗試。建立憑證時，很容易忽略 Activate (啟用) 按鈕。幸好，您可以從 [AWS IoT 主控台](#) 啟用憑證。

### 檢查您的憑證是否啟用

1. 在 [AWS IoT 主控台](#) 的左側選單中，依序選擇 Secure (安全) 和 Certificates (憑證)。
2. 在憑證清單中，找出您為練習建立的憑證，並在 Status (狀態) 欄位中檢查其狀態。

如果您不記得憑證的名稱，請檢查是否有任何為 Inactive (非作用中) 的憑證，以查看它們是否可能是您正在使用的憑證。

選擇清單中的憑證來開啟其詳細資訊頁面。在詳細資訊頁面中，您可以看到其 Create date (建立日期) 來協助您識別憑證。

3. 若要啟動非作用中的憑證，請從憑證的詳細資訊頁面中，選擇 Actions (動作)，然後選擇 Activate (啟用)。

如果您找到正確的憑證且為作用中，但執行範例應用程式時仍有問題，請檢查其政策，如下一個步驟所述。

您也可以嘗試遵循 [the section called “建立物件”](#) 中的步驟，建立新的物件和新的憑證。如果建立新的物件，您需要給它新的物件名稱，並將新的憑證檔案下載到您的裝置。

### 檢查連接至憑證的政策

政策授權中的動作 AWS IoT。如果用來連接至 AWS IoT 的憑證沒有政策，或者沒有允許它連接的政策，則即使憑證處於作用中狀態，連線也會遭到拒絕。

### 檢查連接至憑證的政策

1. 按照上一個項目中所述尋找憑證，並開啟其詳細資訊頁面。
2. 於憑證詳細資訊頁面的左側選單中，選擇 Policies (政策)，查看連接至憑證的政策。
3. 如果沒有任何政策連接至憑證，請選擇 Actions (動作) 選單，然後選擇 Attach policy (連接政策) 來新增一個政策。

選擇您先前在 [the section called “建立 AWS IoT 資源”](#) 中建立的政策。

4. 若有連接的政策，請選擇政策圖標來開啟其詳細資訊頁面。



在詳細資訊頁面中，檢閱 Policy document (政策文件)，以確定它與您在 [the section called “建立 AWS IoT 策略”](#) 中建立的政策包含相同的資訊。

### 檢查命令列

確定您已針對系統使用正確的命令列。在 Linux 和 macOS 系統上使用的命令通常與在 Windows 系統上使用的命令不同。

### 檢查端點地址

檢閱您輸入的命令，然後仔細檢查命令中的端點地址是否為 [AWS IoT 主控台](#) 中的端點地址。

### 檢查憑證檔案的檔案名稱

將您輸入至命令中的檔案名稱與 certs 目錄中的憑證檔案名稱進行比較。

有些系統可能需要以引號括住檔案名稱才能正常運作。

### 檢查 SDK 安裝

確定您的 SDK 安裝完成且正確。

如有疑問，請在裝置上重新安裝 SDK。在大多數情況下，這是找到標題為安裝 SDK **##** 的 AWS IoT 設備 SDK 教程的部分的問題，然後再次按照該過程進行操作。

如果您使用的是裝 AWS IoT 置 SDK JavaScript，請記得先安裝範例應用程式，然後再嘗試執行它們。安裝 SDK 不會自動安裝範例應用程式。在安裝了 SDK 之後必須手動安裝範例應用程式。

## 使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT

本節說明如何使用 [AWS IoT 主控台](#) 中的 AWS IoT MQTT 測試用戶端來觀看由傳送和接收的 MQTT 訊息。AWS IoT 本節中使用的範例與 [開始使用 AWS IoT Core](#) 中使用的範例相關；不過，您可以將範例中使用的 *topicName* 取代為 IoT 解決方案所使用的任何 [主題名稱或主題篩選條件](#)。

裝置會發佈由 [主題](#) 識別的 MQTT 訊息以傳達其狀態 AWS IoT，並 AWS IoT 發佈 MQTT 訊息以通知裝置和應用程式有變更和事件。您可以使用 MQTT 用戶端來訂閱這些主題，並在發生訊息時觀看它們。您也可以使用 MQTT 測試用戶端，將 MQTT 訊息發佈到您的 AWS 帳戶

### 目錄

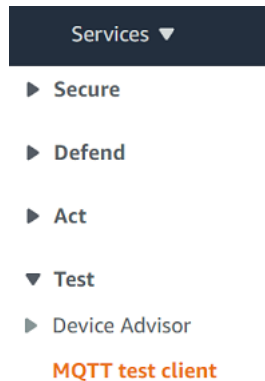
- [在 MQTT 用戶端中檢視 MQTT 訊息](#)

- [從 MQTT 用戶端發佈 MQTT 訊息](#)
- [在 MQTT 用戶端中測試共享訂閱](#)

## 在 MQTT 用戶端中檢視 MQTT 訊息

在 MQTT 測試用戶端中檢視 MQTT 訊息

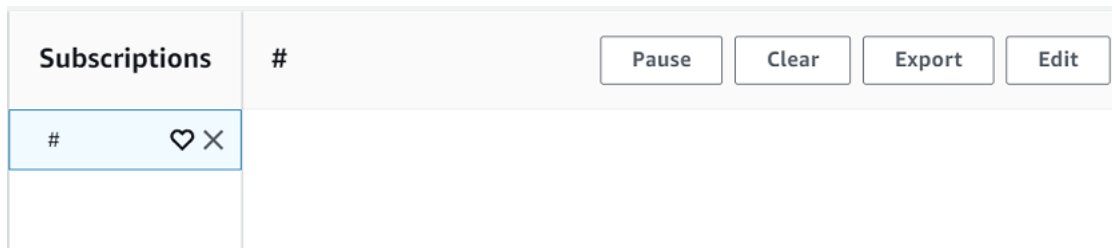
1. 在 [AWS IoT 主控台](#) 中的左側選單中，依序選擇 Test (測試) 和 MQTT test client (MQTT 測試用戶端)。



2. 在 Subscribe to a topic (訂閱主題) 索引標籤中，輸入 *topicName* 來訂閱裝置發佈的主題。如需入門範例應用程式，請訂閱 #，其會訂閱所有訊息主題。

繼續入門範例，在 Subscribe to a topic (訂閱主題) 標籤的 Topic filter (主題篩選條件) 欄位中，輸入 #，然後選擇 Subscribe (訂閱)。

主題訊息日誌頁面 # 即會開啟，而且 # 會出現在 Subscriptions (訂閱) 清單中。如果您配置的設備 [the section called “設定您的裝置”](#) 正在運行示例程序，則應該在 # 消息日誌 AWS IoT 中看到它發送的消息。當收到已訂閱主題的訊息時，訊息記錄項目會出現在 [發佈] 區段下方 AWS IoT。



3. 在 # 訊息日誌頁面上，您也可以將訊息發佈至主題，但您必須指定主題名稱。您無法發佈至 # 主題。

發佈至已訂閱主題的訊息會在收到時出現在訊息日誌中，最近的訊息首先出現。

## 疑難排解 MQTT 訊息

### 使用萬用字元主題篩選條件

如果您的訊息未如預期般顯示在訊息日誌中，請嘗試訂閱萬用字元主題篩選條件，如 [主題篩選條件](#) 中所述。MQTT 多層萬用字元主題篩選條件是雜湊或井號 (#)，並且可以用作 Subscription topic (訂閱主題) 欄位中的主題篩選條件。

訂閱 # 主題篩選條件會訂閱訊息代理程式接收的每個主題。您可以縮小篩選範圍，方法是將主題篩選條件路徑的元素取代為 # 多層萬用字元或 '+' 單層萬用字元。

### 在主題篩選條件中使用萬用字元時

- 多層萬用字元必須是主題篩選條件中的最後一個字元。
- 主題篩選條件路徑的每個主題層只能有一個單層萬用字元。

例如：

主題篩選條件	顯示訊息與
#	任何主題名稱
topic_1/#	開頭為 topic_1/ 的主題名稱
topic_1/level_2/#	開頭為 topic_1/level_2/ 的主題名稱
topic_1/+/level_3	開頭為 topic_1/、結尾為 /level_3，並且在兩者之間有任何值之一個元素的主題名稱。

如需主題篩選條件的詳細資訊，請參閱 [主題篩選條件](#)。

## 檢查主題名稱錯誤

MQTT 主題和主題篩選條件會區分大小寫。例如，如果您的裝置正在將訊息發佈至 Topic\_1 (具有大寫 T) 而不是 topic\_1 (您已訂閱的主題)，其訊息將不會出現在 MQTT 測試用戶端中。不過，訂閱萬用字元主題篩選條件會顯示裝置正在發佈訊息，而且您可以看到它使用的主題名稱不是您預期的。

## 從 MQTT 用戶端發佈 MQTT 訊息

### 將訊息發佈至 MQTT 主題

1. 在 MQTT 測試用戶端頁面的 Publish to a topic (發佈到主題) 標籤中，於 Topic name (主題名稱) 欄位中輸入您訊息的 *topicName*。在此範例中，使用 **my/topic**。

#### Note

無論是在 MQTT 測試用戶端還是在系統實作中使用主題名稱，請不要在這些主題名稱中使用個人識別資訊。主題名稱可以出現在未加密的通訊和報告中。

2. 在訊息承載視窗中，輸入下列 JSON：

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

3. 選擇 Publish (發佈) 以將您的訊息發佈至 AWS IoT。

#### Note

確定您已訂閱 my/topic 主題，然後再發佈您的訊息。

**Subscribe to a topic** | **Publish to a topic**

**Topic name**  
The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

Q my/topic X

**Message payload**

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

▶ **Additional configuration**

**Publish**

4. 在 Subscriptions (訂閱) 清單中，選擇 my/topic 來查看訊息。您應該會在發佈訊息承載視窗下方看到訊息出現在 MQTT 測試用戶端中。

**Subscriptions** # Pause Clear Export Edit

#	
my/topic	November 02, 2021, 11:55:22 (UTC-0700)

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

您可以變更 *Topic name* (主題名稱) 欄位中的 topicName，然後選擇 Publish (發佈) 按鈕，以將 MQTT 郵件發佈至其他主題。

**⚠ Important**

當您建立具有重疊主題的多個訂閱 (例如，probe1/溫度和probe1/#) 時，發佈至符合兩個訂閱的主題的單一訊息可能會多次傳遞，每個重疊的訂閱一次。

## 在 MQTT 用戶端中測試共享訂閱

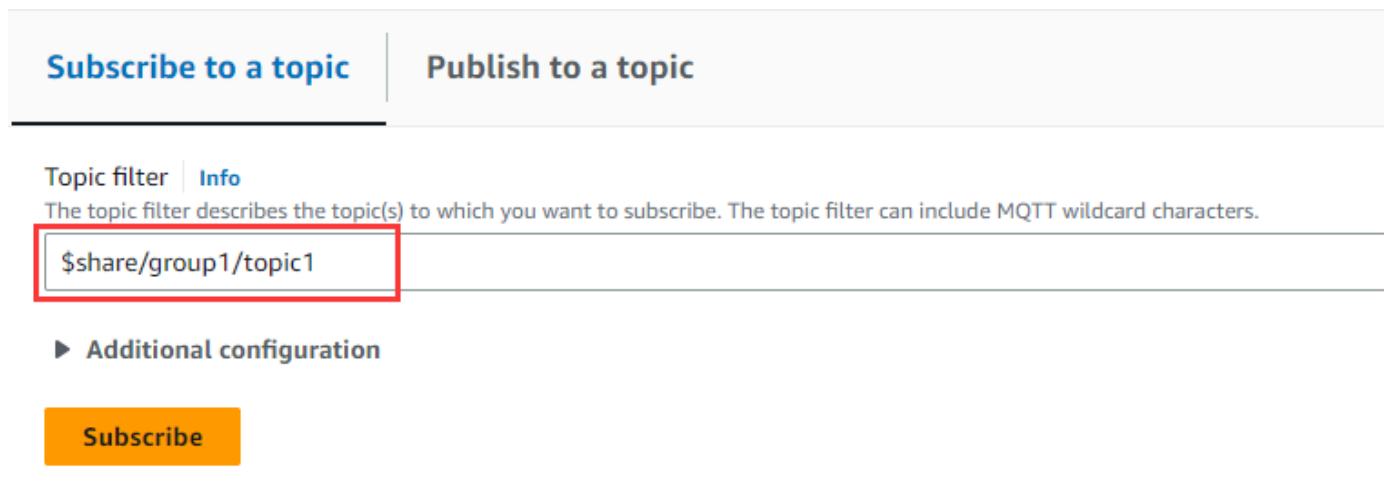
本節說明如何使用 [AWS IoT 主控台](#) 中的 AWS IoT MQTT 用戶端來觀看使用共用訂閱傳送和接收的 MQTT 訊息。AWS IoT [MQTT](#) 允許多個用戶端共用主題的訂閱，只有一個用戶端接收使用隨機發佈到該主題發佈的訊息。[若要模擬共用相同訂閱的多個 MQTT 用戶端 \(在此範例中為兩個 MQTT 用戶端\)，請從多個 Web 瀏覽器在主控台中開啟 AWS IoT MQTT 用戶端。AWS IoT 本節中使用的範例與 \[開始使用 AWS IoT Core\]\(#\) 中使用的範例無關。如需詳細資訊，請參閱 \[共享訂閱\]\(#\)。](#)

### 共享 MQTT 主題的訂閱

1. 在 [AWS IoT 主控台](#) 的導覽窗格中，選擇測試，然後選擇 MQTT 測試用戶端。
2. 在 Subscribe to a topic (訂閱主題) 索引標籤中，輸入 *topicName* 來訂閱裝置發佈的主題。若要使用共享訂閱，請訂閱共享訂閱的主題篩選條件，如下所示：

```
$share/{ShareName}/{TopicFilter}
```

範例主題篩選條件可以是 `$share/group1/topic1`，其會訂閱訊息主題 `topic1`。



The screenshot shows the 'Subscribe to a topic' interface in the AWS IoT console. It features two tabs: 'Subscribe to a topic' (active) and 'Publish to a topic'. Under the active tab, there is a 'Topic filter' section with an 'Info' icon. A text box below it contains the filter string '\$share/group1/topic1', which is highlighted with a red rectangular border. Below the text box is a link for 'Additional configuration' and a prominent orange 'Subscribe' button.

3. 開啟另一個 Web 瀏覽器，然後重複步驟 1 和步驟 2。透過此方式，您模擬兩個共享相同訂閱 `$share/group1/topic1` 的不同 MQTT 客戶端。
4. 選擇 MQTT 用戶端，在發佈到主題標籤的主題名稱欄位中輸入您訊息的 *topicName*。在此範例中，使用 `topic1`。請嘗試發佈訊息幾次。從這兩個 MQTT 用戶端的訂閱清單中，您應該能夠看到用戶端使用隨機分佈接收訊息。在這個範例中，我們發佈相同的訊息 "Hello from AWS IoT console" 三次。左側的 MQTT 用戶端收到兩次訊息，右側的 MQTT 用戶端收到一次訊息。

**Subscribe to a topic** | **Publish to a topic**

Topic filter [Info](#)  
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

▶ Additional configuration

**Subscribe**

---

**Subscriptions** **\$share/group1/topic1**

[♥](#) [✕](#)

Message payload

```
{  
  "message": "Hello from AWS IoT console"  
}
```

▶ Additional configuration

**Publish**

No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.

**Subscribe to a topic** | **Publish to a topic**

Topic filter [Info](#)  
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

▶ Additional configuration

**Subscribe**

---

**Subscriptions** **\$share/group1/topic1**

[♥](#) [✕](#)

Message payload

```
{  
  "message": "Hello from AWS IoT console"  
}
```

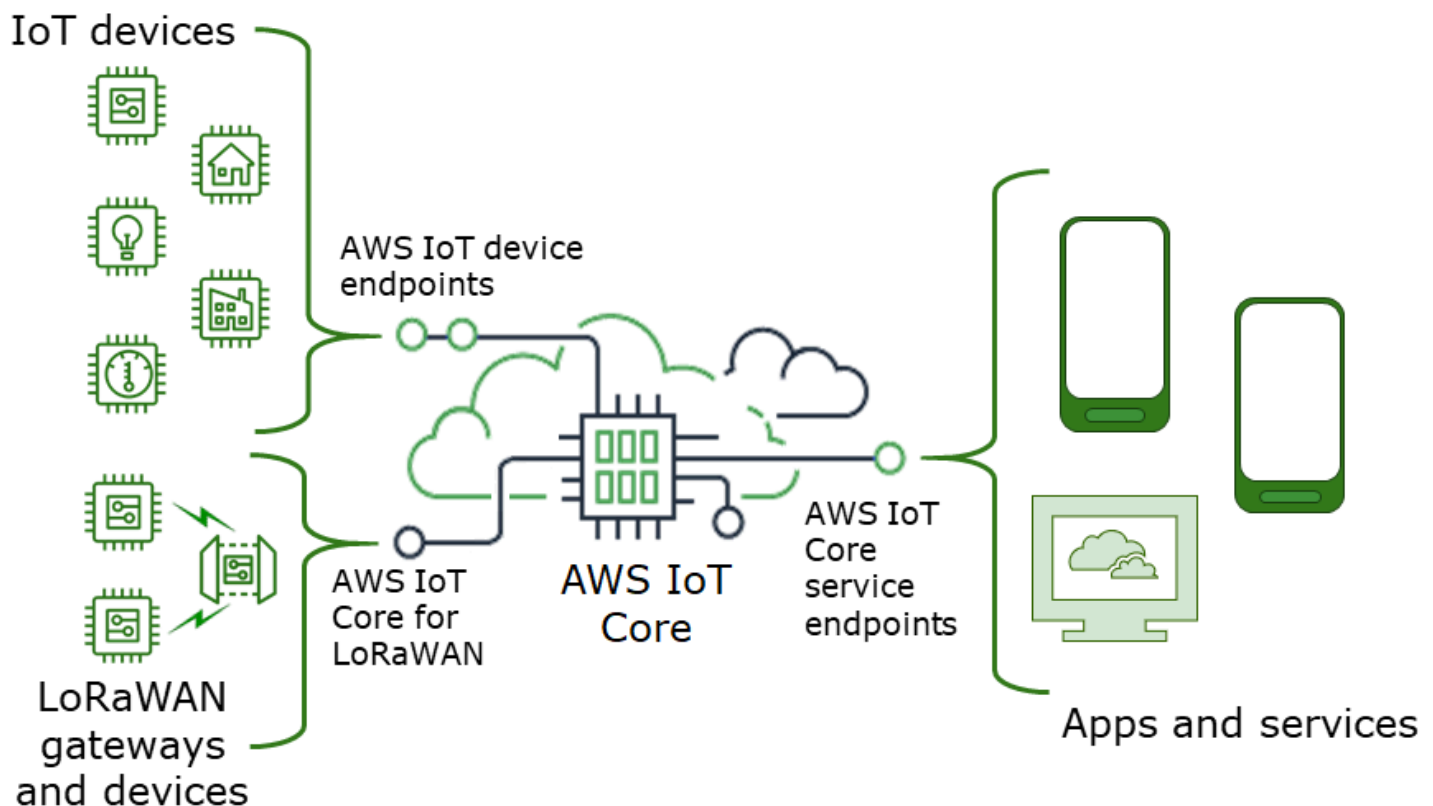
▶ Additional configuration

**Publish**

No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.

## 連接到 AWS IoT Core

AWS IoT Core 支援與 IoT 裝置、無線閘道、服務和應用程式的連線。裝置會連線到，以 AWS IoT Core 便將資料傳送至 AWS IoT 服務和其他裝置，以及從其他裝置接收資料。應用程式和其他服務也會連線 AWS IoT Core 到，以控制和管理 IoT 裝置，以及處理來自 IoT 解決方案的資料。本節說明如何 AWS IoT Core 針對 IoT 解決方案的各個層面選擇最佳連線和通訊方式。



有幾種方法可以與之互動 AWS IoT。應用程式和服務可以使用 [AWS IoT Core：控制平面端點](#) 和裝置可以使 AWS IoT Core 用 [AWS IoT 裝置端點](#) 或用 [AWS IoT Core 於 LoRa WAN 區域和端點](#) 來連線。

### AWS IoT Core：控制平面端點

控制平面端點可讓您存取控制和管理 AWS IoT 解決方案的功能。AWS IoT Core

- 端點

AWS IoT Core - 控制平面和 AWS IoT Core Device Advisor 控制平面端點是 [AWS IoT Core 端點與配額](#) 中列出的特定區域。端點的格式如下。



端點用途	端點格式	服務
AWS IoT Core : 控制平面	<code>iot.<i>aws-region</i> n .amazonaws.com</code>	<a href="#">AWS IoT 控制平面 API</a>
AWS IoT Core 裝置建議程式- 控制平面	<code>api.iotdeviceadvisor.<i>aws-region</i> n .amazonaws.com</code>	<a href="#">AWS IoT Core 裝置建議程式 控制平面 API</a>

- 軟體開發套件和工具

[AWS SDK](#) 為 API 和其他服務的 AWS IoT Core API 提供特定語言支援。AWS [AWS Mobile SDK](#) 為應用程式開發人員提供 AWS IoT Core API 的平台專屬支援，以及行動裝置上的其他 AWS 服務。

提[AWS CLI](#)供 AWS IoT 服務端點所提供之功能的命令列存取。AWS PowerShell提[供工具](#)來管理 PowerShell 指令碼環境中的 AWS 服務和資源。

- 身分驗證

服務端點使用 IAM 使用者和 AWS 登入資料來驗證使用者。

- 進一步了解

如需詳細資訊和軟體開發套件參考的連結，請參閱 [the section called “連線至 AWS IoT Core 服務端點”](#)。

## AWS IoT 裝置端點

裝 AWS IoT 置端點支援 IoT 裝置與 AWS IoT.

- 端點

設備端點支持 AWS IoT Core 和 AWS IoT Device Management 功能。它們是特定於您的 AWS 帳戶，您可以使用[describe-endpoint](#)命令查看它們是什麼。

端點用途	端點格式	服務
AWS IoT Core : 資料平面	請參閱 <a href="#">???</a> 。	<a href="#">AWS IoT 資料平面 API</a>

端點用途	端點格式	服務
AWS IoT Device Management: 任務資料	請參閱 <a href="#">???</a> 。	<a href="#">AWS IoT 工作資料平面 API</a>
AWS IoT 裝置建議程式-資料平面	請參閱 <a href="#">???</a> 。	不適用
AWS IoT Device Management: Fleet Hub	不適用	不適用
AWS IoT Device Management: 安全通道	api.tunneling.iot. <i>aws-region</i> .amazonaws.com	<a href="#">AWS IoT 安全通道 API</a>

如需這些端點及其支援之功能的詳細資訊，請參閱 [the section called “AWS IoT 裝置資料和服務端點”](#)。

- 軟體開發套件

[AWS IoT 裝置 SDK](#) 針對裝置用於通訊的訊息佇列遙測傳輸 (MQTT) 和 WebSocket 安全 (WSS) 通訊協定提供特定語言支援。AWS IoT [AWS 行動開發套件](#) 還為 MQTT 設備通信，AWS IoT API 和移動設備上其他 AWS 服務的 API 提供支持。

- 身分驗證

裝置端點使用具有登入資料的 X.509 憑證或 AWS IAM 使用者來驗證使用者。

- 進一步了解

如需詳細資訊和軟體開發套件參考的連結，請參閱 [the section called “AWS IoT 裝置開發套件”](#)。

## AWS IoT Core 適用於 LoRa WAN 閘道器和裝置

AWS IoT Core 應用於 LoRa WAN 將無線閘道器和設備連接到 AWS IoT Core。

- 端點

AWS IoT Core 用於 LoRa WAN 會管理與帳戶和區域特定 AWS IoT Core 端點的閘道連線。閘道可以連線到 LoRa WAN 提供的帳戶組態和更新伺服器 (CUPS) 端點。AWS IoT Core

端點用途	端點格式	服務
組態與更新伺服器 (CUPS)	<i>account-specific-prefix</i> .cups.lorawan. <i>aws-region</i> .amazonaws.com:443	與 LoRa WAN 所提供之組態及更新伺服器 AWS IoT Core 的閘道通訊
LoRa 廣域網路伺服器	<i>account-specific-prefix</i> .gateway.lorawan. <i>aws-region</i> .amazonaws.com:443	與 WAN 提供的 LoRa WAN 網絡服務器的網關通 AWS IoT Core 信 LoRa

- 軟體開發套件

SDK 支援 AWS IoT Core 針對 LoRa WAN 建置的 AWS IoT 無線 AWS API。如需詳細資訊，請參閱 [AWS 開發套件與工具組](#)。

- 身分驗證

AWS IoT Core 針對 LoRa WAN 裝置通訊，請使用 X.509 憑證來保護與 AWS IoT。

- 進一步了解

如需有關設定和連線無線裝置的詳細資訊，請 [AWS IoT Core 參閱 LoRa WAN 區域和端點](#)。

## 連線至 AWS IoT Core 服務端點

您可以使用您慣用語言的 AWS SDK AWS CLI，或直接呼叫 REST API 來存取 AWS IoT Core-控制平面的功能。我們建議您使用 AWS CLI 或 AWS SDK 與之互動，AWS IoT Core 因為它們包含了呼叫 AWS 服務的最佳做法。直接呼叫 REST API 是一個選項，但您必須提供 [必要的安全憑證](#) 以便存取 API。

### Note

IoT 裝置應使用 [AWS IoT 裝置開發套件](#)。裝置 SDK 已針對裝置使用最佳化、支援 MQTT 通訊 AWS IoT，並支援裝置最常使用的 AWS IoT API。如需裝置軟體開發套件及其提供之功能的詳細資訊，請參閱 [AWS IoT 裝置開發套件](#)。

行動裝置應使用 [AWS 行動開發套件](#)。行動開發套件可支援 AWS IoT API、MQTT 裝置通訊，以及行動裝置上其他 AWS 服務的 API。如需 Mobile 軟體開發套件及其提供之功能的詳細資訊，請參閱 [AWS 行動開發套件](#)。

您可以使用 Web 和移動應用程式中的 AWS Amplify 工具和資源來更輕鬆地連接 AWS IoT Core。如需有關使用 Amplify 連線 AWS IoT Core 的詳細資訊，請參閱 Amplify 文件中的 [Pub 子入門](#)。

下列各節說明您可用來開發及與其他 AWS 服務互動的工具 AWS IoT 和 SDK。如需可用於建置和管理應用程式的 AWS 工具和開發套件的完整清單 AWS，請參閱 [建置的工具 AWS](#)。

## AWS CLI 對於 AWS IoT Core

提 AWS CLI 供 AWS API 的命令列存取權。

- 安裝

如需有關如何安裝的資訊 AWS CLI，請參閱 [安裝 AWS CLI](#)。

- 身分驗證

AWS CLI 使用您的 AWS 帳戶。

- 參考資料

如需這些 AWS IoT Core 服務之 AWS CLI 命令的相關資訊，請參閱：

- [AWS CLI IoT 的命令參考](#)
- [AWS CLI IoT 資料的命令參考](#)
- [AWS CLI IoT 工作資料的命令參考](#)
- [AWS CLI IoT 安全通道的命令參考](#)

如需在 PowerShell 指令碼環境中管理 AWS 服務和資源的 [AWS 工具](#)，請參閱 PowerShell。

## AWS 開發套件

借 AWS 助 SDK，您的應用程式和兼容設備可以調用其他 AWS 服務的 AWS IoT API 和 API。本節提供 AWS SDK 和 AWS IoT Core 服務 API 參考文件的連結。

AWS 開發套件支援這些 API AWS IoT Core

- [AWS IoT](#)

- [AWS IoT 資料平面](#)
- [AWS IoT 工作資料平面](#)
- [AWS IoT 安全隧道](#)
- [AWS IoT 無線](#)

## C++

若要安裝 [AWS SDK for C++](#) 並使用它來連接至 AWS IoT：

1. 遵循 [開始使用適用於 C++ 的 AWS SDK](#) 中的指示

這些指示說明如何：

- 從來源檔案安裝和建置軟體開發套件
  - 提供憑證以搭配您的 AWS 帳戶使用軟體開發套件
  - 在您的應用程式或服務中初始化和關閉軟體開發套件
  - 建立一個 CMake 專案來建置應用程式或服務
2. 建立並執行範例應用程式。如需使用適用於 C++ 的 AWS SDK 的範例應用程式，請參閱 [AWS SDK for C++ 程式碼範例](#)。

AWS SDK for C++ 支援 AWS IoT Core 服務的說明文件

- [AWS::IoTClient" 參考文件](#)
- [Aws::IoT:DataPlane: IoT DataPlaneClient 參考文檔](#)
- [Aws::IoT:JobsDataPlane: IoT JobsDataPlaneClient 參考文檔](#)
- [Aws::IoT:SecureTunneling: IoT SecureTunnelingClient 參考文檔](#)

## Go

若要安裝 [AWS SDK for Go](#) 並使用它來連接至 AWS IoT：

1. 按照 [入門中的說明進行操作 AWS SDK for Go](#)

這些指示說明如何：

- 安裝 AWS SDK for Go
- 取得軟體開發套件的存取金鑰以存取您的 AWS 帳戶

- 將套件匯入我們應用程式或服務的原始程式碼
2. 建立並執行範例應用程式。如需使用 AWS SDK for Go 的範例應用程式，請參閱 [AWS SDK for Go 程式碼範例](#)。

AWS SDK for Go 支援 AWS IoT Core 服務的說明文件

- [物聯網參考文件](#)
- [IoT DataPlane 參考文件](#)
- [IoT JobsDataPlane 參考文件](#)
- [IoT SecureTunneling 參考文件](#)

## Java

若要安裝 [AWS SDK for Java](#) 並使用它來連接至 AWS IoT：

1. 遵循 [開始使用中的](#) 指示 AWS SDK for Java 2.x

這些指示說明如何：

- 註冊 AWS 並建立 IAM 使用者
  - 下載軟體開發套件
  - 設定 AWS 憑證和區域
  - 使用軟體開發套件搭配 Apache Maven
  - 使用軟體開發套件搭配 Gradle
2. 使用其中一個 [AWS SDK for Java 2.x 程式碼範例](#) 來建立和執行範例應用程式。
  3. 檢閱 [軟體開發套件 API 參考文件](#)

AWS SDK for Java 支援 AWS IoT Core 服務的說明文件

- [IoTClient 參考文件](#)
- [IoTDataPlaneClient 參考文件](#)
- [IoTJobsDataPlaneClient 參考文件](#)
- [IoT SecureTunnelingClient 參考文件](#)

## JavaScript

若要安裝 AWS SDK for JavaScript 並使用它來連線到 AWS IoT：

1. 遵循[設定 AWS SDK for JavaScript](#) 中的指示。這些指示適用於 AWS SDK for JavaScript 在瀏覽器和 Node.JS 中使用。請確定您遵循適用於安裝的指示。

這些指示說明如何：

- 查看先決條件
  - 為下列項目安裝 SDK JavaScript
  - 載入下列項目的 SDK JavaScript
2. 建立並執行範例應用程式，以開始使用軟體開發套件，如您環境的入門選項所描述。
    - 在[瀏覽器 JavaScript 中開始使用的AWS SDK](#)，或
    - [JavaScript 在 Node.js 中開始使用適用於的軟體開發AWS 發套件](#)

AWS SDK for JavaScript 支援 AWS IoT Core 服務的說明文件

- [AWS.Iot reference documentation](#)
- [AWS.IotData reference documentation](#)
- [AWS.IotJobsDataPlane reference documentation](#)
- [AWS.IotSecureTunneling reference documentation](#)

## .NET

若要安裝 [AWS SDK for .NET](#) 並使用它來連接至 AWS IoT：

1. 依照設定[AWS SDK for .NET 環境中的指示](#)進行
2. 按照[設置 AWS SDK for .NET 項目中的說明](#)進行操作

這些指示說明如何：

- 啟動新的專案
  - 取得並設定 AWS 認證
  - 安裝 AWS SDK 套件
3. 在 [AWS SDK for .NET](#) 中使用 AWS 服務中建立並執行其中一個範例程式
  4. 檢閱[軟體開發套件 API 參考文件](#)

AWS SDK for .NET 支援 AWS IoT Core 服務的說明文件

- [Amazon.IoT.Model 參考文件](#)
- [Amazon。IoTData. 模型參考文件](#)
- [亞馬遜JobsDataPlane. 物聯網. 模型參考文檔](#)
- [亞馬遜SecureTunneling. 物聯網. 模型參考文檔](#)

## PHP

若要安裝 [AWS SDK for PHP](#) 並使用它來連接至 AWS IoT：

1. 遵循「[開始使用 AWS SDK for PHP 版本 3](#)」中的指示

這些指示說明如何：

- 查看先決條件
  - 安裝軟體開發套件
  - 將軟體開發套件套用於 PHP 指令碼
2. 使用其中一個[AWS SDK for PHP 第 3 版程式碼範例](#)來建立和執行範例應用程式

AWS SDK for PHP 支援 AWS IoT Core 服務的說明文件

- [IoTClient 參考文件](#)
- [IoT DataPlaneClient 參考文件](#)
- [IoT JobsDataPlaneClient 參考文件](#)
- [IoT SecureTunnelingClient 參考文件](#)

## Python

若要安裝 [AWS SDK for Python \(Boto3\)](#) 並使用它來連接至 AWS IoT：

1. 按照[AWS SDK for Python \(Boto3\) 快速入門](#)中的指示操作

這些指示說明如何：

- 安裝軟體開發套件
- 設定軟體開發套件
- 在您的程式碼中使用軟體開發套件



## 2. 建立並執行使用 AWS SDK for Python (Boto3)的範例程式

此程式會顯示帳戶目前設定的日誌記錄選項。安裝軟體開發套件並為您的帳戶進行設定之後，您應該可以執行此程式。

```
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))
```

如需此範例中所使用函數的詳細資訊，請參閱 [the section called “設定 AWS IoT 記錄”](#)。

AWS SDK for Python (Boto3) 支援 AWS IoT Core 服務的說明文件

- [物聯網參考文件](#)
- [IoT DataPlane 參考文件](#)
- [IoT JobsDataPlane 參考文件](#)
- [IoT SecureTunneling 參考文件](#)

## Ruby

若要安裝 [AWS SDK for Ruby](#) 並使用它來連接至 AWS IoT：

- 按照 [入門中的說明進行操作 AWS SDK for Ruby](#)

這些指示說明如何：

- 安裝軟體開發套件
- 設定軟體開發套件
- 建立和執行 [Hello World 教學課程](#)

適用於 Ruby 的 AWS SDK 所支援之 AWS IoT Core 服務的說明文件

- [Aws::IoT::Client 參考文件](#)

- [Aws::IoT:DataPlane: 用戶端參考文件](#)
- [Aws::IoT:JobsDataPlane: 用戶端參考文件](#)
- [Aws::IoT:SecureTunneling: 用戶端參考文件](#)

## AWS 行動開發套件

行 AWS 行動開發套件為服務的 API、使用 MQTT 的 IoT 裝置通訊，以及其他 AWS IoT Core 服務的 API，提供行動應用程式開發人員平台特定的支援。AWS

### Android

#### AWS Mobile SDK for Android

包 AWS Mobile SDK for Android 含程式庫、範例和文件，供開發人員使用 AWS。此 SDK 還包括對 MQTT 設備通信和調用服務的 API 的 AWS IoT Core 支持。如需詳細資訊，請參閱下列內容：

- [AWS 安卓版移動 SDK GitHub](#)
- [AWS 安卓版移動 SDK 自述文件](#)
- [AWS 適用於安卓範例的行動 SDK](#)
- [AWS 適用於安卓 API 參考的 SDK](#)
- [AWSIoTClient 類參考文檔](#)

### iOS

#### AWS Mobile SDK for iOS

AWS Mobile SDK for iOS 這是一個開放原始碼軟體開發套件，在 Apache 開放原始碼授權下散佈。SDK for iOS 提供程式庫、程式碼範例和文件，協助開發人員使用 AWS。此 SDK 還包括對 MQTT 設備通信和調用服務的 API 的 AWS IoT Core 支持。如需詳細資訊，請參閱下列內容：

- [AWS Mobile SDK for iOS 上 GitHub](#)
- [AWS SDK for iOS 自述文件](#)
- [AWS SDK for iOS 範例](#)
- [AWS IoT 適用於 iOS 的 AWS SDK 中的類別參考文件](#)

## AWS IoT Core 服務的其餘 API

AWS IoT Core 服務的其餘 API 可以通過使用 HTTP 請求直接調用。

- 端點 URL

公開 AWS IoT Core 服務之 REST API 的服務端點會因區域而有所不同，並列於 [AWS IoT Core 端點和配額](#) 中。您必須針對具有要存取之 AWS IoT 資源的區域使用端點，因為 AWS IoT 資源是區域特定的。

- 身分驗證

AWS IoT Core 服務的其餘 API 使用 AWS IAM 登入資料進行身份驗證。如需詳細資訊，請參閱 AWS 一般參考中的 [簽署 AWS API 要求](#)。

- API 參考

如需服務之 REST API 所提供之特定功能的相關資 AWS IoT Core 訊，請參閱：

- [適用於 IoT 的 API 參考](#)。
- [適用於 IoT 資料的 API 參考](#)。
- [適用於 IoT 任務資料的 API 參考](#)。
- [適用於 IoT 安全通道的 API 參考](#)。

## 將裝置連接至 AWS IoT

設備通過連 AWS IoT 接到其他服務 AWS IoT Core。透過 AWS IoT Core，裝置會使用您帳戶專屬的裝置端點傳送和接收訊息。[the section called “AWS IoT 裝置開發套件”](#) 支援使用 MQTT 和 WSS 通訊協定的裝置通訊。如需裝置可使用之通訊協定的詳細資訊，請參閱 [the section called “裝置通訊協定”](#)。

### 訊息代理程式

AWS IoT 透過訊息代理程式管理裝置通訊。裝置和用戶端會將訊息發佈至訊息代理程式，並訂閱訊息代理程式發佈的訊息。訊息由應用程式定義的 [主題](#) 識別。當訊息代理程式收到裝置或用戶端發佈的訊息時，會將該訊息重新發佈至已訂閱該訊息之主題的裝置和用戶端。訊息代理程式也會將訊息轉寄至 AWS IoT [規則](#) 引擎，該引擎可對訊息的內容採取行動。

### AWS IoT 消息安全

要 AWS IoT 使用的設備連接 [the section called “X.509 用戶端憑證”](#) 和 [AWS 簽名 V4](#) 進行身份驗證。裝置通訊受 TLS 1.3 版保護，並 AWS IoT 要求裝置在連線時傳送 [伺服器名稱指示 \(SNI\) 延伸](#) 模組。如需詳細資訊，請參閱 [中的傳輸安全性 AWS IoT](#)。

## AWS IoT 裝置資料和服務端點

### ⚠ Important

您可以在裝置中快取或儲存端點。這意味著您不需要在每次連接新裝置時查詢 DescribeEndpoint API。為您的帳戶 AWS IoT Core 建立端點後，端點不會變更。

每個帳戶都有數個帳戶獨有的裝置端點，並支援特定的 IoT 功能。AWS IoT 裝置資料端點支援針對 IoT 裝置的通訊需求而設計的發佈/訂閱通訊協定；但是，如果其他用戶端 (例如應用程式和服務) 的應用程式需要這些端點所提供的特殊功能，也可以使用此介面。AWS IoT 裝置服務端點支援以裝置為中心的安全性和管理服務存取。

若要瞭解帳戶的裝置資料端點，您可以在 AWS IoT Core 主機的 [\[設定\]](#) 頁面中找到該端點。

若要了解您帳戶的裝置端點特定用途，包括裝置資料端點，請使用此處顯示的 describe-endpoint CLI 命令或 DescribeEndpoint REST API，並提供下表中的 *endpointType* 參數值。

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

這個命令會傳回下列格式的 *iot-endpoint : account-specific-prefix.iot.aws-region.amazonaws.com*。

每個客戶都有一個 iot:Data-ATS 和一個 iot:Data 端點。每個端點都使用 X.509 憑證來驗證用戶端。強烈建議客戶使用較新的 iot:Data-ATS 端點類型，以避免與普遍不信任 Symantec 憑證授權單位相關的問題。我們為設備提供 iot:Data 端點，以從使用 VeriSign 證書向後兼容的舊端點中檢索數據。如需詳細資訊，請參閱 [伺服器身分驗證](#)。

### AWS IoT 裝置的端點

端點用途	<i>endpointType</i> 值	描述
AWS IoT Core : 資料平面操作	iot:Data-ATS	用來將資料傳送至訊息代理程式、 <a href="#">Device Shadow</a> 和 AWS IoT 的 <a href="#">規則引擎</a> 元件，以及從中接收資料。

端點用途	<i>endpointType</i> 值	描述
		iot:Data-ATS 傳回 ATS 簽署的資料端點。
AWS IoT Core : 資料平面操作 (舊式)	iot:Data	iot:Data 返回提供向後兼容性的 VeriSign 簽名數據端點。Symantec (iot:Data) 端點不支援 MQTT 5。
AWS IoT Core 認證存取	iot:CredentialProvider	用於將裝置的內建 X.509 憑證交換為臨時憑證，以直接與其他 AWS 服務連線。如需有關連線至其他 AWS 服務的詳細資訊，請參閱 <a href="#">授權直接呼叫 AWS 服務</a> 。
AWS IoT Device Management : 任務資料操作	iot:Jobs	用於使裝置能夠使用 AWS IoT 工作 <a href="#">裝置 HTTPS API 與作業服務互動</a> 。
AWS IoT 裝置建議程式作	iot:DeviceAdvisor	用於使用 Device Advisor 測試裝置的測試端點類型。如需詳細資訊，請參閱 <a href="#">???</a> 。
AWS IoT Core 資料測試版 (預覽版)	iot:Data-Beta	保留給 測試版的端點類型。如需其目前使用方式的詳細資訊，請參閱 <a href="#">???</a> 。

您也可以使用您自己的完整網域名稱 (FQDN)，例如 *example.com*，以及相關的伺服器憑證，透過使用來連接裝置。AWS IoT [the section called “可設定端點”](#)

## AWS IoT 裝置開發套件

AWS IoT 裝置開發套件可協助您將 IoT 裝置連接到，AWS IoT Core 並透過 WSS 通訊協定支援 MQTT 和 MQTT。

AWS IoT 裝置 SDK 與 SDK 不同之處在於 AWS IoT 裝置 SDK 支援 IoT 裝置的特殊通訊需求，但不支援 SDK 支援的所有服務。AWS AWS AWS IoT 裝置 SDK 與支援所有 AWS 服務的 AWS SDK 相容；

但是，它們使用不同的驗證方法並連接到不同的端點，這可能會導致在 IoT 裝置上使用 AWS SDK 不切實際。

## 行動裝置

它 [the section called “AWS 行動開發套件”](#) 支持 MQTT 設備通信，一些 AWS IoT 服務 API 和其他 AWS 服務的 API。如果您是在支援的行動裝置上進行開發，請檢閱其軟體開發套件，以確認這是否為開發 IoT 解決方案的最佳選擇。

## C++

### AWS IoT C++ 設備開發套件

AWS IoT C++ 設備 SDK 允許開發人員使用 AWS 和 AWS IoT Core 服務的 API 構建連接的應用程序。此 SDK 特別是為並未受限於資源的裝置所設計，需要訊佇列、多重執行緒支援、最新語言功能等進階功能。如需詳細資訊，請參閱下列內容：

- [AWS IoT 開啟裝置開發套件 C++ V2 GitHub](#)
- [AWS IoT 設備 SDK C++ V2 自述文件](#)
- [AWS IoT 裝置開發套件 C++ V2 範例](#)
- [AWS IoT 設備開發套件 C++ v2 API 文檔](#)

## Python

### AWS IoT Python 的設備開發套件

適用於 Python 的 AWS IoT 設備 SDK 使開發人員可以編寫 Python 腳本，以使用他們的設備通過 MQTT 或 MQTT 通過 WebSocket 安全 ( WSS ) 協議訪問 AWS IoT 平台。透過將裝置連接到 AWS IoT Core 服務的 API，使用者可以安全地使用 AWS IoT Core 提供 Amazon Kinesis 和 Amazon S3 等 AWS Lambda 其他服務的訊息代理程式、規則和 Device Shadow AWS 服務，並與其他服務搭配使用。

- [AWS IoT 適用於 Python 的設備開發套件 GitHub](#)
- [AWS IoT 適用於 Python v2 自述文件的設備 SDK](#)
- [AWS IoT 適用於 Python V2 範例的裝置 SDK](#)
- [AWS IoT 適用於 Python 的裝置開發套件 API 文件](#)

## JavaScript

### AWS IoT 適用於的裝置 SDK JavaScript

設 AWS IoT 備 SDK JavaScript 使開發人員可以編寫訪問通過協議 AWS IoT Core 使用 MQTT 或 MQTT 的 API 的 JavaScript 應用程式。WebSocket 可用於 Node.js 環境和瀏覽器應用程式中。如需詳細資訊，請參閱下列內容：

- [AWS IoT 適用於 JavaScript v2 的裝置 SDK GitHub](#)
- [AWS IoT JavaScript v2 自述文件的設備 SDK](#)
- [AWS IoT 適用於 JavaScript v2 範例的裝置 SDK](#)
- [AWS IoT 適用於 JavaScript v2 API 文件的裝置 SDK](#)

## Java

### AWS IoT Java 的裝置開發套件

Java 的 AWS IoT 裝置 SDK 可讓 Java 開發人員透過通訊協定存取 AWS IoT Core 透過 MQTT 或 MQTT 的 API。WebSocket 軟體開發套件支援 Device Shadow 服務。您可以使用 HTTP 方法來存取影子，包括 GET、UPDATE 與 DELETE。此軟體開發套件亦支援簡化的影子存取模式，開發人員只需使用 getter 和 setter 方法，即可與影子交換資料，而無需將任何 JSON 文件序列化或還原序列化。如需詳細資訊，請參閱下列內容：

- [AWS IoT 適用於 Java v2 的裝置開發套件 GitHub](#)
- [AWS IoT 適用於 Java v2 自述文件的設備 SDK](#)
- [AWS IoT 適用於 Java V2 範例的裝置 SDK](#)
- [AWS IoT 適用於 Java v2 的裝置開發套件 API 文件](#)

## Embedded C

### AWS IoT 適用於嵌入式 C 的裝置 SDK

#### Important

此 SDK 適合經驗豐富的嵌入式軟體開發人員使用。

適用於 Embedded C 的 AWS IoT Device SDK (C-SDK) 是 MIT 開放原始碼授權下的 C 原始碼檔案集合，可用於嵌入式應用程式，將 IoT 裝置安全地連接到 AWS IoT Core。它包括 MQTT，JSON 解析器和 AWS IoT Device Shadow 庫等。它以原始碼形式分配，並且將會與應用程式碼、其他程式庫，以及 (可選的) RTOS (即時作業系統)，一起內建於客戶韌體中。

適用於 Embedded C 的 AWS IoT Device SDK 常會針對需要最佳化 C 語言執行階段的資源受限裝置。您可以在任何作業系統上使用軟體開發套件，並將其裝載在任何處理器類型 (例如 MCU 和 MPU) 上。如果您的裝置有足夠的記憶體和處理資源，建議您使用其他 AWS IoT 裝置和行動 SDK 之一，例如 C++ JavaScript、Java 或 Python 的 AWS IoT 裝置 SDK。

如需詳細資訊，請參閱下列內容：

- [AWS IoT 適用於嵌入式 C 的裝置 SDK GitHub](#)
- [AWS IoT 嵌入式 C 自述文件的設備 SDK](#)
- [AWS IoT 適用於嵌入式 C 範例的裝置 SDK](#)

## 裝置通訊協定

AWS IoT Core 支援使用 MQTT 和 MQTT 透過 WebSocket 安全 (WSS) 通訊協定發佈和訂閱訊息的裝置和用戶端，以及使用 HTTPS 通訊協定發佈訊息的裝置和用戶端。所有通訊協定皆支援 IPv4 和 IPv6。本節說明裝置和用戶端的不同連線選項。

### TLS 1.2 和 TLS 1.3

AWS IoT Core 使用 [TLS 1.2 版](#) 和 [TLS 1.3 版](#) 來加密所有通訊。將裝置連線到 AWS IoT Core 時，用戶端可以傳送 [伺服器名稱指示 \(SNI\) 擴充功能](#)，雖然這不是必要，但強烈建議您這樣做。您必須使用 SNI 擴充功能，才能使用如 [多帳戶註冊](#)、[自訂網域](#) 及 [VPC 端點](#) 等功能。如需詳細資訊，請參閱 [中的傳輸安全性 AWS IoT](#)。

[AWS IoT 裝置開發套件](#) 支援 MQTT 或是經 WSS 的 MQTT，並支援用戶端連線的安全性需求。建議您使用 [AWS IoT 裝置開發套件](#) 將用戶端連線至 AWS IoT。


### 通訊協定、連接埠映射和身分驗證

裝置或用戶端使用裝置端點連線至訊息代理程式的方式，取決於它所使用的通訊協定。下表列出 AWS IoT 裝置端點支援的通訊協定，以及它們使用的驗證方法和通訊埠。



## 通訊協定、身分驗證和連接埠對應

通訊協定	支援的操作	身分驗證	連線埠	ALPN 通訊協定名稱
MQTT 超過 WebSocket	發佈、訂閱	Signature 第 4 版	443	N/A
MQTT 超過 WebSocket	發佈、訂閱	自訂身分驗證	443	N/A
MQTT	發佈、訂閱	X.509 用戶端憑證	443 <sup>†</sup>	x-amzn-mqtt-ca
MQTT	發佈、訂閱	X.509 用戶端憑證	8883	N/A
MQTT	發佈、訂閱	自訂身分驗證	443 <sup>†</sup>	mqtt
HTTPS	僅發佈	Signature 第 4 版	443	N/A
HTTPS	僅發佈	X.509 用戶端憑證	443 <sup>†</sup>	x-amzn-http-ca
HTTPS	僅發佈	X.509 用戶端憑證	8443	N/A
HTTPS	僅發佈	自訂身分驗證	443	N/A

 應用程式層通訊協定交涉 (ALPN)

<sup>†</sup> 使用 X.509 用戶端憑證驗證連線至連接埠 443 的用戶端必須實作 [應用程式層通訊協定交涉 \(ALPN\)](#) TLS 延伸模組，並使用用戶端 ProtocolNameList 傳送的 [ALPN 通訊協定名稱](#) 做為訊息的一部分。ClientHello

[在連接埠 443 上，物聯網:資料 ATS 端點支援 ALPN x-amzn-http-ca HTTP，但物聯網:工作端點則不支援。](#)

在具有 ALPN 的連接埠 8443 HTTPS 和連接埠 443 MQTT 上 x-amzn-mqtt-ca，無法使用 [自訂驗證](#)。

用戶端會連線到其 AWS 帳戶裝置端點。如需有關如何尋找帳戶的裝置端點的資訊，請參閱 [the section called “AWS IoT 裝置資料和服務端點”](#)。

### Note

AWS SDK 不需要整個 URL。它們只需要端點主機名稱，例如 [適用於 Python 的 AWS IoT 設備 SDK 的 pubsub.py 示例 GitHub](#)。如下表所示，傳遞完整 URL 可能會產生錯誤，例如主機名稱無效。

## 連接到 AWS IoT Core

通訊協定	端點或 URL
MQTT	<i>iot-endpoint</i>
透過 WSS 的 MQTT	<i>wss://iot-endpoint /mqtt</i>
HTTPS	<i>https://iot-endpoint /topics</i>

## 為您的裝置通訊選擇通訊協定

對於大多數透過裝置端點進行的 IoT 裝置通訊，您必須透過 WSS 通訊協定使用 MQTT 或 MQTT；不過，裝置端點也支援 HTTPS。下表比較如何 AWS IoT Core 使用這兩種通訊協定進行裝置通訊。

### AWS IoT 設備協議 side-by-side

功能	<a href="#">MQTT</a>	<a href="#">HTTPS</a>
發佈/訂閱支援	發佈和訂閱	僅發佈
開發套件支援	<a href="#">AWS 裝置開發套件支援 MQTT 和 WSS 通訊協定</a>	不支援軟體開發套件，但您可以使用特定語言的方法來發出 HTTPS 要求
服務品質支援	<a href="#">MQTT QoS 層級 0 和 1</a>	QoS 的支援是透過傳遞查詢字串參數 <code>?qos=qos</code> 來進行，其中值可以是 0 或 1。您可以新

功能	<a href="#">MQTT</a>	<a href="#">HTTPS</a>
		增此查詢字串，以發佈內含您所需之 QoS 值的訊息。
可接收裝置離線時所遺漏的訊息	是	否
clientId 欄位支援	是	否
裝置中斷連線偵測	是	否
安全通訊	是。請參閱 <a href="#">通訊協定、連接埠映射和身分驗證</a>	是。請參閱 <a href="#">通訊協定、連接埠映射和身分驗證</a>
主題定義	應用程式定義	應用程式定義
訊息資料格式	應用程式定義	應用程式定義
通訊協定負荷	較低	較高
耗電量	較低	較高

## 連線持續時間限制

HTTPS 連線的持續時間不保證比接收和回應要求所需時間更長。

MQTT 連線持續時間取決於您使用的身分驗證功能。下表列出每項功能在理想條件下的連線持續時間上限。

依身分驗證功能分類的 MQTT 連線持續時間

功能	持續時間上限 <sup>*</sup>
X.509 用戶端憑證	1-2 週
自訂身分驗證	1-2 週
Signature 第 4 版	長達 24 小時

\* 不保證

使用 X.509 憑證和自訂身分驗證時，連線持續時間沒有硬性限制，但可以短至幾分鐘。有多種原因可能導致連線中斷。下方清單中包含一些最常見的原因。

- Wi-Fi 可用性中斷
- 網際網路服務供應商 (ISP) 連線中斷
- 服務修補程式
- 服務部署
- 服務自動擴展
- 服務主機無法使用
- 負載平衡器問題和更新
- 用戶端錯誤

您的裝置必須實作偵測中斷連線和重新連線的策略。如需中斷連線事件的詳細資訊，以及處理前述事件的指導，請參閱 [???](#) 中的 [???](#)。

## MQTT

[MQTT](#) (訊息佇列遙測傳輸) 是輕量級且廣泛採用的訊息通訊協定，專為受限的裝置所設計。AWS IoT Core 支援的 MQTT 以 [MQTT v3.1.1 規格](#) 和 [MQTT v5.0 規格](#) 為基礎，但如 [the section called “AWS IoT 與 MQTT 規格的差異”](#) 所述具有若干差異。作為該標準的最新版本，MQTT 5 推出數項關鍵功能，使基於 MQTT 的系統更加強大，包括新的可擴展性增強功能、具備原因代碼回應的改良版錯誤報告、訊息和工作階段過期計時器，以及自訂使用者訊息標頭。如需有關支援的 MQTT 5 功能的詳細資訊，請參閱 [MQTT 5 AWS IoT Core 支援的功能](#)。AWS IoT Core 還支持跨 MQTT 版本 (MQTT 3 和 MQTT 5) 通信。MQTT 3 發佈者可以將 MQTT 3 訊息傳送給可接收 MQTT 5 發佈訊息的 MQTT 5 訂閱者，反之亦然。

AWS IoT Core 支援透過 WSS 通訊協定使用 MQTT 通訊協定和 MQTT 的裝置連線，並以用戶端 ID 識別。[AWS IoT 裝置開發套件](#) 支援這兩種通訊協定，並且是將裝置連接至 AWS IoT Core 的建議方式。設 AWS IoT 備 SDK 支持設備和客戶端連接和訪問 AWS IoT 服務所必需的功能。裝置 SDK 支援 AWS IoT 服務所需的驗證通訊協定，以及 MQTT 通訊協定和 WSS 通訊協定所需的連線識別碼需求。如需如何 AWS IoT 使用 AWS 裝置 SDK 連線至的資訊，以及支援語言範例 AWS IoT 的連結，請參閱 [the section called “使用裝置 AWS IoT 開發套件與 MQTT 連線”](#)。如需 MQTT 訊息之身分驗證方法和連接埠映射的詳細資訊，請參閱 [通訊協定、連接埠映射和身分驗證](#)。

雖然我們建議使用 AWS IoT 裝置 SDK 連線到 AWS IoT，但不需要這些軟體開發套件。但是，如果您不使用 AWS IoT 裝置 SDK，則必須提供必要的連線和通訊安全性。用戶端必須在連線請求中傳送 [伺服器名稱指示 \(SNI\) TLS 延伸](#)。不包含 SNI 的連線嘗試將會被拒絕。如需詳細資訊，請參閱 [中的傳輸安全性 AWS IoT](#)。使用 IAM 使用者和 AWS 登入資料驗證用戶端的用戶端必須提供正確的 [簽名版本 4](#) 身份驗證。

在本主題中：

- [使用裝置 AWS IoT 開發套件與 MQTT 連線](#)
- [MQTT 服務品質 \(QoS\) 選項](#)
- [MQTT 持久性工作階段](#)
- [MQTT 保留訊息](#)
- [MQTT Last Will and Testament \(LWT\) 訊息](#)
- [使用 connectAttributes](#)
- [MQTT 5 支援的功能](#)
- [MQTT 5 屬性](#)
- [MQTT 原因代碼](#)
- [AWS IoT 與 MQTT 規格的差異](#)

## 使用裝置 AWS IoT 開發套件與 MQTT 連線

本節包含 AWS IoT 裝置 SDK 的連結，以及說明如何將裝置連接到的範例程式的原始程式碼。AWS IoT 此處鏈接的示例應用程序顯示如何通過 WSS AWS IoT 使用 MQTT 協議和 MQTT 進行連接。

### Note

AWS IoT 裝置開發套件已經發行了 MQTT 5 用戶端。

## C++

### 使用 AWS IoT C++ 裝置 SDK 連接裝置

- [以 C++ 撰寫的 MQTT 連線範例的應用程式原始碼](#)
- [AWS IoT C++ 設備開發套件 V2 GitHub](#)

## Python

使用適用於 Python 的 AWS IoT 設備 SDK 來連接設備

- [以 Python 撰寫的 MQTT 連線範例的應用程式原始碼](#)
- [AWS IoT 適用於 Python V2 的設備開發套件 GitHub](#)

## JavaScript

使用 AWS IoT 裝置 SDK JavaScript 來連線裝置

- [示例應用程序的源代碼，顯示 MQTT 連接示例 JavaScript](#)
- [AWS IoT 適用於 JavaScript v2 的裝置 SDK GitHub](#)

## Java

使用適用於 Java 的 AWS IoT 裝置 SDK 來連接裝置

### Note

適用於 Java v2 的 AWS IoT 裝置開發套件現在支援安卓系統開發。如需詳細資訊，請參閱[適用於 Android 的 AWS IoT 裝置 SDK](#)。

- [以 Java 撰寫的 MQTT 連線範例的應用程式原始碼](#)
- [AWS IoT 適用於 Java v2 的裝置開發套件 GitHub](#)

## Embedded C

使用嵌入式 C 的 AWS IoT 裝置 SDK 連接裝置

### Important

此 SDK 適合經驗豐富的嵌入式軟體開發人員使用。

- [以 Embedded C 撰寫的 MQTT 連線範例的應用程式原始碼](#)
- [AWS IoT 適用於嵌入式 C 的裝置 SDK GitHub](#)

## MQTT 服務品質 (QoS) 選項

AWS IoT 而 AWS IoT 裝置開發套件則支援 [MQTT 服務品質 \(QoS\)](#) 等級和 0 1 MQTT 協議定義了第三級 QoS 級別 2，但 AWS IoT 不支持它。只有 MQTT 通訊協定支援 QoS 功能。HTTPS 透過傳遞查詢字串參數 `?qos=qos` 來支援 QoS，其中值可以是 0 或 1。

此表格說明各個 QoS 層級如何影響訊息發佈至代理程式的方式，以及訊息代理程式發佈訊息的方式。

具有 QoS 層級...	訊息為...	說明
QoS 層級 0	傳送零次或更多次	此層級應用於透過可靠通訊連結傳送的訊息，或者可能會錯過沒有問題的訊息。
QoS 層級 1	至少傳送一次，然後重複，直到接收到 PUBACK 回應	在寄件者收到 PUBACK 回應以表示成功傳遞之後，才會將訊息視為完整的。

## MQTT 持久性工作階段

持久性工作階段會存放用戶端的訂閱和訊息，服務品質 (QoS) 為 1，而這些用戶端尚未確認此服務品質。該裝置重新連線至持久性工作階段時，工作階段會繼續執行，訂閱會復原，且在重新連線之前接收和儲存的未確認的訂閱訊息皆會傳送至用戶端。

存儲的消息的處理被記錄在 CloudWatch 和 CloudWatch 日誌。如需有關寫入 CloudWatch 和 CloudWatch 記錄的項目的資訊，請參閱[訊息代理程式指標](#)和[佇列日誌項目](#)。

## 建立持久性工作階段

在 MQTT 3 中，您可以傳送 CONNECT 訊息並將 `cleanSession` 旗標設定為 0，以建立 MQTT 持久性工作階段。如果傳送 CONNECT 訊息的用戶端沒有工作階段，則會建立新的持久性工作階段。如果用戶端的工作階段已經存在，用戶端會繼續現有的工作階段。若要建立全新工作階段，請傳送 CONNECT 訊息並將 `cleanSession` 旗標設定為 1，當用戶端中斷連線時，代理程式將不會儲存任何工作階段狀態。

在 MQTT 5 中，您可以設定 `Clean Start` 旗標和 `Session Expiry Interval` 來處理持久性會話。「全新啟動」可控制連線工作階段的開始和前一個工作階段的結束。當您設定 `Clean Start = 1` 時，系統會建立新的工作階段，任何既有的先前工作階段都會終止。當您設定 `Clean Start = 0` 時，連線工作階段會繼續先前的工作階段 (若有)。工作階段過期間隔控制了連線工作階段的結束。工作階段

過期間隔指定了工作階段在中斷連線後維持的時間 (以秒為單位的 4 位元組整數)。如果設定 `Session Expiry interval = 0`，系統會在中斷連線時立即終止工作階段。如果 CONNECT 訊息中未指定工作階段過期間隔，則預設值為 0。

### MQTT 5 全新啟動和工作階段過期

屬性值	描述
<code>Clean Start= 1</code>	建立一個新的工作階段，並終止先前的工作階段 (若有)。
<code>Clean Start= 0</code>	若有先前的工作階段，則恢復工作階段。
<code>Session Expiry Interval &gt; 0</code>	維持工作階段。
<code>Session Expiry interval= 0</code>	不維持工作階段。

在 MQTT 5 中，如果您設定 `Clean Start = 1` 和 `Session Expiry Interval > 0`，則等同於 MQTT 3 全新工作階段。如果您設定 `Clean Start = 0` 和 `Session Expiry Interval > 0`，則等同於 MQTT 3 持久性工作階段。

#### Note

不支援跨 MQTT 版本 (MQTT 3 和 MQTT 5) 持久性工作階段。MQTT 3 持久性工作階段無法恢復為 MQTT 5 工作階段，反之亦然。

### 持久性工作階段期間的操作

用戶端使用連線認可 (CONNACK) 訊息中的 `sessionPresent` 屬性，以判斷持久性工作階段是否存在。如果 `sessionPresent` 是 1，表示存在持久性工作階段，且用戶端的任何已儲存訊息會在用戶端收到 CONNACK 後傳遞至用戶端，如[重新連線至持久性工作階段後的訊息流量](#)所述。如果 `sessionPresent` 是 0，則用戶端不需要重新訂閱。但是，如果 `sessionPresent` 是 0，表示不存在持久性工作階段，而且用戶端必須重新訂閱其主題篩選條件。

在用戶端加入持久性工作階段之後，它可以繼續發佈訊息並訂閱主題篩選條件，各個操作上無需任何額外的旗標。



## 重新連線至持久性工作階段後的訊息流量

持久性工作階段代表用戶端與 MQTT 訊息代理程式之間的持續連線。當用戶端使用持久性工作階段連接至訊息代理程式時，訊息代理程式會儲存用戶端在連線期間所進行的所有訂閱。當用戶端中斷連線時，訊息代理程式會存放未認可的 QoS 1 訊息，以及發佈到用戶端所訂閱之主題的新 QoS 1 訊息。系統會根據帳戶限制儲存訊息。將捨棄超出上限的訊息將。如需持久性訊息限制的詳細資訊，請參閱 [AWS IoT Core 端點和配額](#)。當用戶端重新連線至持久性工作階段時，所有訂閱都會恢復，而且所有存放的訊息都會傳送至用戶端，其傳送速率為每秒最多 10 則訊息。在 MQTT 5 中，如果具有訊息過期間隔的傳出 QoS1 在用戶端離線時過期，則用戶端在連線恢復後將不會收到過期的訊息。

重新連線之後，存放的訊息會以每秒 10 個存放訊息為限的速率傳送至用戶端，以及任何目前的訊息流量，直到達到 [Publish requests per second per connection](#) 限制為止。由於已存放訊息的傳遞速率有限，如果工作階段有 10 個以上的存放訊息要在重新連線後傳遞，將需要數秒鐘才能傳遞所有存放訊息。

## 結束持久性工作階段

持久性工作階段可透過下列方式結束：

- 當持久性工作階段過期時間經過時。當訊息代理程式偵測到用戶端已中斷連線時 (包括用戶端中斷連線或連線逾時)，持久性工作階段過期計時器將會啟動。
- 用戶端傳送將 `cleanSession` 旗標設定為 1 的 CONNECT 訊息。

在 MQTT 3 中，持久性工作階段到期時間預設值是一小時，這適用於帳戶中的所有工作階段。

在 MQTT 5 中，您可以設定 CONNECT 和 DISCONNECT 封包上的每個工作階段的工作階段過期間隔。

對於 DISCONNECT 封包的工作階段過期間隔：

- 如果目前工作階段的過期間隔為 0，則無法將 DISCONNECT 封包上的工作階段過期間隔設定為大於 0。
- 如果目前工作階段的過期間隔大於 0，且您將 DISCONNECT 封包上的工作階段過期間隔設為 0，則工作階段將於 DISCONNECT 結束。
- 否則，DISCONNECT 上的工作階段過期間隔會更新目前工作階段的過期間隔。

**Note**

當工作階段結束時，等待傳送給用戶端的存放訊息會被捨棄；但即使無法傳送，仍會依照標準的簡訊費率計費。如需訊息定價的詳細資訊，請參閱 [AWS IoT Core 定價](#)。您可以設定過期時間間隔。

## 持久性工作階段過期後重新連線

如果用戶端在過期之前沒有重新連接至其持久性工作階段，工作階段將會結束並捨棄其存放的訊息。當用戶端在工作階段過期後重新連線，並將 `cleanSession` 旗標設為 0 時，服務會建立新的持久性工作階段。這個工作階段無法使用先前工作階段的任何訂閱或訊息，因為上一個工作階段過期時，這些訂閱或訊息將被捨棄。

## 持久性工作階段訊息費用

AWS 帳戶 當訊息代理程式傳送訊息給用戶端或離線持續工作階段時，訊息會向您收取費用。當具有持久性工作階段的離線裝置重新連線並繼續其工作階段時，存放的訊息會傳遞至裝置，並再次向您的帳戶收費。如需訊息定價的詳細資訊，請參閱 [AWS IoT Core 定價：簡訊](#)。

使用標準限制增加程序，可以增加預設的持久性工作階段一小時過期時間。請注意，增加工作階段到期時間可能會增加您的訊息費用，因為額外的時間可能允許離線裝置存放更多訊息，而且將會按照標準簡訊速率向您的帳戶收取這些額外的訊息費用。工作階段過期時間為近似值，工作階段最多可以比帳戶限制長 30 分鐘；不過，工作階段不會短於帳戶限制。如需工作階段限制的詳細資訊，請參閱 [AWS Service Quotas](#)。

## MQTT 保留訊息

AWS IoT Core 支援 MQTT 通訊協定中所述的「保留」旗標。當用戶端在其發佈的 MQTT 訊息上設定 `RETAIN` 旗標時，會 AWS IoT Core 儲存該訊息。然後可以將其傳送給新訂閱者，透過呼叫 [GetRetainedMessage](#) 操作對其進行檢索，並在 [AWS IoT 主控台](#) 內予以檢視。

## 使用 MQTT 保留訊息的範例

- 作為初始組態訊息

MQTT 保留訊息會在用戶端訂閱主題後傳送至用戶端。如果您希望所有訂閱主題的用戶端在訂閱後立即收到 MQTT 保留訊息，您可以發佈設定 `RETAIN` 旗標的組態訊息。只要發佈新的組態訊息，訂閱用戶端也會收到該組態的更新。

- 作為最後一個已知的狀態訊息

裝置可以在目前狀態訊息上設定 RETAIN 旗標，以便 AWS IoT Core 將其儲存起來。當應用程式連線或重新連線時，他們可以訂閱此主題，並在訂閱保留的訊息主題後立即取得上次回報的狀態。透過這種方式，可以避免等到來自裝置的下一個訊息才能查看目前狀態。

在本節中：

- [AWS IoT Core 中含有 MQTT 保留訊息的常見任務](#)
- [計費和保留訊息](#)
- [比較 MQTT 保留訊息和 MQTT 持久性工作階段](#)
- [MQTT 保留訊息和 AWS IoT 裝置陰影](#)

AWS IoT Core 中含有 MQTT 保留訊息的常見任務

AWS IoT Core 使用已設定「保留」旗標來儲存 MQTT 訊息。這些保留訊息會像一般 MQTT 訊息一樣，傳送給已訂閱主題的所有用戶端，而且這些訊息也會存放起來，以傳送給該主題的新訂閱者。

MQTT 保留訊息需要特定的政策動作，才能授權用戶端存取這些訊息。如需保留訊息政策的使用範例，請參閱 [保留訊息政策範例](#)。

本節說明與保留訊息有關的常見操作。

- 建立保留訊息

用戶端會決定當其發佈 MQTT 訊息時是否保留訊息。用戶端可以在使用 [裝置 SDK](#) 發佈訊息時設定 RETAIN 旗標。應用程式和服務可以在使用 [Publish 動作](#) 發佈 MQTT 訊息時設定 RETAIN 旗標。

每個主題名稱只會保留一則訊息。已設定 RETAIN 旗標並發佈至主題的新訊息，會取代先前傳送至主題的任何現有保留訊息。

請注意：您無法發佈至已設定 RETAIN 旗標的 [預留主題](#)。

- 訂閱保留訊息主題

用戶端會訂閱保留訊息主題，如同訂閱其他任何 MQTT 訊息主題一樣。透過訂閱保留訊息主題而接收的保留訊息已設定 RETAIN 旗標。

AWS IoT Core 當用戶端將具有 0 位元組訊息承載的保留訊息發佈至保留的訊息主題時，會從中刪除保留的訊息。訂閱保留訊息主題的用戶端也會收到 0 個位元組訊息。

訂閱內含保留訊息主題的萬用字元主題篩選條件，可讓用戶端接收向保留訊息主題發佈的後續訊息，但不會在訂閱時即傳遞保留訊息。

請注意：若要在訂閱時即收到保留訊息，訂閱要求中的主題篩選條件必須完全符合保留訊息主題。

訂閱保留訊息主題時接收的保留訊息已設定 RETAIN 旗標。訂閱用戶端在訂閱後接收的保留訊息未設定 RETAIN 旗標。

- 擷取保留訊息

當用戶端訂閱帶有保留訊息的主題時，系統就會自動將保留訊息傳遞給用戶端。若要讓用戶端在訂閱時即接收保留訊息，則必須訂閱保留訊息的確切主題名稱。訂閱內含保留訊息主題的萬用字元主題篩選條件，可讓用戶端接收向保留訊息主題發佈的後續訊息，但不會在訂閱時即傳遞保留訊息。

服務和應用程式可以透過呼叫 [ListRetainedMessages](#) 和 [GetRetainedMessage](#) 來列出和擷取保留訊息。

在未設定 RETAIN 旗標的情況下，無法防止用戶端將訊息發佈至保留訊息主題。這可能會導致非預期的結果，例如保留訊息與訂閱主題接收到的訊息不相符。

使用 MQTT 5 時，如果保留的訊息已設定訊息過期間隔，且保留的訊息已經過期，則該主題的新訂閱者在訂閱成功後將不會收到保留的訊息。

- 列出保留訊息主題

您可以透過呼叫 [ListRetainedMessages](#) 列出保留訊息，且可以在 [AWS IoT 主控台](#) 中檢視保留訊息。

- 取得保留訊息的詳細資訊

您可以透過呼叫 [GetRetainedMessage](#) 取得保留訊息的詳細資訊，且可以在 [AWS IoT 主控台](#) 中檢視這些資訊。

- 保留「Will」訊息

裝置連線時建立的 MQTT [Will 訊息](#)，可以藉由在 Connect Flag bits 欄位中設定 Will Retain 旗標來加以保留。

- 刪除保留訊息

裝置、應用程式和服務可以透過將已設定 RETAIN 旗標的訊息和空白 (0 個位元組) 訊息承載發佈至要刪除的保留訊息主題名稱，以此刪除保留訊息。這類郵件會從中刪除保留的郵件 AWS IoT Core，會傳送至訂閱主題的用戶端，但不會由保留 AWS IoT Core。

您也可以透過在 [AWS IoT 主控台](#) 中存取保留訊息，來刪除保留訊息。使用 [AWS IoT 主控台](#) 刪除的保留訊息，也會將 0 個位元組的訊息傳送給已訂閱保留訊息主題的用戶端。

保留訊息遭刪除後即無法還原。用戶端需要發佈新的保留訊息，以取代已刪除的訊息。

- 對保留訊息進行偵錯和疑難排解

此 [AWS IoT 主控台](#) 提供多種工具，可協助您針對保留訊息進行疑難排解：

- [Retained messages](#) (保留訊息) 頁面

AWS IoT 主控台中的 Retained messages (保留訊息) 頁面會提供分頁清單，列出由您帳戶在目前區域存放的保留訊息。在此頁面上，您可以：

- 查看每則保留訊息的詳細資訊，例如訊息承載、QoS，以及保留訊息的接收時間。
  - 更新保留訊息的內容。
  - 刪除保留訊息。
- [MQTT test client](#) (MQTT 測試用戶端)

AWS IoT 主控台中的 MQTT test client (MQTT 測試用戶端) 頁面可以訂閱和發佈至 MQTT 主題。發佈選項可讓您在發佈的訊息上設定 RETAIN 旗標，以模擬裝置可能出現的行為。

某些非預期的結果可能是中如何實作保留訊息的這些方面所造成的 AWS IoT Core。

- 保留訊息限制

當帳戶儲存保留的訊息數目上限時，會傳回限制 AWS IoT Core 回應以 RETAIN set 發佈的訊息，且承載大於 0 位元組為止，直到刪除某些保留訊息且保留的訊息計數低於限制為止。

- 保留訊息的傳遞順序

我們不能保證保留訊息和訂閱訊息的傳遞順序。

## 計費和保留訊息

如 [AWS IoT Core 定價：簡訊](#) 中所述，藉由使用 AWS IoT 主控台或呼叫 [Publish](#)，從用戶端發佈已設定 RETAIN 旗標的訊息會產生額外的簡訊費用。

用戶端、使用 AWS IoT 主控台或呼叫擷取保留的訊息，除了一般 API 使用費外，還會 [GetRetainedMessage](#) 產生訊息傳遞費用。[AWS IoT Core 定價：簡訊](#) 中說明了額外費用。

裝置意外中斷連線時發佈的 MQTT [Will 訊息](#) 會產生簡訊費用，如 [AWS IoT Core 定價：簡訊](#) 中所述。

如需簡訊成本的詳細資訊，請參閱 [AWS IoT Core 定價：簡訊](#)。

## 比較 MQTT 保留訊息和 MQTT 持久性工作階段

保留訊息和持久性工作階段是 MQTT 的標準功能，可讓裝置接收在離線時發佈的訊息。可以透過持久性工作階段發佈的保留訊息。本節說明這些功能的主要特色及其運作方式。

	保留訊息	持久性工作階段
主要功能	<p>保留訊息可用於在連線後對大型裝置群組進行設定或通知。</p> <p>若您希望裝置在重新連線後，僅接收發佈至主題之最後一則訊息時，即可使用保留訊息。</p>	<p>對於具有間歇性連線且可能會遺漏數則重要訊息的裝置，持久性工作階段能發揮重要作用。</p> <p>裝置可以與持久性工作階段連線，以便接收在離線時傳送的訊息。</p>
範例	<p>保留訊息可以在裝置上線時提供與其環境有關的組態資訊。初始組態可能包含其應訂閱的其他訊息主題清單，或是應該如何設定其本機時區的相關資訊。</p>	<p>透過行動網路連線且具有間歇性連線的裝置，可能會使用持久性工作階段，以避免在裝置超出網路覆蓋範圍或需要關閉行動無線電時遺失所傳送的重要訊息。</p>
在主題的初始訂閱時收到的訊息	<p>訂閱含有保留訊息的主題後，就會收到最近保留訊息。</p>	<p>若訂閱的主題沒有保留訊息，在有發佈至該主題的訊息前，不會收到任何訊息。</p>
重新連線後的訂閱主題	<p>如果沒有持久性工作階段，用戶端必須在重新連線後訂閱主題。</p>	<p>已訂閱的主題會在重新連線後還原。</p>
重新連線後收到的訊息	<p>訂閱含有保留訊息的主題後，就會收到最近保留訊息。</p>	<p>在裝置中斷連線時，以 QOS = 1 發佈且以 QOS = 1 訂閱的所有訊息都會在裝置重新連線後傳送。</p>

	保留訊息	持久性工作階段
資料/工作階段過期	在 MQTT 3 中的訊息不會過期。這些訊息在遭取代或刪除前都會存放起來。在 MQTT 5 中，保留的訊息會在您設定的訊息過期間隔後過期。如需詳細資訊，請參閱 <a href="#">訊息過期</a> 。	如果用戶端未在逾時期間內重新連線，則持久性工作階段會過期。持久性工作階段過期後，在裝置中斷連線時，用戶端的訂閱和以 QOS = 1 發佈並以 QOS = 1 訂閱的已儲存訊息都會遭到刪除。過期的訊息將不會傳遞。如需持久性工作階段之工作階段過期的詳細資訊，請參閱 <a href="#">the section called “MQTT 持久性工作階段”</a> 。

如需持久性工作階段的相關資訊，請參閱 [the section called “MQTT 持久性工作階段”](#)。

發佈用戶端會透過「保留訊息」判斷是否要在連線後保留訊息並將其傳遞至裝置，以及先前是否有工作階段。發佈者可選擇是否存放訊息，而所存放的訊息會傳遞至透過 QoS 0 或 QoS 1 訂閱進行訂閱的所有現有和未來用戶端。保留訊息一次只會保留一則指定主題的相關訊息。

帳戶存放的保留訊息數量達到上限時，在系統刪除部分保留訊息且保留訊息計數低於上限前，AWS IoT Core 會針對已設定 RETAIN 且承載大於 0 個位元組的發佈訊息傳回調節回應。

### MQTT 保留訊息和 AWS IoT 裝置陰影

保留訊息和 Device Shadow 都會保留裝置中的資料，但它們的行為不同，用途也不同。本節說明其異同。

	保留訊息	Device Shadow
訊息承載具有預先定義的結構或結構描述	如實作所定義。MQTT 不會為其訊息承載指定結構或結構描述。	AWS IoT 支援特定的資料結構。
更新訊息承載會產生事件訊息	發佈保留訊息會將訊息傳送至訂閱的用戶端，但不會產生其他更新訊息。	更新 Device Shadow 會產生 <a href="#">更新訊息，其中會描述變更</a> 。

	保留訊息	Device Shadow
已將訊息更新進行編號	不會自動將保留訊息進行編號。	Device Shadow 文件具有自動版本號碼和時間戳記。
訊息承載會連接至物件資源	保留訊息不會連接至物件資源。	Device Shadow 會連接至物件資源。
更新訊息承載的個別元素	如果不更新整個訊息承載，則無法變更訊息的個別元素。	您可以更新 Device Shadow 文件的個別元素，無需更新整個 Device Shadow 文件。
用戶端在訂閱時收到訊息資料	用戶端在訂閱含有保留訊息的主題後，會自動收到保留訊息。	用戶端可以訂閱 Device Shadow 更新，但必須特意請求目前狀態。
索引和可搜尋性	不會為保留訊息建立可供搜尋的索引。	機群索引製作會建立 Device Shadow 資料的索引，以供搜尋和彙總。

## MQTT Last Will and Testament (LWT) 訊息

Last Will and Testament (LWT) 是 MQTT 的功能。用戶端可使用 LWT 指定一則訊息，代理程式將在發生意外中斷連線的情況時，將該訊息發佈至用戶端定義的主題，並傳送至訂閱該主題的所有用戶端。用戶端指定的訊息稱為 LWT 訊息或 Will 訊息，而用戶端定義的主題稱為「Will 主題」。您可以在裝置連線至代理程式時指定 LWT 訊息。這些訊息可藉由在連線期間，於 Connect Flag bits 欄位中設定 Will Retain 旗標的方式加以保留。例如，如果 Will Retain 旗標設定為 1，Will 訊息將會儲存在代理程式中相關聯的 Will 主題中。如需詳細資訊，請參閱 [Will 訊息](#)。

代理程式會儲存 Will 訊息，直到發生意外中斷連線。發生這種情況時，代理程式會將訊息發佈到訂閱 Will 主題的所有用戶端，以通知發生中斷連線。如果用戶端使用 MQTT DISCONNECT 訊息透過用戶端起始的中斷連線從代理程式中斷連線，則代理程式將不會發佈儲存的 LWT 訊息。在所有其他情況下，都會送出 LWT 訊息。如需代理程式將傳送 LWT 訊息的完整中斷連線案例清單，請參閱 [連線/中斷連線事件](#)。



## 使用 connectAttributes

ConnectAttributes 可讓您在 IAM 政策中指定要在連線訊息中使用的屬性，例如 PersistentConnect 和 LastWill。透過 ConnectAttributes，您可以建置預設不允許裝置存取新功能的政策，如果裝置遭到入侵，這會很有幫助。

connectAttributes 支援下列功能：

### PersistentConnect

當用戶端和代理程式之間的連線中斷時，使用 PersistentConnect 功能可儲存用戶端在連線期間進行的所有訂閱。

### LastWill

當用戶端意外中斷連線時，使用 LastWill 功能可將訊息發佈至 LastWillTopic。

根據預設，您的政策具有非持久性連線，而且沒有傳遞此連線的屬性。如果您想要有持久性連線，必須在 IAM 政策中指定持久性連線。

如需 ConnectAttributes 範例，請參閱[連線政策範例](#)。

## MQTT 5 支援的功能

AWS IoT Core 對 MQTT 5 的支援是以 [MQTT v5.0 規格](#) 為基礎，但有一些差異，如中所述。[the section called “AWS IoT 與 MQTT 規格的差異”](#)

AWS IoT Core 支援以下 MQTT 5 功能：

- [共享訂閱](#)
- [全新啟動和工作階段過期](#)
- [所有 ACK 的原因代碼](#)
- [主題別名](#)
- [訊息過期](#)
- [其他 MQTT 5 功能](#)

## 共享訂閱

AWS IoT Core 支援 MQTT 3 和 MQTT 5 的共用訂閱。共享訂閱允許多個用戶端共享一個主題的訂閱，而且只有一個用戶端會使用隨機分佈接收發佈至該主題的訊息。共享訂閱可以在多個訂閱用戶之間

有效地負載平衡 MQTT 訊息。例如，假設您有 1,000 部裝置發佈至相同主題，而 10 個後端應用程式則處理這些訊息。於此種情況下，後端應用程式可訂閱相同的主題，且每個應用程式將隨機接收由裝置發佈至共享主題的訊息。這實際上是「共享」這些訊息的負載。共享訂閱還可提供更好的彈性。當任何後端應用程式中斷連線時，代理程式會將負載分配給群組中剩餘的訂閱用戶。

如要使用共享訂閱，用戶端訂閱共享訂閱的[主題篩選條件](#)，如下所示：

```
$share/{ShareName}/{TopicFilter}
```

- `$share` 是個常值字串，表示共享訂閱的主題篩選條件，必須以 `$share` 開頭。
- `{ShareName}` 是個字元字串，用來指定一組訂閱用戶使用的共享名稱。共享訂閱的主題篩選條件必須包含 `ShareName`，且後面接著 `/` 字元。`{ShareName}` 不得包含下列字元：`/`、`+`、或 `#`。`{ShareName}` 的大小上限為 128 個位元組。
- `{TopicFilter}` 會遵循與非共享訂閱相同的[主題篩選條件](#)語法。`{TopicFilter}` 的大小上限為 256 個位元組。
- `$share/{ShareName}/{TopicFilter}` 所需的兩個斜線 (`/`) 不包含於[主題和主題篩選條件中最大斜線數目](#)限制中。

具有相同 `{ShareName}/{TopicFilter}` 的訂閱屬於相同的共享訂閱群組。您可以建立多個共享訂閱群組，且不超過[每個群組的共享訂閱限制](#)。如需詳細資訊，請參閱《AWS 一般參考》中的[AWS IoT Core 端點和配額](#)。

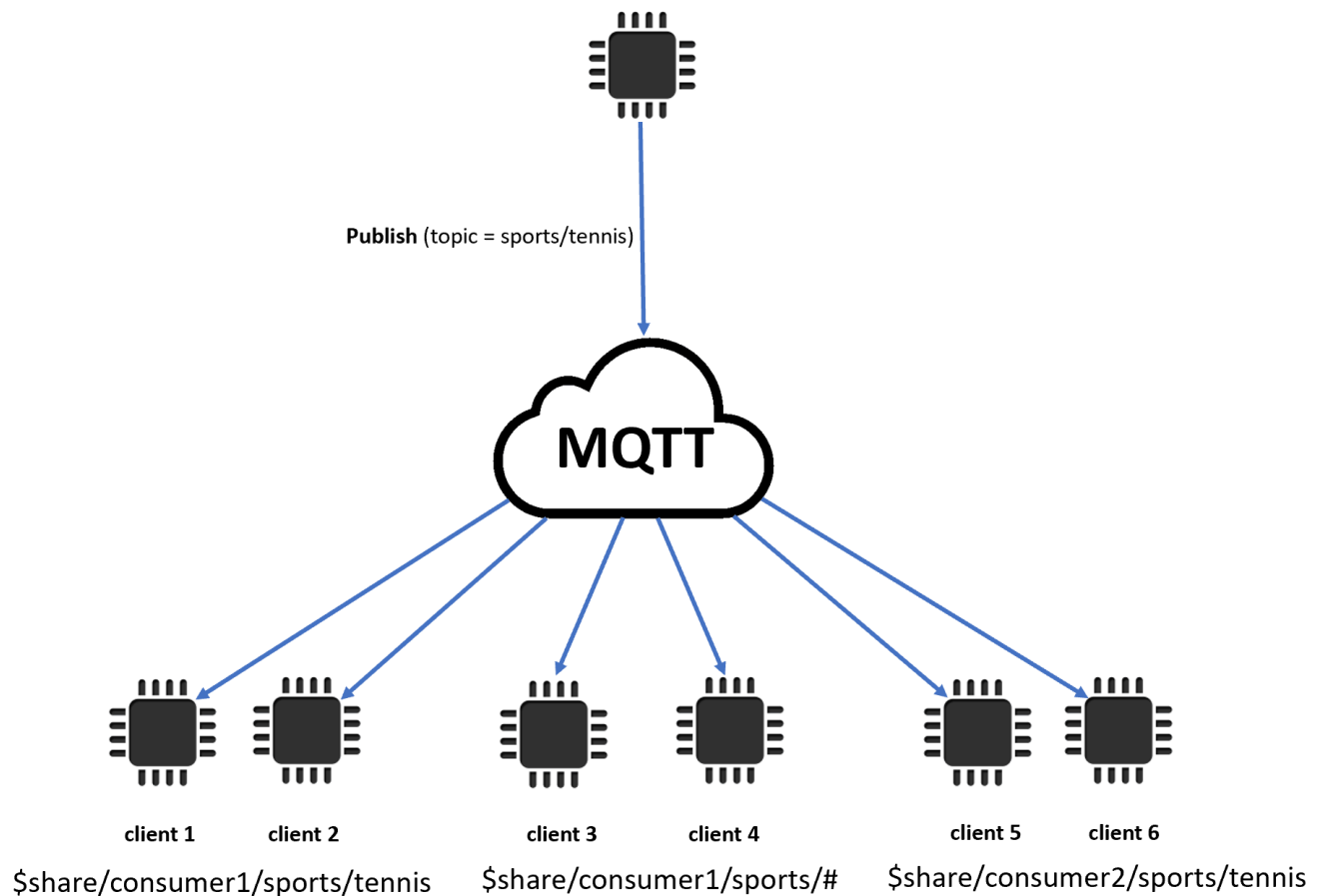
下表對非共享訂閱與共享訂閱進行了比較：

訂閱	描述	主題篩選條件範例
非共享訂閱	每個用戶端都會建立個別訂閱以接收已發佈的訊息。將訊息發佈至某個主題時，該主題的所有訂閱用戶都會收到該訊息的副本。	<pre>sports/tennis sports/#</pre>
共享訂閱	多個用戶端可共享一個主題的訂閱，且只有一個用戶端會使用隨機分佈接收發佈至該主題的訊息。	<pre>\$share/consumer/sports/tennis \$share/consumer/sports/#</pre>



### 使用共享訂閱的重要注意事項

- 當發佈至 QoS0 訂閱用戶的嘗試失敗時，不會發生重試嘗試，且會捨棄該訊息。
- 當對具有全新工作階段的 QoS1 訂閱用戶嘗試發佈失敗時，該訊息將會傳送至群組中的另一個訂閱用戶以進行多次重試。在所有重試嘗試之後無法傳遞的訊息將遭捨棄。
- 當對具有[持續性工作階段](#)的 QoS1 訂閱用戶嘗試發佈因為訂閱用戶離線而失敗時，該訊息將不會排入佇列，且會嘗試傳送至群組中的另一個訂閱用戶。在所有重試嘗試之後無法傳遞的訊息將遭捨棄。
- 共享訂閱不會收到[保留的訊息](#)。
- 當共享訂閱包含萬用字元 (# 或 +) 時，可能會有多個與主題相符的共享訂閱。若發生這種情況，訊息代理程式會複製發佈訊息，並將其傳送至每個相符之共享訂閱中的隨機用戶端。共享訂閱的萬用字元行為可說明於下圖中。



於此範例中，有三個與發佈 MQTT 主題 `sports/tennis` 相符的共享訂閱。訊息代理程式會複製已發佈的訊息，並將該訊息傳送至每個相符群組中的隨機用戶端。

用戶端 1 和用戶端 2 共享訂閱：`$share/consumer1/sports/tennis`

用戶端 3 和用戶端 4 共享訂閱：`$share/consumer1/sports/#`

用戶端 5 和用戶端 6 共享訂閱：`$share/consumer2/sports/tennis`

如需有關共享訂閱限制的詳細資訊，請參閱《AWS 一般參考》中的 [AWS IoT Core 端點和配額](#)。若要在 [AWS IoT 主控台](#) 中使用 AWS IoT MQTT 用戶端測試共用訂閱，請參閱 [???](#) 如需有關共享訂閱的更多資訊，請參閱 MQTTv5.0 規格中的 [共享訂閱](#)。

### 全新啟動和工作階段過期

您可以使用「全新啟動」和「工作階段過期」，以更具彈性的方式處理持久性工作階段。「全新啟動」旗標指出工作階段是否應在不使用現有工作階段的情況下啟動。「工作階段過期」間隔指出中斷

連線後保留工作階段的時間長度。可在中斷連線時修改工作階段過期間隔。如需詳細資訊，請參閱 [the section called “MQTT 持久性工作階段”](#)。

## 所有 ACK 的原因代碼

您可以使用原因代碼更輕鬆地偵錯或處理錯誤訊息。訊息代理程式會根據與代理程式的互動類型（「訂閱」、「發佈」、「確認」）傳回原因代碼。如需更多詳細資訊，請參閱 [MQTT 原因代碼](#)。如需 MQTT 原因代碼的完整清單，請參閱 [MQTT v5 規格](#)。

## 主題別名

主題別名是一種雙位元組整數，可用來取代主題名稱。使用主題別名可以最佳化主題名稱的傳輸，以降低計量資料服務的資料成本。AWS IoT Core 預設限制為 8 個主題別名。如需詳細資訊，請參閱《AWS 一般參考》中的 [AWS IoT Core 端點和配額](#)。

## 訊息過期

您可以對已發佈的訊息新增訊息過期值。這些值表示訊息過期間隔（以秒為單位）。如果訊息尚未在該間隔內傳送給訂閱者，則訊息將會過期並予以移除。如果未設定訊息過期值，訊息就不會過期。

在傳出流量中，訂閱者會收到一條訊息，告知過期間隔的剩餘時間。例如，如果傳入發佈訊息的訊息過期時間為 30 秒，且在 20 秒後路由給訂閱者，則訊息過期欄位會更新為 10。訂閱者收到的訊息可能具有值為 0 的更新 MEI。這是因為在剩餘時間達到 999 毫秒以下，它將更新為 0。

在中 AWS IoT Core，郵件到期間隔下限為 1。如果從用戶端將間隔設定為 0，則系統會將其調整為 1。訊息過期間隔上限為 604800（7 天）。任何較高的值都將調整為最大值。

在跨版本通訊中，訊息過期行為取決於傳入發佈訊息的 MQTT 版本。例如，若某個訊息的訊息過期時間是由透過 MQTT5 連線的工作階段所傳送，則其可能會在使用 MQTT3 工作階段訂閱的裝置上過期。下表列出訊息過期功能如何支援下列類型的發佈訊息：

發佈訊息類型	訊息過期間隔
常規發佈	如果伺服器無法在指定的時間內傳遞訊息，則過期的訊息將遭移除，而訂閱者將不會收到這些訊息。這包括裝置未發佈其 QoS 1 訊息等情況。
保留	如果保留的訊息到期，而新用戶端訂閱了該主題，則該用戶端將不會在訂閱時收到訊息。
遺言	遺言訊息的間隔會在用戶端中斷連線之後啟動，而伺服器會嘗試將遺言訊息傳遞給其訂閱者。

發佈訊息類型	訊息過期間隔
佇列訊息	如果具有訊息過期間隔的傳出 QoS1 在用戶端離線時過期，則用戶端在 <a href="#">持久性工作階段</a> 恢復後將不會收到過期的訊息。

## 其他 MQTT 5 功能

### 伺服器中斷連線

中斷連線時，伺服器可以主動向用戶端傳送 DISCONNECT 以通知連線關閉，並提供中斷連線的原因代碼。

### 請求/回應

發佈者可以請求接收者在接收時將回應傳送至發佈者指定的主題。

### 封包大小上限

用戶端和伺服器可以分別指定其支援的最大封包大小。

### 承載格式和內容類型

您可以在發佈訊息時指定承載格式 (二進位、文字) 和內容類型。這些會轉發給訊息接收者。

## MQTT 5 屬性

MQTT 5 屬性是 MQTT 標準的重要新增項目，旨在支援新的 MQTT 5 功能，例如工作階段過期和請求/回應模式。在中 AWS IoT Core，您可以建立可轉寄輸出郵件內容的[規則](#)，或使用[HTTP 發佈來發佈](#)具有某些新內容的 MQTT 訊息。

下表列出所有支援的 MQTT 5 內容。AWS IoT Core

屬性	描述	輸入類型	封包
承載格式指示器	此布林值可列舉字串值，用於表示承載是否已格式化為 UTF-8。	位元組	PUBLISH、CONNECT
內容類型	描述承載內容的 UTF-8 字串。	UTF-8 字串。	PUBLISH、CONNECT

屬性	描述	輸入類型	封包
回應主題	此 UTF-8 字串描述了要在請求-回應流程中作為接收者發佈目標的主題。主題不得包含萬用字元。	UTF-8 字串。	PUBLISH、CONNECT
相互關聯資料	請求訊息的傳送者使用的二進位資料，用以識別回應訊息所對應的請求。	二進位	PUBLISH、CONNECT
使用者屬性	一組 UTF-8 字串對。這個屬性可能在單一封包中出現多次。接收者將以與傳送索引鍵值對相同的順序來接收索引鍵。	UTF-8 字串對	CONNECT、PUBLISH、遺言屬性、SUBSCRIBE、DISCONNECT、UNSUBSCRIBE
訊息過期間隔時間	4 位元組整數，代表訊息過期間隔時間 (以秒為單位)。如果不存在，則訊息不會到期。	4 位元組整數	PUBLISH、CONNECT
工作階段過期間隔時間	4 位元組整數，代表工作階段到期間隔 (以秒為單位)。AWS IoT Core 最多支持 7 天，默認最多為一小時。如果您設置的值超過了帳戶的最大值，AWS IoT Core 將在 CONNACK 中返回調整後的值。	4 位元組整數	CONNECT、CONNACK、DISCONNECT
已指派的用戶端識別碼	裝置未指定用戶端 ID AWS IoT Core 時所產生的隨機用戶端 ID。隨機用戶端 ID 必須是新的用戶端識別碼，且目前由代理程式管理的任何其他工作階段都未予以使用。	UTF-8 字串。	CONNACK
伺服器保持連線	2 位元組整數，代表伺服器指派的保持連線時間。如果用戶端離線的時間超過保持連線時間，伺服器將中斷用戶端的連線。	2 位元組整數	CONNACK
請求問題資訊	此布林值指出發生失敗時是否傳送原因字串或使用者屬性。	位元組	CONNECT

屬性	描述	輸入類型	封包
接收上限	2 位元組整數，代表無需接收 PUBACK 即可傳送的 PUBLISH QOS > 0 封包數目上限。	2 位元組整數	CONNECT、CONNACK
主題別名上限	此值表示將接受為主題別名的最高值。預設值為 0。	2 位元組整數	CONNECT、CONNACK
QoS 上限	AWS IoT Core 支援的 QoS 的最大值。預設為 1。AWS IoT Core 不支援 QoS2。	位元組	CONNACK
保留可用	Boolean 值，指出 AWS IoT Core 訊息代理程式是否支援保留訊息。預設為 1。	位元組	CONNACK
封包大小上限	AWS IoT Core 接受和發送的最大數據包大小。不能超過 128 KB。	4 位元組整數	CONNECT、CONNACK
可用的萬用字元訂閱	Boolean 值，指出 AWS IoT Core 訊息代理程式是否支援可用萬用字元訂閱。預設為 1。	位元組	CONNACK
可用的訂閱識別碼	布林值，指出 AWS IoT Core 訊息代理程式是否支援可用的訂閱識別碼。預設值為 0。	位元組	CONNACK

## MQTT 原因代碼

MQTT 5 引入了改進的錯誤報告與原因代碼響應。AWS IoT Core 可以返回原因代碼，包括但不限於以下按數據包分組的代碼。如需 MQTT 5 支援原因代碼的完整清單，請參閱 [MQTT v5 規格](#)。

## CONNACK 原因代碼

Value	Hex	原因代碼名稱	描述
0	0x00	Success (成功)	已接受連線。
128	0x80	未指定的錯誤	伺服器不願揭露失敗原因，或者沒有其他適用的原因代碼。



Value	Hex	原因代碼名稱	描述
133	0x85	用戶端識別碼無效	用戶端識別碼是有效字串，但未獲伺服器允許。
134	0x86	使用者名稱或密碼錯誤	伺服器不接受用戶端指定的使用者名稱或密碼。
135	0x87	未獲授權	用戶端未獲得連線的授權。
144	0x90	主題名稱無效	遺言主題名稱已正確產生，但未獲伺服器接受。
151	0x97	超過配額	已超出實作或管理施加的限制。
155	0x9B	不支援 QoS。	伺服器不支援遺言 QoS 中的 QoS 集。

## PUBACK 原因代碼

Value	Hex	原因代碼名稱	描述
0	0x00	Success (成功)	已接受訊息。QoS 1 訊息的發佈作業繼續進行。
128	0x80	未指定的錯誤	接收者不接受發佈內容，但不願揭露原因，或不符其中一個其他值。
135	0x87	未獲授權	PUBLISH 未經授權。
144	0x90	主題名稱無效	主題名稱格式不正確，但未獲用戶端或伺服器接受。
145	0x91	使用中的封包識別碼	封包識別碼已在使用中。這可能表示用戶端與伺服器之間的工作階段狀態不相符。
151	0x97	超過配額	已超出實作或管理施加的限制。

## DISCONNECT 原因代碼

Value	Hex	原因代碼名稱	描述
129	0x81	格式錯誤的封包	接收到的封包不符合此規格。
130	0x82	協定錯誤	收到非預期或亂序的封包。
135	0x87	未獲授權	請求未獲得授權。
139	0x8B	伺服器關閉	伺服器正在關閉。
141	8D	保持連線逾時	連線已關閉，因為未接收任何封包的時間已達到保持連線時間的 1.5 倍。
142	0x8E	已接管工作階段	系統已連接到使用相同 ClientID 的另一個連線，導致此連線關閉。
143	0x8F	主題篩選條件無效	意願主題名稱格式正確，但未獲伺服器接受。
144	0x90	主題名稱無效	主題名稱格式正確，但未獲此用戶端或伺服器接受。
147	0x93	超過接收上限	用戶端或伺服器已收到超過其未傳送 PUBACK 或 PUBCOMP 的發佈項目接收上限。
148	0x94	主題別名無效	用戶端或伺服器已收到 PUBLISH 封包，其中包含的主題別名大於 CONNECT 或 CONNACK 封包中傳送的主題別名上限。
151	0x97	超過配額	已超出實作或管理施加的限制。
152	0x98	管理動作	由於系統執行管理動作，導致連線關閉。
155	0x9B	不支援 QoS。	用戶端指定的 QoS 大於 CONNACK 中 QoS 上限指定的 QoS。
161	0xA1	不支援訂閱識別碼	伺服器不支援訂閱識別碼；未接受訂閱。

## PUBACK 原因代碼

Value	Hex	原因代碼名稱	描述
0	0x00	授予 QoS 0	已接受訂閱，且傳送的 QoS 上限為 QoS 0。這可能是低於請求的 QoS。
1	0x01	授予 QoS 1	已接受訂閱，且傳送的 QoS 上限為 QoS 1。這可能是低於請求的 QoS。
128	0x80	未指定的錯誤	未接受訂閱，伺服器不願揭露原因，或者其他原因代碼均不適用。
135	0x87	未獲授權	用戶端未獲授權進行此訂閱。
143	0x8F	主題篩選條件無效	主題篩選器的格式正確，但不允許此用戶端使用。
145	0x91	使用中的封包識別碼	指定的封包識別碼已在使用中。
151	0x97	超過配額	已超出實作或管理施加的限制。

## UNSUBACK 原因代碼

Value	Hex	原因代碼名稱	描述
0	0x00	Success (成功)	訂閱已刪除。
128	0x80	未指定的錯誤	取消訂閱無法完成，伺服器不願揭露原因，或者其他原因代碼均不適用。
143	0x8F	主題篩選條件無效	主題篩選器的格式正確，但不允許此用戶端使用。
145	0x91	使用中的封包識別碼	指定的封包識別碼已在使用中。

## AWS IoT 與 MQTT 規格的差異

雖然訊息代理程式實作以 MQTT v3.1.1 規格與 [MQTT 5.0 規格](#) 為基礎，但有些部分與規格有異，內容如下：

- AWS IoT 不支援 MQTT 3 的下列封包：發生錯誤、公開和發佈。
- AWS IoT 不支持 MQTT 5 的以下數據包：發布，發布，發布版本和身份驗證。
- AWS IoT 不支援 MQTT 5 伺服器重新導向。
- AWS IoT 僅支援 MQTT 服務品質 (QoS) 等級 0 和 1 級。AWS IoT 不支援使用 QoS 層級 2 進行發佈或訂閱。請求 QoS 2 時，訊息代理程式不會傳送 PUBACK 或 SUBACK。
- 在中 AWS IoT，訂閱 QoS 層級為 0 的主題表示郵件會傳遞零次或多次。一則訊息可能會傳送超過一次。傳送超過一次的訊息在發送時可能會使用不同的封包 ID。在這些情況下，DUP 旗標就不會設置。
- 在回應連線請求時，訊息代理程式會傳送 CONNACK 訊息。此訊息包含了一個旗標，指出連線是否會恢復先前的工作階段。
- 在傳送額外的控制封包或中斷連線請求之前，用戶端必須等待其裝置從 AWS IoT 訊息代理程式接收 CONNACK 訊息。
- 當用戶端訂閱主題，訊息代理程式傳送 SUBACK 以及用戶端開始接收符合的新訊息這段期間可能會發生延遲。
- 當用戶端在主題篩選條件中使用萬用字元 # 以訂閱主題時，主題階層中位於及低於其層級的所有字串都會相符。不過，父系主題不相符。例如，訂閱主題 sensor/# 會收到向主題 sensor/、sensor/temperature、sensor/temperature/room1 發佈的訊息，但不會收到向 sensor 發佈的訊息。如需萬用字元的詳細使用資訊，請參閱 [主題篩選條件](#)。
- 訊息代理程式使用用戶端 ID 來識別每一個用戶端。用戶端 ID 作為 MQTT 承載的一部分從用戶端傳遞至訊息代理程式。具有相同用戶端 ID 的兩個用戶端，不能同時連線至訊息代理程式。當用戶端使用另一個用戶端正在使用的用戶端 ID 連線至訊息代理程式，系統會接受新的用戶端連線，而先前連線的用戶端會中斷連線。
- 在極少數情況下，訊息代理程式可能會以不同的封包 ID 重新傳送相同的邏輯 PUBLISH 訊息。
- 包含萬用字元的主題篩選條件訂閱無法接收保留訊息。若要接收保留訊息，訂閱請求必須包含與保留訊息主題完全相符的主題篩選條件。
- 訊息代理程式不保證接收訊息及 ACK 的順序。
- AWS IoT 可能有與規格不同的限制。如需詳細資訊，請參閱 AWS IoT 參考指南中的 [AWS IoT Core 訊息代理程式和通訊協定限制和配額](#)。
- 不支援 MQTT DUP 旗標。

## HTTPS

用戶端可以使用 HTTP 1.0 或 1.1 通訊協定向 REST API 發出請求來發佈訊息。如需 HTTP 請求所使用的身分驗證和連接埠對應的相關資訊，請參閱 [通訊協定、連接埠映射和身分驗證](#)。

### Note

HTTPS 不像 MQTT 一樣可支援 `clientId` 值。使用 MQTT 時可使用 `clientId`，但使用 HTTPS 時則無法使用。

## HTTPS 訊息 URL

裝置和用戶端會將 POST 請求發佈至用戶端特定端點和特定主題的 URL，以發佈其訊息：

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

- *IoT\_data\_endpoint* 是 [AWS IoT 裝置資料端點](#)。您可以使用以下 AWS CLI 指令，在物件的詳細資訊頁面或用戶端上尋找 AWS IoT 主控台當中的端點：

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

此端點看起來如下：`a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`

- *url\_encoded\_topic\_name* 是所傳送郵件的完整 [主題名稱](#)。

## HTTPS 訊息程式碼範例

以下是如何將 HTTPS 訊息傳送至 AWS IoT 的一些範例。

### 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 AWS 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 AWS 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 [curl](#) 從用戶端或裝置傳送訊息至 AWS IoT。

使用 curl 從用 AWS IoT 戶端裝置傳送訊息

1. 檢查 curl 版本。
  - a. 在您的用戶端上，請在命令提示中執行此命令。

```
curl --help
```

在說明文字中，尋找 TLS 選項。您應該看到 `--tlsv1.2` 選項。

- b. 如果您看到 `--tlsv1.2` 選項，請繼續。
  - c. 如果您沒有看到 `--tlsv1.2` 選項或您收到 `command not found` 錯誤訊息，您可能需要在用戶端上更新或安裝 curl 或安裝 openssl，然後才能繼續進行。
2. 在用戶端上安裝憑證。

複製您在 AWS IoT 主控台中註冊用戶端 (物件) 時建立的憑證檔案。在繼續之前，請確定您的用戶端上有這三個憑證檔案。

- 憑證授權機構憑證檔案 (此範例中的 *Amazon-root-CA-1.pem*)。
- 用戶端的憑證檔案 (此範例中的 *device.pem.crt*)。
- 用戶端的私密金鑰檔案 (此範例中的 *private.pem.key*)。

### 3. 建立 curl 命令列，取代您帳戶和系統的可取代值。

```
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*

驗證對等的憑證授權機構憑證檔案名稱和路徑 (如有必要)。

--cert *device.pem.crt*

用戶端的憑證檔案名稱和路徑 (如有必要)。

--key *private.pem.key*

用戶端的私密金鑰檔案名稱和路徑 (如有必要)。

--request POST

HTTP 請求的類型 (此處指 POST)。

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

您要發佈的 HTTP POST 資料。在這種情況下，這是一個 JSON 字串，其內部引號用反斜線字元 (\) 逸出。

"https://*IoT\_data\_endpoint*:8443/topics/*topic*?qos=1"

用戶端 AWS IoT 裝置資料端點的 URL，後面接著 HTTPS 連接埠:8443，然後接著關鍵字/topics/和主題名稱 (在這種情況下)。topic指定服務品質作為查詢參數?qos=1。

### 4. 在主控台中開啟 MQTT 測試用戶端。AWS IoT

遵循[使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)中的指示並設定主控台，以訂閱用於 curl 命令中且主題名稱為 *topic* 的訊息，或使用 # 的萬用字元主題篩選條件。

### 5. 測試命令。



監控 AWS IoT 主控台測試用戶端中的主題時，請前往您的用戶端並發出您在步驟 3 中建立的 curl 命令列。您應該會在主控台中看到用戶端的訊息。

## MQTT 主題

MQTT 主題可識別 AWS IoT 訊息。AWS IoT 客戶端通過提供消息主題名稱來識別他們發布的消息。用戶端能透過用 AWS IoT Core 註冊主題篩選條件，來識別想要訂閱 (接收) 的簡訊。訊息代理程式使用主題名稱和主題篩選條件，將訊息從發佈用戶端路由至訂閱用戶端。

訊息代理程式會使用主題來識別使用 MQTT 傳送和使用 HTTP 傳送至 [HTTPS 訊息 URL](#) 的訊息

雖然 AWS IoT 支援某些[保留的系統主題](#)，但大多數 MQTT 主題都是由您 (系統設計師) 建立和管理。AWS IoT 使用主題來識別從發佈用戶端接收的郵件，並選取要傳送至訂閱用戶端的郵件，如下列各節所述。為您的系統建立主題命名空間之前，請檢閱 MQTT 主題的特性，以建立最適合您 IoT 系統的主題名稱階層。

### 主題名稱

主題名稱和主題篩選條件均為 UTF-8 編碼的字串。它們可以使用正斜線 (/) 字元來表示資訊層級，以分離層次的層級。例如，本主題名稱會提及房間 1 中的溫度感應器：

- sensor/temperature/room1

在此範例中，其他房間中可能也有其他類型的感應器，例如主題名為：

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

#### Note

考慮系統中的訊息主題名稱時，請謹記：

- 主題名稱和主題篩選條件會區分大小寫。
- 主題名稱不得包含個人識別資訊。
- 以 \$ 開頭的主題名為僅由 AWS IoT Core 使用的[預留主題](#)。

- AWS IoT Core 無法在 AWS 帳戶 s 或區域之間傳送或接收訊息。

如需有關設計主題名稱和命名空間的詳細資訊，請參閱我們的白皮書，[針對 AWS IoT Core 設計 MQTT 主題](#)。

如需應用程式如何發佈和訂閱訊息的範例，請從 [開始使用 AWS IoT Core](#) 和 [AWS IoT 裝置 SDK、行動 SDK 和 AWS IoT 裝置用戶端](#) 開始。

### Important

主題名稱空間僅限於 AWS 帳戶 和區域。例如，在一個區域 AWS 帳戶 中使用的sensor/temp/room1主題與另一個區域中同一帳戶使用的sensor/temp/room1主題不同，或由任何區域中的任何其他 AWS 帳戶使用 AWS 帳戶 的主題不同。

## ARN 主題

所有主題 ARN (Amazon Resource Names) 都具有以下形式：

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

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

## 主題篩選條件

訂閱用戶端會使用訊息代理程式登錄主題篩選條件，以指定訊息代理程式應該傳送給他們的訊息主題。主題篩選條件可以是單一的訂閱主題名稱，也可以包含萬用字元，以便一次訂閱多個主題名稱。

發佈用戶端無法在其發佈的主題名稱中使用萬用字元。

以下表格列出可用於主題篩選條件的萬用字元。

### 主題萬用字元

萬用字元	相符	備註
#	位於主題階層中及低於其層級的所有字串。	必須是主題篩選條件中的最後一個字元。

萬用字元	相符	備註
		<p>必須是其主題階層的層級中唯一的字元。</p> <p>可以在也包含 + 萬用字元的主題篩選條件中使用。</p>
+	包含該字元的層級中的任何字串。	<p>必須是其主題階層的層級中唯一的字元。</p> <p>可以在主題篩選條件的多個層級中使用。</p>

對上一個感應器主題名稱範例使用萬用字元：

- 訂閱 `sensor/#` 會收到發佈至 `sensor/`、`sensor/temperature`、`sensor/temperature/room1` 的訊息，但不會收到發佈至 `sensor` 的訊息。
- 訂閱 `sensor+/room1` 會收到發佈至 `sensor/temperature/room1` 和 `sensor/humidity/room1` 的訊息，但不會收到 `sensor/temperature/room2` 和 `sensor/humidity/room2` 的訊息。

## 主題篩選條件 ARN

所有主題篩選條件 ARN (Amazon Resource Names) 都具有以下形式：

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

例如，`arn:aws:iot:us-west-2:123EXAMPLE456:topicfilter/application/topic/+sensor` 是主題篩選條件 `application/topic/+sensor` 的 ARN。

## MQTT 訊息承載

在 MQTT 訊息中傳送的訊息承載不會由指定 AWS IoT，除非是其中一個。[the section called “預留主題”](#) 為了滿足您的應用程式的需求，我們建議您在[通訊協定的 AWS IoT Core 服務配額](#) 限制範圍內定義主題的訊息承載。

針對訊息承載使用 JSON 格式，可讓 AWS IoT 規則引擎剖析訊息並對其套用 SQL 查詢。如果您的應用程式不需要規則引擎將 SQL 查詢套用至訊息承載，您可以使用應用程式需要的任何資料格式。如需 JSON 文件 (使用於 SQL 查詢) 的限制和預留字元的相關資訊，請參閱 [JSON Extensions](#)。

如需設計 MQTT 主題及其對應訊息承載的詳細資訊，請參閱 [針對 AWS IoT Core 設計 MQTT 主題](#)。

如果訊息大小限制超過了服務配額，則會導致 CLIENT\_ERROR，原因為 PAYLOAD\_LIMIT\_EXCEEDED，並顯示 Message payload exceeds size limit for message type (訊息承載超過訊息類型的大小限制)。如需訊息大小限制的詳細資訊，請參閱 [AWS IoT Core 訊息代理程式限制和配額](#)。

## 預留主題

以美元符號 (\$) 開頭的主題會保留供使用 AWS IoT。在允許的情況下，您可以訂閱並發佈至這些預留主題；然而您無法建立以貨幣符號開頭的新主題。不支援的發佈或訂閱預留主題作業可能會導致連線終止。

### 資產模型主題

主題	允許的用戶端操作	描述
<code>\$ aws/####/####// #/assetId ##/##/## ID assetModelId</code>	訂閱	AWS IoT SiteWise 將資產性質通知發佈至此主題。如需詳細資訊，請參閱《AWS IoT SiteWise 使用指南》中的「 <a href="#">與其他 AWS 服務互動</a> 」。

### AWS IoT Device Defender 主題

這些訊息支援簡潔二進位物件表示法 (CBOR) 格式和 JavaScript 物件標記法 (JSON) 的回應緩衝區，視主題的有####而定。AWS IoT Device Defender 主題僅支援 MQTT 發佈。

####	回應格式資料類型
cbor	Concise Binary Object Representation (CBOR)
json	JavaScript 对象标记法

如需詳細資訊，請參閱[從裝置傳送量度](#)。

主題	允許操作	描述
<code>\$aws/things/<i>thingName</i> / defender/metrics/<i>payload-format</i></code>	發佈	AWS IoT Device Defender 代理程式會將量度發佈至此主題。如需詳細資訊，請參閱 <a href="#">從裝置傳送量度</a> 。
<code>\$aws/things/<i>thingName</i> / defender/metrics/<i>payload-format</i> /accepted</code>	訂閱	AWS IoT # <i>AWS IoT Device Defender</i> ##### <i>\$aws/##/ThingthingName /###/##/ #####</i> 如需詳細資訊，請參閱 <a href="#">從裝置傳送量度</a> 。
<code>\$aws/things/<i>thingName</i> / defender/metrics/<i>payload-format</i> /rejected</code>	訂閱	AWS IoT # <i>AWS IoT Device Defender</i> ##### <i>\$aws/##/ThingthingName /###/##/ # #####</i> 如需詳細資訊，請參閱 <a href="#">從裝置傳送量度</a> 。

## AWS IoT Core 裝置位置主題

AWS IoT Core 裝置位置可以解析裝置中的測量資料，並提供 IoT 裝置的估計位置。來自設備的測量數據可以包括 GNSS，Wi-Fi，蜂窩和 IP 地址。AWS IoT Core 然後，「裝置位置」選擇可提供最佳精度的量測類型，並解析裝置位置資訊。如需詳細資訊，請參閱 [AWS IoT Core 裝置位置](#) 及 [使用 AWS IoT Core 裝置位置 MQTT 主題解析裝置位置](#)。

主題	允許操作	描述
<code>\$aws/device_location/<i>customer_device_id</i> /get_position_estimate</code>	發佈	裝置會發佈至此主題，以取得由裝置位 AWS IoT Core 置解析的掃描原始量測資料。
<code>\$aws/device_location/<i>customer_</i></code>	訂閱	AWS IoT Core 裝置位置成功解析裝置位置後，會發佈至此主題。

主題	允許操作	描述
<i>device_id</i> /get_position_estimate/accepted		
\$aws/device_location/ <i>customer_device_id</i> /get_position_estimate/rejected	訂閱	AWS IoT Core 由於 4xx 錯誤而無法成功解析裝置位置時，裝置位置會發佈至此主題。

## 事件主題

### Note

如需 LoRa WAN 事件的保留 MQTT 主題的詳細資訊，請參閱[連線狀態](#)事件。

主題	允許的用戶端操作	描述
\$aws/events/證書/已登記/ <i>caCertificateId</i>	訂閱	AWS IoT 當 AWS IoT 自動註冊憑證，以及當用戶端顯示具有 PENDING_ACTIVATION 狀態的憑證時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">the section called “設定用戶端的首次連線進行自動註冊”</a> 。
\$aws/events/job/ <i>jobID</i> /canceled	訂閱	AWS IoT 取消工作時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/job/ <i>jobID</i> /cancellation_in_progress	訂閱	AWS IoT 當工作取消進行時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/job/ <i>jobID</i> /completed	訂閱	AWS IoT 當工作完成時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/job/ <i>jobID</i> /deleted	訂閱	AWS IoT 刪除工作時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。

主題	允許的用戶端操作	描述
\$aws/events/job/ <i>jobID</i> /deletion_in_progress	訂閱	AWS IoT 當工作刪除正在進行時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobID</i> /canceled	訂閱	AWS IoT 取消工作執行時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobID</i> /deleted	訂閱	AWS IoT 刪除工作執行時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobID</i> /failed	訂閱	AWS IoT 當工作執行失敗時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobID</i> /rejected	訂閱	AWS IoT 拒絕工作執行時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobID</i> /removed	訂閱	AWS IoT 移除工作執行時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobID</i> /succeeded	訂閱	AWS IoT 當工作執行成功時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobID</i> /timed_out	訂閱	AWS IoT 當工作執行逾時時，會發佈此訊息。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/presence/connected/ <i>clientId</i>	訂閱	AWS IoT 當具有指定用戶端 ID 的 MQTT 用戶端連線至時，會發佈至此主題。AWS IoT 如需詳細資訊，請參閱 <a href="#">連線/中斷連線事件</a> 。
\$aws/events/presence/disconnected/ <i>clientId</i>	訂閱	AWS IoT 當具有指定用戶端 ID 的 MQTT 用戶端中斷連線時，會發佈至此主題。AWS IoT 如需詳細資訊，請參閱 <a href="#">連線/中斷連線事件</a> 。

主題	允許的用戶端操作	描述
\$aws/events/subscriptions/subscribed/ <i>clientId</i>	訂閱	AWS IoT 當具有指定用戶端識別碼的 MQTT 用戶端訂閱 MQTT 主題時，會發佈至本主題。如需詳細資訊，請參閱 <a href="#">訂閱/取消訂閱事件</a> 。
\$aws/events/subscriptions/unsubscribed/ <i>clientId</i>	訂閱	AWS IoT 當具有指定用戶端 ID 的 MQTT 用戶端取消訂閱 MQTT 主題時，會發佈至本主題。如需詳細資訊，請參閱 <a href="#">訂閱/取消訂閱事件</a> 。
\$aws/events/thing/ <i>thingName</i> /created	訂閱	AWS IoT 建立「 <i>thingName</i> 」物件時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
\$aws/events/thing/ <i>thingName</i> /updated	訂閱	AWS IoT 當更新「 <i>thingName</i> 」物件時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
\$aws/events/thing/ <i>thingName</i> /deleted	訂閱	AWS IoT 刪除「 <i>thingName</i> 」物件時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
\$aws/events/ <i>thingGroup</i> <i>pName</i> 物件群組//已建立的	訂閱	AWS IoT 建立物群組 <i>thingGroup</i> <i>pName</i> 時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
\$aws/events/ <i>thingGroup</i> <i>pName</i> 物件群組//已更新的	訂閱	AWS IoT 更新物件群組 <i>thingGroup</i> <i>pName</i> 時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
\$aws/events/ <i>thingGroup</i> <i>pName</i> 物件群組//已刪除的	訂閱	AWS IoT 刪除物件群組 <i>thingGroup</i> <i>pName</i> 時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。



主題	允許的用戶端操作	描述
\$aws/events/ 類型/ /已建立 <i>thingTypeName</i> 的	訂閱	AWS IoT 建立 <i>thingTypeName</i> 物件類型時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
\$aws/events/ 類型/ /已更新 <i>thingTypeName</i> 新	訂閱	AWS IoT 當 <i>thingTypeName</i> 物件類型更新時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
\$aws/events/ 類型/ /已刪除 <i>thingTypeName</i> 的	訂閱	AWS IoT 刪除 <i>thingTypeName</i> 物件類型時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
<i>\$aws/events/ thingName/ thingTypeAssociation thingTypeName</i>	訂閱	AWS IoT 當 ## <i>ThingName</i> 與物類型相關聯或與物類型取消關聯時，會發佈至此主題。 <i>thingTypeName</i> 如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
<i>\$aws/events/ ##/ #/ ##/## thingName /thingGroupMembers hip### thingGroupName</i>	訂閱	AWS IoT 將 ## <i>ThingName</i> 新增至物件群組 <i>thingGroupName</i> 時，會發佈至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
<i>\$aws/events/ ##/ #/ ##/## thingName /thingGroupMembers hip### thingGroupName</i>	訂閱	AWS IoT 從物件群 <i>thingGroupName</i> 組中移除 ## <i>ThingName</i> 時，會發行至此主題。如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。
<i>\$aws/events/ ## /##/##/##thingGroupHierarchy/# parentThingGroup # childThingGroup childThingGroup</i>	訂閱	AWS IoT ##### <i>childThingGroup## #####parentThingGroup</i> 如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。

主題	允許的用戶端操作	描述
<code>\$aws/events/ ## #/#/#/#thingGro upHierarchy/parent ThingGroup### childThingGroup childThingGroup</code>	訂閱	AWS IoT ##### <code>childThingGroup##</code> ##### <code>parentThingGroup</code> 如需詳細資訊，請參閱 <a href="#">the section called “登錄檔事件”</a> 。

## 機群佈建主題

### Note

此表格中標示為「接收」的用戶端作業會指出直接 AWS IoT 發佈至要求該主題的用戶端的主題，無論用戶端是否已訂閱主題。即使用戶端尚未訂閱回應訊息，也應該預期會收到這些訊息。這些回應訊息不會透過訊息代理程式傳遞，而且其他用戶端或規則無法訂閱這些訊息。

這些訊息支援簡潔二進位物件表示法 (CBOR) 格式和 JavaScript 物件標記法 (JSON) 的回應緩衝區，視主題的有 ##### 而定。

#####	回應格式資料類型
cbor	Concise Binary Object Representation (CBOR)
json	JavaScript 对象标记法

如需詳細資訊，請參閱 [裝置佈建 MQTT API](#)。

主題	允許的用戶端操作	描述
<code>\$aws/certificates/ create/payload-format</code>	發佈	發佈到這個主題，以從憑證簽署要求 (CSR) 建立憑證。

主題	允許的用戶端操作	描述
<code>\$aws/certificates/create/payload-format / accepted</code>	訂閱、接收	AWS IoT ##### \$ aws/##/##/ ##### #####
<code>\$aws/certificates/create/payload-format / rejected</code>	訂閱、接收	AWS IoT ## \$aws/ ##/##/##### #####
<code>\$aws/##/create-from-csr#####</code>	發佈	發佈至本主題，以從 CSR 建立憑證。
<code>\$aws/##/#####/create-from-csr##</code>	訂閱、接收	AWS IoT ##### \$ aws/##/c reate-from-csr#####
<code>\$aws/##/#####/## create-from-csr</code>	訂閱、接收	AWS IoT ##### \$ aws/##/c reate-from-csr#####
<code>\$aws/provisioning-templates/templateName /provision/payload-format</code>	發佈	發佈至此主題以註冊實物。
<code>\$aws/provisioning-templates/templateName /provision/payload-format /accepted</code>	訂閱、接收	AWS IoT ##### \$aws/#####/ ## templateName #/##/##### ###
<code>\$aws/provisioning-templates/templateName /provision/payload-format /rejected</code>	訂閱、接收	AWS IoT ## \$ aws/#####/ #####/##/ #####

## 任務主題

**Note**

此表格中標示為「接收」的用戶端作業會指出直接 AWS IoT 發佈至要求該主題的用戶端的主題，無論用戶端是否已訂閱主題。即使用戶端尚未訂閱回應訊息，也應該預期會收到這些訊息。

這些回應訊息不會透過訊息代理程式傳遞，而且其他用戶端或規則無法訂閱這些訊息。若要訂閱任務活動相關訊息，請使用 `notify` 和 `notify-next` 主題。

為機群監控解決方案訂閱任務和 `jobExecution` 事件主題時，您必須先啟用[任務和任務執行事件](#)來接收雲端上的任何事件。

如需詳細資訊，請參閱[任務裝置 MQTT API 操作](#)。

主題	允許的用戶端操作	描述
<code>\$aws/things/<i>thingName</i> / jobs/get</code>	發佈	裝置發佈訊息到這個主題來提出 <code>GetPendingJobExecutions</code> 請求。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
<code>\$aws/things/<i>thingName</i> / jobs/get/accepted</code>	訂閱、接收	裝置訂閱此主題，以接收來自 <code>GetPendingJobExecutions</code> 請求的成功回應。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
<code>\$aws/things/<i>thingName</i> / jobs/get/rejected</code>	訂閱、接收	當 <code>GetPendingJobExecutions</code> 請求被拒時，裝置會訂閱此主題以接收回應。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
<code>\$aws/things/<i>thingName</i> / jobs/start-next</code>	發佈	裝置發佈訊息到這個主題來提出 <code>StartNextPendingJobExecution</code> 請求。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
<code>\$aws/things/<i>thingName</i> / jobs/start-next/accepted</code>	訂閱、接收	裝置訂閱此主題，以接收送往 <code>StartNextPendingJobExecution</code>

主題	允許的用戶端操作	描述
		n 請求的成功回應。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
\$aws/things/ <i>thingName</i> / jobs/start-next/rejected	訂閱、接收	當 StartNextPendingJobExecution 請求被拒時，裝置會訂閱此主題以接收回應。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
\$aws/things/ <i>thingName</i> / jobs/ <i>jobId</i> /get	發佈	裝置發佈訊息到這個主題來提出 DescribeJobExecution 請求。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
\$aws/things/ <i>thingName</i> / jobs/ <i>jobId</i> /get/accepted	訂閱、接收	裝置訂閱此主題，以接收送往 DescribeJobExecution 請求的成功回應。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
\$aws/things/ <i>thingName</i> / jobs/ <i>jobId</i> /get/rejected	訂閱、接收	當 DescribeJobExecution 請求被拒時，裝置會訂閱此主題以接收回應。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。
\$aws/things/ <i>thingName</i> / jobs/ <i>jobId</i> /update	發佈	裝置發佈訊息到這個主題來提出 UpdateJobExecution 請求。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a> 。

主題	允許的用戶端操作	描述
\$aws/things/ <i>thingName</i> /jobs/ <i>jobId</i> /update/accepted	訂閱、接收	<p>裝置訂閱此主題，以接收送往 UpdateJobExecution 請求的成功回應。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a>。</p> <div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p><b>注意</b></p> <p>只有發佈到 \$aws/things/<i>thingName</i> /jobs/<i>jobId</i>/update 的設備才會收到此主題的訊息。</p> </div>
\$aws/things/ <i>thingName</i> /jobs/ <i>jobId</i> /update/rejected	訂閱、接收	<p>當 UpdateJobExecution 請求被拒時，裝置會訂閱此主題以接收回應。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a>。</p> <div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p><b>注意</b></p> <p>只有發佈到 \$aws/things/<i>thingName</i> /jobs/<i>jobId</i>/update 的設備才會收到此主題的訊息。</p> </div>
\$aws/things/ <i>thingName</i> /jobs/notify	訂閱、接收	<p>裝置訂閱此主題，以在物件的待處理執行清單新增或移除任務執行時接收通知。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a>。</p>
\$aws/things/ <i>thingName</i> /jobs/notify-next	訂閱、接收	<p>裝置訂閱此主題，以在物件的下一個待處理任務執行變更時接收通知。如需詳細資訊，請參閱 <a href="#">任務裝置 MQTT API 操作</a>。</p>

主題	允許的用戶端操作	描述
\$aws/events/job/ <i>jobId</i> /completed	訂閱	任務完成時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/job/ <i>jobId</i> /canceled	訂閱	任務取消時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/job/ <i>jobId</i> /deleted	訂閱	任務刪除時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/job/ <i>jobId</i> /cancellation_in_progress	訂閱	開始取消任務時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/job/ <i>jobId</i> /deletion_in_progress	訂閱	開始刪除任務時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobId</i> /succeeded	訂閱	任務執行成功時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobId</i> /failed	訂閱	任務執行失敗時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobId</i> /rejected	訂閱	任務執行遭拒時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$aws/events/jobExecution/ <i>jobId</i> /canceled	訂閱	任務執行取消時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。

主題	允許的用戶端操作	描述
\$saws/events/jobExecution/ <i>jobId</i> /timed_out	訂閱	任務執行逾時時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$saws/events/jobExecution/ <i>jobId</i> /removed	訂閱	任務執行移除時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。
\$saws/events/jobExecution/ <i>jobId</i> /deleted	訂閱	任務執行刪除時，任務服務會發佈這個主題的事件。如需詳細資訊，請參閱 <a href="#">任務事件</a> 。

### 規則主題

主題	允許的用戶端操作	描述
\$saws/rules/ <i>ruleName</i>	發佈	裝置或應用程式會發佈至此主題，以直接觸發規則。如需詳細資訊，請參閱 <a href="#">使用基本擷取減少簡訊費用</a> 。

### 安全通道主題

主題	允許的用戶端操作	描述
\$saws/things/ <i>thing-name</i> / tunnels/notify	訂閱	AWS IoT 針對 IoT 代理程式發佈此訊息，以便在遠端裝置上啟動本機 Proxy。如需詳細資訊，請參閱 <a href="#">the section called "IoT Agent Snippet"</a> 。

### 影子主題

已命名的影子和未命名的影子會使用本節中的主題。各影子所使用的主題只有在主題字首中有所不同。此表格會顯示每種影子類型所使用的主題字首。



<i>ShadowTopicPrefix</i> 值	影子類型
<code>\$aws/things/<i>thingName</i> /shadow</code>	未命名 (經典) 影子
<code>\$aws/things/<i>thingName</i> /shadow/name/<i>shadowName</i></code>	已命名影子

欲建立完整主題，請 *ShadowTopicPrefix* 針對您要參照的陰影類型選取，將 *thingName* 取代為適用的 shadow *Name*，並使用其對應值將其附加至主題虛設常式，如下表所示。請記住，主題會區分大小寫。

主題	允許的用戶端操作	描述
<i>ShadowTopicPrefix</i> /刪除	發佈/訂閱	裝置或應用程式會發佈至此主題，以刪除影子。如需詳細資訊，請參閱 <a href="#">/delete</a> 。
<i>ShadowTopicPrefix</i> /刪除/接受	訂閱	當影子刪除時，Device Shadow 服務會傳送訊息至此主題。如需詳細資訊，請參閱 <a href="#">/delete/accepted</a> 。
<i>ShadowTopicPrefix</i> /刪除/拒絕	訂閱	拒絕刪除影子的要求時，Device Shadow 服務會傳送訊息至此主題。如需詳細資訊，請參閱 <a href="#">/delete/rejected</a> 。
<i>ShadowTopicPrefix</i> /得到	發佈/訂閱	應用程式或物件會發佈空白訊息至此主題，以取得影子。如需詳細資訊，請參閱 <a href="#">Device Shadow MQTT 主題</a> 。
<i>ShadowTopicPrefix</i> /獲得/接受	訂閱	當要求影子成功時，Device Shadow 服務會傳送訊息至此主題。如需詳細資訊，請參閱 <a href="#">/get/accepted</a> 。
<i>ShadowTopicPrefix</i> /獲得/拒絕	訂閱	當要求影子被拒絕時，Device Shadow 服務會傳送訊息至此主題。如需詳細資訊，請參閱 <a href="#">/get/rejected</a> 。

主題	允許的用戶端操作	描述
<i>ShadowTopicPrefix</i> /更新	發佈/訂閱	物件或應用程式會發佈至此主題，以更新影子。如需詳細資訊，請參閱 <a href="#">/update</a> 。
<i>ShadowTopicPrefix</i> /更新/已接受	訂閱	當更新影子成功時，Device Shadow 服務會傳送訊息至此主題。如需詳細資訊，請參閱 <a href="#">/update/accepted</a> 。
<i>ShadowTopicPrefix</i> /更新/拒絕	訂閱	當更新影子被拒絕時，Device Shadow 服務會傳送訊息至此主題。如需詳細資訊，請參閱 <a href="#">/update/rejected</a> 。
<i>ShadowTopicPrefix</i> /更新/三角	訂閱	當偵測到回報的區段與所需的區段之間發生差異時，Device Shadow 服務會傳送訊息至此主題。如需詳細資訊，請參閱 <a href="#">/update/delta</a> 。
<i>ShadowTopicPrefix</i> /更新/文件	訂閱	AWS IoT 每當成功執行陰影更新時，都會將狀態文件發佈至此主題。如需詳細資訊，請參閱 <a href="#">/update/documents</a> 。

## MQTT 型檔案交付主題

### Note

此表格中標示為「接收」的用戶端作業會指出直接 AWS IoT 發佈至要求該主題的用戶端的主題，無論用戶端是否已訂閱主題。即使用戶端尚未訂閱回應訊息，也應該預期會收到這些訊息。這些回應訊息不會透過訊息代理程式傳遞，而且其他用戶端或規則無法訂閱這些訊息。

這些訊息支援簡潔二進位物件表示法 (CBOR) 格式和 JavaScript 物件標記法 (JSON) 的回應緩衝區，視主題的有#####而定。

#####	回應格式資料類型
cbor	Concise Binary Object Representation (CBOR)

####	回應格式資料類型
json	JavaScript 对象标记法

主題	允許的用戶端操作	描述
<code>\$aws/###/#/ # #/ThingName ##### StreamId</code>	訂閱、接收	AWS 如果接受來自裝置的 "GetStream" 要求，以 MQTT 為基礎的檔案傳遞會發佈至本主題。承載包含串流資料。如需詳細資訊，請參閱 <a href="#">在裝置中使用以 AWS IoT MQTT 為基礎的檔案傳遞</a> 。
<code>\$ aws/###/#/ /##/ ##### ThingName # StreamId</code>	發佈	裝置會發佈至此主題以執行 "GetStream" 要求。如需詳細資訊，請參閱 <a href="#">在裝置中使用以 AWS IoT MQTT 為基礎的檔案傳遞</a> 。
<code>\$aws/#/ /#/ /##/ #####ThingName ## StreamId</code>	訂閱、接收	AWS 如果接受來自裝置的 "DescribeStream" 要求，以 MQTT 為基礎的檔案傳遞會發佈至本主題。承載包含串流描述。如需詳細資訊，請參閱 <a href="#">在裝置中使用以 AWS IoT MQTT 為基礎的檔案傳遞</a> 。
<code>\$aws/#/ /#/ /##/ #####ThingName ## StreamId</code>	發佈	裝置會發佈至此主題以執行 "DescribeStream" 要求。如需詳細資訊，請參閱 <a href="#">在裝置中使用以 AWS IoT MQTT 為基礎的檔案傳遞</a> 。
<code>\$aws/#/ /#/ /##/ ##### ThingName StreamId</code>	訂閱、接收	AWS 如果來自裝置的 "" 或 "DescribeStreamGetStream" 要求遭到拒絕，MQTT 型檔案傳遞會發佈至此主題。如需詳細資訊，請參閱 <a href="#">在裝置中使用以 AWS IoT MQTT 為基礎的檔案傳遞</a> 。

## 預留的主題 ARN

所有預留的主題 ARN (Amazon Resource Names) 都具有以下形式：

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

例如，`arn:aws:iot:us-west-2:123EXAMPLE456:topic/$aws/things/thingName/jobs/get/accepted` 是預留主題 `$aws/things/thingName/jobs/get/accepted` 的 ARN。

## 可設定端點

在中 AWS IoT Core，您可以使用網域設定來設定和管理資料端點的行為。透過網域設定，您可以產生多個 AWS IoT Core 資料端點、使用您自己的完整網域名稱 (FQDN) 和關聯的伺服器憑證自訂這些資料端點，以及建立自訂授權者的關聯。如需詳細資訊，請參閱 [自訂身分驗證和授權](#)。

### Note

在中不提供此功能 GovCloud AWS 區域。

## 網域組態使用案例

您可以使用網域組態來簡化如下任務。

- 將裝置移轉至 AWS IoT Core.
- 維護個別裝置類型的不同網域組態，以支援異質裝置機群。
- 將應用程式基礎結構移轉至 AWS IoT Core.

## 在中使用網域組態的重要注意事項 AWS IoT Core

AWS IoT Core 使用 [伺服器名稱指示 \(SNI\) TLS 延伸](#) 來套用網域組態。設備連接時必須使用此擴展程序 AWS IoT Core。它們也必須傳遞與您在網域組態中指定之網域名稱相同的伺服器名稱。若要測試此服務，請在中 GitHub 使用 v2 版本的 [AWS IoT 裝置 SDK](#)。

如果您在中建立多個資料端點 AWS 帳戶，它們將共用 MQTT 主題、裝置陰影和規則等 AWS IoT Core 資源。

當您為 AWS IoT Core 自訂網域組態提供伺服器憑證時，憑證最多有四個網域名稱。如需詳細資訊，請參閱 [AWS IoT Core 端點和配額](#)。

在本章中：

- [建立和設定 AWS 受管理的網域](#)
- [建立和設定自訂網域](#)
- [管理網域組態](#)
- [在網域組態中設定 TLS 設定](#)
- [OCSP 裝訂的伺服器憑證組態](#)

## 建立和設定 AWS 受管理的網域

您可以使用 [CreateDomainConfiguration](#) API 在 AWS 受管理的網域上建立可設定的端點。AWS 受管理網域的網域組態包含下列項目：

- `domainConfigurationName`

識別網域組態和值的使用者定義名稱必須是您的唯一名稱 AWS 區域。您無法使用以 `IoT:` 開頭的網域組態名稱，因為它們是保留給預設端點。

- `defaultAuthorizerName` (選用)

要在端點上使用的自訂授權方名稱。

- `allowAuthorizerOverride` (選用)

布林值，用來指定裝置是否可以在請求的 HTTP 標頭中指定不同的授權方來覆寫預設授權方。如果指定了 `defaultAuthorizerName` 的值，則需要此值。

- `serviceType` (選用)

端點提供的服務類型。AWS IoT Core 僅支援服務類型。當您指定 `DATA` 時，AWS IoT Core 會傳回端點類型為 `iot:Data-ATS` 的端點。您無法建立可設定 `iot:Data (VeriSign)` 端點。

- `TlsConfig` (選用)

為網域指定 TLS 組態的物件。如需詳細資訊，請參閱 [???](#)。

下列範例 AWS CLI 命令會建立 Data 端點的網域組態。

```
aws iot create-domain-configuration --domain-configuration-name
  "myDomainConfigurationName" --service-type "DATA"
```

命令的輸出可能如下所示。

```
{
  "domainConfigurationName": "myDomainConfigurationName",
  "domainConfigurationArn": "arn:aws:iot:us-east-1:123456789012:domainconfiguration/myDomainConfigurationName/itihw"
}
```

## 建立和設定自訂網域

網域組態可讓您指定自訂完整網域名稱 (FQDN) 來連線至 AWS IoT Core。使用自訂網域有許多好處：您可以為客戶公開您自己的網域或公司自己的網域；您可以輕鬆地將自己的網域變更為指向新的代理商；您可以支援多租戶服務，以便在相同範圍內為不同網域的客戶提供服務 AWS 帳戶；您可以管理自己的伺服器憑證詳細資料，例如用於簽署憑證的根憑證授權單位 (CA)、憑證鏈結和簽名的演算法、簽名鍵、生命週期和簽名的演算法證書。

使用自訂網域來設定網域組態的工作流程包含下列三個階段。

1. [在中註冊伺服器憑證 AWS Certificate Manager](#)
2. [建立網域組態](#)
3. [建立 DNS 記錄](#)

### 在憑證管理員註冊伺服器憑 AWS 證

使用自訂網域建立網域組態之前，您必須在 [AWS Certificate Manager \(ACM\)](#) 中註冊伺服器憑證鏈。您可以使用下列三種類型的伺服器憑證。

- [ACM 產生的公有憑證](#)
- [由公有 CA 簽署的外部憑證](#)
- [由私有 CA 簽署的外部憑證](#)

#### Note

AWS IoT Core 如果憑證包含在 [Mozilla 受信任的 CA 組合包](#)中，則會考慮由公共 CA 簽署的憑證。

## 憑證需求

如需將憑證匯入至 ACM 的需求，請參閱[匯入憑證的先決條件](#)。除了這些需求之外，AWS IoT Core 會新增下列需求。

- 分葉憑證必須包含具有伺服器驗證 (TLS 網頁伺服器驗證) 值的延伸金鑰使用方式 x509 v3 延伸模組。如果您從 ACM 請求憑證，則會自動新增此延伸。
- 憑證鏈深度上限為 5 個憑證。
- 憑證鏈大小上限為 16KB。
- 支援的密碼編譯演算法和金鑰大小包括 RSA 2048 位元 (RSA) 和 256 位元。

## 針對多個網域使用一個憑證

如果您打算使用一個憑證來涵蓋多個子網域，請在通用名稱 (CN) 或主體替代名稱 (SAN) 欄位中使用萬用字元網域。例如，使用 `*.iot.example.com` 來涵蓋 `dev.iot.example.com`、`qa.iot.example.com` 和 `prod.iot.example.com`。每個 FQDN 都需要自己的網域組態，但多個網域組態可以使用相同的萬用字元值。CN 或 SAN 必須涵蓋您要用作自訂網域的 FQDN。如果 SAN 存在，CN 會遭忽略，且 SAN 必須涵蓋您要用作自訂網域的 FQDN。此涵蓋範圍可以是完全相符或萬用字元相符。驗證萬用字元憑證並向帳戶註冊後，該區域中的其他帳戶將無法建立與憑證重疊的自訂網域。

下列各節說明如何取得每種類型的憑證。每個憑證資源都需要向 ACM 註冊的 Amazon 資源名稱 (ARN)，您在建立網域組態時會使用此 ARN。

## ACM 產生的公有憑證

您可以使用 [RequestCertificate](#) API 為自訂網域產生公用憑證。當您以這種方式產生憑證時，ACM 會驗證您對自訂網域的擁有權。如需詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的[請求公有憑證](#)。

## 由公有 CA 簽署的外部憑證

如果您已經擁有的伺服器憑證是由公用 CA (包含在 Mozilla 受信任的 ca 組合中的 CA) 所簽署的伺服器憑證，您可以使用 API 將憑證鏈直接匯入 ACM。[ImportCertificate](#) 若要進一步了解此任務以及必要條件和憑證格式需求，請參閱[匯入憑證](#)。

## 由私有 CA 簽署的外部憑證

如果您已經擁有由私有 CA 簽署或自我簽署的伺服器憑證，則可以使用憑證來建立您的網域組態，但也必須在 ACM 中建立額外的公有憑證，以驗證網域的擁有權。若要這麼做，請使用 [ImportCertificate](#) API

在 ACM 中註冊伺服器憑證鏈結。若要進一步了解此任務以及必要條件和憑證格式需求，請參閱[匯入憑證](#)。

## 建立驗證憑證

將憑證匯入 ACM 之後，請使用 [RequestCertificate](#) API 為自訂網域產生公用憑證。當您以這種方式產生憑證時，ACM 會驗證您對自訂網域的擁有權。如需詳細資訊，請參閱[請求公有憑證](#)。建立網域組態時，請使用此公有憑證作為您的驗證憑證。

## 建立網域組態

您可以使用 [CreateDomainConfiguration](#) API 在自訂網域上建立可設定的端點。自訂網域的網域組態包含下列項目：

- `domainConfigurationName`

識別網域組態的使用者定義名稱。以 IoT: 開頭的網域組態名稱會保留給預設端點，且無法使用。此外，此值必須是您的唯一值 AWS 區域。

- `domainName`

您的裝置用來連線的 FQDN。AWS IoT Core 利用伺服器名稱指示 (SNI) TLS 延伸來套用網域組態。裝置在連線並傳遞與網域組態中指定之網域名稱相同的伺服器名稱時，必須使用此延伸。

- `serverCertificateArns`

您在 ACM 註冊之伺服器憑證鏈結的 ARN。AWS IoT Core 目前僅支援一個伺服器憑證。

- `validationCertificateArn`

您在 ACM 中產生之公有憑證的 ARN，此憑證用驗證自訂網域的擁有權。如果您使用公開簽署或 ACM 產生的伺服器憑證，則不需要此引數。

- `defaultAuthorizerName` (optional)

要在端點上使用的自訂授權方名稱。

- `allowAuthorizerOverride`

布林值，用來指定裝置是否可以在請求的 HTTP 標頭中指定不同的授權方來覆寫預設授權方。如果指定了 `defaultAuthorizerName` 的值，則需要此值。

- `serviceType`



AWS IoT Core 目前僅支援服務類型。當您指定服務類型，會 AWS IoT 傳回端點類型為的端點 `iot:Data-ATS`。

- `TlsConfig` (選用)


為網域指定 TLS 組態的物件。如需詳細資訊，請參閱 [???](#)。

- `serverCertificateConfig` (選用)

指定網域之伺服器憑證組態的物件。如需詳細資訊，請參閱 [???](#)。

下列 AWS CLI 指令會為 `iot.example.com` 建立網域組態。

```
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

建立網域組態後，最多可能需要 60 分鐘才能 AWS IoT Core 提供您的自訂伺服器憑證。

如需詳細資訊，請參閱 [???](#)。

## 建立 DNS 記錄

在註冊您的伺服器憑證並建立您的網域組態之後，請建立 DNS 記錄，讓您的自訂網域可以指向 AWS IoT 網域。此記錄必須指向類型的 AWS IoT 端點 `iot:Data-ATS`。您可以使用 [DescribeEndpoint](#) API 取得端點。

以下 AWS CLI 命令顯示如何獲取端點。

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

取得 `iot:Data-ATS` 端點後，請建立從自訂網域到此 AWS IoT 端點的 CNAME 記錄。如果您在同一個中建立多個自訂網域 AWS 帳戶，請將它們別名為此相同的 `iot:Data-ATS` 端點。

## 故障診斷

如果您無法將裝置連接至自訂網域，請確定 AWS IoT Core 已接受並套用您的伺服器憑證。您可以使用 AWS IoT Core 主控台或驗證是否 AWS IoT Core 已接受您的憑證 AWS CLI。

若要使用主 AWS IoT Core 控制台，請瀏覽至「設定」頁面，然後選取網域組態名稱。在 Server certificate details (伺服器憑證詳細資訊) 區段中，檢查狀態和狀態詳細資訊。如果憑證無效，請在 ACM 中將其取代為符合上節所列[憑證需求](#)的憑證。如果證書具有相同的 ARN，AWS IoT Core 將被拾起並自動應用它。

若要使用檢查憑證狀態 AWS CLI，請呼叫 [DescribeDomainConfiguration](#) API 並指定您的網域組態名稱。

### Note

如果您的憑證無效，AWS IoT Core 將繼續提供最後一個有效的憑證。

您可以使用下列 openssl 命令，檢查正在端點上為哪個憑證提供服務。

```
openssl s_client -connect custom-domain-name:8883 -showcerts -servername custom-domain-name
```

## 管理網域組態

您可以使用下列 API 來管理現有組態的生命週期。

- [ListDomainConfigurations](#)
- [DescribeDomainConfiguration](#)
- [UpdateDomainConfiguration](#)
- [DeleteDomainConfiguration](#)

## 檢視網域組態

要返回您的所有域配置的分頁列表 AWS 帳戶，請使用 [ListDomainConfigurations](#) API。您可以使用 [DescribeDomainConfiguration](#) API 查看特定域配置的詳細信息。此 API 採用單一 domainConfigurationName 參數，並傳回所指定組態的詳細資訊。

## 範例

## 更新網域組態

若要更新網域設定的狀態或自訂授權者，請使用 [UpdateDomainConfiguration](#) API。您可以將狀態設為 ENABLED 或 DISABLED。如果您停用網域組態，連線至該網域的裝置會收到驗證錯誤。目前您無法在網域組態中更新伺服器憑證。若要變更網域組態的憑證，您必須刪除並重新建立它。

### 範例

#### 刪除網域組態

刪除網域組態之前，請使用 [UpdateDomainConfiguration](#) API 將狀態設定為 DISABLED。這可協助您避免意外刪除端點。停用網域設定後，請使用 [DeleteDomainConfiguration](#) API 將其刪除。您必須將 AWS 受管理網域的 DISABLED 狀態設為 7 天，才能刪除它們。您可以將自訂網域置於 DISABLED 狀態，然後立即刪除它們。

### 範例

刪除網域組態後，就 AWS IoT Core 不再提供與該自訂網域相關聯的伺服器憑證。

#### 在自訂網域中輪換憑證

您可能需要定期將伺服器憑證取代為更新的憑證。您執行此動作的速率取決於憑證的有效期間。如果您已使用 AWS Certificate Manager (ACM) 產生伺服器憑證，則可以將憑證設定為自動更新。當 ACM 更新您的憑證時，AWS IoT Core 會自動取得新憑證。您不必執行任何額外的動作。如果您已從不同來源匯入伺服器憑證，則可以將其重新匯入至 ACM 來輪換憑證。如需重新匯入憑證的詳細資訊，請參閱 [重新匯入憑證](#)。

#### Note

AWS IoT Core 只有在下列情況下才會取得憑證更新。

- 新憑證具有與舊憑證相同的 ARN。
- 新憑證具有與舊憑證相同的簽署演算法、通用名稱或主體別名。

## 在網域組態中設定 TLS 設定

AWS IoT Core 提供 [預先定義的安全性原則](#)，讓您在網域組態中自訂 TLS [1.2](#) 和 [TLS 1.3](#) 的 [傳輸層安全性 \(TLS\)](#) 設定。安全政策是 TLS 通訊協定及其密碼的組合，用來決定在用戶端與伺服器之間進行 TLS 交涉期間支援的通訊協定和密碼。透過支援的安全性原則，您可以更靈活地管理裝置的 TLS 設定、在連接新裝置時套用最多的 up-to-date 安全性措施，並維持現有裝置的 TLS 設定一致。

下表說明安全政策、其 TLS 版本及支援的區域：

安全政策名稱	支援 AWS 區域
SecurityPolicyIoT	所有 AWS 區域
SecurityPolicyIoT	所有 AWS 區域
SecurityPolicyIoT	所有 AWS 區域
SecurityPolicyIoT	ap-east-1, AP-東北 -2, 在南 -1, AP-東南 -2, ca-central-1, CN-北 -1, CN-西北 -1, 歐盟-西部 -1, eu-west-2, eu-west-3, 我南 -1, SA-東 -1, 美東 -2, 美國西部 -1, 美國西部 -1, us-west-1
SecurityPolicyIoT	ap-northeast-1、ap-southeast-1、eu-central-1、eu-west-1、us-east-1、us-west-2

中的安全性原則名稱 AWS IoT Core 包含以發行年份和月份為基礎的版本資訊。如果您建立新的網域組態，安全政策將預設為 IoTSecurityPolicy\_TLS13\_1\_2\_2022\_10。如需包含通訊協定、TCP 連接埠和密碼詳細資料的安全性原則的完整資料表，請參閱[安全性原則](#)。AWS IoT Core 不支援自訂安全性原則。如需詳細資訊，請參閱[???](#)。

若要在網域組態中設定 TLS 設定，您可以使用 AWS IoT 主控台或 AWS CLI。

## 目錄

- [在網域組態中設定 TLS 設定 \(主控台\)](#)
- [在網域組態中設定 TLS 設定 \(CLI\)](#)

### 在網域組態中設定 TLS 設定 (主控台)

#### 使用 AWS IoT 主控台設定 TLS 設定

1. 登入 AWS Management Console 並開啟[AWS IoT 主控台](#)。
2. 若要在建立新網域組態時設定 TLS 設定，請遵循下列步驟。
  1. 在左側導覽窗格中，選擇設定，然後從網域組態區段中選擇建立網域組態。
  2. 在建立網域組態頁面的自訂網域設定 - 選用區段中，從選取安全政策中選擇安全政策。
  3. 遵循小工具並完成其餘步驟。選擇建立網域組態。

- 若要更新現有網域組態中的 TLS 設定，請遵循下列步驟。
  - 在左側導覽窗格中，選擇設定，然後在網域組態下選擇網域組態。
  - 在網域組態詳細資訊頁面中，選擇編輯。然後，在自訂網域設定 - 選用區段的選取安全政策下，選擇安全政策。
  - 選擇更新網域組態。

如需詳細資訊，請參閱[建立網域組態](#)和[管理網域組態](#)。

### 在網域組態中設定 TLS 設定 (CLI)

您可以使用 [create-domain-configuration](#) 或 [update-domain-configuration](#) CLI 命令，在網域組態中設定 TLS 設定。

- 若要使用 [create-domain-configuration](#) CLI 命令指定 TLS 設定：

```
aws iot create-domain-configuration \  
  --domain-configuration-name domainConfigurationName \  
  --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

此命令的輸出如下所示：

```
{  
  "domainConfigurationName": "test",  
  "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/  
test/34ga9"  
}
```

如果您建立新的網域組態，而未指定安全政策，則值將預設為 `IoTSecurityPolicy_TLS13_1_2_2022_10`。

- 若要使用 [describe-domain-configuration](#) CLI 命令說明 TLS 設定：

```
aws iot describe-domain-configuration \  
  --domain-configuration-name domainConfigurationName
```

此命令可以傳回網域組態詳細資訊，其中包含 TLS 設定，如下所示：

```
{  
  "tlsConfig": {
```

```
"securityPolicy": "IoTSecurityPolicy_TLS13_1_2_2022_10"
},
"domainConfigurationStatus": "ENABLED",
"serviceType": "DATA",
"domainType": "AWS_MANAGED",
"domainName": "d1234567890abcdefghij-ats.iot.us-west-2.amazonaws.com",
"serverCertificates": [],
"lastStatusChangeDate": 1678750928.997,
"domainConfigurationName": "test",
"domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

3. 若要使用 [update-domain-configuration](#) CLI 命令更新 TLS 設定：

```
aws iot update-domain-configuration \
  --domain-configuration-name domainConfigurationName \
  --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

此命令的輸出如下所示：

```
{
  "domainConfigurationName": "test",
  "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
test/34ga9"
}
```

4. 若要更新 ATS 端點的 TLS 設定，請執行 [update-domain-configuration](#) CLI 命令。ATS 端點的網域組態名稱為 `iot:Data-ATS`。

```
aws iot update-domain-configuration \
  --domain-configuration-name "iot:Data-ATS" \
  --tls-config securityPolicy=IoTSecurityPolicy_TLS13_1_2_2022_10
```

此命令的輸出如下所示：

```
{
  "domainConfigurationName": "iot:Data-ATS",
  "domainConfigurationArn": "arn:aws:iot:us-west-2:123456789012:domainconfiguration/
iot:Data-ATS"
}
```

如需詳細資訊，請參閱 AWS API 參考 [UpdateDomainConfiguration](#) 中的 [CreateDomainConfiguration](#) 和。

## OCSP 裝訂的伺服器憑證組態

AWS IoT Core 支援伺服器憑證的 [線上憑證狀態通訊協定 \(OCSP\)](#) 裝訂，也稱為伺服器憑證 OCSP 裝訂或 OCSP 裝訂。這是一種安全性機制，用於在傳輸層安全性 (TLS) 交握中檢查伺服器憑證上的撤銷狀態。OCSP 裝訂功 AWS IoT Core 能可讓您在自訂網域的伺服器憑證有效性中新增額外的驗證層。

您可以透過定期查詢 OCSP 回應程式 AWS IoT Core 來啟用伺服器憑證 OCSP 裝訂，以檢查憑證的有效性。OCSP 裝訂設定是使用自訂網域建立或更新網域組態的程序的一部分。OCSP 裝訂會持續檢查伺服器憑證上的撤銷狀態。這有助於確認 CA 撤銷的任何憑證不再受連線至您自訂網域的用戶端信任。如需詳細資訊，請參閱 [???](#)。

伺服器憑證 OCSP 裝訂提供即時撤銷狀態檢查，減少與檢查撤銷狀態相關的延遲，並提高安全連線的隱私權和可靠性。如需使用 OCSP 裝訂優點的詳細資訊，請參閱 [???](#)

### Note

在中不提供此功能 AWS GovCloud (US) Regions。

在本主題中：

- [什麼是 OCSP ?](#)
- [OCSP 裝訂的運作方式](#)
- [啟用伺服器憑證 OCSP 裝訂 AWS IoT Core](#)
- [在中使用伺服器憑證 OCSP 裝訂的重要注意事項 AWS IoT Core](#)
- [疑難排解伺服器憑證 OCSP 裝訂 AWS IoT Core](#)

什麼是 OCSP ?

重要概念

下列概念提供 OCSP 及相關概念的詳細資料。

OCSP

[OCSP](#) 用於在傳輸層安全性 (TLS) 交握期間檢查憑證撤銷狀態。OCSP 允許即時驗證憑證。這會確認憑證發行後尚未撤銷或過期。與傳統的憑證撤銷清單 (CRL) 相比，OCSP 也更具擴充性。OCSP 回應較小且可以有效率地產生，因此更適合大型私密金鑰基礎結構 (PKI)。

## OCSP 回應者

OCSP 回應程式 (也稱為 OCSP 伺服器) 會從尋求驗證憑證撤銷狀態的用戶端接收並回應 OCSP 要求。

## 用戶端 OCSP

在用戶端 OCSP 中，用戶端會使用 OCSP 連絡 OCSP 回應程式，以在傳輸層安全性 (TLS) 交握期間檢查憑證的撤銷狀態。

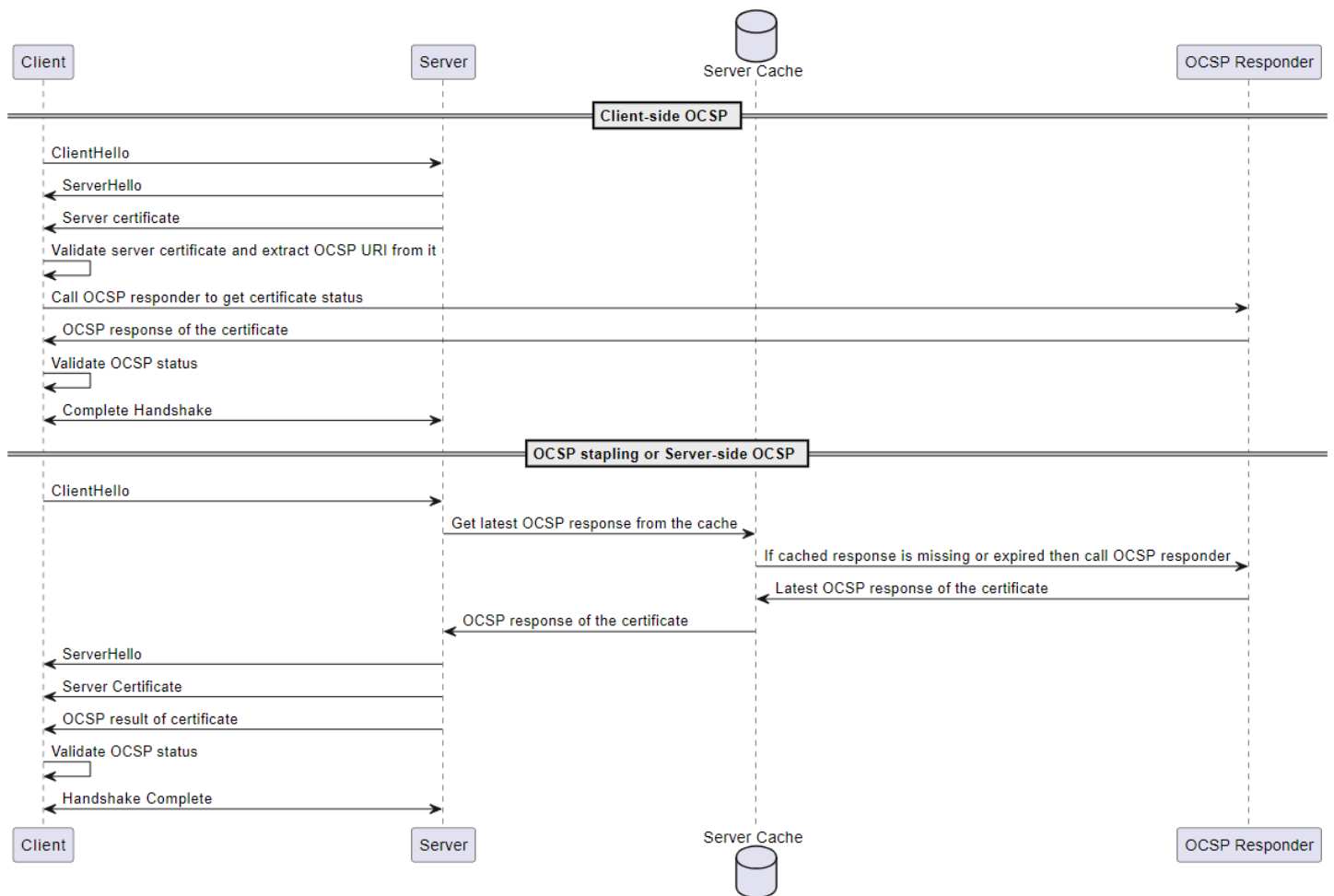
## 伺服器端 OCSP

在伺服器端 OCSP (也稱為 OCSP 裝訂) 中，伺服器會啟用 (而非用戶端) 向 OCSP 回應程式發出要求。伺服器會將 OCSP 回應裝訂到憑證，並在 TLS 交握期間將其傳回給用戶端。

## OCSP 圖表

下圖說明用戶端 OCSP 和伺服器端 OCSP 的運作方式。





## 用戶端 OCSP

1. 用戶端會傳送ClientHello訊息以啟動與伺服器的 TLS 交涉。
2. 服務器接收消息並以消息ServerHello進行響應。伺服器也會將伺服器憑證傳送至用戶端。
3. 用戶端會驗證伺服器憑證，並從中擷取 OCSP URI。
4. 用戶端會傳送憑證撤銷檢查要求給 OCSP 回應程式。
5. OCSP 回應程式會傳送 OCSP 回應。
6. 用戶端會驗證 OCSP 回應的憑證狀態。
7. TLS 握手已完成。

## 伺服器端 OCSP

1. 用戶端會傳送ClientHello訊息以啟動與伺服器的 TLS 交涉。

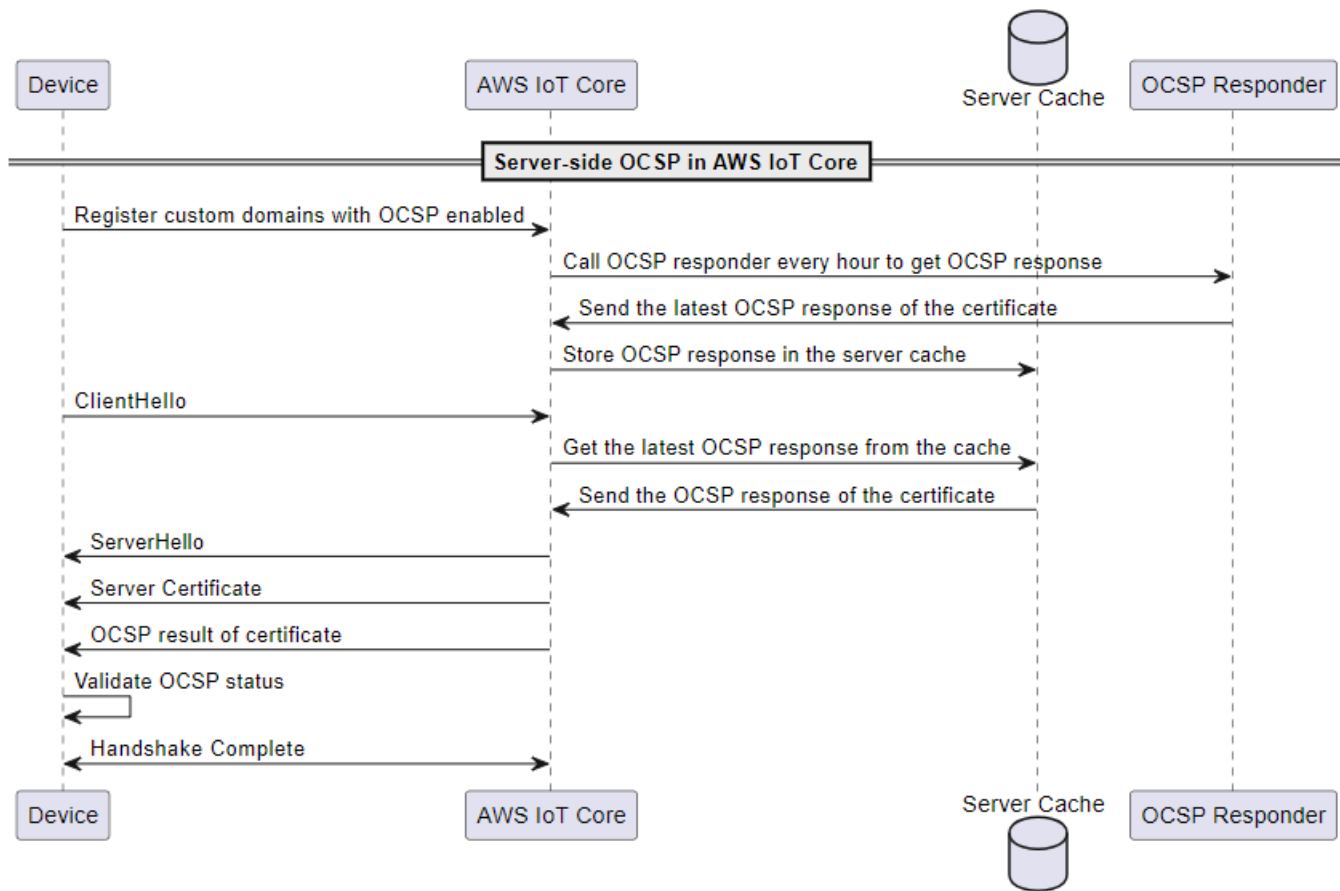
2. 伺服器會收到訊息並取得最新的快取 OCSP 回應。如果快取回應遺失或已過期，伺服器會呼叫 OCSP 回應程式以取得憑證狀態。
3. OCSP 回應程式會傳送 OCSP 回應至伺服器。
4. 伺服器發送一ServerHello條消息。伺服器也會將伺服器憑證和憑證狀態傳送給用戶端。
5. 用戶端會驗證 OCSP 憑證狀態。
6. TLS 握手已完成。

## OCSP 裝訂的運作方式

用戶端與伺服器之間的傳輸層安全性 (TLS) 交涉期間會使用 OCSP 裝訂，以檢查伺服器憑證撤銷狀態。伺服器向 OCSP 回應程式發出 OCSP 要求，並將 OCSP 回應裝訂至傳回給用戶端的憑證。藉由讓伺服器向 OCSP 回應程式發出要求，可快取回應，然後針對許多用戶端多次使用。

### OCSP 裝訂的運作方式 AWS IoT Core

下圖顯示伺服器端 OCSP 裝訂的運作方式。AWS IoT Core



1. 裝置必須在啟用 OCSP 裝訂的情況下註冊自訂網域。

2. AWS IoT Core 每小時呼叫 OCSP 回應程式以取得憑證狀態。
3. OCSP 回應程式會接收要求、傳送最新的 OCSP 回應，並儲存快取的 OCSP 回應。
4. 裝置會傳送ClientHello訊息來啟動 TLS 交握 AWS IoT Core。
5. AWS IoT Core 從伺服器快取取得最新的 OCSP 回應，此回應會以憑證的 OCSP 回應來回應。
6. 伺服器會傳送ServerHello訊息至裝置。伺服器也會將伺服器憑證和憑證狀態傳送給用戶端。
7. 裝置會驗證 OCSP 憑證狀態。
8. TLS 握手已完成。

與用戶端 OCSP 檢查相比，使用 OCSP 裝訂的好處

使用伺服器憑證 OCSP 裝訂的幾個優點總結如下：

#### 改善隱私

如果沒有 OCSP 裝訂，用戶端的裝置可能會將資訊公開給協力廠商 OCSP 回應者，進而可能損害使用者隱私權。OCSP 裝訂可讓伺服器取得 OCSP 回應並直接傳送給用戶端，藉此緩解此問題。

#### 提高可靠性

OCSP 裝訂可以提高安全連線的可靠性，因為它可以降低 OCSP 伺服器中斷的風險。裝訂 OCSP 回應時，伺服器會包含憑證的最新回應。如此一來，即使 OCSP 回應程式暫時無法使用，用戶端也能存取撤銷狀態。OCSP 裝訂有助於緩解這些問題，因為伺服器會定期擷取 OCSP 回應，並在 TLS 交握中包含快取的回應，進而減少對 OCSP 回應程式即時可用性的依賴。

#### 降低伺服器負載

OCSP 裝訂可將 OCSP 要求的回應負擔從 OCSP 回應程式卸載至伺服器。這有助於更均勻地分配負載，使憑證驗證程序更有效率和可擴充性。

#### 減少延遲

OCSP 裝訂可減少 TLS 交涉期間檢查憑證撤銷狀態的相關延遲。伺服器不必個別查詢 OCSP 伺服器，而是在交握期間傳送要求並附加 OCSP 回應與伺服器憑證。

#### 啟用伺服器憑證 OCSP 裝訂 AWS IoT Core

若要在中啟用伺服器憑證 OCSP 裝訂 AWS IoT Core，您必須為自訂網域建立網域組態或更新現有的自訂網域組態。如需使用自訂網域建立網域組態的一般資訊，請參閱[???](#)。

使用下列指示來啟用使用或的 OCSP 伺服器 AWS Management Console 裝訂。AWS CLI

## 主控台

若要使用主控台啟用伺服器憑證 OCSP 裝 AWS IoT 訂：

1. 從左側選單中選擇「設定」，然後選擇「建立網域組態或自訂網域的現有網域組態」。
2. 如果您選擇從上一個步驟建立新的網域組態，就會看到 [建立網域組態] 頁面。在 [網域組態內容] 區段中，選擇 [自訂網域]。輸入資訊以建立網域組態。

如果您選擇更新自訂網域的現有網域組態，您會看到網域組態詳細資訊頁面。選擇編輯。

3. 若要啟用 OCSP 伺服器裝訂，請選擇 [伺服器憑證組態] 子區段中的 [啟用伺服器憑證 OCSP 裝訂]。
4. 選擇建立網域組態或更新網域組態。

## AWS CLI

若要使用以下方式啟用伺服器憑證 OCSP 裝訂：AWS CLI

1. 如果您為自訂網域建立新的網域組態，啟用 OCSP 伺服器裝訂的命令可能如下所示：

```
aws iot create-domain-configuration --domain-configuration-name
  "myDomainConfigurationName" \
    --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
    --server-certificate-config "enableOCSPCheck=true|false"
```

2. 如果您更新自訂網域的現有網域組態，啟用 OCSP 伺服器裝訂的命令可能如下所示：

```
aws iot update-domain-configuration --domain-configuration-name
  "myDomainConfigurationName" \
    --server-certificate-arns arn:aws:iot:us-
east-1:123456789012:cert/
f8c1e5480266caef0fdb1bf97dc1c82d7ba2d3e2642c5f25f5ba364fc6b79ba3 \
    --server-certificate-config "enableOCSPCheck=true|false"
```

如需詳細資訊，請參閱「API 參考」[CreateDomainConfiguration](#)和[UpdateDomainConfiguration](#)「AWS IoT API 參考」。

## 在中使用伺服器憑證 OCSP 裝訂的重要注意事項 AWS IoT Core

當您在中使用伺服器憑證 OCSP 時 AWS IoT Core，請記住下列事項：

1. AWS IoT Core 只支援那些可透過公用 IPv4 位址存取的 OCSP 回應程式。
2. 中的 OCSP 裝訂功能 AWS IoT Core 不支援授權回應者。所有 OCSP 回應都必須由簽署憑證的 CA 簽署，而且 CA 必須是自訂網域憑證鏈結的一部分。
3. 中的 OCSP 裝訂功能 AWS IoT Core 不支援使用自我簽署憑證建立的自訂網域。
4. AWS IoT Core 每小時呼叫 OCSP 回應程式，並快取回應。如果對響應者的呼叫失敗，AWS IoT Core 將裝訂最新的有效響應。
5. 如果不再nextUpdateTime有效，則 AWS IoT Core 會從快取中移除回應，而 TLS 交握將不會包含 OCSP 回應資料，直到下次成功呼叫 OCSP 回應程式為止。當伺服器從 OCSP 回應程式取得有效回應之前，快取回應已過期時，就可能會發生這種情況。的值nextUpdateTime表示 OCSP 回應在此時間之前仍然有效。如需 nextUpdateTime 的相關資訊，請參閱 [???](#)。
6. 有時候，AWS IoT Core 無法接收 OCSP 回應或移除現有的 OCSP 回應，因為它已過期。如果發生這類情況，AWS IoT Core 將繼續使用自訂網域所提供的伺服器憑證，而不使用 OCSP 回應。
7. OCSP 回應的大小不得超過 4 KiB。

### 疑難排解伺服器憑證 OCSP 裝訂 AWS IoT Core

AWS IoT Core 會發出RetrieveOCSPStapleData.Success測量結果和記RetrieveOCSPStapleData錄項目。CloudWatch測量結果和記錄項目可協助偵測與擷取 OCSP 回應相關的問題。如需更多詳細資訊，請參閱 [???](#) 及 [???](#)。

## 連接至 AWS IoT FIPS 端點

AWS IoT 提供支援[聯邦資訊處理標準 \(FIPS\) 140-2](#) 的端點。FIPS 相容端點與標準 AWS 端點不同。若要以 FIPS 合規方式與 AWS IoT 互動，您必須使用下述端點搭配 FIPS 合規用戶端。主 AWS IoT 控制台不符合 FIPS 規範。

下列各節說明如何使用 REST API、SDK 或存取 FIPS 相容的 AWS IoT 端點。AWS CLI

### 主題

- [AWS IoT Core：控制平面端點](#)
- [AWS IoT Core：資料平面端點](#)
- [AWS IoT Device Management：任務資料端點](#)

- [AWS IoT Device Management : Fleet Hub 端點](#)
- [AWS IoT Device Management : 安全通道端點](#)

## AWS IoT Core : 控制平面端點

支援 [AWS IoT](#) 操作及其相關 [CLI 命令](#) 的 FIPS 合規 AWS IoT Core : 控制平面端點，都列於[依服務列出的 FIPS 端點](#)。在[依服務列出的 FIPS 端點](#)中，尋找 AWS IoT Core : 控制平面服務，並查詢適合 AWS 區域的端點。

若要在存取[AWS IoT](#)作業時使用 FIPS 相容端點，請將 AWS SDK 或 REST API 搭配適合您 AWS 區域的端點使用。

若要在執行 [aws iot CLI 命令](#)時使用 FIPS 合規端點，請將具有適合 AWS 區域 端點的 --endpoint 參數新增至命令。

## AWS IoT Core : 資料平面端點

FIPS 合規 AWS IoT Core : 資料平面端點列於[依服務列出的 FIPS 端點](#)。在[依服務列出的 FIPS 端點](#)中，尋找 AWS IoT Core : 資料平面服務，並查詢適合 AWS 區域的端點。

您可以使用 AWS IoT 裝置 SDK，並將端點提供給 SDK 的連線功能，以取代帳戶的預設 AWS IoT Core 資料平面端點，將 FIPS 相容的端點用於 AWS 區域 搭配 FIPS 相容的用戶端。連接功能特定於設 AWS IoT 備 SDK。如需連線函數的範例，請參閱 [Python AWS IoT 裝置 SDK 中的連線函數](#)。

### Note

AWS IoT 不支援 FIPS 相容的 AWS 帳戶 AWS IoT Core 特定資料平面端點。無法使用需要 [伺服器名稱指示 \(SNI\)](#) 中 AWS 帳戶特定端點的服務功能。FIPS 合規 AWS IoT Core- 資料平面端點無法支援「[多帳戶註冊憑證](#)」、「[自訂網域](#)」、「[自訂授權者](#)」和「[可設定的端點](#)」(包括支援的 [TLS 政策](#))。

## AWS IoT Device Management : 任務資料端點

FIPS 合規 AWS IoT Device Management : 任務資料端點列於[依服務列出的 FIPS 端點](#)。在[依服務列出的 FIPS 端點](#)中，尋找 AWS IoT Device Management : 任務資料服務，並查詢適合 AWS 區域的端點。

若要在執行 [aws iot-jobs-data CLI 命令](#) 時，使用 FIPS 合規 AWS IoT Device Management - 任務資料端點，請將具有適合 AWS 區域 端點的 `--endpoint` 參數新增至命令。您也可以將 REST API 與此端點搭配起來使用。

您可以使用 AWS IoT 裝置 SDK，並將端點提供給 SDK 的連線功能，以取代帳戶的預設 AWS IoT Device Management 任務資料端點，將 FIPS 相容的端點用於 AWS 區域 搭配 FIPS 相容的用戶端。連線函數是 AWS IoT 裝置 SDK 的專用函數。如需連線函數的範例，請參閱 [Python AWS IoT 裝置 SDK 中的連線函數](#)。

## AWS IoT Device Management : Fleet Hub 端點

FIPS 相容 AWS IoT Device Management-叢集中樞端點可與用於 [AWS IoT 裝置管理 CLI 命令的叢集中樞端點](#) 列在 [FIPS 端點 \(依服務分類\)](#) 中。在 [依服務列出的 FIPS 端點](#) 中，尋找 AWS IoT Device Management : Fleet Hub 服務，並查詢適合 AWS 區域的端點。

若要在執行 [aws iotfleethub CLI 命令](#) 時使用 FIPS 相容 AWS IoT Device Management-叢集中心端點，請將具有適當端點的 `--endpoint` 參數新增 AWS 區域 至命令。您也可以將 REST API 與此端點搭配起來使用。

## AWS IoT Device Management : 安全通道端點

適用於 [AWS IoT 安全通道 API](#) 的 FIPS 合規 AWS IoT Device Management : 安全通道端點以及對應的 [CLI 命令](#)，都列於 [依服務列出的 FIPS 端點](#)。在 [依服務列出的 FIPS 端點](#) 中，尋找 AWS IoT Device Management : 安全通道服務，並查詢適合 AWS 區域的端點。

若要在執行 [aws iotsecuretunneling CLI 命令](#) 時，使用 AWS IoT Device Management FIPS 合規 - 安全通道端點，請將具有適合 AWS 區域 端點的 `--endpoint` 參數新增至命令。您也可以將 REST API 與此端點搭配起來使用。

# AWS IoT 教學課程

此 AWS IoT 教學課程分為兩個學習路徑，支援兩個不同的目標。為目標選擇最佳的學習路徑。

- 想要建置概念驗證來測試或演示 AWS IoT 解決方案構思

若要使用 AWS IoT 裝置用戶端示範常見的 IoT 任務和應用程式，請依照 [the section called “使用 AWS IoT 裝置用戶端建置示範”](#) 學習路徑。此 AWS IoT 裝置用戶端提供裝置軟體，可供您套用自己的雲端資源，以最少的開發示範端對端解決方案。

如需 AWS IoT 裝置用戶端的相關資訊，請參閱 [AWS IoT 裝置用戶端](#)。

- 想了解如何建置生產軟體來部署解決方案

若要使用 AWS IoT 裝置 SDK 建立符合特定需求的解決方案軟體，請依照 [the section called “使用 AWS IoT 裝置 SDK 建置解決方案”](#) 學習路徑。

如需有關可用 AWS IoT 裝置 SDK 的資訊，請參閱 [???](#)。如需 AWS 開發套件的相關資訊，請參閱在 [AWS 上建置的工具](#)。

## AWS IoT 教學課程學習路徑選項

- [使用 AWS IoT 裝置用戶端建置示範](#)
- [使用 AWS IoT 裝置 SDK 建置解決方案](#)

## 使用 AWS IoT 裝置用戶端建置示範

此學習路徑中的教學課程會逐步引導您使用 AWS IoT 裝置用戶端來開發示範軟體。此 AWS IoT 裝置用戶端提供在 IoT 裝置上執行的軟體，以測試和示範建置於 AWS IoT 上的 IoT 解決方案的各個面向。

這些教學課程的目標是促進探索和實驗，讓您可以確信 AWS IoT 能夠在開發裝置軟體前支援解決方案。

您會在本教學課程中學到什麼：

- 如何準備 Raspberry Pi，作為搭配 AWS IoT 使用的 IoT 裝置
- 如何藉由使用您裝置上的 AWS IoT 裝置用戶端示範 AWS IoT 功能



在此學習路徑中，您要在 Raspberry Pi 上安裝 AWS IoT 裝置用戶端並在雲端中建立 AWS IoT 資源，以便示範 IoT 解決方案構思。雖然此學習路徑中的教學課程會透過使用 Raspberry Pi 來示範功能，但它們會說明目標和程序，以便協助您調整它們來適應其他裝置。

## 使用 AWS IoT 裝置用戶端建置示範的先決條件

本節說明在此學習路徑中開始教學課程之前，您需要擁有的先決條件。

若要完成此學習路徑中的教學課程，您需要：

- 一個 AWS 帳戶

您可以使用現有的 AWS 帳戶 (如果有)，但可能會需要新增其他角色或許可，才能使用這些自學課程使用的 AWS IoT 功能。

如需建立新的 AWS 帳戶，請參閱 [the section called “設定您的 AWS 帳戶”](#)。

- Raspberry Pi 或兼容的 IoT 裝置

教學課程使用 [Raspberry Pi](#)，因為它有不同形式，它無處不在，也是一個相對便宜的示範裝置。教學課程已在 [Raspberry Pi 3 Model B+](#)、[Raspberry Pi 4 Model B](#)，以及在執行 Ubuntu Server 20.04 LTS (HVM) 的 Amazon EC2 執行個體上進行測試。若要使用 AWS CLI 並執行命令，我們建議您使用最新版本的 Raspberry Pi OS ([Raspberry Pi OS \(64 位元\)](#) 或 OS Lite)。較早版本的作業系統可能適用，但我們尚未對其進行測試。

### Note

教學課程會說明每個步驟的目標，以便協助您調整它們來適應尚未嘗試過的 IoT 硬體；不過，這些教學課程並不會明確說明如何調整它們來適應其他裝置。

- 熟悉 IoT 裝置的作業系統

這些教學課程中的步驟是假設您熟悉使用基本的 Linux 命令和操作 Raspberry Pi 支援的命令列介面。如果不熟悉這些作業，您就需要給自己更多時間來完成教學課程。

若要完成這些教學課程，您必須了解如何：

- 安全地執行基本裝置作業，例如組裝和連接元件、將裝置連接至所需電源，以及安裝和移除記憶卡。
- 在裝置上上傳及下載系統軟體和檔案。如果裝置不使用卸除式儲存裝置 (例如 microSD 卡)，您必須要知道如何連接至裝置，並在裝置上上傳及下載系統軟體和檔案。

- 將裝置連接至打算使用該裝置的網路上。
- 使用 SSH 終端機或類似程式從其他電腦連接至裝置。
- 使用命令列介面來建立、複製、移動、重新命名以及設定裝置上的檔案和目錄許可。
- 在裝置上安裝新程式。
- 使用 FTP 或 SCP 等工具在裝置之間傳輸檔案。
- IoT 解決方案的開發與測試環境

教學課程會說明所需的軟體和硬體；不過，教學課程會假設您可以執行可能未明確說明的作業。這類硬體和操作的範例包括：

- 要下載並存放檔案的本機主機電腦

對於 Raspberry Pi，這通常是個人電腦或筆記型電腦，可以讀取和寫入 microSD 記憶卡。本機主機電腦必須：

- 連線到網際網路。
- 已安裝並設定妥 [AWS CLI](#)。
- 有一個支援 AWS 主控台的 Web 瀏覽器
- 將本機主機電腦連接到裝置，以便與裝置進行通訊、輸入命令及傳輸檔案的方式

在 Raspberry Pi 上，這通常是使用 SSH 和 SCP 從本機主機電腦完成。

- 要連接至 IoT 裝置的顯示器和鍵盤

這些設備很有幫助，但不需要這些設備就能完成教學課程。

- 本機主機電腦和 IoT 裝置連線至網際網路的方式

這可能是連接至網際網路的路由器或閘道的有線或無線網路連線。本機主機也必須能夠連接至 Raspberry Pi。這可能需要它們位於相同的區域網路上。教學課程無法示範如何針對特定裝置或裝置組態進行設定，但是會示範如何測試此連線。

- 存取區域網路的路由器，檢視連網裝置

若要完成此學習路徑中的教學課程，您必須能夠找到 IoT 裝置的 IP 地址。

在區域網路上，您可以透過存取裝置所連線之網路路由器的管理介面來完成這項操作。如果可以在路由器中為裝置指派固定 IP 地址，則可以在每次裝置重新啟動後簡化重新連線的流程。

如果裝置連接了鍵盤和顯示器，ifconfig 可顯示出裝置的 IP 地址。

在擁有所有材料之後，請繼續 [the section called “準備用於 AWS IoT 裝置用戶端的裝置”](#)。

此學習路徑中的教學課程

- [教學課程：準備用於 AWS IoT 裝置用戶端的裝置](#)
- [教學課程：安裝和設定 AWS IoT 裝置用戶端](#)
- [教學課程：示範 MQTT 訊息與 AWS IoT 裝置用戶端通訊](#)
- [教學課程：使用 AWS IoT 裝置用戶端來示範遠端動作 \(任務\)](#)
- [教學課程：在執行 AWS IoT 裝置用戶端教學課程後清除](#)

## 教學課程：準備用於 AWS IoT 裝置用戶端的裝置

本教學課程會帶您逐步進行 Raspberry Pi 的初始化，準備好用於學習路徑的後續教學課程。

本教學課程的目標是安裝裝置作業系統的當前版本，並確保您可以在開發環境中與裝置進行通訊。

若要開始此教學課程：

- 讓這些列於 [the section called “使用 AWS IoT 裝置用戶端建置示範的先決條件”](#) 的項目可供使用並準備就緒。

此教學課程約需 90 分鐘方能完成。

完成本教學課程時：

- IoT 裝置會擁有最新的作業系統。
- IoT 裝置將具有後續教學課程所需的額外軟體。
- 您會知道裝置已連線至網際網路。
- 您會在裝置上安裝必要的憑證。

完成本教學課程後，下一個教學課程會準備好裝置，以進行使用 AWS IoT 裝置用戶端的示範。

本教學課程中的程序

- [步驟 1：安裝並更新裝置的作業系統](#)
- [步驟 2：在裝置上安裝並確認必要軟體](#)
- [步驟 3：測試裝置並儲存 Amazon CA 憑證](#)

## 步驟 1：安裝並更新裝置的作業系統

本節中的程序是描述如何將 Raspberry Pi 用於其系統磁碟機的 microSD 卡初始化。Raspberry Pi 的 microSD 卡包含其作業系統 (OS) 軟體及其應用程式檔案儲存的空間。如果不使用 Raspberry Pi，請依照裝置的指示安裝並更新裝置的作業系統軟體。

完成本節之後，您應該可以啟動 IoT 裝置，並從本機主機電腦上的終端機程式連線至該裝置。

必要設備：

- 本機開發和測試環境
- 可以連接至網際網路的 Raspberry Pi 或 IoT 裝置
- 具有至少 8 GB 容量或足夠儲存作業系統和所需軟體的 microSD 記憶卡。

### Note

在為這些練習選取 microSD 卡時，請盡量選擇容量大但體積小的卡片。小型 SD 卡的備份和更新速度會更快。在 Raspberry Pi 上，完成這些教學課程不需要超過 8 GB 的 microSD 卡。如果特定應用程式需要更多空間，您在這些教學課程中儲存的較小映像檔案可以調整較大記憶卡上的檔案系統大小，以使用所選記憶卡的所有支援空間。

選用設備：

- 連接到 Raspberry Pi 的 USB 鍵盤
- HDMI 顯示器和將顯示器至 Raspberry Pi 的纜線

本節中的程序：

- [將裝置的作業系統載入至 microSD 卡](#)
- [使用新的作業系統啟動 IoT 裝置](#)
- [將本機主機電腦連接至裝置](#)

將裝置的作業系統載入至 microSD 卡

此程序會使用本機主機電腦，將裝置的作業系統載入至 microSD 卡。

**Note**

如果裝置的作業系統未使用卸除式儲存媒體，請使用該裝置的程序安裝作業系統，然後繼續進行 [the section called “使用新的作業系統啟動 IoT 裝置”](#)。

## 要在 Raspberry Pi 上安裝作業系統

1. 在本機主機電腦上，下載並解壓縮要使用的 Raspberry Pi 作業系統映像。最新版本可從 <https://www.raspberrypi.com/software/operating-systems/> 上下載

### 選擇 Raspberry Pi 作業系統的版本

本教學課程會使用 Raspberry Pi OS Lite 版本，因為這是在此學習路徑中支援這些教學課程的最小版本。這個版本的 Raspberry Pi 作業系統只有一個命令列介面，沒有圖形使用者介面。具有圖形化使用者介面的最新 Raspberry Pi 作業系統版本也可以使用這些教學課程；不過，此學習路徑中描述的程序只會使用命令列介面，以便在 Raspberry Pi 上執行操作。

2. 請將 microSD 卡插入主機電腦。
3. 使用 SD 卡影像工具，將解壓縮的作業系統映像檔案寫入 microSD 卡。
4. 寫入 Raspberry Pi 作業系統映像至 microSD 卡後：
  - a. 在命令列視窗或檔案總管視窗中，開啟 microSD 卡上的 BOOT 分割區。
  - b. 在 microSD 卡的 BOOT 分割區中，於根目錄中建立一個名為 ssh 且沒有檔案副檔名和內容的空白檔案。這樣會告知 Raspberry Pi 在它第一次啟動時啟用 SSH 通訊。
5. 退出 microSD 卡並安全地從本機主機電腦移除。

您的 microSD 記憶卡已準備進行 [the section called “使用新的作業系統啟動 IoT 裝置”](#)。

### 使用新的作業系統啟動 IoT 裝置

此程序會安裝 microSD 卡，並使用下載的作業系統首次啟動 Raspberry Pi。

### 使用新的作業系統啟動 IoT 裝置

1. 當電源與裝置斷開時，在上一步驟中將 microSD 卡 ([the section called “將裝置的作業系統載入至 microSD 卡”](#)) 插入 Raspberry Pi。
2. 將裝置連線至有線網路。
3. 這些教學課程將使用 SSH 終端機與本機主機電腦的 Raspberry Pi 進行互動。

如果也想直接與裝置互動，您可以：

- a. 在將 HDMI 顯示器連接至裝置來查看 Raspberry Pi 的主控制台消息之前，您可以將本機主機電腦上的終端機視窗連接至 Raspberry Pi。
  - b. 如果想直接與 Raspberry Pi 互動，請將 USB 鍵盤連接至裝置。
4. 連接電源至 Raspberry Pi，接著等待大約一分鐘來進行初始化。

如果有顯示器連接至 Raspberry Pi，您可以觀看它的啟動過程。

5. 找出裝置的 IP 地址：
  - 如果將 HDMI 顯示器連接至 Raspberry Pi，IP 地址會出現在顯示器上的訊息中
  - 如果能存取 Raspberry Pi 連接的路由器，您可以在路由器的管理介面中查看它的地址。

在取得 Raspberry Pi 的 IP 地址後，您已準備好進行 [the section called “將本機主機電腦連接至裝置”](#)。

將本機主機電腦連接至裝置

此程序會使用本機主機電腦上的終端程式連接至 Raspberry Pi，並更改其預設密碼。

將本機主機電腦連接至裝置

1. 在本機主機電腦上，開啟 SSH 終端程式：

- Windows : PuTTY
- Linux/macOS : Terminal

#### Note

PuTTY 不會在 Windows 上自動安裝。如果電腦中沒有，您需要下載並安裝 PuTTY。

2. 將終端程式連接至 Raspberry Pi 的 IP 地址，並使用其預設憑證登入。

```
username: pi  
password: raspberrypi
```

3. 登入至 Raspberry Pi 之後，變更 pi 使用者密碼。

```
passwd
```

依照提示變更密碼。

```
Changing password for pi.  
Current password: raspberrypi  
New password: YourNewPassword  
Retype new password: YourNewPassword  
passwd: password updated successfully
```

在終端視窗收到 Raspberry Pi 命令列提示並更改密碼之後，即表示您已經準備好繼續進行 [the section called “步驟 2：在裝置上安裝並確認必要軟體”](#)。

## 步驟 2：在裝置上安裝並確認必要軟體

本節中的程序將繼續[上一節](#)的程序，將 Raspberry Pi 的作業系統更新至最新版本，並在 Raspberry Pi 上安裝下一節要用來建置並安裝 AWS IoT 裝置用戶端的軟體。

完成本節後，Raspberry Pi 會擁有最新的作業系統、本學習路徑教學課程所需的軟體，且會根據位置進行設定。

必要設備：

- [上一節](#)中的本機開發和測試環境
- [上一節](#)中使用的 Raspberry Pi
- [上一節](#)中的 microSD 記憶卡

### Note

Raspberry Pi Model 3+ 和 Raspberry Pi Model 4 可執行此學習路徑中描述的所有命令。如果 IoT 裝置無法編譯軟體或執行 AWS Command Line Interface，您可能需要在本機主機電腦上安裝必要的編譯器來建置軟體，然後將其傳輸至 IoT 裝置。如需有關如何為裝置安裝和建置軟體的詳細資訊，請參閱裝置軟體的說明文件。

本節中的程序：

- [更新作業系統和軟體](#)
- [安裝必要的應用程式和程式庫](#)
- [\(選用\) 儲存 microSD 卡映像](#)

## 更新作業系統和軟體

此程序會更新作業系統軟體。

### 更新 Raspberry Pi 上的作業系統軟體

在本機主機電腦的終端機視窗中執行這些步驟。

1. 輸入這些命令來更新 Raspberry Pi 上的系統軟體。

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y autoremove
```

2. 更新 Raspberry Pi 的地區設定和時區設定 (選用)。

輸入此指令來更新裝置的地區設定和時區設定。

```
sudo raspi-config
```

- a. 若要設定裝置的地區設定：

- i. 在 Raspberry Pi Software Configuration Tool (raspi-config) (Raspberry Pi 軟體組態工具 (raspi-config)) 畫面中，選擇選項 5。

#### **5 Localisation Options Configure language and regional settings**

使用 Tab 鍵來移動至 <Select> (選擇)，然後按 space bar。

- ii. 在當地語系化選項選單中，選擇選項 L1。

#### **L1 Locale Configure language and regional settings**

使用 Tab 鍵來移動至 <Select> (選擇)，然後按 space bar。

- iii. 在地區選項清單中，使用方向鍵捲動並使用 space bar 標記所需地區，選擇要在 Raspberry Pi 上安裝的地區設定。

建議為美國選擇 **en\_US.UTF-8**。

- iv. 選取裝置的地區後，請使用 Tab 鍵選擇 <OK> (確定)，然後按 space bar 來顯示 Configuring locales (設定地區設定) 的確認頁面。

- b. 若要設定裝置的時區：



- i. 在 `raspi-config` 畫面中，選擇選項 5。

### 5 Localisation Options Configure language and regional settings

使用 Tab 鍵來移動至 <Select> (選擇)，然後按 space bar。

- ii. 在當地語系化選項選單中，使用方向鍵選擇選項 L2：

#### L2 time zone Configure time zone

使用 Tab 鍵來移動至 <Select> (選擇)，然後按 space bar。

- iii. 在 Configuring tzdata 選單中，從清單中選擇地理區域。

使用 Tab 鍵來移動至 <OK> (確定)，然後按 space bar。

- iv. 在城市清單中，使用方向鍵來選擇您所在時區的城市。

若要設定時區，請使用 Tab 鍵來移動至 <OK> (確定)，然後按 space bar。

- c. 完成更新設定時，請使用 Tab 鍵來移動至 <Finish> (完成)，然後按 space bar 來關閉 `raspi-config` 應用程式。

3. 輸入此命令來重新啟動 Raspberry Pi。

```
sudo shutdown -r 0
```

4. 等待 Raspberry Pi 重新啟動。
5. 待 Raspberry Pi 重新啟動後，請將本機主機電腦上的終端機視窗重新連接至 Raspberry Pi。

Raspberry Pi 系統軟體已設定完成，您已經準備好繼續進行 [the section called “安裝必要的應用程式和程式庫”](#)。

### 安裝必要的應用程式和程式庫

此程序會安裝後續教學課程使用的應用程式軟體和程式庫。

如果正在使用 Raspberry Pi，或者如果可以在 IoT 裝置上編譯所需軟體，請在本機主機電腦上的終端機視窗中執行這些步驟。如果必須在本機主機電腦上編譯 IoT 裝置的軟體，請檢閱 IoT 裝置的軟體說明文件，取得如何在裝置上執行這些步驟的相關資訊。

### 在 Raspberry Pi 上安裝應用程式軟體和程式庫

1. 輸入此指令來安裝應用程式軟體和程式庫。

```
sudo apt-get -y install build-essential libssl-dev cmake unzip git python3-pip
```

2. 輸入這些指令來確認是否已安裝正確版本的軟體。

```
gcc --version  
cmake --version  
openssl version  
git --version
```

3. 確認已安裝下列版本的應用程式軟體：

- gcc : 9.3.0 或更新版本
- cmake : 3.10.x 或更新版本
- OpenSSL : 1.1.1 或更新版本
- git : 2.20.1 或更新版本

如果 Raspberry Pi 有所需應用程式軟體的可接受版本，您就可以繼續進行 [the section called “\(選用\) 儲存 microSD 卡映像”](#)。

### (選用) 儲存 microSD 卡映像

在此學習路徑的整個教學課程中，您會遇到這些程序來將 Raspberry Pi 的 microSD 卡映像儲存至本機主機電腦上的檔案。雖然鼓勵這樣操作，但這不是必要任務。透過在建議位置上儲存 microSD 卡映像，您可以略過此學習路徑中儲存點之前的程序；若您發現需要重試某些項目，這就可以節省時間。不定期儲存 microSD 卡映像的結果是，當 microSD 卡損壞或意外錯誤設定應用程式或其設定時，您可能需要從頭開始重新開始學習路徑中的教學課程。

此時，Raspberry Pi 的 microSD 卡已經擁有更新的作業系統和加載的基本應用程式軟體。您現在可以將 microSD 卡的內容儲存至檔案，節省完成上述步驟所花費的時間。擁有裝置 microSD 卡映像目前的映像，可讓您從此開始繼續或重試教學課程或程序，無需從頭開始安裝和更新軟體。

### 將 microSD 卡映像儲存至檔案

1. 輸入此命令來關閉 Raspberry Pi。

```
sudo shutdown -h 0
```

2. Raspberry Pi 完全關閉後，請移除其電源。
3. 從 Raspberry Pi 中取出 microSD 卡。

4. 在本機主機電腦上：
  - a. 插入 microSD 卡。
  - b. 使用 SD 卡映像工具，將 microSD 卡的映像儲存至檔案中。
  - c. 儲存 microSD 卡的映像後，請從本機主機電腦中退出記憶卡。
5. 在 Raspberry Pi 斷開電源後，將 microSD 卡插入 Raspberry Pi。
6. 將電源連接至 Raspberry Pi。
7. 等待大約一分鐘後，在本機主機電腦上重新連線至連接至 Raspberry Pi 的本機主機電腦終端機視窗，然後登入 Raspberry Pi。

### 步驟 3：測試裝置並儲存 Amazon CA 憑證

本節中的程序將繼續[上一節](#)的程序，將安裝 AWS Command Line Interface 和用來驗證與 AWS IoT Core 之間連線的憑證授權機構憑證。

完成本節後，您就會知道 Raspberry Pi 擁有必要的系統軟體來安裝 AWS IoT 裝置用戶端，並且它可以連線至網際網路。

必要設備：

- [上一節](#)中的本機開發和測試環境
- [上一節](#)中使用的 Raspberry Pi
- [上一節](#)中的 microSD 記憶卡

本節中的程序：

- [安裝 AWS Command Line Interface](#)
- [設定 AWS 帳戶 憑證](#)
- [下載 Amazon 根憑證授權機構憑證](#)
- [\(選用\) 儲存 microSD 卡映像](#)

安裝 AWS Command Line Interface

此程序會安裝 AWS CLI 到 Raspberry Pi。

如果使用的是 Raspberry Pi，或者如果可以在 IoT 裝置上編譯軟體，請在本機主機電腦上的終端機視窗中執行這些步驟。如果必須在本機主機電腦上編譯 IoT 裝置的軟體，請檢閱 IoT 裝置的軟體說明文件，取得其所需之程式庫的相關資訊。

## 在 Raspberry Pi 上安裝 AWS CLI

1. 執行這些命令來下載和安裝 AWS CLI。

```
export PATH=$PATH:~/local/bin # configures the path to include the directory with
the AWS CLI
git clone https://github.com/aws/aws-cli.git # download the AWS 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. 執行此命令來安裝 AWS CLI。此命令最多需要 15 分鐘的時間即可完成。

```
pip3 install . # install the AWS CLI
```

3. 執行此命令來確認現在已安裝正確版本的 AWS CLI。

```
aws --version
```

AWS CLI 版本至少應為 2.2 或更新版本。

如果 AWS CLI 顯示為目前版本，您就可以繼續進行 [the section called “設定 AWS 帳戶 憑證”](#)。

### 設定 AWS 帳戶 憑證

在此過程中，您會取得 AWS 帳戶 憑證，並將憑證新增至 Raspberry Pi 上使用。

### 新增 AWS 帳戶 憑證到裝置

1. 從 AWS 帳戶 取得 Access Key ID (存取金鑰 ID) 和 Secret Access Key (私密存取金鑰)，在裝置上驗證 AWS CLI。

如果您剛使用 AWS IAM，<https://aws.amazon.com/premiumsupport/knowledge-center/create-access-key/> 介紹了要在 AWS 主控台上執行的程序，以建立要在裝置上使用的 AWS IAM 憑證。

2. 在連接至 Raspberry Pi 的本機主機電腦終端機視窗上，以及裝置的 Access Key ID (存取金鑰 ID) 和 Secret Access Key (私密存取金鑰) 憑證：

- a. 使用此命令執行 AWS 設定應用程式：

```
aws configure
```

- b. 出現提示時，請輸入憑證和組態資訊：

```
AWS Access Key ID: your Access Key ID  
AWS Secret Access Key: your Secret Access Key  
Default region name: your AWS ## code  
Default output format: json
```

3. 執行此命令來測試裝置對 AWS 帳戶 和 AWS IoT Core 端點的存取權。

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

應該會返回 AWS 帳戶 專用 AWS IoT 資料端點，如下列範例所示：

```
{  
  "endpointAddress": "a3EXAMPLEffp-ats.iot.us-west-2.amazonaws.com"  
}
```

如果看到 AWS 帳戶 專用 AWS IoT 資料端點，表示 Raspberry Pi 有繼續進行 [the section called “下載 Amazon 根憑證授權機構憑證”](#) 的連線能力和許可。

#### Important

AWS 帳戶 憑證現在儲存在 Raspberry Pi 上的 microSD 卡。這會方便 AWS 未來與您和您在這些教學課程中所建立的軟體互動，它們也會根據預設被儲存並複製到您在此步驟之後製作的任一個 microSD 卡映像中。

為了保護 AWS 帳戶 憑證的安全，在儲存任何其他 microSD 卡映像之前，請考慮再次執行 `aws configure` 並輸入 Access Key ID (存取金鑰 ID) 和 Secret Access Key (私密存取金鑰) 的隨機字元來清除憑證，防止 AWS 帳戶 憑證遭到洩露。

如果不小心儲存了 AWS 帳戶 憑證，您可以在 AWS IAM 主控台上停用憑證。

## 下載 Amazon 根憑證授權機構憑證

此程序會下載並儲存 Amazon 根憑證憑證授權機構 (CA) 的憑證副本。下載並儲存此憑證以供後續教學課程使用，也可以測試裝置與 AWS 服務之間的連線能力。

### 下載並儲存 Amazon 根憑證授權機構憑證

1. 執行此命令，為憑證建立目錄。

```
mkdir ~/certs
```

2. 執行此命令來下載 Amazon 根憑證授權機構憑證。

```
curl -o ~/certs/AmazonRootCA1.pem https://www.amazontrust.com/repository/AmazonRootCA1.pem
```

3. 執行這些命令來設定憑證目錄及其檔案的存取權。

```
chmod 745 ~  
chmod 700 ~/certs  
chmod 644 ~/certs/AmazonRootCA1.pem
```

4. 執行此命令來查看新目錄中的憑證授權機構憑證檔案。

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

您應該會看到類似這樣的項目。日期和時間會有所不同；不過，檔案大小和所有其他資訊應該與此處所示相同。

```
-rw-r--r-- 1 pi pi 1188 Oct 28 13:02 AmazonRootCA1.pem
```

如果檔案大小不是 1188，請檢查 curl 命令參數。您可能下載了不正確的檔案。

### (選用) 儲存 microSD 卡映像

此時，Raspberry Pi 的 microSD 卡已經擁有更新的作業系統和加載的基本應用程式軟體。

### 將 microSD 卡映像儲存至檔案

1. 在本機主機電腦上的終端機視窗中清除 AWS 憑證。

- a. 使用此命令執行 AWS 設定應用程式：

```
aws configure
```

- b. 出現提示時，請取代憑證。您可以按 Enter 鍵，讓 Default region name (預設區域名稱) 和 Default output format (預設輸出格式) 保持不變。

```
AWS Access Key ID [*****YT2H]: XYXYXYXYX
AWS Secret Access Key [*****9p1H]: XYXYXYXYX
Default region name [us-west-2]:
Default output format [json]:
```

2. 輸入此命令來關閉 Raspberry Pi。

```
sudo shutdown -h 0
```

3. Raspberry Pi 完全關閉之後，移除其電源連接器。
4. 從裝置中取出 microSD 卡。
5. 在本機主機電腦上：
  - a. 插入 microSD 卡。
  - b. 使用 SD 卡映像工具，將 microSD 卡的映像儲存至檔案中。
  - c. 儲存 microSD 卡的映像後，請從本機主機電腦中退出記憶卡。
6. 在 Raspberry Pi 斷開電源後，將 microSD 卡插入 Raspberry Pi。
7. 將電源連接至裝置。
8. 大約一分鐘後，在本機主機電腦上重新啟動終端機視窗工作階段，然後登入裝置。

先不要重新輸入 AWS 帳戶憑證。

在重新啟動並登入至 Raspberry Pi 之後，您就可以繼續進行 [the section called “安裝並設定 AWS IoT 裝置用戶端”](#)。

## 教學課程：安裝和設定 AWS IoT 裝置用戶端

此教學課程會逐步解說 AWS IoT 裝置用戶端的安裝與設定，以及建立將在此示範和其他示範中使用的 AWS IoT 資源。

若要開始此教學課程：

- 準備好[先前教學課程](#)中所述的本機主機電腦和 Raspberry Pi。

此教學課程約需 90 分鐘方能完成。

完成此主題時：

- IoT 裝置已準備就緒，可在其他 AWS IoT 裝置用戶端示範中使用。
- 您已經在 AWS IoT Core 中佈建 IoT 裝置。
- 您已經在裝置上下載並安裝 AWS IoT 裝置用戶端。
- 您已經儲存了裝置 microSD 卡的映像，以供後續教學課程使用。

必要設備：

- [上一節](#)中的本機開發和測試環境
- [上一節](#)中使用的 Raspberry Pi
- [上一節](#)中使用的 Raspberry Pi microSD 記憶卡

本教學課程中的程序

- [步驟 1：下載並儲存 AWS IoT 裝置用戶端](#)
- [\(選用\) 儲存 microSD 卡映像](#)
- [第 2 步：在 AWS IoT 中佈建 Raspberry Pi](#)
- [步驟 3：設定要測試連線能力的 AWS IoT 裝置用戶端](#)

## 步驟 1：下載並儲存 AWS IoT 裝置用戶端

本節中的程序會下載、編譯 AWS IoT 裝置用戶端，並將其安裝在 Raspberry Pi 上。測試安裝後，您可以儲存 Raspberry Pi 的 microSD 卡映像，以便在再次嘗試教學課程時使用。

本節中的程序：

- [下載並建置 AWS IoT 裝置用戶端](#)
- [建立教學課程所使用的目錄](#)



## 下載並建置 AWS IoT 裝置用戶端

此程序會在 Raspberry Pi 上安裝 AWS IoT 裝置用戶端。

在連接至 Raspberry Pi 本機主機電腦上的終端機視窗中執行這些命令。

在 Raspberry Pi 上安裝 AWS IoT 裝置用戶端

1. 輸入這些命令，在 Raspberry Pi 上下載並建置 AWS IoT 裝置用戶端。

```
cd ~
git clone https://github.com/aws-labs/aws-iot-device-client aws-iot-device-client
mkdir ~/aws-iot-device-client/build && cd ~/aws-iot-device-client/build
cmake ../
```

2. 執行此命令來建置 AWS IoT 裝置用戶端。此命令最多需要 15 分鐘的時間即可完成。

```
cmake --build . --target aws-iot-device-client
```

AWS IoT 裝置用戶端編譯時出現的警告訊息可予以忽略。

這些教學課程已經搭配在 gcc 上建置的 AWS IoT 裝置用戶端進行過測試，(Raspbian 10.2.1-6+rpi1) 10.2.1 20210110 版本在 Raspberry Pi OS (bullseye) 2021 年 10 月 30 日的版本在 gcc 上測試過，(Raspbian 8.3.0-6+rpi1) 8.3.0 在 Raspberry Pi OS (buster) 2021 年 5 月 7 日的版本上測試過。

3. 在 AWS IoT 裝置用戶端完成建置後，透過執行此命令來進行測試。

```
./aws-iot-device-client --help
```

如果看到 AWS IoT 裝置用戶端的命令列說明，表示 AWS IoT 裝置用戶端已成功建置且已準備好供您使用。

### 建立教學課程所使用的目錄

此程序會在 Raspberry Pi 上建立目錄，用來存放教學課程在此學習路徑中使用的檔案。

若要在這學習路徑中建立教學課程所使用的目錄：

1. 執行這些命令來建立必要的目錄。

```
mkdir ~/dc-configs
mkdir ~/policies
mkdir ~/messages
mkdir ~/certs/testconn
mkdir ~/certs/pubsub
mkdir ~/certs/jobs
```

2. 執行這些命令來設定新目錄的許可。

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 700 ~/certs/pubsub
chmod 700 ~/certs/jobs
```

建立這些目錄並設定其許可後，請繼續進行 [the section called “\(選用\) 儲存 microSD 卡映像”](#)。

### (選用) 儲存 microSD 卡映像

此時，Raspberry Pi 的 microSD 卡已經擁有更新的作業系統、基本的應用程式軟體和 AWS IoT 裝置用戶端。

如果想再次嘗試這些練習和教學課程，可以略過上述程序，方法是將隨此程序儲存的 microSD 卡映像寫入新的 microSD 卡，然後繼續進行 [the section called “第 2 步：在 AWS IoT 中佈建 Raspberry Pi”](#)。

若要將 microSD 記憶卡映像儲存至檔案中：

在連接至 Raspberry Pi 本機主機電腦上的終端機視窗中：

1. 確認尚未儲存 AWS 帳戶 憑證。
  - a. 使用此命令執行 AWS 設定應用程式：

```
aws configure
```

- b. 如果憑證已儲存 (若顯示在提示中)，請在提示出現時輸入 **XYXYXYXYX** 字串，如下所示。將 Default region name (預設區域名稱) 和 Default output format (預設輸出格式) 保留為空白。

```
AWS Access Key ID [*****XYXYX]: XYXYXYXYX
AWS Secret Access Key [*****XYXYX]: XYXYXYXYX
```

```
Default region name:  
Default output format:
```

2. 輸入此命令來關閉 Raspberry Pi。

```
sudo shutdown -h 0
```

3. Raspberry Pi 完全關閉之後，移除其電源連接器。
4. 從裝置中取出 microSD 卡。
5. 在本機主機電腦上：
  - a. 插入 microSD 卡。
  - b. 使用 SD 卡映像工具，將 microSD 卡的映像儲存至檔案中。
  - c. 儲存 microSD 卡的映像後，請從本機主機電腦中退出記憶卡。

您可以在 [the section called “第 2 步：在 AWS IoT 中佈建 Raspberry Pi”](#) 中繼續使用此 microSD 卡。

## 第 2 步：在 AWS IoT 中佈建 Raspberry Pi

本節的程序是從已安裝了 AWS CLI 和 AWS IoT 裝置用戶端的儲存 microSD 映像開始，然後是建立會在 AWS IoT 中佈建 Raspberry Pi 的 AWS IoT 資源和裝置憑證。

在 Raspberry Pi 中安裝 microSD 卡

此程序會安裝載入了所需軟體並設定至 Raspberry Pi 的 microSD 卡，然後設定 AWS 帳戶，以便您可以繼續使用此學習路徑中的教學課程。

使用 [the section called “\(選用\) 儲存 microSD 卡映像”](#) 中的 microSD 卡，其中包含此學習路徑中的練習和教學課程所需的軟體。

在 Raspberry Pi 安裝 microSD 卡

1. 在 Raspberry Pi 斷開電源後，將 microSD 卡插入 Raspberry Pi。
2. 將電源連接至 Raspberry Pi。
3. 大約一分鐘後，在本機主機電腦上重新啟動終端機視窗工作階段，然後登入 Raspberry Pi。
4. 在本機主機電腦的終端機視窗中，使用 Raspberry Pi 的 Access Key ID (存取金鑰 ID) 和 Secret Access Key (私密存取金鑰) 憑證：
  - a. 使用此命令執行 AWS 設定應用程式：

**aws configure**

- b. 出現提示時，請輸入 AWS 帳戶 憑證和組態資訊：

```
AWS Access Key ID [*****YXYX]: your Access Key ID
AWS Secret Access Key [*****YXYX]: your Secret Access Key
Default region name [us-west-2]: your AWS ## code
Default output format [json]: json
```

在還原 AWS 帳戶 憑證之後，您就可以繼續進行 [the section called “在 AWS IoT Core 上佈建裝置”](#)。

### 在 AWS IoT Core 上佈建裝置

本節中的程序會建立能在 AWS IoT 中佈建 Raspberry Pi 的 AWS IoT 資源。建立這些資源時，系統會要求您記錄各種資訊。AWS IoT 裝置用戶端組態會在接下來的程序中使用此資訊。

若要讓 Raspberry Pi 搭配 AWS IoT 工作，則必須對其進行佈建。佈建是建立和設定 AWS IoT 資源的程序，這些資源是支援 Raspberry Pi 作為 IoT 裝置的必要資源。

在 Raspberry Pi 開啟電源和重新啟動之後，請在本機主機電腦上的終端機視窗中連接至 Raspberry Pi 並完成這些程序。

本節中的程序：

- [建立並下載裝置憑證檔案](#)
- [建立 AWS IoT 資源](#)

### 建立並下載裝置憑證檔案

此程序會建立此示範的裝置憑證檔案。

為 Raspberry Pi 建立並下載裝置憑證檔案

1. 在本機主機電腦的終端機視窗中，輸入這些命令來建立裝置的裝置憑證檔案。

```
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"
```

此命令會傳回類似以下的回應。記錄 *certificateArn* 值，供之後使用。

```
{
  "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-----
\nMIIIEowIBAAKCAQE_SHORTENED_FOR_EXAMPLE_T9RoDiukY\n-----END RSA PRIVATE KEY-----\n"
  }
}
```

2. 輸入下列命令來設定憑證目錄及其檔案的許可。

```
chmod 745 ~
chmod 700 ~/certs/testconn
chmod 644 ~/certs/testconn/*
chmod 600 ~/certs/testconn/private.pem.key
```

3. 執行此命令來檢閱憑證目錄和檔案的許可。

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

命令的輸出應該與您在此處看到的相同，但檔案日期和時間會有所不同。

```
-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
```

此時，您已在 Raspberry Pi 上安裝了裝置憑證檔案，可以繼續進行 [the section called “建立 AWS IoT 資源”](#)。

## 建立 AWS IoT 資源

此程序會在 AWS IoT 中佈建裝置，方法是建立裝置需要存取 AWS IoT 功能與服務的資源。

### 在 AWS IoT 中佈建裝置

1. 在本機主機電腦的終端機視窗中輸入下列命令，取得 AWS 帳戶 裝置資料端點的地址。

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

先前步驟的命令會傳回類似以下的回應：記錄 *endpointAddress* 值，供之後使用。

```
{
  "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. 輸入此命令來為 Raspberry Pi 建立 AWS IoT 物件資源。

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

如果 AWS IoT 物件資源已建立，此命令會傳回類似這樣的回應。

```
{
  "thingName": "DevCliTestThing",
  "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/DevCliTestThing",
  "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. 在終端機視窗中：
  - a. 開啟文字編輯器，例如 nano。
  - b. 複製此 JSON 政策文件並將其貼入開啟的文字編輯器中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
```

```

        "iot:Receive",
        "iot:Connect"
    ],
    "Resource": [
        "*"
    ]
}
]
}

```

#### Note

本政策文件會慷慨地授予每個資源許可，以便進行連線、接收、發佈和訂閱。通常，政策只會授予特定資源許可，以便執行特定動作。不過，對於初始裝置連線能力測試，這個過度寬鬆的政策能夠在此測試期間盡力降低出現存取問題的機率。在後續教學課程中，將使用範圍較窄的政策文件來示範政策設計的更佳實務。

- c. 將文字編輯器中的檔案儲存為 `~/policies/dev_cli_test_thing_policy.json`。
4. 執行此命令，以使用先前步驟的政策文件來建立 AWS IoT 政策。

```

aws iot create-policy \
--policy-name "DevCliTestThingPolicy" \
--policy-document "file:///~/policies/dev_cli_test_thing_policy.json"

```

如果建立此政策，此命令會傳回類似以下的回應。

```

{
  "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  }",
  "policyVersionId": "1"
}

```

5. 執行此命令，將此政策連接至裝置憑證。使用之前儲存的 `certificateArn` 值來取代 *certificateArn*

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

如果成功，此命令不會傳回任何內容。

- 執行此命令，將裝置憑證連接至 AWS IoT 物件資源。使用之前儲存的 `certificateArn` 值來取代 `certificateArn`

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

如果成功，此命令不會傳回任何內容。

成功在 AWS IoT 中佈建裝置之後，您就可以繼續進行 [the section called “步驟 3：設定要測試連線能力的 AWS IoT 裝置用戶端”](#)。

### 步驟 3：設定要測試連線能力的 AWS IoT 裝置用戶端

本節中的程序會設定 AWS IoT 裝置用戶端來從 Raspberry Pi 發佈 MQTT 訊息。

本節中的程序：

- [建立組態檔](#)
- [開啟 MQTT 測試用戶端](#)
- [執行 AWS IoT 裝置用戶端](#)

#### 建立組態檔

此程序會建立組態檔來測試 AWS IoT 裝置用戶端。

#### 建立組態檔來測試 AWS IoT 裝置用戶端

- 在連接至 Raspberry Pi 本機主機電腦上的終端機視窗中：
  - a. 輸入下列命令來建立組態檔的目錄，並設定目錄的許可：

```
mkdir ~/dc-configs
```



```
chmod 745 ~/dc-configs
```

- b. 開啟文字編輯器，例如 nano。
- c. 複製此 JSON 文件並將其貼入開啟的文字編輯器中。

```
{
  "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. 使用在 [the section called “在 AWS IoT Core 上佈建裝置”](#) 中找到的 AWS 帳戶 裝置資料端點 取代##值。
- e. 將文字編輯器中的檔案儲存為 `~/dc-configs/dc-testconn-config.json`。
- f. 執行此命令來設定新組態檔的許可。

```
chmod 644 ~/dc-configs/dc-testconn-config.json
```

儲存檔案之後，您就可以繼續進行 [the section called “開啟 MQTT 測試用戶端”](#)。

## 開啟 MQTT 測試用戶端

此程序會準備 AWS IoT 主控台中的 MQTT test client (MQTT 測試用戶端) 來訂閱 MQTT 訊息，AWS IoT 裝置用戶端會在執行時發佈訊息。

準備 MQTT test client (MQTT 測試用戶端) 來訂閱所有 MQTT 訊息

1. 在 [AWS IoT 主控台](#) 的本機主機電腦上，選擇 MQTT test client (MQTT 測試用戶端)。
2. 在 Subscribe to a topic (訂閱主題) 索引標籤的 Topic filter (主題篩選條件) 中，輸入 # (單個井字符號)，然後選擇 Subscribe (訂閱)，即可訂閱每個 MQTT 主題。
3. 在 Subscriptions (訂閱) 標籤下方，請確認看到 # (單個井字符號)。

在繼續進行 [the section called “執行 AWS IoT 裝置用戶端”](#) 時，請讓有 MQTT test client (MQTT 測試用戶端) 的視窗保持開啟。

## 執行 AWS IoT 裝置用戶端

此程序會執行 AWS IoT 裝置用戶端，使其發佈 MQTT test client (MQTT 測試用戶端) 可接收和顯示的單則 MQTT 訊息。

從 AWS IoT 裝置用戶端傳送 MQTT 訊息

1. 請確保這兩個連接到 Raspberry Pi 的終端機視窗和有 MQTT test client (MQTT 測試用戶端) 的視窗會在您執行此程序時顯示出來。
2. 在終端機視窗中輸入這些命令，使用在 [the section called “建立組態檔”](#) 中建立的組態檔來執行 AWS IoT 裝置用戶端。

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-testconn-config.json
```

在終端機視窗中，AWS IoT 裝置用戶端會顯示資訊訊息和執行時所發生的任何錯誤。

如果終端機視窗中未顯示任何錯誤，請檢閱 MQTT test client (MQTT 測試用戶端)。

3. 在 MQTT test client (MQTT 測試用戶端) 的 Subscriptions (訂閱) 視窗中，請參閱傳送至 test/dc/pubtopic 訊息主題的 Hello World! 訊息。
4. 如果 AWS IoT 裝置用戶端未顯示任何錯誤，您會在 MQTT test client (MQTT 測試用戶端) 中看到傳送至 test/dc/pubtopic 訊息的 Hello World!，這表示您已經示範了一次成功的連線。
5. 在終端機視窗中輸入 **^C** (Ctrl-C)，即可停止 AWS IoT 裝置用戶端。

在示範 AWS IoT 裝置用戶端正在 Raspberry Pi 上正常運作並可以與 AWS IoT 通訊之後，您就可以繼續進行 [the section called “示範 MQTT 訊息與 AWS IoT 裝置用戶端通訊”](#)。

## 教學課程：示範 MQTT 訊息與 AWS IoT 裝置用戶端通訊

本教學課程會示範 AWS IoT 裝置用戶端訂閱和發佈 MQTT 訊息的方式，這些訊息通常用於 IoT 解決方案。

若要開始此教學課程：

- 將本機主機電腦和 Raspberry Pi 設定成 [上一節](#) 中的使用方式。

如果在安裝 AWS IoT 裝置用戶端後儲存了 microSD 卡映像，您可以將帶有該映像的 microSD 卡搭配 Raspberry Pi 一起使用。

- 如果您曾經執行過此示範，請檢閱 [???](#) 來刪除之前執行中的所有 AWS IoT 資源，以避免重複的資源錯誤。

此教學課程約需 45 分鐘方能完成。

完成此主題時：

- 您會示範 IoT 裝置向 AWS IoT 訂閱 MQTT 訊息並將 MQTT 訊息發佈至 AWS IoT 的不同方式。

必要設備：

- [上一節](#)中的本機開發和測試環境
- [上一節](#)中使用的 Raspberry Pi
- [上一節](#)中使用的 Raspberry Pi microSD 記憶卡

本教學課程中的程序

- [步驟 1：準備 Raspberry Pi 來示範 MQTT 訊息通訊](#)
- [步驟 2：示範使用 AWS IoT 裝置用戶端發佈訊息](#)
- [步驟 3：示範使用 AWS IoT 裝置用戶端來訂閱訊息](#)

## 步驟 1：準備 Raspberry Pi 來示範 MQTT 訊息通訊

此程序會在 AWS IoT 和 Raspberry Pi 中建立資源，以此示範使用 AWS IoT 裝置用戶端進行 MQTT 訊息通訊。

本節中的程序：

- [建立憑證檔案來示範 MQTT 通訊](#)
- [佈建裝置來示範 MQTT 通訊](#)
- [設定 AWS IoT 裝置用戶端組態檔和 MQTT 測試用戶端來示範 MQTT 通訊](#)

### 建立憑證檔案來示範 MQTT 通訊

此程序會建立此示範的裝置憑證檔案。

為 Raspberry Pi 建立並下載裝置憑證檔案

1. 在本機主機電腦的終端機視窗中輸入下列命令，為裝置建立裝置憑證檔案。

```
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"
```

此命令會傳回類似以下的回應。儲存 *certificateArn* 值，供之後使用。

```
{
  "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. 輸入下列命令來設定憑證目錄及其檔案的許可。

```
chmod 700 ~/certs/pubsub
chmod 644 ~/certs/pubsub/*
chmod 600 ~/certs/pubsub/private.pem.key
```

3. 執行此命令來檢閱憑證目錄和檔案的許可。

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

命令的輸出應該與您在此處看到的相同，但檔案日期和時間會有所不同。

```
-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. 輸入這些命令來建立記錄檔案的目錄。

```
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/*
```

## 佈建裝置來示範 MQTT 通訊

本節會建立能在 AWS IoT 中佈建 Raspberry Pi 的 AWS IoT 資源。

若要在 AWS IoT 中佈建裝置：

1. 在本機主機電腦的終端機視窗中輸入下列命令，取得 AWS 帳戶 裝置資料端點的地址。

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

自您依照上一個教學課程執行此命令後，端點值便未發生變更。此處再次執行命令是為了輕鬆找到資料端點值並將其貼入本教學課程中使用的組態檔。

先前步驟的命令會傳回類似以下的回應：記錄 *endpointAddress* 值，供之後使用。

```
{
  "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. 輸入此命令來為 Raspberry Pi 建立新的 AWS IoT 物件資源。

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

由於 AWS IoT 物件資源是雲端裝置的虛擬表示，所以我們可以在 AWS IoT 中建立多個物件資源並將其投入不同用途。這些資源都可以由同一個實體 IoT 裝置使用，用以代表裝置的不同面向。

這些教學課程一次只會使用一個物件資源來表示 Raspberry Pi。如此一來，在這些教學課程中，它們代表了不同的示範，以便在為演示建立 AWS IoT 資源之後，您可以返回並使用專門為每個示範建立的資源重複示範。

如果 AWS IoT 物件資源已建立，此命令會傳回類似這樣的回應。

```
{
  "thingName": "PubSubTestThing",
  "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/PubSubTestThing",
  "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

### 3. 在終端機視窗中：

- a. 開啟文字編輯器，例如 nano。
- b. 複製此 JSON 文件並將其貼入開啟的文字編輯器中。

```
{
  "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. 在編輯器中，政策文件的每一個 Resource 區段中，使用 AWS 區域來取代 *us-west-2:57EXAMPLE833*、一個冒號字元 (:) 和 12 位數的 AWS 帳戶編號。
  - d. 將文字編輯器中的檔案儲存為 `~/policies/pubsub_test_thing_policy.json`。
4. 執行此命令，以使用先前步驟的政策文件來建立 AWS IoT 政策。

```

aws iot create-policy \
--policy-name "PubSubTestThingPolicy" \
--policy-document "file://~/policies/pubsub_test_thing_policy.json"

```

如果建立此政策，此命令會傳回類似以下的回應。

```

{
  "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}",
  "policyVersionId": "1"
}

```



5. 執行此命令，將此政策連接至裝置憑證。使用之前在本節儲存的 `certificateArn` 值來取代 `certificateArn`

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

如果成功，此命令不會傳回任何內容。

6. 執行此命令，將裝置憑證連接至 AWS IoT 物件資源。使用之前在本節儲存的 `certificateArn` 值來取代 `certificateArn`

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

如果成功，此命令不會傳回任何內容。

成功在 AWS IoT 中佈建裝置之後，您就可以繼續進行 [the section called “設定 AWS IoT 裝置用戶端組態檔和 MQTT 測試用戶端來示範 MQTT 通訊”](#)。

設定 AWS IoT 裝置用戶端組態檔和 MQTT 測試用戶端來示範 MQTT 通訊

此程序會建立組態檔來測試 AWS IoT 裝置用戶端。

建立組態檔來測試 AWS IoT 裝置用戶端

1. 在連接至 Raspberry Pi 本機主機電腦上的終端機視窗中：
  - a. 開啟文字編輯器，例如 nano。
  - b. 複製此 JSON 文件並將其貼入開啟的文字編輯器中。

```
{  
  "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. 使用在 [the section called “在 AWS IoT Core 上佈建裝置”](#) 中找到的 AWS 帳戶 裝置資料端點 取代##值。

- d. 將文字編輯器中的檔案儲存為 `~/dc-configs/dc-pubsub-config.json`。
- e. 執行此命令來設定新組態檔的許可。

```
chmod 644 ~/dc-configs/dc-pubsub-config.json
```

2. 若要準備 MQTT test client (MQTT 測試用戶端) 來訂閱所有 MQTT 訊息：
  - a. 在 [AWS IoT 主控台](#) 的本機主機電腦上，選擇 MQTT test client (MQTT 測試用戶端)。
  - b. 在 Subscribe to a topic (訂閱主題) 索引標籤的 Topic filter (主題篩選條件) 中，輸入 # (單個井字符號)，然後選擇 Subscribe (訂閱)。
  - c. 在 Subscriptions (訂閱) 標籤下方，請確認看到 # (單個井字符號)。

在繼續進行此教學課程時，請讓有 MQTT test client (MQTT 測試用戶端) 的視窗保持開啟。

在儲存檔案並設定 MQTT test client (MQTT 測試用戶端) 之後，您就可以繼續進行 [the section called “步驟 2：示範使用 AWS IoT 裝置用戶端發佈訊息”](#)。

## 步驟 2：示範使用 AWS IoT 裝置用戶端發佈訊息

本節中的程序將示範 AWS IoT 裝置用戶端傳送預設和自訂 MQTT 訊息的方式。

在上一個步驟為這些練習建立的此政策中的政策陳述式，會授予 Raspberry Pi 執行這些動作的許可：

### • **iot:Connect**

提供名為 PubSubTestThing 的用戶端、執行 AWS IoT 裝置用戶端的 Raspberry Pi 來進行連線。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Connect"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing"
  ]
}
```

### • **iot:Publish**

授予 Raspberry Pi 許可來發佈擁有 test/dc/pubtopic MQTT 主題的訊息。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Publish"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic"
  ]
}
```

此 `iot:Publish` 動作會授予許可來發佈至資源陣列中列出的 MQTT 主題。這些訊息的內容不受政策陳述式的控制。

### 使用 AWS IoT 裝置用戶端發佈預設訊息

此程序會執行 AWS IoT 裝置用戶端，使其發佈 MQTT test client (MQTT 測試用戶端) 可接收和顯示的單則預設 MQTT 訊息。

### 從 AWS IoT 裝置用戶端傳送預設 MQTT 訊息

1. 請確保這兩個連接到 Raspberry Pi 的本機主機電腦終端機視窗和有 MQTT test client (MQTT 測試用戶端) 的視窗，會在您執行此程序時顯示出來。
2. 在終端機視窗中輸入這些命令，使用在 [the section called “建立組態檔”](#) 中建立的組態檔來執行 AWS IoT 裝置用戶端。

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-config.json
```

在終端機視窗中，AWS IoT 裝置用戶端會顯示資訊訊息和執行時所發生的任何錯誤。

如果終端機視窗中未顯示任何錯誤，請檢閱 MQTT test client (MQTT 測試用戶端)。

3. 在 MQTT test client (MQTT 測試用戶端) 的 Subscriptions (訂閱) 視窗中，請參閱傳送至 `test/dc/pubtopic` 訊息主題的 Hello World! 訊息。
4. 如果 AWS IoT 裝置用戶端未顯示任何錯誤，您會在 MQTT test client (MQTT 測試用戶端) 中看到傳送至 `test/dc/pubtopic` 訊息的 Hello World!，這表示您已經示範了一次成功的連線。
5. 在終端機視窗中輸入 `^C` (Ctrl-C)，即可停止 AWS IoT 裝置用戶端。

在示範 AWS IoT 裝置用戶端發佈了預設 MQTT 訊息之後，您就可以繼續進行 [the section called “使用 AWS IoT 裝置用戶端來發佈自訂訊息”](#)。

### 使用 AWS IoT 裝置用戶端來發佈自訂訊息

本節中的程序會建立自訂 MQTT 訊息，然後執行 AWS IoT 裝置用戶端，以便它發佈一次自訂 MQTT 訊息，供 MQTT test client (MQTT 測試用戶端) 接收和顯示。

### 為 AWS IoT 裝置用戶端建立自訂 MQTT 訊息

在連接至 Raspberry Pi 本機主機電腦上的終端機視窗中執行這些步驟。

### 建立可讓 AWS IoT 裝置用戶端發佈的自訂訊息

1. 在終端機視窗中，開啟文字編輯器，例如 nano。
2. 在文字編輯器中複製並貼入以下 JSON 文件。這會是 AWS IoT 裝置用戶端發佈的 MQTT 訊息承載。

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

3. 將文字編輯器的內容儲存為 `~/messages/sample-ws-message.json`。
4. 輸入下列命令，為剛建立的訊息檔案設定許可。

```
chmod 600 ~/messages/*
```

### 為 AWS IoT 裝置用戶端建立用來傳送自訂訊息的組態檔

1. 在終端機視窗的文字編輯器 (例如 nano) 中，開啟現有 AWS IoT 裝置用戶端的組態檔：`~/dc-configs/dc-pubsub-config.json`。
2. 按如下所示編輯 `samples` 物件。無需變更此檔案的其他部分。

```
"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. 將文字編輯器的內容儲存為 `~/dc-configs/dc-pubsub-custom-config.json`。
4. 執行此命令來設定新組態檔的許可。

```
chmod 644 ~/dc-configs/dc-pubsub-custom-config.json
```

### 使用 AWS IoT 裝置用戶端發佈自訂 MQTT 訊息

此項變更只會影響 MQTT 訊息承載的內容，因此目前的政策會繼續運作。不過，如果 MQTT 主題 (如 `~/dc-configs/dc-pubsub-custom-config.json` 中定義的 `publish-topic` 值) 已變更，`iot::Publish` 政策陳述式也需要進行修改，以允許 Raspberry Pi 發佈至新的 MQTT 主題。

### 從 AWS IoT 裝置用戶端傳送 MQTT 訊息

1. 請確保這兩個終端機視窗和有 MQTT test client (MQTT 測試用戶端) 的視窗，會在您執行此程序時顯示出來。此外，請確保 MQTT test client (MQTT 測試用戶端) 仍然訂閱了 # 主題篩選條件。如果未訂閱，請訂閱 # 主題篩選條件。
2. 在終端機視窗中輸入這些命令，使用在 [the section called “建立組態檔”](#) 中建立的組態檔來執行 AWS IoT 裝置用戶端。

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
```

在終端機視窗中，AWS IoT 裝置用戶端會顯示資訊訊息和執行時所發生的任何錯誤。

如果終端機視窗中未顯示任何錯誤，請檢閱 MQTT test client (MQTT 測試用戶端)。

3. 在 MQTT test client (MQTT 測試用戶端) 的 Subscriptions (訂閱) 視窗中，參閱傳送至 `test/dc/pubtopic` 訊息主題的自訂訊息承載。
4. 如果 AWS IoT 裝置用戶端不會顯示任何錯誤，而且您在 MQTT test client (MQTT 測試用戶端) 看到發佈至 `test/dc/pubtopic` 的自訂訊息承載，表示您已成功發佈自訂訊息。
5. 在終端機視窗中輸入 `^C` (Ctrl-C)，即可停止 AWS IoT 裝置用戶端。

在示範 AWS IoT 裝置用戶端發佈了自訂訊息承載之後，您就可以繼續進行 [the section called “步驟 3：示範使用 AWS IoT 裝置用戶端來訂閱訊息”](#)。

### 步驟 3：示範使用 AWS IoT 裝置用戶端來訂閱訊息

在本節中，您會示範兩種類型的訊息訂閱：

- 單一主題訂閱
- 萬用字元主題訂閱

為這些練習建立的此政策中的政策陳述式會授予 Raspberry Pi 執行這些動作的許可：

- **iot:Receive**

授予 AWS IoT 裝置用戶端許可來接收與 Resource 物件中名稱相符的 MQTT 主題。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/subtopic"
  ]
}
```

- **iot:Subscribe**

授予 AWS IoT 裝置用戶端許可來訂閱與 Resource 物件中名稱相符的 MQTT 主題篩選條件。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic"
  ]
}
```

## 訂閱單一 MQTT 訊息主題

此程序示範 AWS IoT 裝置用戶端訂閱和記錄 MQTT 訊息的方式。

在連接至 Raspberry Pi 本機主機電腦上的終端機視窗中，列出 `~/dc-configs/dc-pubsub-custom-config.json` 的內容或在文字編輯器中開啟檔案來檢閱其內容。找到 `samples` 物件，它應該如下所示。

```
"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"
```

請注意 `subscribe-topic` 值是 MQTT 主題，AWS IoT 裝置用戶端會在執行時訂閱該主題。此 AWS IoT 裝置用戶端會將從此訂閱接收到的訊息承載寫入 `subscribe-file` 值中指定的檔案。

### 從 AWS IoT 裝置用戶端訂閱 MQTT 訊息主題

1. 請確保這兩個終端機視窗和有 MQTT test client (MQTT 測試用戶端) 的視窗，會在您執行此程序時顯示出來。此外，請確保 MQTT test client (MQTT 測試用戶端) 仍然訂閱了 # 主題篩選條件。如果未訂閱，請訂閱 # 主題篩選條件。
2. 在終端機視窗中輸入這些命令，使用在 [the section called “建立組態檔”](#) 中建立的組態檔來執行 AWS IoT 裝置用戶端。

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-custom-config.json
```

在終端機視窗中，AWS IoT 裝置用戶端會顯示資訊訊息和執行時所發生的任何錯誤。

如果終端機視窗中未顯示任何錯誤，請在 AWS IoT 主控台中繼續進行。

3. 在 AWS IoT 主控台的 MQTT test client (MQTT 測試用戶端) 中，選擇 Publish to a topic (發佈至主題) 索引標籤。
4. 在 Topic name (主題名稱) 中輸入 `test/dc/subtopic`。
5. 在 Message payload (訊息承載) 中，檢閱郵件內容。
6. 請選擇 Publish (發佈) 來發佈 MQTT 訊息。
7. 在終端機視窗中，監看 AWS IoT 裝置用戶端的已接收訊息項目，如下所示。



```
2021-11-10T16:02:20.890Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
subscribe topic, size: 45 bytes
```

8. 在看到顯示訊息已收到的已接收訊息項目時，請輸入 **^C** (Ctrl-C) 來停止 AWS IoT 裝置用戶端。
9. 輸入此命令來檢視訊息日誌檔案的結尾，查看您從 MQTT test client (MQTT 測試用戶端) 發佈的訊息。

```
tail ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

藉由檢視日誌檔案中的訊息，您示範了 AWS IoT 裝置用戶端收到您從 MQTT 測試用戶端發佈的訊息。

### 使用萬用字元訂閱多個 MQTT 訊息主題

這些程序會示範 AWS IoT 裝置用戶端可以使用萬用字元訂閱並記錄 MQTT 訊息。若要做到這一點，您必須：

1. 更新 AWS IoT 裝置用戶端用來訂閱 MQTT 主題的主題篩選條件。
2. 更新裝置所使用的政策，以允許新的訂閱。
3. 執行 AWS IoT 裝置用戶端並從 MQTT 測試主控台發佈訊息。

### 使用萬用字元 MQTT 主題篩選條件建立用於訂閱多個 MQTT 訊息主題的組態檔

1. 在連接至 Raspberry Pi 本機主機電腦的終端機視窗中，開啟 `~/dc-configs/dc-pubsub-custom-config.json` 進行編輯，並找出 `samples` 物件。
2. 在文字編輯器中，找出 `samples` 物件並更新 `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` 值是末尾具有 MQTT 萬用字元的 [MQTT 主題篩選條件](#)。本節說明訂閱所有以 `test/dc/` 為開頭的 MQTT 主題。此 AWS IoT 裝置用戶端會將從此訂閱接收到的訊息承載寫入 `subscribe-file` 中指定的檔案。

3. 將修改的組態檔儲存為 `~/dc-configs/dc-pubsub-wild-config.json`，然後退出編輯器。

修改 Raspberry Pi 使用的政策，以允許訂閱和接收多個 MQTT 消息主題

1. 在連接至 Raspberry Pi 的本機主機電腦上的終端機視窗中，於最喜歡的文字編輯器中開啟 `~/policies/pubsub_test_thing_policy.json` 進行編輯，然後找出檔案中的 `iot::Subscribe` 和 `iot::Receive` 政策陳述式。
2. 在 `iot::Subscribe` 政策陳述式中，更新資源物件中的字串，用 `*` 取代 `subtopic`，使其如下所示。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Subscribe"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/*"
  ]
}
```

#### Note

此 [MQTT 主題篩選條件萬用字元](#) 是 `+` (加號) 和 `#` (井號)。結尾帶有 `#` 的訂閱請求會訂閱所有以 `#` 字元前面的字串為開頭的主題 (例如本例中的 `test/dc/`)。不過，政策陳述式中授權此訂閱的資源值必須在主題篩選條件 ARN 中使用 `*` (星號) 取代 `#` (井號)。這是因為政策處理器使用的萬用字元與 MQTT 使用的不同。如需有關在政策中為主題和主題篩選條件使用萬用字元的詳細資訊，請參閱 [在 MQTT 和 AWS IoT Core 政策中使用萬用字元](#)。

3. 在 `iot::Receive` 政策陳述式中，更新資源物件中的字串，用 `*` 取代 `subtopic`，使其如下所示。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:Receive"
  ],
  "Resource": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*"
  ]
}
```

```
    ]
  }
```

- 將更新後的政策文件儲存為 `~/policies/pubsub_wild_test_thing_policy.json`，然後退出編輯器。
- 輸入此指令來更新此教學課程的政策，以使用新的資源定義。

```
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，它會傳回類似這樣的回應。請注意，`policyVersionId` 現在是 2，表示這是此政策的第二個版本。

如果成功更新了政策，您可以繼續進行下一個程序。

```
{
  "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}",
  "policyVersionId": "2",
  "isDefaultVersion": true
}
```

如果收到因政策版本過多而無法儲存新政策的錯誤，請輸入此命令來列出政策的目前版本。檢閱此命令傳回的清單，找出可以刪除的政策版本。

```
aws iot list-policy-versions --policy-name "PubSubTestThingPolicy"
```

輸入此命令來刪除不再需要的版本。請注意，您無法刪除預設政策版本。預設政策版本的 `isDefaultVersion` 值為 `true`。

```
aws iot delete-policy-version \  
--policy-name "PubSubTestThingPolicy" \  
--policy-version-id policyId
```

刪除政策版本後，請重試此步驟。

更新了組態檔和政策後，您就可以使用 AWS IoT 裝置用戶端來示範萬用字元訂閱。

示範 AWS IoT 裝置用戶端訂閱並接收多個 MQTT 訊息主題的方式

1. 在 MQTT test client (MQTT 測試用戶端) 中，檢查訂閱。如果 MQTT test client (MQTT 測試用戶端) 訂閱了 # 主題篩選條件，請繼續下一個步驟。如果沒有，進入 MQTT test client (MQTT 測試用戶端)，在 Subscribe to a topic (訂閱主題) 索引標籤的 Topic filter (主題篩選條件) 中，輸入 # (井號字元)，然後選擇 Subscribe (訂閱) 進行訂閱。
2. 在連接至 Raspberry Pi 的本機主機電腦終端機視窗中，輸入這些命令來啟動 AWS IoT 裝置用戶端。

```
cd ~/aws-iot-device-client/build  
./aws-iot-device-client --config-file ~/dc-configs/dc-pubsub-wild-config.json
```

3. 在觀看 AWS IoT 裝置用戶端在本機主機電腦的終端機視窗中輸出的同時，返回至 MQTT test client (MQTT 測試用戶端)。在 Publish to a topic (發佈至主題) 索引標籤中的 Topic name (主題名稱) 輸入 `test/dc/subtopic`，然後選擇 Publish (發佈)。
4. 在終端機視窗中，尋找如下訊息來確認已收到訊息：

```
2021-11-10T16:34:20.101Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on  
subscribe topic, size: 76 bytes
```

5. 在觀看 AWS IoT 裝置用戶端在本機主機電腦的終端機視窗中輸出的同時，返回至 MQTT test client (MQTT 測試用戶端)。在 Publish to a topic (發佈至主題) 索引標籤中的 Topic name (主題名稱) 輸入 `test/dc/subtopic2`，然後選擇 Publish (發佈)。
6. 在終端機視窗中，尋找如下訊息來確認已收到訊息：

```
2021-11-10T16:34:32.078Z [DEBUG] {samples/PubSubFeature.cpp}: Message received on
subscribe topic, size: 77 bytes
```

7. 在您看到確認已接收到兩則訊息的訊息之後，請輸入 **^C** (Ctrl-C) 來停止 AWS IoT 裝置用戶端。
8. 輸入此命令來檢視訊息日誌檔案的結尾，查看您從 MQTT test client (MQTT 測試用戶端) 發佈的訊息。

```
tail -n 20 ~/.aws-iot-device-client/log/pubsub_rx_msgs.log
```

### Note

記錄檔案僅包含訊息承載。訊息主題不會記錄在接收到的訊息日誌檔案中。您也可能看到由接收日誌中的 AWS IoT 裝置用戶端所發佈的訊息。這是因為萬用字元主題篩選條件會包含該訊息主題，有時候，訂閱請求可能由訊息代理程式處理，然後才將發佈的訊息傳送給訂閱者。

日誌檔案中的項目會示範已收到訊息。您可以使用其他主題名稱重複此程序。所有主題名稱以 `test/dc/` 為開頭的訊息應該會被接收並記錄。主題名稱以任何其他文字為開頭的訊息都會被忽略。

在示範 AWS IoT 裝置用戶端發佈和訂閱 MQTT 訊息的方式之後，請繼續進行 [教學課程：使用 AWS IoT 裝置用戶端來示範遠端動作 \(任務\)](#)。

## 教學課程：使用 AWS IoT 裝置用戶端來示範遠端動作 (任務)

在這些教學課程中，您要設定並部署任務至 Raspberry Pi，以此示範如何將遠程操作發送至 IoT 裝置。

若要開始此教學課程：

- 將本機主機電腦和 Raspberry Pi 設定成 [上一節](#) 中的使用方式。
- 如果還未完成前一節中的教學課程，您可以嘗試使用帶有 microSD 卡的 Raspberry Pi，該記憶卡中有您在 [\(選用\) 儲存 microSD 卡映像](#) 安裝 AWS IoT 裝置用戶端後儲存的映像。
- 如果您曾經執行過此示範，請檢閱 [???](#) 來刪除之前執行中的所有 AWS IoT 資源，以避免重複的資源錯誤。

此教學課程約需 45 分鐘方能完成。

完成此主題時：

- 您要示範 IoT 裝置使用 AWS IoT Core 的不同方式來執行由 AWS IoT 管理的遠端操作。

必要設備：

- [上一節](#)中測試的本機開發和測試環境
- [上一節](#)中測試的 Raspberry Pi
- [上一節](#)中測試的 Raspberry Pi microSD 記憶卡

本教學課程中的程序

- [步驟 1：準備 Raspberry Pi 來執行任務](#)
- [步驟 2：在 AWS IoT 中建立並執行任務](#)

## 步驟 1：準備 Raspberry Pi 來執行任務

本節中的程序會描述如何準備好 Raspberry Pi 來使用 AWS IoT 裝置用戶端執行任務。

### Note

這些程序是裝置特定的程序。如果想要同時使用多個裝置執行本節中的程序，每個裝置都需要有自己的政策和唯一的裝置特定憑證及物件名稱。若要為每個裝置提供獨特資源，請在按程序所述變更裝置特定元素時，針對每個裝置執行此程序一次。

本教學課程中的程序

- [佈建 Raspberry Pi 來示範任務](#)
- [設定 AWS IoT 裝置用戶端來執行任務代理程式](#)

## 佈建 Raspberry Pi 來示範任務

在本節中的程序會透過建立 AWS IoT 資源和其裝置憑證在 AWS IoT 中佈建 Raspberry Pi。

建立並下載裝置憑證檔案來示範 AWS IoT 任務

此程序會建立此示範的裝置憑證檔案。

如果要準備一個以上的裝置，則必須在每個裝置上執行此程序。

若要為 Raspberry Pi 建立並下載裝置憑證檔案：

在連接至 Raspberry Pi 的本機主機電腦終端機視窗中，輸入這些命令。

1. 請輸入下列命令，為裝置建立裝置憑證檔案。

```
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"
```

此命令會傳回類似以下的回應。儲存 *certificateArn* 值，供之後使用。

```
{  
  "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. 輸入下列命令來設定憑證目錄及其檔案的許可。

```
chmod 700 ~/certs/jobs  
chmod 644 ~/certs/jobs/*  
chmod 600 ~/certs/jobs/private.pem.key
```

3. 執行此命令來檢閱憑證目錄和檔案的許可。

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

命令的輸出應該與您在此處看到的相同，但檔案日期和時間會有所不同。

```
-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
```

在下載裝置憑證檔案至 Raspberry Pi 之後，您就可以繼續進行 [the section called “佈建 Raspberry Pi 來示範任務”](#)。

建立要示範 AWS IoT 任務的 AWS IoT 資源

建立此裝置的 AWS IoT 資源。

如果要準備一個以上的裝置，則必須為每個裝置上執行此程序。

若要在 AWS IoT 中佈建裝置：

在連接至 Raspberry Pi 本機主機電腦上的終端機視窗中：

1. 輸入下列命令，取得 AWS 帳戶 裝置資料端點地址。

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

自您上一次執行此命令後，端點值便未發生變更。此處再次執行命令是為了輕鬆找出資料端點值並貼入本教學課程中使用的組態檔。

describe-endpoint 命令會傳回類似以下的回應。記錄 *endpointAddress* 值，供之後使用。

```
{
  "endpointAddress": "a3qjEXAMPLEffp-ats.iot.us-west-2.amazonaws.com"
}
```

2. 使用裝置的唯一名稱來取代 *uniqueThingName*。如果想要使用多個裝置執行本教學課程，請賦予每個裝置自己的名稱。例如 **TestDevice01**、**TestDevice02** 等等。

輸入此命令來為 Raspberry Pi 建立新的 AWS IoT 物件資源。

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



由於 AWS IoT 物件資源是雲端裝置的虛擬表示，所以我們可以在 AWS IoT 中建立多個物件資源並將其投入不同用途。這些資源都可以由同一個實體 IoT 裝置使用，用以代表裝置的不同面向。

#### Note

當您想要保護多個裝置的政策時，可以使用 `${iot:Thing.ThingName}` 來取代靜態物件名稱 *uniqueThingName*。

這些教學課程每次只會使用一個物件資源。如此一來，在這些教學課程中，它們代表了不同的示範，以便在為演示建立 AWS IoT 資源之後，您可以返回並使用專門為每個示範建立的資源重複示範。

如果 AWS IoT 物件資源已建立，此命令會傳回類似這樣的回應。記錄 *thingArn* 值以供稍後在此裝置上建立要執行的任務時使用。

```
{
  "thingName": "uniqueThingName",
  "thingArn": "arn:aws:iot:us-west-2:57EXAMPLE833:thing/uniqueThingName",
  "thingId": "8ea78707-32c3-4f8a-9232-14bEXAMPLEfd"
}
```

3. 在終端機視窗中：
  - a. 開啟文字編輯器，例如 nano。
  - b. 複製此 JSON 文件並將其貼入開啟的文字編輯器中。

```
{
  "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. 在編輯器的每個政策陳述式 Resource 區段中，使用 AWS 區域 來取代 *us-west-2:57EXAMPLE833*、一個冒號字元 (:) 和 12 位數的 AWS 帳戶 編號。
- d. 在編輯器的在每個政策陳述式中，使用您賦予此物件資源的物件名稱來取代 *uniqueThingName*。
- e. 將文字編輯器中的檔案儲存為 `~/policies/jobs_test_thing_policy.json`。

如果為多個裝置執行此程序，請在每個裝置上將檔案儲存為此檔案名稱。

4. 使用裝置的物間名稱來取代 *uniqueThingName*，然後執行此命令建立為該裝置量身訂製的 AWS IoT 政策。

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

如果建立此政策，此命令會傳回類似以下的回應。

```
{
  "policyName": "JobTestPolicyForuniqueThingName",
  "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/JobTestPolicyForuniqueThingName",
  "policyDocument": "{\n\"Version\": \"2012-10-17\", \n\"Statement\": [\n{\n\n\"Effect\": \"Allow\", \n\"Action\": [\n\"iot:Connect\"\n], \n\"Resource\": [\n\n\"arn:aws:iot:us-west-2:57EXAMPLE833:client/PubSubTestThing\"\n]\n}, \n{\n\n\"Effect\": \"Allow\", \n\"Action\": [\n\"iot:Publish\"\n], \n\"Resource\": [\n\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/pubtopic\"\n]\n}, \n{\n\n\"Effect\": \"Allow\", \n\"Action\": [\n\"iot:Subscribe\"\n], \n\"Resource\": [\n\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topicfilter/test/dc/subtopic\"\n]\n}, \n{\n\n\"Effect\": \"Allow\", \n\"Action\": [\n\"iot:Receive\"\n], \n\"Resource\": [\n\n\"arn:aws:iot:us-west-2:57EXAMPLE833:topic/test/dc/*\"\n]\n}\n]\n}\n",
  "policyVersionId": "1"
```

5. 使用裝置的物件名稱來取代 *uniqueThingName*，再使用稍早您在本節為此裝置儲存的 `certificateArn` 值來取代 *certificateArn*，然後執行此命令來將政策連接至裝置憑證。

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

如果成功，此命令不會傳回任何內容。

6. 使用裝置的物件名稱來取代 *uniqueThingName*，再使用稍早您在本節儲存的 *certificateArn* 值來取代 *certificateArn*，然後執行此命令來將裝置憑證連接至 AWS IoT 物件資源。

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

如果成功，此命令不會傳回任何內容。

在成功佈建 Raspberry Pi 之後，您就可以在測試中為另一 Raspberry Pi 重複本節程序；或者，如果所有裝置都已佈建，則繼續進行 [the section called “設定 AWS IoT 裝置用戶端來執行任務代理程式”](#)。

設定 AWS IoT 裝置用戶端來執行任務代理程式

此程序會建立組態檔，讓 AWS IoT 裝置用戶端執行任務代理程式：

注意：如果要準備一個以上的裝置，則必須在每個裝置上執行此程序。

若要建立組態檔來測試 AWS IoT 裝置用戶端：

1. 在連接至 Raspberry Pi 本機主機電腦上的終端機視窗中：
  - a. 開啟文字編輯器，例如 nano。
  - b. 複製此 JSON 文件並將其貼入開啟的文字編輯器中。

```
{  
  "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. 使用在 [the section called “在 AWS IoT Core 上佈建裝置”](#) 中找到的 AWS 帳戶 裝置資料端點值取代##值。

- d. 用此裝置使用的物件名稱取代 *uniqueThingName*。
  - e. 將文字編輯器中的檔案儲存為 `~/dc-configs/dc-jobs-config.json`。
2. 執行此命令來設定新組態檔的檔案許可。

```
chmod 644 ~/dc-configs/dc-jobs-config.json
```

您不必使用 MQTT test client (MQTT 測試用戶端) 進行此測試。雖然裝置會與 AWS IoT 交換任務相關的 MQTT 訊息，但任務進度訊息只會與執行任務的裝置交換。由於任務進度訊息只會與執行任務的裝置交換，因此您無法從其他裝置 (例如 AWS IoT 主控台) 訂閱訊息。

儲存組態檔之後，您就可以繼續進行 [the section called “步驟 2：在 AWS IoT 中建立並執行任務”](#)。

## 步驟 2：在 AWS IoT 中建立並執行任務

本節中的程序會建立任務文件和 AWS IoT 任務資源。建立任務資源後，AWS IoT 會將任務文件傳送至特定任務目標，再由任務代理程式將任務文件套用至裝置或用戶端。

本節中的程序

- [建立並存放任務的任務文件](#)
- [為單一 IoT 裝置在 AWS IoT 中執行任務](#)

### 建立並存放任務的任務文件

此程序會建立簡單的任務文件，以包含在 AWS IoT 任務資源中。此任務文件會在任務目標上顯示 "Hello world!"。

若要建立和儲存任務文件：

1. 選取要儲存任務文件的 Amazon S3 儲存貯體。如果沒有可使用的現有 Amazon S3 儲存貯體，就需要建立一個。如需如何建立 Amazon S3 儲存貯體的詳細資訊，請參閱 [Amazon S3 入門](#) 中的主題。
2. 建立並儲存此任務的任務文件
  - a. 在本機主機電腦上，開啟文字編輯器。
  - b. 將此文字複製並貼入編輯器中。

```
{
```

```
"operation": "echo",
"args": ["Hello world!"]
}
```

- c. 在本機主機電腦上，將編輯器的內容儲存至名為 **hello-world-job.json** 的檔案。
  - d. 請確認檔案已正確儲存。某些文字編輯器會在儲存文字檔案時自動附加 `.txt` 至檔案名稱。如果編輯器附加了 `.txt` 至檔案名稱，請更正檔案名稱後再繼續。
3. 使用 **hello-world-job.json** 路徑來取代 *path\_to\_file*；如果該路徑不在當前目錄中，則使用所選儲存貯體的 Amazon S3 儲存貯體路徑取代 *s3\_bucket\_name*，然後執行此命令將任務文件放入 Amazon S3 儲存貯體中。

```
aws s3api put-object \
--key hello-world-job.json \
--body path_to_file/hello-world-job.json --bucket s3_bucket_name
```

識別儲存在 Amazon S3 中的任務文件的任務文件 URL 是透過取代以下 URL 中 *s3\_bucket\_name* 和 *AWS\_region* 來決定。記錄產生的 URL 以便稍後使用為 *job\_document\_path*

```
https://s3_bucket_name.s3.AWS_Region.amazonaws.com/hello-world-job.json
```

#### Note

AWS 安全功能會造成您無法在 AWS 帳戶 外部開啟此 URL，例如使用瀏覽器開啟。該 URL 是由 AWS IoT 任務引擎使用，依預設 URL 會有該檔案的存取權。在生產環境中，您需要確保 AWS IoT 服務具有存取儲存在 Amazon S3 中任務文件的許可。

在儲存任務文件的 URL 之後，請繼續進行 [the section called “為單一 IoT 裝置在 AWS IoT 中執行任務”](#)。

### 為單一 IoT 裝置在 AWS IoT 中執行任務

本節中的程序會在 Raspberry Pi 上啟動 AWS IoT 裝置用戶端，以在裝置上執行任務代理程式來等待任務執行。它也會在 AWS IoT 中建立一個任務資源，這會將任務傳送至 IoT 裝置並在 IoT 裝置上執行。

**Note**

此程序只會在單一裝置上執行任務。

若要啟動 Raspberry Pi 任務代理程式：

1. 在連接至 Raspberry Pi 的本機主機電腦終端機視窗中，輸入這些命令來啟動 AWS IoT 裝置用戶端。

```
cd ~/aws-iot-device-client/build
./aws-iot-device-client --config-file ~/dc-configs/dc-jobs-config.json
```

2. 在終端機視窗中確認 AWS IoT 裝置用戶端並顯示這些訊息

```
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. 在終端機視窗中看到此訊息之後，請繼續執行下一個程序並建立任務資源。請注意，它可能不是清單中的最後一個項目。

```
2021-11-15T18:45:56.785Z [INFO] {JobsFeature.cpp}: No pending jobs are scheduled,
  waiting for the next incoming job
```

## 建立 AWS IoT 任務資源

1. 在本機主機電腦上：
  - a. 使用來自 [the section called “建立並存放任務的任務文件”](#) 的任務文件 URL 來取代 *job\_document\_url*。
  - b. 使用為裝置建立的物件資源 ARN 來取代 *thing\_arn*，然後執行此命令。

```
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，命令會傳回類似以下的結果。

```
{
  "jobArn": "arn:aws:iot:us-west-2:57EXAMPLE833:job/hello-world-job-1",
  "jobId": "hello-world-job-1"
}
```

2. 在終端機視窗中，您應該會看到這樣的 AWS IoT 裝置用戶端輸出。

```
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 EphemeralPromise 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 EphemeralPromise 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. 在 AWS IoT 裝置用戶端正在執行並等待任務的同時，您也可以透過變更 `job-id` 值並重新執行步驟 1 的 `create-job` 來提交另一個任務。

完成執行任務後，在終端機視窗中輸入 `^C` (control-C) 來停止 AWS IoT 裝置用戶端。

## 教學課程：在執行 AWS IoT 裝置用戶端教學課程後清除

本教學課程中的程序會在完成此學習路徑中的教學課程時，逐步引導您移除所建立的檔案和資源。

本教學課程中的程序

- [步驟 1：在使用 AWS IoT 裝置用戶端建置示範後清除裝置](#)
- [步驟 2：在使用 AWS IoT 裝置用戶端建置示範後清除 AWS 帳戶](#)

### 步驟 1：在使用 AWS IoT 裝置用戶端建置示範後清除裝置

本教學課程說明在此學習路徑中建置示範後，如何清除 microSD 卡的兩個選項。選擇提供所需安全等級的選項。

請注意，清理裝置的 microSD 卡並不會移除任何 AWS IoT 資源。若要在清理裝置的 microSD 卡後清除 AWS IoT 資源，請參閱 [the section called “在使用 AWS IoT 裝置用戶端建置示範後清除”](#) 上的教學課程。

選項 1：透過重新寫入 microSD 記憶卡進行清除

在完成此學習路徑中的教學課程後，清除 microSD 卡最簡單且最徹底的方法是，使用您在第一次準備裝置時所建立的儲存映像檔案覆寫 microSD 卡。

此程序會使用本機主機電腦，將儲存的 microSD 卡映像寫入 microSD 卡。

#### Note

如果裝置未針對其作業系統使用卸除式儲存媒體，請參閱該裝置的程序。

## 若要將新映像寫入 microSD 卡

1. 在本機主機電腦上，找出要寫入至 microSD 卡的儲存 microSD 卡映像。
2. 請將 microSD 卡插入主機電腦。
3. 使用 SD 卡映像工具，將選取的影像檔案寫入至 microSD 卡。
4. 寫入 Raspberry Pi 作業系統映像至 microSD 卡後，請退出 microSD 卡並安全地將其從本機主機電腦中移除。

您的 microSD 卡已可供使用。

## 選項 2：透過刪除使用者目錄進行清除

若要在完成教學課程後清理 microSD 卡而不重寫 microSD 卡映像，您可以個別刪除使用者目錄。這並不像從儲存映像重寫 microSD 卡那麼徹底，因為此操作不會移除任何可能已安裝的系統檔案。

如果移除使用者目錄足以滿足需求，您可以遵循此程序。

### 從裝置刪除此學習路徑的使用者目錄

1. 在連接至裝置的終端機視窗中，執行這些命令來刪除在此學習路徑中建立的使用者目錄、子目錄及其所有檔案。

#### Note

在刪除這些目錄和檔案之後，您必須再次完成教學課程，才能執行示範。

```
rm -Rf ~/dc-configs
rm -Rf ~/policies
rm -Rf ~/messages
rm -Rf ~/certs
rm -Rf ~/.aws-iot-device-client
```

2. 在連接至裝置的終端機視窗中，執行這些命令來刪除應用程式來源目錄和檔案。

#### Note

這些命令不會解除安裝任何程式。它們只會刪除用於建置和安裝它們的來源檔案。刪除這些檔案之後，AWS CLI 與 AWS IoT 裝置用戶端可能無法運作。

```
rm -Rf ~/aws-cli
rm -Rf ~/aws
rm -Rf ~/aws-iot-device-client
```

## 步驟 2：在使用 AWS IoT 裝置用戶端建置示範後清除 AWS 帳戶

在完成此學習路徑中的教學課程後，這些程序可協助您識別並移除所建立的 AWS 資源。

### 清除 AWS IoT 資源

在完成此學習路徑中的教學課程後，此程序可協助您識別並移除您建立的 AWS IoT 資源。

在此學習路徑中建立的 AWS IoT 資源

教學課程	物件資源	政策資源
<a href="#">the section called “安裝並設定 AWS IoT 裝置用戶端”</a>	DevCliTestThing	DevCliTestThingPolicy
<a href="#">the section called “示範 MQTT 訊息與 AWS IoT 裝置用戶端通訊”</a>	PubSubTestThing	PubSubTestThingPolicy
<a href="#">the section called “使用 AWS IoT 裝置用戶端來示範遠端動作 (任務)”</a>	使用者定義 (可能不止一個)	使用者定義 (可能不止一個)

若要刪除 AWS IoT 資源，請為建立的每個物件資源執行此程序

1. 使用要刪除的物件資源名稱來取代 *thing\_name*，然後執行此命令從本機主機電腦列出連接至物件資源的憑證。

```
aws iot list-thing-principals --thing-name thing_name
```

此命令會傳回像這樣的回應，其中會列出連接至 *thing\_name* 的憑證。在大多數情況下，清單中只會有一個憑證。

```
{
  "principals": [
    "arn:aws:iot:us-west-2:57EXAMPLE833:cert/23853eea3cf0edc7f8a69c74abeafa27b2b52823cab5b3e156295e94b26ae8ac"
  ]
}
```

2. 針對上一個命令列出的每個憑證：

- a. 使用上一個命令的憑證 ID 來取代 *certificate\_ID*。憑證 ID 是上一個命令傳回的 ARN 中 cert/ 後面的英數字元。然後執行此命令來停用憑證。

```
aws iot update-certificate --new-status INACTIVE --certificate-id certificate_ID
```

如果成功，此命令不會傳回任何內容。

- b. 使用之前傳回的憑證清單中的憑證 ARN 來取代 *certificate\_ARN*，然後執行此命令來列出連接至此憑證的政策。

```
aws iot list-attached-policies --target certificate_ARN
```

此命令會傳回像這樣的回應，其中會列出連接至憑證的政策。在大多數情況下，清單中只會有一個政策。

```
{
  "policies": [
    {
      "policyName": "DevCliTestThingPolicy",
      "policyArn": "arn:aws:iot:us-west-2:57EXAMPLE833:policy/DevCliTestThingPolicy"
    }
  ]
}
```

- c. 每個連接至憑證的政策：

- i. 使用上一個命令的 policyName 值來取代 *policy\_name*，使用憑證的 ARN 來取代 *certificate\_ARN*，然後執行此命令從憑證分離政策。

```
aws iot detach-policy --policy-name policy_name --target certificate_ARN
```

如果成功，此命令不會傳回任何內容。

- ii. 使用 `policyName` 值來取代 *policy\_name*，然後執行此命令，查看政策是否已連接至任何憑證。

```
aws iot list-targets-for-policy --policy-name policy_name
```

如果命令傳回類似這樣的空白清單，代表政策未連接至任何憑證，您可以繼續列出政策版本。如果仍有連接至政策的憑證，請繼續進行 `detach-thing-principal` 步驟。

```
{
  "targets": []
}
```

- iii. 使用 `policyName` 值來取代 *policy\_name*，然後執行此命令來檢查政策版本。若要刪除政策，該政策必須只有一個版本。

```
aws iot list-policy-versions --policy-name policy_name
```

如果政策只有一個版本 (例如此範例)，您可以跳至 `delete-policy` 步驟，然後立即刪除政策。

```
{
  "policyVersions": [
    {
      "versionId": "1",
      "isDefaultVersion": true,
      "createDate": "2021-11-18T01:02:46.778000+00:00"
    }
  ]
}
```

如果政策具有多個版本 (例如此範例)，就必須先刪除具有 `isDefaultVersion` 值的 `false`，才能刪除政策。

```
{
  "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"
}
]
```

如果需要刪除政策版本，請使用 `policyName` 值來取代 `policy_name`，使用上一個命令的 `versionId` 值取代 `version_ID`，然後執行此命令來刪除政策版本。

```
aws iot delete-policy-version --policy-name policy_name --policy-version-id version_ID
```

如果成功，此命令不會傳回任何內容。

刪除政策版本後，請重複此步驟，直到政策只有一個政策版本為止。

- iv. 使用 `policyName` 值來取代 `policy_name`，然後執行此命令來刪除政策。

```
aws iot delete-policy --policy-name policy_name
```

- d. 使用物件的名稱來取代 `thing_name`，再使用憑證的 ARN 來取代 `certificate_ARN`，然後執行此命令來從物件資源分離憑證。

```
aws iot detach-thing-principal --thing-name thing_name --principal certificate_ARN
```

如果成功，此命令不會傳回任何內容。

- e. 使用上一個命令的憑證 ID 來取代 `certificate_ID`。憑證 ID 是上一個命令傳回的 ARN 中 `cert/` 後面的英數字元。然後執行此命令來刪除憑證資源。

```
aws iot delete-certificate --certificate-id certificate_ID
```

如果成功，此命令不會傳回任何內容。



3. 使用物件的名稱來取代 *thing\_name*，然後執行此命令來刪除該物件。

```
aws iot delete-thing --thing-name thing_name
```

如果成功，此命令不會傳回任何內容。

## 清除 AWS 資源

在完成此學習路徑中的教學課程後，此程序可協助您識別並移除所建立的其他 AWS 資源。

在此學習路徑中建立的其他 AWS 資源

教學課程	資源類型	資源名稱或 ID
<a href="#">the section called “使用 AWS IoT 裝置用戶端來示範遠端動作 (任務)”</a>	Amazon S3 物件	hello-world-job.json
<a href="#">the section called “使用 AWS IoT 裝置用戶端來示範遠端動作 (任務)”</a>	AWS IoT 任務資源	使用者定義

刪除在此學習路徑中建立的 AWS 資源

1. 刪除在此學習路徑中建立的任務
  - a. 執行此命令來列出 AWS 帳戶 中的任務。

```
aws iot list-jobs
```

此命令會在 AWS 帳戶 和 AWS 區域 中傳回 AWS IoT 任務的清單，如下所示。

```
{
  "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. 針對您從清單中辨識出在此學習路徑中建立的任務的每個任務，請使用要刪除任務的 `jobId` 值來取代 `jobId`，然後執行此命令來刪除 AWS IoT 任務。

```
aws iot delete-job --job-id jobId
```

如果成功，此命令不會傳回任何內容。

## 2. 刪除儲存在此學習路徑 Amazon S3 儲存貯體中的任務文件

- a. 使用您採用的儲存貯體名稱來取代 `bucket`，然後執行此命令來列出您使用的 Amazon S3 儲存貯體中的物件。

```
aws s3api list-objects --bucket bucket
```

此命令會傳會儲存貯體中的 Amazon S3 物件清單，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. 對於您從清單中辨識為在此學習路徑中建立的物件的每個物件，請使用儲存貯體名稱來取代 *bucket*，使用要刪除物件的鍵值來取代 *key*，然後執行此命令來刪除 Amazon S3 物件。

```
aws s3api delete-object --bucket bucket --key key
```

如果成功，此命令不會傳回任何內容。

在完成此學習路徑時刪除建立的所有 AWS 資源和物件之後，您可以重新開始並重複教學課程。

# 使用 AWS IoT 裝置 SDK 建置解決方案

本節中的教學課程可逐步引導您開發 IoT 解決方案，該解決方案可使用 AWS IoT 佈署至生產環境。

要完成這些教學課程的時間可能會比完成 [the section called “使用 AWS IoT 裝置用戶端建置示範”](#) 小節中的教學課程更久，因為它們使用 AWS IoT 裝置 SDK 並詳細說明所套用的概念，以協助您建立安全且可靠的解決方案。

## 開始使用 AWS IoT 裝置 SDK 建置解決方案

這些教學課程會逐步引導你完成不同的 AWS IoT 方案。在適當的情況下，教學課程會使用 AWS IoT 裝置 SDK。

### 主題

- [教學課程：使用 AWS IoT 裝置 SDK 將裝置連接至 AWS IoT Core](#)
- [建立 AWS IoT 規則以將裝置資料路由至其他服務](#)
- [在裝置離線時保留裝置狀態](#)
- [教學課程：建立 AWS IoT Core 的自訂授權方](#)
- [教程:監測土壤水分 AWS IoT 與樹莓派](#)

## 教學課程：使用 AWS IoT 裝置 SDK 將裝置連接至 AWS IoT Core

此教學課程示範如何將裝置連線至 AWS IoT Core，使其可以將資料傳送至接收自 AWS IoT。完成本教學課程後，您的裝置會配置為連接至 AWS IoT Core，而您將會了解裝置如何與 AWS IoT 進行通訊。

在此教學課程中，您將會：

1. [the section called “為 AWS IoT 準備您的裝置”](#)
2. [the section called “檢閱 MQTT 通訊協定”](#)
3. [the section called “檢閱 pubsub.py 裝置 SDK 範例應用程式”](#)
4. [the section called “連線您的裝置並與 AWS IoT Core 進行通訊”](#)
5. [the section called “檢視結果”](#)

此教學課程約需 1 小時方能完成。

開始本教學課程之前，請確定您有：

- 已完成 [開始使用 AWS IoT Core](#)

在該教學課程中您必須進行 [the section called “設定您的裝置”](#) 的部分中，請為您的裝置選取 [the section called “連接 Raspberry Pi 或其他裝置”](#) 選項，並使用 Python 語言選項來配置您的裝置。

讓該教學課程中的終端機視窗處於開啟狀態，因為您還會在本教學課程中用到它。

- 一個可執行適用 Python 的 AWS IoT 裝置 SDK v2 的裝置。

本教學課程示範如何使用 Python 程式碼範例將裝置連線至 AWS IoT Core，這需要一個相對強大的裝置。

若您使用的是資源受限的裝置，這些程式碼範例可能無法在這些裝置上運作。在這種狀況下，您可能會因為 [the section called “使用 適用於 Embedded C 的 AWS IoT Device SDK”](#) 教學課程而學到更多。

## 為 AWS IoT 準備您的裝置

在 [開始使用 AWS IoT Core](#) 中，您準備好裝置和 AWS 帳戶，使其可進行通訊。本節檢閱適用於與 AWS IoT Core 的任何裝置連線之各方面的準備工作。

若為要連線至 AWS IoT Core 的裝置：

1. 您必須具有 AWS 帳戶。

若您尚無 AWS 帳戶，[設定您的 AWS 帳戶](#) 中的程序會說明如何將其建立。

2. 在該帳戶中，您必須具有為您 AWS 帳戶 和區域中之裝置定義的下列 AWS IoT 資源。

程序[建立 AWS IoT 資源](#) 中的程序說明如何為您在 AWS 帳戶 和區域中的裝置建立這些資源。

- 以 AWS IoT 註冊並啟用以驗證裝置的裝置憑證。

憑證通常是使用 AWS IoT 物件建立並加以連接。雖然裝置不需要一個連線至 AWS IoT 的物件，其使裝置可使用其他的 AWS IoT 功能。

- 連接至授權其連線至 AWS IoT Core 之裝置憑證的政策，執行希望其執行的所有動作。

3. 可存取您 AWS 帳戶 之裝置端點的網際網路連線。

裝置端點的說明請參閱 [AWS IoT 裝置資料和服務端點](#)，可於 [AWS IoT 主控台](#) 中的 [設定頁面](#) 找到。

4. 通訊軟體，例如 AWS IoT 裝置 SDK 提供。本教學課程使用 [適用於 Python 的 AWS IoT 裝置 SDK v2](#)。

## 檢閱 MQTT 通訊協定

在我們談論範例應用程式之前，其有助於了解 MQTT 通訊協定。MQTT 通訊協定提供了一些優於其他網路通訊協定 (如 HTTP) 的優勢，這使其成為 IoT 裝置的熱門選擇。本節將會檢視 MQTT 適用於本教學課程的主要層面。如需 MQTT 與 HTTP 比較的資訊，請參閱 [為您的裝置通訊選擇通訊協定](#)。

MQTT 使用一個發佈/訂閱通訊模型。

MQTT 通訊協定會與其主機一起使用發佈/訂閱通訊模型。此模型與 HTTP 使用的請求/回應模型不同。使用 MQTT，裝置會與由唯一用戶端 ID 識別的主機建立工作階段。如要傳送資料，裝置會將主題識別的訊息發佈至主機中的訊息代理程式。為了收到訊息代理程式的訊息，裝置會在訂閱請求中傳送主題篩選條件給訊息代理程式，以訂閱其會收到的主題。

### MQTT 支援持久性工作階段

訊息代理程式會收到來自裝置的訊息，並將訊息發佈至已加以訂閱的裝置。利用[持久性工作階段](#) (即使啟動裝置以中斷連線，工作階段仍保持活動狀態)，裝置可擷取中斷連線時所發佈的訊息。在裝置端，MQTT 支援服務品質層級 ([QoS](#))，以確保主機接收裝置所傳送的訊息。

## 檢閱 pubsub.py 裝置 SDK 範例應用程式

本節會從用於本教學課程中之適用於 Python 的 AWS IoT 裝置 SDK v2 檢閱 pubsub.py 範例應用程式。我們將在此檢閱其如何連接至 AWS IoT Core 以發佈及訂閱 MQTT 訊息。下一節將會展示一些練習，可協助您探索裝置如何連接並與 AWS IoT Core 通訊。

pubsub.py 範例應用程式會示範與 AWS IoT Core 連線之 MQTT 的這些層面：

- [通訊協定](#)
- [持久性工作階段](#)
- [服務品質](#)
- [訊息發佈](#)
- [訊息訂閱](#)
- [裝置中斷連線及重新連線](#)

### 通訊協定

pubsub.py 範例會示範使用 MQTT 和透過 WSS 的 MQTT 通訊協定的 MQTT 連線。[AWS 通用執行時間 \(AWS CRT\)](#) 程式庫提供低階通訊協定支援，並包含適用於 Python 的 AWS IoT 裝置 SDK v2。

## MQTT

pubsub.py 範例呼叫 [mqtt\\_connection\\_builder](#) 中的 `mtls_from_path` (如圖所示)，使用 MQTT 通訊協定來建立與 AWS IoT Core 的連線。`mtls_from_path` 會使用 X.509 憑證和 TLS v1.2 來驗證裝置。AWS CRT 程式庫會處理該連線的較低階詳細資料。

```
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 AWS 帳戶 的 IoT 裝置端點

於範例應用程式中，此值會從命令列傳入。

### cert\_filepath

裝置憑證檔案的路徑。

於範例應用程式中，此值會從命令列傳入。

### pri\_key\_filepath

使用其憑證檔案建立的裝置私密金鑰檔案路徑

於範例應用程式中，此值會從命令列傳入。

### ca\_filepath

根 CA 檔案的路徑。僅當 MQTT 伺服器使用不在您信任庫中的憑證時才需要。

於範例應用程式中，此值會從命令列傳入。

### client\_bootstrap

處理通訊端通訊活動的通用執行時間物件

在範例應用程式中，此物件就在呼叫 `mqtt_connection_builder.mtls_from_path` 之前進行實例化。

`on_connection_interrupted`, `on_connection_resumed`

裝置連線遭到中斷並回復時呼叫的回呼函數。

`client_id`

唯一在 AWS 區域 中識別此裝置的 ID。

於範例應用程式中，此值會從命令列傳入。

`clean_session`

是否啟動新的持續性工作階段，或者，若存在，則重新連接到現有的工作階段

`keep_alive_secs`

在 CONNECT 請求中傳送的保持活動值 (以秒為單位)。Ping 會在此時間間隔自動傳送。若在此值的 1.5 倍之後未收到 ping，則伺服器會假設連線中斷。

## 透過 WSS 的 MQTT

`pubsub.py` 範例呼叫 [mqtt\\_connection\\_builder](#) 中的 `websockets_with_default_aws_signing` (如圖所示)，使用透過 WSS 的 MQTT 通訊協定來建立與 AWS IoT Core 的連線。`websockets_with_default_aws_signing` 會使用 [Signature 第 4 版](#) 透過 WSS 建立 MQTT 連線，來驗證裝置。

```
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 AWS 帳戶 的 IoT 裝置端點

於範例應用程式中，此值會從命令列傳入。

## client\_bootstrap

處理通訊端通訊活動的通用執行時間物件

在範例應用程式中，此物件就在呼叫

`mqtt_connection_builder.websockets_with_default_aws_signing` 之前進行實例化。

## region

Signature 第 4 版身分驗證所使用的 AWS 簽名區域。於 `pubsub.py` 中，其會傳遞在命令列中輸入的參數。

於範例應用程式中，此值會從命令列傳入。

## credentials\_provider

提供用於身分驗證的 AWS 憑證

在範例應用程式中，此物件就在呼叫

`mqtt_connection_builder.websockets_with_default_aws_signing` 之前進行實例化。

## websocket\_proxy\_options

HTTP 代理選項 (若使用代理主機)

在範例應用程式中，此值就在呼叫

`mqtt_connection_builder.websockets_with_default_aws_signing` 之前進行初始化。

## ca\_filepath

根 CA 檔案的路徑。僅當 MQTT 伺服器使用不在您信任庫中的憑證時才需要。

於範例應用程式中，此值會從命令列傳入。

## on\_connection\_interrupted, on\_connection\_resumed

裝置連線遭到中斷並回復時呼叫的回呼函數。

## client\_id

在 AWS 區域 中唯一識別此裝置的 ID。

於範例應用程式中，此值會從命令列傳入。

## clean\_session

是否啟動新的持續性工作階段，或者，若存在，則重新連接到現有的工作階段

## keep\_alive\_secs

在 CONNECT 請求中傳送的保持活動值 (以秒為單位)。Ping 會在此時間間隔自動傳送。若在此值的 1.5 倍之後未收到 ping，則伺服器會假設連線中斷。

## HTTPS

HTTPS 呢？AWS IoT Core 支援發佈 HTTPS 請求的裝置。從程式設計的角度來看，裝置會將 HTTPS 請求傳送至 AWS IoT Core，就像任何其他應用程式一樣。有關從裝置傳送 HTTP 訊息的 Python 程式範例，請參閱使用 Python requests 程式庫的 [HTTPS 程式碼範例](#)。此範例會使用 HTTPS 將訊息傳送至 AWS IoT Core，以便 AWS IoT Core 將其解譯為 MQTT 訊息。

AWS IoT Core 支援來自裝置的 HTTPS 請求時，請務必檢閱 [為您的裝置通訊選擇通訊協定](#) 的相關資訊，此可讓您就您裝置通訊使用何種通訊協定做出明智的決定。

## 持久性工作階段

於範例應用程式中，將 clean\_session 參數設定為 False 表示連線應該是持續的。於實踐中，這意味著透過此呼叫開啟的連線重新連接至現有的持久性工作階段 (如若存在)。否則，其會建立並連線至新的持續性工作階段。

使用持續性工作階段，在裝置未連線時，訊息代理程式會儲存傳送至裝置的訊息。當裝置重新連線至持續性工作階段時，訊息代理程式會將所有已訂閱的儲存訊息傳送至裝置。

若無持續性工作階段，裝置將不會收到裝置未連線時傳送的訊息。要使用哪個選項取決於您的應用程式，及是否必須對裝置未連接時產生訊息進行通訊。如需更多詳細資訊，請參閱 [MQTT 持久性工作階段](#)。

## 服務品質

當裝置發佈並訂閱訊息時，可設定偏好的服務品質 (QoS)。AWS IoT 支援用於發佈和訂閱作業的 QoS 層級 0 和 1。如需在 AWS IoT 中之 QoS 層級的詳細資訊，請參閱 [MQTT 服務品質 \(QoS\) 選項](#)。

Python 的 AWS CRT 執行時間會為其支援的 QoS 層級定義這些常數：

## Python 服務品質層級

MQTT QoS 層級	SDK 所使用的 Python 符號值	描述
QoS 層級 0	<code>mqtt.QoS.AT_MOST_ONCE</code>	無論是否收到訊息，都只會嘗試傳送一次訊息。該訊息可能根本不會傳送，例如，若裝置未連線或網路發生錯誤。
QoS 層級 1	<code>mqtt.QoS.AT_LEAST_ONCE</code>	訊息會重複傳送，直到收到 PUBACK 確認為止。

於範例應用程式中，發佈和訂閱請求的 QoS 層級為 1 (`mqtt.QoS.AT_LEAST_ONCE`)。

- 發佈時的 QoS

當裝置發佈 QoS 層級 1 的訊息時，其會重複傳送訊息，直到收到來自訊息代理程式的 PUBACK 回應為止。若裝置未連線，訊息會在重新連線後佇列等待傳送。

- 訂閱時的 QoS

當裝置訂閱 QoS 層級 1 的訊息時，訊息代理程式會儲存裝置訂閱的訊息，直至其可傳送至裝置為止。訊息代理程式會重新傳送訊息，直至收到來自裝置的 PUBACK 回應。

## 訊息發佈

成功建立 AWS IoT Core 連線之後，裝置便可發佈訊息。`pubsub.py` 範例經由呼叫 `mqtt_connection` 物件的 `publish` 操作來完成此作業。

```
mqtt_connection.publish(  
    topic=args.topic,  
    payload=message,  
    qos=mqtt.QoS.AT_LEAST_ONCE  
)
```

## topic

識別訊息的訊息主題名稱

於範例應用程式中，這會從命令列傳入。

## payload

將訊息承載格式化為字串 (例如，JSON 文件)

於範例應用程式中，這會從命令列傳入。

JSON 文件是一種常用的承載格式，可為其他 AWS IoT 服務識別；然而，訊息承載的資料格式可為發佈者和訂閱者同意的任何內容。但在某些狀況下，對大多數的作業而言，其他 AWS IoT 服務僅識別 JSON 和 CBOR。

## qos

此訊息的 QoS 層級

## 訊息訂閱

如要自 AWS IoT 和其他服務和裝置接收訊息，裝置會依其主題名稱訂閱這些訊息。裝置可透過指定[主題名稱](#)訂閱個別訊息，及透過指定可包含萬用字元字的[主題篩選條件](#)訂閱一組訊息。pubsub.py 範例會使用顯示於此處的程式碼來訂閱訊息，並註冊回呼函數，以於收到訊息之後進行處理。

```
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

要訂閱的主題。此可為主題名稱或主題篩選條件。

於範例應用程式中，這會從命令列傳入。

## qos

訊息代理程式是否應該在裝置中斷連線斷時儲存這些訊息。

mqtt.QoS.AT\_LEAST\_ONCE 的值 (QoS 層級 1) 需要在建立連線時指定持久性工作階段 (clean\_session=False)。

## callback

呼叫來處理訂閱訊息的函數。

此 `mqtt_connection.subscribe` 函式會傳回一個未來和一個封包 ID。若訂閱請求已順利啟動，傳回的封包 ID 大於 0。如要確定訊息代理程式收到訂閱並已註冊，您必須等待非同步作業的結果傳回，如程式碼範例中所示。

## 回呼函數

`pubsub.py` 範例中的回呼會在裝置收到訂閱訊息時進行處理。

```
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

### 訊息的主題

即使您已訂閱了主題篩選條件，此為所接收訊息的特定主題名稱。

## payload

### 訊息承載

此格式為應用程式專用。

## kwargs

可能的其他引數，如 [mqtt.Connection.subscribe](#) 中所述。

在 `pubsub.py` 範例中，`on_message_received` 只會顯示主題及其承載。其還會計算在達到限制後，所收到結束程式的訊息。

您的應用程式會評估主題和承載，以決定要執行的動作。

## 裝置中斷連線及重新連線

`pubsub.py` 範例包含在裝置中斷連線及重新建立連線時呼叫的回呼函數。您的裝置對這些事件採取的動作為應用程式專用。

裝置首次連線時，必須訂閱要接收的主題。若裝置的工作階段在重新連線時存在，則會還原其訂閱，並在重新連線後將來自這些訂閱的任何儲存訊息傳送至裝置。

若裝置的工作階段在重新連線時不復存在，則必須重新訂閱其訂閱。持續性工作階段具有有限的存留期，當裝置中斷連線過久時，可能會過期。

## 連線您的裝置並與 AWS IoT Core 進行通訊

本節提供一些練習，協助您探索將裝置連線至 AWS IoT Core 的不同層面。於這些練習中，您將會使用 AWS IoT 主控台中的 [MQTT 測試用戶端](#)，查看裝置發佈的內容，及將訊息發佈至裝置。這些練習使用來自適用於 Python 的 [AWS IoT 裝置 SDK v2](#) 的 `pubsub.py` 範例，並以您 [開始使用 AWS IoT Core](#) 教學課程經驗為基礎。

在本節中，您會：

- [訂閱萬用字元主題篩選條件](#)
- [處理主題篩選條件訂閱](#)
- [從您的裝置發佈訊息](#)

對於這些練習，您會從 `pubsub.py` 範例程式開始。

### Note

這些練習假設您已完成 [開始使用 AWS IoT Core](#) 教學課程，並使用該教學課程中的裝置終端機視窗。

### 訂閱萬用字元主題篩選條件

在本練習中，您會修改用來呼叫 `pubsub.py` 以訂閱萬用字元主題篩選條件並根據訊息主題處理所接收訊息的命令列。

### 演練程序

在此練習中，想像您的裝置包含溫度控制和光源控制。其會使用這些主題名稱來識別與其相關的訊息。

1. 開始練習之前，請試著從裝置上的 [開始使用 AWS IoT Core](#) 教學課程執行此命令，確保所有項目都準備好進行練習。

```
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
```

您應該會看到與在[教學課程入門](#)中看到的相同輸出。

- 在此練習中，請變更這些命令列參數。

動作	命令列參數	Effect
add	--message ""	僅限配置 pubsub.py 加以聆聽
add	--count 2	收到兩則訊息後結束程式
change	--topic device/+/details	定義要訂閱的主題篩選條件

對初始命令列進行這些變更會產生此命令列。在裝置的終端機視窗中輸入此命令。

```
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
```

程式應該會顯示如下所示的內容：

```
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...
```

若您在終端機上看到類似的內容，表示您的裝置已準備就緒，且會收聽主題名稱開頭為 device，結尾為 /detail 的訊息。所以，讓我們來進行測試。

- 以下為您的裝置可能會收到的幾個訊息。

主題名稱	訊息承載
device/temp/details	{ "desiredTemp": 20, "currentTemp": 15 }

主題名稱	訊息承載
device/light/details	{ "desiredLight": 100, "currentLight": 50 }

4. 在 AWS IoT 主控台中使用 MQTT 測試用戶端，將上一個步驟中說明的訊息傳送至您的裝置。
  - a. 在 AWS IoT 主控台中開啟 [MQTT 測試用戶端](#)。
  - b. 於 Subscribe to a topic (訂閱主題) 的 Subscription topic (訂閱主題) 欄位中，輸入主題篩選條件：**device/+/details**，再選擇 Subscribe to topic (訂閱主題)。
  - c. 在 MQTT 測試用戶端的 Subscriptions (訂閱) 欄中，選擇 device/+/details。
  - d. 若為上表中的每個主題，請於 MQTT 測試用戶端中執行下列動作：
    1. 於 Publish (發佈) 中，輸入來自表格中 Topic name (主題名稱) 欄的值。
    2. 在主題名稱下方的訊息承載欄位中，輸入來自表格中 Message payload (訊息承載) 欄的值。
    3. 觀看 pubsub.py 正在執行的終端機視窗，並在 MQTT 測試用戶端中，選擇 Publish to topic (發佈至主題)。

您應該看到該訊息是由終端機視窗中的 pubsub.py 所收到。

## 演練結果

有了此 pubsub.py，使用萬用字元主題篩選條件訂閱訊息，加以訊息，並將其顯示於終端機視窗中。請注意您如何訂閱單一主題篩選條件，並呼叫回呼函數來處理具有兩個不同主題的訊息。

## 處理主題篩選條件訂閱

建置於上一個練習上，修改 pubsub.py 範例應用程式來評估訊息主題，並根據主題處理訂閱的訊息。

## 演練程序

### 如要評估訊息主題

1. 將 pubsub.py 複製至 pubsub2.py。
2. 以您最愛的文字編輯器或 IDE 中開啟 pubsub2.py 檔案。
3. 在 pubsub2.py 中，尋找 on\_message\_received 函數。



- 在 `on_message_received` 中，請在以 `print("Received message` 開頭的行之後及以 `global received_count` 開頭的行之前插入下列程式碼。

```

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.")

```

- 儲存您的變更，並使用此命令列執行修改後的程式。

```

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

```

- 在 AWS IoT 主控台中，開啟 [MQTT 測試用戶端](#)。
- 於 Subscribe to a topic (訂閱主題) 的 Subscription topic (訂閱主題) 欄位中，輸入主題篩選條件：**device/+/details**，再選擇 Subscribe to topic (訂閱主題)。
- 在 MQTT 測試用戶端的 Subscriptions (訂閱) 欄中，選擇 device/+/details。
- 若為此表格中的每個主題，請在 MQTT 測試用戶端中執行下列動作：

主題名稱	訊息承載
device/temp/details	{ "desiredTemp": 20, "currentTemp": 15 }
device/light/details	{ "desiredLight": 100, "currentLight": 50 }

1. 於 Publish (發佈) 中，輸入來自表格中 Topic name (主題名稱) 欄的值。
2. 在主題名稱下方的訊息承載欄位中，輸入來自表格中 Message payload (訊息承載) 欄的值。
3. 觀看 pubsub.py 正在執行的終端機視窗，並在 MQTT 測試用戶端中，選擇 Publish to topic (發佈至主題)。

您應該看到該訊息是由終端機視窗中的 pubsub.py 所收到。

您應該會在終端機視窗中看到類似的內容。

```
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!
```

## 演練結果

在本練習中，您已新增程式碼，讓範例應用程式能辨識並處理回呼函數中的多個訊息。有了這個，您的裝置可以接收訊息並對其採取行動。

裝置接收和處理多則訊息的另一種方式是分別訂閱不同訊息，並將每個訂閱分配給各自的回呼函數。

## 從您的裝置發佈訊息

您可使用 pubsub.py 範例應用程式，以發佈來自您裝置的訊息。在其會發佈訊息時，這些訊息無法作為 JSON 文件加以讀取。此練習會修改範例應用程式，以便能夠發佈在可由 AWS IoT Core 讀取之訊息承載中的 JSON 文件。

## 演練程序

在本練習中，下列訊息將會與 `device/data` 主題一起傳送。

```
{
  "timestamp": 1601048303,
  "sensorId": 28,
  "sensorData": [
    {
      "sensorName": "Wind speed",
      "sensorValue": 34.2211224
    }
  ]
}
```

如要準備 MQTT 測試用戶端監控本練習的訊息

1. 於 `Subscribe to a topic` (訂閱主題) 的 `Subscription topic` (訂閱主題) 欄位中，輸入主題篩選條件：**`device/data`**，再選擇 `Subscribe to topic` (訂閱主題)。
2. 在 MQTT 測試用戶端的 `Subscriptions` (訂閱) 欄中，選擇 `device/data`。
3. 讓 MQTT 測試用戶端視窗保持開啟狀態，以等待來自您裝置的訊息。

如要使用 `pubsub.py` 範例應用程式傳送 JSON 文件

1. 在您的裝置上，將 `pubsub.py` 複製至 `pubsub3.py`。
2. `Edit` (編輯) `pubsub3.py`，變更其格式化所發佈訊息的方式。
  - a. 在文字編輯器中開啟 `pubsub3.py`。
  - b. 找出這行程式碼：

```
message = "{} [{}]" .format(message_string, publish_count)
```

- c. 將其變更為：

```
message = "{}" .format(message_string)
```

- d. 找出這行程式碼：

```
message_json = json.dumps(message)
```

- e. 將其變更為：

```
message = "{}".json.dumps(json.loads(message))
```

f. 儲存您的變更。

3. 在您的裝置上，執行此命令以傳送訊息兩次。

```
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. 於 MQTT 測試用戶端中，檢查是否已解譯並格式化訊息承載中的 JSON 文件，例如：

```
device/data          September 25, 2020, 08:57:14 (UTC-0700)          Export Hide
{
  "timestamp": 1601048303,
  "sensorId": 28,
  "sensorData": [
    {
      "sensorName": "Wind speed",
      "sensorValue": 34.2211224
    }
  ]
}
```

依預設，pubsub3.py 也會訂閱其傳送的訊息。您應會看到其在應用程式輸出中收到訊息。終端機視窗看起來與下列內容相似。

```
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!
```

## 演練結果

有了這個，您的裝置可產生訊息傳送至 AWS IoT Core 來測試基本連線能力，並提供裝置訊息以供 AWS IoT Core 處理。例如，您可使用此應用程式，從您的裝置傳送測試資料，以測試 AWS IoT 規則動作。

## 檢視結果

本教學課程中的範例為您提供了有關裝置如何與 AWS IoT Core 進行通訊之基礎知識的實作經驗，此為您 AWS IoT 解決方案的基本部分。當您的裝置可與 AWS IoT Core 進行通訊時，其可將訊息傳遞至 AWS 服務及其可採取行動的其他裝置。同樣地，AWS 服務和其他裝置可處理造成訊息傳回您裝置的資訊。

當您準備好進一步探索 AWS IoT Core 時，請試試這些教學課程：

- [the section called “正在傳送 Amazon SNS 通知”](#)
- [the section called “將裝置資料儲存於 DynamoDB 表格中”](#)
- [the section called “使用 AWS Lambda 函數格式化通知”](#)

## 教學課程：使用適用於 Embedded C 的 AWS IoT Device SDK

本節說明如何執行適用於 Embedded C 的 AWS IoT Device SDK。

本節中的程序

- [步驟 1：安裝適用於 Embedded C 的 AWS IoT Device SDK](#)
- [步驟 2：設定範例應用程式](#)
- [步驟 3：建置並執行範例應用程式](#)

### 步驟 1：安裝適用於 Embedded C 的 AWS IoT Device SDK

適用於 Embedded C 的 AWS IoT Device SDK 常針對需要優化 C 語言運行時的資源受限設備。您可以在任何作業系統上使用軟體開發套件，並將其裝載在任何處理器類型 (例如 MCU 和 MPU) 上。如

果您有更多可用的記憶體和處理資源，建議您使用其中一個較高階的 AWS IoT 裝置和行動 SDK (例如 C++ JavaScript、Java 和 Python)。

一般而言，適用於 Embedded C 的 AWS IoT Device SDK 於使用 MCU 或執行嵌入式作業系統的低階 MPU 的系統。在本節中的程式設計範例，我們假設您的裝置使用 Linux。

## Example

1. 從下載 適用於 Embedded C 的 AWS IoT Device SDK 到您的設備[GitHub](#)。

```
git clone https://github.com/aws/aws-iot-device-sdk-embedded-c.git --recurse-submodules
```

這會在目前目錄中建立名為 `aws-iot-device-sdk-embedded-c` 的目錄。

2. 導航至該目錄並查看最新版本。如需最新的發行版本標籤，請參閱 [github.com/aws/ aws-iot-device-sdk-embedded-c](https://github.com/aws/aws-iot-device-sdk-embedded-c) 嵌入式 C / 標籤。

```
cd aws-iot-device-sdk-embedded-c
git checkout latest-release-tag
```

3. 安裝 OpenSSL 1.1.0 版或更新版本。當透過套件管理工具進行安裝時，OpenSSL 開發程式庫通常會稱為 "libssl-dev" 或 "openssl-devel"。

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

## 步驟 2：設定範例應用程式

適用於 Embedded C 的 AWS IoT Device SDK 其中包括示例應用程序供您嘗試。為了簡單起見，本教學課程會使用 `mqtt_demo_mutual_auth` 應用程式，說明如何連線至 AWS IoT Core 訊息代理程式，以及訂閱並發佈至 MQTT 主題。

1. 將您建立於 [開始使用 AWS IoT Core](#) 中的憑證和私有金鑰複製至 `build/bin/certificates` 目錄中。

### Note

裝置和根憑證授權機構憑證會有過期或遭撤銷的可能。若這些憑證過期或遭到撤銷，您必須將新的憑證授權機構憑證或私有金鑰和裝置憑證複製到裝置上。

2. 您必須使用個人 AWS IoT Core 端點、私密金鑰、憑證和根 CA 憑證來設定範例。請前往 `aws-iot-device-sdk-embedded-c/demos/mqtt/mqtt_demo_mutual_auth` 目錄。

如果您已 AWS CLI 安裝，則可以使用此命令尋找帳戶的端點 URL。

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

如果您沒有 AWS CLI 安裝，請開啟 [AWS IoT 主機](#)。在導覽窗格中，選擇 Manage (管理)，然後選擇 Things (物件)。為您的裝置選擇 IoT 物件，然後選擇 Interact (互動)。您的端點會顯示在物件詳細資料頁面的 HTTPS 區段中。

3. 開啟 `demo_config.h` 檔案，並更新下列內容的值：

AWS\_IOT\_ENDPOINT

您的個人端點。

CLIENT\_CERT\_PATH

您的憑證檔案路徑，例如 `certificates/device.pem.crt`。

CLIENT\_PRIVATE\_KEY\_PATH

您的私有金鑰檔案名稱，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. 使用此命令來檢查您的裝置是否已安裝 CMake。

```
cmake --version
```

若您看見編譯器的版本資訊，則可繼續到下一節。

若發生錯誤，或者沒看見資任何訊，您將需要使用此命令來安裝 cmake 套件。

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

再次執行 `cmake --version` 命令，確認已安裝 CMake，備妥繼續。

5. 使用此命令來檢查您的裝置是否已安裝開發工具。

```
gcc --version
```

若您看見編譯器的版本資訊，則可繼續到下一節。

如果發生錯誤，或者沒看見編譯器資訊，您將需要使用此命令來安裝 build-essential 套件。

```
sudo apt-get install build-essential
```

再次執行 `gcc --version` 命令，確認已安裝建置工具，準備好繼續。

### 步驟 3：建置並執行範例應用程式

執行適用於 Embedded C 的 AWS IoT Device SDK 範例應用程式

1. 導覽至 `aws-iot-device-sdk-embedded-c`，建立一個建置目錄。

```
mkdir build && cd build
```

2. 輸入下列 CMake 命令，產生需要建置的 Makefile。

```
cmake ..
```

3. 輸入下列命令來建置可執行的應用程式檔案。

```
make
```

4. 使用此命令執行 `mqtt_demo_mutual_auth` 應用程式。

```
cd bin
```



```
./mqtt_demo_mutual_auth
```

您應該會看到類似下列的輸出：

```
[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.
```

您的裝置現在已連接到 AWS IoT 使用 適用於 Embedded C 的 AWS IoT Device SDK。

您也可以使用 AWS IoT 主控台來檢視範例應用程式正在發佈的 MQTT 訊息。如需關於如何在 [AWS IoT 主控台](#) 中使用 MQTT 用戶端的資訊，請參閱 [the section called “使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT”](#)。

## 建立 AWS IoT 規則以將裝置資料路由至其他服務

這些教學課程會示範如何使用一些較常見的 AWS IoT 規則動作來建立和測試規則。

AWS IoT 規則會將資料從您的裝置傳送至其他 AWS 服務。其會聆聽特定的 MQTT 訊息、格式化訊息承載中的資料，並將結果傳送至其他 AWS 服務。

即使目標是要建立使用 Lambda 函數或更複雜的規則，我們仍建議您依照此處顯示的順序進行嘗試。教學課程會依照基本到複雜的順序顯示。其會逐步提供新概念，協助您學習可用來建立沒有特定教學課程的規則動作概念。

**Note**

AWS IoT 規則可協助您將資料從 IoT 裝置傳送至其他 AWS 服務。不過，如要成功地達到這一點，您需具備要傳送資料的其他服務的相關工作知識。雖然這些教學課程提供了完成任務所需的資訊，但您可能會發現，在解決方案中使用這些服務之前，進一步了解將資料傳送至服務的相關資訊可能更有幫助。其他 AWS 服務的詳細說明不在這些教程的範圍內。

**教學課程方案概觀**

這些教學課程的方案為定期發佈其資料的天氣感應器裝置。在此虛構的系統中有很多這樣的感應器裝置。不過，本節中的教學課程會聚焦在單一裝置上，同時展現容納多個感應器的方法。

本節中的教學課程將說明如何使用 AWS IoT 規則來使用此虛構的天氣感測器裝置系統執行下列工作。

- [教學課程：重新發佈 MQTT 訊息](#)

本教學課程將會顯示如何將氣象感應器所接收的 MQTT 訊息重新發佈為僅包含感應器 ID 和溫度值的訊息。其僅使用 AWS IoT Core 服務，並展示簡單的 SQL 查詢，及如何使用 MQTT 用戶端來測試您的規則。

- [教學課程：傳送 Amazon SNS 通知](#)

本教學課程會顯示天氣感應器裝置的值超出特定值時，如何傳送 SNS 訊息。它建立在上一個教程中提到的概念，並添加如何使用另一個 AWS 服務，即 [Amazon 簡單通知服務](#) (Amazon SNS)。

若您初次使用 Amazon SNS，請檢閱其[入門練習](#)，然後再開始本教學課程。

- [教學課程：將裝置資料儲存在 DynamoDB 表格中](#)

本教學課程會顯示如何從天氣感應器裝置存放資料庫表中的資料。其會使用規則查詢陳述式和替代範本來格式化目的地服務 [Amazon DynamoDB](#) 的訊息資料。

若您初次使用 DynamoDB，請檢閱其[入門練習](#)，然後再開始本教學課程。

- [教學課程：使用 AWS Lambda 函數來格式化通知](#)

本教學課程顯示如何呼叫 Lambda 函數來重新格式化裝置資料，然後以文字訊息進行傳送。它在函數中添加了 Python 腳本和 AWS SDK [AWS Lambda](#) 函數，以使用來自天氣傳感器設備的消息有效負載數據進行格式化並發送文本消息。

若您初次使用 Lambda，請檢閱其[入門練習](#)，然後再開始本教學課程。

## AWS IoT 規則概觀

所有這些自學課程都會建立 AWS IoT 規則。

對於將數據從設備發送到另一個 AWS 服務的 AWS IoT 規則，它使用：

- 包含下列項目的規則查詢陳述式：
  - SQL SELECT 子句，用於選取並格式化訊息承載中的資料
  - 主題篩選條件 (規則查詢陳述式中的 FROM 物件)，用來識別要使用的訊息
  - 選用性的條件陳述式 (SQL WHERE 子句)，指定要採取行動的特定條件
- 至少一個規則動作

裝置會發佈訊息至 MQTT 主題。SQL SELECT 陳述式中的主題篩選條件會識別要套用規則的 MQTT 主題。SQL SELECT 陳述式中指定的欄位會格式化傳入 MQTT 訊息承載的資料，以供規則動作使用。如需規則動作的完整清單，請參閱 [AWS IoT 規則動作](#)。

本節中的教學課程

- [教學課程：重新發佈 MQTT 訊息](#)
- [教學課程：傳送 Amazon SNS 通知](#)
- [教學課程：將裝置資料儲存在 DynamoDB 表格中](#)
- [教學課程：使用 AWS Lambda 函數來格式化通知](#)

### 教學課程：重新發佈 MQTT 訊息

本教學課程示範如何建立在收到指定的 MQTT 訊息時發佈 MQTT 訊息的 AWS IoT 規則。傳入的訊息承載可在發佈前由規則進行修改。這樣就可以建立專為特定應用程式量身訂製的訊息，而不需要更改裝置或其韌體。您還可使用規則的篩選條件面向，僅在符合特定條件時才發佈訊息。

由規則重新發佈的訊息，就像任何其他 AWS IoT 裝置或用戶端所傳送的訊息一樣。裝置可訂閱重新發佈的訊息，就像裝置可訂閱任何其他 MQTT 訊息主題一樣。

您會在本教學課程中學到什麼：

- 如何在規則查詢陳述式中使用簡單的 SQL 查詢和函數
- 如何使用 MQTT 用戶端測試規則 AWS IoT

此教學課程約需 30 分鐘方能完成。

於本教學課程中，您將會：

- [檢閱 MQTT 主題與規則 AWS IoT](#)
- [步驟 1：建立 AWS IoT 規則以重新發佈 MQTT 訊息](#)
- [步驟 2：測試新規則](#)
- [步驟 3：檢閱結果及後續步驟](#)

開始本教學課程之前，請確定您有：

- [設定您的 AWS 帳戶](#)

您需要您 AWS 帳戶 和 AWS IoT 主控台才能完成本教學課程。

- [檢閱 使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)

請確定您可使用 MQTT 用戶端來訂閱並發佈至主題。您會使用 MQTT 用戶端，在此程序中測試您的新規則。

## 檢閱 MQTT 主題與規則 AWS IoT

在談論 AWS IoT 規則之前，了解 MQTT 協議有助於。在 IoT 解決方案中，MQTT 通訊協定提供了一些優於其他網路通訊協定 (如 HTTP) 的優勢，這使其成為 IoT 裝置使用的熱門選擇。本節將會檢視 MQTT 適用於本教學課程的主要層面。如需 MQTT 與 HTTP 比較的資訊，請參閱 [為您的裝置通訊選擇通訊協定](#)。

## MQTT 通訊協定

MQTT 通訊協定會與其主機一起使用發佈/訂閱通訊模型。若要傳送資料，裝置會將由主題識別的訊息發佈至 AWS IoT 訊息代理程式。若要收到訊息代理程式的訊息，裝置會在訂閱請求中傳送主題篩選條件給訊息代理程式，以訂閱其會收到的主題。AWS IoT 規則引擎會收到來自訊息代理程式的 MQTT 訊息。

## AWS IoT 規則

AWS IoT 規則包含規則查詢陳述式和一或多個規則動作。當 AWS IoT 規則引擎收到 MQTT 訊息時，這些元素對訊息產生的作用如下。

- 規則查詢陳述式

規則的查詢陳述式說明要使用的 MQTT 主題、解譯來自訊息承載的資料，及依據 SQL 陳述式所描述的 SQL 陳述式來格式化資料，這些陳述式類似於常用 SQL 資料庫所使用的陳述式。查詢陳述式的結果為傳送至規則動作的資料。

- 規則動作

規則中的每個規則動作都會對規則查詢陳述式產生的資料作用。AWS IoT 支援[許多規則動作](#)。不過，在本教學課程中，您會專注於 [Republish](#) 規則動作，其會將查詢陳述式的結果發佈為具有特定主題的 MQTT 訊息。

### 步驟 1：建立 AWS IoT 規則以重新發佈 MQTT 訊息

您將在本教學課程中建立的 AWS IoT 規則會訂閱 `device/device_id/data` MQTT 主題，其中 *device\_id* 是傳送訊息之裝置的 ID。[主題篩選條件](#)會將這些主題描述為 `device/+ /data`，其中+ 為與兩個正斜線字元間之任何字串相符的萬用字元。

規則收到來自相符主題的訊息時，其會重新發佈 `device_id` 和 `temperature` 值，作為具 `device/data/temp` 主題的新 MQTT 訊息。

例如，具 `device/22/data` 主題的 MQTT 訊息承載如下所示：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

該規則採用來自訊息承載的 `temperature` 值及來自主題的 `device_id`，並將其重新發佈為具 `device/data/temp` 主題的 MQTT 訊息，及如下所示的訊息承載：

```
{
  "device_id": "22",
  "temperature": 28
}
```

使用此規則，則只需要裝置 ID 和溫度資料的裝置會訂閱 `device/data/temp` 主題，以僅接收該資訊。

如要建立一個重新發佈 MQTT 訊息的規則

1. 開啟[主 AWS IoT 控制台](#)的 **[規則]** 中樞。
2. 在 Rules (規則) 中，選擇 Create(建立)，然後開始建立新規則。
3. 在 Create a rule (建立規則) 的頂部中：
  - a. 在 Name (名稱) 中，輸入規則的名稱。在本教學課程中，請將其命名為 **republish\_temp**。

請記住，規則名稱在您的帳戶和區域內必須是唯一的，且不能有任何空格。我們在此名稱中使用底線字元來分隔規則名稱中的兩個單字。

- b. 在 Description (說明) 中，說明規則。

有意義的說明可幫助您記住此規則的作用及建立規則的原因。說明可依所需而定，因此請盡可能詳細說明。
4. 在 Create a rule (建立規則) 的 Rule query statement (規則查詢陳述式) 中：
    - a. 在使用 SQL 版本中，選取 **2016-03-23**。
    - b. 在 Rule query statement (規則查詢陳述式) 編輯方塊中輸入陳述式：

```
SELECT topic(2) as device_id, temperature FROM 'device+/data'
```

本陳述式：

- 聆聽具與 `device+/data` 主題篩選條件相符之主題的 MQTT 訊息。
  - 從主題字串中選取第二個元素，並將其指定給 `device_id` 欄位。
  - 從訊息承載選取值 `temperature` 欄位，並將其指派給 `temperature` 欄位。
5. 在 Set one or more actions (設定一個或多個動作) 中：
    - a. 若要開啟此規則的規則動作清單，請選擇 Add action (新增動作)。
    - b. 在 [選取動作] 中，選擇 [將郵件重新發佈至 AWS IoT 主題]。
    - c. 在動作清單底部，選擇設定動作以開啟所選動作的組態頁面。
  6. 在 Configure action (設定動作)：
    - a. 在 Topic (主題) 中，輸入 **device/data/temp**。這是此規則將發佈訊息的 MQTT 主題。

- b. 在 Quality of Service (服務品質) 中，選擇 0 - The message is delivered zero or more times (0 - 訊息傳遞零次或多次)。
  - c. 在 [選擇或建立角色以授與執行此動作的 AWS IoT 存取權限] 中：
    - i. 選擇 Create Role (建立角色)。Create a new role (建立新角色) 對話方塊隨即開啟。
    - ii. 輸入可說明新角色的名稱。在本教學課程中，使用 **republish\_role**。

當您建立新角色時，會建立執行規則動作的正確政策，並將其連接至新角色。如果您變更此規則動作的主題，或在其他規則動作中使用此角色，則必須更新該角色的政策，以授權新的主題或動作。若要更新現有角色，請選擇本節中的 Update role (更新角色)。
    - iii. 選擇 Create Role (建立角色)，以建立角色並關閉對話方塊。
  - d. 選擇 Add action (新增動作)，將動作新增至規則，並返回 Create a rule (建立規則) 頁面。
7. [重新發佈郵件至 AWS IoT 主題] 動作現在會列在 [設定一或多個動作] 中。
- 在新動作的圖標中，Republish a message to an AWS IoT topic (將訊息重新發佈至 IoT 主題) 之下，您可看到重新發佈動作將發佈的主題。
- 這是您將新增至此規則的唯一規則動作。
8. 在 Create a rule (建立規則) 中，向下捲動至底部，然後選擇 Create rule (建立規則)，建立規則並完成此步驟。

## 步驟 2：測試新規則

若要測試新規則，您將使用 MQTT 用戶端來發佈和訂閱此規則所使用的 MQTT 訊息。

在新視窗的 [AWS IoT 主控台中開啟 MQTT 用戶端](#)。這可讓您編輯規則，而不會遺失 MQTT 用戶端的組態。如果您讓它轉到主控台的另一個頁面，MQTT 用戶端不會保留任何訂閱或訊息記錄。

如要使用 MQTT 用戶端來測試您的規則。

1. 在 AWS IoT 主控台的 [MQTT 用戶端](#) 中，訂閱輸入主題，在此案例中為 `device/+ /data`。
  - a. 在 MQTT 用戶端中的 Subscriptions (訂閱) 下選擇 Subscribe to a topic (訂閱主題)。
  - b. 在 Subscription topic (訂閱主題) 中，輸入輸入主題篩選條件 **device/+ /data** 的主題。
  - c. 將剩下的欄位保留為其預設設定。
  - d. 請選擇 Subscribe to topic (訂閱主題)。

在 Subscriptions (訂閱) 欄中，Publish to a topic (發佈到主題) 之下，**device/+/data** 隨即顯示。

2. 訂閱規則將發佈的主題：device/data/temp。

- a. 在 Subscriptions (訂閱) 之下，選擇 Subscribe to a topic (訂閱主題)，並在 Subscription topic (訂閱主題) 中，輸入重新發佈訊息 **device/data/temp** 的主題。
- b. 將剩下的欄位保留為其預設設定。
- c. 請選擇 Subscribe to topic (訂閱主題)。

在 Subscriptions (訂閱) 欄中，device/+/data 之下，**device/data/temp** 隨即顯示。

3. 使用特定裝置 ID **device/22/data**，將訊息發佈至輸入主題。您無法發佈至包含萬用字元的 MQTT 主題。

- a. 在 MQTT 用戶端中的 Subscriptions (訂閱) 下選擇 Publish to topic (發佈至主題)。
- b. 在 Publish (發佈) 欄位中輸入輸入主題名稱 **device/22/data**。
- c. 複製此處顯示的範例資料，並在主題名稱下方的編輯方塊中貼上範例資料。

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. 若要傳送 MQTT 訊息，請選擇 Publish to topic (發佈至主題)。

4. 檢閱傳送的訊息。

- a. 在 MQTT 用戶端中的 Subscriptions (訂閱) 下，您先前訂閱的兩個主題旁會有一個綠點。

該綠點表示自上次查看後，已收到一個或多個新訊息。

- b. 於 Subscriptions (訂閱) 下，選擇 device/+/data，來檢查訊息承載是否與您剛剛發佈的內容相符，如下所示：

```
{
  "temperature": 28,
```



```
"humidity": 80,  
"barometer": 1013,  
"wind": {  
  "velocity": 22,  
  "bearing": 255  
}  
}
```

- c. 於 Subscriptions (訂閱) 下，選擇 device/data/temp，來檢查您重新發佈的訊息承載，如下所示：

```
{  
  "device_id": "22",  
  "temperature": 28  
}
```

請注意，device\_id 值是個帶有引號的字串，而 temperature 值是個數字。這是因為 [topic\(\)](#) 函數從輸入訊息的主題名稱中提取字串，而 temperature 值會使用輸入訊息承載的數值。

若要將 device\_id 值設為數值，請將規則查詢陳述式中的 topic(2) 替換為：

```
cast(topic(2) AS DECIMAL)
```

請注意，將 topic(2) 值轉換為數值，僅適用於主題的該部分僅包含數字字元時。

5. 若您看到正確的訊息已發佈至 device/data/temp 主題，則您的規則有效。請參閱下一節，了解更多 Republish (重新發佈) 規則動作的相關資訊。

若您看不到正確的訊息已發佈至 device+/data 或 device/data/temp 主題中，請查看疑難排解提示。

## 疑難排解重新發佈訊息規則

若您並未看到預期的結果，請查看以下事項。

- 您收到錯誤的橫幅

若在您發佈輸入訊息時出現錯誤，請先更正該錯誤。下列步驟可協助您修正該錯誤。

- 您並未在 MQTT 用戶端中看到輸入訊息

每次您將輸入訊息發佈至 `device/22/data` 主題時，若您依程序中所述訂閱了 `device/+data` 主題篩選條件，則該訊息應會顯示於 MQTT 用戶端中。

### 要檢查的事項

- 檢查您訂閱的主題篩選條件

若您依程序中所述訂閱了輸入訊息主題，則每次發佈輸入訊息時都應該會看到其複本。

若您並未訊息，請檢查您訂閱的主題名稱，並將其與所發佈的主題進行比較。主題名稱區分大小寫，且您訂閱的主題必須與所發佈訊息承載的主題相同。

- 檢查訊息發佈功能

在 MQTT 用戶端中的 Subscriptions (訂閱) 下，選擇 `device/+data`，檢查發佈訊息的主題，然後選擇 Publish to topic (發佈至主題)。您應該會在訊息清單中出現主題下方的編輯方塊中看到訊息承載。

- 您在 MQTT 用戶端中並未看到已重新發佈的訊息

若要讓您的規則運作，其必須具有授權其接收和重新發佈訊息的正確政策，且必須接收訊息。

### 要檢查的事項

- 檢查您 AWS 區域的 MQTT 用戶端和您建立的規則

您正在執行 MQTT 用戶端的主控制台必須與您建立的規則處於相同的 AWS 區域。

- 檢查規則查詢陳述式中的輸入訊息主題

若要讓規則運作，其必須收到一則訊息，其主題名稱與規則查詢陳述式之 FROM 子句中的主題篩選條件相符。

檢查規則查詢陳述式中主題篩選條件的拼字與 MQTT 用戶端中主題的拼字。主題名稱區分大小寫，且郵件的主題必須與規則查詢陳述式中的主題篩選條件相符。

- 檢查輸入訊息承載的內容

若要讓規則運作，其必須在 SELECT 陳述式中宣告的訊息承載中尋找資料欄位。

檢查規則查詢陳述式中 `temperature` 欄位的拼字與 MQTT 用戶端中訊息承載的拼字。欄位名稱區分大小寫，規則查詢陳述式中的 `temperature` 欄位必須與訊息承載中的 `temperature` 欄位相符。

請確定訊息承載中的 JSON 文件格式正確。若 JSON 有任何錯誤，例如缺少逗號，則規則將無法進行讀取。

- 檢查規則動作中重新發佈的訊息主題

Republish (重新發佈) 規則動作發佈新訊息的主題必須與您在 MQTT 用戶端中訂閱的主題相符。

開啟您建立於主控台下的規則，並檢查規則動作重新發佈訊息的主題。

- 檢查規則所使用的角色

規則動作必須具有接收原始主題及發佈新主題的權限。

授權規則接收訊息資料的政策並加以重新發佈為所使用的主題所特定的。若變更為重新發佈訊息資料的主題，則必須更新規則動作的角色，來更新其政策以與目前主題相符。

若您懷疑這會是問題，請編輯 Republish (重新發佈) 規則動作並建立新角色。規則動作建立的新角色會收到執行這些動作所需的授權。

### 步驟 3：檢閱結果及後續步驟

於本教學課程中

- 您使用了簡單的 SQL 查詢和規則查詢陳述式中的幾個函數來產生一個新的 MQTT 訊息。
- 您已建立一個重新發佈該新訊息的規則。
- 您使用 MQTT 用戶端來測試您的 AWS IoT 規則。

### 後續步驟

使用此規則重新發佈一些訊息之後，請嘗試使用其來查看教學課程的某些層面如何影響重新發佈的訊息。此處有幾種簡單的入門方式。

- 變更輸入訊息主題中的 *device\_id*，並觀察重新發佈之訊息承載中的影響。
- 變更規則查詢陳述式中所選取的欄位，並觀察重新發佈之訊息承載中的影響。
- 請嘗試本系列中的下一個教學課程，並了解如何進行 [教學課程：傳送 Amazon SNS 通知](#)。

此教學課程中所使用的 Republish (重新發佈) 規則動作亦可協助您對規則查詢陳述式進行偵錯。例如，您可將此動作新增至規則，以查看其規則查詢陳述式如何格式化用於其規則動作的資料。

## 教學課程：傳送 Amazon SNS 通知

本教學課程示範如何建立將 MQTT 訊息資料傳送至 Amazon SNS 主題的 AWS IoT 規則，以便以 SMS 文字訊息的形式傳送。

在本教學課程中，您會建立一個規則，在溫度超過規則中所設定的值時，將訊息資料從天氣感應器傳送至 Amazon SNS 主題的所有訂閱者。該規則會在回報的溫度超過規則設定值時進行偵測，然後建立新的訊息承載資料，其中僅包含裝置 ID、回報的溫度及超過的溫度限制。該規則會將新訊息承載作為 JSON 文件傳送至 SNS 主題，其會通知 SNS 主題的所有訂閱者。

您會在本教學課程中學到什麼：

- 如何建立及測試 Amazon SNS 通知
- 如何從 AWS IoT 規則呼叫 Amazon SNS 通知
- 如何在規則查詢陳述式中使用簡單的 SQL 查詢和函數
- 如何使用 MQTT 用戶端測試規則 AWS IoT

此教學課程約需 30 分鐘方能完成。

於本教學課程中，您將會：

- [步驟 1：建立一個傳送 SMS 簡訊的 Amazon SNS 主題](#)
- [步驟 2：創建 AWS IoT 規則以發送短信](#)
- [步驟 3：測試 AWS IoT 規則和 Amazon SNS 通知](#)
- [步驟 4：檢閱結果及後續步驟](#)

開始本教學課程之前，請確定您有：

- [設定您的 AWS 帳戶](#)

您需要您 AWS 帳戶 和 AWS IoT 主控台才能完成本教學課程。

- 檢閱 [使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)

請確定您可使用 MQTT 用戶端來訂閱並發佈至主題。您會使用 MQTT 用戶端，在此程序中測試您的新規則。

- 檢閱 [Amazon Simple Notification Service](#)

若您未曾使用過 Amazon SNS，請查閱[設定 Amazon SNS 的存取](#)。如果您已經完成其他 AWS IoT 教學課程，您 AWS 帳戶 應該已經正確設定。

## 步驟 1：建立一個傳送 SMS 簡訊的 Amazon SNS 主題

若要建立一個傳送 SMS 簡訊的 Amazon SNS 主題

1. 建立一個 Amazon SNS 主題。
  - a. 登入 [Amazon SNS 主控台](#)。
  - b. 在左側導覽窗格中，選擇 Topics (主題)。
  - c. 在 Topics (主題) 頁面上，選擇 Create new topic (建立新主題)。
  - d. 於 Details (詳細資訊) 中，選擇 Standard (標準) 類型。依預設，主控台會建立一個 FIFO 主題。
  - e. 於 Name (名稱) 中，輸入 SNS 主題名稱。針對本教學，輸入 **high\_temp\_notice**。
  - f. 向下捲動到頁面底部，並選擇 Create topic (建立主題)。

主控台會開啟新主題的 Details (詳細資訊) 頁面。

2. 建立 Amazon SNS 訂閱。

### Note

您在此訂閱中使用的電話號碼可能會因您將在本教學課程中傳送訊息而產生簡訊費用。

- a. 於 high\_temp\_notice 主題詳細資訊頁面中，選擇 Create subscription (建立訂閱)。
  - b. 於 Create subscription (建立訂閱) 的 Details (詳細資訊) 區段下，在 Protocol (通訊協定) 清單中選擇 SMS。
  - c. 於 Endpoint (端點) 中，輸入可接收簡訊的電話號碼。請務必將其輸入，使其以 + 開頭，包含國碼和地區碼，且不包含任何其他標點符號字元。
  - d. 選擇建立訂閱。
3. 測試 Amazon SNS 通知。
    - a. 於 [Amazon SNS 主控台](#) 的左側導覽窗格中，選擇 Topics (主題)。
    - b. 若要開啟主題的詳細資料頁面，請於 Topics (主題) 的主題清單中，選擇 high\_temp\_notice。

- c. 如要開啟 Publish message to topic (將訊息發佈至主題) 頁面，請於 high\_temp\_notice 詳細資訊頁面中，選擇 Publish message (發佈訊息)。
- d. 在 Publish message to topic (將訊息發佈至主題) 的 Message body (訊息內文) 區段下，在 Message body to send to the endpoint (要傳送至端點的訊息內文) 中，輸入簡短訊息。
- e. 捲動到頁面底部，並選擇 Publish message (發佈訊息)。
- f. 在您先前建立訂閱時所使用的手機號碼上，確認已收到訊息。

若您並未收到測試訊息，請再次檢查電話號碼及手機設定。

請確保您可從 [Amazon SNS 主控台](#) 發佈測試訊息，然後再繼續教學課程。

## 步驟 2：創建 AWS IoT 規則以發送短信

您將在本教學課程中建立的 AWS IoT 規則會訂閱 device/*device\_id*/data MQTT 主題，其中 *device\_id* 是傳送訊息之裝置的 ID。主題篩選條件會將這些主題描述為 device/+ /data，其中 + 為與兩個正斜線字元間之任何字串相符的萬用字元。此規則也會測試訊息裝載中的 temperature 欄位值。

當規則收到來自相符主題的訊息時，其會採用來自主題名稱的 *device\_id*、來自訊息承載的 temperature 值、及為其測試的限制新增一個常數值，並將這些值做為 JSON 文件傳送至 Amazon SNS 通知主題。

例如，來自天氣感應器裝置編號 32 的 MQTT 訊息會使用 device/32/data 主題，並具有看起來如下的訊息承載：

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

規則的規則查詢陳述式採用訊息承載的 temperature 值、主題名稱的 *device\_id*，並新增常數 max\_temperature 值，以將看起來像這樣的訊息承載傳送至 Amazon SNS 主題：

```
{
```

```
"device_id": "32",
"reported_temperature": 38,
"max_temperature": 30
}
```

建立 AWS IoT 規則以偵測超限溫度值，並建立要傳送至 Amazon SNS 主題的資料

1. 開啟[主 AWS IoT 控制台](#)的 **[規則]** 中樞。
2. 若此為您的第一個規則，請選擇 **Create (建立)** 或 **Create a rule (建立規則)**。
3. 在 **Create a rule (建立規則)** 中：

- a. 在 **Name (名稱)** 中，輸入 **temp\_limit\_notify**。

請記住，規則名稱在您 AWS 帳戶 和區域中必須是唯一的，並且不能有任何空格。我們在此名稱中使用底線字元來分隔規則名稱中的字詞。

- b. 在 **Description (說明)** 中，說明規則。

有意義的說明可讓您更容易記住此規則的作用及您建立規則的原因。說明可依所需而定，因此請盡量詳細說明。

4. 在 **Create a rule (建立規則)** 的 **Rule query statement (規則查詢陳述式)** 中：
  - a. 在使用 SQL 版本中，選取 2016-03-23。
  - b. 在 **Rule query statement (規則查詢陳述式)** 編輯方塊中輸入陳述式：

```
SELECT topic(2) as device_id,
       temperature as reported_temperature,
       30 as max_temperature
FROM 'device/+/data'
WHERE temperature > 30
```

本陳述式：

- 聆聽主題與 `device/+/data` 主題篩選條件相符及 `temperature` 值大於 30 的 MQTT 訊息。
- 從主題字串中選取第二個元素，並將其指定給 `device_id` 欄位。
- 從訊息承載選取值 `temperature` 欄位，並將其指派給 `reported_temperature` 欄位。
- 建立常數值 30 來表示限制值，並將其指定給 `max_temperature` 欄位。

5. 若要開啟此規則的規則動作清單，請於 Set one or more actions (設定一個或多個動作) 中，選擇 Add action (新增動作)。
6. 在 Select an action (選取動作) 中，選擇 Send a message as an SNS push notification (傳送 SNS 推送通知形式的訊息)。
7. 若要開啟所選取動作的組態頁面，請在動作清單底部選擇 Configure action (設定動作)。
8. 於 Configure action (設定動作)：
  - a. 於 SNS target (SNS 目標) 中，選擇 Select (選取)，找出您名為 high\_temp\_notice 的 SNS 主題，然後選擇 Select (選取)。
  - b. 在 Message format (訊息格式) 中，選擇 RAW。
  - c. 在 [選擇或建立角色以授與執行此動作的 AWS IoT 存取權限] 中，選擇 [建立角色]。
  - d. 在 Create a new role (建立新角色) 的 Name (名稱) 中，輸入新角色的唯一名稱。在本教學課程中，使用 **sns\_rule\_role**。
  - e. 選擇 Create Role (建立角色)。

若您要重複此教學課程或重複使用現有的角色，請先選擇 Update role (更新角色)，再繼續進行。此會更新角色的政策文件，以使用 SNS 目標。

9. 選擇 Add action (新增動作)，並返回 Create a rule (建立規則) 頁面。

在新動作的圖標中，Send a message as an SNS push notification (以 SNS 推送通知形式傳送訊息)，您可看到規則呼叫的 SNS 主題。

這是您將新增至此規則的唯一規則動作。

10. 如要建立並完成此步驟，請於 Create a rule (建立規則) 中，向下捲動至底部，然後選擇 Create rule (建立規則)。

### 步驟 3：測試 AWS IoT 規則和 Amazon SNS 通知

若要測試新規則，您將使用 MQTT 用戶端來發佈和訂閱此規則所使用的 MQTT 訊息。

在新視窗的 [AWS IoT 主控台中開啟 MQTT 用戶端](#)。這可讓您編輯規則，而不會遺失 MQTT 用戶端的組態。如果您讓 MQTT 用戶端前往主控台的另一個頁面，其不會保留任何訂閱或訊息記錄。

如要使用 MQTT 用戶端來測試您的規則。

1. 在 AWS IoT 主控台的 [MQTT 用戶端](#) 中，訂閱輸入主題，在此案例中為 device/+ /data。



- a. 在 MQTT 用戶端中的 Subscriptions (訂閱) 下選擇 Subscribe to a topic (訂閱主題)。
- b. 在 Subscription topic (訂閱主題) 中，輸入輸入主題篩選條件 **device/+/data** 的主題。
- c. 將剩下的欄位保留為其預設設定。
- d. 請選擇 Subscribe to topic (訂閱主題)。

在 Subscriptions (訂閱) 欄中，Publish to a topic (發佈到主題) 之下，**device/+/data** 隨即顯示。

2. 使用特定裝置 ID **device/32/data**，將訊息發佈至輸入主題。您無法發佈至包含萬用字元的 MQTT 主題。
  - a. 在 MQTT 用戶端中的 Subscriptions (訂閱) 下選擇 Publish to topic (發佈至主題)。
  - b. 在 Publish (發佈) 欄位中輸入輸入主題名稱 **device/32/data**。
  - c. 複製此處顯示的範例資料，並在主題名稱下方的編輯方塊中貼上範例資料。

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. 選擇 Publish to topic (發佈至主題)，發佈您的 MQTT 訊息。

3. 確認簡訊已傳送。

- a. 在 MQTT 用戶端中 Subscriptions (訂閱) 下，您先前訂閱的主題旁會有一個綠點。

該綠點表示自上次查看後，已收到一個或多個新訊息。

- b. 於 Subscriptions (訂閱) 下，選擇 **device/+/data**，來檢查訊息承載是否與您剛剛發佈的內容相符，如下所示：

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
```

```
"velocity": 22,  
  "bearing": 255  
}  
}
```

- c. 檢查您用來訂閱 SNS 主題的手機，並確認訊息承載內容如下所示：

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

請注意，`device_id` 值是個帶有引號的字串，而 `temperature` 值是個數字。這是因為 `topic()` 函數從輸入訊息的主題名稱中提取字串，而 `temperature` 值會使用輸入訊息承載的數值。

若要將 `device_id` 值設為數值，請將規則查詢陳述式中的 `topic(2)` 替換為：

```
cast(topic(2) AS DECIMAL)
```

請注意，將 `topic(2)` 值轉換為數值，`DECIMAL` 值僅適用於該部分主題僅包含數字字元時。

4. 請嘗試傳送溫度不超過限制的 MQTT 訊息。
- 在 MQTT 用戶端中的 Subscriptions (訂閱) 下選擇 Publish to topic (發佈至主題)。
  - 在 Publish (發佈) 欄位中輸入輸入主題名稱 **device/33/data**。
  - 複製此處顯示的範例資料，並在主題名稱下方的編輯方塊中貼上範例資料。

```
{  
  "temperature": 28,  
  "humidity": 80,  
  "barometer": 1013,  
  "wind": {  
    "velocity": 22,  
    "bearing": 255  
  }  
}
```

- d. 若要傳送 MQTT 訊息，請選擇 Publish to topic (發佈至主題)。

您應該能看到您在 **device/+/data** 訂閱中傳送的訊息。不過，因溫度值低於規則查詢陳述式中的最高溫度，所以您不應收到簡訊。

若您並未看到正確的行為，請查看疑難排解提示。

## 疑難排解 SNS 訊息規則

若您並未看到預期的結果，請查看以下事項。

- 您收到錯誤的橫幅

若在您發佈輸入訊息時出現錯誤，請先更正該錯誤。下列步驟可協助您修正該錯誤。

- 您並未在 MQTT 用戶端中看到輸入訊息

每次您將輸入訊息發佈至 `device/22/data` 主題中，若您依程序中所述訂閱了 `device/+data` 主題篩選條件，則該訊息應會顯示於 MQTT 用戶端中。

### 要檢查的事項

- 檢查您訂閱的主題篩選條件

若您依程序中所述訂閱了輸入訊息主題，則每次發佈輸入訊息時都應該會看到其複本。

若您並未訊息，請檢查您訂閱的主題名稱，並將其與所發佈的主題進行比較。主題名稱區分大小寫，且您訂閱的主題必須與所發佈訊息承載的主題相同。

- 檢查訊息發佈功能

在 MQTT 用戶端中的 Subscriptions (訂閱) 下，選擇 `device/+data`，檢查發佈訊息的主題，然後選擇 Publish to topic (發佈至主題)。您應該會在訊息清單中出現主題下方的編輯方塊中看到訊息承載。

- 您並未收到 SMS 訊息：

若要讓您的規則運作，其必須具有授權其接收訊息和傳送 SNS 通知的正確政策，且必須接收訊息。

### 要檢查的事項

- 檢查您 AWS 區域的 MQTT 用戶端和您建立的規則

您正在執行 MQTT 用戶端的主控制台必須與您建立的規則處於相同的 AWS 區域。

- 檢查訊息承載中的溫度值是否超過測試閾值

若溫度值小於或等於 30 (如規則查詢陳述式中所定義)，則規則將不會執行其任何動作。

- 檢查規則查詢陳述式中的輸入訊息主題

若要讓規則運作，其必須收到一則訊息，其主題名稱與規則查詢陳述式之 FROM 子句中的主題篩選條件相符。

檢查規則查詢陳述式中主題篩選條件的拼字與 MQTT 用戶端中主題的拼字。主題名稱區分大小寫，且郵件的主題必須與規則查詢陳述式中的主題篩選條件相符。

- 檢查輸入訊息承載的內容

若要讓規則運作，其必須在 SELECT 陳述式中宣告的訊息承載中尋找資料欄位。

檢查規則查詢陳述式中 temperature 欄位的拼字與 MQTT 用戶端中訊息承載的拼字。欄位名稱區分大小寫，規則查詢陳述式中的 temperature 欄位必須與訊息承載中的 temperature 欄位相符。

請確定訊息承載中的 JSON 文件格式正確。若 JSON 有任何錯誤，例如缺少逗號，則規則將無法進行讀取。

- 檢查規則動作中重新發佈的訊息主題

Republish (重新發佈) 規則動作發佈新訊息的主題必須與您在 MQTT 用戶端中訂閱的主題相符。

開啟您建立於主控台的主題，並檢查規則動作重新發佈訊息的主題。

- 檢查規則所使用的角色

規則動作必須具有接收原始主題及發佈新主題的權限。

授權規則接收訊息資料的政策並加以重新發佈為所使用的主題所特定的。若變更新用於重新發佈訊息資料的主題，則必須更新規則動作的角色，來更新其政策以與目前主題相符。

若您懷疑這會是問題，請編輯 Republish (重新發佈) 規則動作並建立新角色。規則動作建立的新角色會收到執行這些動作所需的授權。

#### 步驟 4：檢閱結果及後續步驟

於本教學課程中：

- 您已建立並測試了 Amazon SNS 通知主題和訂閱。
- 您在規則查詢陳述式中使用了一個簡單的 SQL 查詢和函數，可為您的通知建立一個新訊息。
- 您已建立 AWS IoT 規則來傳送使用自訂訊息承載的 Amazon SNS 通知。
- 您使用 MQTT 用戶端來測試您的 AWS IoT 規則。

## 後續步驟

使用此規則傳送一些簡訊之後，請嘗試使用其來查看教學課程的某些層面如何影響訊息，及訊息傳送的時間。此處有幾種簡單的入門方式。

- 變更輸入訊息主題中的 `device_id`，並觀察簡訊內容中的影響。
- 變更規則查詢陳述式中所選取的欄位，並觀察簡訊內容中的影響。
- 變更規則查詢陳述式中的測試，以測試最低溫度，而非最高溫度。請記得變更 `max_temperature` 的名稱！
- 新增重新發佈規則動作，以於傳送 SNS 通知時傳送 MQTT 訊息。
- 請嘗試本系列中的下一個教學課程，並了解如何進行 [教學課程：將裝置資料儲存在 DynamoDB 表格中](#)。

## 教學課程：將裝置資料儲存在 DynamoDB 表格中

本教學課程示範如何建立將訊息資料傳送至 DynamoDB 表的 AWS IoT 規則。

在此教學課程中，您會建立一個規則，將訊息資料從虛構天氣感應器裝置傳送至 DynamoDB 表格。該規則會格式化來自多個天氣感應器的資料，由此可將其新增至單個資料庫表格中。

您會在本教學課程中學到什麼

- 如何建立 DynamoDB 表格
- 如何從規則將訊息資料傳送至 DynamoDB 表格 AWS IoT
- 如何在 AWS IoT 規則中使用替代範本
- 如何在規則查詢陳述式中使用簡單的 SQL 查詢和函數
- 如何使用 MQTT 用戶端測試規則 AWS IoT

此教學課程約需 30 分鐘方能完成。

於本教學課程中，您將會：

- [步驟 1：為本教學課程建立 DynamoDB 表格](#)
- [步驟 2：建立 AWS IoT 規則以將資料傳送至 DynamoDB 表格](#)
- [步驟 3：測試 AWS IoT 規則和 DynamoDB 資料表](#)
- [步驟 4：檢閱結果及後續步驟](#)

開始本教學課程之前，請確定您有：

- [設定您的 AWS 帳戶](#)

您需要您 AWS 帳戶 和 AWS IoT 主控台才能完成本教學課程。

- 檢閱 [使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)

請確定您可使用 MQTT 用戶端來訂閱並發佈至主題。您會使用 MQTT 用戶端，在此程序中測試您的新規則。

- 檢閱了 [Amazon DynamoDB 概觀](#)

若您未曾使用過 DynamoDB，請查閱 [DynamoDB 入門](#)，以熟悉 DynamoDB 的基本概念和作業。

### 步驟 1：為本教學課程建立 DynamoDB 表格

於本教學課程中，您會建立一個具下列屬性的 DynamoDB 表格，以記錄來自虛構氣候感應器裝置的資料：

- `sample_time` 為主索引鍵，並說明記錄範例的時間。
- `device_id` 為排序索引鍵，並說明提供範例的裝置
- `device_data` 為從裝置接收並由規則查詢陳述式格式化的資料

### 如要建立本教學課程的 DynamoDB 表格

1. 開啟 [DynamoDB 主控台](#)，然後選擇 Create table (建立表格)。
2. 在 Create table (建立資料表) 中：
  - a. 於 Table name (表格名稱) 中，輸入表格名稱：`wx_data`。
  - b. 在 Primary key (主索引鍵) 中，輸入 `sample_time`，然後在欄位旁的選項清單中，選擇 **Number**。
  - c. 在 Sort key (排序索引鍵) 中，輸入 `device_id`，然後在欄位旁的選項清單中，選擇 **Number**。
  - d. 請在頁面底部，選擇 Create (建立)。

您稍後將在設定 DynamoDB 規則動作時定義 `device_data`。

## 步驟 2：建立 AWS IoT 規則以將資料傳送至 DynamoDB 表格

於此步驟中，您會使用規則查詢陳述式，格式化虛構天氣感應器裝置的資料，以寫入資料庫表格。

從天氣感應器裝置接收的訊息承載範例如下所示：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

對於資料庫項目，您會使用規則查詢陳述式將訊息承載的結構平面化，如下所示：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind_velocity": 22,
  "wind_bearing": 255
}
```

於此規則中，您還會使用一些 [替代範本](#)。替代範本是可讓您從函數和訊息資料插入動態值的運算式。

若要建立將資料傳送至 DynamoDB 表格的 AWS IoT 規則

1. 開啟 [AWS IoT 主控台的 Rules \(規則\) 中樞](#)。或者，您可以在中開啟 AWS IoT 首頁，AWS Management Console 然後瀏覽至 [郵件路由] > [規則]。
2. 如要在 Rules (規則) 中開始建立新規則，請選擇 Create rule (建立規則)。
3. 在 Rule properties (規則屬性) 中：
  - a. 在 Rule name (規則名稱) 中，輸入 **wx\_data\_ddb**。

請記住，規則名稱在您 AWS 帳戶 和區域中必須是唯一的，並且不能有任何空格。我們在此名稱中使用底線字元來分隔規則名稱中的兩個單字。

- b. 在 Rule description (規則說明) 中，說明規則。

有意義的說明可讓您更容易記住此規則的作用及您建立規則的原因。說明可依所需而定，因此請盡可能詳細說明。


4. 選擇 Next (下一步) 繼續。
5. 在 SQL statement (SQL 陳述式) 中：
  - a. 在 SQL version (SQL 版本) 中，選取 **2016-03-23**。
  - b. 在 SQL statement (SQL 陳述式) 編輯方塊中，輸入陳述式：

```
SELECT temperature, humidity, barometer,  
       wind.velocity as wind_velocity,  
       wind.bearing as wind_bearing,  
FROM 'device/+/data'
```

本陳述式：

- 聆聽具與 device/+/data 主題篩選條件相符之主題的 MQTT 訊息。
- 將 wind 屬性的元素格式化為個別屬性。
- 傳遞 temperature、humidity 和 barometer 屬性不變。

6. 選擇 Next (下一步) 繼續。
7. 在 Rule actions (規則動作) 中：
  - a. 若要開啟此規則的規則動作清單，請於 Action 1 (動作 1) 中，選擇 **DynamoDB**。

 Note

請確定您選擇的是 DynamoDB 而非 dynamodb 做為規則動作。

- b. 於 Table name (表格名稱) 中，選擇您在先前步驟中建立的 DynamoDB 表格名稱：**wx\_data**。

Partition key type (分割區索引鍵類型) 和 Sort key type (排序索引鍵類型) 欄位，都會填入 DynamoDB 表格中的值。

- c. 在 Partition key (分區索引鍵)，輸入 **sample\_time**。
- d. 在 Partition key value (分割區索引鍵值) 中，輸入 **\${timestamp()}**。

這是您將用於此規則中 [替代範本](#) 的首項。而非使用訊息承載中的值，其會使用 timestamp 函數傳回的值。如需進一步了解，請參閱 AWS IoT Core 開發人員指南中的 [時間戳記](#)。



- e. 在 Sort key (排序索引鍵) 中，輸入 **device\_id**。
- f. 在 Sort key value (排序索引鍵值) 中，輸入 **`${cast(topic(2) AS DECIMAL)}`**。

這是您將用於此規則中 [替代範本](#) 第二項。其會在主題名稱中插入第二個元素值，即裝置的 ID，然後將其轉換為 DECIMAL 值，以與索引鍵的數值格式相符。若要進一步了解主題，請參閱 AWS IoT Core 開發人員指南中的 [主題](#)。或者，若要了解將值轉換為數值的詳細資訊，請參閱 AWS IoT Core 開發人員指南中的 [將值轉換為數值](#)。

- g. 在 Write message data to this column (寫入訊息資料至此欄) 中輸入 **device\_data**。

這會建立 DynamoDB 表格中的 `device_data` 欄。

- h. 將 Operation (操作) 保持空白。
  - i. 在 IAM Role (IAM 角色) 中，選擇 Create a new Role (建立新角色)。
  - j. 在 Create role (建立角色) 對話方塊中，請為 Role name (角色名稱) 輸入 `wx_ddb_role`。這個新角色會自動包含前置詞為 "aws-iot-rule" 的原則，允許 `wx_data_ddb` 規則將資料傳送到您建立的 `wx_data` DynamoDB 表。
  - k. 在 IAM role (IAM 角色) 中，選擇 **`wx_ddb_role`**。
  - l. 請選擇頁面最下方的 Next (下一頁)。
8. 請在 Review and create (檢閱和建立) 頁面底部，選擇 Create (建立) 來建立規則。

### 步驟 3：測試 AWS IoT 規則和 DynamoDB 資料表

若要測試新規則，您會使用 MQTT 用戶端來發佈和訂閱用於此測試的 MQTT 訊息。

在新視窗的 [AWS IoT 主控台中開啟 MQTT 用戶端](#)。這可讓您編輯規則，而不會遺失 MQTT 用戶端的組態。如果您讓它轉到主控台的另一個頁面，MQTT 用戶端不會保留任何訂閱或訊息記錄。您還需要一個單獨的主控台視窗開啟到主控 [AWS IoT 台中的 DynamoDB 表格中樞](#)，以檢視規則傳送的新項目。

如要使用 MQTT 用戶端來測試您的規則。

1. 於 [AWS IoT 主控台的 MQTT 用戶端](#) 中，訂閱輸入主題 `device/+ /data`。
  - a. 於 MQTT 用戶端中，選擇 Subscribe to a topic (訂閱主題)。
  - b. 若為 Topic filter (主題篩選條件)，請輸入輸入主題篩選條件的主題 **`device/+ /data`**。
  - c. 選擇 Subscribe (訂閱)。

2. 現在，使用特定裝置 ID **device/22/data** 將訊息發佈至輸入主題。您無法發佈至包含萬用字元的 MQTT 主題。
  - a. 於 MQTT 用戶端中，請選擇 Publish to a topic (發佈至主題)。
  - b. 若為 Topic name (主題名稱)，請輸入輸入主題名稱 **device/22/data**。
  - c. 若為 Message payload (訊息承載)，請輸入下列範例資料。

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. 如要發佈 MQTT 訊息，請選擇 Publish (發佈)。
  - e. 現在，於 MQTT 用戶端中，選擇 Subscribe to a topic (訂閱主題)。於 Subscribe (訂閱) 欄中，選擇 **device/+data** 訂閱。確認上一步的範例資料會顯示於此處。
3. 查看您規則所建立 DynamoDB 表格中的列。

- a. 在[AWS IoT 主控台的 DynamoDB 表格中樞](#)中，選擇 **wx\_data**，然後選擇 [項目] 索引標籤。

若您已在 Items (項目) 索引標籤上，您可能需要選擇表格標題右上角的重新整理圖示來重新整理該顯示。

- b. 請注意，表格中的 `sample_time` 值為連結並開啟一個。若您剛傳送了第一則訊息，則其將是清單中唯一的訊息。

此連結會顯示表格該列中的所有資料。

- c. 展開 `device_data` 項目，查看規則查詢陳述式所產生的資料。
  - d. 探索此顯示中可用資料的不同表示法。您亦可於此顯示中編輯資料。
  - e. 完成查閱此資料列之後，如要儲存所做的任何變更，請選擇 Save (儲存)，或者，如要結束而不儲存任何變更，請選擇 Cancel (取消)。

若您並未看到正確的行為，請查看疑難排解提示。

## 對您的 DynamoDB 規則進行疑難排解

若您並未看到預期的結果，請查看以下事項。

- 您收到錯誤的橫幅

若在您發佈輸入訊息時出現錯誤，請先更正該錯誤。下列步驟可協助您修正該錯誤。

- 您並未在 MQTT 用戶端中看到輸入訊息

每次您將輸入訊息發佈至 `device/22/data` 主題時，若您依程序中所述訂閱了 `device/+data` 主題篩選條件，則該訊息應會顯示於 MQTT 用戶端中。

### 要檢查的事項

- 檢查您訂閱的主題篩選條件

若您依程序中所述訂閱了輸入訊息主題，則每次發佈輸入訊息時都應該會看到其複本。

若您並未訊息，請檢查您訂閱的主題名稱，並將其與所發佈的主題進行比較。主題名稱區分大小寫，且您訂閱的主題必須與所發佈訊息承載的主題相同。

- 檢查訊息發佈功能

在 MQTT 用戶端中的 Subscriptions (訂閱) 下，選擇 `device/+data`，檢查發佈訊息的主題，然後選擇 Publish to topic (發佈至主題)。您應該會在訊息清單中出現主題下方的編輯方塊中看到訊息承載。

- 您在 DynamoDB 表格中看不到您的資料

首先要做的是選擇表格標題右上角的重新整理圖示，以重新整理顯示。若未顯示您正在尋找的資料，請檢查下列內容。

### 要檢查的事項

- 檢查您 AWS 區域的 MQTT 用戶端和您建立的規則

您正在執行 MQTT 用戶端的主控制台必須與您建立的規則處於相同的 AWS 區域。

- 檢查規則查詢陳述式中的輸入訊息主題

若要讓規則運作，其必須收到一則訊息，其主題名稱與規則查詢陳述式之 FROM 子句中的主題篩選條件相符。

檢查規則查詢陳述式中主題篩選條件的拼字與 MQTT 用戶端中主題的拼字。主題名稱區分大小寫，且郵件的主題必須與規則查詢陳述式中的主題篩選條件相符。

- 檢查輸入訊息承載的內容

若要讓規則運作，其必須在 SELECT 陳述式中宣告的訊息承載中尋找資料欄位。

檢查規則查詢陳述式中 temperature 欄位的拼字與 MQTT 用戶端中訊息承載的拼字。欄位名稱區分大小寫，規則查詢陳述式中的 temperature 欄位必須與訊息承載中的 temperature 欄位相符。

請確定訊息承載中的 JSON 文件格式正確。若 JSON 有任何錯誤，例如缺少逗號，則規則將無法進行讀取。

- 檢查用於規則動作中的索引鍵和欄位名稱

用於主題規則中的欄位名稱必須與已發佈訊息之 JSON 訊息承載中找到的欄位名稱相符。

開啟您建立於主控台內的規則，並檢查規則動作組態中的欄位名稱與用於 MQTT 用戶端中的欄位名稱。

- 檢查規則所使用的角色

規則動作必須具有接收原始主題及發佈新主題的權限。

授權規則以接收訊息資料及更新 DynamoDB 表格的政策是所使用主題特有的。若您變更規則使用的主題或 DynamoDB 表格名稱，則必須更新規則動作的角色，以更新其相符的政策。

若您懷疑這會是問題，請編輯規則動作並建立新角色。規則動作建立的新角色會收到執行這些動作所需的授權。

#### 步驟 4：檢閱結果及後續步驟

在您使用此規則將數則訊息傳送至 DynamoDB 表格後，請試著對其進行試驗，以查看變更教學課程中的某些層面如何影響寫入表格的資料。此處有幾種簡單的入門方式。

- 變更輸入訊息主題中的 *device\_id*，並觀察對資料的影響。您可使用此來模擬從多個天氣感應器接收資料。
- 變更規則查詢陳述式中所選取的欄位，並觀察對資料的影響。您可以此來篩選儲存於表格中的資料。
- 新增重新發佈規則動作，為新增至表格中的每一列傳送 MQTT 訊息。您可以此來進行除錯。

完成本教學課程後，請查看 [the section called “使用 AWS Lambda 函數格式化通知”](#)。

## 教學課程：使用 AWS Lambda 函數來格式化通知

本教學課程示範如何將 MQTT 訊息資料傳送至 AWS Lambda 動作，以便格式化並傳送至其他 AWS 服務。在本教學課程中，AWS Lambda 動作會使用 AWS SDK 將格式化的訊息傳送至您在教學課程中建立的 Amazon SNS 主題，以瞭解如何進行[the section called “正在傳送 Amazon SNS 通知”](#)。

於有關如何進行 [the section called “正在傳送 Amazon SNS 通知”](#) 的教學課程中，由規則查詢陳述式產生的 JSON 文件 作為簡訊的主文傳送。結果為一則看似此範例的簡訊：

```
{"device_id":"32","reported_temperature":38,"max_temperature":30}
```

在本教學課程中，您將使用 AWS Lambda 規則動作來呼叫 AWS Lambda 函數，將規則查詢陳述式中的資料格式化為更友善的格式，例如下列範例：

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

您將在本教學課程中建立的 AWS Lambda 函數會使用規則查詢陳述式中的資料來格式化訊息字串，並呼叫 AWS SDK 的 [SNS 發佈](#) 函數來建立通知。

您會在本教學課程中學到什麼

- 如何創建和測試 AWS Lambda 函數
- 如何在 AWS Lambda 函數中使用 AWS SDK 發佈 Amazon SNS 通知
- 如何在規則查詢陳述式中使用簡單的 SQL 查詢和函數
- 如何使用 MQTT 用戶端測試規則 AWS IoT

此教學課程約需 45 分鐘方能完成。

於本教學課程中，您將會：

- [步驟 1：創建一個發送短信的 AWS Lambda 函數](#)
- [步驟 2：使用 AWS IoT 規則動作建立 AWS Lambda 規則](#)
- [步驟 3：測試 AWS IoT 規則和規 AWS Lambda 則動作](#)
- [步驟 4：檢閱結果及後續步驟](#)

開始本教學課程之前，請確定您有：

- [設定您的 AWS 帳戶](#)

您需要您 AWS 帳戶 和 AWS IoT 主控台才能完成本教學課程。

- 檢閱 [使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)

請確定您可使用 MQTT 用戶端來訂閱並發佈至主題。您會使用 MQTT 用戶端，在此程序中測試您的新規則。

- 已完成本節中的其他規則教學課程

本教學課程需要您於教學課程中建立有關如何進行 [the section called “正在傳送 Amazon SNS 通知”](#) 的 SNS 通知主題。其亦假設您已完成本節中其他與規則相關的教學課程。

- 檢閱了 [AWS Lambda](#) 概觀

如果您 AWS Lambda 之前從未使用過，請檢閱[AWS Lambda](#)並[開始使用 Lambda](#) 以了解其術語和概念。

## 步驟 1：創建一個發送短信的 AWS Lambda 函數

本教學課程中的 AWS Lambda 函數會接收規則查詢陳述式的結果、將元素插入文字字串，然後將產生的字串做為通知中的訊息傳送至 Amazon SNS。

與有關如何[the section called “正在傳送 Amazon SNS 通知”](#)使用 AWS IoT 規則動作傳送通知的教學課程不同，本教學課程會使用 AWS SDK 函數從 Lambda 函數傳送通知。然而，用於本教學課程中的實際 Amazon SNS 通知主題與您在教學課程中有關如何進行 [the section called “正在傳送 Amazon SNS 通知”](#) 相同。

若要建立傳送文字訊息的 AWS Lambda 函數

1. 建立新 AWS Lambda 函數。

- a. 於 [AWS Lambda 主控台](#) 中，選擇 Create function (建立函數)。
- b. 於 Create function (建立函數) 中，選取 Use a blueprint (使用藍圖)。

搜尋並選取 **hello-world-python** 藍圖，然後選擇 Configure (設定)。

- c. 於基本資訊中：

- i. 在 Function name (函數名稱) 中，輸入此函數的名稱 **format-high-temp-notification**。
  - ii. 在 [執行角色] 中，選擇 [從 AWS 原則範本建立新角色]。
  - iii. 於 Role name (角色名稱) 中，輸入新角色 **format-high-temp-notification-role** 的名稱。
  - iv. 在 Policy templates - optional (政策範本：選用) 中，搜尋並選取 Amazon SNS publish policy (Amazon SNS 發佈政策)。
  - v. 選擇 建立函式。
2. 修改藍圖程式碼以進行格式化並傳送 Amazon SNS 通知。
- a. 創建函數後，您應該會看到format-high-temp-notification詳細信息頁面。若您未看到，請從 [Lambda 函數](#) 頁面上加以開啟。
  - b. 在format-high-temp-notification詳細資訊頁面中，選擇「組態」索引標籤，然後捲動至「函數程式碼」面板。
  - c. 於 Function code (函數程式碼) 視窗的 Environment (環境) 窗格中，選擇 Python 檔案 lambda\_function.py。
  - d. 於 Function code (函數程式碼) 視窗中，刪除藍圖中的所有原始程式碼，並以此程式碼進行取代。

```
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. 選擇部署。
3. 在新視窗中，從教學課程有關如何進行 [the section called “正在傳送 Amazon SNS 通知”](#) 中查詢您 Amazon SNS 主題的 Amazon 資源名稱 (ARN)。
    - a. 於新視窗中，開啟 [Amazon SNS 主控台的主題頁面](#)。
    - b. 於 Topics (主題) 頁面中，於 Amazon SNS 主題清單中尋找 high\_temp\_notice 通知主題。
    - c. 尋找 high\_temp\_notice 通知主題的 ARN，以用於下一個步驟中。
  4. 建立您 Lambda 函數的測試案例。
    - a. 在 [Lambda Functions](#) 頁面的 format-high-temp-notification 詳細資料頁面上，選擇頁面右上角的 [選取測試事件] (即使看起來已停用)，然後選擇 [設定測試事件]。
    - b. 於 Configure test event (設定測試事件) 中，選擇 Create new test event (建立新的測試事件)。
    - c. 於 Event name (事件名稱) 中，輸入 **SampleRuleOutput**。
    - d. 在 Event name (事件名稱) 下的 JSON 編輯器，請貼上此範例 JSON 文件。這是您的 AWS IoT 規則將傳送給 Lambda 函數的範例。



```
{
  "device_id": "32",
  "reported_temperature": 38,
  "max_temperature": 30,
  "notify_topic_arn": "arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice"
}
```

- e. 請參閱具有 high\_temp\_notice 通知主題之 ARN 的視窗，然後複製 ARN 值。
- f. 以您通知主題的 ARN 取代 JSON 編輯器中的 notify\_topic\_arn 值。

保持此視窗的開啟狀態，您可於建立 AWS IoT 規則時再次使用此 ARN 值。

- g. 選擇建立。
5. 以範例資料測試函數。
- a. 在 format-high-temp-notification 詳細資料頁面的頁面右上角，確認 SampleRuleOutput 出現在「測試」按鈕旁邊。若未顯示，請從可用的測試事件清單中進行選擇。
  - b. 如要將範例規則輸出訊息傳送至您的函數，請選擇 Test (測試)。

若函數和通知皆有效，您會在訂閱通知的手機上收到一則簡訊。

若手機上並未收到簡訊，請檢查作業結果。於 Function code (函數程式碼) 面板中的 Execution result (執行結果) 索引標籤中，查閱回應以找出發生的任何錯誤。在您的函數可將通知傳送至手機之前，請勿繼續進行下一個步驟。

## 步驟 2：使用 AWS IoT 規則動作建立 AWS Lambda 規則

於此步驟中，您會使用規則查詢陳述式，格式化虛構天氣感應器裝置的資料，以傳送至 Lambda 函數，該函數將會格式化並傳送簡訊。

從天氣裝置接收到的訊息承載裝置的範例如下所示：

```
{
  "temperature": 28,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

```
}
```

於此規則中，您會使用規則查詢陳述式來建立 Lambda 函數的訊息承載，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"
}
```

此包含 Lambda 函數格式化及傳送正確簡訊所需的所有資訊。

若要建立 AWS IoT 規則以呼叫 Lambda 函數

1. 開啟主 [AWS IoT 控制台](#) 的 [\[規則\]](#) 中樞。
2. 如要於 Rules (規則) 中開始建立新規則，請選擇 Create (建立)。
3. 在 Create a rule (建立規則) 的頂部中：

- a. 於 Name (名稱) 中，輸入規則名稱 **wx\_friendly\_text**。

請記住，規則名稱在您 AWS 帳戶 和區域中必須是唯一的，並且不能有任何空格。我們在此名稱中使用底線字元來分隔規則名稱中的兩個單字。

- b. 在 Description (說明) 中，說明規則。

有意義的說明可讓您更容易記住此規則的作用及您建立規則的原因。說明可依所需而定，因此請盡量詳細說明。

4. 在 Create a rule (建立規則) 的 Rule query statement (規則查詢陳述式) 中：
  - a. 在使用 SQL 版本中，選取 **2016-03-23**。
  - b. 在 Rule query statement (規則查詢陳述式) 編輯方塊中輸入陳述式：

```
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
```

本陳述式：

- 聆聽主題與 `device/+ /data` 主題篩選條件相符及 `temperature` 值大於 30 的 MQTT 訊息。
  - 從主題字串選取第二個元素，將其轉換為十進位數字，進而指派給 `device_id` 欄位。
  - 從訊息承載選取值 `temperature` 欄位的值，然後將其指派給 `reported_temperature` 欄位。
  - 建立常數值 30 來表示限制值，並將其指定給 `max_temperature` 欄位。
  - 建立 `notify_topic_arn` 欄位的常數值。
- c. 請參閱具有 `high_temp_notice` 通知主題之 ARN 的視窗，然後複製 ARN 值。
  - d. 以您通知主題的 ARN 取代規則查詢陳述式編輯器中的 ARN 值 (`arn:aws:sns:us-east-1:57EXAMPLE833:high_temp_notice`)。
5. 於 Set one or more actions (設定一個或多個動作) 中：
    - a. 若要開啟此規則的規則動作清單，請選擇 Add action (新增動作)。
    - b. 於 Select an action (選取動作) 中，選擇 Send a message to a Lambda function (將訊息傳送至 Lambda 函數)。
    - c. 若要開啟所選取動作的組態頁面，請在動作清單底部選擇 Configure action (設定動作)。
  6. 於 Configure action (設定動作)：
    - a. 於 Function name (函數名稱) 中，選擇 Select (選取)。
    - b. 選擇 `format-high-temp-notification`。
    - c. 在 Configure action (設定動作) 的底部，選擇 Add action (新增動作)。
    - d. 如要建立規則，請於 Create a rule (建立規則) 中，選擇 Create rule (建立規則)。

### 步驟 3：測試 AWS IoT 規則和規 AWS Lambda 則動作

若要測試新規則，您將使用 MQTT 用戶端來發佈和訂閱此規則所使用的 MQTT 訊息。

在新視窗的 [AWS IoT 主控台中開啟 MQTT 用戶端](#)。您現在可以編輯規則，而不會遺失 MQTT 用戶端的設定。如果您讓 MQTT 用戶端前往主控台的另一個頁面，您將會遺失您的訂閱或訊息記錄。

如要使用 MQTT 用戶端來測試您的規則。

1. 在 AWS IoT 主控台的 [MQTT 用戶端](#) 中，訂閱輸入主題，在此案例中為 `device/+ /data`。
  - a. 在 MQTT 用戶端中的 Subscriptions (訂閱) 下選擇 Subscribe to a topic (訂閱主題)。

- b. 在 Subscription topic (訂閱主題) 中，輸入輸入主題篩選條件 **device/+/data** 的主題。
- c. 將剩下的欄位保留為其預設設定。
- d. 請選擇 Subscribe to topic (訂閱主題)。

在 Subscriptions (訂閱) 欄中，Publish to a topic (發佈到主題) 之下，**device/+/data** 隨即顯示。

2. 使用特定裝置 ID **device/32/data**，將訊息發佈至輸入主題。您無法發佈至包含萬用字元的 MQTT 主題。
  - a. 在 MQTT 用戶端中的 Subscriptions (訂閱) 下選擇 Publish to topic (發佈至主題)。
  - b. 在 Publish (發佈) 欄位中輸入輸入主題名稱 **device/32/data**。
  - c. 複製此處顯示的範例資料，並在主題名稱下方的編輯方塊中貼上範例資料。

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

- d. 如要發佈您的 MQTT 訊息，請選擇 Publish to topic (發佈至主題)。
3. 確認簡訊已傳送。
    - a. 在 MQTT 用戶端中 Subscriptions (訂閱) 下，您先前訂閱的主題旁會有一個綠點。

該綠點表示自上次查看後，已收到一個或多個新訊息。
    - b. 於 Subscriptions (訂閱) 下，選擇 **device/+/data**，來檢查訊息承載是否與您剛剛發佈的內容相符，如下所示：

```
{
  "temperature": 38,
  "humidity": 80,
  "barometer": 1013,
  "wind": {
    "velocity": 22,
    "bearing": 255
  }
}
```

```
}  
}
```

- c. 檢查您用來訂閱 SNS 主題的手機，並確認訊息承載內容如下所示：

```
Device 32 reports a temperature of 38, which exceeds the limit of 30.
```

若您變更訊息主題中的主題 ID 元素，請記得，僅當訊息主題中的該元素僅包含數字字元時，將 `topic(2)` 值轉換為數值才會有效。

4. 請嘗試傳送溫度不超過限制的 MQTT 訊息。
  - a. 在 MQTT 用戶端中的 Subscriptions (訂閱) 下選擇 Publish to topic (發佈至主題)。
  - b. 在 Publish (發佈) 欄位中輸入輸入主題名稱 **device/33/data**。
  - c. 複製此處顯示的範例資料，並在主題名稱下方的編輯方塊中貼上範例資料。

```
{  
  "temperature": 28,  
  "humidity": 80,  
  "barometer": 1013,  
  "wind": {  
    "velocity": 22,  
    "bearing": 255  
  }  
}
```

- d. 若要傳送 MQTT 訊息，請選擇 Publish to topic (發佈至主題)。

您應該會看到您在 **device/+ /data** 訂閱中傳送的訊息；但因溫度值低於規則查詢陳述式中的最高溫度，您不應收到簡訊。

若您並未看到正確的行為，請查看疑難排解提示。

## 疑難排解 AWS Lambda 規則和通知

若您並未看到預期的結果，請查看以下事項。

- 您收到錯誤的橫幅

若在您發佈輸入訊息時出現錯誤，請先更正該錯誤。下列步驟可協助您修正該錯誤。

- 您並未在 MQTT 用戶端中看到輸入訊息

每次您將輸入訊息發佈至 `device/32/data` 主題中，若您依程序中所述訂閱了 `device/+data` 主題篩選條件，則該訊息應會顯示於 MQTT 用戶端中。

#### 要檢查的事項

- 檢查您訂閱的主題篩選條件

若您依程序中所述訂閱了輸入訊息主題，則每次發佈輸入訊息時都應該會看到其複本。

若您並未訊息，請檢查您訂閱的主題名稱，並將其與所發佈的主題進行比較。主題名稱區分大小寫，且您訂閱的主題必須與所發佈訊息承載的主題相同。

- 檢查訊息發佈功能

在 MQTT 用戶端中的 Subscriptions (訂閱) 下，選擇 `device/+data`，檢查發佈訊息的主題，然後選擇 Publish to topic (發佈至主題)。您應該會在訊息清單中出現主題下方的編輯方塊中看到訊息承載。

- 您並未收到 SMS 訊息：

若要讓您的規則運作，其必須具有授權其接收訊息和傳送 SNS 通知的正確政策，且必須接收訊息。

#### 要檢查的事項

- 檢查您 AWS 區域的 MQTT 用戶端和您建立的規則

您正在執行 MQTT 用戶端的主控制台必須與您建立的規則處於相同的 AWS 區域。

- 檢查訊息承載中的溫度值是否超過測試閾值

若溫度值小於或等於 30 (如規則查詢陳述式中所定義)，則規則將不會執行其任何動作。

- 檢查規則查詢陳述式中的輸入訊息主題

若要讓規則運作，其必須收到一則訊息，其主題名稱與規則查詢陳述式之 FROM 子句中的主題篩選條件相符。

檢查規則查詢陳述式中主題篩選條件的拼字與 MQTT 用戶端中主題的拼字。主題名稱區分大小寫，且郵件的主題必須與規則查詢陳述式中的主題篩選條件相符。

- 檢查輸入訊息承載的內容

若要讓規則運作，其必須在 SELECT 陳述式中宣告的訊息承載中尋找資料欄位。

檢查規則查詢陳述式中 temperature 欄位的拼字與 MQTT 用戶端中訊息承載的拼字。欄位名稱區分大小寫，規則查詢陳述式中的 temperature 欄位必須與訊息承載中的 temperature 欄位相符。

請確定訊息承載中的 JSON 文件格式正確。若 JSON 有任何錯誤，例如缺少逗號，則規則將無法進行讀取。

- 檢查 Amazon SNS 通知

於 [步驟 1：建立一個傳送 SMS 簡訊的 Amazon SNS 主題](#) 中，請參閱步驟 3，說明如何測試 Amazon SNS 通知並測試通知以確保通知可正常運作。

- 檢查 Lambda 函數

於 [步驟 1：創建一個發送短信的 AWS Lambda 函數](#) 中，請參閱步驟 5，說明如何使用測試資料來測試 Lambda 函數，並測試 Lambda 函數。

- 檢查規則所使用的角色

規則動作必須具有接收原始主題及發佈新主題的權限。

授權規則接收訊息資料的政策並加以重新發佈為所使用的主題所特定的。若變更用於重新發佈訊息資料的主題，則必須更新規則動作的角色，來更新其政策以與目前主題相符。

若您懷疑這會是問題，請編輯 Republish (重新發佈) 規則動作並建立新角色。規則動作建立的新角色會收到執行這些動作所需的授權。

#### 步驟 4：檢閱結果及後續步驟

於本教學課程中：

- 您建立了一個 AWS IoT 規則來呼叫 Lambda 函數，該函數會傳送使用您自訂訊息承載的 Amazon SNS 通知。
- 您於規則查詢陳述式中使用了一個簡單的 SQL 查詢和函數，可為您的 Lambda 函數建立一個新訊息承載。
- 您使用 MQTT 用戶端來測試您的 AWS IoT 規則。

#### 後續步驟

使用此規則傳送一些簡訊之後，請嘗試使用其來查看教學課程的某些層面如何影響訊息，及訊息傳送的時間。此處有幾種簡單的入門方式。

- 變更輸入訊息主題中的 `device_id`，並觀察簡訊內容中的影響。
- 變更規則查詢陳述式中所選取的欄位、更新 Lambda 函數以於新訊息中加以使用，並觀察簡訊內容中的影響。
- 變更規則查詢陳述式中的測試，以測試最低溫度，而非最高溫度。更新 Lambda 函數以格式化新的訊息，並記得變更 `max_temperature` 的名稱。
- 若要深入瞭解如何尋找在開發和使用 AWS IoT 規則時可能發生的錯誤，請參閱[監控 AWS IoT](#)。

## 在裝置離線時保留裝置狀態

這些教學課程向您展示如何使用 AWS IoT Device Shadow 服務來儲存及更新裝置的狀態資訊。Shadow 文件是一個 JSON 文件，其根據裝置、本機應用程式或服務發佈的訊息，顯示裝置狀態的變化。於本教學課程中，Shadow 文件會顯示燈泡顏色的變化。這些教學課程還會顯示影子如何在裝置與網際網路中斷連線時，儲存此資訊，並在裝置回復連線並請求此資訊時，將最新狀態資訊傳回裝置。

建議您依此處顯示的順序試試這些教學課程，從您需要建立的 AWS IoT 資源和必要的硬體設定開始，這亦可協助您逐步學習概念。這些教學課程顯示如何配置及連接 Raspberry Pi 裝置，以與 AWS IoT 使用。若您並無所需的硬體，您可依照這些教學課程進行調整，以適應您選擇的裝置或[使用 Amazon EC2 建立虛擬裝置](#)。

### 教學課程案例概觀

這些教學課程的案例為本機應用程式或服務，可變更燈泡的顏色，及將其資料發佈預留的影子主題。這些教學課程類似於[互動式入門教學課程](#)中說明的 Device Shadow 功能，並在 Raspberry Pi 裝置實作。本節中的教學課程側重於單一經典影子裝置，同時展現如何容納已命名影子或多個裝置的方法。

下列教學課程將會協助您了解如何使用 AWS IoT Device Shadow 服務。

- [教學課程：準備好 Raspberry Pi 來執行影子應用程式](#)

本教學課程展示如何設定 Raspberry Pi 以與 AWS IoT 連線。您還會建立一個 AWS IoT 政策文件和物件資源，下載憑證，然後將政策連接至該物件資源。此教學課程約需 30 分鐘方能完成。

- [教學課程：安裝裝置 SDK 並執行 Device Shadows 的範例應用程式](#)



本教學課程會展示如何安裝所需的工具、軟體和 AWS IoT Python 的裝置 SDK，然後執行範例影子應用程式。本教學課程以 [連接 Raspberry Pi 或其他裝置](#) 中提出的概念為基礎，需 20 分鐘才能完成。

- [教學課程：使用範例應用程式和 MQTT 測試用戶端，與 Device Shadow 互動](#)

本教學課程將會展示如何使用 shadow.py 範例應用程式和 AWS IoT 主控台來觀察 AWS IoT Device Shadows 和燈泡狀態變化間的交互影響。本教學課程也會展示如何將 MQTT 訊息傳送至 Device Shadow 的預留主題。此教學課程約需 45 分鐘方能完成。

## AWS IoT Device Shadow 概觀

Device Shadow 是個裝置的持續虛擬表示法，由您在 AWS IoT 登錄檔中建立的 [物件資源](#) 所管理。Shadow 文件是個 JSON 或 JavaScript 標記法文件，用來存儲和檢索裝置的目前狀態資訊。您可透過 MQTT 或 HTTP REST API，使用影子來取得及設定裝置的狀態 (無論該裝置是否連線至網際網路)。

Shadow 的文件包含 state 屬性，說明裝置狀態的下列層面：

- desired：應用程式會透過更新 desired 物件來指定裝置屬性的所需狀態。
- reported：裝置會報告其在 reported 物件中的目前狀態。
- delta：AWS IoT 會報告 delta 物件中所需狀態與報告狀態之間的差異。

以下為 Shadow 狀態文件的範例：

```
{
  "state": {
    "desired": {
      "color": "green"
    },
    "reported": {
      "color": "blue"
    },
    "delta": {
      "color": "green"
    }
  }
}
```

如要更新裝置的 Shadow 文件，您可使用預留的 [MQTT 主題](#)、以 HTTP 支援 GET、UPDATE 和 DELETE 操作的 [Device Shadow REST API](#)，以及 [AWS IoT CLI](#)。

在上一個範例中，假設您想將 desired 顏色變更為 yellow。如要執行此動作，請傳送請求至 [UpdateThingShadow](#) API 或將訊息發佈至更新主題 `$aws/things/THING_NAME/shadow/update`。

```
{
  "state": {
    "desired": {
      "color": yellow
    }
  }
}
```

更新只會影響請求中所指定的欄位。成功更新 Device Shadow 後，AWS IoT 會發佈新 desired 狀態到 delta 主題 `$aws/things/THING_NAME/shadow/delta`。於此狀況下，Shadow 文件看起來像這樣：

```
{
  "state": {
    "desired": {
      "color": yellow
    },
    "reported": {
      "color": green
    },
    "delta": {
      "color": yellow
    }
  }
}
```

然後，使用 Update 主題 `$aws/things/THING_NAME/shadow/update` 及下列 JSON 訊息，對 AWS IoT Device Shadow 報告新的狀態：

```
{
  "state": {
    "reported": {
      "color": yellow
    }
  }
}
```

```
}
```

若您想要取得目前的狀態資訊，請將請求傳送至 [GetThingShadow](#) API 或將 MQTT 訊息發佈至 [取得主題](#) `$aws/things/THING_NAME/shadow/get`。

如需使用 Device Shadow 服務的相關資訊，請參閱 [AWS IoT Device Shadow 服務](#)。

如需在裝置、應用程式和服務中使用 Device Shadows 的詳細資訊，請參閱 [在裝置中使用影子](#) 和 [在應用程式和服務中使用影子](#)。

如需有關與 AWS IoT 影子交互影響的資訊，請參閱 [與影子互動](#)。

如需 MQTT 預留主題和 HTTP REST API 的相關資訊，請參閱 [Device Shadow MQTT 主題](#) 和 [Device Shadow REST API](#)。

## 教學課程：準備好 Raspberry Pi 來執行影子應用程式

本教學課程示範如何設定和配置 Raspberry Pi 裝置，並建立裝置連接和交換 MQTT 訊息所需的 AWS IoT 資源。

### Note

若您打算 [the section called “使用 Amazon EC2 建立虛擬裝置”](#)，則可跳過本頁，並繼續 [the section called “設定您的裝置”](#)。當您建立虛擬物件時，您將會建立這些資源。若您想使用不同的裝置，而非 Raspberry Pi，您可嘗試依照這些教學課程進行調整，使其適應您選擇的裝置。

於本教學課程中，您會了解如何：

- 設定一個 Raspberry Pi 裝置，並進行配置以與 AWS IoT 使用。
- 建立 AWS IoT 政策文件，其授權您的裝置與 AWS IoT 服務互動。
- 在 AWS IoT X.509 裝置憑證中建立物件資源，然後連接政策文件。

問題是您的裝置在 AWS IoT 登錄檔中的虛擬表示。憑證會對 AWS IoT 核心驗證您的裝置，而政策文件授權您的裝置與 AWS IoT 進行互動。

## 如何執行本教學課程

如要執行 Device Shadows 的 `shadow.py` 範例應用程式，您需要一個連接至 AWS IoT 的 Raspberry Pi 裝置。我們建議您依照此處顯示的順序學習本教學課程，從設定 Raspberry Pi 及其配件開始，然後

建立政策，並將政策連接至您建立的物件資源。然後，您可使用 Raspberry Pi 支援的圖形使用者介面 (GUI)，依照本教學課程開啟裝置 Web 瀏覽器上的 AWS IoT 主控台，這也使其更輕鬆地將憑證直接下載至您的 Raspberry Pi 以連接至 AWS IoT。

開始本教學課程之前，請確定您有：

- AWS 帳戶。若您沒有帳戶，請完成 [設定您的 AWS 帳戶](#) 所述的步驟，然後再繼續。您會需要 AWS 帳戶和 AWS IoT 主控台，才能完成此教學課程。
- Raspberry Pi 及其必要的配件。您會需要：
  - [Raspberry Pi 3 代 B 型](#) 或更新的型號。本教學課程可能適用於較早版本的 Raspberry Pi，但我們尚未對其進行測試。
  - [Raspberry Pi OS \(32 位元\)](#) 或更新版本。我們建議您使用最新版本的 Raspberry Pi 作業系統。較早版本的作業系統可能適用，但我們尚未對其進行測試。
  - 乙太網路或 Wi-Fi 連線。
  - 鍵盤、滑鼠、顯示器、纜線和電源供應器。

此教學課程約需 30 分鐘方能完成。

## 步驟 1：設定及配置 Raspberry Pi 裝置

在本節中，我們將會配置一個 Raspberry Pi 設備以 AWS IoT 使用。

### Important

調整這些指示以適用其他裝置和作業系統可能是一項艱鉅的挑戰。您必須充分了解您的裝置，才能解譯這些指示並將其套用至您的裝置。若遇到困難，您可嘗試使用其他裝置選項的一項作為替代選項，例如 [使用 Amazon EC2 建立虛擬裝置](#) 或 [使用您的視窗或 Linux 個人電腦或 Mac 作為 AWS IoT 裝置](#)。

您需要配置 Raspberry Pi，使其可以啟動作業系統 (OS)，連接至網際網路，並可讓您在命令列介面與其互動。您還可使用 Raspberry Pi 支援的圖形使用者介面 (GUI) 來開啟 AWS IoT 主控台，並執行本教學課程的其餘部分。

## 設定 Raspberry Pi

1. 將 SD 卡插入 Raspberry Pi 上的 MicroSD 記憶卡插槽。有些 SD 卡會預先載入安裝管理員，在啟動主機板後提示您安裝作業系統的選單。您也可使用 Raspberry Pi 成像器，在卡上安裝作業系統。

2. 將 HDMI 電視或顯示器連接至連接到 Raspberry Pi 之 HDMI 連接埠的 HDMI 纜線。
3. 將鍵盤和滑鼠連接至 Raspberry Pi 的 USB 連接埠，然後插入電源整流器以啟動機板。

Raspberry Pi 啟動後，若 SD 卡預先載入安裝管理員，會出現一個安裝作業系統的選單。若於安裝作業系統時發生問題，您可試試下列步驟。如需有關 Raspberry Pi 的設定資訊，請參閱[設定 Raspberry Pi](#)。

若您在設定 Raspberry Pi 時發生問題：

- 在啟動機板之前，請檢查是否已插入 SD 卡。若您在啟動機板後插入 SD 卡，則安裝選單可能不會顯示。
- 確認電視或顯示器已開啟，且已選取正確的輸入。
- 確保您正在使用 Raspberry Pi 相容的軟體。

在安裝和設定 Raspberry Pi 作業系統之後，請開啟 Raspberry Pi 的 web 瀏覽器，前往 AWS IoT Core 主控台，以繼續本教學課程中的其餘步驟。

如果您可以開啟 AWS IoT Core 主控台，表示 Raspberry Pi 已準備就緒，您可以繼續進行 [the section called “在 AWS IoT 中佈建裝置”](#)。

若您仍然無法解決問題或需要其他協助，請參閱[獲取有關 Raspberry Pi 的協助](#)。

## 教學課程：在 AWS IoT 中佈建裝置

本節會建立教學課程會使用的 AWS IoT Core 資源。

### 在 AWS IoT 中佈建裝置的步驟

- [步驟 1：建立 Device Shadow 的 AWS IoT 政策](#)
- [步驟 2：建立物件資源並將政策連接至該物件](#)
- [步驟 3：檢閱結果及後續步驟](#)

### 步驟 1：建立 Device Shadow 的 AWS IoT 政策

X.509 憑證使用 AWS IoT Core 驗證您的裝置。將 AWS IoT 政策連接至憑證，允許裝置執行 AWS IoT 作業，例如訂閱或發佈至 Device Shadow 服務使用的 MQTT 預留主題。您的裝置會在連接並將訊息傳送至 AWS IoT Core 時顯示憑證。

在此過程中，您將會建立一個政策，可讓您的裝置執行 AWS IoT 執行範例程式所需的作業。建議您建立政策，該政策僅授予執行任務所需的許可權。您先建立 AWS IoT 政策，然後將其連接至稍後建立的裝置憑證。

## 如要建立 AWS IoT 政策

1. 請在左側選單上選擇 Secure (安全)，然後選擇 Policies (政策)。若您的帳戶具有現有政策，請選擇 Create (建立)，否則，在 You don't have a policy yet (您尚未設定政策) 頁面上，選擇 Create a policy (建立政策)。
2. 在 Create policy (建立政策) 頁面上：
  - a. 在 Name (名稱) 欄位中，輸入政策的名稱 (例如，**My\_Device\_Shadow\_policy**)。請勿在政策名稱中使用個人識別資訊。
  - b. 在政策文件中，您說明了連線、訂閱、接收和發佈動作，這些動作授予裝置發佈和訂閱 MQTT 預留主題的許可權限。

複製下列範例政策，並將其貼入您的政策文件中：以您所建立物件的名稱 (例如 My\_light\_bulb) 替換 thingname，以您使用服務的 AWS IoT 區域替換 region，及以您的 AWS 帳戶編號替換 account。如需 AWS IoT 政策的詳細資訊，請參閱 [AWS IoT Core 政策](#)。

```
{
  "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-*"
  }
]
}
```

## 步驟 2：建立物件資源並將政策連接至該物件

連線至 AWS IoT 的裝置可由 AWS IoT 登錄檔中的物件資源展示。物件資源意指特定裝置或邏輯實體，例如本教學課程中的燈泡。

要了解如何在 AWS IoT 中建立物件，請遵循 [建立物件](#) 中所述的步驟。當您依照該教學課程中的步驟時，請注意以下一些關鍵事項：

1. 選擇 Create a single thing (建立單一物件)，在 Name (名稱) 欄位中，輸入與您先前建立政策時指定之 thingname 相同的物件名稱 (例如，My\_light\_bulb)。

物件類型建立之後，您就無法變更其名稱。若您給它一個不同於 thingname 的名字，請建立名為 thingname 的新物件並刪除舊物件。

#### Note

請勿在物件名稱中使用個人識別資訊。物件名稱可以出現在未加密的通訊和報告中。

2. 我們建議您將憑證已建立！頁面上的每個憑證檔案下載至您可輕鬆找到的位置。您必須安裝這些檔案，才能執行範例應用程式。

建議您將檔案下載至 Raspberry Pi home 目錄中的 certs 子目錄上，並使用如下表中所建議更簡單名稱對其進行命名。

#### 憑證檔案名稱

檔案	檔案路徑
根 CA 憑證	~/certs/Amazon-root-CA-1.pem
裝置憑證	~/certs/device.pem.crt
私有金鑰	~/certs/private.pem.key

3. 在您啟用憑證以啟用 AWS IoT 的連線後，請選擇 Attach a policy (連接政策)，並確定您已將先前建立的政策 (例如，My\_Device\_Shadow\_policy) 連接至該物件。

建立物件之後，您可看到顯示於 AWS IoT 主控台之物件清單中的物件資源。

### 步驟 3：檢閱結果及後續步驟

在本教學課程中，您會了解如何：

- 設定及配置 Raspberry Pi 裝置。
- 建立 AWS IoT 政策文件，其授權您的裝置以與 AWS IoT 服務互動。



- 建立物件資源和相關聯的 X.509 裝置憑證，並將政策文件與其進行連接。

## 後續步驟

您現在可以安裝適用於 Python 的 AWS IoT 裝置 SDK，執行 shadow.py 範例應用程式，並使用 Device Shadows 控制狀態。如需如何執行本教學課程的詳細資訊，請參閱 [教學課程：安裝裝置 SDK 並執行 Device Shadows 的範例應用程式](#)。

## 教學課程：安裝裝置 SDK 並執行 Device Shadows 的範例應用程式

本節顯示如何安裝必要的軟體和適用於 Python 的 AWS IoT 裝置 SDK，執行 shadow.py 範例應用程式來編輯 Shadow 文件並控制影子的狀態。

於本教學課程中，您會了解如何：

- 使用已安裝的軟體和適用於 Python 的 AWS IoT 裝置 SDK，來執行範例應用程式。
- 了解如何使用範例應用程式輸入值，以於 AWS IoT 主控台中發佈所需的值。
- 檢閱 shadow.py 範例應用程式，及其如何使用 MQTT 通訊協定來更新影子的狀態。

在您執行此教學課程之前：

您必須已設定您的 AWS 帳戶、配置您的 Raspberry Pi 裝置，並建立 AWS IoT 物件和政策，授予裝置發佈和訂閱 Device Shadow 服務之 MQTT 預留主題的許可權。如需更多詳細資訊，請參閱 [教學課程：準備好 Raspberry Pi 來執行影子應用程式](#)。

您還必須安裝 Git、Python 和適用於 Python 的 AWS IoT 裝置 SDK。本教學課程是以教學課程 [連接 Raspberry Pi 或其他裝置](#) 中提出的概念為基礎。若您尚未嘗試該教學課程，建議您依照該教學課程中說明的步驟，安裝憑證檔案和裝置 SDK，然後回到本教學課程，執行 shadow.py 範例應用程式。

於本教學課程中，您將會：

- [步驟 1：執行 shadow.py 範例應用程式](#)
- [步驟 2：檢閱 shadow.py 裝置 SDK 範例應用程式](#)
- [步驟 3：使用 shadow.py 範例應用程式進行疑難排解](#)
- [步驟 4：檢閱結果及後續步驟](#)

此教學課程約需 20 分鐘方能完成。

## 步驟 1：執行 shadow.py 範例應用程式

執行 shadow.py 範例應用程式之前，除了您所安裝之憑證檔案的名稱和位置之外，您還需要下列資訊。

### 應用程式參數值

參數	可在哪裡找到值
<i>your-iot-thing-name</i>	<p>您先前在 <a href="#">the section called “步驟 2：建立物件資源並將政策連接至該物件”</a> 中建立的 AWS IoT 物件名稱。</p> <p>如要尋找此值，請於 <a href="#">AWS IoT 主控台</a> 中，依序選擇 Manage (管理) 和 Things (物件)。</p>
<i>your-iot-endpoint</i>	<p><i>your-iot-endpoint</i> 值的格式為：<i>endpoint_id</i> -ats.iot.<i>region</i>.amazonaws.com，例如 a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com。如要尋找此值：</p> <ol style="list-style-type: none"> <li>於 <a href="#">AWS IoT 主控台</a> 中，依序選擇 Manage (管理) 和 Things (物件)。</li> <li>選擇您先前使用為裝置 My_light_bulb 建立的 IoT，然後選擇 Interact (互動)。在物件詳細資料頁面上，您的端點會顯示於 HTTPS 區段中。</li> </ol>

### 安裝並執行範例應用程式

1. 導覽至範例應用程式目錄。

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

2. 在命令列視窗中，按照指示替換 *your-iot-endpoint* 和 *your-iot-thing-name* 並執行此命令。

```
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. 觀察範例應用程式：

1. 連線至您帳戶的 AWS IoT 服務。
2. 訂閱 Delta 事件和 Update 與 Get 回應。
3. 提示您在終端機終輸入所需的值。
4. 顯示類似下列內容的輸出：

```
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

若在執行 shadow.py 範例應用程式時發生問題，請檢閱 [the section called “步驟 3：使用 shadow.py 範例應用程式進行疑難排解”](#)。如要取得可協助您修正問題的其他資訊，請將 `--verbosity debug` 參數新增至命令列，以便範例應用程式會顯示有關其正在執行之動作的詳細訊息。

### 輸入值並觀察 Shadow 文件中的更新

您可於終端機中輸入值，以指定 desired 值，其還會更新 reported 值。假設您在終端機中輸入顏色 yellow。reported 值也會更新為顏色 yellow。下列展示顯示於終端機中的訊息：

```
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'.
```

當您發佈此更新請求時，AWS IoT 會為物件資源建立一個預設經典影子。您可透過查看您所建立物件資源 (例如 `My_light_bulb`) 的 Shadow 文件，來觀察您發佈至 AWS IoT 主控台內的 `reported` 和 `desired` 值的更新請求。如要在 Shadow 文件中查看更新：

1. 於 AWS IoT 主控台中，依序選擇 `Manage` (管理) 和 `Things` (物件)。
2. 在顯示的物件清單中，選取您所建立的物件，然後依序選擇 `Shadows` (影子) 和 `Classic Shadow` (經典影子)。

Shadow 文件應看似如下列內容，顯示設定為顏色 `yellow` 的 `reported` 和 `desired` 值。您會在文件的 `Shadow state` (影子狀態) 區段中看到這些值。

```
{
  "desired": {
    "welcome": "aws-iot",
    "color": "yellow"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "yellow"
  }
}
```

您還會看到 `Metadata` (中繼資料) 區段，其包含請求之時間戳記資訊和版本號碼。

您可以使用狀態文件版本，確認您正在更新的裝置影子文件為最新版本。若您傳送另一個更新請求，版本編號會遞增 1。當您為更新請求提供版本，若狀態文件的目前版本與提供的版本不符，則服務會拒絕請求並顯示 HTTP 409 衝突回應代碼。

```
{
  "metadata": {
    "desired": {
      "welcome": {
        "timestamp": 1620156892
      },

```

```
    "color": {
      "timestamp": 1620156893
    }
  },
  "reported": {
    "welcome": {
      "timestamp": 1620156892
    },
    "color": {
      "timestamp": 1620156893
    }
  }
},
"version": 10
}
```

如要了解 Shadow 文件的更多資訊並觀察狀態資訊的變更，請繼續參閱說明於本教學課程 [步驟 4：檢閱結果及後續步驟](#) 部分中的下一個教學課程 [教學課程：使用範例應用程式和 MQTT 測試用戶端，與 Device Shadow 互動](#)。或者，您還可於下一節中了解 shadow.py 範本程式碼及其如何使用 MQTT 通訊協定。

## 步驟 2：檢閱 shadow.py 裝置 SDK 範例應用程式

本節會從用於本教學課程中之適用於 Python 的 AWS IoT 裝置 SDK v2 檢閱 shadow.py 範例應用程式。我們將會在此檢閱其如何使用 MQTT 和透過 WSS 的 MQTT 通訊協定連線至 AWS IoT Core。[AWS 通用執行時間 \(AWS-CRT\)](#) 程式庫提供低階通訊協定支援，並包含適用於 Python 的 AWS IoT 裝置 SDK v2。

在本教學課程使用 MQTT 和透過 WSS 的 MQTT 時，AWS IoT 會支援發佈 HTTPS 請求的裝置。有關從裝置傳送 HTTP 訊息的 Python 程式範例，請參閱使用 Python requests 程式庫的 [HTTPS 程式碼範例](#)。

如需有關如何決定裝置通訊所使用之通訊協定的詳細資訊，請參閱 [為您的裝置通訊選擇通訊協定](#)。

## MQTT

shadow.py 範例呼叫 [mqtt\\_connection\\_builder](#) 中的 `mtls_from_path` (如圖所示)，使用 MQTT 通訊協定來建立與 AWS IoT Core 的連線。`mtls_from_path` 會使用 X.509 憑證和 TLS v1.2 來驗證裝置。AWS-CRT 程式庫會處理該連線的較低階詳細資料。

```
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` 是您從命令列傳入的 AWS IoT 端點，`client_id` 是在 AWS 區域中唯一標識此裝置的 ID。
- `cert_filepath`、`pri_key_filepath` 和 `ca_filepath` 是裝置憑證和私密金鑰檔案及根 CA 檔案的路徑。
- `client_bootstrap` 是處理通訊端通訊活動的通用執行時間物件，並在呼叫 `mqtt_connection_builder.mtls_from_path` 之前實例化。
- `on_connection_interrupted` 和 `on_connection_resumed` 是在裝置連線遭到中斷並回復時呼叫的回呼函數。
- `clean_session` 為是否啟動一個新的、持續的工作階段，或若存在，則進行重新連線。`keep_alive_secs` 為 CONNECT 請求中傳送的保持活動值，以秒為單位。Ping 會在此時間間隔自動傳送。若在此值的 1.5 倍之後未收到 ping，則伺服器會假設連線中斷。

`shadow.py` 範例還會呼叫 [mqtt\\_connection\\_builder](#) 中的 `websockets_with_default_aws_signing`，使用透過 WSS 的 MQTT 通訊協定建立與 AWS IoT Core 的連線。透過 WSS 的 MQTT 還會使用與 MQTT 相同的參數，並採用下列其他參數：

- `region` 是 Signature V4 身分驗證使用的 AWS 簽名區域，`credentials_provider` 是提供用於身分驗證的 AWS 憑證。從命令列傳入 Region (區域)，並在呼叫 `mqtt_connection_builder.websockets_with_default_aws_signing` 之前將 `credentials_provider` 物件實例化。
- 若使用代理主機，`websocket_proxy_options` 是 HTTP 代理選項。在 `shadow.py` 範例應用程式中，此值在呼叫 `mqtt_connection_builder.websockets_with_default_aws_signing` 之前進行實例化。

## 訂閱 Shadow 主題和事件

`shadow.py` 範例嘗試建立連線，並等待完整連線。若未連線，則指令會排入佇列中。連線後，範例會訂閱差異事件及更新和取得訊息，並發佈服務品質 (QoS) 層級為 1 (`mqtt.QoS.AT_LEAST_ONCE`) 的訊息。

當裝置訂閱 QoS 層級 1 的訊息時，訊息代理程式會儲存裝置訂閱的訊息，直至其可傳送至裝置為止。訊息代理程式會重新傳送訊息，直至收到來自裝置的 PUBACK 回應。

如需 MQTT 通訊協定的詳細資訊，請參閱 [檢閱 MQTT 通訊協定](#) 和 [MQTT](#)。

如需用於本教學課程中的 MQTT、透過 WSS 的 MQTT、持續性工作階段及 QoS 層級的詳細資訊，請參閱 [檢閱 pubsub.py 裝置 SDK 範例應用程式](#)。

### 步驟 3：使用 `shadow.py` 範例應用程式進行疑難排解

當您執行 `shadow.py` 範例應用程式時，您應會看到一些顯示於終端機中的訊息，並提示輸入 `desired` 值。若程式拋出錯誤，則如要偵錯，您可先檢查是否為您的系統執行正確的命令。

在某些狀況下，錯誤訊息可能表示連線問題，且看似如下：`Host name was invalid for dns resolution` 或 `Connection was closed unexpectedly`。於此狀況下，您可檢查下列事項：

- 檢查命令中的端點的地址

請檢閱您為執行範例應用程式 (例如 `a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com`) 輸入命令中的 `endpoint` 引數，並在 AWS IoT 主控台中檢查此值。

如要檢查您是否使用了正確的值：

1. 於 AWS IoT 主控台中，依序選擇 **Manage (管理)** 和 **Things (物件)**。
2. 選擇您為範例應用程式建立的物件 (例如，`My_light_bulb`)，然後選擇 **Interact (互動)**。

在物件詳細資料頁面上，您的端點會顯示於 **HTTPS** 區段中。您應該也會看到訊息：`This thing already appears to be connected.`

- 檢查憑證啟用

憑證會以 AWS IoT Core 驗證您的裝置。

如要檢查您的憑證是否為作用中：

1. 於 AWS IoT 主控台中，依序選擇 **Manage (管理)** 和 **Things (物件)**。
2. 選擇您為範例應用程式建立的物件 (例如，`My_light_bulb`)，然後選擇 **Security (安全性)**。
3. 選取憑證，然後從憑證的詳細資料頁面選擇 **Select the certificate (選取憑證)**，然後從憑證的詳細資料頁面選擇 **Actions (動作)**。

若於下拉式清單中 Activate (啟用) 無法使用，則您只能選擇 Deactivate(停用)，表示您的憑證處於作用中。若無，請選擇 Activate (啟用)，然後重新執行範例程式。

若程式仍然無法執行，請檢查 certs 資料夾中的憑證檔案名稱。

- 檢查連接至該物件資源的政策

當憑證驗證您的裝置時，AWS IoT 政策允許裝置執行 AWS IoT 作業，例如訂閱或發佈至 MQTT 預留主題。

如要檢查是否已連接正確的政策：

1. 如先前所述尋找憑證，然後選擇 Policies (政策)。
2. 選擇顯示的政策，並檢查其是否說明了授予裝置發佈和訂閱 MQTT 預留主題之許可權限的 connect、subscribe、receive 和 publish 動作。

若是範例政策，請參閱 [步驟 1：建立 Device Shadow 的 AWS IoT 政策](#)。

若您看到表示無法連線至 AWS IoT 的錯誤訊息，這可能是因為您對政策使用的許可權限。若是這種情況，我們建議您先從提供完整存取 AWS IoT 資源的政策開始，然後重新執行範例程式。您可編輯目前的政策，或選擇目前的政策，選擇 Detach (分離)，然後建立另一個提供完整存取權的政策，並將其連接至您的物件資源。您可於稍後將政策限制為僅執行程式所需的動作和政策。

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

- 檢查您的裝置 SDK 安裝

若程式仍然無法執行，您可重新安裝裝置 SDK，以確定您的 SDK 安裝完整且正確。



## 步驟 4：檢閱結果及後續步驟

在本教學課程中，您會了解如何：

- 安裝必要的軟體、工具和適用於 Python 的 AWS IoT 裝置 SDK。
- 了解範例應用程式 `shadow.py` 會如何使用 MQTT 通訊協定來擷取和更新影子的目前狀態。
- 執行 Device Shadows 的範例應用程式，並觀察 AWS IoT 主控台內的 Shadow 文件的更新。您還會了解在執行程式時如何疑難排解任何問題並修復錯誤。

### 後續步驟

您現在可以執行 `shadow.py` 範例應用程式，並使用 Device Shadows 來控制狀態。您可以觀察 AWS IoT 主控台中 Shadow 文件的更新，並觀察範例應用程式回應的差異事件。使用 MQTT 測試用戶端，您可以訂閱預留的影子主題，並在執行範例程式時觀察主題所收到的訊息。如需如何執行本教學課程的詳細資訊，請參閱 [教學課程：使用範例應用程式和 MQTT 測試用戶端，與 Device Shadow 互動](#)。

### 教學課程：使用範例應用程式和 MQTT 測試用戶端，與 Device Shadow 互動

如要與 `shadow.py` 範例應用程式產生互動，請為 `desired` 值在終端機中輸入一個值。例如，您可指定類似交通號誌的顏色，而 AWS IoT 會回應該請求並更新回報的值。

於本教學課程中，您會了解如何：

- 使用 `shadow.py` 範例應用程式來指定所需的狀態並更新影子的目前狀態。
- 編輯 Shadow 文件，以觀察差異事件，及 `shadow.py` 範例應用程序如何加以回應。
- 使用 MQTT 測試用戶端訂閱影子主題，並在執行範例程式時觀察更新。

執行本教學課程之前，您必須具備：

設定您的 AWS 帳戶，配置您的 Raspberry Pi 裝置，並建立 AWS IoT 物件和政策。您也必須安裝了必要的軟體、裝置 SDK、憑證檔案，並在終端機中執行範例程式。如需詳細資訊，請參閱先前的教學課程 [教學課程：準備好 Raspberry Pi 來執行影子應用程式](#) 和 [步驟 1：執行 `shadow.py` 範例應用程式](#)。若您尚未完成，您必須完成這些教學課程。

於本教學課程中，您將會：

- [步驟 1：使用 `shadow.py` 範例應用程式更新所需和回報的值](#)
- [步驟 2：檢視 MQTT 測試用戶端中 `shadow.py` 範例應用程式的訊息](#)
- [步驟 3：疑難排解 Device Shadow 互動的錯誤](#)

## • [步驟 4：檢閱結果及後續步驟](#)

此教學課程約需 45 分鐘方能完成。

### 步驟 1：使用 `shadow.py` 範例應用程式更新所需和回報的值

在先前的教學課程 [步驟 1：執行 `shadow.py` 範例應用程式](#) 中，您已了解當您如本節 [教學課程：安裝裝置 SDK 並執行 Device Shadows 的範例應用程式](#) 所述輸入所需值時，如何觀察發佈至 AWS IoT 主控台之 Shadow 文件。

在上一個範例中，我們將所需的顏色設定為 `yellow`。輸入每個值後，終端機會提示您輸入另一個 `desired` 值。若您再次輸入相同的值 (`yellow`)，應用程式會辨識這一點，並提示您輸入新的 `desired` 值。

```
Enter desired value:
yellow
Local value is already 'yellow'.
Enter desired value:
```

現在假設您輸入顏色 `green`。AWS IoT 會回應該請求，並更新 `reported` 值為 `green`。當 `desired` 狀態與 `reported` 狀態不同而造成差異時，這就是更新發生了。

### `shadow.py` 範例應用程式如何模擬 Device Shadow 互動：

1. 於終端機中輸入 `desired` 值 (例如 `yellow`) 以發布所需的狀態。
2. 由於 `desired` 狀態與 `reported` 狀態 (例如，顏色 `green`) 不同，就會發生差異，而訂閱差異的應用程式會收到這個訊息。
3. 應用程式會回應訊息，並將其狀態更新為 `desired` 值，`yellow`。
4. 然後，應用程式會發佈更新訊息，其包含裝置狀態 `yellow` 的新回報值。

下列展示了終端機中顯示的訊息，其中顯示如何發佈更新請求。

```
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 AWS IoT 主控台中，Shadow 文件會將 reported 和 desired 兩個欄位的更新值反映為 green，且版本編號會遞增 1。例如，若先前的版本編號顯示為 10，則目前的版本編號會顯示為 11。

#### Note

刪除影子不會將版本編號重置為 0。當您發佈更新請求或建立另一個具相同名稱的陰子時，您會看到陰子版本會遞增 1。

### 編輯 Shadow 文件以觀察差異事件

shadow.py 範例應用程式也會訂閱 delta 事件，並在 desired 值發生變更時加以回應。例如，您可將 desired 值變更為顏色 red。如要進行，請於 AWS IoT 主控台中，按一下 Edit (編輯) 來編輯 Shadow 文件，然後在 JSON 中設定 desired 值為 red，同時保留 reported 值為 green。儲存變更之前，請讓 Raspberry Pi 上的終端機保持開啟狀態，因為您會在變更發生時看到顯示於終端機中的訊息。

```
{
  "desired": {
    "welcome": "aws-iot",
    "color": "red"
  },
  "reported": {
    "welcome": "aws-iot",
    "color": "green"
  }
}
```

儲存新值之後，shadow.py 範例應用程式會回應此變更，並在終端機中顯示指出差異的訊息。接著，您應該會在輸入 desired 值的提示下方看到下列訊息。

```
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'.
```

## 步驟 2：檢視 MQTT 測試用戶端中 `shadow.py` 範例應用程式的訊息

您可使用 AWS IoT 主控台內的 MQTT 測試用戶端，監控傳入您 AWS 帳戶中的 MQTT 訊息。訂閱 Device Shadow 服務使用的預留 MQTT 主題，您可在執行範例應用程式時觀察主題收到的訊息。

若您尚未使用 MQTT 測試用戶端，您可檢閱 [使用 MQTT 用戶端檢視 MQTT 訊息 AWS IoT](#)。這有助於您了解如何使用 AWS IoT 主控台內的 MQTT 測試用戶端，在 MQTT 訊息通過訊息代理程式時，檢視訊息。

### 1. 開啟 MQTT 測試用戶端

在新視窗開啟 [AWS IoT 主控台內的 MQTT 測試用戶端](#)，如此您便可觀察 MQTT 主題所收到的訊息，不會遺失 MQTT 測試用戶端的組態。若您讓 MQTT 用戶端前往主控台內另一個頁面，其不會保留任何訂閱或訊息記錄。在本教學課程的這一節中，您可在個別的視窗中開啟您 AWS IoT 物件的 Shadow 文件和 MQTT 測試用戶端，以便更輕鬆地觀察與 Device Shadows 的互動。

### 2. 訂閱 MQTT 預留的 Shadow 主題

您可使用 MQTT 測試用戶端輸入 Device Shadow 之 MQTT 預留主題的名稱，並加以訂閱，以於執行 `shadow.py` 範例應用程式時收到更新。如要訂閱主題：

- 於 AWS IoT 主控台的 MQTT 測試用戶端中，選擇 `Subscribe to a topic` (訂閱主題)。
- 於主題篩選條件區段中，輸入：`$aws/things/thingname/shadow/update/#`。在此，*thingname* 是您先前所建立物件資源的名稱 (例如，`My_light_bulb`)。
- 保留其他組態設定的預設值，然後選擇 `Subscribe` (訂閱)。

在主題訂閱中使用 # 萬用字元，您可同時訂閱多個 MQTT 主題，並在單一視窗中觀察裝置及其 Shadow 間交換的所有訊息。如需萬用字元及其用法的詳細資訊，請參閱 [MQTT 主題](#)。

### 3. 執行 `shadow.py` 範例程式並觀察訊息

在 Raspberry Pi 的命令列視窗中，若您已中斷程式的連線，請再次執行範例應用程式，並於 AWS IoT 主控台中查看 MQTT 測試用戶端中的訊息。

- 執行下列命令以重新啟動範例程式。以您之前建立的 AWS IoT 物件名稱替換 `your-iot-thing-name` 和 `your-iot-endpoint` (例如 `My_light_bulb`)，及與裝置互動的端點。

```
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 範例應用程式接著會執行並擷取目前的影子狀態。若您已刪除影子或清除目前狀態，該程式會將目前值設定為 off，然後提示您輸入 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，若程式正在執行，而您將其重新啟動，則會在終端機中看到回報的最新顏色值。於 MQTT 測試用戶端中，您會看到主題更新 \$aws/things/*thingname*/shadow/get 和 \$aws/things/*thingname*/shadow/get/accepted。

假設回報的最新顏色是 green。下列會顯示 \$aws/things/*thingname*/shadow/get/accepted JSON 檔案的內容。

```
{
  "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. 於終端機中輸入 desired 值，例如 yellow。shadow.py 範例應用程式會在終端機中回應並顯示下列訊息，顯示 reported 值變更為 yellow。

```
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'.
```

於 AWS IoTconsole (主控台) MQTT test client (MQTT 測試用戶端) 的 Subscriptions (訂閱) 之下，您會看到下列主題收到訊息：

- \$aws/things/**thingname**/shadow/update : 顯示 desired 和 updated 值會變更為顏色 yellow。
- \$aws/things/**thingname**/shadow/update/accepted : 顯示 desired 和 reported 狀態的目前值及其中繼資料和版本資訊。

- `$aws/things/thingname/shadow/update/documents` : 顯示 `desired` 和 `reported` 狀態的以前的值和目前值及其中繼資料和版本資訊。

作為文件 `$aws/things/thingname/shadow/update/documents` 也包含其他兩個主題中所包含的資訊，我們可以加以檢閱以查看狀態資訊。先前狀態會顯示設定為 `green` 的回報值、其中繼資料和版本資訊，目前狀態會顯示已更新為 `yellow` 的回報值。

```
{
  "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. 若您現在輸入另一個 `desired` 值，您會看到這些主題所收到 `reported` 值和訊息更新的進一步變更。版本編號也會遞增 1。例如，若您輸入值 `green` 時，先前的狀態會報告值 `yellow`，而目前狀態會報告值 `green`。

#### 4. 編輯 Shadow 文件以觀察差異事件

若要觀察差異主題的變更，請編輯 AWS IoT 主控台中的 Shadow 文件。例如，您可將 `desired` 值變更為顏色 `red`。若要這麼做，請於 AWS IoT 主控台中，選擇 `Edit` (編輯)，然後在 JSON 中設定 `desired` 值為紅色，同時保持 `reported` 值設定為 `green`。儲存變更之前，請保持終端開啟狀態，因為您會在終端機中看到回報的差異訊息。

```
{
```



```
"desired": {
  "welcome": "aws-iot",
  "color": "red"
},
"reported": {
  "welcome": "aws-iot",
  "color": "green"
}
}
```

shadow.py 範例應用程式會回應此變更，並在終端機中顯示指出差異的訊息。於 MQTT 測試用戶端中，update 主題將會收到一則訊息，顯示 desired 和 reported 值的變更。

您也會看到主題 \$aws/things/*thingname*/shadow/update/delta 收到訊息。如要查看訊息，請選擇此列於 Subscriptions (訂閱) 之下的主題。

```
{
  "version": 13,
  "timestamp": 1617318480,
  "state": {
    "color": "red"
  },
  "metadata": {
    "color": {
      "timestamp": 1617318480
    }
  }
}
```

### 步驟 3：疑難排解 Device Shadow 互動的錯誤

當您執行 Shadow 範例應用程式時，您可能會遇到觀察與 Device Shadow 服務互動的問題。

若程式執行成功，並提示您輸入 desired 值，您應可使用如前所述的 Shadow 文件和 MQTT 測試用戶端，觀察 Device Shadow 互動。不過，若您無法看到互動，您可檢查下列事項：

- 檢查 AWS IoT 主控台中物件名稱及其影子

若您看不到 Shadow 文件中的訊息，請檢閱命令，並確定其符合 AWS IoT 主控台中的物件名稱。您還可依序選擇您的物件名稱和 Shadows (影子)，來檢查您是否有經典影子。本教學課程主要著重於與經典影子的互動。

您也可以確認您使用的裝置已連線至網際網路。於 AWS IoT 主控台中，選擇您先前建立的物件，然後選擇 Interact (互動)。在物件詳細資訊頁面上，您應會在此處看到訊息：This thing already appears to be connected.

- 查看您訂閱的 MQTT 預留主題

若您在 MQTT 測試用戶端中看不到訊息，請檢查您訂閱的主題是否已正確格式化。MQTT Device Shadow 主題具有格式 `$aws/things/thingname/shadow/`，且可能具有 `update`、`get` 或 `delete`，根據您希望對影子執行的動作而定。本教學課程使用主題 `$aws/things/thingname/shadow/#`，因此，在訂閱測試用戶端之主題篩選條件部分中的主題時，請確定您已正確輸入。

當您輸入主題名稱時，請確定 *thingname* 與您先前建立的 AWS IoT 物件名稱相同。您還可訂閱其他 MQTT 主題，查看是否已成功執行更新。例如，您可訂閱主題 `$aws/things/thingname/shadow/update/rejected`，在更新請求失敗時收到訊息，您便可對連線問題進行偵錯。如需有關預留主題的更多資訊，請參閱 [the section called “影子主題”](#) 和 [Device Shadow MQTT 主題](#)。

#### 步驟 4：檢閱結果及後續步驟

在本教學課程中，您會了解如何：

- 使用 `shadow.py` 範例應用程式來指定所需的狀態並更新影子的目前狀態。
- 編輯 Shadow 文件，以觀察差異事件，及 `shadow.py` 範例應用程序如何加以回應。
- 使用 MQTT 測試用戶端訂閱影子主題，並在執行範例程式時觀察更新。

#### 後續步驟

您可訂閱其他 MQTT 預留主題，以觀察影子應用程式的更新。例如，若您僅訂閱主題 `$aws/things/thingname/shadow/update/accepted`，則更新成功執行時，您只會看到目前的狀態資訊。

您也可訂閱其他影子主題來偵錯問題，或深入了解 Device Shadow 互動，還可對 Device Shadow 互動的任何問題進行偵錯。如需詳細資訊，請參閱 [the section called “影子主題”](#) 及 [Device Shadow MQTT 主題](#)。

您也可選擇使用命名影子或使用為 LED 與 Raspberry Pi 連接其他的硬體擴展您的應用程式，並使用從終端傳送訊息來觀察其狀態的變化。

如需有關 Device Shadows 服務及在裝置、應用程式和服務中使用服務的詳細資訊，請參閱 [AWS IoT Device Shadow 服務](#)、[在裝置中使用影子](#) and [在應用程式和服務中使用影子](#)。

## 教學課程：建立 AWS IoT Core 的自訂授權方

本教學課程示範使用 AWS CLI 來建立、驗證和使用自訂身分驗證的步驟。使用本教學課程，您可以選擇性地使用 Postman，藉由使用 HTTP 發佈 API 將資料傳送到 AWS IoT Core。

本教學課程會向您展示如何建立範例 Lambda 函數，搭配使用 create-authorizer 呼叫與啟用的字符簽署，以實作授權和身分驗證邏輯，以及自訂授權方。然後會使用驗證授權者 test-invoke-authorizer，最後您可以使用 HTTP 發佈 API 將 AWS IoT Core 資料傳送至測試 MQTT 主題。範例要求會指定要使用標 x-amz-customauthorizer-name 頭呼叫的授權者，並傳遞 token-key-name and x-amz-customauthorizer-signature in 要求標頭。

您會在本教學課程中學到什麼：

- 如何將 Lambda 函數建立為自訂授權方處理常式
- 如何使用啟用令牌簽名來創建自定義授權者 AWS CLI
- 如何使用 test-invoke-authorizer 命令測試您的自訂授權方
- 如何使用 [Postman](#) 發佈 MQTT 主題，以及如何利用您的自訂授權方驗證請求

此教學課程約需 60 分鐘方能完成。

在本教學課程中，您將會執行下列動作：

- [步驟 1：為自訂授權方建立 Lambda 函數](#)
- [步驟 2：為自訂授權方建立公有和私有金鑰對](#)
- [步驟 3：建立客戶授權方資源及其授權](#)
- [第 4 步：通過調用測試授權者 test-invoke-authorizer](#)
- [步驟 5：使用 Postman 測試發佈 MQTT 郵件](#)
- [步驟 6：在 MQTT 測試用戶端中檢視訊息](#)
- [步驟 7：檢閱結果及後續步驟](#)
- [步驟 8：清除](#)

開始本教學課程之前，請確定您有：

- [設定您的 AWS 帳戶](#)

您需要您 AWS 帳戶 和 AWS IoT 主控台才能完成本教學課程。

當您用於本教學課程的帳戶至少包含這些 AWS 受管政策時，此帳戶最適用：

- [IAMFullAccess](#)
- [AWSIoTFullAccess](#)
- [AWSLambda\\_FullAccess](#)

#### Important

本教學課程中使用的 IAM 政策比您應該在生產實作中遵循的 IAM 政策更寬鬆。在生產環境中，請確定您的帳戶和資源策略僅授予必要的許可。

當您建立 IAM 政策進行生產時，請判斷使用者和角色需要哪些存取權，然後設計政策，讓使用者只能執行這些任務。

如需詳細資訊，請參閱 [IAM 中的安全最佳實務](#)。

- 安裝了 AWS CLI

如需有關如何安裝的資訊 AWS CLI，請參閱 [安裝 AWS CLI](#)。本教學課程需要 AWS CLI 版本 `aws-cli/2.1.3 Python/3.7.4 Darwin/18.7.0 exe/x86_64` 或更新版本。

- OpenSSL 工具

本教學課程中的範例使用 [LibreSSL 2.6.5](#)。您也可以針對本教學課程使用 [OpenSSL v1.1.1i](#) 工具。

- 檢閱了 [AWS Lambda](#) 概觀

如果您 AWS Lambda 之前從未使用過，請檢閱 [AWS Lambda](#) 並 [開始使用 Lambda](#) 以了解其術語和概念。

- 已檢閱如何在 Postman 中建置請求

如需詳細資訊，請參閱 [建置請求](#)。

- 已從前一個教學課程中移除自訂授權方

您一次只 AWS 帳戶 能設定有限數量的自訂授權者。如需如何移除自訂授權方的相關資訊，請參閱 [the section called “步驟 8：清除”](#)。

## 步驟 1：為自訂授權方建立 Lambda 函數

中的自訂驗證 AWS IoT Core 會使用您建立的 [授權者資源](#) 來驗證和授權用戶端。您將在本節中創建的功能在客戶端連接和訪問資源時對其進行身份驗證 AWS IoT Core 和授權。AWS IoT

Lambda 函數會執行下列動作：

- 如果請求來自 test-invoke-authorizer，它會傳回一個具有 Deny 動作的 IAM 政策。
- 如果請求來自使用 HTTP 的 Passport，而且 actionToken 參數的值為 allow，它會傳回一個具有 Allow 動作的 IAM 政策。否則，它會傳回一個具有 Deny 動作的 IAM 政策。

若要為自訂授權方建立 Lambda 函數

1. 在 [Lambda](#) 主控台中，開啟 [Functions](#) (函數)。
2. 選擇 建立函式。
3. 確認已選取 Author from scratch (從頭開始撰寫)。
4. 在 Basic information (基本資訊) 下：
  - a. 在函數名稱中，輸入 **custom-auth-function**。
  - b. 在執行期中，確認 Node.js 18.x
5. 選擇 建立函式。

Lambda 會建立 Node.js 函數和許可函數上傳記錄的 [執行角色](#)。當您叫用函數並使用執行角色為 AWS SDK 建立認證以及從事件來源讀取資料時，Lambda 函數會擔任執行角色。

6. 若要在 [AWS Cloud9](#) 編輯器中查看函式的程式碼和設定，請 custom-auth-function 在設計工具視窗中選擇，然後在編輯器的導覽窗格中選擇 index.js。

對於指令碼語言 (例如 Node.js)，Lambda 包含可傳回成功回應的基本函數。您可以使用 [AWS Cloud9](#) 編輯器來編輯函數，只要原始碼不超過 3 MB。

7. 將編輯器中的 index.js 程式碼取代為下列程式碼：

```
// A simple Lambda function for an authorizer. It demonstrates
// How to parse a CLI and Http password to generate a response.

export const 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. 選擇部署。
9. 在 Changes deployed (已部署變更) 出現在編輯器上方之後：
  - a. 捲動至編輯器上方的 Function overview (函數概觀) 區段。
  - b. 複製 Function ARN (函數 ARN) 並加以儲存，以便稍後可在本教學課程中使用。
10. 測試您的函數
  - a. 選擇 Test (測試) 標籤。
  - b. 使用預設測試設定，選擇 Invoke (叫用)。

- c. 如果測試成功，請在 Execution results (執行結果) 下，開啟 Details (詳細資訊) 檢視。您應該會看到函數傳回的政策文件。

如果測試失敗或看不到政策文件，請檢閱程式碼以尋找並更正錯誤。

## 步驟 2：為自訂授權方建立公有和私有金鑰對

您的自訂授權方需要公有和私有金鑰來驗證它。本節中的命令會使用 OpenSSL 工具來建立此金鑰對。

若要為自訂授權方建立公有和私有金鑰對

1. 建立私有金鑰檔案。

```
openssl genrsa -out private-key.pem 4096
```

2. 驗證您剛建立的私有金鑰檔案。

```
openssl rsa -check -in private-key.pem -noout
```

如果命令未顯示任何錯誤，則私有金鑰檔案是有效的。

3. 建立公有金鑰檔案。

```
openssl rsa -in private-key.pem -pubout -out public-key.pem
```

4. 驗證公有金鑰檔案。

```
openssl pkey -inform PEM -pubin -in public-key.pem -noout
```

如果命令未顯示任何錯誤，則公有金鑰檔案是有效的。

## 步驟 3：建立客戶授權方資源及其授權

AWS IoT 自訂授權者是將先前步驟中建立的所有元素結合在一起的資源。在本節中，您將建立自訂授權方資源，並許可其執行您先前建立的 Lambda 函數。您可以使用 AWS IoT 控制台、或 AWS API 來建立自訂授權者資源。AWS CLI

在本教學課程中，您只需建立一個自訂授權方即可。本節說明如何使用 AWS IoT 主控台和建立 AWS CLI，以便您可以使用最方便的方法。由任一種方法建立的自訂授權方資源之間沒有任何差異。



## 建立客戶授權方資源

選擇其中一個選項，來建立自訂授權方資源

- [使用控制台創建自定義授權者 AWS IoT](#)
- [使用 AWS CLI 建立自訂授權方](#)

若要建立自訂授權方 (主控台)

1. 開啟 [AWS IoT 主控台的 \[自訂授權者\] 頁面](#)，然後選擇 [建立授權者]。
2. 在建立授權方中：
  - a. 在授權方名稱中，輸入 **my-new-authorizer**。
  - b. 在授權方狀態中，勾選作用中。
  - c. 在 Authorizer function (授權方函數) 中，選擇您先前建立的 Lambda 函數。
  - d. 在 Token validation - optional (字符驗證 - 選用) 中：
    - i. 切換字符驗證。
    - ii. 在符記金鑰名稱中，輸入 **tokenKeyName**。
    - iii. 選擇 Add key (新增金鑰)。
    - iv. 在金鑰名稱中，輸入 **FirstKey**。
    - v. 在公有金鑰中，輸入 `public-key.pem` 檔案的內容。務必包括檔案中具有 -----BEGIN PUBLIC KEY----- 和 -----END PUBLIC KEY----- 的字行，並且不得從檔案內容中新增或刪除任何換行字元、歸位字元或其他字元。您輸入的字串應該看起來與這個範例相似。

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAvEBz0k4vhN+3Lgs1vEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQ0e5tyzz1MsKdmtFGxMqQ3qjEXAMPLE0mqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU50SANpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCE1f0csa1S/Rk4phD5
oa4Y0GHISRnevypg5C8n9Rrz91PWGqP6M/q5DNJJXjMyleG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGdfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sq8Lgzpey9N50N1o1Cvldwvc
KrJJtgwW6hVqRGuShnownLpgG86M6neZ5sRMbVNZ080zcobLngJ0Ibw9KkcUdk1W
gvZ6HEJqBY2XE70iEXAMPLETPHzhqV6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN717Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJ1Uzn62Q+VeNV2tdA7MfPfMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kF12y0BmGAP0RBivRd9
```

```
JWBUCG0bqcLQPeQyjbXS0fUCAwEAAQ==
-----END PUBLIC KEY-----
```

3. 選擇 Create Authorizer (建立授權方)。
4. 如果已建立自訂授權方資源，您會看到自訂授權方的清單，而且您的新自訂授權方應該會出現在清單中，接著您可以繼續進行下一節來測試它。

如果您看到錯誤，請檢閱錯誤並嘗試重新建立您的自訂授權方，然後再次檢查這些項目。請注意，每個自訂授權方資源都必須具有唯一名稱。

## 建立自訂授權方 (AWS CLI)

1. 將您的值替代為 `authorizer-function-arn` 和 `token-signing-public-keys`，然後執行下列命令：

```
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-----
MIICIjANBgkqhkiG9w0BAQEFAAACg8AMIICCgKCAgEAvEBz0k4vhN+3LgslvEWt
sLCqNmt5Damas3bmiTRvq2gjRJ6KXGTGQChqArAJwL1a9dkS9+maaXC3vc6xzx9z
QPu/vQ0e5tyzz1MsKdmtFGxMqQ3qjEXAMPLE0mqyUKPP5mff58k6ePSfXAnzBH0q
lg2HioefrpU50SANpuRAjYKofKjbc2Vrn6N2G7hV+IfTBvCElf0csa1S/Rk4phD5
oa4Y0GHISRnevyppg5C8n9Rrz91PWGqP6M/q5DNJJXjMyLeG92hQgu1N696bn5Dw8
FhedszFa6b2x6xrItZFzewNQkPMLMFhNrQIIyvshtT/F1LVCS5+v8AQ8UGGDfZmv
QeqAMAF7WgagDMXcfgKSVU8yid2sIm56qsCLMvD2Sg8Lgzpey9N50N1o1Cvldwvc
KrJjtgwW6hVqRGuShnownLpgG86M6neZ5sRmbVNZ080zcobLNgJ0Ibw9KkcUdk1W
gvZ6HEJqBY2XE70iEXAMPLETPHzhqvK6Ei1HGxpHsXx6BNft582J1VpgYjXha8oa
/NN7L7Zbj/euAb41IVtmX8JrD9z613d1iM5L8HluJlUzn62Q+VeNV2tdA7MfPFMC
8btGYladFAnitThaz6+F0VSBJPu7pZQoLnqyEp5zLMtF+kFL2y0BmGAP0RBivRd9
JWBUCG0bqcLQPeQyjbXS0fUCAwEAAQ==
-----END PUBLIC KEY-----"
```

其中：

- `authorizer-function-arn` 值是您為自訂授權方建立之 Lambda 函數的 Amazon 資源名稱 (ARN)。

- `token-signing-public-keys` 值包含金鑰的名稱 **FirstKey** 以及 `public-key.pem` 檔案的內容。務必包括檔案中具有 `-----BEGIN PUBLIC KEY-----` 和 `-----END PUBLIC KEY-----` 的字行，並且不得從檔案內容中新增或刪除任何換行字元、歸位字元或其他字元。

注意：輸入公有金鑰時要小心，因為公有金鑰值若有任何更改，都會使其無法使用。

2. 如果已建立自訂授權方，則命令會傳回新資源的名稱和 ARN，如下所示。

```
{
  "authorizerName": "my-new-authorizer",
  "authorizerArn": "arn:aws:iot:Region:57EXAMPLE833:authorizer/my-new-authorizer"
}
```

儲存 `authorizerArn` 值，以使用於下一個步驟。

記住，每個自訂授權方資源都必須具有唯一名稱。

## 授權自訂授權方資源

在本節中，您將許可您剛建立的自訂授權方資源執行 Lambda 函數。若要授與許可，您可以使用 [add-permission](#) CLI 命令。

使用以下方法授予 Lambda 函數的權限 AWS CLI

1. 在插入您的值之後，輸入以下命令。請注意，`statement-id` 值必須是唯一的。如果您之前已執行本教學課程，或者如果您得到 `ResourceConflictException` 錯誤，請將 `Id-1234` 取代為另一個值，。

```
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，它會傳回許可陳述式，例如此範例。您可以繼續下一節來測試自訂授權方。

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

如果命令未成功，它會傳回錯誤，例如此範例。您必須先檢閱並更正錯誤，然後才能繼續進行。

```
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
```

## 第 4 步：通過調用測試授權者 test-invoke-authorizer

在本節中定義了所有資源後，您將 test-invoke-authorizer 從命令行調用以測試授權通過。

請注意，從命令列調用授權方時，未定義 protocolData，因此授權方將一律傳回 DENY 文件。不過，此測試會確認您的自訂授權方和 Lambda 函數已正確設定，即使它未完全測試 Lambda 函數也一樣。

若要測試您的自訂授權者及其 Lambda 函數，請使用 AWS CLI

1. 在具有您在前一個步驟中建立之 private-key.pem 檔案的目錄中，執行下列命令。

```
echo -n "tokenKeyValue" | openssl dgst -sha256 -sign private-key.pem | openssl
base64 -A
```

此命令會建立要在下一個步驟中使用的簽章字串。簽章字串看起來像這樣：

```
dBwykz1b+fo+JmSGdwoGr8dyC2qB/IyLefJJr+rbCvmu9Jl4KHAA9DG+V
+MMWu09YSA86+64Y3Gt4t0ykpZqn9mn
VB1wyxp+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQNqBZzbCvsluv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaaeh
bQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm900yZ94R4bdJsHNig1JePgnu0BvMGCEFE09jGjj
szEHfgAUAQIWXiVGQj16BU1xKpTGSiTawheLKUjITOEXAMPLEECK3aHKYKY
+d1vTvdthKtYHBq8MjhzJ0kggbt29V
QJCb8Ri1N/P5+vcVniSXWPplyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtcspACi6ca
+tsDuX
f3LzCwQQF/YSUy02u5Xkwn
+sto6KCKpNlkD0wU8gl3+k0zxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o+K
```

```
EWckTe91I1mokDr5sJ4JXixvnJTVSx11li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+f1FeLoZLAWQFH
xRlXsPqiVKS1ZIUClaZWprh/orDjPlpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

複製此簽章字串以在下一個步驟中使用。請小心不要包含任何額外的字元或留下任何額外的字元。

- 在此命令中，會將 `token-signature` 值取代為來自前一個步驟的簽章字串，並執行此命令來測試您的授權方。

```
aws iot test-invoke-authorizer \
--authorizer-name my-new-authorizer \
--token tokenKeyValue \
--token-signature dBwykzLb+fo+JmSGdwoGr8dyC2qB/IyLefJJr
+rbCvmu9Jl4KHAA9DG+V+MMWu09YSA86+64Y3Gt4t0ykpZqn9mnVB1wyxp
+0bDZh8hmqUAUH3fwi3fPjBvCa4cwNuLQnqBZzbCvsLuv7i2IMjEg
+CPY0zrWt1jr9BikgGPDxWkjaeehbQHHTo357TegKs9pP30Uf4TrxypNmFswA5k7QIc01n4bIyRTm900yZ94R4bdJsh
+d1vTvdthKtYHBq8MjhzJ0kggbt29VQJCb8RilN/
P5+vcVniSXWpPlyB5jkYs9UvG08REoy64AtizfUhvSul/r/F3VV8ITtQp3aXiUtccspACi6ca
+tsDuXf3LzCwQQF/YSUy02u5XkWn
+sto6KCKpNlKD0wU8gl3+k0zxrthnQ8gEajd5Iylx230iqcXo3osjPha7JDyWM5o
+KEWckTe91I1mokDr5sJ4JXixvnJTVSx11li49IalW4en1DAkc1a0s2U2UNm236EXAMPLELotyh7h
+f1FeLoZLAWQFHxRlXsPqiVKS1ZIUClaZWprh/orDjPlpiWfBgBIOgokJIDGP9gwhXIIk7zWrGmWpMK9o=
```

如果命令成功，它會傳回由您的客戶授權方函數所產生的資訊，例如此範例。

```
{
  "isAuthenticated": true,
  "principalId": "principalId",
  "policyDocuments": [
    [{"Version": "2012-10-17", "Statement": [{"Action": "iot:*", "Effect": "Deny", "Resource": "arn:aws:iot:Region:57EXAMPLE833:*"}]}],
  "refreshAfterInSeconds": 600,
  "disconnectAfterInSeconds": 3600
}
```

如果指令傳回錯誤，請檢閱錯誤並再次檢查您在本節中使用的命令。

## 步驟 5：使用 Postman 測試發佈 MQTT 郵件

- 若要從命令列取得您的裝置資料端點，請呼叫 [describe-endpoint](#)，如這裡所示

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

儲存此地址，以在稍後步驟中用作 *device\_data\_endpoint\_address*。

2. 開啟新的 Postman 視窗並建立新的 HTTP POST 請求。
  - a. 從您的電腦中，開啟 Postman 應用程式。
  - b. 在 Postman 的 File (檔案) 選單中，選擇 New (新增)。
  - c. 在 New (新增) 對話方塊中，選擇 Request (請求)。
  - d. 在儲存請求中，
    - i. 在 Request name (請求名稱) 中，輸入 **Custom authorizer test request**。
    - ii. 在 Select a collection or folder to save to: (選取要儲存到哪個集合或資料夾：) 中，選擇或建立要儲存此請求的集合。
    - iii. 選擇 Save to *collection\_name* (儲存至 collection\_name)。
3. 建立 POST 請求來測試您的自訂授權方。
  - a. 在 URL 欄位旁邊的請求方法選取器中，選擇 POST。
  - b. 在 URL 欄位中，為您的請求建立 URL，方法為搭配使用下列 URL 與來自前一個步驟中 [describe-endpoint](#) 命令的 *device\_data\_endpoint\_address*。

```
https://device_data_endpoint_address:443/topics/test/cust-auth/topic?  
qos=0&actionToken=allow
```

請注意，此 URL 包含 `actionToken=allow` 查詢參數，它會告訴您的 Lambda 函數傳回允許存取 AWS IoT 的政策文件。輸入 URL 之後，查詢參數也會出現在 Postman 的 Params (參數) 標籤中。

- c. 在 Auth (身分驗證) 標籤的 Type (類型) 欄位中，選擇 No Auth (無需身分驗證)。
- d. 在標頭標籤中：
  - i. 如果有已核取的 Host (主機) 金鑰，請取消核取此金鑰。
  - ii. 在標頭清單的底部，加入這些新標題並確認已核取它們。將 **Host** 值取代為您的 *device\_data\_endpoint\_address*，並將 **x-amz-customauthorizer-signature** 值取代為在上節中與 `test-invoke-authorize` 命令搭配使用的簽章字串。

金鑰	值
<b>x-amz-customauthorizer-name</b>	<b>my-new-authorizer</b>
<b>Host</b>	<i>device_data_endpoint_addresses</i>
<b>tokenKeyName</b>	<b>tokenKeyValue</b>
<b>x-amz-customauthorizer-signature</b>	<i>##### + ## + JMS 8 ### QB/ ## JR+RBCVM4 # 9 # + V + ## 9 ## 9nVB1WQB +0B1WZB +0BZH8H3fwi GdwoGr 3 CWQB 9bikGGp Qhto357 9PP30UF4 A5K7QIC01N4 ### 90 # # 4bjshnig1 ## F09JQQ0KJQ # ### 1 KGGBT29 ## /P5 + ##### # 5 ## 9UVG8 /R/F3VV8i 3 # 6 # + 3 QQF/IyLef# 02U5 + ## 6# tOykpZqn fPjBv ZzbCvsluv MjEg DxWkjaeehb TegKs TrxypNmFs w JePgnu vTvdthKt AtizfUhvS ul TtQp aXiUtcsp tsDuXf LzCw XkWnCkpNlkD0wu8gl3 + ##### Q8 ### 230IQ03 ## 7 ## 5 0 + ## 91I1MOKDR5SJ4JK 4 ### V1L49IA WckTe 4A0S2U 克拉茲普拉/德生物 9gWhxik7WPM9O= fIFeloZI HxRIXsPqi JlpIWfBg zWrGm</i>

- e. 在主體標籤中：
- 在資料格式選項方塊中，選擇 Raw (原始)。
  - 在資料類型清單中，選擇 JavaScript。
  - 在文字欄位中，為您的測試訊息輸入此 JSON 訊息承載：

```
{
  "data_mode": "test",
```

```
"vibration": 200,  
"temperature": 40  
}
```

#### 4. 選擇 Send (傳送) 以傳送請求。

如果請求成功，它會傳回：

```
{  
  "message": "OK",  
  "traceId": "ff35c33f-409a-ea90-b06f-fbEXAMPLE25c"  
}
```

成功回應表示您的自訂授權者允許連線，AWS IoT 並且測試訊息已傳送給中 AWS IoT Core 的 Broker。

如果傳回錯誤，請檢閱錯誤訊息、*device\_data\_endpoint\_address*、簽章字串，以及其他標頭值。

請將此請求保留在 Postman，以供下一節使用。

### 步驟 6：在 MQTT 測試用戶端中檢視訊息

在上一個步驟中，您使用 Postman 將模擬的裝置訊息傳 AWS IoT 送至。成功回應指出您的自訂授權方允許連線到 AWS IoT，並指出測試訊息已傳遞至 AWS IoT Core 中的代理程式。在本節中，您將使用 AWS IoT 主控台中的 MQTT 測試用戶端來查看來自該訊息的訊息內容，就像其他裝置和服務一樣。

若要查看自訂授權方所授權的測試訊息

1. 在 AWS IoT 主控台中，開啟 [MQTT 測試](#) 用戶端。
2. 在 Subscribe to topic (訂閱主題) 標籤的 Topic filter (主題篩選條件) 中，輸入 **test/cust-auth/topic**，這是前一節的 Postman 範例中使用的訊息主題。
3. 選擇 Subscribe (訂閱)。

保留此視窗讓後續步驟可以看到它。

4. 在 Postman 中，於您為前一節建立的請求中，選擇 Send (傳送)。

檢閱回應以確定回應成功。如果未成功，請依照前一節所述疑難排解錯誤。



5. 在 MQTT test client (MQTT 測試用戶端) 中，您應該會看到顯示訊息主題的新項目，而且如果展開的話，則會看到您從 Postman 傳送之請求中的訊息承載。

如果未在 MQTT test client (MQTT 測試用戶端) 中看到您的訊息，以下是一些需要檢查的事項：

- 確定您的 Postman 請求成功傳回。如果 AWS IoT 拒絕連線並傳回錯誤，則要求中的訊息不會傳遞至訊息代理程式。
- 請確定 AWS 帳戶 和 AWS 區域 用來開啟主 AWS IoT 控制台與您在 Postman URL 中使用的相同。
- 確定您已在 MQTT test client (MQTT 測試用戶端) 中正確地輸入主題。主題篩選條件會區分大小寫。如有疑問，您也可以訂閱#主題，該主題會訂閱所有 MQTT 訊息，這些訊息會透過訊息代理程式傳遞 AWS 帳戶 並 AWS 區域 用來開啟主控台。AWS IoT

## 步驟 7：檢閱結果及後續步驟

在本教學課程中：

- 已將 Lambda 函數建立為自訂授權方處理常式
- 已在啟用字符簽署的情況下建立自訂授權方
- 已使用 test-invoke-authorizer 命令測試您的自訂授權方
- 已使用 [Postman](#) 發佈 MQTT 主題，並利用您的自訂授權方驗證請求
- 已使用 MQTT test client (MQTT 測試用戶端) 來檢視從 Postman 測試傳送的訊息

### 後續步驟

從 Postman 傳送一些訊息以驗證自訂授權方是否正在運作之後，請試驗看看變更本教學課程的不同層面如何影響結果。以下是幾個入門範例。

- 變更簽章字串，以便查看未經授權之連線嘗試的處理方式不再有效。您應該會得到錯誤回應 (例如這個錯誤回應)，且訊息應該不會出現在 MQTT test client (MQTT 測試用戶端) 中。

```
{
  "message": "Forbidden",
  "traceId": "15969756-a4a4-917c-b47a-5433e25b1356"
}
```

- 若要深入瞭解如何尋找在開發和使用 AWS IoT 規則時可能發生的錯誤，請參閱[監控 AWS IoT](#)。

## 步驟 8：清除

如果想要重複本教學課程，您可能需要移除某些自訂授權方。您一次只 AWS 帳戶 能配置有限數量的自定義授權者，`LimitExceededException`當您嘗試添加新授權者而不刪除現有的自定義授權者時，您可以獲得一個。

### 移除自訂授權方 (主控台)

1. 打開[AWS IoT 控制台的「自定義授權者」頁面](#)，然後在自定義授權者列表中找到要刪除的自定義授權者。
2. 開啟自訂授權方詳細資訊頁面，然後從 Actions (動作) 選單中，選擇 Edit (編輯)。
3. 取消核取 Activate authorizer (啟用授權方)，然後選擇 Update (更新)。

在自訂授權方作用中時，您無法將其刪除。

4. 從自訂授權方詳細資訊頁面中，開啟 Actions (動作) 選單，然後選擇 Delete (刪除)。

### 移除自訂授權方 (AWS CLI)

1. 列出您已安裝的自訂授權方，並找出要刪除的自訂授權方名稱。

```
aws iot list-authorizers
```

2. 在將 `Custom_Auth_Name` 取代為要刪除之自訂授權方的 `authorizerName` 之後，執行此命令將自訂授權方設定為 `inactive`。

```
aws iot update-authorizer --status INACTIVE --authorizer-name Custom_Auth_Name
```

3. 在將 `Custom_Auth_Name` 取代為要刪除之自訂授權方的 `authorizerName` 之後，執行此命令來刪除自訂授權方。

```
aws iot delete-authorizer --authorizer-name Custom_Auth_Name
```

## 教程:監測土壤水分 AWS IoT 與樹莓派

本教程向您展示如何使用 [Raspberry Pi](#) ( 濕度傳感器 )，以及如 AWS IoT 何監測室內植物或花園的土壤水分含量。Raspberry Pi 運行代碼，該代碼從傳感器讀取水分水平和溫度，然後將數據發送到 AWS IoT。您可以在中建立規則 AWS IoT，當濕度等級低於閾值時，會將電子郵件傳送至訂閱 Amazon SNS 主題的地址。

### Note

本教學課程可能不是最新版本。自本主題最初發佈後，某些參考文件可能已被取代。

## 內容

- [必要條件](#)
- [設定 AWS IoT](#)
  - [步驟 1：建立原 AWS IoT 則](#)
  - [步驟 2：建立物 AWS IoT 件、憑證和私密金鑰](#)
  - [步驟 3：建立 Amazon SNS 主題與訂閱。](#)
  - [步驟 4：建立 AWS IoT 規則以傳送電子郵件](#)
- [設定您的 Raspberry Pi 和濕度感應器](#)

## 必要條件

為完成此教學課程您需要：

- 一個 AWS 帳戶。
- 擁有管理員許可的 IAM 使用者。
- 執行 Windows、macOS、Linux 或 Unix 的開發電腦，用來存取 [AWS IoT 主控台](#)。
- 執行最新 [Raspbian 作業系統](#)的 [Raspberry Pi 3B 或 4B](#)。如需安裝指示，請參閱 Raspberry Pi 網站上的 [Installing operating system images \(安裝作業系統映像\)](#)。
- 用於 Raspberry Pi 的顯示器、鍵盤、滑鼠和 Wi-Fi 網路或乙太網路連線。
- 與 Raspberry Pi 相容的濕度感應器。本教學課程中使用的感應器是 [Adafruit STEMMA I2C 電容式濕度感應器](#)搭配 [JST 4 針腳轉母頭插槽纜線接頭](#)。

## 設定 AWS IoT

若要完成此教學課程，您需要建立以下資源。若要將裝置連線到 AWS IoT，您可以建立 IoT 物件、裝置憑證和 AWS IoT 原則。

- 一 AWS IoT 件事

代表實體裝置 (在此案例中為 Raspberry Pi)，包含有關裝置的靜態中繼資料的物件。

- 裝置憑證。

所有裝置都必須擁有裝置憑證，才能連接至並向 AWS IoT 驗證。

- 一個 AWS IoT 政策。

每個裝置憑證都有一或多個與其相關聯的 AWS IoT 規則。這些原則會決定裝置可存取的 AWS IoT 資源。

- AWS IoT 根 CA 憑證。

裝置和其他用戶端會使用 AWS IoT 根 CA 憑證來驗證與其通訊的 AWS IoT 伺服器。如需詳細資訊，請參閱 [伺服器驗證](#)。

- 一個 AWS IoT 規則

規則包含查詢和一或多個規則動作。查詢會從裝置訊息擷取資料，以判斷是否應處理訊息資料。規則動作會指定當資料符合查詢時要採取的動作。

- Amazon SNS 主題和主題訂閱。

此規則會監聽來自 Raspberry Pi 的濕度資料。如果值低於閾值，它會將訊息傳送到 Amazon SNS 主題。Amazon SNS 會將該訊息傳送到訂閱該主題的所有電子郵件地址。

## 步驟 1：建立原 AWS IoT 則

創建一個 AWS IoT 策略，允許您的 Raspberry Pi 連接並將消息發送到 AWS IoT。

1. 在 [AWS IoT 主控台](#) 中，如果 Get started (開始) 按鈕出現，請選擇它。否則，請在導覽窗格中展開 Security (安全性)，然後選擇 Policies (政策)。
2. 如果 You don't have any policies yet (您尚未有任何政策) 對話方塊出現，請選擇 Create a policy (建立政策)。否則，請選擇 Create (建立)。
3. 輸入 AWS IoT 策略的名稱 (例如，**MoistureSensorPolicy**)。
4. 在 Add statements (新增陳述式) 區段中，將現有政策取代為下列 JSON。使用您的和號碼取代## AWS 區域 和 # AWS ## #。

```
{
  "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. 選擇建立。

### 步驟 2：建立物 AWS IoT 件、憑證和私密金鑰

在 AWS IoT 註冊表中創建一個東西來表示您的樹莓派。

1. 在 [AWS IoT 主控台](#) 的導覽窗格中，依序選擇 Manage (管理) 和 Things (物件)。
2. 如果顯示 You don't have any things yet (尚無任何物件) 對話方塊，請選擇 Register a thing (註冊物件)。否則，請選擇 Create (建立)。
3. 在 [建立 AWS IoT 物件] 頁面上，選擇 [建立單一物件]。
4. 在 Add your device to the device registry (將裝置新增至裝置登錄檔) 頁面上，輸入您 IoT 物件的名稱 (例如 **RaspberryPi**)，然後選擇 Next (下一步)。您無法在建立之後變更物件的名稱。要變更物件的名稱，您必須建立一個新的物件並為它命名，然後刪除舊的物件。
5. 在 Add a certificate for your thing (新增物件的憑證) 頁面上，選擇 Create certificate (建立憑證)。
6. 選擇 Download (下載) 連結來下載憑證、私有金鑰和根憑證授權機構憑證。

#### Important

這是您可以下載憑證和私有金鑰唯一機會。

7. 若要啟用憑證，請選擇 Activate (啟用)。憑證必須作用中，裝置才能連接到 AWS IoT。
8. 選擇 Attach a policy (連接政策)。
9. 針對 [新增物件的原則]，選擇 MoistureSensorPolicy，然後選擇 [註冊物件]。

### 步驟 3：建立 Amazon SNS 主題與訂閱。

建立 Amazon SNS 主題與訂閱。

1. 在 [AWS SNS 主控台](#) 的導覽窗格中，選擇 Topics (主題)，然後選擇 Create topic (建立主題)。
2. 選擇「類型」作為「標準」，然後輸入主題的名稱 (例如，**MoistureSensorTopic**)。
3. 輸入主題的顯示名稱 (例如，**Moisture Sensor Topic**)。這是在 Amazon SNS 主控台中針對您的主題顯示的名稱。
4. 請選擇 建立主題。
5. 在 Amazon SNS 主題詳細資訊頁面中，選擇 Create subscription (建立訂閱)。
6. 對於通訊協定，選擇電子郵件。
7. 針對 Endpoint (端點)，輸入電子郵件地址。
8. 選擇建立訂閱。
9. 開啟您的電子郵件用戶端，並尋找主旨為 **MoistureSensorTopic** 的訊息。開啟電子郵件，然後按一下 Confirm subscription (確認訂閱) 連結。

#### Important

在您確認訂閱之前，不會收到來自此 Amazon SNS 主題的任何電子郵件提醒。

您應該會收到含有您輸入文字的電子郵件訊息。

### 步驟 4：建立 AWS IoT 規則以傳送電子郵件

AWS IoT 規則會定義查詢，以及從裝置接收訊息時要採取的一或多個動作。AWS IoT 規則引擎偵聽設備發送的消息，並使用消息中的數據來確定是否應採取某些操作。如需詳細資訊，請參閱 [規則 AWS IoT](#)。

在此教學中，您的 Raspberry Pi 會將訊息發佈至 `aws/things/RaspberryPi/shadow/update`。這是內部的 MQTT 主題，供裝置和物件影子服務使用。Raspberry Pi 發佈的訊息會具備以下格式：

```
{
  "reported": {
    "moisture" : moisture-reading,
    "temp" : temperature-reading
  }
}
```

您會建立一個查詢，從傳入的訊息擷取濕度和溫度資料。您也可以建立 Amazon SNS 動作，在濕度讀數低於閾值時，取得資料並將其傳送給 Amazon SNS 主題訂閱者。

## 建立 Amazon SNS 規則

1. 在[AWS IoT 主控台](#)中，選擇 [郵件路由]，然後選擇 [規則]。如果 You don't have any rules yet (您尚未有任何規則) 對話方塊出現，請選擇 Create a rule (建立規則)。否則，請選擇「建立規則」。
2. 在 [規則屬性] 頁面中，輸入規則名稱，例如 **MoistureSensorRule**，並提供簡短的規則描述，例如 **Sends an alert when soil moisture level readings are too low**。
3. 選擇下一步並設定您的 SQL 敘述句。選擇 SQL 版本作為，然後輸入以下 AWS IoT SQL 查詢語句：

```
SELECT * FROM '$aws/things/RaspberryPi/shadow/update/accepted' WHERE
state.reported.moisture < 400
```

此陳述式會在 moisture 讀數小於 400 時觸發規則動作。

### Note

您可能需要使用不同的值。在 Raspberry Pi 上執行程式碼之後，您可以碰觸感應器、將其放入水中或將其放入花盆中，以查看從感應器取得的值。

4. 選擇「下一步」並附加規則動作。針對「作業 1」，選擇「簡易通知服務」。此規則動作的說明為以 SNS 推播通知傳送訊息。
5. 對於 SNS 主題，請選擇您在中建立的主題 [步驟 3：建立 Amazon SNS 主題與訂閱](#)。MoistureSensorTopic，並將訊息格式保留為 RAW。對於 IAM 角色，選擇建立新角色。輸入角色的名稱，例如 **LowMoistureTopicRole**，然後選擇 [建立角色]。
6. 選擇 [下一步] 以檢閱，然後選擇 [建立] 以建立規則。

## 設定您的 Raspberry Pi 和濕度感應器

將 microSD 卡插入 Raspberry Pi，連接顯示器、鍵盤、滑鼠，以及乙太網路纜線 (如果未使用 Wi-Fi)。還不要連接電源線。

將 JST 跳線纜線連接至濕度感應器。跳線的另一端有四條配線：



- 綠色：I2C SCL
- 白色：I2C SDA
- 紅色：功率 (3.5 V)
- 黑色：接地

按住 Raspberry Pi 與右側的乙太網路插孔。在此方向中，上方有兩列 GPIO 接腳。按照以下順序，將濕度感應器的配線連接到接腳的下方排。從最左側接腳開始，連接紅色 (電源)、白色 (SDA) 和綠色 (SCL)。略過一個接腳，然後連接黑色 (接地) 配線。如需詳細資訊，請參閱 [Python Computer Wiring](#)。

將電源線連接至 Raspberry Pi，並將另一端插入牆上插座，以將其開啟。

設定您的 Raspberry Pi

1. 在 Welcome to Raspberry Pi (歡迎使用 Raspberry Pi) 上，選擇 Next (下一步)。
2. 選擇您的國家/地區、語言、時區及鍵盤配置。選擇 Next (下一步)。
3. 輸入您的 Raspberry Pi 的密碼，然後選擇 Next (下一步)。
4. 選擇您的 Wi-Fi 網路，然後選擇 Next (下一步)。如果您不是使用 Wi-Fi 網路，請選擇 Skip (略過)。
5. 選擇 Next (下一步) 以檢查軟體更新。更新完成時，選擇 Restart (重新啟動) 以重新啟動您的 Raspberry Pi。

在 Raspberry Pi 啟動後，啟用 I2C 介面。

1. 在 Raspbian 桌面的左上角，按一下 Raspberry 圖示，選擇 Preferences (偏好設定)，然後選擇 Raspberry Pi Configuration (Raspberry Pi 組態)。
2. 在 Interfaces (介面) 標籤上，針對 I2C，選擇 Enable (啟用)。
3. 選擇 確定。

Adafruit STEMMA 濕度傳感器的庫是為了編寫的。CircuitPython 若要在 Raspberry Pi 上執行它們，您需要安裝最新版本的 Python 3。

1. 從命令提示執行下列命令，以更新您的 Raspberry Pi 軟體：

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

2. 執行以下命令來更新您的 Python 3 安裝：

```
sudo pip3 install --upgrade setuptools
```

3. 執行以下命令來安裝 Raspberry Pi GPIO 程式庫：

```
pip3 install RPI.GPIO
```

4. 執行以下命令來安裝 Adafruit Blinka 程式庫：

```
pip3 install adafruit-blinka
```

如需詳細資訊，請參閱在 [Raspberry Pi 上安裝 CircuitPython 程式庫](#)。

5. 執行以下命令來安裝 Adafruit Seesaw 程式庫：

```
sudo pip3 install adafruit-circuitpython-seesaw
```

6. 執行下列命令以安裝適用於 Python 的裝 AWS IoT 置 SDK：

```
pip3 install AWSIoTPythonSDK
```

您的 Raspberry Pi 現在擁有所有必要的程式庫。建立名為 `moistureSensor.py` 的檔案，並將下列 Python 程式碼複製到檔案：

```
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 AWS 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)
```

將檔案儲存到您可以找到的位置。從命令列搭配下列參數執行 `moistureSensor.py`：

### 端點

您的自訂 AWS IoT 端點。如需詳細資訊，請參閱 [Device Shadow REST API](#)。

### rootCA

AWS IoT 根 CA 憑證的完整路徑。

### cert

AWS IoT 裝置憑證的完整路徑。

### 金鑰

AWS IoT 裝置憑證私密金鑰的完整路徑。

### thingName

您的物件名稱 (在此案例中為 `RaspberryPi`)。

### clientId

MQTT 用戶端 ID。請使用 `RaspberryPi`。

命令列看起來應該如下：

```
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
```

嘗試碰觸感應器、將其放入花盆中，或將其放入杯中，以查看感應器如何回應各種濕度。如有需要，您可以在 `MoistureSensorRule` 中變更閾值。當濕度感測器讀數低於規則的 SQL 查詢陳述式中指定的值時，會向 Amazon SNS 主題 AWS IoT 發佈訊息。您應該會收到包含濕度和溫度資料的電子郵件訊息。

在您驗證收到來自 Amazon SNS 的電子郵件訊息後，請按 CTRL+C 來停止 Python 程式。Python 程式傳送的訊息量不足以產生費用，但最佳實務是在完成時停止程式。

## 管理裝置 AWS IoT

AWS IoT 提供可協助您管理事物的登錄。物件是特定裝置或邏輯實體的代表。它可以是實體裝置或感應器 (例如燈泡或牆上的開關)。它也可以是邏輯實體，例如應用程式或實體實體的執行個體，這些實體不會連線，AWS IoT 但與其他裝置有關 (例如，具有引擎感應器或控制面板的汽車)。

物件的相關資訊會以 JSON 資料的形式存放於登錄檔中。以下為物件的範例：

```
{
  "version": 3,
  "thingName": "MyLightBulb",
  "defaultClientId": "MyLightBulb",
  "thingTypeName": "LightBulb",
  "attributes": {
    "model": "123",
    "wattage": "75"
  }
}
```

物件以名稱識別。物件可以具有屬性，亦即名稱/值對，可讓您用來存放關於物件的資訊，例如其序號或製造商。

典型的裝置使用案例包括使用物件名稱做為預設 MQTT 用戶端 ID。雖然我們不會強制對物件登錄檔名稱及其使用的 MQTT 用戶端 ID、憑證或影子狀態進行映射，但仍建議您選擇物件名稱，並用其作為登錄檔與 Device Shadow 服務的 MQTT 用戶端 ID。這能為您的 IoT 裝置機群帶來秩序與便利，同時仍能保有基礎裝置憑證模型或影子的彈性。

您不必在登錄檔中建立物件，即可將裝置連線至 AWS IoT。將物件新增至登錄檔，可更輕鬆地管理及搜尋裝置。

## 如何使用登錄檔管理物件

您可以使用 AWS IoT 主控台、AWS IoT API 或與登錄互動。AWS CLI 下列章節說明如何利用 CLI 使用登錄檔。

命名您的物件物件時：

- 不要在您的物品名稱中使用可識別個人身份的信息。物件名稱可以出現在未加密的通訊和報告中。

## 建立物件

下列命令顯示如何使用 CLI 中的 AWS IoT CreateThing 指令來建立物件。建立物件之後，就無法變更物件的名稱。若要變更物件的名稱，請建立新物件，指定新名稱，然後刪除舊物件。

```
$ aws iot create-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\": {\"wattage\": \"75\", \"model\": \"123\"}}"
```

該 CreateThing 命令會顯示新物件的名稱和 Amazon Resource Name (ARN)：

```
{
  "thingArn": "arn:aws:iot:us-east-1:123456789012:thing/MyLightBulb",
  "thingName": "MyLightBulb",
  "thingId": "12345678abcdefgh12345678ijklmnop12345678"
}
```

### Note

我們不建議在物件名稱中使用個人識別資訊。

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [create-thing](#)。

## 列出物件

您可以使用 ListThings 命令列出您帳戶中的所有物件：

```
$ 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` 命令搜尋特定物件類型的所有物件：

```
$ 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` 命令搜尋具特定值之屬性的所有物件。此命令最多可搜尋三個屬性。

```
$ 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"
    }
  ]
}
```

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [list-things](#)。

## 說明物件

您可使用 DescribeThing 命令顯示物件的相關詳細資訊：

```
$ 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"
}
}
```

若要取得更多資訊，請參閱《[指令參考](#)》中的 [AWS CLI 描述物件](#)。

## 更新物件

您可以使用 UpdateThing 命令更新物件。此指令只會更新物件的屬性。您不能改變物件的名字。若要變更物件的名稱，請建立新物件，指定新名稱，然後刪除舊物件。

```
$ aws iot update-thing --thing-name "MyLightBulb" --attribute-payload "{\"attributes\": {\"wattage\": \"150\", \"model\": \"456\"}}"
```

UpdateThing 命令不會產生輸出。您可以使用 DescribeThing 命令來查看結果：

```
$ aws iot describe-thing --thing-name "MyLightBulb"
{
  "attributes": {
    "model": "456",
    "wattage": "150"
  },
  "version": 2,
  "thingName": "MyLightBulb"
}
```

如需詳細資訊，請參閱《[AWS CLI 命令參考](#)》中的 [update-thing](#)。

## 刪除物件

您可以使用 DeleteThing 命令來刪除物件：

```
$ aws iot delete-thing --thing-name "MyThing"
```

如果刪除成功或您指定的物件不存在，則此命令會成功傳回且不含錯誤。

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [delete-thing](#)。

## 將主體連接至物件

實體裝置必須具 AWS IoT 有要與之通訊的 X.509 憑證。您可以將裝置上的憑證與登錄檔中代表您裝置的物件相關聯。若要將憑證連接至物件，請使用 `AttachThingPrincipal` 命令：

```
$ aws iot attach-thing-principal --thing-name "MyLightBulb" --principal  
"arn:aws:iot:us-east-1:123456789012:cert/  
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

`AttachThingPrincipal` 命令不會產生任何輸出。

若要取得更多資訊，請參閱 [《指令參考》中的貼附主體](#)。AWS CLI

## 將主體與物件分離

您可以使用 `DetachThingPrincipal` 命令從物件分離出憑證：

```
$ aws iot detach-thing-principal --thing-name "MyLightBulb" --principal  
"arn:aws:iot:us-east-1:123456789012:cert/  
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

`DetachThingPrincipal` 命令不會產生任何輸出。

如需詳細資訊，請參閱從 [指令參考中卸離主體](#)。AWS CLI

## 物件類型

物件類型能讓您存放與相同物件類型關聯之所有物件通用的描述和組態資訊。如此可簡化登錄檔中物件的管理。例如，您可以定義物 `LightBulb` 件類型。與物 `LightBulb` 件類型相關聯的所有物件都共用一組屬性：序號、製造商和瓦數。當您建立物件類型 `LightBulb` (或將現有物件的類型變更為 `LightBulb`) 時，您可以針對在物件類型中定義的每個屬性指定值。 `LightBulb`

雖然物件類型為選用，但使用時可讓您探索物件更輕鬆。

- 有物件類型的物件最多可具有 50 個屬性。
- 無物件類型的物件最多可具有三個屬性。
- 一個物件只能與一個物件類型建立關聯。
- 在您的帳戶中，可以建立的物件類型數量不限。

物件類型不可變。物件類型建立之後，您就無法變更其名稱。您可以隨時棄用物件類型，避免新的物件與其建立關聯。您也可以刪除未和任何物件關聯的物件類型。

## 建立物件類型

您可以使用 `CreateThingType` 命令來建立物件類型：

```
$ aws iot create-thing-type

    --thing-type-name "LightBulb" --thing-type-properties
"thingTypeDescription=light bulb type, searchableAttributes=wattage,model"
```

`CreateThingType` 命令的回應會包含物件類型及其 ARN：

```
{
  "thingTypeName": "LightBulb",
  "thingTypeId": "df9c2d8c-894d-46a9-8192-9068d01b2886",
  "thingTypeArn": "arn:aws:iot:us-west-2:123456789012:thingtype/LightBulb"
}
```

## 列出物件類型

您可以使用 `ListThingTypes` 命令列出物件類型：

```
$ aws iot list-thing-types
```

此命 `ListThingTypes` 令會傳回在您的下列項目中定義之物件類型的清單 AWS 帳戶：

```
{
  "thingTypes": [
    {
      "thingTypeName": "LightBulb",
      "thingTypeProperties": {
        "searchableAttributes": [
          "wattage",
          "model"
        ],
        "thingTypeDescription": "light bulb type"
      },
      "thingTypeMetadata": {
```

```

        "deprecated": false,
        "creationDate": 1468423800950
    }
}
]
}

```

## 描述物件類型

您可以使用 `DescribeThingType` 命令獲得關於物件類型的資訊：

```
$ aws iot describe-thing-type --thing-type-name "LightBulb"
```

`DescribeThingType` 命令會傳回關於指定類型的資訊：

```

{
  "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
  }
}

```

## 將物件類型與物件建立關聯

在建立物件時，您可以使用 `CreateThing` 命令指定物件類型：

```
$ aws iot create-thing --thing-name "MyLightBulb" --thing-type-name "LightBulb" --
attribute-payload "{\"attributes\": {\"wattage\": \"75\", \"model\": \"123\"}}"
```

您可以隨時使用 `UpdateThing` 命令變更與物件關聯的物件類型：

```
$ aws iot update-thing --thing-name "MyLightBulb"
                        --thing-type-name "LightBulb" --attribute-payload '{"attributes\":
{"wattage\":"75\", \"model\":"123\"}}'
```

您也可以使用 UpdateThing 命令取消物件與物件類型的關聯。

## 棄用物件類型

物件類型不可變。物件類型定義之後就無法變更。不過，您可以棄用物件類型，避免使用者將其與任何新物件建立關聯。所有與該物件類型關聯的現有物件都不會變更。

若要棄用物件類型，請使用 DeprecateThingType 命令：

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType"
```

您可以使用 DescribeThingType 命令來查看結果：

```
$ 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
  }
}
```

棄用物件類型是可逆的操作。您可以使用 --undo-deprecate 旗標以及 DeprecateThingType CLI 命令來取消棄用：

```
$ aws iot deprecate-thing-type --thing-type-name "myThingType" --undo-deprecate
```

您可以使用 DescribeThingType CLI 命令查看結果：

```
$ 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,
  }
}
```

## 刪除物件類型

物件類型只有在棄用後才能刪除。若要刪除物件類型，請使用 DeleteThingType 命令：

```
$ aws iot delete-thing-type --thing-type-name "StopLight"
```

### Note

在您刪除物件類型之前，請先等待五分鐘後再淘汰該物件類型。

## 靜態物件群組

物件群組可讓您將多個物件分類成群組以方便同時管理。靜態物件群組包含一組使用主控台、CLI 或 API 來管理的物件。另一方面，[動態物件群組](#)會包含與指定查詢相符的物件。靜態物件群組還可包含其他靜態物件群組，您可建置一個群組階層。您可以將政策連接至父群組，而此政策會被其子群組以及父、子群組內的所有物件繼承。這可讓您輕易控制大量物件的許可。



**Note**

物件群組原則不允許存取 AWS IoT Greengrass 資料平面作業。若要允許物件存取 AWS IoT Greengrass 資料平面作業，請將權限新增至您附加至物件憑證的 AWS IoT 原則。如需詳細資訊，請參閱《AWS IoT Greengrass 開發人員指南》中的[裝置身分驗證和授權](#)。

您可以對靜態物件群組執行下列操作：

- 建立、描述或刪除群組。
- 新增物件至一個多個群組。
- 從群組移除物件。
- 列出您建立的群組。
- 列出群組的所有子群組 (直系與旁系的後代)。
- 列出群組裡的物件，包括其子群組裡的所有物件。
- 列出群組的所有上階群組 (直系與旁系的父群組)。
- 新增、刪除或更新群組的屬性。(屬性是您可用於存放群組相關資訊的名稱/值對。)
- 讓政策與群組連接或分離。
- 列出連接至群組的政策。
- 列出物件所繼承的政策 (透過連接至其群組或其父群組的政策)。
- 設定群組中物件的記錄選項。請參閱[設定 AWS IoT 記錄](#)。
- 建立傳送至群組及其子群組中所有物件並執行的任務。請參閱[任務](#)。

**Note**

將物件附加至 AWS IoT Core 原則所附加的靜態物件群組時，物件名稱必須符合用戶端 ID。

以下為靜態物件群組的限制：

- 一個群組最多只能有一個直系的父群組。
- 如果群組是另一個群組的子系，請在建立群組時指定該群組。
- 您之後便無法變更群組的父群組，因此請務必在建立群組所包含的任何子群組之前，先規劃群組階層並建立父群組。

- 物件所屬群組數有限。
- 在相同的階層內，您無法讓一個物件加入超過一個群組。(亦即，您無法讓物件加入父群組相同的兩個群組。)
- 您無法重新命名群組。
- 物件群組名稱無法包含國際字元，例如 û、é 和 ñ。
- 請勿在您的物件群組名稱中使用可識別個人身分的資訊。物件群組名稱可顯示於未加密的通訊和報告中。

讓政策跟群組連接和分離，可以在許多重要方面強化您的 AWS IoT 操作安全。透過裝置將政策連接至憑證再連接到物件的方式相當耗時，而且在裝置機群中要迅速更新或變更政策也很困難。若政策連接至物件的群組，在輪換物件的憑證時就可以省卻許多步驟。此外，物件變更群組成員資格時，政策就會動態套用至物件，所以每當群組中有裝置變更成員資格，您就不必重新建立一組複雜的許可。

## 建立靜態物件群組

使用 `CreateThingGroup` 命令建立動態物件群組：

```
$ aws iot create-thing-group --thing-group-name LightBulbs
```

該 `CreateThingGroup` 命令返回一個包含靜態物件群組的名稱、ID 和 ARN 的響應：

```
{
  "thingGroupName": "LightBulbs",
  "thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwxyz",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs"
}
```

### Note

我們不建議在物件群組名稱中使用個人識別資訊。

這個範例是在建立靜態物件群組時，為其指定一個父群組：

```
$ aws iot create-thing-group --thing-group-name RedLights --parent-group-name
LightBulbs
```

如先前所述，CreateThingGroup 命令傳回的回應會包含物件群組、ID 及 ARN：

```
{
  "thingGroupName": "RedLights",
  "thingGroupId": "abcdefghijklmnop12345678qrstuvwx",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
}
```

### ⚠ Important

在建立物件群組階層時，請注意下列限制：

- 物件群組只能有一個直接的父系群組。
- 物件群組可以擁有的直接子系群組數量有限。
- 群組階層的最大深度有限。
- 物件群組可以擁有的屬性數量有限。(屬性是您可用於存放群組相關資訊的名稱/值對。) 每個屬性名稱和每個值的長度也有限。

## 描述物件群組

您可以使用 DescribeThingGroup 命令獲得關於物件群組的資訊：

```
$ aws iot describe-thing-group --thing-group-name RedLights
```

DescribeThingGroup 命令會傳回關於指定群組的資訊：

```
{
  "thingGroupName": "RedLights",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights",
  "thingGroupId": "12345678abcdefghijklmnop12345678",
  "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"
},
}
```

## 新增一個物件到一個靜態的物件群組

您可以使用 `AddThingToThingGroup` 命令將物件新增至靜態物件群組：

```
$ aws iot add-thing-to-thing-group --thing-name MyLightBulb --thing-group-name
RedLights
```

`AddThingToThingGroup` 命令不會產生任何輸出。

### Important

一個物件最多可加入 10 個群組。但是在相同的階層內，您無法讓一個物件加入超過一個群組。(亦即，您無法讓物件加入父群組相同的兩個群組。)

如果物件屬於 10 個物件群組，而其中有一或多個群組為動態物件群組，則您可以使用 [overrideDynamicGroups](#) 旗標讓靜態群組的優先順序高於動態群組。

## 從靜態物件群組中移除一個物件

您可以使用 `RemoveThingFromThingGroup` 命令從群組移除物件：

```
$ aws iot remove-thing-from-thing-group --thing-name MyLightBulb --thing-group-name
RedLights
```

RemoveThingFromThingGroup 命令不會產生任何輸出。

## 列出物件群組中的物件

您可以使用 ListThingsInThingGroup 命令列出屬於群組的物件：

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs
```

ListThingsInThingGroup 命令會傳回該群組的物件清單：

```
{
  "things": [
    "TestThingA"
  ]
}
```

使用 --recursive 參數，您可以列出屬於某個群組及其任一子群組中的物件：

```
$ aws iot list-things-in-thing-group --thing-group-name LightBulbs --recursive
```

```
{
  "things": [
    "TestThingA",
    "MyLightBulb"
  ]
}
```

### Note

此操作為[最終一致](#)。換句話說，對物件群組的變更可能不會立即反映出來。

## 列出物件群組

您可以使用 ListThingGroups 命令來列出帳戶的物件群組：

```
$ aws iot list-thing-groups
```

此命ListThingGroups令會傳回您的下列項目中物件群組的清單 AWS 帳戶：

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

使用選用的篩選條件列出以特定群組為父群組的群組 (--parent-group) 或是名稱開頭為特定字首 (--name-prefix-filter) 的群組。--recursive 參數可讓您列出所有子群組，不只是物件群組的直系子群組：

```
$ aws iot list-thing-groups --parent-group LightBulbs
```

在此情況下，ListThingGroups命令會傳回在您的下列項目中定義之物群組的直接子群組清單 AWS 帳戶：

```
{
  "childGroups": [
    {
      "groupName": "RedLights",
      "groupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
    }
  ]
}
```

```
]
}
```

使用 `--recursive` 參數與 `ListThingGroups` 命令來列出物件群組的所有子群組，不只是直系子群組：

```
$ aws iot list-thing-groups --parent-group LightBulbs --recursive
```

`ListThingGroups` 命令會傳回一份清單，列出物件群組的所有子群組：

```
{
  "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

此操作為[最終一致](#)。換句話說，對物件群組的變更可能不會立即反映出來。

## 列出物件的群組

您可以使用 `ListThingGroupsForThing` 命令列出物件所屬的群組：

```
$ aws iot list-thing-groups-for-thing --thing-name MyLightBulb
```

`ListThingGroupsForThing` 命令會傳回一份清單，其中會列出此物件所屬的直接物件群組：

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

## 更新靜態物件群組

您可以使用 `UpdateThingGroup` 命令更新靜態物件群組的屬性：

```
$ 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` 命令的回應會包含更新之後的群組版本編號：

```
{
  "version": 4
}
```

### Note

一個物件可以擁有的屬性數量有限。

## 刪除物件群組

若要刪除物件群組，請使用 `DeleteThingGroup` 命令：



```
$ aws iot delete-thing-group --thing-group-name "RedLights"
```

DeleteThingGroup 命令不會產生任何輸出。

### Important

如果您嘗試刪除具有子物件群組的物件群組，就會收到錯誤：

```
A client error (InvalidRequestException) occurred when calling the
DeleteThingGroup
operation: Cannot delete thing group : RedLights when there are still child
groups attached to it.
```

刪除群組之前，請先刪除任何子群組。

您可以刪除具有子物件的群組，但由群組成員資格授予物件的任何許可將不再適用。在刪除已連接政策的群組之前，請確認移除這些許可不會使群組中的物件無法正常運作。此外，當雲端中的記錄正在更新時，顯示物件所屬群組的指令 (例如ListGroupsForThing) 可能會繼續顯示群組。

## 將政策附加到靜態物件群組

您可以使用 AttachPolicy 命令將政策連線至靜態物件群組，進而連線至該群組及其子群組中的所有物件。

```
$ aws iot attach-policy \  
  --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" \  
  --policy-name "myLightBulbPolicy"
```

AttachPolicy 命令不會產生任何輸出。

### Important

您可以附加至群組的政策數上限為兩個。

### Note

我們不建議在政策名稱中使用個人識別資訊。

--target 參數可為物件群組 ARN (如上所述)、憑證 ARN 或 Amazon Cognito Identity。如需關於政策、憑證與身分驗證的詳細資訊，請參閱 [身分驗證](#)。

如需詳細資訊，請參閱 [AWS IoT Core 政策](#)。

## 將政策與靜態物件群組分離

您可以使用 DetachPolicy 命令從物件群組分離政策，進而讓政策與該群組及其子群組中的所有物件分離。

```
$ aws iot detach-policy --target "arn:aws:iot:us-west-2:123456789012:thinggroup/LightBulbs" --policy-name "myLightBulbPolicy"
```

DetachPolicy 命令不會產生任何輸出。

## 列出連線至靜態物件群組的政策

您可以使用 ListAttachedPolicies 命令列出連線至群組的政策：

```
$ aws iot list-attached-policies --target "arn:aws:iot:us-west-2:123456789012:thinggroup/RedLights"
```

--target 參數可為物件群組 ARN (如上所述)、憑證 ARN 或 Amazon Cognito Identity。

加入選用的 --recursive 參數，納入與該群組之父群組連接的所有政策。

ListAttachedPolicies 命令會傳回政策清單：

```
{
  "policies": [
    "MyLightBulbPolicy"
    ...
  ]
}
```

## 列出政策所在的群組

您可以使用 ListTargetsForPolicy 命令列出政策連接的目標，包括任何群組：

```
$ aws iot list-targets-for-policy --policy-name "MyLightBulbPolicy"
```

新增選用的 `--page-size` *number* 參數來指定每次查詢所回傳的結果數量上限，並在後續呼叫使用 `--marker` *string* 參數以擷取下一組結果。

ListTargetsForPolicy 命令會傳回一份目標清單以及用於擷取更多結果的符記：

```
{
  "nextMarker": "string",
  "targets": [ "string" ... ]
}
```

## 為物件取得有效政策

您可以使用 GetEffectivePolicies 命令列出物件的有效政策，包括連接至該物件所屬任何群組的政策 (無論此群組是直系的父群組或旁系的上階群組)：

```
$ aws iot get-effective-policies \
  --thing-name "MyLightBulb" \
  --principal "arn:aws:iot:us-east-1:123456789012:cert/
a0c01f5835079de0a7514643d68ef8414ab739a1e94ee4162977b02b12842847"
```

使用 `--principal` 參數指定連接物件之憑證的 ARN。如果您使用 Amazon Cognito 身分驗證，請使用 `--cognito-identity-pool-id` 參數並新增選用的 `--principal` 參數來指定一個 Amazon Cognito 身分。如果您僅指定 `--cognito-identity-pool-id`，則會傳回該身分集區角色對於未授權使用者的相關政策。如果您兩者都使用，則會傳回該 identity pool 角色對於授權使用者的相關政策。

`--thing-name` 參數為選用，可用於取代 `--principal` 參數。若使用此參數，則會傳回連接物件所屬群組的政策，以及連接這些群組之父群組 (最高到階層裡之根群組) 的政策。

GetEffectivePolicies 命令會傳回政策清單：

```
{
  "effectivePolicies": [
    {
      "policyArn": "string",
      "policyDocument": "string",
      "policyName": "string"
    }
    ...
  ]
}
```

## MQTT 動作的測試授權

您可以使用 `TestAuthorization` 命令測試 [MQTT](#) 動作 (Publish、Subscribe) 是否可用於物件：

```
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` 參數指定連接物件之憑證的 ARN。若使用 Amazon Cognito Identity 驗證，請指定一個 Cognito Identity 為 `--principal` 或使用 `--cognito-identity-pool-id` 參數，或兩者並用。(如果您僅指定 `--cognito-identity-pool-id`，則需考量該 identity pool 角色對於未授權使用者的相關政策。如果您兩者都使用，則需考量該 identity pool 角色對於授權使用者的相關政策。

若要指定一個或多個您想要測試的 MQTT 動作，請列出資源以及執行 `--auth-infos` 參數後的動作類型。 `actionType` 欄位應包含 "PUBLISH"、"SUBSCRIBE"、"RECEIVE" 或 "CONNECT"。 `resources` 欄位應包含一份資源 ARN 的清單。如需詳細資訊，請參閱 [AWS IoT Core 政策](#)。

若要測試新增政策的效果，請為其指定 `--policy-names-to-add` 參數。若要測試移除政策的效果，請為其指定 `--policy-names-to-skip` 參數。

您可以使用選用的 `--client-id` 參數進一步加強結果。

`TestAuthorization` 命令會傳回您指定的每筆 `--auth-infos` 查詢所允許或拒絕的動作詳細資訊：

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

## 動態物件群組

動態物件群組是從登錄中的特定搜尋查詢建立的。搜索查詢參數，例如設備連接，設備陰影創建和 AWS IoT Device Defender 違規數據支持此。動態物件群組需要啟用叢集索引，才能建立裝置資料的索引、搜尋及彙總。您可以在建立動態物件群組之前，使用叢集索引搜尋查詢來預覽動態物件群組中的物件。如需詳細資訊，請參閱 [機群索引](#) 及 [查詢語法](#)。

### Note

動態物件群組作業會在登錄作業下計量。如需詳細資訊，請參閱 [AWS IoT Core 其他測量詳細資訊](#)。

動態物件群組與靜態物件群組的差異如下：

- 物件成員資格沒有明確定義。欲建立動態物件群組，請定義 [搜尋查詢字串](#) 以決定群組成員資格。

- 動態物件群組無法成為階層的一部分。
- 動態物件群組無法套用政策。
- 您會使用不同的命令集來建立、更新和刪除動態物件群組。對於所有其他作業，您可以對這兩種類型的物件群組使用相同的指令。
- 每個動態群組的數目 AWS 帳戶 是[有限的](#)。
- 請勿在您的物件群組名稱中使用可識別個人身分的資訊。物件群組名稱可顯示於未加密的通訊和報告中。

如需靜態物件群組的詳細資訊，請參閱 [靜態物件群組](#)。

## 動態物件群組的使用案例

您可以針對下列使用案例使用動態物件群組：

### 指定動態物件群組作為工作的目標

以動態物件群組作為目標建立連續工作，可讓您在裝置符合所需準則時自動鎖定目標裝置。準則可以是連線狀態或儲存在登錄或陰影 (例如軟體版本或模型) 中的任何準則。如果動態物件群組中沒有出現物件，它就不會從工作接收工作的工作文件。

例如，如果您的裝置群需要進行韌體更新，以將更新程序中斷的風險降到最低，而您只想在電池續航力超過 80% 的裝置上更新韌體。您可以建立名為 80 的動態物件群組 PercentBatteryLife，其中僅包含電池續航力超過 80% 的裝置，並將其用作工作的目標。只有符合電池壽命標準的裝置才會收到韌體更新。當裝置達到 80% 的電池壽命標準時，它們會自動新增至動態物件群組，並會收到韌體更新。

您也可能有多種不同韌體或作業系統的裝置型號，因此需要進行不同版本的新軟體更新。對於具有連續工作的動態群組，這是最常見的使用案例，您可以在其中為每個裝置型號、韌體和作業系統組合建立動態群組。然後，您可以為每個動態群組設定連續工作，以在裝置根據定義的準則自動成為這些群組的成員時推送軟體更新。

如需將物件群組指定為工作目標的詳細資訊，請參閱 [CreateJob](#)。

### 使用動態群組成員資格變更來執行所需動作

每次將裝置新增至動態物件群組或從動態物件群組中移除時，都會傳送通知至 MQTT 主題，做為 [登錄事件](#) 更新的一部分。您可以設定 [AWS IoT Core 規則](#) 以根據動態群組成員資格更新與 AWS 服務互

動，並採取所需的動作。範例動作包括寫入 Amazon DynamoDB、叫用 Lambda 函數，或傳送通知至 Amazon SNS。

## 將裝置新增至動態物件群組以進行自動違規偵測

AWS IoT Device Defender 偵測客戶可以在動態物件群組上[定義安全性設定檔](#)。群組上定義的安全性設定檔會自動偵測動態物件群組的裝置是否有違規。

## 在動態物群組上設定記錄層級，以使用精細的記錄來觀察裝置

您可以在動態物群組上指定記錄層級。如果您只想為符合特定準則的裝置自訂記錄層級和詳細資料，此功能非常有用。例如，如果您懷疑具有特定韌體版本的裝置在特定規則的已發佈主題上造成錯誤，您可能會想要設定詳細的記錄來偵錯這些問題。在這種情況下，您可以為具有此固件版本的所有設備創建一個動態組，我們假設將其存儲為註冊表屬性或設備陰影中。然後，您可以設定偵錯層級，並將記錄目標定義為此動態物件群組。如需有關精細記錄的詳細資訊，請參閱[AWS IoT 使用記 CloudWatch 錄監視](#)。如需有關如何為特定物件群組指定記錄層級的詳細資訊，請參閱[設定資源特定的登入](#)。AWS IoT

## 建立動態物件群組

使用 `CreateDynamicThingGroup` 命令建立動態物件群組。若要為 80 個 `PercentBatteryLife` 案例建立動態物件群組，請使用 `create-dynamic-thing-group` CLI 命令：

```
$ aws iot create-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --query-string "attributes.batteryLife80"
```

### Note

請勿在動態物件群組名稱中使用個人識別資訊。

指 `CreateDynamicThingGroup` 命令會傳回回應。回應包含物件群組的索引名稱、查詢字串、查詢版本、物件群組名稱、物群組 ID 以及 Amazon 資源名稱 (ARN)：

```
{
  "indexName": "AWS_Things",
  "queryVersion": "2017-09-30",
  "thingGroupName": "80PercentBatteryLife",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/80PercentBatteryLife",
```

```
"queryString": "attributes.batterylife80\n",
"thingGroupId": "abcdefgh12345678ijklmnop12345678qrstuvwxyz"
}
```

動態物件群組的建立不會立即發生。動態物件群組回填需要一些時間才能完成。當您建立動態物件群組時，群組的狀況會設定為BUILDING。回填處理完成時，狀態會變更為ACTIVE。欲檢查動態物件群組的狀況，請使用「群組」([DescribeThingGroup](#)) 指令。

## 描述動態物件群組

使用 DescribeThingGroup 命令以取得動態物件群組的相關資訊：

```
$ aws iot describe-thing-group --thing-group-name "80PercentBatteryLife"
```

DescribeThingGroup 命令會傳回關於指定群組的資訊：

```
{
  "status": "ACTIVE",
  "indexName": "AWS_Things",
  "thingGroupName": "80PercentBatteryLife",
  "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/80PercentBatteryLife",
  "queryString": "attributes.batterylife80\n",
  "version": 1,
  "thingGroupMetadata": {
    "creationDate": 1548716921.289
  },
  "thingGroupProperties": {},
  "queryVersion": "2017-09-30",
  "thingGroupId": "84dd9b5b-2b98-4c65-84e4-be0e1ecf4fd8"
}
```

在動態物件群組DescribeThingGroup上執行會傳回動態物件群組特定的屬性。示例返回屬性是 queryString 和狀態。

動態物件群組的狀態可能會需要以下的值：

ACTIVE

動態物件群組已準備就緒。



## BUILDING

正在建立動態群組，且正在處理物件成員資格。

## REBUILDING

正在更新動態物件群組的成員資格，接著是調整該群組搜尋查詢。

### Note

建立動態物件群組之後，不論其狀態為何，都可以使用它。只有狀態為 ACTIVE 的動態物件群組，才會包含所有與該動態物件群組搜尋查詢相符的物件。狀態為 BUILDING 和 REBUILDING 的動態物件群組，則可能不會包含所有符合搜尋查詢的物件。

## 更新動態物件群組

使用 `UpdateDynamicThingGroup` 命令來更新動態物件群組的屬性，包括該群組的搜尋查詢。下列指令會更新兩個屬性。一個是物件群組描述，另一個是將成員資格條件變更為電池壽命 > 85 的查詢字串：

```
$ aws iot update-dynamic-thing-group --thing-group-name "80PercentBatteryLife" --thing-group-properties "thingGroupDescription=\"This thing group contains devices with a battery life greater than 85 percent.\" --query-string "attributes.batteryLife85"
```

`UpdateDynamicThingGroup` 命令的回應會包含更新之後的群組版本編號：

```
{
  "version": 2
}
```

動態物件群組的更新不會立即發生。動態物件群組回填需要一些時間才能完成。當您更新動態物件群組時，群組的狀態會變更為 `REBUILDING` 而群組更新其成員資格。回填處理完成時，狀態會變更為 `ACTIVE`。欲檢查動態物件群組的狀況，請使用「群組」([DescribeThingGroup](#)) 指令。

## 刪除動態物件群組

使用 `DeleteDynamicThingGroup` 命令來刪除動態物件群組：

```
$ aws iot delete-dynamic-thing-group --thing-group-name "80PercentBatteryLife"
```

DeleteDynamicThingGroup 命令不會產生任何輸出。

雲端中的記錄正在更新時，顯示物件所屬群組的命令 (例如 ListGroupsForThing) 可能仍會顯示出該群組。

## 動態與靜態物件群組限制

動態物件群組與靜態物群組有下列限制：

- 物件群組可擁有的屬性數目有[限](#)。
- 物件所屬群組數有[限](#)。
- 您無法重新命名物件群組。
- 物件群組名稱無法包含國際字元，例如 û、é 和 ñ。

## 動態物件群組限制

動態物件群組具有下列限制：

### 機群索引

啟用叢集索引服務後，您可以在裝置叢集上執行搜尋查詢。您可以在叢集索引回填完成後建立及管理動態物件群組。回填程序的完成時間直接取決於您在註冊的裝置叢集的大小。AWS 雲端在您為動態物件群組啟用機群索引服務後，除非您刪除所有動態物件群組，否則無法停用服務。

#### Note

如果您有查詢機群索引的許可，您就可以存取整個機群的物件資料。

## 動態物件群組的數量有限

動態物群組的數目是[有限](#)的。

## 成功的命令可以記錄錯誤

當您建立或更新動態物件群組時，有些物件可能符合包含在動態物群組中的資格，但不會將它們新增至動態物群組。[在記錄錯誤並產生指標時，該案例將導致成功的建立或更新命令。AddThingToDynamicThingGroupsFailed](#)單一測量結果可以代表多個記錄項目。

發生下列情況時，會在 CloudWatch 記錄檔中建立[錯誤記錄項目](#)：

- 合格物件無法新增至動態物件群組。
- 物件會從動態物群組中移除，以將其新增至另一個群組。

當物件符合新增至動態物件群組的資格時，請考慮下列事項：

- 該物件是否已經在盡可能多的群組中存在？(請參閱[限制數量](#))。
  - 否：該物件被新增到動態物件組。
  - 是：事情是否是任何動態物件群組的成員？
    - 否：無法將物件新增至動態物件群組，系統已記錄錯誤，並產生 [AddThingToDynamicThingGroupsFailed](#) 指標。
    - 是：要加入的動態物件群組是否比已是成員的任何動態物件群組還舊？
      - 否：無法將物件新增至動態物件群組，系統已記錄錯誤，並產生 [AddThingToDynamicThingGroupsFailed](#) 指標。
      - 是：從最近的動態物件群組中移除物件、記錄錯誤，然後將物件新增至動態物群組。這會產生錯誤和移除物件之動態物件群組的 [AddThingToDynamicThingGroupsFailed](#) 指標。

當動態物件群組中的物件不再符合搜尋查詢時，該物件會從動態物群組中移除。同樣地，當物件更新以符合動態物件群組的搜尋查詢時，物件會如先前所述新增至群組。這些新增和移除都是正常的，不會產生錯誤日誌項目。

啟用 `overrideDynamicGroups` 時，靜態群組會優先於動態群組

物件所屬群組數有限。當您使用「[AddThingToThing群組](#)」或「[UpdateThingGroupsFor物件](#)」指令來更新物件成員資格時，新增 `--overrideDynamicGroups` 參數會使靜態物件群組優先於動態物件群組。

當您將物件新增至靜態物件群組時，請考慮下列事項：

- 物件是否已屬於群組的最大數量？
  - 否：該物件會新增到靜態物件群組。
  - 是：該物件是否在任何動態群組中？
    - 否：不能將該物件新增到物件群組。命令會引發例外狀況。
    - 是：是否已啟用 `--overrideDynamicGroups`？
      - 否：不能將該物件新增到物件群組。命令會引發例外狀況。

- 是：該物件會從最近建立的動態物件群組中移除、記錄錯誤，並針對移除該物件的動態物件群組產生 [AddThingToDynamicThingGroupsFailed 指標](#)。然後，事情會新增到靜態物件群組中。

## 較舊的動態物件群組優先於較新的動態物件群組

物件所屬群組數**有限**。當建立或更新作業建立物件的其他群組資格，且物件已達到其群組限制時，可能會從其他動態物件群組移除以啟用此新增。如需有關如何發生這種情況的詳細資訊，請參閱 [成功的命令可以記錄錯誤](#) 和 [啟用 `overrideDynamicGroups` 時，靜態群組會優先於動態群組](#) 以取得範例。

從動態物件群組中移除物件時，會記錄錯誤並引發事件。

## 您無法將政策套用至動態物件群組

嘗試將政策套用至動態物件群組會產生例外狀況。

## 動態物件群組成員資格為最終一致

只會針對登錄檔對物件的最終狀態進行評估。如果狀態快速更新，可能會略過中繼狀態。避免將規則或工作與其成員資格取決於中介狀態的動態物件群組產生關聯。

# 標記您的 AWS IoT 資源

為了協助您管理和組織物件群組、物件類型、主題規則、任務、排程稽核及安全性描述檔，您可以選擇性地以標籤形式將您自己的中繼資料指派給其中每個資源。本節說明標籤並示範如何建立它們。

為了協助您管理與物件相關的成本，您可以建立包含物件的[帳單群組](#)。您可以將包括中繼資料的標籤指派到每個帳單群組。本節也會討論可用來建立和管理的帳單群組和命令。

## 標籤基本概念

您可以使用標籤以不同的方式 (例如，依目的、擁有者或環境) 對 AWS IoT 資源進行分類。當您有許多相同類型的資源時，這會很有用，因為您可以依據先前指派的標籤快速識別資源。每個標籤皆包含由您定義的一個「索引鍵」與選擇性的「值」。例如，您可以為您的物件類型定義一組標籤，協助您以類型追蹤裝置。我們建議您為每種資源類型建立符合您需求的一組標籤金鑰。使用一致的標籤金鑰組可讓您更輕鬆管理您的資源。

您可以根據新增或套用的標籤來搜尋與篩選資源。您也可以使用帳單群組標籤來分類和追蹤您的成本。您也可以使用標籤來控制對您資源的存取，如 [搭配 IAM 政策使用標籤](#) 所述。

為了方便使用，AWS 管理主控台標籤編輯器提供了一種集中且統一的方式來建立和管理標籤。如需詳細資訊，請參閱[使用AWS 管理主控台標籤編輯器](#)。

您也可以使用 AWS CLI 和 AWS IoT API 處理標籤。當您使用下列命令在 Tags 欄位中建立標籤時，您可將標籤與物件群組、物件類型、主題規則、任務、安全性設定檔、政策、帳單群組、物件相關套件和版本建立關聯：

- [CreateBillingGroup](#)
- [CreateDestination](#)
- [CreateDeviceProfile](#)
- [CreateDynamicThingGroup](#)
- [CreateJob](#)
- [CreateOTAUpdate](#)
- [CreatePolicy](#)
- [CreateScheduledAudit](#)
- [CreateSecurityProfile](#)

- [CreateServiceProfile](#)
- [CreateStream](#)
- [CreateThingGroup](#)
- [CreateThingType](#)
- [CreateTopicRule](#)
- [CreateWirelessGateway](#)
- [CreateWirelessDevice](#)

您可以使用下列命令新增、修改或刪除支援標記功能的現有資源標籤：

- [TagResource](#)
- [ListTagsForResource](#)
- [UntagResource](#)

您可以編輯標籤金鑰和值，並且可以隨時從資源移除標籤。您可以將標籤的值設為空白字串，但您無法將標籤的值設為 Null。若您將與現有標籤具有相同鍵的標籤新增到該資源，則新值會覆寫舊值。如果您刪除資源，也會刪除與該資源相關聯的任何標籤。

## 標籤的限制與上限

以下基本限制適用於標籤：

- 每個資源的標籤數上限：50
- 金鑰長度上限：127 個 UTF-8 Unicode 字元
- 值長度上限：255 個 UTF-8 Unicode 字元
- 標籤鍵與值皆區分大小寫。
- 請勿於標籤名稱或值中使用 `aws:` 字首。它保留供 AWS 使用。您不可編輯或刪除具此字首的標籤名稱或值。具此字首的標籤，不算在受資源限制的標籤計數內。
- 如果您的標記結構描述是跨多項服務和資源使用，請記得其他服務可能會有字元使用限制。允許使用的字元包括可用 UTF-8 表示的英文字母、空格和數字，以及以下特殊字元：`+ - = . _ : / @`。

## 搭配 IAM 政策使用標籤

您可以在用於 AWS IoT API 動作的 IAM 政策中，套用以標籤為基礎的資源層級許可。這可讓您更有效地控制使用者可以建立、修改或使用哪些資源。您可以使用 Condition 元素 (也稱為 Condition 區塊)，以及 IAM 政策中的以下條件內容金鑰和值，來根據資源標籤控制使用者存取 (許可)：

- 使用 `aws:ResourceTag/tag-key: tag-value` 以允許或拒絕資源上具有特定標籤的使用者動作。
- 使用 `aws:RequestTag/tag-key: tag-value` 以在提出 API 請求時，要求使用 (或不使用) 特定標籤，以建立或修改允許標籤的資源。
- 使用 `aws:TagKeys: [tag-key, ...]` 以在提出 API 請求時，要求使用 (或不使用) 特定標籤金鑰集，以建立或修改允許標籤的資源。

### Note

IAM 政策中的條件內容索引鍵和值僅適用於那些 AWS IoT 動作，其中能夠標記之資源的識別碼是必要參數。例如，根據條件前後關聯索引鍵與值，不允許或拒絕使用，因為此請求中未參考任何可標記資源 (物群組、物件類型、主題規則、工作或安全性設定檔)。 [DescribeEndpoint](#) 如需有關可標記 AWS IoT 資源及其支援之條件索引鍵的詳細資訊，請參閱的 [動作、資源和條件索引鍵](#)。AWS IoT

如需使用標籤的詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的 [使用標籤控制](#)。該指南的 [IAM JSON 政策參考](#) 章節有詳細的語法、說明，還有元素、變數範例，以及在 IAM 中的 JSON 政策評估邏輯。

以下範例政策會針對 ThingGroup 動作套用兩個以標籤為基礎的限制。受到此政策限制的 IAM 使用者：

- 無法建立包含 "env=prod" 標籤的物件群組 (在範例中，請參閱行 "aws:RequestTag/env" : "prod")。
- 無法修改或存取具有現有標籤 "env = prod" 的物件群組 (在範例中，請參閱此行 "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": "*"
  }
]
}

```

您也可以透過將其包含在清單中，為特定標籤金鑰指定多個標籤值，如下所示：

```
"StringEquals" : {
```



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

### Note

如果您允許或拒絕使用者根據標籤存取資源，請務必考慮明確拒絕使用者將這些標籤新增至相同資源或從中移除的能力。否則，使用者可能透過修改標籤來避開您的限制，並取得資源的存取。

## 帳單群組

AWS IoT 不允許您將標籤直接套用至個別項目，但它可讓您將項目放在帳單群組中，並將標籤套用至這些項目。對於 AWS IoT，以標籤為基礎的成本和使用情況資料配置僅限於帳單群組。

AWS IoT Core 無法將 LoRa WAN 資源 (例如無線裝置和閘道) 新增至計費群組。不過，它們可以與 AWS IoT 項目相關聯，可以新增至帳單群組。

下列命令可供使用：

- [AddThingToBillingGroup](#) 將項目新增至帳單群組。
- [CreateBillingGroup](#) 建立帳單群組。
- [DeleteBillingGroup](#) 刪除帳單群組。
- [DescribeBillingGroup](#) 傳回帳單群組的相關資訊。
- [ListBillingGroups](#) 列出您建立的帳單群組。
- [ListThingsInBillingGroup](#) 列出您已新增至指定帳單群組的項目。
- [RemoveThingFromBillingGroup](#) 從計費群組中移除指定的項目。
- [UpdateBillingGroup](#) 更新帳單群組的相關資訊。
- [CreateThing](#) 可讓您在建立物件時指定物件的帳單群組。
- [DescribeThing](#) 傳回物件的描述，包括物件所屬的帳單群組 (如果有的話)。

AWS IoT 無線 API 提供這些動作，將無線裝置和閘道與 AWS IoT 物件建立關聯。

- [AssociateWirelessDeviceWithThing](#)
- [AssociateWirelessGatewayWithThing](#)

## 檢視成本分配和用量資料

您可以使用帳單群組標籤來分類和追蹤您的成本。當您將標記套用至帳單群組 (以及其包含的項目) 時, AWS 會以逗號分隔值 (CSV) 檔案的形式產生成本分配報告, 其中包含您的使用量和費用依標記彙總。您可以套用代表業務類別 (例如成本中心、應用程式名稱或擁有者) 的標籤, 來整理多個服務中的成本。如需使用標籤進行成本分配的詳細資訊, 請參閱 [《AWS 帳單和成本管理使用者指南》](#) 中的 [使用成本分配標籤](#)。

### Note

若要準確地將用量和成本資料與您已置於帳單群組的物件建立關聯, 每個裝置或應用程式皆必須:

- 在中註冊為物件 AWS IoT。如需詳細資訊, 請參閱 [管理裝置 AWS IoT](#)。
- 只使用物件名稱作為用戶端識別碼, 透過 MQTT Connect 線至 AWS IoT 訊息代理程式。如需詳細資訊, 請參閱 [the section called “裝置通訊協定”](#)。
- 使用與物件關聯的用戶端憑證驗證。

帳單群組可使用以下定價維度 (根據與帳單群組關聯的物件活動):

- 連線能力 (根據做為用戶端 ID 以用於連線的物件名稱)。
- 簡訊 (根據物件傳入和傳出的訊息; 僅限 MQTT)。
- 影子操作 (根據其訊息觸發影子更新的物件)。
- 觸發的規則 (根據其傳入訊息已觸發規則的物件; 不適用於 MQTT 生命週期事件所觸發的那些規則)。
- 物件索引更新 (根據新增到索引的物件)。
- 遠端動作 (根據更新的物件)。
- [AWS IoT Device Defender 偵測報告](#) (根據報告其活動的物件)。

根據標籤的成本和用量資料 (以及帳單群組的報告) 不會反映以下活動:

- 裝置登錄操作 (包括物件、物件群組和物件類型的更新)。如需詳細資訊, 請參閱 [管理裝置 AWS IoT](#)。
- 物件群組索引更新 (在新增物件群組時)。
- 索引搜尋查詢。

- [裝置佈建](#).
- [AWS IoT Device Defender 稽核報告](#)。

# 中的安全性 AWS IoT

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。[共同責任模型](#) 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。AWS 還為您提供可以安全使用的服務。在 [AWS 合規計畫](#) 中，第三方稽核員會定期測試並驗證我們的安全功效。若要深入瞭解適用於的規範遵循計劃 AWS IoT，請參閱 [合規方案的 AWS 服務範圍](#)。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您的公司的要求和適用法律和法規。

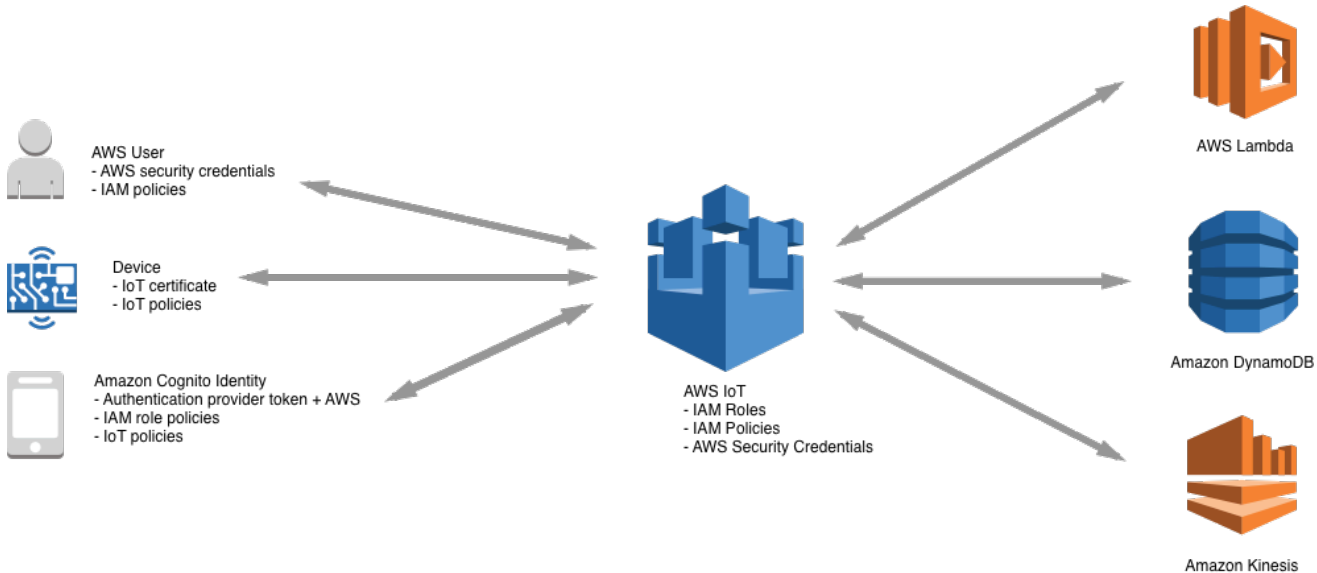
本文件可協助您瞭解如何在使用時套用共同責任模型 AWS IoT。下列主題說明如何設定 AWS IoT 以符合安全性與合規性目標。您也會學到如何使用其他可協助您監控和保護 AWS IoT 資源的 AWS 服務。

## 主題

- [AWS IoT 安全性](#)
- [身分驗證](#)
- [授權](#)
- [資料保護 AWS IoT Core](#)
- [的身分識別與存取管理 AWS IoT](#)
- [記錄和監控](#)
- [AWS IoT 核心的合規性驗證](#)
- [AWS IoT 核心的韌性](#)
- [AWS IoT Core 與介面 VPC 端點搭配使用](#)
- [基礎結構安全 AWS IoT](#)
- [使用 Core 對生產車隊或設備進 AWS IoT 行安全監控](#)
- [安全性最佳做法 AWS IoT Core](#)
- [AWS 培訓和認證](#)

# AWS IoT 安全性

每個連線的裝置或用戶端都必須具有憑證才能與 AWS IoT 互動。所有進出 AWS IoT 的流量都會透過傳輸層安全性 (TLS) 安全地傳送。AWS 雲端安全機制可在資料 AWS IoT 與其他 AWS 服務之間移動時保護資料。



- 您需負責管理 AWS IoT 中的裝置憑證 (X.509 憑證、AWS 憑證、Amazon Cognito 身分、聯合身分或自訂身分驗證字符) 以及政策。如需詳細資訊，請參閱 [金鑰管理 AWS IoT](#)。您務必為每一裝置指派唯一身分，並管理每個裝置或裝置群組的許可。
- 您的裝置透過安全 TLS 連線 AWS IoT 使用 X.509 憑證或 Amazon Cognito 身分識別來連線。在研究和開發過程中，對於進行 API 調用或使用的某些應用程序 WebSockets，您還可以使用 IAM 用戶和組或自定義身分驗證令牌進行身份驗證。如需詳細資訊，請參閱 [IAM 使用者、群組和角色](#)。
- 使用驗 AWS IoT 證時，訊息代理程式負責驗證您的裝置、安全地擷取裝置資料，以及授與或拒絕您使用規則為裝置指定的存取權限。AWS IoT
- 使用自訂驗證時，自訂授權者必須負責驗證您的裝置，以及授予或拒絕您使用 AWS IoT 或 IAM 政策為裝置指定的存取權限。
- AWS IoT 規則引擎根據您定義的規則將設備數據轉發到其他設備或其他 AWS 服務。它使用安全地 AWS Identity and Access Management 將數據傳輸到其最終目的地。如需詳細資訊，請參閱 [的身分識別與存取管理 AWS IoT](#)。

# 身分驗證

身分驗證是一種機制，您可以用來驗證用戶端或伺服器的身分。伺服器驗證是設備或其他客戶端確保它們與實際 AWS IoT 端點進行通信的過程。客戶端身份驗證是設備或其他客戶端進行身份驗證的過程 AWS IoT。

## AWS 培訓和認證

參加以下課程以了解中的身份驗證 AWS IoT：[深入了解 AWS IoT 身份驗證和授權](#)。

## X.509 憑證概觀

X.509 憑證為數位憑證，其依據 [X.509 公有金鑰基礎設施標準](#)，將公有金鑰與憑證內含的身分建立關聯。X.509 憑證由稱為認證機構 (CA) 的可信任實體發行。CA 負責維護一個或多個稱為憑證授權機構憑證的特殊憑證，用以發行 X.509 憑證。僅認證機構可存取憑證授權機構憑證。X.509 憑證鏈同時用於用戶端的伺服器身分驗證，和伺服器的用戶端身分驗證。

## 伺服器驗證

當您的裝置或其他用戶端嘗試連線時 AWS IoT Core，AWS IoT Core 伺服器會傳送 X.509 憑證，讓您的裝置用來驗證伺服器。身分驗證是透過在 TLS 層驗證 [X.509 憑證鏈](#) 進行。這與您造訪 HTTPS URL 時瀏覽器所使用的方法相同。如果您要使用自己的憑證授權單位的憑證，請參閱 [管理您的憑證授權機構憑證](#)。

當您的裝置或其他用戶端與 AWS IoT Core 端點建立 TLS 連線時，會 AWS IoT Core 顯示憑證鏈結，供裝置用來驗證裝置是否正在與其他伺服器進行通訊，AWS IoT Core 而非模擬 AWS IoT Core 其他伺服器。所顯示的鏈結取決於裝置連線到的端點類型，以及用戶端在 TLS 握手期間交 AWS IoT Core 涉的 [加密套件](#) 的組合。

## 端點類型

AWS IoT Core 支援兩種不同的資料端點類型，`iot:Data` 以及 `iot:Data-ATS`。`iot:Data` 端點顯示由 [VeriSign 類別 3 公用主要 G5 根 CA 憑證簽署的憑證](#)。`iot:Data-ATS` 端點顯示由 [Amazon 信任服務 CA 簽署的伺服器憑證](#)。

ATS 端點呈現的憑證由 Starfield 交叉簽署。某些 TLS 用戶端實作需要驗證信任根，並要求在用戶端的信任存放區中安裝 Starfield 憑證授權機構憑證。

**⚠ Warning**

不建議使用雜湊整個憑證 (包括發行者名稱等) 的憑證關聯方法，因為這會導致憑證驗證失敗，因為我們提供的 ATS 憑證是由 Starfield 交叉簽署，且具有不同的發行者名稱。

**⚠ Important**

除非您的裝置需要 Symantec 或 Verisign 憑證授權機構憑證，否則請使用 `iot:Data-ATS` 端點。Symantec 和 Verisign 憑證已被棄用，大多數 Web 瀏覽器不再支援。

您可以使用 `describe-endpoint` 命令建立 ATS 端點。

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

`describe-endpoint` 命令會傳回下列格式的端點。

```
account-specific-prefix.iot.your-region.amazonaws.com
```

**i Note**

第一次呼叫 `describe-endpoint` 時，會建立一個端點。所有後續呼叫 `describe-endpoint` 會傳回相同的端點。

為了向下相容，AWS IoT Core 仍支援賽門鐵克端點。如需詳細資訊，請參閱 [AWS IoT Core 如何協助客戶應對即將到來的不信任 Symantec 憑證授權單位](#)。在 ATS 端點上運作的裝置將能與同一帳戶中運作於 Symantec 端點上的裝置完全互通，而不需要進行任何重新註冊。

**i Note**

若要在 AWS IoT Core 主控台中查看您的 `iot:Data-ATS` 端點，請選擇 [設定]。主控台只會顯示 `iot:Data-ATS` 端點。依預設，`describe-endpoint` 命令會顯示 `iot:Data` 端點以提供回溯相容性。若要查看 `iot:Data-ATS` 端點，請指定 `--endpointType` 參數，如前一個範例所示。

## 使用適 `IotDataPlaneClient` 用於 Java 的 AWS 開發套件建立

預設情況下，[適用於 Java 的 AWS SDK-版本 2](#) 會使用 `iot:Data` 端點建立一個 `IotDataPlaneClient`。若要建立使用 `iot:Data-ATS` 端點的用戶端，您必須執行下列動作。

- 使用 [DescribeEndpoint](#) API 建立 `iot:Data-ATS` 端點。
- 在建立 `IotDataPlaneClient` 時指定該端點。

下列範例會同時執行這些操作。

```
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();
}
```

## 伺服器身分驗證的憑證授權機構憑證

根據您使用的資料端點類型以及您已交涉的加密套件而定，AWS IoT Core 伺服器驗證憑證會由下列其中一個根 CA 憑證簽署：

Amazon Trust Services 端點 (首選)

### Note

您可能需要在這些連結上按一下滑鼠右鍵，然後選取 `Save link as...` (另存連結為...) 以將這些憑證儲存為檔案。

- RSA 2048 位元金鑰：[Amazon Root CA 1](#)。
- RSA 4096 位元金鑰：Amazon Root CA 2。保留以供日後使用。
- ECC 256 位元金鑰：[Amazon Root CA 3](#)。
- ECC 384 位元金鑰：Amazon Root CA 4。保留以供日後使用。



這些憑證都是由 [Starfield 根憑證授權機構憑證](#) 交叉簽署的。從 2018 年 5 月 9 日 AWS IoT Core 在亞太區 AWS IoT Core 域 (孟買) 區域推出的所有新區域都只提供 ATS 憑證。

### VeriSign 端點 (舊版)

- RSA 2048 位元金鑰：[VeriSign 類別 3 公用主要 G5 根 CA 憑證](#)

### 伺服器身分驗證準則

有許多變數會影響裝置驗證 AWS IoT Core 伺服器驗證憑證的能力。例如，裝置的記憶體可能受限，無法保存所有可能的根憑證授權機構憑證，或者裝置可能實作非標準的憑證驗證方法。基於這些原因，我們建議您遵循以下準則：

- 我們建議您使用 ATS 端點並安裝所有支援的 Amazon Root CA 憑證。
- 如果您無法在裝置上存放全部這些憑證，而且您的裝置未使用 ECC 式驗證，則可以略過 [Amazon Root CA 3](#) 和 [Amazon Root CA 4](#) ECC 憑證。如果您的裝置未實作 RSA 式憑證驗證，則可以略過 [Amazon Root CA 1](#) 和 [Amazon Root CA 2](#) RSA 憑證。您可能需要在這些連結上按一下滑鼠右鍵，然後選取 Save link as... (另存連結為...) 以將這些憑證儲存為檔案。
- 如果您在連線至 ATS 端點時遇到伺服器憑證驗證問題，請嘗試將相關的交叉簽署 Amazon Root 憑證授權機構憑證新增至信任存放區。您可能需要在這些連結上按一下滑鼠右鍵，然後選取 Save link as... (另存連結為...) 以將這些憑證儲存為檔案。
  - [交叉簽署 Amazon Root CA 1](#)
  - [交叉簽署的 Amazon Root CA 2](#) - 預留以供日後使用。
  - [交叉簽署 Amazon Root CA 3](#)
  - [交叉簽署的 Amazon Root CA 4](#) - 預留以供日後使用。
- 如果您遇到伺服器憑證驗證問題，則您的裝置可能需要明確地信任根 CA。嘗試將 [Starfield Root CA Certificate](#) 新增到您的信任存放區。
- 如果您在執行上述步驟後仍然遇到問題，請聯絡 [AWS 開發人員支援](#)。

#### Note

憑證授權機構憑證在過期日期之後，即無法用於驗證伺服器的憑證。憑證授權機構憑證過期日期之前，可能需要進行替代。請確認您能夠於所有裝置或用戶端更新根憑證授權機構憑證，以維持連線能力並取得最新的安全最佳實務。

**Note**

在裝置程式碼 AWS IoT Core 中連線時，請將憑證傳遞至您用來連線的 API。您使用的 API 會因 SDK 而異。如需詳細資訊，請參閱 [AWS IoT Core 裝置開發套件](#)。

## 用戶端身分驗證

AWS IoT 支援三種類型的身分識別主體進行裝置或用戶端驗證：

- [X.509 用戶端憑證](#)
- [IAM 使用者、群組和角色](#)
- [Amazon Cognito 身分](#)

這些身分可用於裝置、行動裝置、網頁或桌面應用程式。它們甚至可以由輸入 AWS IoT 命令行界面 ( CLI ) 命令的用戶使用。一般而言，AWS IoT 裝置會使用 X.509 憑證，而行動應用程式則使用 Amazon Cognito 身分識別。Web 和桌面應用程式則使用 IAM 或聯合身分。AWS CLI 命令會使用 IAM。如需 IAM 身分的詳細資訊，請參閱 [身分識別與存取管理 AWS IoT](#)。

### X.509 用戶端憑證

X.509 憑證提供 AWS IoT 驗證用戶端和裝置連線的能力。用戶端憑證必須 AWS IoT 先註冊，用戶端才能與之通訊 AWS IoT。客戶端證書可以在同一個地區的多個 AWS 帳戶中註冊，AWS 區域 以便於在同一地區之間移動設備。AWS 帳戶如需詳細資訊，請參閱 [在多個帳戶註冊中使用多個 X.509 AWS 帳戶用戶端憑證](#)。

我們建議為每個裝置或用戶端提供唯一的憑證，藉此更精細地管理用戶端，包括憑證撤銷作業。裝置必須支援憑證的輪換和替代，以確保憑證過期時操作順暢。

如需使用 X.509 憑證支援數個裝置的詳細資訊，請參閱 [裝置佈建](#)，以檢閱 AWS IoT 支援的不同憑證管理和佈建選項。

AWS IoT 支援下列 X.509 用戶端憑證類型：

- 由所產生的 X.509 憑證 AWS IoT
- 由使 AWS IoT 用註冊的 CA 簽署的 X.509 憑證。
- 未註冊 AWS IoT 且有 CA 簽署的 X.509 憑證。

本節說明如何在 AWS IoT 中管理 X.509 憑證。您可以使用主 AWS IoT 控制台或執 AWS CLI 行下列憑證作業：

- [建立 AWS IoT 用戶端憑證](#)
- [建立您自己的用戶端憑證](#)
- [註冊用戶端憑證](#)
- [啟用或停用用戶端憑證](#)
- [撤銷用戶端憑證](#)

如需有關執行這些作業之 AWS CLI 命令的詳細資訊，請參閱 [AWS IoT CLI 參考](#)。

### 使用 X.509 用戶端憑證

X.509 憑證會驗證與的用戶端和裝置連線。AWS IoT 相較於其他識別和身分驗證機制，X.509 憑證具備多種好處。X.509 憑證可讓裝置使用非對稱金鑰。例如，您可以將私密金鑰燒錄到裝置上的安全儲存空間，這樣敏感的密碼編譯資料永遠不會離開裝置。由於私有金鑰絕對不會離開裝置，因此 X.509 憑證提供更勝於其他機制 (例如使用者名稱和密碼或承載符記) 的強大用戶端身分驗證。

AWS IoT 使用 TLS 通訊協定的用戶端驗證模式來驗證用戶端憑證。TLS 支援多種程式設計語言與作業系統，且普遍用於加密資料。在 TLS 用戶端驗證中，AWS IoT 要求 X.509 用戶端憑證，並驗證憑證的狀態和憑 AWS 帳戶 證登錄。然後，它會向用戶端提出與憑證中包含的公開金鑰對應的私密金鑰的擁有權證明。AWS IoT 要求用戶端傳送 [伺服器名稱指示 \(SNI\) 延伸](#) 至傳輸層安全性 (TLS) 通訊協定。如需設定 SNI 延伸模組的詳細資訊，請參閱 [中的運輸安全 AWS IoT Core](#)。

為了促進與 AWS IoT 核心之間的安全且一致的用戶端連線，X.509 用戶端憑證必須具備下列項目：

- 已在 AWS IoT 核心註冊。如需詳細資訊，請參閱 [註冊用戶端憑證](#)。
- 狀態為 ACTIVE。如需詳細資訊，請參閱 [啟用或停用用戶端憑證](#)。
- 尚未到達憑證過期日期。

您可以建立使用 Amazon 根 CA 的用戶端憑證，也可以使用由其他憑證授權單位 (CA) 簽署的自有用戶端憑證。如需使用 AWS IoT 主控台建立使用 Amazon 根 CA 之憑證的詳細資訊，請參閱 [建立 AWS IoT 用戶端憑證](#)。如需使用您自己的 X.509 憑證的詳細資訊，請參閱 [建立您自己的用戶端憑證](#)。

對於有憑證授權機構憑證簽署的憑證，其過期日期與時間會於憑證建立時設定。由於所產生的 X.509 憑證在二零四年十二月三十一日午夜 AWS IoT 到期 (2049-12-31T23 : 59 : 59 Z)。

AWS IoT Device Defender 可以在支援常見 IoT 安全性最佳實務的 AWS 帳戶 裝置上執行稽核。這包括管理您的 CA 或 Amazon 根 CA 簽署的 X.509 憑證的到期日。如需管理憑證到期日的詳細資訊，請參閱[裝置憑證即將到期](#)和[CA 憑證即將到期](#)。

在官方 AWS IoT 部落格上，[請參閱如何使 AWS IoT 用管理 IoT 裝置憑證輪換的方式](#)，深入瞭解裝置憑證輪換和安全性最佳實務的管理。

在多個帳戶註冊中使用多個 X.509 AWS 帳戶用戶端憑證

多帳戶註冊可讓您在相同區域或不同區域的 AWS 帳戶之間移動裝置。您可以在生產前帳戶中註冊、測試和設定裝置，然後在生產帳戶中註冊並使用相同的裝置和裝置憑證。您也可以裝置或沒有已註冊 CA 的裝置憑證上註冊用戶端憑證 AWS IoT。如需詳細資訊，請參閱[註冊未註冊 CA 所簽署的用戶端憑證 \(CLI\)](#)。

#### Note

iot:Data-ATS、iot:Data (舊式)、iot:Jobs 及 iot:CredentialProvider 端點類型上支援用於多帳戶註冊的憑證。如需有關 AWS IoT 裝置端點的詳細資訊，請參閱[AWS IoT 裝置資料和服務端點](#)。

使用多帳戶註冊的裝置必須將[伺服器名稱指示 \(SNI\) 延伸](#)傳送至傳輸層安全性 (TLS) 通訊協定，並在連線時在 host\_name 欄位中提供完整的端點位址。AWS IoT 使用中的端點位址 host\_name 將連線路由到正確的 AWS IoT 帳戶。未在 host\_name 中傳送有效端點位址的現有裝置將繼續運作，但無法使用需要此資訊的功能。如需 SNI 延伸模組的詳細資訊，以及如何識別 host\_name 欄位的端點位址，請參閱[中的運輸安全 AWS IoT Core](#)。

使用多帳戶註冊

1. 您可以在有 CA 的情況下註冊裝置憑證。您可以在 SNI\_ONLY 模式下在多個帳戶中註冊簽署 CA，並使用該 CA 將相同的用戶端憑證註冊到多個帳戶。如需詳細資訊，請參閱[在 SNI\\_ONLY 模式下註冊 CA 憑證 \(CLI\) - 建議](#)。
2. 您可以在沒有 CA 的情況下註冊裝置憑證。請參閱[註冊未註冊 CA 所簽署的用戶端憑證 \(CLI\)](#)。註冊 CA 非必須。您不需要註冊簽署裝置憑證的 CA AWS IoT。

支援的憑證簽章演算法 AWS IoT

AWS IoT 支援下列憑證簽署演算法：

- SHA256WITHRSA
- SHA384WITHRSA
- SHA512WITHRSA
- SHA256WITHRSAANDMGF1 (RSASSA-PSS)
- SHA384WITHRSAANDMGF1 (RSASSA-PSS)
- SHA512WITHRSAANDMGF1 (RSASSA-PSS)
- DSA\_WITH\_SHA256
- ECDSA-WITH-SHA256
- ECDSA-WITH-SHA384
- ECDSA-WITH-SHA512

如需有關憑證驗證和安全性的詳細資訊，請參閱[裝置憑證金鑰品質](#)。

#### Note

憑證簽署請求 (CSR) 必須包含公開金鑰。該金鑰可以是公開金鑰 (長度至少 2,048 位元) 或 ECC 金鑰 (至少 NIST P-256、NIST P-384 或 NIST P-521 曲線)。如需詳細資訊，請參閱 AWS IoT API 參考指南[CreateCertificateFromCsr](#)中的。

## 支援的關鍵演算法 AWS IoT

下表顯示了如何支持關鍵算法：

金鑰演算法	憑證簽章演算法	TLS 版本控制	支援？ 是或否
RSA 的密鑰大小至少為 2048 位	全部	英國天然安全 TLS	是
ECC 尼斯特 P-256 /P-384/ P-521	全部	英國天然安全 TLS	是
具有至少 2048 位元金鑰大小的 RSA PSS	全部	TLS 1.2	否

金鑰演算法	憑證簽章演算法	TLS 版本控制	支援？ 是或否
具有至少 2048 位元金鑰大小的 RSA PSS	全部	TLS 1.3	是

若要使用 [CreateCertificateFromCSR](#) 建立憑證，您可以使用支援的金鑰演算法來產生 CSR 的公開金鑰。若要使用 [RegisterCertificate](#) 或 [RegisterCertificateWithoutCa](#) 註冊您自己的憑證，您可以使用支援的金鑰演算法來產生憑證的公開金鑰。

如需詳細資訊，請參閱[安全性原則](#)。

## 建立 AWS IoT 用戶端憑證

AWS IoT 提供由 Amazon 根憑證授權單位 (CA) 簽署的用戶端憑證。

本主題說明如何建立由 Amazon 根憑證授權機構簽署的用戶端憑證，以及憑證檔案的下載方式。建立用戶端憑證檔案之後，您必須將它們安裝在用戶端上。

### Note

每個由提供的 X.509 用戶端憑證都會 AWS IoT 保留您在建立憑證時設定的簽發者和主旨屬性。只在建立憑證之後，憑證屬性才是不可變的。

您可以使用 AWS IoT 主控台或建 AWS CLI 立由 Amazon 根 AWS IoT 憑證授權單位簽署的憑證。

## 建立 AWS IoT 憑證 (主控台)

### 使用 AWS IoT 控制台建立 AWS IoT 憑證

1. 登入 AWS Management Console 並開啟[AWS IoT 主控台](#)。
2. 在導覽窗格中，依序選擇安全、憑證，然後選擇建立。
3. 選擇 One-click certificate creation (一鍵建立憑證) - Create certificate (建立憑證)。
4. 從已建立的憑證頁面上，將該物件的用戶端憑證檔案、公開金鑰和私密金鑰下載到安全的位置。這些由產生的憑證 AWS IoT 只能與 AWS IoT 服務搭配使用。

如果您還需要 Amazon 根 CA 憑證檔案，此頁面上也有可下載該憑證的頁面連結。

5. 現在已建立用戶端憑證，並已註冊 AWS IoT。您必須先啟動憑證，才能在用戶端中使用憑證。

若要立即啟用用戶端憑證，請選擇啟用。如果您不想立即啟用憑證，請參閱 [啟動用戶端憑證 \(主控台\)](#) 以了解如何稍後再啟用憑證。

6. 如果您要將政策連接至憑證，請選擇 Attach a policy (連接政策)。

如果您不想立即連接政策，請選擇 Done (完成) 來完成。您可以稍後附加政策。

完成程序之後，請在用戶端上安裝憑證檔案。

### 建立 AWS IoT 憑證 (CLI)

AWS CLI 提供用於建立 Amazon 根憑證授權單位簽署的用戶端憑證的 [create-keys-and-certificate](#) 命令。但是，此命令不會下載 Amazon 根憑證授權機構憑證檔案。您可以從 [伺服器身分驗證的憑證授權機構憑證](#) 下載 Amazon 根 CA 憑證檔案。

此命令會建立私密金鑰、公開金鑰和 X.509 憑證檔案，並使用註冊和啟動憑 AWS IoT 證。

```
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
```

如果您不想在建立並註冊憑證時啟動憑證，此命令會建立私有金鑰、公有金鑰和 X.509 憑證檔案並註冊憑證，但不會啟動憑證。[啟動用戶端憑證 \(CLI\)](#) 描述稍後如何啟用憑證。

```
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
```

在用戶端上安裝憑證檔案。

## 建立您自己的用戶端憑證

AWS IoT 支援由任何根或中繼憑證授權單位 (CA) 簽署的用戶端憑證。AWS IoT 使用 CA 憑證來驗證憑證的擁有權。若要使用不是 Amazon CA 的 CA 簽署的裝置憑證，必須在中註冊 CA 的憑證，以 AWS IoT 便我們可以驗證裝置憑證的擁有權。

AWS IoT 支援使用自有憑證 (BYOC) 的多種方式：

- 首先，註冊用於簽署用戶端憑證的 CA，然後註冊個別的用戶端憑證。如果您想要在裝置或用戶端第一次連線時向其用戶端憑證註冊 AWS IoT (也稱為 [Just-in-Time 佈建](#))，則必須使用註冊已簽署的 CA AWS IoT 並啟用自動註冊。
- 如果您無法註冊簽署 CA，則可以選擇不使用 CA 註冊用戶端憑證。對於不使用 CA 註冊的裝置，在將它們連接到 AWS IoT 時需要出示 [伺服器名稱指示 \(SNI\)](#)。

### Note

若要使用 CA 註冊用戶端憑證，您必須註冊已簽署的 CA AWS IoT，而不是階層中的任何其他 CA。

### Note

一個 CA 憑證只能由一個區域中的一個帳戶在 DEFAULT 模式下註冊。一個 CA 憑證可以由一個區域中的多個帳戶在 SNI\_ONLY 模式下註冊。

如需有關使用 X.509 憑證支援數個以上裝置的詳細資訊，請參閱 [裝置佈建](#)，以檢閱 AWS IoT 支援的不同憑證管理和佈建選項。

## 主題

- [管理您的憑證授權機構憑證](#)
- [使用您的憑證授權機構憑證建立用戶端憑證](#)

## 管理您的憑證授權機構憑證

本節說明管理您自己的憑證授權機構 (CA) 憑證的一般工作。



AWS IoT 如果您使用的是 AWS IoT 無法辨識的 CA 簽署的用戶端憑證，您可以註冊憑證授權單位 (CA)。

如果您希望用戶端在第一次連線 AWS IoT 時自動註冊其用戶端憑證，則必須使用簽署用戶端憑證的 CA 進行註冊 AWS IoT。否則，您不需要註冊已簽署用戶端憑證的憑證授權機構憑證。

#### Note

一個 CA 憑證只能由一個區域中的一個帳戶在 DEFAULT 模式下註冊。一個 CA 憑證可以由一個區域中的多個帳戶在 SNI\_ONLY 模式下註冊。

主題：

- [建立憑證授權機構憑證](#)
- [註冊您的憑證授權機構憑證](#)
- [停用憑證授權機構憑證](#)

### 建立憑證授權機構憑證

如果沒有 憑證授權機構憑證，您可以使用 [OpenSSL v1.1.1i](#) 工具來建立憑證授權機構憑證。

#### Note

您無法在 AWS IoT 主控台中執行此程序。

使用 [OpenSSL v1.1.1i](#) 工具來建立憑證授權機構憑證

1. 產生金鑰對。

```
openssl genrsa -out root_CA_key_filename.key 2048
```

2. 透過該金鑰對的私有金鑰來產生憑證授權機構憑證。

```
openssl req -x509 -new -nodes \  
-key root_CA_key_filename.key \  
-sha256 -days 1024 \  
-out root_CA_cert_filename.pem
```

## 註冊您的憑證授權機構憑證

這些程序說明如何從非 Amazon CA 的憑證授權單位 (CA) 註冊憑證。AWS IoT Core 使用 CA 憑證來驗證憑證的擁有權。若要使用不是 Amazon CA 的 CA 簽署的裝置憑證，您必須使用註冊 CA 憑證，以 AWS IoT Core 便驗證裝置憑證的擁有權。

註冊憑證授權機構憑證 (主控台)。

### Note

若要在主控台中註冊憑證授權機構憑證，請在主控台中的[註冊憑證授權機構憑證](#)開始。您可以在多帳戶模式下註冊您的 CA，而無需提供驗證憑證或私有金鑰的存取權。一個 CA 可由相同 AWS 區域中的多個 AWS 帳戶 以多帳戶模式進行註冊。您可透過提供驗證憑證和 CA 私有金鑰的擁有權證明，以單一帳戶模式註冊您的 CA。

## 註冊憑證授權機構憑證 (CLI)

您可以在 DEFAULT 模式或 SNI\_ONLY 模式下註冊 CA 憑證。CA 可以在 DEFAULT 模式中逐一 AWS 帳戶 註冊 AWS 區域。CA 可以在 SNI\_ONLY 模式下通過在相同 AWS 帳戶 的多個註冊 AWS 區域。如需 CA 憑證模式的詳細資訊，請參閱 [certificateMode](#)。

### Note

我們建議您在 SNI\_ONLY 模式下註冊 CA。您不需要提供驗證證書或對私鑰的訪問權限，並且可以通過多個相同 AWS 帳戶 的註冊 CA AWS 區域。

## 在 SNI\_ONLY 模式下註冊 CA 憑證 (CLI) - 建議

### 先決條件

繼續操作之前，請確定您的電腦上具有下列資訊：

- 根 CA 憑證檔案 (於下列範例中引用為 *root\_CA\_cert\_filename.pem*)
- [OpenSSL v1.1.1i](#) 或更新版本

## 若要在 **SNI\_ONLY** 模式中使用註冊 CA 憑證 AWS CLI

1. 使用註冊 CA 憑證 AWS IoT。使用 `register-ca-certificate` 命令，輸入 CA 憑證檔案名稱。如需詳細資訊，請參閱 AWS CLI 命令參考中的 [register-ca-certificate](#)。

```
aws iot register-ca-certificate \  
  --ca-certificate file://root_CA_cert_filename.pem \  
  --certificate-mode SNI_ONLY
```

若成功，此命令會傳回 *certificateId*。

2. 此時，CA 憑證已在中註冊，AWS IoT 但處於非作用中狀態。CA 憑證必須處於作用中狀態，您才能註冊已簽署的任何用戶端憑證。

這個步驟會啟動 CA 憑證。

若要啟用該 CA 憑證，請按如下所示使用 `update-certificate` 命令。如需詳細資訊，請參閱 AWS CLI 命令參考中的 [update-certificate](#)。

```
aws iot update-ca-certificate \  
  --certificate-id certificateId \  
  --new-status ACTIVE
```

若要查看 CA 憑證的狀態，請使用 `describe-ca-certificate` 命令。如需詳細資訊，請參閱 AWS CLI 命令參考中的 [describe-ca-certificate](#)。

## 在 **DEFAULT** 模式下註冊 CA 憑證 (CLI)

### 先決條件

繼續操作之前，請確定您的電腦上具有下列資訊：

- 根 CA 憑證檔案 (於下列範例中引用為 *root\_CA\_cert\_filename.pem*)
- 根 CA 憑證私有金鑰檔案 (於下列範例中引用為 *root\_CA\_key\_filename.key*)
- [OpenSSL v1.1.1i](#) 或更新版本

## 若要在DEFAULT模式中使用註冊 CA 憑證 AWS CLI

1. 要從中獲取註冊碼 AWS IoT，請使用get-registration-code。將傳回的 registrationCode 儲存為私有金鑰驗證憑證的 Common Name。如需詳細資訊，請參閱 AWS CLI 命令參考中的 [get-registration-code](#)。

```
aws iot get-registration-code
```

2. 產生私有金鑰驗證憑證的金鑰對：

```
openssl genrsa -out verification_cert_key_filename.key 2048
```

3. 建立私密金鑰驗證憑證的憑證簽署要求 (CSR)。將憑證的 Common Name 欄位設定為 get-registration-code 傳回的 registrationCode。

```
openssl req -new \  
-key verification_cert_key_filename.key \  
-out verification_cert_csr_filename.csr
```

將顯示提示要求您輸入一些此憑證的資訊，包括 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 建立私有金鑰驗證憑證：

```
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. 使用註冊 CA 憑證 AWS IoT。將 CA 憑證檔案名稱和私有金鑰驗證憑證檔案名稱傳遞給 `register-ca-certificate` 命令，如下所示。如需詳細資訊，請參閱 AWS CLI 命令參考中的 [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，此命令會傳回 *certificateId*。

6. 此時，CA 憑證已在中註冊，AWS IoT 但並非作用中。CA 憑證必須處於作用中狀態，您才能註冊已簽署的任何用戶端憑證。

這個步驟會啟動 CA 憑證。

若要啟用該 CA 憑證，請按如下所示使用 `update-certificate` 命令。如需詳細資訊，請參閱 AWS CLI 命令參考中的 [update-certificate](#)。

```
aws iot update-ca-certificate \  
  --certificate-id certificateId \  
  --new-status ACTIVE
```

若要查看 CA 憑證的狀態，請使用 `describe-ca-certificate` 命令。如需詳細資訊，請參閱 AWS CLI 命令參考中的 [describe-ca-certificate](#)。

建立憑證授權機構驗證憑證以在主控台中註冊憑證授權機構憑證

#### Note

只有當您從 AWS IoT 主控台註冊 CA 憑證時，才能使用此程序。

如果您未從 AWS IoT 主控台執行此程序，請在 [註冊 CA 憑證] 的主控台中啟動 [CA 憑證註冊](#) 程序。

繼續操作之前，請確定您在同一台電腦上具有下列可用項目：

- 根 CA 憑證檔案 (於下列範例中引用為 *root\_CA\_cert\_filename.pem*)
- 根 CA 憑證私有金鑰檔案 (於下列範例中引用為 *root\_CA\_key\_filename.key*)
- [OpenSSL v1.1.1i](#) 或更新版本

如要使用命令列介面來建立 CA 驗證憑證，以於主控台中註冊您的 CA 憑證

1. 將 *verification\_cert\_key\_filename.key* 替換為您要建立的驗證憑證金鑰檔案名稱 (例如 *verification\_cert.key*)。然後執行此命令以產生私有金鑰驗證憑證的金鑰對：

```
openssl genrsa -out verification_cert_key_filename.key 2048
```

2. 將 *verification\_cert\_key\_filename.key* 替換為您已在步驟 1 建立的金鑰檔案名稱。

*verification\_cert\_csr\_filename.csr* 替換為您要建立的憑證簽署請求 (CSR) 檔案名稱。例如：*verification\_cert.csr*。

執行此命令以建立 CSR 檔案。

```
openssl req -new \  
-key verification_cert_key_filename.key \  
-out verification_cert_csr_filename.csr
```

命令會提示您輸入其他資訊 (稍後會說明)。

3. 在 AWS IoT 主控台的「驗證憑證」容器中，複製註冊碼。
4. openssl 命令提示您輸入的資訊如下列範例所示。除了 Common Name 欄位，您可輸入自己的值或將其留空。

於 Common Name 欄位中，請貼上您在上一個步驟中複製的註冊碼。

```
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 []:

```

完成後，命令將建立 CSR 檔案。

5. *verification\_cert\_csr\_filename.csr* 替換為您在上一個步驟中使用的 *verification\_cert\_csr\_filename.csr*。

將 *root\_CA\_cert\_filename.pem* 替換為您要註冊的 CA 憑證檔案名稱。

將 *root\_CA\_key\_filename.key* 替換為 CA 憑證私有金鑰檔案的檔案名稱。

將 *verification\_cert\_filename.pem* 替換為您要建立的驗證憑證檔案名稱。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 命令完成後，您應準備好這些檔案，以供返回主控台時使用。
  - 您的憑證授權機構憑證檔案 (在上一個命令中使用的 *root\_CA\_cert\_filename.pem*)
  - 您在上一個步驟中建立的驗證憑證 (用於上一個命令中的 *verification\_cert\_filename.pem*)

## 停用憑證授權機構憑證

啟用憑證授權單位 (CA) 憑證以進行自動用戶端憑證註冊時，請 AWS IoT 檢查 CA 憑證以確定 CA 為 ACTIVE。如果 CA 憑證是 INACTIVE，則 AWS IoT 不允許註冊用戶端憑證。

藉由將 CA 憑證設定為 INACTIVE，您可防止自動註冊 CA 所核發的任何新用戶端憑證。

### Note

由有風險之憑證授權機構憑證簽署的任何已註冊裝置憑證將持續運作，直到您確實逐一撤銷這些憑證為止。

## 停用憑證授權機構憑證 (主控台)

使用 AWS IoT 主控台停用憑證授權機構憑證

1. 登入 AWS Management Console 並開啟 [AWS IoT 主控台](#)。
2. 請在左側導覽窗格中，選擇 Secure (安全)、CAs (CA)。
3. 於憑證授權單位清單中，找出您要停用的憑證授權單位，接著選擇省略號圖示開啟選項選單。
4. 在選項選單上，選擇 Deactivate (停用)。

憑證授權機構應在清單中顯示為 Inactive (非作用中)。

### Note

主 AWS IoT 控制台未提供列出您停用之 CA 所簽署之憑證的方法。如需列出這些憑證的 AWS CLI 選項，請參閱 [停用憑證授權機構憑證 \(CLI\)](#)。

## 停用憑證授權機構憑證 (CLI)

提 AWS CLI 供停用 CA 憑證的 [update-ca-certificate](#) 命令。

```
aws iot update-ca-certificate \  
  --certificate-id certificateId \  
  --new-status INACTIVE
```



使用 [list-certificates-by-ca](#) 命令，取得由指定 CA 簽署的所有已註冊裝置憑證清單。針對由指定憑證授權機構憑證所簽署的每一裝置憑證，請使用 [update-certificate](#) 命令來撤銷該裝置憑證，以避免受到使用。

使用 [describe-ca-certificate](#) 命令來查看 CA 憑證的狀態。

使用您的憑證授權機構憑證建立用戶端憑證

您可以使用自己的憑證授權機構 (CA) 來建立用戶端憑證。使用前必須 AWS IoT 先註冊用戶端憑證。如需用戶端憑證之註冊選項的詳細資訊，請參閱 [註冊用戶端憑證](#)。

建立用戶端憑證 (CLI)

**Note**

您無法在 AWS IoT 主控台中執行此程序。

使用建立用戶端憑證 AWS CLI

1. 產生金鑰對。

```
openssl genrsa -out device_cert_key_filename.key 2048
```

2. 建立用戶端憑證的 CSR。

```
openssl req -new \  
-key device_cert_key_filename.key \  
-out device_cert_csr_filename.csr
```

將出現提示，要求您輸入一些資訊，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. 從 CSR 建立用戶端憑證。

```
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
```

此時，已建立用戶端憑證，但尚未在中註冊 AWS IoT。如需如何和何時註冊用戶端憑證的詳細資訊，請參閱 [註冊用戶端憑證](#)。

#### 註冊用戶端憑證

用戶端憑證必須在中註冊，AWS IoT 才能啟用用戶端與之間的通訊 AWS IoT。您可以手動註冊每個用戶端憑證，也可以將用戶端憑證設定為在用戶端首 AWS IoT 次連線時自動註冊。

如果想要您的用戶端和裝置在第一次連線時註冊，您必須使用 [註冊您的憑證授權機構憑證](#)，在要使用用戶端憑證的區域中透過 AWS IoT 來簽署該用戶端憑證。Amazon 根 CA 會自動註冊 AWS IoT。

用戶端憑證可由 AWS 帳戶 和區域共用。這些主題中的程序必須在您要使用用戶端憑證的每個帳戶和區域中執行。在一個帳戶或區域中註冊用戶端憑證時，另一個帳戶或區域不會自動加以辨識。

#### Note

使用 Transport Layer Security (TLS) 通訊協定來連線 AWS IoT 的用戶端必須支援 TLS 的 [伺服器名稱指示 \(SNI\) 延伸](#)。如需詳細資訊，請參閱 [中的運輸安全 AWS IoT Core](#)。

## 主題

- [手動註冊用戶端憑證](#)
- [當用戶端連線到註冊 \(JITR\) 時註冊 AWS IoT just-in-time 冊用戶端憑證](#)

## 手動註冊用戶端憑證

您可以使用 AWS IoT 主控台和手動註冊用戶端憑證 AWS CLI。

使用的註冊程序取決於是否會由 AWS 帳戶 s 和 Region 共用憑證。在一個帳戶或區域中註冊用戶端憑證時，另一個帳戶或區域不會自動加以辨識。

本主題中的程序必須在您要使用用戶端憑證的每個帳戶和區域中執行。客戶端證書可以由 AWS 帳戶 s 和區域共享。

註冊已註冊 CA 所簽署的用戶端憑證 (主控台)

### Note

在執行此程序之前，請確定您擁有用戶端憑證的 .pem 檔案，而且用戶端憑證是由您[註冊](#)的 CA 簽署。AWS IoT

## AWS IoT 使用主控台註冊現有憑證

1. 登入 AWS 管理主控台並開啟主[AWS IoT 控制台](#)。
2. 請在導覽窗格的 Manage (管理) 區段中，選擇 Security (安全性)，然後選擇 Certificates (憑證)。
3. 在 Certificates (憑證) 頁面的 Certificates (憑證) 對話方塊中，選擇 Add certificate (新增憑證)，然後選擇 Register certificates (註冊憑證)。
4. 在 Register certificate (登錄憑證) 頁面的 Certificates to upload (要上傳的憑證) 對話方塊中執行下列操作：
  - 請選擇 CA is registered with AWS IoT (已用註冊 CA)。
  - 從 Choose a CA certificate (選擇 CA 憑證) 中，選取您的 Certification authority (憑證授權機構)。
    - 選擇 Register a new CA (註冊新的憑證) 以註冊未使用 AWS IoT 註冊的新 Certification authority (憑證授權機構)。
    - 若 Amazon Root certificate authority (Amazon 根憑證授權機構) 是您的憑證授權機構，Choose a CA certificate (選擇 CA 憑證) 請保留空白。

- 選擇最多 10 個憑證以上傳和註冊 AWS IoT。
- 請選擇您在 [建立 AWS IoT 用戶端憑證](#) 和 [使用您的憑證授權機構憑證建立用戶端憑證](#) 中建立的憑證檔案。
- 請選擇 Activate (啟動) 或 Deactivate (停用)。若您選擇 Deactivate (停用)，[啟用或停用用戶端憑證](#) 會說明如何在註冊憑證後啟動憑證。
- 選擇註冊。

您已註冊的憑證將會出現在 Certificates (憑證) 頁面的 Certificates (憑證) 對話方塊中。

註冊未註冊 CA 所簽署的用戶端憑證 (主控台)

#### Note

執行此程序之前，請確定您擁有用戶端憑證的 .pem 檔案。

AWS IoT 使用主控台註冊現有憑證

1. 登入 AWS 管理主控台並開啟主 [AWS IoT 控制台](#)。
2. 在左側導覽窗格中，選擇 Secure (安全)、選擇 Certificates (憑證)，然後選擇 Create (建立)。
3. 在 Create a certificate (建立憑證) 上，找出 Use my certificate (使用我的憑證) 項目，然後選擇 Get started (開始使用)。
4. 在 Select a CA (選取 CA) 上，選擇 Next (下一步)。
5. 在 Register existing device certificates (註冊現有裝置憑證) 上，選擇 Select certificates (選取憑證)，然後選取最多 10 個要註冊的憑證檔案。
6. 關閉檔案對話方塊後，請選取您要在註冊用戶端憑證時啟用，還是撤銷用戶端憑證。

如果您未在註冊憑證時啟用憑證，[啟動用戶端憑證 \(主控台\)](#) 描述稍後如何啟用憑證。

如果憑證在註冊時遭到撤銷，則無法稍後啟用。

選擇要註冊的憑證檔案，並選取註冊後要採取的動作之後，請選取 Register certificates (註冊憑證)。

已成功註冊的用戶端憑證會出現在憑證清單中。

## 註冊已註冊 CA 所簽署的用戶端憑證 (CLI)

### Note

執行此程序之前，請確定您擁有憑證授權機構 (CA) .pem 和用戶端憑證的 .pem 檔案。用戶端憑證必須由您註冊的憑證授權單位 (CA) 簽署 AWS IoT。

使用 [register-certificate](#) 命令來註冊 (但不是啟動) 用戶端憑證。

```
aws iot register-certificate \  
  --certificate-pem file://device_cert_filename.pem \  
  --ca-certificate-pem file://ca_cert_filename.pem
```

用戶端憑證已在中註冊 AWS IoT，但尚未啟動。如需稍後如何啟用用戶端憑證的詳細資訊，請參閱 [啟動用戶端憑證 \(CLI\)](#)。

當您使用這個命令註冊用戶端憑證時，也可以啟動用戶端憑證。

```
aws iot register-certificate \  
  --set-as-active \  
  --certificate-pem file://device_cert_filename.pem \  
  --ca-certificate-pem file://ca_cert_filename.pem
```

如需啟動憑證以使用於連線的詳細資訊 AWS IoT，請參閱 [啟用或停用用戶端憑證](#)

## 註冊未註冊 CA 所簽署的用戶端憑證 (CLI)

### Note

執行此程序之前，請確定您擁有憑證的 .pem 檔案。

使用 [register-certificate-without-ca](#) 命令來註冊 (但不是啟動) 用戶端憑證。

```
aws iot register-certificate-without-ca \  
  --certificate-pem file://device_cert_filename.pem
```

用戶端憑證已在中註冊 AWS IoT，但尚未啟動。如需稍後如何啟用用戶端憑證的詳細資訊，請參閱 [啟動用戶端憑證 \(CLI\)](#)。

當您使用這個命令註冊用戶端憑證時，也可以啟動用戶端憑證。

```
aws iot register-certificate-without-ca \  
  --status ACTIVE \  
  --certificate-pem file://device_cert_filename.pem
```

如需啟動憑證以使用於連線的詳細資訊 AWS IoT，請參閱[啟用或停用用戶端憑證](#)。

當用戶端連線到註冊 (JITR) 時註冊 AWS IoT just-in-time 冊用戶端憑證

您可以設定 CA 憑證，以啟用已簽署的用戶端憑證，以便在用戶端第一次連線時 AWS IoT 自動註冊 AWS IoT。

若要在用戶端第一次連線時註冊用 AWS IoT 戶端憑證，您必須啟用 CA 憑證以進行自動註冊，並由用戶端設定第一個連線以提供所需的憑證。

設定憑證授權機構憑證以支援自動註冊 (主控台)

設定 CA 憑證以支援使用 AWS IoT 主控台自動註冊用戶端憑證

1. 登入 AWS 管理主控台並開啟主[AWS IoT 控制台](#)。
2. 請在左側導覽窗格中，選擇 Secure (安全)、CAs (CA)。
3. 在憑證授權單位清單中，尋找您要啟用自動註冊的憑證授權單位，然後使用省略符號圖示開啟選項選單。
4. 在選項選單上，選擇 Enable auto-registration (啟用自動註冊)。

#### Note

自動註冊狀態不會顯示在憑證授權單位清單中。若要查看憑證授權機構的自動註冊狀態，您必須開啟憑證授權機構的 Details (詳細資料) 頁面。

設定憑證授權機構憑證以支援自動註冊 (CLI)

如果您已經使用註冊 CA 憑證 AWS IoT，請使用[update-ca-certificate](#)指令將 CA 憑證設定autoRegistrationStatus為ENABLE。

```
aws iot update-ca-certificate \  
  --certificate-id caCertificateId \  
  --status status
```

```
--new-auto-registration-status ENABLE
```

如果您想要在註冊憑證授權機構憑證時啟用 `autoRegistrationStatus`，請使用 [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](#) 命令來查看 CA 憑證的狀態。

設定用戶端的首次連線進行自動註冊

當用戶端第一次嘗試連線時，由 CA 憑證簽署的用戶端憑證必須在傳輸層安全性 (TLS) 交握期間出現在用戶端上。AWS IoT

當用戶端連線到時 AWS IoT，請使用您在 [\[建立用戶端憑證\]](#) 或 [\[建立您自己的用 AWS IoT 戶端憑證\]](#) 中建立的用戶端憑證。AWS IoT 將 CA 憑證辨識為已註冊的 CA 憑證、註冊用戶端憑證，並將其狀態設定為 `PENDING_ACTIVATION`。這代表系統會自動註冊該用戶端憑證，並準備啟用。用戶端憑證的狀態必須先是 `ACTIVE` 才能用來連線 AWS IoT。請參閱 [啟用或停用用戶端憑證](#) 以了解啟用用戶端憑證的相關資訊。

#### Note

您可以使用即 AWS IoT Core 時註冊 (JITR) 功能佈建裝置，而不必在裝置的首次連線上傳送整個信任鏈。AWS IoT Core 雖不要求顯示憑證授權機構憑證，但需要裝置在連線時傳送 [伺服器名稱指示 \(SNI\)](#) 延伸。

當 AWS IoT 自動註冊憑證或當用戶端呈現 `PENDING_ACTIVATION` 狀態的憑證時，會將訊息 AWS IoT 發佈至下列 MQTT 主題：

```
$aws/events/certificates/registered/caCertificateId
```

*caCertificateId* 為發行該用戶端憑證的憑證授權機構憑證 ID。

發佈至此主題的訊息具有以下結構：

```
{  
  "certificateId": "certificateId",
```

```
"caCertificateId": "caCertificateId",
"timestamp": timestamp,
"certificateStatus": "PENDING_ACTIVATION",
"awsAccountId": "awsAccountId",
"certificateRegistrationTimestamp": "certificateRegistrationTimestamp"
}
```

您可以建立規則以監聽此主題，並執行部分動作。我們建議您建立 Lambda 規則，此規則可驗證用戶端憑證未列於憑證撤銷清單 (CRL) 中，同時啟用憑證，並建立和連接政策至憑證。政策會決定用戶端能夠存取哪些資源。如需如何建立接聽 `$aws/events/certificates/registered/caCertificateID` 主題並執行這些動作的 [Lambda 規則的詳細資訊](#)，請參閱在 [just-in-time AWS IoT 註冊用戶端憑證](#)。

如果在用戶端憑證的自動註冊期間發生任何錯誤或例外狀況，請 AWS IoT 將事件或訊息傳送到記錄檔中的記錄 CloudWatch 檔。如需有關為帳戶設定日誌的詳細資訊，請參閱 [Amazon CloudWatch 文件](#)。

## 啟用或停用用戶端憑證

AWS IoT 驗證連線時，驗證用戶端憑證是否處於作用中狀態。

您可以建立和註冊用戶端憑證而不需啟用它們，這樣在您想要使用前，它們就不會被使用。您也可以停用作用中的用戶端憑證，以暫時停用這些憑證。最後，您可以撤銷用戶端憑證，以防止未來使用這些憑證。

## 啟動用戶端憑證 (主控台)

使用 AWS IoT 控制台啟動用戶端憑證

1. 登入 AWS 管理主控台並開啟主 [AWS IoT 控制台](#)。
2. 在左側導覽窗格中，選擇 Secure (安全)，然後選擇 Certificates (憑證)。
3. 在憑證清單中，找出您要啟用的憑證，然後使用省略符號圖示開啟選項選單。
4. 在選項選單中，選擇 Activate (啟動)。

憑證在憑證清單應顯示為 Active (作用中)。

## 停用用戶端憑證 (主控台)

使用 AWS IoT 主控台停用用戶端憑證

1. 登入 AWS 管理主控台並開啟主 [AWS IoT 控制台](#)。



2. 在左側導覽窗格中，選擇 Secure (安全)，然後選擇 Certificates (憑證)。
3. 在憑證清單中，找出您要停用的憑證，然後使用省略符號圖示開啟選項選單。
4. 在選項選單中，選擇 Deactivate (停用)。

憑證在憑證清單中應顯示為 Inactive (非作用中)。

#### 啟動用戶端憑證 (CLI)

提 AWS CLI 供啟動憑證的 [update-certificate](#) 命令。

```
aws iot update-certificate \  
  --certificate-id certificateId \  
  --new-status ACTIVE
```

如果命令成功，則憑證的狀態將會是 ACTIVE。執行 [describe-certificate](#) 以查看憑證的狀態。

```
aws iot describe-certificate \  
  --certificate-id certificateId
```

#### 停用用戶端憑證 (CLI)

提 AWS CLI 供停用憑證的 [update-certificate](#) 命令。

```
aws iot update-certificate \  
  --certificate-id certificateId \  
  --new-status INACTIVE
```

如果命令成功，則憑證的狀態將會是 INACTIVE。執行 [describe-certificate](#) 以查看憑證的狀態。

```
aws iot describe-certificate \  
  --certificate-id certificateId
```

#### 將物件或政策連接至用戶端憑證

當您建立並註冊與物 AWS IoT 件分開的憑證時，它不會有授權任何 AWS IoT 作業的任何原則，也不會與任何物件物 AWS IoT 件相關聯。本節說明如何在已註冊憑證上新增這些關聯。

**⚠ Important**

要完成這些程序，您必須先建立要連接至憑證的物件或政策。

憑證會對裝置進行驗證，AWS IoT 以便連線。將憑證連接至物件資源會在裝置 (透過憑證) 與物件資源之間建立關係。若要授權裝置執行 AWS IoT 動作 (例如允許裝置連線和發佈訊息)，必須將適當的原則附加至裝置的憑證。

將物件連接至用戶端憑證 (主控台)

您需要物件名稱才能完成此程序。

將物件連接至已註冊憑證

1. 登入 AWS 管理主控台並開啟主 [AWS IoT 控制台](#)。
2. 在左側導覽窗格中，選擇 Secure (安全)，然後選擇 Certificates (憑證)。
3. 在憑證清單中，找出要連接政策的憑證、選擇省略符號圖示來開啟憑證的選項選單，然後選擇 Attach thing (連接物件)。
4. 在彈出式視窗中，找出要連接至憑證的物件名稱、選擇其核取方塊，然後選擇 Attach (連接)。

物件現應出現在憑證詳細資訊頁面上的物件清單中。

將政策連接至用戶端憑證 (主控台)

您需要政策物件名稱才能完成此程序。

將政策物件連接至已註冊憑證

1. 登入 AWS 管理主控台並開啟主 [AWS IoT 控制台](#)。
2. 在左側導覽窗格中，選擇 Secure (安全)，然後選擇 Certificates (憑證)。
3. 在憑證清單中，找出要連接政策的憑證、選擇省略符號圖示來開啟憑證的選項選單，然後選擇 Attach policy (連接政策)。
4. 在彈出式視窗中，找出要連接至憑證的政策名稱、選擇其核取方塊，然後選擇 Attach (連接)。

政策物件現應出現在憑證詳細資訊頁面上的政策清單中。

## 將物件連接至用戶端憑證 (CLI)

提 AWS CLI 供將物件物件附加至憑證的 [attach-thing-principal](#) 指令。

```
aws iot attach-thing-principal \  
  --principal certificateArn \  
  --thing-name thingName
```

## 將政策連接至用戶端憑證 (CLI)

提 AWS CLI 供將原則物件附加至憑證的 [attach-policy](#) 命令。

```
aws iot attach-policy \  
  --target certificateArn \  
  --policy-name policyName
```

## 撤銷用戶端憑證

如果您在已註冊的用戶端憑證上偵測到可疑活動，您可以撤銷該憑證，使其無法再次使用。

### Note

憑證一旦撤銷，就無法變更其狀態。也就是說，憑證狀態無法變更為 Active 或任何其他狀態。

## 撤銷用戶端憑證 (主控台)

使用 AWS IoT 主控台撤銷用戶端憑證

1. 登入 AWS 管理主控台並開啟主 [AWS IoT 控制台](#)。
2. 在左側導覽窗格中，選擇 Secure (安全)，然後選擇 Certificates (憑證)。
3. 在憑證清單中，找出您要撤銷的憑證，然後使用省略符號圖示開啟選項選單。
4. 在選項選單中，選擇 Revoke (撤銷)。

如果憑證已成功撤銷，它會在憑證清單中顯示為 Revoked (已撤銷)。

## 撤銷用戶端憑證 (CLI)

提 AWS CLI 供撤銷憑證的 [update-certificate](#) 命令。

```
aws iot update-certificate \  
  --certificate-id certificateId \  
  --new-status REVOKED
```

如果命令成功，則憑證的狀態將會是 REVOKED。執行 [describe-certificate](#) 以查看憑證的狀態。

```
aws iot describe-certificate \  
  --certificate-id certificateId
```

將憑證傳輸至另一個帳戶

屬於某個憑證的 X.509 憑證 AWS 帳戶 可以轉移到另 AWS 帳戶一個憑證。

將 X.509 憑證從一個憑證轉移 AWS 帳戶 到另一個憑證

1. [the section called “開始傳輸憑證”](#)

在起始傳輸之前，必須先停用憑證，並將其與所有政策和物件分開。

2. [the section called “接受或拒絕憑證傳輸”](#)

接收帳戶必須明確地接受或拒絕傳輸的憑證。在接收帳戶接受憑證之後，必須先啟用憑證才能使用。

3. [the section called “取消憑證傳輸”](#)

如果憑證未被接受，原始帳戶可以取消傳輸。

開始傳輸憑證

您可以使用[AWS IoT 主控台](#)或開始將憑證傳輸到另一個 AWS 帳戶 AWS CLI。

開始傳輸憑證 (主控台)

若要完成此程序，將需要您想要傳輸之憑證的 ID。

從具有要傳輸之憑證的帳戶執行此程序。

開始將憑證傳輸到另一個 AWS 帳戶

1. 登入 AWS 管理主控台並開啟主[AWS IoT 控制台](#)。
2. 在左側導覽窗格中，選擇 Secure (安全)，然後選擇 Certificates (憑證)。

選擇您想要傳輸且狀態為 Active (作用中) 或 Inactive (非作用中) 的憑證，並開啟其詳細資訊頁面。

3. 在憑證 Details (詳細資訊) 頁面的 Actions (動作) 選單中，如果 Deactivate (停用) 選項可用，請選擇 Deactivate (停用) 選項來停用憑證。
4. 在憑證的 Details (詳細資訊) 頁面的左側選單中，選擇 Policies (政策)。
5. 在憑證的 Policies (政策) 頁面上，如果有任何政策連接至憑證，請開啟政策的選項選單並選擇 Detach (分開)，來分開每個政策。

在您繼續之前，憑證不得有任何連接的政策。

6. 在憑證的 Policies (政策) 頁面的左側選單中，選擇 Things (物件)。
7. 在憑證的 Things (物件) 頁面上，如果有任何物件連接至憑證，請開啟物件的選項選單並選擇 Detach (分開)，來分開每個物件。

在您繼續之前，憑證不得有任何連接的物件。

8. 在憑證的 Things (物件) 頁面的左側選單中，選擇 Details (詳細資訊)。
9. 在憑證的 Details (詳細資訊) 頁面的 Actions (動作) 選單中，選擇 Start transfer (開始傳輸) 以開啟 Start transfer (開始傳輸) 對話方塊。
10. 在 [開始傳輸] 對話方塊中，輸入要接收憑證的帳戶 AWS 帳戶 號碼和選擇性的簡訊。
11. 選擇 Start transfer (開始傳輸) 以傳輸憑證。

主控台應該會顯示訊息，指出傳輸成功或失敗。如果傳輸已開始，則憑證的狀態會更新為 Transferred (已傳輸)。

### 開始傳輸憑證 (CLI)

若要完成此程序，將需要您想要傳輸之憑證的 *certificateId* 和 *certificateArn*。

從具有要傳輸之憑證的帳戶執行此程序。

### 開始將憑證傳輸到另一個 AWS 帳戶

1. 使用 [update-certificate](#) 命令來停用憑證

```
aws iot update-certificate --certificate-id certificateId --new-status INACTIVE
```

2. 分開所有策略。

1. 使用 [list-attached-policies](#) 命令來列出已連接至憑證的政策。

```
aws iot list-attached-policies --target certificateArn
```

2. 對於每個連接的政策，使用 [detach-policy](#) 命令來分開政策。

```
aws iot detach-policy --target certificateArn --policy-name policy-name
```

3. 分開所有物件。

1. 使用 [list-principal-things](#) 命令來列出已連接至憑證的物件。

```
aws iot list-principal-things --principal certificateArn
```

2. 對於每個連接的物件，使用 [detach-thing-principal](#) 命令來分開物件。

```
aws iot detach-thing-principal --principal certificateArn --thing-name thing-name
```

4. 使用 [transfer-certificate](#) 命令來啟動憑證傳輸。

```
aws iot transfer-certificate --certificate-id certificateId --target-aws-account account-id
```

## 接受或拒絕憑證傳輸

您可以使用 [AWS IoT 控制台](#) 或接受或拒絕 AWS 帳戶 從另一個 AWS 帳戶 傳輸給您的憑證 AWS CLI。

### 接受或拒絕憑證傳輸 (主控台)

若要完成此程序，將需要已傳輸至您帳戶之憑證的 ID。

從接收所傳輸憑證的帳戶執行此程序。

### 接受或拒絕已傳輸至您 AWS 帳戶的憑證

1. 登入 AWS 管理主控台並開啟主 [AWS IoT 控制台](#)。
2. 在左側導覽窗格中，選擇 Secure (安全)，然後選擇 Certificates (憑證)。

選擇您想要接受或拒絕且狀態為 Pending transfer (待定傳輸) 的憑證，並開啟其詳細資訊頁面。

3. 在憑證的 Details (詳細資訊) 頁面的 Actions (動作) 選單中，

- 若要接受憑證，請選擇 Accept transfer (接受傳輸)。
- 若不要接受憑證，請選擇 Reject transfer (拒絕傳輸)。

## 接受或拒絕憑證傳輸 (CLI)

若要完成此程序，將需要您想要接受或拒絕之憑證傳輸的 *certificateId*。

從接收所傳輸憑證的帳戶執行此程序。

### 接受或拒絕已傳輸至您 AWS 帳戶的憑證

1. 使用 [accept-certificate-transfer](#) 命令來接受憑證。

```
aws iot accept-certificate-transfer --certificate-id certificateId
```

2. 使用 [reject-certificate-transfer](#) 命令來拒絕憑證。

```
aws iot reject-certificate-transfer --certificate-id certificateId
```

## 取消憑證傳輸

您可以在接受憑證傳輸之前，使用 [AWS IoT 主控台](#) 或 AWS CLI 來取消憑證傳輸。

### 取消憑證傳輸 (主控台)

若要完成此程序，將需要您想要取消之憑證傳輸的 ID。

從已起始憑證傳輸的帳戶執行此程序。

### 取消憑證傳輸

1. 登入 AWS 管理主控台並開啟主 [AWS IoT 控制台](#)。
2. 在左側導覽窗格中，選擇 Secure (安全)，然後選擇 Certificates (憑證)。選擇您想要取消其傳輸且狀態為 Transferred (已傳輸) 的憑證，並開啟其選項選單。
3. 在憑證的選項選單上，選擇 Revoke transfer (撤銷傳輸) 選項來取消憑證傳輸。

#### Important

請小心不要弄錯 Revoke transfer (撤銷傳輸) 選項與 Revoke (撤銷) 選項。

Revoke transfer (撤銷傳輸) 選項會取消憑證傳輸，而 Revoke (撤銷) 選項會使 AWS IoT 無法復原且無法使用憑證。

## 取消憑證傳輸 (CLI)

若要完成此程序，將需要您想要取消之憑證傳輸的 *certificateId*。

從已起始憑證傳輸的帳戶執行此程序。

使用 [cancel-certificate-transfer](#) 命令來取消憑證傳輸。

```
aws iot cancel-certificate-transfer --certificate-id certificateId
```

## IAM 使用者、群組和角色

IAM 使用者、群組和角色為管理 AWS 中身分及身分驗證的標準機制，您可以使用它們來連接到使用 AWS SDK 和的 AWS IoT HTTP 介面 AWS CLI。

IAM 角色也 AWS IoT 允許代表您存取帳戶中的其他 AWS 資源。例如，如果您想要讓裝置將其狀態發佈到 DynamoDB 表格，則 IAM 角色可 AWS IoT 以與 Amazon DynamoDB 互動。如需詳細資訊，請參閱 [IAM 角色](#)。

對於透過 HTTP 的訊息代理程式連線，請使用簽 AWS IoT 章第 4 版簽署程序驗證使用者、群組和角色。如需詳細資訊，請參閱 [簽署 AWS API 要求](#)。

搭配使用 AWS 簽章版本 4 時 AWS IoT，用戶端必須在其 TLS 實作中支援下列項目：

- TLS 1.2
- SHA-256 RSA 憑證簽章驗證
- TLS 密碼套件支援部分的一種密碼套件

如需相關資訊，請參閱 [的身分識別與存取管理 AWS IoT](#)。

## Amazon Cognito 身分

Amazon Cognito 身分可讓您建立臨時、有限的特權 AWS 登入資料，以便在行動應用程式和 Web 應用程式中使用。當您使用 Amazon Cognito 身分時，請建立身分集區，為您的使用者建立唯一身分，並透過身分提供者 (例如使用亞馬遜、Facebook 和 Google 登入) 進行驗證。您也可以將 Amazon



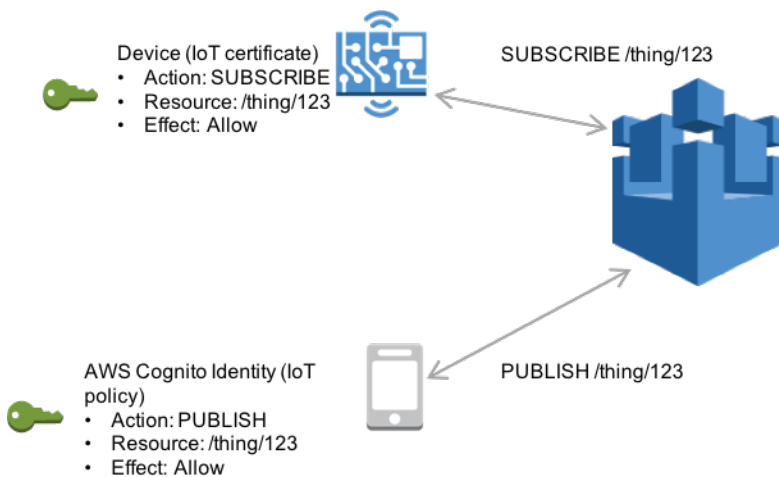
Cognito 身分識別與自己的開發人員已驗證身分搭配使用。如需詳細資訊，請參閱 [Amazon Cognito 身分](#)。

若要使用 Amazon Cognito 身分識別，請定義與 IAM 角色相關聯的 Amazon Cognito 身分識別集區。IAM 角色與 IAM 政策相關聯，該政策授予來自身分識別集區的身分權限，以存取呼叫 AWS 服務等 AWS 資源。

Amazon Cognito 身分會建立未驗證和已驗證的身分。未驗證的身分適用於行動裝置或 Web 應用程式中想使用應用程式但不想登入的訪客使用者。未驗證的使用者只會獲得與身分集區相關聯之 IAM 政策中所指定的許可。

使用已驗證身分時，除了附加至身分集區的 IAM 政策外，您還必須將 AWS IoT 政策附加到 Amazon Cognito 身分。若要附加 AWS IoT 原則，請使用 [AttachPolicy](#) API 並將權限授與應用 AWS IoT 程式的個別使用者。您可以使用此 AWS IoT 原則為特定客戶及其裝置指派精細的權限。

已驗證和未驗證的使用者是不同的身分類型。如果您未將 AWS IoT 政策附加到 Amazon Cognito 身分識別，則經過驗證的使用者會失敗授權，AWS IoT 且無法存取 AWS IoT 資源和動作。如需為 Amazon Cognito 身分建立政策的詳細資訊，請參閱 [發佈/訂閱政策範例](#) 和 [使用 Amazon Cognito 身分授權](#)。



## 自訂身分驗證和授權

AWS IoT Core 可讓您定義自訂授權者，以便您可以管理自己的用戶端驗證和授權。當您需要使用本機支 AWS IoT Core 援的驗證機制以外的驗證機制時，這很有用。(如需原生支援機制的詳細資訊，請參閱 [the section called “用戶端身分驗證”](#))。

例如，如果您要將現場中的現有裝置移轉至，AWS IoT Core 且這些裝置使用自訂承載 Token 或 MQTT 使用者名稱和密碼進行驗證，則可以將其移轉至，AWS IoT Core 而無需為其佈建新身分。您

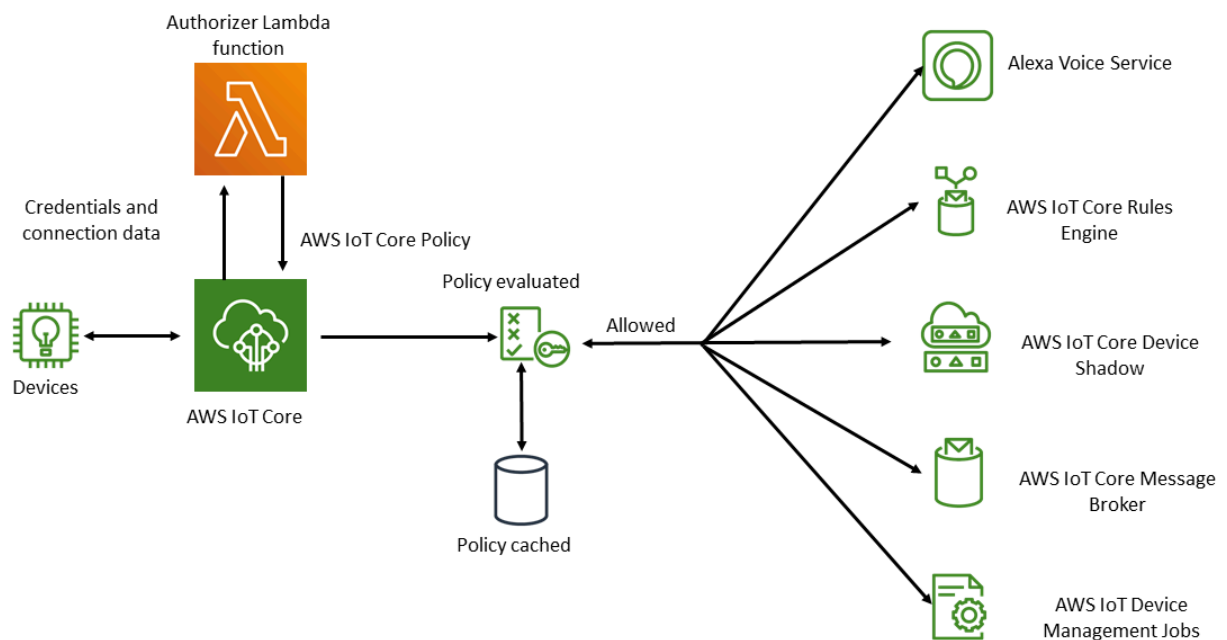
可以將自訂驗證與 AWS IoT Core 支援的任何通訊協定搭配使用。如需 AWS IoT Core 支援的通訊協定的詳細資訊，請參閱 [the section called “裝置通訊協定”](#)。

## 主題

- [了解自訂身分驗證工作流程](#)
- [建立和管理自訂授權方](#)
- [使用自訂驗證連線到 AWS IoT Core](#)
- [疑難排解您的授權方](#)

## 了解自訂身分驗證工作流程

自訂身分驗證可讓您定義如何使用[授權方資源](#)來驗證和授權用戶端。每個授權者都包含客戶管理的 Lambda 函數參考、用於驗證裝置登入資料的選用公開金鑰，以及其他組態資訊。下圖說明中自訂驗證的授權工作流程 AWS IoT Core。



## AWS IoT Core 自訂驗證和授權工作流程

下方清單說明自訂身分驗證和授權工作流程中的每個步驟。

1. 裝置會使用其中一個支援的裝置連線至客戶的 AWS IoT Core 資料端點 [the section called “裝置通訊協定”](#)。裝置會在要求的標頭欄位或查詢參數 (針對 HTTP 發佈或 MQTT 透過通訊 WebSockets 協定), 或在 MQTT CONNECT 訊息的使用者名稱和密碼欄位中傳遞認證 (針對 MQTT 和 MQTT 透過通訊協定)。 WebSockets
  2. AWS IoT Core 檢查下列兩個條件之一：
    - 傳入的請求指定授權方。
    - 接收請求的 AWS IoT Core 資料端點已為其設定預設授權者。
- 如果以上述任一方式 AWS IoT Core 找到授權者，則 AWS IoT Core 會觸發與授權者相關聯的 Lambda 函數。
3. (選擇性) 如果您已啟用權杖簽章，請先使用儲存在授權者中的公開金鑰來 AWS IoT Core 驗證請求簽章，然後再觸發 Lambda 函數。如果驗證失敗，AWS IoT Core 會停止請求，而不會叫用 Lambda 函數。
  4. Lambda 函數會接收請求中的憑證和連線中繼資料，並做出身分驗證決策。
  5. Lambda 函數會傳回驗證決 AWS IoT Core 策的結果，以及指定連線允許哪些動作的政策文件。Lambda 函數也會傳回資訊，指定呼叫 Lambda 函數來 AWS IoT Core 重新驗證要求中認證的頻率。
  6. AWS IoT Core 根據從 Lambda 函數接收到的原則評估連線上的活動。
  7. 建立連線並初始呼叫您的自訂授權者 Lambda 之後，閒置連線的下一次呼叫可能會延遲最多 5 分鐘，而無需執行任何 MQTT 作業。之後，後續的呼叫將遵循自訂授權者 Lambda 中的重新整理間隔。這種方法可以防止過度調用可能超過 AWS 帳戶

## 擴展考量

因為 Lambda 函數會為您的授權方處理身分驗證和授權，所以此函數會受制於 Lambda 定價和服務配額，例如並行執行速率。如需 Lambda 定價的詳細資訊，請參閱 [Lambda 定價](#)。您可以調整 Lambda 函數回應中的 `refreshAfterInSeconds` 和 `disconnectAfterInSeconds` 參數，管理 Lambda 函數上的負載。如需 Lambda 函數回應內容的詳細資訊，請參閱 [the section called “定義您的 Lambda 函數”](#)。

### Note

如果將簽署保持啟用狀態，您可以防止無法辨識的用戶端過度觸發 Lambda。在您的授權方中停用簽署之前，請考慮這一點。

**Note**

自訂授權方的 Lambda 函數逾時限制為 5 秒。

## 建立和管理自訂授權方

AWS IoT Core 通過使用授權[者資源](#)實現自定義身份驗證和授權方案。每個授權方都包含以下元件：

- 名稱：識別授權方的唯一使用者定義字串。
- Lambda 函數 ARN：Lambda 函數的 Amazon 資源名稱 (ARN)，用於實作授權和身分驗證邏輯。
- 字符金鑰名稱：用來從 HTTP 標頭、查詢參數或 MQTT CONNECT 使用者名稱擷取字符以執行簽章驗證的金鑰名稱。如果在您的授權方中啟用簽署，則此值為必要的。
- 已停用簽署旗標 (選用)：布林值，用來指定是否要停用憑證上的簽署需求。這對於簽署憑證沒有意義的案例很有用，例如使用 MQTT 使用者名稱和密碼的身分驗證結構述。預設值為 `false`，因此預設會啟用簽署。
- 字符簽署公有金鑰：AWS IoT Core 用來驗證字符簽章的公有金鑰。其長度下限為 2,048 位元。如果在您的授權方中啟用簽署，則此值為必要的。

Lambda 會依據 Lambda 函數執行的次數，以及在您函數中執行程式碼所需的時間，向您收費。如需 Lambda 定價的詳細資訊，請參閱 [Lambda 定價](#)。如需建立 Lambda 函數的詳細資訊，請參閱《[Lambda 開發人員指南](#)》。

**Note**

如果將簽署保持啟用狀態，您可以防止無法辨識的用戶端過度觸發 Lambda。在您的授權方中停用簽署之前，請考慮這一點。

**Note**

自訂授權方的 Lambda 函數逾時限制為 5 秒。

## 定義您的 Lambda 函數

AWS IoT Core 呼叫授權者時，它會透過包含下列 JSON 物件的事件觸發與授權者相關聯的 Lambda。範例 JSON 物件包含所有可能的欄位。不包含與連線請求無關的任何欄位。

```
{
  "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 函數應該使用此資訊來驗證傳入連線，並決定連線中允許哪些動作。此函數應該傳送包含下列值的回應。

- `isAuthenticated`：指出是否驗證請求的布林值。
- `principalId`：英數字串，作為自訂授權請求所傳送之字符的識別符。該值必須為英數字串（字元數量不低於 1 個且不超過 128 個），且符合此規則表達式 (regex) 模式：`([a-zA-Z0-9]{1,128})`。不是英數字元的特殊字元不允許與 `principalId` 搭配使用 AWS IoT Core。如果允許使用非英數字元的特殊字元，請參閱其他 AWS 服務的文件 `principalId`。

- `policyDocuments`: JSON 格式的 AWS IoT Core 政策文件清單如需有關建立原 AWS IoT Core 則的詳細資訊，請參閱 [the section called “AWS IoT Core 政策”](#) 政策文件的數目上限為 10 份政策文件。每份政策文件最多可包含 2,048 個字元。
- `disconnectAfterInSeconds`: 整數，用來指定連接至 AWS IoT Core 閘道的持續時間上限 (以秒為單位)。最小值為 300 秒，最大值為 86,400 秒。預設值為 86,400。

#### Note

的值 `disconnectAfterInSeconds` (由 Lambda 函數傳回) 會在連線建立時設定。在後續的政策重新整理 Lambda 呼叫期間，無法修改此值。

- `refreshAfterInSeconds`: 整數，用來指定政策重新整理的間隔。一旦超出此間隔，AWS IoT Core 便會叫用 Lambda 函數，以允許政策重新整理。最小值為 300 秒，最大值為 86,400 秒。

下列 JSON 物件包含您的 Lambda 函數可以傳送的回應範例。

```
{
  "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` 值必須包含有效的 AWS IoT Core 政策文件。如需有關 AWS IoT Core 策略的詳細資訊，請參閱 [the section called “AWS IoT Core 政策”](#)。在透過 TLS 和 MQTT 透過 WebSockets 連線的 MQTT 中，按欄位值中指定的間隔 AWS IoT Core 快取此原則。`refreshAfterInSeconds` 如果採用 HTTP 連線，除非裝置使用的是 HTTP 持續連線 (也稱為 HTTP 保持連線或 HTTP 連線重複使

用), 否則每個授權請求都會呼叫 Lambda 函數。您可以在設定授權方時選擇啟用快取。在此間隔期間, 會針對此快取政策 AWS IoT Core 授權已建立連線中的動作, 而不會再次觸發 Lambda 函數。如果在自訂驗證期間發生失敗, 則會 AWS IoT Core 終止連線。AWS IoT Core 如果連線的開啟時間超過 `disconnectAfterInSeconds` 參數中指定的值, 也會終止連線。

以下內容 JavaScript 包含 Node.js Lambda 函數範例, 該函數會在 MQTT Connect 訊息中尋找值為的密碼, `test` 並傳回原則, 該原則會授 AWS IoT Core 與名為的用戶端連線 `myClientName` 並發佈至包含相同用戶端名稱的主題的權限。如果找不到預期的密碼, 它會傳回拒絕這兩個動作的政策。

```
// 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'));
      break;
    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;
}

```

當上述 Lambda 函數在 MQTT Connect 訊息中收到 `test` 的預期密碼時，它會傳回下列 JSON。 `password` 和 `principalId` 屬性的值會是 MQTT 連結訊息的值。

```

{
  "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
}

```

## 建立授權方

您可以通過使用 [CreateAuthorizerAPI](#) 創建一個授權者。下列範例說明命令。

```

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` 參數，在每次叫用授權方時選擇退出簽章驗證。除非您必須停用簽署，否則強烈建議您不要這樣做。簽章驗證可保護您防範未知裝置過度叫用 Lambda 函數。在建立授權方之後，您無法更新授權方的 `signing-disabled` 狀態。若要變更此行為，您必須使用不同的 `signing-disabled` 參數值建立另一個自訂授權方。

如果您已停用簽署，`tokenKeyName` 和 `tokenSigningPublicKeys` 參數的值是選用值。如果啟用簽署，則它們是必要值。

建立 Lambda 函數和自訂授權者之後，您必須明確授與 AWS IoT Core 服務權限，以代表您叫用該函數。您可以使用以下命令執行此操作。

```

aws lambda add-permission --function-name <lambda_function_name> --
principal iot.amazonaws.com --source-arn <authorizer_arn> --statement-id
Id-123 --action "lambda:InvokeFunction"

```

## 測試您的授權方

您可以使用 [TestInvoke授權者](#) API 來測試授權者的調用和返回值。此 API 可讓您指定通訊協定中繼資料，並在授權者中測試簽名驗證。

下列索引標籤顯示如何使用 AWS CLI 來測試授權者。

### 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` 參數的值是簽署的字符。若要了解如何取得此值，請參閱 [the section called “簽署字符”](#)。

如果您的授權方取得使用者名稱和密碼，您可以使用 `--mqtt-context` 參數來傳遞此資訊。下列標籤顯示如何使用 `TestInvokeAuthorizer` API，將包含使用者名稱、密碼和用戶端名稱的 JSON 物件傳送給您的自訂授權方。

### 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"}'
```

密碼必須為 base64 編碼。下列範例顯示如何在類似 Unix 環境中編碼密碼。

```
echo -n PASSWORD | base64
```

## 管理自訂授權方

您可以使用下列 API 來管理授權方。

- [ListAuthorizers](#)：顯示您帳戶中的所有授權者。
- [DescribeAuthorizer](#)：顯示指定授權者的特性。這些值包括建立日期、上次修改日期和其他屬性。
- [SetDefault授權者](#)：指定資 AWS IoT Core 料端點的預設授權者。AWS IoT Core 如果裝置未傳遞 AWS IoT Core 認證且未指定授權者，則會使用此授權者。如需使用 AWS IoT Core 認證的詳細資訊，請參閱 [the section called “用戶端身分驗證”](#)。
- [UpdateAuthorizer](#)：變更指定授權者的狀態、權杖金鑰名稱或公開金鑰。
- [DeleteAuthorizer](#)：刪除指定的授權者。

### Note

您無法更新授權方的簽署需求。這表示您無法在需要簽署的現有授權方中停用簽署。您也無法在不需要簽署的現有授權方中要求簽署。

## 使用自訂驗證連線到 AWS IoT Core

設備可以通 AWS IoT Core 過使用自定義身份驗證與任何 AWS IoT Core 支持設備消息傳遞的協議進行連接。如需受支援通訊協定的詳細資訊，請參閱 [the section called “裝置通訊協定”](#)。傳遞至授權方 Lambda 函數的連線資料取決於您使用的通訊協定。如需建立授權方 Lambda 函數的詳細資訊，請參閱 [the section called “定義您的 Lambda 函數”](#)。以下各節說明如何使用每個支援的通訊協定來連線，以便進行身分驗證。

## HTTPS

使用 [HTTP 發佈 API](#) 傳送資料至 AWS IoT Core 的裝置可透過要求標頭或 HTTP POST 要求中的查詢參數來傳遞認證。裝置可以指定要使用 `x-amz-customauthorizer-name` 標頭或查詢參數叫用的授權方。如果您已在授權方中啟用字符簽署，則必須在請求標頭或查詢參數中傳遞 `token-key-name` 和 `x-amz-customauthorizer-signature`。請注意，在瀏覽器 JavaScript 中使用時，該 `token-signature` 值必須以 URL 編碼。

### Note

HTTPS 通訊協定的客戶授權方僅支援發佈操作。如需有關 HTTP 通訊協定的詳細資訊，請參閱 [the section called “裝置通訊協定”](#)。

下列範例請求會顯示如何在請求標頭和查詢參數中傳遞這些參數。

```
//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 連線連線至 AWS IoT Core 的裝置可以透過 MQTT 訊息的 `username` 和 `password` 欄位傳遞認證。此 `username` 值也可以選擇包含一個查詢字串，將其他值 (包括字符、簽章和授權方名稱) 傳遞給您的授權方。如果您想要使用字符型身分驗證結構描述，而不是 `username` 和 `password` 值，則可使用此查詢字串。

### Note

密碼欄位中的資料是由基 64 編碼。AWS IoT Core 您的 Lambda 函數必須將其解碼。

以下範例包含一個 `username` 字串，其中包含指定字符和簽章的額外參數。

```
username?x-amz-customauthorizer-name=authorizer-name&x-amz-customauthorizer-  
signature=token-signature&token-key-name=token-value
```

若要呼叫授權者，使用 MQTT 和自訂驗證連線至 AWS IoT Core 的裝置必須在連接埠 443 上連線。他們還必須通過應用程序層協商 ( ALPN ) TLS 擴展，其值為 `mqtt` 和服務器名稱指示 ( SNI ) 擴展與其 AWS IoT Core 數據端點的主機名稱。為了避免潛在的錯誤，`x-amz-customauthorizer-signature` 的值必須進行 URL 編碼。我們也強烈建議您對 `x-amz-customauthorizer-name` 和 `token-key-name` 值進行 URL 編碼。如需這些值的詳細資訊，請參閱 [the section called “裝置通訊協定”](#)。V2 [AWS IoT 裝置 SDK](#)、[行動 SDK](#) 和 [AWS IoT 裝置用戶端](#) 可以設定這些延伸模組。

## MQTT 超過 WebSockets

AWS IoT Core 透過使用 MQTT 透過連線的裝置 WebSockets 可以透過下列兩種方式之一傳遞認證。

- 通過 HTTP 升級請求中的請求標頭或查詢參數來建立 WebSockets 連接。
- 透過 MQTT CONNECT 訊息中的 `username` 和 `password` 欄位。

如果透過 MQTT CONNECT 訊息傳遞憑證，則需要 ALPN 和 SNI TLS 延伸。如需這些延伸的詳細資訊，請參閱 [the section called “MQTT”](#)。下列範例顯示如何透過 HTTP Upgrade 請求傳遞憑證。

```
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
```

## 簽署字符

您必須利用您在 `create-authorizer` 呼叫中所使用之公有/私有金鑰對中的私有金鑰簽署字符。下列範例說明如何使用類 Unix 命令和來建立權杖簽章。JavaScript 它們會使用 SHA-256 雜湊演算法來編碼簽章。

## 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')
```

## 疑難排解您的授權方

本主題逐步解說可能會在自訂身分驗證工作流程中產生問題的常見問題，以及解決這些問題的步驟。若要最有效地疑難排解問題，請啟用 CloudWatch 記錄檔，AWS IoT Core 並將記錄層級設定為 DEBUG。您可以在 AWS IoT Core 主控台中啟用 CloudWatch 記錄檔 (<https://console.aws.amazon.com/iot/>)。如需針對 AWS IoT Core 啟用和設定記錄的詳細資訊，請參閱 [the section called “設定 AWS IoT 記錄”](#)。

### Note

如果您長時間將記錄檔層級保留在 DEBUG，CloudWatch 可能會儲存大量的記錄資料。這可能會增加您的 CloudWatch 費用。請考慮使用資源型記錄，以增加特定物件群組中僅限裝置的詳細資訊。如需資源型記錄的詳細資訊，請參閱 [the section called “設定 AWS IoT 記錄”](#)。此外，當您完成疑難排解時，請將日誌層級降低到較不詳細的層級。

開始疑難排解之前，請檢閱 [the section called “了解自訂身分驗證工作流程”](#)，以取得自訂身分驗證程序的高階檢視。這協助您了解可在哪裡尋找問題的來源。

本主題討論下列兩個可供您調查的領域。

- 與您授權方 Lambda 函數相關的問題。
- 與您裝置相關的問題。

## 檢查您的授權方 Lambda 函數中的問題

執行下列步驟，以確定您裝置的連線嘗試正在叫用 Lambda 函數。

1. 確認哪個 Lambda 函數與您的授權方相關聯。

您可以通過調用 [DescribeAuthorizer](#) API 或在 AWS IoT Core 控制台的「安全」部分中單擊所需的授權者來執行此操作。

2. 檢查 Lambda 函數的叫用指標。執行下列步驟來執行此動作。
  - a. 打開 AWS Lambda 控制台 (<https://console.aws.amazon.com/lambda/>)，然後選擇與您的授權者相關聯的功能。
  - b. 選擇 Monitor (監控) 標籤，並檢視與您問題相關之時間範圍的指標。
3. 如果沒有看到叫用，請確認是否 AWS IoT Core 具有叫用 Lambda 函數的權限。如果您看到叫用，請跳到下一個步驟。執行下列步驟來驗證您的 Lambda 函數是否具有必要的許可。
  - a. 在 AWS Lambda 控制台中為您的功能選擇「權限」選項卡。
  - b. 在頁面底部尋找 Resource-based Policy (資源型政策) 區段。如果您的 Lambda 函數具有必要的許可，政策看起來像下列範例。

```
{
  "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. 此政策將您的功能的InvokeFunction權限授予 AWS IoT Core 主體。如果沒有看到它，則必須使用 [AddPermission](#) API 添加它。下列範例顯示如何使用 AWS CLI 來做到這一點。

```

aws lambda add-permission --function-name FunctionName --principal
iot.amazonaws.com --source-arn AuthorizerARN --statement-id Id-123 --action
"lambda:InvokeFunction"

```

4. 如果您看到叫用，請確認沒有錯誤。發生錯誤可能表示 Lambda 函數未正確處理 AWS IoT Core 傳送給它的連線事件。

如需在 Lambda 函數中處理事件的相關資訊，請參閱 [the section called “定義您的 Lambda 函數”](#)。您可以使用 AWS Lambda 控制台 (<https://console.aws.amazon.com/lambda/>) 中的測試功能對函數中的測試值進行硬編碼，以確保函數正確處理事件。

5. 如果您看到叫用沒有錯誤，但您的裝置無法連接 (或發佈、訂閱和接收訊息)，問題可能是您 Lambda 函數傳回的政策未對您裝置正在嘗試採取的動作提供許可。執行下列步驟，以判斷函數傳回的政策是否發生任何錯誤。
- a. 使用 Amazon CloudWatch 日誌洞察查詢在短時間內掃描日誌以檢查是否有故障。下列範例查詢會依時間戳記排序事件並尋找失敗。

```

display clientId, eventType, status, @timestamp | sort @timestamp desc | filter
status = "Failure"

```

- b. 更新 Lambda 函數以記錄傳回的資料以 AWS IoT Core 及觸發函數的事件。您可以使用這些記錄來檢查函數建立的政策。
6. 如果您看到叫用沒有錯誤，但您的裝置無法連接 (或發佈、訂閱和接收訊息)，另一個原因可能是您 Lambda 函數超過逾時限制。自訂授權方的 Lambda 函數逾時限制為 5 秒。您可以檢查 CloudWatch 日誌或指標中的功能持續時間。



## 調查裝置問題

如果您發現叫用 Lambda 函數沒有問題，函數傳回的政策也沒有問題，請尋找裝置嘗試連線方面的問題。格式錯誤的連接請求可能會導致 AWS IoT Core 無法觸發您的授權者。在 TLS 和應用程式層可能會同時發生連線問題。

可能的 TLS 層問題：

- 客戶必須在所有自訂驗證要求中傳遞主機名稱標頭 (HTTP、MQTT WebSockets) 或伺服器名稱指示 TLS 延伸模組 (HTTP、MQTT 超過 WebSockets、MQTT)。在這兩種情況下，傳遞的值必須與您帳戶的其中一個 AWS IoT Core 資料端點相符。這些是當您執行下列 CLI 命令時傳回的端點。
  - `aws iot describe-endpoint --endpoint-type iot:Data-ATS`
  - `aws iot describe-endpoint --endpoint-type iot:Data`(適用於舊版 VeriSign 端點)
- 使用自訂身分驗證進行 MQTT 連線的裝置也須傳遞應用程式層通訊協定交涉 (ALPN) TLS 延伸 (其值為 `mqtt`)。
- 自訂身分驗證目前僅適用於連接埠 443。

可能的應用程式層問題：

- 如果已啟用簽署 (在您的授權方中，`signingDisabled` 欄位是 `false`)，請尋找下列簽章問題。
  - 確定您是在 `x-amz-customauthorizer-signature` 標頭或在查詢字串參數中傳遞字符簽章。
  - 確定服務不是簽署字符以外的值。
  - 確定您在標頭或查詢參數中傳遞字符，而此標頭或查詢參數是您在授權方的 `token-key-name` 欄位中所指定的。
- 確定您在 `x-amz-customauthorizer-name` 標頭或查詢字串參數中傳遞的授權方名稱是有效的，或者您已為您的帳戶定義預設授權方。

## 授權

授權是授與許可給已驗證身分的程序。您可以在 AWS IoT Core 使用 AWS IoT Core 和 IAM 政策中授予許可。本主題涵蓋 AWS IoT Core 政策。如需建立 IAM 政策的詳細資訊，請參閱 [的身分識別與存取管理 AWS IoT](#) 和 [如何與 IAM AWS IoT 搭配使用](#)。

AWS IoT Core 原則會決定已驗證身分可執行的動作。裝置、行動應用程式、Web 應用程式和桌面應用程式，都會使用未驗證的身分，經過身份驗證的身份甚至可以是輸入 AWS IoT Core CLI 命令的用戶。身分識別只有在具有授與這些 AWS IoT Core 作業權限的原則時，才能執行作業。

AWS IoT Core 政策和 IAM 政策都可搭配使用 AWS IoT Core 來控制身分識別 (也稱為主體) 可執行的作業。您使用的原則類型取決於您用來進行驗證的身分識別類型 AWS IoT Core。

AWS IoT Core 操作分為兩組：

- 控制平面 API 可讓您執行管理任務，例如建立或更新憑證、物件、規則等。
- 資料平面 API 可讓您將資料傳送至資料，並從中接收資料 AWS IoT Core。

您使用的政策類型，取決於您正使用控制平面或資料平面 API。

下表說明身分類型、其使用的通訊協定以及可用於授權的政策類型。

#### AWS IoT Core 資料平面 API 和政策類型

通訊協定和身分驗證機制	SDK	身分類型	Policy type (政策類型)		
MQTT over TLS/TCP、TLS 相互授權 (連接埠 8883 或 443) <sup>†</sup> )	AWS IoT 裝置 SDK	X.509 憑證	AWS IoT Core 政策		
MQTT 透過網址/WebSocket、AWS 簽署 V4 驗證 (連接埠 443)	AWS 行動 SDK	已驗證的 Amazon Cognito 身分	IAM 和 AWS IoT Core 政策		
		未驗證的 Amazon Cognito 身分	IAM 政策		
		IAM 或聯合身分	IAM 政策		
HTTPS , AWS 簽章版本 4 驗證 (連接埠 443)	AWS CLI	Amazon Cognito、IAM 或聯合身分	IAM 政策		

通訊協定和身分驗證機制	SDK	身分類型	Policy type (政策類型)		
HTTPS、TLS 相互授權 (通訊埠 8443)	無 SDK 支援項目	X.509 憑證	AWS IoT Core 政策		
透過自訂身分驗證的 HTTPS(連接埠 443)	AWS IoT 裝置 SDK	自訂授權方	自訂授權方政策		

### AWS IoT Core 控制平面 API 和政策類型

通訊協定和身分驗證機制	SDK	身分類型	Policy type (政策類型)		
HTTPS AWS 簽章版本 4 驗證 (連接埠 443)	AWS CLI	Amazon Cognito 身分	IAM 政策		
		IAM 或聯合身分	IAM 政策		

AWS IoT Core 政策會附加至 X.509 憑證、Amazon Cognito 可身分或物群組。IAM 政策會連接至 IAM 使用者、群組或角色。如果您使用 AWS IoT 主控台或 AWS IoT Core CLI 附加政策 (至憑證、Amazon Cognito 身分或物群組)，則使用 AWS IoT Core 政策。否則，您可以使用 IAM 政策。AWS IoT Core 附加至物件群組的原則會套用至該物件群組內的任何物件。若要使 AWS IoT Core 原則生效，`clientId`與物件名稱必須相符。

政策型授權是一項強大工具。能夠讓您完全控制裝置、使用者或應用程式可在 AWS IoT Core 執行的動作。例如，假設使用憑證連線到 AWS IoT Core 的裝置。您可以允許該裝置存取所有 MQTT 主題，或者限制其存取單一主題。以另一個範例而言，假設一名使用者於命令列輸入 CLI 命令。透過使用策略，您可以允許或拒絕使用者存取任何命令或 AWS IoT Core 資源。您也可以控制應用程式存取 AWS IoT Core 資源。

由於 AWS IoT 快取政策文件的方式，對政策所做的變更可能需要幾分鐘的時間才能生效。亦即，存取最近已授與存取權的資源可能需要幾分鐘的時間，而且在撤銷資源的存取權之後，可能仍有數分鐘的時間可存取該資源。

## AWS 培訓和認證

如需中授權的相關資訊 AWS IoT Core，請參閱 AWS 訓練與認證網站上的[深入探討 AWS IoT Core 驗證與授權](#)課程。

## AWS IoT Core 政策

AWS IoT Core 原則是 JSON 文件。它們遵循與 IAM 政策相同的慣例。AWS IoT Core 支援具名原則，因此許多身分識別可以參考相同的原則文件。具名政策會經過版本控制，因此可輕鬆還原。

AWS IoT Core 原則可讓您控制對 AWS IoT Core 資料平面的存取。AWS IoT Core 資料平面向您可進行的操作，像是連線至 AWS IoT Core 訊息代理程式、傳送及接收 MQTT 訊息，以及取得或更新物件的 Device Shadow。

AWS IoT Core 策略是包含一個或多個策略陳述式的 JSON 文件。每個陳述式都包含：

- Effect，指定是否允許或拒絕該動作。
- Action，指定政策允許或拒絕的動作。
- Resource，指定資源或動作可用或不可用的資源。

由於 AWS IoT 快取原則文件的方式，對策略所做的變更可能需要 6 到 8 分鐘之間的任何時間才會生效。亦即，存取最近已授與存取權的資源可能需要幾分鐘的時間，而且在撤銷資源的存取權之後，可能仍有數分鐘的時間可存取該資源。

AWS IoT Core 政策可以附加到 X.509 憑證、Amazon Cognito 可身分識別和物件群組。附加至物件群組的政策會套用至該群組內的任何物件。若要使政策生效，clientId 與物件名稱必須相符。AWS IoT Core 政策遵循與 IAM 政策相同的政策規避邏輯。根據預設，所有的政策都會以隱含方式拒絕。任何身分型或資源型政策中的明確允許會覆寫預設行為。任何政策中的明確拒絕會覆寫任何允許。如需詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中的[政策評估邏輯](#)。

### 主題

- [AWS IoT Core 政策動作](#)
- [AWS IoT Core 動作資源](#)

- [AWS IoT Core 策略變數](#)
- [預防跨服務混淆代理人](#)
- [AWS IoT Core 政策範例](#)
- [使用 Amazon Cognito 身分授權](#)

## AWS IoT Core 政策動作

以下為 AWS IoT Core 定義的政策動作：

### MQTT 政策動作

#### iot:Connect

代表連線至 AWS IoT Core 訊息代理程式的權限。代理程式每次接收 `iot:Connect` 要求，就會檢查 `CONNECT` 許可。訊息代理程式不允許兩個相同用戶端 ID 的用戶端同時保持連線。第二個用戶端連線之後，代理程式會關閉現有的連線。使用 `iot:Connect` 許可，以確保僅使用特定用戶端 ID 的已授權用戶端可連線。

#### iot:GetRetainedMessage

代表單一保留訊息內容的取得許可。保留的郵件是以 `RETAIN` 旗標設定和儲存的發佈訊息 AWS IoT Core。如需取得所有帳戶保留訊息清單的許可，請參閱 [iot:ListRetainedMessages](#)。

#### iot:ListRetainedMessages

代表與帳戶保留訊息相關之摘要資訊的擷取許可，但不包括訊息內容。保留的郵件是以 `RETAIN` 旗標設定和儲存的發佈訊息 AWS IoT Core。針對此動作指定的資源 ARN 必須是 `*`。如需取得單一保留訊息內容的許可，請參閱 [iot:GetRetainedMessage](#)。

#### iot:Publish

代表發佈 MQTT 主題的許可。代理程式每次接收 `PUBLISH` 請求，就會檢查此許可。您可以使用此許可來允許用戶端發佈至特定主題模式。

#### Note

您必須也授予 `iot:Connect` 的許可，方能授予 `iot:Publish` 的許可。

## iot:Receive

表示接收訊息的權限 AWS IoT Core。每次在將訊息傳送至用戶端時，系統就會確認 `iot:Receive` 許可。由於每次交付都會檢查此許可，因此您可以用其來撤銷目前訂閱某主題的用戶端許可。

## iot:RetainPublish

代表發佈已設定 RETAIN 旗標之 MQTT 訊息的許可。

### Note

您必須也授予 `iot:Publish` 的許可，方能授予 `iot:RetainPublish` 的許可。

## iot:Subscribe

代表主題篩選條件的訂閱許可。代理程式每次接收 SUBSCRIBE 請求，就會檢查此許可。可以使用此許可來允許用戶端訂閱與特定主題模式相符的主題。

### Note

您必須也授予 `iot:Connect` 的許可，方能授予 `iot:Subscribe` 的許可。

## Device Shadow 政策動作

### iot>DeleteThingShadow

代表可刪除物件 Device Shadow 的許可。每次發出刪除物件 Device Shadow 內容的請求時，就會檢查一次 `iot>DeleteThingShadow` 許可。

### iot:GetThingShadow

代表可擷取物件 Device Shadow 的許可。每次發出擷取物件 Device Shadow 內容的請求時，就會檢查一次 `iot:GetThingShadow` 許可。

### iot:ListNamedShadowsForThing

代表可列出物件具名影子的許可。每次發出列出物件具名影子的請求時，就會檢查一次 `iot:ListNamedShadowsForThing` 許可。

## iot:UpdateThingShadow

代表可更新裝置影子的許可。每次發出更新物件 Device Shadow 內容的請求時，就會檢查一次 iot:UpdateThingShadow 許可。

### Note

任務執行政策動作僅適用於 HTTP TLS 端點。如果您使用 MQTT 端點，必須使用此主題上述所定義的 MQTT 政策動作。

如需示範此情況的任務執行政策範例，請參閱可與 MQTT 通訊協定搭配使用的 [the section called “基本任務政策範例”](#)。

## Job 執行 AWS IoT Core 原則動作

### iot:DescribeJobExecution

代表可擷取指定物件之任務執行的許可。每次發出取得任務執行的要求時，就會檢查一次 iot:DescribeJobExecution 許可。

### iot:GetPendingJobExecutions

代表可擷取任務 (對物件來說並非結束狀態) 之清單的許可。每次發出擷取清單的要求時，就會檢查一次 iot:GetPendingJobExecutions 許可。

### iot:UpdateJobExecution

代表可更新任務執行的許可。每次發出更新任務執行狀態的要求時，就會檢查一次 iot:UpdateJobExecution 許可。

### iot:StartNextPendingJobExecution

代表可取得並啟動物件之下一個待定任務執行的許可。(也就是將狀態為 QUEUED 的任務執行，更新為 IN\_PROGRESS。) 每次發出啟動下一個待定任務執行的要求時，就會檢查一次 iot:StartNextPendingJobExecution 許可。

## AWS IoT Core 認證提供者原則動作

### iot:AssumeRoleWithCertificate

代表呼叫 AWS IoT Core 憑證提供者以假設具有憑證型驗證的 IAM 角色的權限。每次向 AWS IoT Core 認證提供者發出要求以擔任角色時，都會檢查 iot:AssumeRoleWithCertificate 權限。

## AWS IoT Core 動作資源

若要指定 AWS IoT Core 政策動作的資源，請使用資源的 Amazon 資源名稱 (ARN)。所有資源 ARN 都遵循下列格式：

```
arn:partition:iot:region:AWS-account-ID:Resource-type/Resource-name
```

下表顯示要為每個動作類型指定的資源。ARN 範例適用於帳戶 ID 123456789012、磁碟分割中，以及該區aws域us-east-1特有的帳戶 ID。如需 ARN 格式的詳細資訊，請參閱 AWS Identity and Access Management 使用者指南中的 [Amazon 資源名稱 \(ARN\)](#)。

動作	資源類型	資源名稱	ARN 範例
iot:Connect	client	用戶端的用戶端 ID	arn:aws:iot:us-east-1:123456789012:client/myClientId
iot:DeleteThingShadow	thing	物件的名稱和影子的名稱，如果適用的話	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-east-1:123456789012:thing/thingOne/shadowOne
iot:DescribeJobExecution	thing	物件的名稱	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetPendingJobExecutions	thing	物件的名稱	arn:aws:iot:us-east-1:123456789012:thing/thingOne
iot:GetRetainedMessage	topic	保留訊息主題	arn:aws:iot:us-east-1:123456789012:topic/myTopicName
iot:GetThingShadow	thing	物件的名稱和影子的名稱，如果適用的話	arn:aws:iot:us-east-1:123456789012:thing/thingOne arn:aws:iot:us-



動作	資源類型	資源名稱	ARN 範例
<code>iot:ListNamedShadowsForThing</code>	全部	全部	<code>east-1:123456789012:thing/thingOne/shadowOne</code> *
<code>iot:ListRetainedMessages</code>	全部	全部	*
<code>iot:Publish</code>	topic	主題字串	<code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>
<code>iot:Receive</code>	topic	主題字串	<code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>
<code>iot:RetainPublish</code>	topic	發佈已設定 RETAIN 旗標的主題	<code>arn:aws:iot:us-east-1:123456789012:topic/myTopicName</code>
<code>iot:StartNextPendingJobExecution</code>	thing	物件的名稱	<code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code>
<code>iot:Subscribe</code>	topicfilter	主題篩選條件字串	<code>arn:aws:iot:us-east-1:123456789012:topicfilter/myTopicFilter</code>
<code>iot:UpdateJobExecution</code>	thing	物件的名稱	<code>arn:aws:iot:us-east-1:123456789012:thing/thingOne</code>

動作	資源類型	資源名稱	ARN 範例
iot:UpdateThingShadow	thing	物件的名稱和影子的名稱，如果適用的話	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 的角色別名	arn:aws:iot:us-east-1:123456789012:rolealias/CredentialProviderRole_alias

## AWS IoT Core 策略變數

AWS IoT Core 定義可在 Resource 或區塊中的策略中 AWS IoT Core 使用的原則變數 Condition。當政策受到評估時，實際值就會替代政策變數。例如，如果裝置連線到用戶端識別碼為 100-234-3456 的 AWS IoT Core 訊息代理程式，則原則文件中的 `iot:ClientId` 原則變數會被 100-234-3456 取代。

AWS IoT Core 政策可以使用萬用字元，並遵循與 IAM 政策類似的慣例。在字串中插入 \* (星號) 可以視為萬用字元，比對任何字元。例如，您可以使用 \* 在政策的 Resource 屬性描述多個 MQTT 主題名稱。字元 + 和 # 在政策中視為文字字串。如需示範如何使用萬用字元的政策範例，請參閱 [在 MQTT 和 AWS IoT Core 政策中使用萬用字元](#)。

您也可以使用具有固定值的預先定義政策變數，表示具有特殊含意的字元。這些特殊字元包括 `$(*)`、`$(?)` 和 `$( $ )`。如需更多關於政策變數與特殊字元的資訊，請參閱 [IAM 政策元素：變數與標籤](#) 和 [建立具有多個索引鍵或值的條件](#)。

### 主題

- [基本 AWS IoT Core 原則變數](#)
- [物件政策變數](#)
- [X.509 憑證 AWS IoT Core 原則變數](#)

### 基本 AWS IoT Core 原則變數

AWS IoT Core 定義下列基本原則變數：

- `iot:ClientId` : 用於連接至 AWS IoT Core 訊息代理程式的用戶端 ID。
- `aws:SourceIp` : 連線至 AWS IoT Core 訊息代理程式的用戶端 IP 位址。

下列 AWS IoT Core 策略顯示使用策略變數的策略。 `aws:SourceIp` 您可以在原則的「條件」元素中使用，以允許主體僅在特定位址範圍內發出 API 要求。如需範例，請參閱 [授權使用者和雲端服務使用 AWS IoT 任務](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/clientid1"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "123.45.167.89"
        }
      }
    }
  ]
}
```

在這些範例中，會 `${iot:ClientId}` 在評估原則時，由連線至 AWS IoT Core 訊息代理程式的用戶端識別碼取代。在使用諸如 `${iot:ClientId}` 的政策變數時，您可能會意外開放部分主題的存取權限。例如，若您的政策使用 `${iot:ClientId}` 來指定主題篩選條件：

```
{
```

```
"Effect": "Allow",
"Action": ["iot:Subscribe"],
"Resource": [
  "arn:aws:iot:us-east-1:123456789012:topicfilter/my/${iot:ClientId}/topic"
]
}
```

用戶端可使用 + 作為用戶端 ID 來連線，使用者即可訂閱符合主題篩選條件 my/+/topic 的任何主題。若要避免這類安全漏洞，請使用 `iot:Connect` 政策動作來控制可連接的用戶端 ID。例如，此政策僅允許那些用戶端 ID 為 `clientid1` 的用戶端連接：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["iot:Connect"],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/clientid1"
      ]
    }
  ]
}
```

#### Note

不建議搭配 Connect 使用政策變數 `${iot:ClientId}`。沒有對 `ClientId` 的值進行檢查，因此具有不同用戶端 ID 的附加工具可以通過驗證，但會導致連線中斷。因為允許任何 `ClientId`，所以設定隨機用戶端 ID 可以略過物件群組政策。

## 物件政策變數

物件原則變數可讓您根據物件名稱、物類型及物屬性值等物內容來編寫 AWS IoT Core 授與或拒絕權限的原則。您可以使用物件原則變數來套用相同的原則來控制許多 AWS IoT Core 裝置。如需裝置佈建的詳細資訊，請參閱[裝置佈建](#)。物件名稱是從物件連線時傳送的 MQTT Connect 訊息中的用戶端識別碼取得。AWS IoT Core

使用 AWS IoT Core 政策中的物件政策變數時，請注意下列事項。

- 使用[AttachThing主體](#) API 將憑證或主體 (經過驗證的 Amazon Cognito 身分) 附加至物件。

- 當您使用物件政策變數取代物件名稱時，MQTT 連線訊息或 TLS 連線中的 `clientId` 值必須完全符合物件名稱。

現已提供下列物件政策變數：

- `iot:Connection.Thing.ThingName`

這會解析為正在評估原則之 AWS IoT Core 登錄中物件的名稱。AWS IoT Core 使用裝置驗證時所顯示的憑證，以判斷要使用哪個物件來驗證連線。只有當裝置透過通訊協定透過 MQTT 或 MQTT 連線時，才能使用此原則變數。WebSocket

- `iot:Connection.Thing.ThingTypeName`

當政策受到評估時，這可以解析為所屬物件相關的物件類型。MQTT/ WebSocket 連線的用戶端識別碼必須與物件名稱相同。只有在透過通訊協定透過 MQTT 或 MQTT 連線時，才能使用此原則變數。WebSocket

- `iot:Connection.Thing.Attributes[attributeName]`

當政策受到評估時，這可以解析為所屬物件相關的特定屬性值。單一物件至多可具備 50 個屬性。每一屬性均可作為政策變數：`iot:Connection.Thing.Attributes[attributeName]`，其中 *attributeName* 為屬性名稱。MQTT/ WebSocket 連線的用戶端識別碼必須與物件名稱相同。只有在透過通訊協定透過 MQTT 或 MQTT 連線時，才能使用此原則變數。WebSocket

- `iot:Connection.Thing.IsAttached`

`iot:Connection.Thing.IsAttached: ["true"]` 強制只有在中註冊 AWS IoT 和連接到主體的設備才能存取策略內的權限。AWS IoT Core [如果裝置呈現未附加至登錄中 IoT 物件的憑證，您可以使用此變數來防止連線 AWS IoT Core。此變數具有值，true 或 false 指示連線物件已使用 AttachThing 用主體 API 在登錄中附加至登錄中的憑證或 Amazon Cognito 身分。](#) 物件名稱作為用戶端 ID。

## X.509 憑證 AWS IoT Core 原則變數

X.509 憑證原則變數可協助撰寫原 AWS IoT Core 則。這些原則會根據 X.509 憑證屬性授與權限。下列各節說明如何使用這些憑證原則變數。

**⚠ Important**

如果您的 X.509 憑證不包含特定的憑證屬性，但在原則文件中使用了對應的憑證原則變數，則原則評估可能會導致未預期的行為。

## CertificateId

在 [RegisterCertificate](#) API 中，`certificateId` 會出現在回應本文中。若要取得有關憑證的資訊，請使用 `certificateId` in [DescribeCertificate](#)。

## 發行者屬性

下列 AWS IoT Core 原則變數根據憑證簽發者設定的憑證屬性，支援允許或拒絕權限。

- `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`

## 主體屬性

根據憑證簽發者設定的憑證主體屬性，下列 AWS IoT Core 原則變數支援授與或拒絕權限。

- `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 憑證為這些屬性提供包含一或多個值的選項。根據預設，每個多值屬性的政策變數會回傳第一個值。例如，`Certificate.Subject.Country` 屬性可能包含國家/地區名稱的清單，但在政策中評估時，會將 `iot:Certificate.Subject.Country` 取代為第一個國家/地區的名稱。

您可以使用開頭為一的索引，要求第一個值以外的特定屬性值。例如，在 `iot:Certificate.Subject.Country.1` 屬性中，第二個國家/地區名稱將取代 `Certificate.Subject.Country`。若您指定的索引值不存在 (例如，僅對該屬性指派兩個值但您要求第三個值)，將不會進行替換，而授權會失敗。您可以在政策變數名稱添加 `.List` 尾碼，指定屬性全部的值。

### 發行者別名屬性

下列 AWS IoT Core 原則變數根據憑證簽發者設定的發行者替代名稱屬性，支援授與或拒絕權限。

- `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`

### 主體別名屬性

根據憑證簽發者設定的主體替代名稱屬性，下列 AWS IoT Core 原則變數支援授與或拒絕權限。

- `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`

## 其他屬性

您可以根據憑證的序號，使用 `iot:Certificate.SerialNumber` 來允許或拒絕 AWS IoT Core 資源的存取。`iot:Certificate.AvailableKeys` 政策變數包含的所有憑證政策變數名稱都具備值。

## 使用 X.509 憑證原則變數

本主題提供如何使用憑證原則變數的詳細資訊。當您建立 AWS IoT Core 根據 X.509 憑證屬性授予權限的原則時，X.509 憑證原則變數非常重要。如果您的 X.509 憑證不包含特定的憑證屬性，但在原則文件中使用了對應的憑證原則變數，則原則評估可能會導致未預期的行為。這是因為缺少的策略變量不會在策略聲明中進行評估。

在本主題中：

- [X.509 憑證範例](#)
- [使用憑證簽發者屬性作為憑證原則變數](#)
- [使用憑證主體屬性作為憑證原則變數](#)
- [使用憑證簽發者替代名稱屬性作為憑證原則變數](#)
- [使用憑證主體替代名稱屬性作為憑證原則變數](#)
- [使用其他憑證屬性作為憑證原則變數](#)
- [X.509 憑證政策變數限制](#)
- [使用憑證原則變數的範例原則](#)

## X.509 憑證範例

典型的 X.509 憑證可能顯示如下。此範例憑證包含憑證屬性。在評估原 AWS IoT Core 則期間，下列憑證屬性將填入為憑證原則變數：`Serial NumberIssuerSubject`、`X509v3 Issuer Alternative Name`、和 `X509v3 Subject Alternative Name`。

```
Certificate:  
  Data:
```



```

Version: 3 (0x2)
Serial Number:
    92:12:85:cb:b7:a5:e0:86
Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=US, O=IoT Devices, OU=SmartHome, ST=WA, CN=IoT Devices Primary CA,
    GN=Primary CA1/initials=XY/dnQualifier=Example corp,
    SN=SmartHome/ title=CA1/pseudonym=Primary_CA/generationQualifier=2/serialNumber=987

Validity
    Not Before: Mar 26 03:25:40 2024 GMT
    Not After : Apr 28 03:25:40 2025 GMT
    Subject: C=US, O=IoT Devices, OU=LightBulb, ST=NY, CN=LightBulb Device Cert,
    GN=Bulb/initials=ZZ/dnQualifier=Bulb001,
    SN=Multi Color/title=RGB/pseudonym=RGB Device/generationQualifier=4/
serialNumber=123
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public-Key: (2048 bit)
    Modulus:
        << REDACTED >>
    Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    X509v3 Key Usage:
        Digital Signature, Non Repudiation, Key Encipherment
    X509v3 Subject Alternative Name:
        DNS:example.com, IP Address:1.2.3.4, URI:ResourceIdentifier001,
        email:device1@example.com, DirName:/C=US/O=IoT/OU=SmartHome/CN=LightBulbCert
    X509v3 Issuer Alternative Name:
        DNS:issuer.com, IP Address:5.6.7.8, URI:PrimarySignerCA,
        email:primary@issuer.com, DirName:/C=US/O=Issuer/OU=IoT Devices/CN=Primary Issuer CA
    Signature Algorithm: sha256WithRSAEncryption
    << REDACTED >>

```

## 使用憑證簽發者屬性作為憑證原則變數

下表提供如何在 AWS IoT Core 原則中填入憑證簽發者屬性的詳細資訊。

要在策略中填入的發行者屬性

憑證簽發者屬性	憑證原則變數
• C = 我們	• <code>iot:Certificate.Issuer.Country = US</code>

憑證簽發者屬性	憑證原則變數
<ul style="list-style-type: none"> <li>• o = IoT 設備</li> <li>• 歐元 = SmartHome</li> <li>• ST= 瓦</li> <li>• cn= IoT 設備主要 CA</li> <li>• GN= 主要的加工 1</li> <li>• 縮寫字母 = XY</li> <li>• DNC 限定元 = 範例公司</li> <li>• SN= SmartHome</li> <li>• 標題 = 加 1</li> <li>• 假名 = 主要加拿大</li> <li>• 生成限定符 = 2</li> <li>• 序列號碼 = 987</li> </ul>	<ul style="list-style-type: none"> <li>• <code>iot:Certificate.Issuer.Organization = IoT Devices</code></li> <li>• <code>iot:Certificate.Issuer.OrganizationalUnit = SmartHome</code></li> <li>• <code>iot:Certificate.Issuer.State = WA</code></li> <li>• <code>iot:Certificate.Issuer.CommonName = IoT Devices Primary CA</code></li> <li>• <code>iot:Certificate.Issuer.GivenName = Primary CA1</code></li> <li>• <code>iot:Certificate.Issuer.initials = XY</code></li> <li>• <code>iot:Certificate.Issuer.DistinguishedNameQualifier = Example corp</code></li> <li>• <code>iot:Certificate.Issuer.Surname = SmartHome</code></li> <li>• <code>iot:Certificate.Issuer.Title = CA1</code></li> <li>• <code>iot:Certificate.Issuer.Pseudonym = Primary_CA</code></li> <li>• <code>iot:Certificate.Issuer.GenerationQualifier = 2</code></li> <li>• <code>iot:Certificate.Issuer.SerialNumber = 987</code></li> </ul>

使用憑證主體屬性作為憑證原則變數

下表提供如何在 AWS IoT Core 原則中填入憑證主體屬性的詳細資訊。

要在策略中填入的主旨屬性

憑證主旨屬性	憑證原則變數
<ul style="list-style-type: none"> <li>• C = 我們</li> <li>• o = IoT 設備</li> <li>• ST= 新</li> <li>• CN= LightBulb 裝置憑證</li> <li>• GN= 燈泡</li> </ul>	<ul style="list-style-type: none"> <li>• <code>iot:Certificate.Subject.Country = US</code></li> <li>• <code>iot:Certificate.Subject.Organization = IoT Devices</code></li> <li>• <code>iot:Certificate.Subject.State = NY</code></li> </ul>

憑證主旨屬性	憑證原則變數
<ul style="list-style-type: none"> <li>• 初始字母 = ZZ</li> <li>• DNAQ 資格 = 燈泡 001</li> <li>• SN= 多重顏色</li> <li>• 標題 = RGB</li> <li>• 假名 = RGB 設備</li> <li>• 生成限定符 = 4</li> <li>• 序列號碼 = 123</li> </ul>	<ul style="list-style-type: none"> <li>• <code>iot:Certificate.Subject.CommonName = LightBulb Device Cert</code></li> <li>• <code>iot:Certificate.Subject.GivenName = Bulb</code></li> <li>• <code>iot:Certificate.Subject.initials = ZZ</code></li> <li>• <code>iot:Certificate.Subject.DistinguishedNameQualifier = Bulb001</code></li> <li>• <code>iot:Certificate.Subject.Surname = Multi Color</code></li> <li>• <code>iot:Certificate.Subject.Title = RGB</code></li> <li>• <code>iot:Certificate.Subject.Pseudonym = RGB Device</code></li> <li>• <code>iot:Certificate.Subject.GenerationQualifier = 4</code></li> <li>• <code>iot:Certificate.Subject.SerialNumber = 123</code></li> </ul>

使用憑證簽發者替代名稱屬性作為憑證原則變數

下表提供如何在 AWS IoT Core 原則中填入憑證簽發者替代名稱屬性的詳細資訊。

要在策略中填入的發行者替代名稱屬性

X509v3 發行人替代名稱	策略中的屬性
<ul style="list-style-type: none"> <li>• DNS: 發行者</li> <li>• IP 地址:5.6.7.8</li> <li>• URI: PrimarySigner 加利福尼亞</li> <li>• 電子郵件:primary@issuer.com</li> <li>• DirName: /C = US/O = 發行者 /OU = IoT 裝置 /CN = 主要發行者 CA</li> </ul>	<ul style="list-style-type: none"> <li>• <code>iot:Certificate.Issuer.AlternativeName.DNSName = issuer.com</code></li> <li>• <code>iot:Certificate.Issuer.AlternativeName.IPAddress = 5.6.7.8</code></li> <li>• <code>iot:Certificate.Issuer.AlternativeName.UniformResourceIdentifier = PrimarySignerCA</code></li> <li>• <code>iot:Certificate.Issuer.AlternativeName.RFC822Name = primary@issuer.com</code></li> </ul>

X509v3 發行人替代名稱	策略中的屬性
	<ul style="list-style-type: none"> <li>• <code>iot:Certificate.Issuer.AlternativeName.DirectoryName = cn=Primary Issuer CA,ou=IoT Devices,o=Issuer,c=US</code></li> </ul>

使用憑證主體替代名稱屬性作為憑證原則變數

下表提供如何在 AWS IoT Core 原則中填入憑證主體替代名稱屬性的詳細資訊。

要在策略中填入的主旨替代名稱屬性

X509v3 主旨替代名稱	策略中的屬性
<ul style="list-style-type: none"> <li>• DNS: 例子</li> <li>• IP 地址 : 1.2.3.4</li> <li>• 類型 : ResourceIdentifier001</li> <li>• 電子郵件:device1@example.com</li> <li>• DirName: /C = 美國/O = IoT/歐 = /CN = 證SmartHome書LightBulb</li> </ul>	<ul style="list-style-type: none"> <li>• <code>iot:Certificate.Subject.AlternativeName.DNSName = example.com</code></li> <li>• <code>iot:Certificate.Subject.AlternativeName.IPAddress = 1.2.3.4</code></li> <li>• <code>iot:Certificate.Subject.AlternativeName.UniformResourceIdentifier = ResourceIdentifier001</code></li> <li>• <code>iot:Certificate.Subject.AlternativeName.RFC822Name = device1@example.com</code></li> <li>• <code>iot:Certificate.Subject.AlternativeName.DirectoryName = cn=LightBulbCert,ou=SmartHome,o=IoT,c=US</code></li> </ul>

使用其他憑證屬性作為憑證原則變數

下表提供 AWS IoT Core 原則中如何填入其他憑證屬性的詳細資訊。

## 要在策略中填入的其他屬性

其他憑證屬性	憑證原則變數
Serial Number: 92:12:85:cb:b7:a5: e0:86	iot:Certificate.SerialNumber = 105256223 89124227206

## X.509 憑證政策變數限制

以下限制適用於 X.509 憑證政策變數：

### 缺少政策變數

如果您的 X.509 憑證不包含特定的憑證屬性，但在原則文件中使用了對應的憑證原則變數，則原則評估可能會導致未預期的行為。這是因為缺少的策略變量不會在策略聲明中進行評估。

### 證書 SerialNumber 格式

AWS IoT Core 將憑證序號視為十進位整數的字串表示。例如，如果原則只允許具有與憑證序號相符的 Client ID 的連線，則用戶端 ID 必須是十進位格式的序號。

### 萬用字元

如果憑證屬性中有萬用字元，則原則變數不會被憑證屬性值取代。這將保留政策\${policy-variable}文件中的文本。如此可能導致授權失敗。可使用下列萬用字元：\*、\$、+、? 及 #。

### 陣列欄位

具備陣列的憑證屬性僅限五個項目，其他項目會遭到忽略。

### 字串長度

所有字串值上限為 1024 個字元。如果憑證屬性包含長度超過 1024 個字元的字串，則原則變數不會被憑證屬性值取代。這將保留\${policy-variable}在政策文件中。如此可能導致授權失敗。

### 特殊字元

在政策變數中使用時，,、"、\、+、=、<、> 及 ; 等任何特殊字元的字首必須加上反斜線 (\)。例如，Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US 會變成 Amazon Web Service O\=Amazon.com Inc. L\=Seattle ST\=Washington C\=US。

## 使用憑證原則變數的範例原則

下列原則文件允許使用符合憑證序號的用戶端 ID 進行連線，並發佈至符合模式的主\${iot:Certificate.Subject.Organization}/device-stats/\${iot:ClientId}/\*題：

### Important

如果您的 X.509 憑證不包含特定的憑證屬性，但在原則文件中使用了對應的憑證原則變數，則原則評估可能會導致未預期的行為。這是因為缺少的策略變量不會在策略聲明中進行評估。例如，如果您將下列原則文件附加至不包含`iot:Certificate.Subject.Organization`屬性的憑`iot:Certificate.Subject.Organization`證，則在原則評估期間將不會填入憑證原則變數。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/${iot:Certificate.SerialNumber}"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/${iot:Certificate.Subject.Organization}/
device-stats/${iot:ClientId}/*"
      ]
    }
  ]
}
```

您也可以使用 [Null 條件運算子](#)，確保在原則評估期間填入原則中使用的憑證原則變數。只有當「憑證序號」和「`iot:Connect`憑證主體一般名稱」屬性存在時，下列原則文件才允許使用憑證。

所有憑證原則變數都有 String 值，因此支援所有[字串條件運算子](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/*"
      ],
      "Condition": {
        "Null": {
          "iot:Certificate.SerialNumber": "false",
          "iot:Certificate.Subject.CommonName": "false"
        }
      }
    }
  ]
}
```

## 預防跨服務混淆代理人

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多權限的實體執行該動作。在中 AWS，跨服務模擬可能會導致混淆的副問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了預防這種情況，AWS 提供的工具可協助您保護所有服務的資料，而這些服務主體已獲得您帳戶中資源的存取權。

若要限制將其他服務 AWS IoT 提供給資源的權限，建議您在資源策略中使用 [aws:SourceArn](#) 和 [aws:SourceAccount](#) 全域條件內容索引鍵。如果同時使用全域條件內容索引鍵，則在相同政策陳述式中使用 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的帳戶時，必須使用相同的帳戶 ID。

防範混淆代理人問題的最有效方法是使用 `aws:SourceArn` 全域條件內容索引鍵，其中包含資源的完整 Amazon Resource Name (ARN)。對於 AWS IoT，您 `aws:SourceArn` 必須遵守以下格式：`arn:aws:iot:region:account-id:*`。請確認該 `##### AWS IoT ##` 相符，且 `## ID ### ##### ID` 相符。

下列範例顯示如何在 AWS IoT 角色信任原則中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容索引鍵，以避免混淆的副問題。

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

## AWS IoT Core 政策範例

本節中的範例政策說明在 AWS IoT Core 中完成一般任務所用的政策文件。在為解決方案建立政策時，您可以將其作為範例來開始。

本節中的範例使用這些政策元素：

- [the section called “AWS IoT Core 政策動作”](#)
- [the section called “AWS IoT Core 動作資源”](#)
- [the section called “身分型政策範例”](#)
- [the section called “基本 AWS IoT Core 原則變數”](#)
- [the section called “X.509 憑證 AWS IoT Core 原則變數”](#)

本節中的政策範例：

- [連接政策範例](#)



- [發佈/訂閱政策範例](#)
- [連線和發佈政策範例](#)
- [保留訊息政策範例](#)
- [憑證政策範例](#)
- [物件政策範例](#)
- [基本任務政策範例](#)

## 連接政策範例

下列原則會拒絕對用戶端 ID 的權限client1和client2連線 AWS IoT Core，同時允許裝置使用用戶端 ID 進行連線。用戶端識別碼符合登錄中已註 AWS IoT Core 冊並附加至用於連線之主體的物件名稱：

### Note

對於已註冊的裝置，建議您使用 Connect 動作的[物件政策變數](#)，並將物件附加至用於連線的主體。

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

```

    ],
    "Condition": {
      "Bool": {
        "iot:Connection.Thing.IsAttached": "true"
      }
    }
  }
]
}

```

下列原則授 AWS IoT Core 與使用用戶端 ID 連線的權限client1。此政策範例適用於未註冊的裝置。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/client1"
      ]
    }
  ]
}

```

### MQTT 持續工作階段政策範例

connectAttributes 可讓您在 IAM 政策中指定要在連線訊息中使用的屬性，例如 PersistentConnect 和 LastWill。如需詳細資訊，請參閱 [使用 connectAttributes](#)。

下列政策允許與 PersistentConnect 功能連接：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",

```

```
"Condition": {
  "ForAllValues:StringEquals": {
    "iot:ConnectAttributes": [
      "PersistentConnect"
    ]
  }
}
```

下列政策不允許 PersistentConnect，允許其他功能：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
      "Condition": {
        "ForAllValues:StringNotEquals": {
          "iot:ConnectAttributes": [
            "PersistentConnect"
          ]
        }
      }
    }
  ]
}
```

上述政策也可以使用 StringEquals 表達，並允許任何其他功能，包括新功能：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
```

```

    "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
  },
  {
    "Effect": "Deny",
    "Action": [
      "iot:Connect"
    ],
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "iot:ConnectAttributes": [
          "PersistentConnect"
        ]
      }
    }
  }
]
}

```

下列政策允許透過 PersistentConnect 和 LastWill 連接，但不允許任何其他新功能：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
      "Condition": {
        "ForAllValues:StringEquals": {
          "iot:ConnectAttributes": [
            "PersistentConnect",
            "LastWill"
          ]
        }
      }
    }
  ]
}

```

下列政策允許透過具有或沒有 LastWill 的用戶端進行全新連接，但不允許任何其他功能：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
      "Condition": {
        "ForAllValues:StringEquals": {
          "iot:ConnectAttributes": [
            "LastWill"
          ]
        }
      }
    }
  ]
}
```

下列政策只允許使用預設功能進行連接：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
      "Condition": {
        "ForAllValues:StringEquals": {
          "iot:ConnectAttributes": []
        }
      }
    }
  ]
}
```

下列政策僅允許與 PersistentConnect 連接，而且只要連線使用 PersistentConnect，就允許任何新功能：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "iot:ConnectAttributes": [
            "PersistentConnect"
          ]
        }
      }
    }
  ]
}
```

下列政策指出連接必須同時使用 `PersistentConnect` 和 `LastWill`，而且不允許任何新功能：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
      "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": []
    }
  }
}
]
}

```

下列政策不得具有 PersistentConnect，但可以具有 LastWill，而且不允許任何其他新功能：

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Deny",
    "Action": [
      "iot:Connect"
    ],
    "Resource": "*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "iot:ConnectAttributes": [
          "PersistentConnect"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "iot:Connect"
    ],
    "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
    "Condition": {
      "ForAllValues:StringEquals": {
        "iot:ConnectAttributes": [
          "LastWill"
        ]
      }
    }
  }
]
}

```

下列政策只允許透過具有 LastWill 與主題 "my/lastwill/topicName" 的用戶端連接，而且只要其使用 LastWill 主題，就允許任何功能：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ]
    }
  ]
}

```



```

    ],
    "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
    "Condition": {
      "ArnEquals": {
        "iot:LastWillTopic": "arn:aws:iot:region:account-id:topic/my/
lastwill/topicName"
      }
    }
  }
]
}

```

下列政策僅允許使用特定的 LastWillTopic 進行全新連接，而且只要其使用 LastWillTopic，就允許任何功能：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": "arn:aws:iot:us-east-1:123456789012:client/client1",
      "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"
          ]
        }
      }
    }
  ]
}

```

```
    }  
  }  
]  
}
```

## 發佈/訂閱政策範例

您使用的政策取決於您連接的方式 AWS IoT Core。您可以使 AWS IoT Core 用 MQTT 用戶端、HTTP 或進行連線。WebSocket 當您使用 MQTT 用戶端連接，必須使用 X.509 憑證進行驗證。當您透過 HTTP 或 WebSocket 通訊協定進行連線時，您正在使用簽名版本 4 和 Amazon Cognito 進行驗證。

### Note

對於已註冊的裝置，建議您使用 Connect 動作的 [物件政策變數](#)，並將物件附加至用於連線的主體。

在本節中：

- [在 MQTT 和 AWS IoT Core 政策中使用萬用字元](#)
- [向/從特定主題發佈、訂閱及接收訊息的政策](#)
- [向/從具有特定前綴主題發佈、訂閱及接收訊息的政策](#)
- [向/從各項裝置特定主題發佈、訂閱及接收訊息的政策](#)
- [向/從主題名稱中具有物件屬性的主題發佈、訂閱和接收訊息的政策](#)
- [拒絕向特定主題名稱的子主題進行發佈的政策](#)
- [拒絕從特定主題名稱的子主題進行接收的政策](#)
- [從使用 MQTT 萬用字元的主題進行訂閱的政策](#)
- [HTTP 和用 WebSocket 戶端的政策](#)

## 在 MQTT 和 AWS IoT Core 政策中使用萬用字元

MQTT 和 AWS IoT Core 政策具有不同的萬用字元，您應該在仔細考慮之後選擇它們。在 MQTT 中，萬用字元 + 和 # 用於 [MQTT 主題篩選器](#)，以訂閱多個主題名稱。AWS IoT Core 政策使用 \* 和 ? 做為萬用字元，並遵循 [IAM 政策](#) 的慣例。在政策文件中，\* 代表任意字元組合，? 則代表任何單一字元。在政策文件中，MQTT 萬用字元 + 和 # 視為沒有特殊含義的字元。若要在政策中的 resource 屬性描述多個主題名稱和主題篩選條件，請使用 \* 和 ? 萬用字元代替 MQTT 萬用字元。

當您選擇要在策略文件中使用的萬用字元時，請考慮該\*字元不限於單一主題層級。該+字元僅限於 MQTT 主題篩選器中的單一主題層級。要幫助將萬用字元規範限制為單一 MQTT 主題篩選條件層級，請考慮使用多個？字元。更多有關在政策資源中使用萬用字元的資訊以及它們比對的更多範例，請參閱[在資源 ARN 中使用萬用字元](#)。

下表顯示 MQTT 用戶端的 AWS IoT Core 政策中使用的不同萬用字元。

萬用字元	是 MQTT 萬用字元	MQTT 中的範例	為 AWS IoT Core 原則萬用字元	MQTT 用戶 AWS IoT Core 端政策中的範例
#	是	some/#	否	N/A
+	是	some/+/topic	否	不適用
*	否	N/A	是	topicfilter/some/*/topic topicfilter/some/sensor*/topic
?	否	N/A	是	topic/some/?????/topic topicfilter/some/sensor???/topic

### 向/從特定主題發佈、訂閱及接收訊息的政策

以下範例顯示已註冊和未註冊裝置向/從名為 "some\_specific\_topic" 的主題發佈、訂閱和接收訊息的情形。這些範例也突顯 Publish 和 Receive 使用 "topic" 作為資源，以及 Subscribe 使用 "topicfilter" 作為資源。

#### Registered devices

針對在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置與登錄中物件名稱相符的 clientId 連線。針對名為 "some\_specific\_topic" 的主題，它也提供 Publish、Subscribe 和 Receive 的許可。

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

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "iot:Connect"  
    ],  
    "Resource": [  
      "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"  
    ],  
    "Condition": {  
      "Bool": {  
        "iot:Connection.Thing.IsAttached": "true"  
      }  
    }  
  },  
  {  
    "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

對於未在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置使用用戶端 1、ClientD2 或 ClientId3 進行連線。針對名為 "some\_specific\_topic" 的主題，它也提供 Publish、Subscribe 和 Receive 的許可。

```
{
  "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/從具有特定前綴主題發佈、訂閱及接收訊息的政策

以下範例顯示已註冊和未註冊裝置向/從具有 "topic\_prefix" 前綴的主題發佈、訂閱和接收訊息的情形。

#### Note

請注意此範例中萬用字元\*的使用方式。雖然在單一陳述式中提供多個主題名稱的權限很有用，但\*是對裝置提供超過所需權限，可能會導致意想不到的後果。因此，我們建議您只\*在仔細考慮之後才使用萬用字元。

## Registered devices

針對在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置與登錄中物件名稱相符的 clientId 連線。針對以 "topic\_prefix" 為前綴的主題，它也提供 Publish、Subscribe 和 Receive 的許可。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
      ],
      "Condition": {
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
        }
      }
    }
  ]
}

```

```
},
{
  "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

對於未在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置使用用戶端 1、ClientD2 或 ClientId3 進行連線。它也為以 "topic\_prefix" 為前綴的主題提供 Publish、Subscribe 和 Receive 的許可。

```
{
  "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/從各項裝置特定主題發佈、訂閱及接收訊息的政策

以下範例顯示已註冊和未註冊裝置向/從特定裝置專屬的主題發佈、訂閱和接收訊息的情形。

### Registered devices

針對在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置與登錄中物件名稱相符的 `clientId` 連線。它提供向物件特定主題 (`sensor/device/${iot:Connection.Thing.ThingName}`) 進行發佈的許可，以及從物件特定主題 (`command/device/${iot:Connection.Thing.ThingName}`) 進行訂閱和接收的許可。如果登錄中的物件名稱是「thing1」，則裝置將能夠發佈至「感測器/裝置/物件 1」主題。該設備還可以訂閱和接收主題「命令/設備/事物 1」。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ]
    }
  ]
}

```



```
],
  "Resource": [
    "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
  ],
  "Condition": {
    "Bool": {
      "iot:Connection.Thing.IsAttached": "true"
    }
  }
},
{
  "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

對於未在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置使用用戶端 1、ClientD2 或 ClientId3 進行連線。它提供向用戶端特定主題 (sensor/device/\${iot:ClientId}) 進行發佈的許可，以及從用戶端特定主題 (command/device/\${iot:ClientId}) 進行訂閱和接收的許可。如果設備以 clientId 1 的身份連接 ClientID，它將能夠發布到主題「傳感器/設備/客戶端 1」。該設備還可以訂閱並從主題接收 device/clientId1/command。

```
{
  "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/從主題名稱中具有物件屬性的主題發佈、訂閱和接收訊息的政策

以下範例顯示已註冊裝置向/從名稱含有物件屬性的主題發佈、訂閱和接收訊息的情形。

#### Note

物件屬性僅存在於登 AWS IoT Core 錄中註冊的裝置。未註冊的裝置沒有對應的範例。

## Registered devices

針對在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置與登錄中物件名稱相符的 `clientId` 連線。它提供向主題 (`sensor/${iot:Connection.Thing.Attributes[version]}`) 進行發佈的許可，以及從主題名稱含有物件屬性的主題 (`command/${iot:Connection.Thing.Attributes[location]}`) 進行訂閱和接收的許可。如果註冊表中的事物名稱具有 `version=v1` 和 `location=Seattle`，則設備將能夠發布到主題「Sensor/V1」，並訂閱並從主題「命令/西雅圖」接收。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
      ]
    }
  ]
}

```

```
    ],
    "Condition": {
      "Bool": {
        "iot:Connection.Thing.IsAttached": "true"
      }
    }
  },
  {
    "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

因為物件屬性僅存在於登 AWS IoT Core 錄中註冊的裝置，因此沒有對應的未註冊物件範例。

## 拒絕向特定主題名稱的子主題進行發佈的政策

以下範例顯示已註冊和未註冊裝置向特定子主題以外的所有主題發佈訊息的情形。

### Registered devices

針對在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置與登錄中物件名稱相符的 `clientId` 連線。它提供向所有以 `department/` 為前綴的主題進行發佈的許可，但未及於 `department/admins` 子主題。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
      ],
      "Condition": {
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
        }
      }
    },
    {
      "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

對於未在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置使用用戶端 1、ClientD2 或 ClientId3 進行連線。它提供向所有以 "department/" 為前綴的主題進行發佈的許可，但未及於 "department/admins" 子主題。

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

```
]
}
```

## 拒絕從特定主題名稱的子主題進行接收的政策

以下範例顯示已註冊和未註冊裝置從特定子主題以外所有具特定前綴主題訂閱和接收訊息的情形。

### Registered devices

針對在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置與登錄中物件名稱相符的 `clientId` 連線。此政策允許裝置訂閱任何前綴為 `topic_prefix` 的主題。藉由在 `iot:Receive` 的陳述式中使用 `NotResource` 我們允許裝置接收所有來自其已訂閱主題的訊息，但前綴為 `topic_prefix/restricted` 的主題除外。例如在此政策中，裝置可以訂閱 `topic_prefix/topic1` 甚至 `topic_prefix/restricted`，但是它們只會接收來自 `topic_prefix/topic1` 主題的訊息，且不會接收來自 `topic_prefix/restricted` 主題的訊息。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
      ],
      "Condition": {
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
        }
      }
    },
    {
      "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

對於未在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置使用用戶端 1、ClientD2 或 ClientId3 進行連線。此政策允許裝置訂閱任何前綴為 "topic\_prefix" 的主題。藉由在 `iot:Receive` 的陳述式中使用 `NotResource`，我們允許裝置接收所有來自其已訂閱主題的訊息，但前綴為 "topic\_prefix/restricted" 的主題除外。例如，使用此策略，設備可以訂閱「topic\_首碼/主題1」，甚至「主題\_首碼/受限制」。但是，他們只會接收來自主題「主題/主題 1」的消息，並且不會收到來自「主題\_首碼/受限制」主題的消息。

```

{
  "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/*"
    }
  ]
}

```



```
}
```

## 從使用 MQTT 萬用字元的主題進行訂閱的政策

MQTT 萬用字元 + 和 # 會被視為常值字串，但在原則中使用時不會被視為萬用字元。AWS IoT Core 在 MQTT 中，僅在訂閱主題篩選條件時才會將 + 和 # 視為萬用字元，但在所有其他情況下視為常值字串。我們建議您在仔細考量之後，僅使用這些 MQTT 萬用字元作為 AWS IoT Core 政策的一部分。

以下顯示在原則中使用 MQTT 萬用字元的已註冊和未註冊物件的範例。AWS IoT Core 這些萬用字元會視為常值字串。

### Registered devices

針對在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置與登錄中物件名稱相符的 clientId 連線。此政策允許裝置訂閱 "department+/employees" 和 "location/#" 主題。由於 + 和 # 會被視為原則中的 AWS IoT Core 文字字串，因此裝置可以訂閱主題「部門 /+/員工」，但不能訂閱「部門/工程/員工」主題。同樣，裝置可以訂閱 "location/#" 主題，但不能訂閱 "location/Seattle" 主題。但是，一旦裝置訂閱了 "department+/employees" 主題，此政策將允許其接收來自 "department/engineering/employees" 主題的訊息。同樣，一旦裝置訂閱了 "location/#" 主題，其也將接收來自 "location/Seattle" 主題的訊息。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
      ],
      "Condition": {
        "Bool": {
          "iot:Connection.Thing.IsAttached": "true"
        }
      }
    },
    {
      "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

對於未在登 AWS IoT Core 錄中註冊的裝置，下列原則允許裝置使用用戶端 1、ClientId2 或 ClientId3 進行連線。此政策允許裝置訂閱 "department+/employees" 和 "location/#" 主題。由於 + 和 # 會被視為原則中的 AWS IoT Core 文字字串，因此裝置可以訂閱主題「部門 /+/員工」，但不能訂閱「部門/工程/員工」主題。同樣，裝置可以訂閱 "location/#" 主題，但不能訂閱 "location/Seattle"。但是，一旦裝置訂閱了 "department+/employees" 主題，此政策將允許其接收來自 "department/engineering/employees" 主題的訊息。同樣，一旦裝置訂閱了 "location/#" 主題，其也將接收來自 "location/Seattle" 主題的訊息。

```

{
  "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/*"
  }
]
}
```

## HTTP 和用 WebSocket 戶端的政策

當您透過 HTTP 或 WebSocket 通訊協定進行連線時，您正在使用簽名版本 4 和 Amazon Cognito 進行驗證。Amazon Cognito 身分可以是已驗證或未驗證的身分。已驗證的身分屬於已由任何支援的身分提供者驗證的使用者。未驗證的身分通常屬於未向身分提供者驗證身分的訪客使用者。Amazon Cognito 提供唯一的識別碼和 AWS 登入資料，以支援未經驗證的身分。如需詳細資訊，請參閱 [the section called “使用 Amazon Cognito 身分授權”](#)。

對於下列操作，請 AWS IoT Core 使用透過 AttachPolicy API 連接至 Amazon Cognito 身分識別的 AWS IoT Core 政策。這會將附加至 Amazon Cognito 身分集區的許可範圍降低使用已驗證身分的範圍。

- `iot:Connect`
- `iot:Publish`
- `iot:Subscribe`
- `iot:Receive`
- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot>DeleteThingShadow`

這表示 Amazon Cognito 身分需要 IAM 角色政策和政策的 AWS IoT Core 許可。您可以透過 AWS IoT Core AttachPolicy API 將 IAM 角色政策附加到集區，並將 AWS IoT Core 政策附加到 Amazon Cognito 身分識別。

已驗證和未驗證的使用者是不同的身分類型。如果您未將 AWS IoT 政策附加到 Amazon Cognito 身分識別，則經過驗證的使用者會失敗授權，AWS IoT 且無法存取 AWS IoT 資源和動作。

### Note

對於其他 AWS IoT Core 操作或未驗證身分，AWS IoT Core 不會縮小連接到 Amazon Cognito 身分集區角色的許可範圍。對於已驗證和未驗證兩者的身分而言，這是我們針對連接至 Amazon Cognito 集區角色的政策，所提供的最寬容建議。

## HTTP

若要允許未驗證 Amazon Cognito 身分透過 HTTP 對 Amazon Cognito 身分特定的主題發佈訊息，請將下列 IAM 政策連接至 Amazon Cognito 身分集區角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
      ],
      "Resource": ["arn:aws:iot:us-east-1:123456789012:topic/${cognito-identity.amazonaws.com:sub}"]
    }
  ]
}
```

若要允許經過驗證的使用者，請使用 API 將上述政策附加到 Amazon Cognito 身分集區角色和 Amazon Cognito 身分識別。AWS IoT Core [AttachPolicy](#)

**Note**

授權 Amazon Cognito 身分時，請同時 AWS IoT Core 考慮這兩個政策，並授予指定的最低權限。只有在兩個政策都允許要求的動作時，才會允許執行動作。如果兩個政策都禁止動作，則該動作為未授權。

## MQTT

若要允許未經驗證的 Amazon Cognito 身分針對您帳戶中的 Amazon Cognito 身分特定主題發佈 MQTT 訊息，請將以下 IAM 政策附加到 Amazon Cognito 身分集區角色：WebSocket

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

若要允許經過驗證的使用者，請使用 API 將上述政策附加到 Amazon Cognito 身分集區角色和 Amazon Cognito 身分識別。AWS IoT Core [AttachPolicy](#)

**Note**

授權 Amazon Cognito 身分時，請 AWS IoT Core 考慮兩者並授與指定的最低權限。只有在兩個政策都允許要求的動作時，才會允許執行動作。如果兩個政策都禁止動作，則該動作為未授權。

**連線和發佈政策範例**

對於在登 AWS IoT Core 錄中註冊為物件的裝置，下列原則會授 AWS IoT Core 與使用符合物名稱的用戶端 ID 連線的權限，並限制裝置在 Client ID 或物名特定 MQTT 主題上發佈。若要使連線成功，物件名稱必須在登 AWS IoT Core 錄中註冊，並使用附加至物件的識別或主參與者進行驗證：

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

對於未在登 AWS IoT Core 錄中註冊為物件的裝置，下列原則會授予 AWS IoT Core 與用戶端 ID 連線的權限，client1 並限制裝置在用戶端特定 MQTT 主題上發佈：

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

## 保留訊息政策範例

使用[保留訊息](#)需要特定的政策。保留的郵件是使用 RETAIN 旗標設定和儲存的發佈 MQTT 訊息。AWS IoT Core 本節提供允許保留訊息常見用途的政策範例。

在本節中：

- [連線和發佈保留訊息的政策](#)
- [連線和發佈保留 Will 訊息的政策](#)
- [列出和取得保留訊息的政策](#)

## 連線和發佈保留訊息的政策

對於發佈保留訊息的裝置，裝置必須能夠連線和發佈任何 MQTT 訊息，也能發佈 MQTT 保留訊息。下列政策會授予以下主題的許可：device/sample/configuration 至用戶端 **device1**。如需授予連線許可的其他範例，請參閱 [the section called “連線和發佈政策範例”](#)。

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

## 連線和發佈保留 Will 訊息的政策

用戶端可以設定在用戶端意外中斷連線時發佈的訊息。AWS IoT Core MQTT 會將這類訊息稱為 [Will 訊息](#)。用戶端必須將擁有的其他條件新增至其連線許可，才能包含這些條件。

下列政策文件會授予所有用戶端連線和發佈 Will 訊息的許可；此類訊息由其主題 (will) 識別，AWS IoT Core 也將保留。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/device1"
      ],
      "Condition": {
        "ForAllValues:StringEquals": {
          "iot:ConnectAttributes": [
            "LastWill"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:RetainPublish"
      ],
      "Resource": [
```



```

    "arn:aws:iot:us-east-1:123456789012:topic/will"
  ]
}
]
}

```

## 列出和取得保留訊息的政策

服務和應用程式可以透過呼叫 [ListRetainedMessages](#) 和 [GetRetainedMessage](#) 來存取保留訊息，不需支援 MQTT 用戶端。必須透過使用如下範例的政策來對呼叫這些動作的服務和應用程式進行授權。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:ListRetainedMessages"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/device1"
      ],
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:GetRetainedMessage"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:topic/foo"
      ]
    }
  ]
}

```

## 憑證政策範例

對於在登 AWS IoT Core 錄中註冊的裝置，下列原則會授 AWS IoT Core 與使用符合物名稱的用戶端 ID 連線至，以及發佈至名稱等於用於驗證本身certificateId之裝置之憑證的主題的權限：

```

{
  "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}"]
      }
    ]
  }

```

對於未在登 AWS IoT Core 錄中註冊的裝置，下列原則會授 AWS IoT Core 與連線到用戶端 ID、client1client2、client3和發佈至主題的權限，其名稱與用於驗證本身certificateId之裝置的憑證名稱相同的主題：

```

{
  "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"
    ]
}
]
}

```

對於在登 AWS IoT Core 錄中註冊的裝置，下列原則會授 AWS IoT Core 與使用符合物名稱的用戶端 ID 連線到的權限，以及發佈至名稱等於用於驗證本身之裝置之憑證的主旨 CommonName 欄位的主題：

```

{
  "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

在這個範例中，憑證的主體通用名稱欄位會用作為主題識別符，並假設主體通用名稱對每個登錄憑證是唯一的。如果憑證在多個裝置間共用，所有共用此憑證之裝置的主體通用名稱都是相同的，因此允許從多個裝置對相同主題的發佈權限 (不建議)。

對於未在登 AWS IoT Core 錄中註冊的裝置，下列原則會授 AWS IoT Core 與連線到用戶端 ID client1 client2、client3 和發佈至名稱等於用於驗證本身之裝置之憑證主旨 CommonName 欄位的主題的權限：

```

{
  "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

在這個範例中，憑證的主體通用名稱欄位會用作為主題識別符，並假設主體通用名稱對每個登錄憑證是唯一的。如果憑證在多個裝置間共用，所有共用此憑證之裝置的主體通用名稱都是相同的，因此允許從多個裝置對相同主題的發佈權限 (不建議)。

對於在登 AWS IoT Core 錄中註冊的裝置，下列原則會授 AWS IoT Core 與使用符合物名稱的用戶端 ID 連線到的權限，並在用於驗證裝置的憑證的 Subject.CommonName.2 欄位設定為 admin/時，發佈至名稱前綴為主題的主題：Administrator

```

{
  "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"
      }
    }
  }
]
}

```

對於未在登 AWS IoT Core 錄中註冊的裝置，當用於驗證裝置 AWS IoT Core 的憑證設定為admin/時client1client2，下列策略會授與與用戶端 ID 連線，以client3及發佈至名稱前綴為主題的Subject.CommonName.2權限：Administrator

```

{
  "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"
      }
    }
  }
]
}

```

對於在登 AWS IoT Core 錄中註冊的裝置，下列原則允許設備使用其物件名稱在特定主題上發佈，該主題包括用於驗證裝置的憑證的任何一個Subject.CommonName欄位設定為ThingName時Administrator : admin/

```

{
  "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"
        }
      }
    }
  ]
}

```

對於未在登 AWS IoT Core 錄中註冊的裝置，當用於驗證裝置的憑證的任何一個Subject.CommonName欄位設定為admin時 client1client2，下列策略會授 AWS IoT Core 與用戶端 ID 連線、client3和發佈至主題的權限Administrator：

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

## 物件政策範例

如果用來進行驗證的憑證附加至要評估原則的 AWS IoT Core 物件，下列原則會允許裝置連線：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["iot:Connect"],
```

```

    "Resource": [ "*" ],
    "Condition": {
      "Bool": {
        "iot:Connection.Thing.IsAttached": ["true"]
      }
    }
  }
]
}

```

下列政策在憑證連接至具有特定物件類型的物件，且該物件的 `attributeName` 屬性具有 `attributeValue` 值時，允許裝置發佈。如需物件政策變數的詳細資訊，請參閱[物件政策變數](#)。

```

{
  "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"
        }
      }
    }
  ]
}

```

下列政策允許裝置發佈至以物件屬性開頭的主題。如果裝置憑證與物件沒有關聯，則無法解析此變數，並會導致存取遭拒錯誤。如需物件政策變數的詳細資訊，請參閱[物件政策變數](#)。

```

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

```



```

    "Effect": "Allow",
    "Action": [
      "iot:Publish"
    ],
    "Resource": "arn:aws:iot:us-east-1:123456789012:topic/
    ${iot:Connection.Thing.Attributes[attributeName]}/*"
  }
]
}

```

## 基本任務政策範例

此範例展示任務目標所需的政策陳述式，該目標為單一裝置，可接收任務請求並與 AWS IoT 溝通任務執行狀態。

將 *us-west-2:57 EXAMPLE833* 替換為冒 AWS 區域號字符 (:) 和您的 12 位 AWS 帳戶數字，然後 ThingName 用代表中設備的物件資源的名稱替換 # 一的。AWS IoT

```

{
  "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 身分授權

有兩種類型的 Amazon Cognito 身分：未驗證和已驗證。如果您的應用程式支援未驗證 Amazon Cognito 身分，則不會執行身分驗證，因此您不知道使用者是誰。

**Unauthenticated Identities**：(未驗證的身分) 對於未驗證的 Amazon Cognito 身分，您可以將 IAM 角色連接至未驗證的身分集區來授予許可。建議只授予您想要提供給未知使用者使用之資源的存取權。

**⚠ Important**

對於連線到的未經驗證的 Amazon Cognito 使用者 AWS IoT Core，建議您授予 IAM 政策中非常有限資源的存取權。

Authenticated Identities (已驗證的身分)：對於已驗證的 Amazon Cognito 身分，您需要在兩個位置指定許可：

- 將 IAM 政策連接到已驗證的 Amazon Cognito 身分集區，
- 將 AWS IoT Core 政策附加到 Amazon Cognito 身分識別 (已驗證的使用者)。

未經驗證和已驗證的 Amazon Cognito 使用者連線 AWS IoT Core 的政策範例

下列範例顯示在 IAM 政策與 Amazon Cognito 身分 IoT 政策中的許可。已驗證身分的使用者希望發佈到裝置特定主題 (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"
      ]
    }
  ]
}
```

```
}
```

下列範例顯示在 Amazon Cognito 未經驗證角色 IAM 政策中的許可。未驗證身分的使用者希望發佈到不需驗證身分的非裝置特定主題。

```
{
  "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

下面的示例 Web 應用程式 GitHub 顯示了如何將政策附件合併到用戶註冊和身份驗證過程中。

- [MQTT 發布/訂閱反應網絡應用程式使用和 AWS Amplify適用於 JavaScript 的 AWS IoT Device SDK](#)
- [MQTT 發布/訂閱反應網絡應用程式 AWS Amplify，使用，和 Lambda 函數適用於 JavaScript 的 AWS IoT Device SDK](#)

Amplify 是一組工具和服務，可協助您建置與 AWS 服務整合的 Web 和行動應用程式。如需有關 Amplify 的詳細資訊，請參閱 [Amplify Framework 說明文件](#)。

這兩個範例都會執行以下步驟。

1. 當使用者註冊帳戶時，應用程式會建立 Amazon Cognito 使用者集區和身分。
2. 當使用者進行身分驗證時，應用程式會建立政策並將其連接至身分。這會為使用者提供發佈和訂閱許可。
3. 使用者可以使用應用程式來發佈和訂閱 MQTT 主題。

第一個範例在身分驗證操作中直接使用 AttachPolicy API 操作。下面的範例示範如何在使用 Amplify 和 適用於 JavaScript 的 AWS IoT Device SDK 的 React Web 應用程式中實作此 API 呼叫。

```
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);
    }
  });
}
```

此代碼出現在 [AuthDisplay.js](#) 文件中。

第二個範例會在 Lambda 函數中實作 AttachPolicy API 操作。下列範例展示 Lambda 如何使用此 API 呼叫。

```
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});
    }
  });
```

此程式碼出現在 [app.js](#) 檔案的 `iot.GetPolicy` 函數中。

#### Note

當您使用透過 Amazon Cognito 身分識別集區取得的 AWS 登入資料呼叫函數時，Lambda 函數中的內容物件會包含的 `context.cognito_identity_id` 值。如需更多資訊，請參閱下列內容。

- [AWS Lambda Node.js 中的上下文對象](#)
- [AWS Lambda Python 中的上下文對象](#)
- [AWS Lambda Ruby 中的上下文對象](#)
- [AWS Lambda Java 中的上下文對象](#)
- [AWS Lambda Go 中的上下文對象](#)
- [AWS Lambda C# 中的上下文對象](#)
- [AWS Lambda 上下文對象 PowerShell](#)

## 使用 AWS IoT Core 認證提供者授權直接呼叫 AWS 服務

裝置可以使用 X.509 憑證來連線到 AWS IoT Core 使用 TLS 相互驗證通訊協定。其他 AWS 服務不支援憑證型驗證，但可以使用 [AWS 簽章版本 4](#) 格式的認 AWS 證來呼叫這些服務。[簽名版本 4 演算法](#) 通常要求呼叫者具有存取金鑰 ID 和秘密存取金鑰。AWS IoT Core 具有憑證提供者，可讓您使用內建的 [X.509 憑證](#) 作為唯一裝置身分識別來驗證 AWS 要求。這樣就不需要在裝置上儲存存取金鑰 ID 與私密存取金鑰。

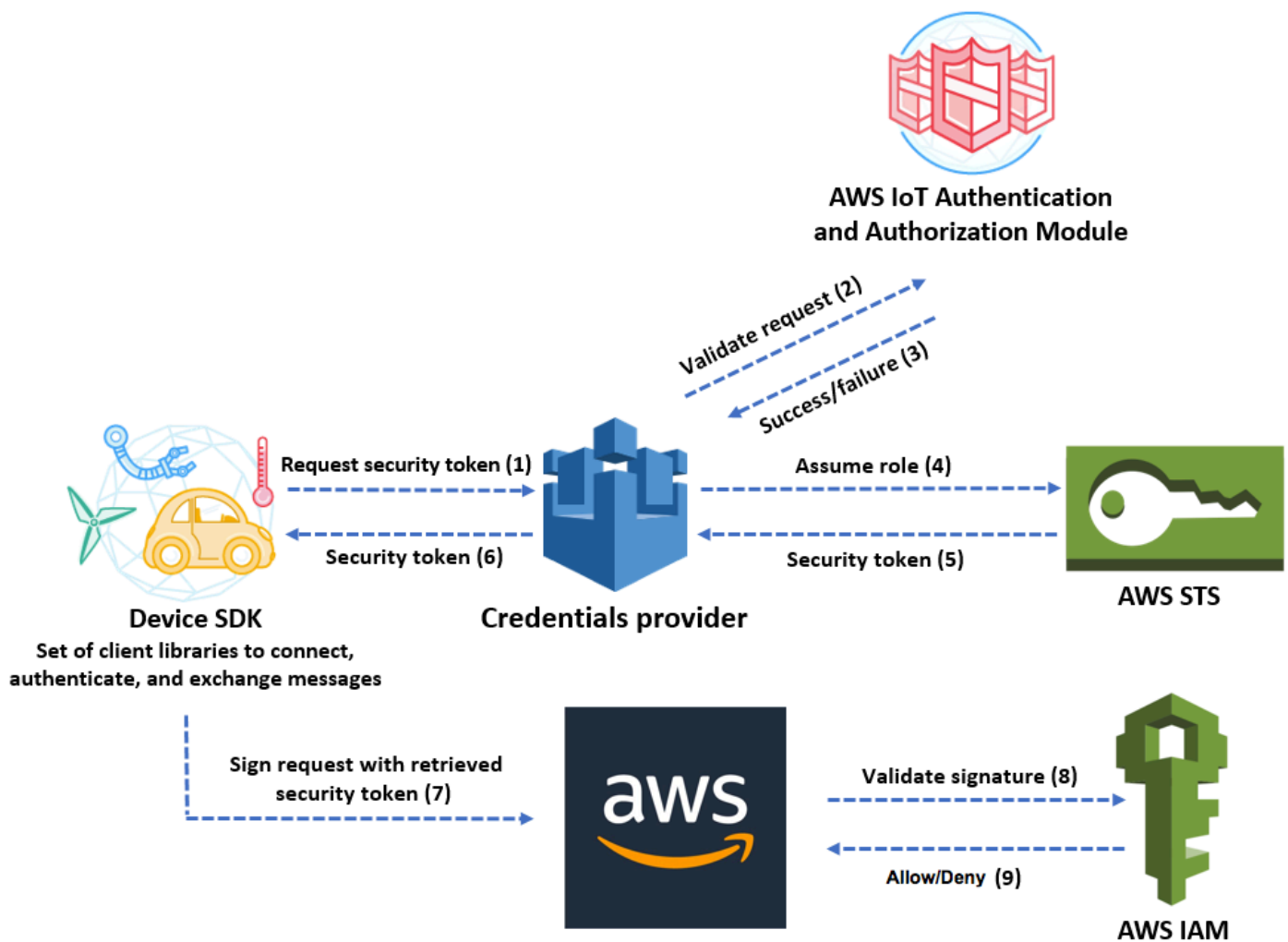
憑證提供者會使用 X.509 憑證來對呼叫者進行身分驗證並發出有限權限的臨時安全性字符。該令牌可用於簽署和驗證任何 AWS 請求。這種驗證 AWS 請求的方式需要您建立和設定 [AWS Identity and Access Management \(IAM\) 角色](#)，並將適當的 IAM 政策附加到該角色，以便登入資料提供者可以代表您擔任該角色。如需 AWS IoT Core 和 IAM 的詳細資訊，請參閱 [的身分識別與存取管理 AWS IoT](#)。

AWS IoT 要求裝置傳送 [伺服器名稱指示 \(SNI\) 延伸](#) 至傳輸層安全性 (TLS) 通訊協定，並在 `host_name` 欄位中提供完整的端點位址。該 `host_name` 字段必須包含您正在呼叫的端點，並且它必須是：

- 由 `aws iot describe-endpoint --endpoint-type iot:CredentialProvider` 傳回的 `endpointAddress`

沒有正確 `host_name` 值的裝置所嘗試的連線將會失敗。

下圖說明憑證提供者的工作流程。



1. AWS IoT Core 裝置會向憑證提供者發出 HTTPS 要求以取得安全性權杖。請求包含用於身分驗證的裝置 X.509 憑證。
2. 憑據提供者將請求轉發到身份驗 AWS IoT Core 證和授權模塊，以驗證證書並驗證設備是否具有請求安全令牌的權限。

3. 如果證書有效並且具有請求安全令牌的權限，則 AWS IoT Core 身份驗證和授權模塊返回成功。否則，它會傳送例外到裝置。
4. 在成功驗證憑證後，憑證提供者會叫用 [AWS Security Token Service](#)[AWS STS](#) 來擔任您為該憑證建立的 IAM 角色。
5. AWS STS 返回一個臨時的，有限的權限安全令牌給憑據提供者。
6. 憑證提供者傳回安全性字符到裝置。
7. 裝置會使用安全性權杖以簽 AWS 章版本 4 簽署 AWS 要求。
8. 請求的服務會叫用 IAM 來驗證簽章有效性並根據附加到您為憑證提供者建立的 IAM 角色的存取政策來授權請求。
9. 如果 IAM 成功驗證簽章有效性並授權請求，表示請求成功。否則，IAM 會傳送例外。

下列章節說明如何使用憑證來取得安全性字符。該流程假設您已經[註冊裝置](#)且[已建立並啟用自己的憑證](#)。

## 如何使用憑證來取得安全性字符

1. 設定憑證提供者代表您的裝置所擔任的 IAM 角色。將下列信任政策連接至該角色。

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

針對您要呼叫的每個 AWS 服務，將存取原則附加至該角色。憑證提供者支援以下政策變數：

- `credentials-iot:ThingName`
- `credentials-iot:ThingTypeName`
- `credentials-iot:AwsCertificateId`

當裝置提供其對 AWS 服務的請求中的物件名稱時，憑證提供者會將 `credentials-iot:ThingName` 與 `credentials-iot:ThingTypeName` 當作內容變數新增到安全性字符。



憑證提供者會提供 `credentials-iot:AwsCertificateId` 作為上下文變數，即使裝置在請求中未提供物件名稱。您以 `x-amzn-iot-thingname` HTTP 請求標頭的值來傳遞物件名稱。

這三個變數僅適用於 IAM 政策，不適用於 AWS IoT Core 政策。

- 請確認執行下一步驟 (建立角色別名) 的使用者擁有傳遞這個新建角色到 AWS IoT Core 的權限。下列原則同時提供 `iam:GetRole` 和 `iam:PassRole` 權限給 AWS 使用者。`iam:GetRole` 許可讓使用者取得有關您剛建立的角色資訊。該 `iam:PassRole` 權限允許用戶將角色傳遞給另一個 AWS 服務。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:GetRole",
      "iam:PassRole"
    ],
    "Resource": "arn:aws:iam::your AWS ## id:role/your role name"
  }
}
```

- 建立 AWS IoT Core 角色別名。要直接呼叫 AWS 服務的裝置必須知道連線時要使用哪個角色 ARN。AWS IoT Core 硬式編碼的角色 ARN 不是好的解決方案，因為您需要隨角色 ARN 變更而更新裝置。更好的解決方案是使用 `CreateRoleAlias` API 來建立指向該角色 ARN 的角色別名。如果角色 ARN 變更時，您只需更新角色別名。無需在裝置上進行變更。此 API 會使用下列參數：

`roleAlias`

必要。辨識角色別名的任意字串。可作為在角色別名資料模型中的主索引鍵功能。它包含 1-128 字元且必須僅包含英數字元和 =、@ 和 - 符號。可使用大寫和小寫字母字元。

`roleArn`

必要。角色別名所指的角色之 ARN。

`credentialDurationSeconds`

選用。憑證有效期間 (秒)。最低值為 900 秒 (15 分鐘)。最高值為 43,200 秒 (12 小時)。預設值為 3,600 秒 (1 小時)。

**⚠ Important**

AWS IoT Core 憑據提供程序可以發出最大生命週期為 43,200 秒 ( 12 小時 ) 的憑據。讓憑證有效時間長達 12 小時，有助於藉由快取憑證更長的時間，減少呼叫憑證提供者的次數。

`credentialDurationSeconds` 值必須小於或等於該角色別名參考的 IAM 角色最大工作階段持續時間。如需詳細資訊，請參閱《AWS Identity and Access Management 使用者指南》中[修改角色工作階段持續時間上限 \(AWS API\)](#)。

如需有關此 API 的詳細資訊，請參閱[CreateRole 別名](#)。

4. 將政策連接至裝置憑證。連接至本裝置憑證的政策，必須授予裝置承擔該角色的許可。您需要授予角色別名 `iot:AssumeRoleWithCertificate` 動作的許可，如下列範例。

```
{
  "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. 對憑證提供者發出 HTTPS 請求，以取得安全性字符。請提供下列資訊：

- 憑證：因為這是一個對 TLS 共同身分驗證的 HTTP 請求，您提出請求時需提供憑證與私有金鑰給用戶端。使用與註冊憑證時所使用的相同憑證和私密金鑰 AWS IoT Core。

若要確保您的裝置與通訊 AWS IoT Core (而非模擬裝置的服務)，請參閱[伺服器驗證](#)，點選連結下載適當的 CA 憑證，然後將它們複製到您的裝置。

- `RoleAlias`：您為認證提供者建立的角色別名名稱。
- `ThingName`：您註冊物件時建立的 AWS IoT Core 物件名稱。這會以 `x-amzn-iot-thingname` HTTP 標頭的值來傳遞。只有當您在 AWS IoT Core 或 IAM 政策中使用物屬性做為政策變數時，才需要此值。

**Note**

您ThingName在中提供的x-amzn-iot-thingname必須與指派給憑證的 AWS IoT 物件資源的名稱相符。如果不符合，則會傳回 403 錯誤。

在中執行下列命令，AWS CLI 以取得您的認證提供者端點 AWS 帳戶。如需此 API 的詳細資訊，請參閱[DescribeEndpoint](#)。

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

以下 JSON 物件為 describe-endpoint 命令的範例輸出。它包含用來要求安全字符的 endpointAddress。

```
{
  "endpointAddress": "your_aws_account_specific_prefix.credentials.iot.your region.amazonaws.com"
}
```

使用端點對憑證提供者發出 HTTPS 請求，以傳回安全性字符。下列範例命令使用 curl，但您可以使用任何 HTTP 用戶端。

```
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
```

此命令會傳回一個包含 accessKeyId、secretAccessKey、sessionToken 以及一個過期情況的安全性字符物件。以下 JSON 物件為 curl 命令的範例輸出。

```
{"credentials":{"accessKeyId":"access key","secretAccessKey":"secret access key","sessionToken":"session token","expiration":"2018-01-18T09:18:06Z"}}
```

然後，您可以使用accessKeyIdsecretAccessKey、和sessionToken值來簽署對 AWS 服務的要求。如需 [end-to-end 示範](#)，請參閱[如何使用AWS 安全部落格上的 AWS IoT 認證提供者部落格文章](#)，消除裝置中硬式編碼 AWS 認證的需求。

## 利用 IAM 跨帳戶存取

AWS IoT Core 可讓您讓主參與者發行或訂閱在非主參與者擁 AWS 帳戶 有的主題中定義的主題。您可以建立 IAM 政策和 IAM 角色，藉此設定跨帳戶存取，並將政策連接至該角色。

首先，如[建立 IAM 政策](#)中所述建立客戶受管的 IAM 政策，如同為 AWS 帳戶中的其他使用者和憑證所做的一樣。

對於在登 AWS IoT Core 錄中註冊的裝置，下列原則會授與 AWS IoT Core 使用符合裝置物件名稱的用戶端 ID 連線至的裝置，並發佈至 *Thing-name ##my/topic/thing-name #####* 的裝置的權限：

```
{
  "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/my/topic/
${iot:Connection.Thing.ThingName}"],
    }
  ]
}
```

對於未在登 AWS IoT Core 錄中註冊的裝置，下列策略會授予裝置使用在您帳戶 (123456789012) 登 AWS IoT Core 錄中client1註冊的物件名稱的權限，以連線至用戶端 ID 特定主題 AWS IoT Core 並將其發佈至名稱前綴為 : 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:Publish"
    ],
    "Resource": [
      "arn:aws:iot:us-east-1:123456789012:topic/my/topic/${iot:ClientId}"
    ]
  }
]
}

```

接著，依照[建立角色以將許可委派給 IAM 使用者](#)中的步驟。輸入您欲共用存取的 AWS 帳戶之帳戶 ID。最後一個步驟是，將政策連接至您剛建立的角色。若稍後您需要修改授予存取的 AWS 帳戶 ID，您可以使用下列信任政策的格式：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam:us-east-1:567890123456:user/MyUser"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

## 資料保護 AWS IoT Core

AWS [共用責任模型](#)適用於中的資料保護 AWS IoT Core。如此模型中所述，AWS 負責保護執行所有 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務的安全組

態和管理任務。如需資料隱私權的詳細資訊，請參閱[資料隱私權常見問答集](#)。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 [AWS 共同的責任模型和 GDPR](#) 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱[聯邦資訊處理標準 \(FIPS\) 140-2 概觀](#)。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用主控台、API AWS IoT 或 AWS SDK 時 AWS 服務 使用或其他使用時。AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

如需關於資料保護的詳細資訊，請參閱 AWS 安全部落格上的 [AWS 共同責任模型和歐盟《一般資料保護規範》\(GDPR\)](#) 部落格文章。

AWS IoT 設備收集數據，對該數據執行一些操作，然後將該數據發送到另一個 Web 服務。您可以選擇在裝置上短時間存放一些資料。您有責任對靜態資料提供資料保護。當您的裝置傳送資料至時 AWS IoT，它會透過 TLS 連線執行，如本節稍後所述。AWS IoT 設備可以將數據發送到任何 AWS 服務。如需有關每個服務資料安全性的詳細資訊，請參閱該服務的說明文件。AWS IoT 可以配置為將日誌寫入 CloudWatch 日誌並將 AWS IoT API 調用記錄到 AWS CloudTrail。如需有關這些服務資料安全性的詳細資訊，請參閱 [Amazon 的身份驗證和存取控制 CloudWatch](#) 和 [使用 AWS KMS 受管金鑰加密 CloudTrail 日誌檔案](#)。

## 中的資料加密 AWS IoT

依預設，傳輸中和靜態中的所有 AWS IoT 資料都會加密。[傳輸中的資料會使用 TLS 加密](#)，靜態資料會使用 AWS 擁有的金鑰加密。AWS IoT 目前不支援金鑰管理服務 AWS KMS keys () 的客戶管理 (KMS 金鑰 AWS KMS)；不過，「裝置建議程式」和「AWS IoT 無線」只會使用「」AWS 擁有的金鑰 來加密客戶資料。AWS

## 中的運輸安全 AWS IoT Core

TLS (Transport Layer Security) 是一種加密通訊協定，專為透過電腦網路進行安全通訊而設計。AWS IoT Core 裝置閘道要求客戶在傳輸途中加密所有通訊，方法是使用 TLS 來從裝置到閘道的連線。TLS 是用來實現所支援之應用程式通訊協定 (MQTT、HTTP 和 WebSocket) 的機密性。AWS IoT Core 多種程式設計語言與作業系統提供 TLS 支援。其中 AWS 的資料會由特定 AWS 服務加密。如需有關其他 AWS 服務之資料加密的詳細資訊，請參閱該服務的安全性說明文件。

### 目錄

- [TLS 通訊協定](#)
- [安全政策](#)
- [AWS IoT Core 中傳輸安全的重要注意事項](#)
- [LoRaWAN 無線裝置的傳輸安全性](#)

## TLS 通訊協定

AWS IoT Core 支援下列 TLS 通訊協定版本：

- TLS 1.3
- TLS 1.2

使用時 AWS IoT Core，您可以在網域組態中設定 [TLS 設定 \(適用於 TLS 1.2 和 TLS 1.3\)](#)。如需詳細資訊，請參閱 [???](#)。

## 安全政策

安全政策是 TLS 通訊協定及其密碼的組合，用來決定在用戶端與伺服器之間進行 TLS 交涉期間所支援的通訊協定和密碼。您可根據需要將裝置設定為使用預先定義的安全政策。請注意，AWS IoT Core 不支援自訂安全性原則。

將裝置連線至時，您可以為裝置選擇其中一個預先定義的安全性原則 AWS IoT Core。中最新預先定義的安全策略名稱 AWS IoT Core 包括以發行年份和月份為基礎的版本資訊。預設的預先定義安全政策為 `IotSecurityPolicy_TLS13_1_2_2022_10`。若要指定安全性原則，您可以使用 AWS IoT 主控台或 AWS CLI。如需詳細資訊，請參閱 [???](#)。

下表說明 AWS IoT Core 支援的最新預先定義安全政策。已從標頭列的政策名稱移除 `IotSecurityPolicy_`，使其相符。

安全政策	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*	TLS12_1_0_2015_01*		
TCP 連接埠	443/8443/8883	443/8443/8883	443/8443/8883	443	8443/8883	443	8443/8883
TLS 通訊協定							
TLS 1.2		✓	✓	✓	✓	✓	✓
TLS 1.3	✓	✓					
TLS 加密							
TLS_AES_128_GCM_SHA256	✓	✓					
TLS_AES_256_GCM_SHA384	✓	✓					
TLS_CHACHA20_POLY1305_SHA256	✓	✓					
ECDHE-RSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-SHA256		✓	✓	✓	✓	✓	✓



安全政策	TLS13_1_3 _2022_10	TLS13_1_2 _2022_10	TLS12_1_2 _2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE- RSA- AES128- SHA		✓	✓	✓	✓	✓	✓
ECDHE- RSA- AES256- GCM- SHA384		✓	✓	✓	✓	✓	✓
ECDHE- RSA- AES256- SHA384		✓	✓	✓	✓	✓	✓
ECDHE- RSA- AES256- SHA		✓	✓	✓	✓	✓	✓
AES128- GCM- SHA256		✓	✓	✓	✓	✓	✓
AES128- SHA256		✓	✓	✓		✓	✓
AES128- SHA		✓	✓	✓	✓	✓	✓
AES256- GCM- SHA384		✓	✓	✓	✓	✓	✓

安全政策	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
AES256-SHA256		✓	✓	✓	✓	✓	✓
AES256-SHA		✓	✓	✓	✓	✓	✓
DHE-RSA-AES256-SHA						✓	✓
ECDHE-ECDSA-AES128-GCM-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA256		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES256-GCM-SHA384		✓	✓	✓	✓	✓	✓

安全政策	TLS13_1_3_2022_10	TLS13_1_2_2022_10	TLS12_1_2_2022_10	TLS12_1_0_2016_01*		TLS12_1_0_2015_01*	
ECDHE-ECDSA-AES256-SHA384		✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES256-SHA		✓	✓	✓	✓	✓	✓

### Note

TLS12\_1\_0\_2016\_01僅適用於以下內容 AWS 區域：ap-east-1，ap-東北-2，ap-south-1，AP-東南-2，ca-central-1，CN-北-1，CN-西北-1，歐洲-西-1，eu-west-3-2，美國-me-south-1，美國 sa-east-1，us-east-2-1，us-west-1，us-gov-west us-gov-west

TLS12\_1\_0\_2015\_01僅適用於以下內容 AWS 區域：ap-northeast-1，ap-southeast-1，eu-central-1，eu-west-1，us-east-1，us-west-2。

## AWS IoT Core中傳輸安全的重要注意事項

對於 AWS IoT Core 使用 [MQTT](#) 連線的裝置，TLS 會加密裝置與代理程式之間的連線，並 AWS IoT Core 使用 TLS 用戶端驗證來識別裝置。如需更多資訊，請參閱[用戶端身分驗證](#)。對於 AWS IoT Core 使用 [HTTP](#) 連線的裝置，TLS 會加密裝置與代理程式之間的連線，並將驗證委派給 AWS 簽章版本 4。如需詳細資訊，請參閱《AWS 一般參考》中的[使用 Signature 第 4 版簽署請求](#)。

當您將裝置連線到時 AWS IoT Core，不需要傳送[伺服器名稱指示 \(SNI\) 延伸功能](#)，但強烈建議您傳送。若要使用[多帳戶註冊](#)、[自訂網域](#)、[VPC 端點](#)和[設定的 TLS 策略](#)等功能，您必須使用 SNI 延伸模組，並在欄位中提供完整的端點位址。host\_name該 host\_name 欄位必須包含您正在呼叫的端點。該端點必須是下列其中一個：

- aws iot [describe-endpoint](#) --endpoint-type iot:Data-ATS 返回的 endpointAddress

- `aws iot describe-domain-configuration --domain-configuration-name "domain_configuration_name"` 返回的 `domainName`

`host_name` 值不正確或無效的裝置所嘗試的連線將會失敗。AWS IoT Core 會將 [自訂驗證](#) CloudWatch 的驗證類型的失敗記錄到。

AWS IoT Core 不支援 [SessionTicket TLS 擴充功能](#)。

## LoRaWAN 無線裝置的傳輸安全性

LoRaWAN 設備遵循 [金雅拓，動力和 Semtech 的 LoRa WAN™ SECURITY：為 LoRa 聯盟準備的白皮書中描述的安全實踐](#)。™

如需 LoRa WAN 裝置傳輸安全性的詳細資訊，請參閱 [LoRaWAN 資料和傳輸安全性](#)。

## 中的資料加密 AWS IoT

資料保護是指在傳輸中（往返 AWS IoT）和靜態（當資料儲存在裝置或由其他 AWS 服務時）時保護資料。傳送至的所有資料都會使用 MQTT、HTTPS 和 WebSocket 通訊協定透過 TLS 連線傳送，因 AWS IoT 此在傳輸過程中預設為安全。AWS IoT 設備收集數據，然後將其發送到其他 AWS 服務以進一步處理。如需有關其他 AWS 服務上資料加密的詳細資訊，請參閱該服務的安全性文件。

FreeRTOS 提供 PKCS#11 程式庫，可抽象化金鑰儲存、存取密碼編譯物件和管理工作階段。您有責任使用此程式庫來加密裝置上的靜態資料。如需詳細資訊，請參閱 [FreeRTOS 公有金鑰加密標準 \(PKCS\) #11 程式庫](#)。

## Device Advisor

### 傳輸中加密

傳送至 Device Advisor 或從中傳送的資料都會在傳輸中加密。使用 Device Advisor API 時，所有傳送至服務或從中傳送的資料都會使用 Signature 第 4 版加密。如需如何簽署 AWS API 要求的詳細資訊，請參閱 [簽署 AWS API 要求](#)。從測試裝置傳送至 Device Advisor 測試端點的所有資料都會透過 TLS 連線傳送，因此在傳輸時預設為安全的。

## 金鑰管理 AWS IoT

所有的連線 AWS IoT 都是使用 TLS 完成的，因此初始 TLS 連線不需要用戶端加密金鑰。

裝置必須使用 X.509 憑證或 Amazon Cognito 身分來進行身分驗證。您可以讓 AWS IoT 為您產生憑證，在這種情況下，它會產生公有/私有金鑰對。如果您使用主 AWS IoT 控制台，系統會提示您下載憑證

和金鑰。如果您使用 [create-keys-and-certificate](#) CLI 命令，CLI 命令會傳回憑證和金鑰。您有責任將憑證和私有金鑰複製到您的裝置上，並保護它的安全。

AWS IoT 目前不支援 () 的客戶管理 AWS KMS keys (KMS 金鑰AWS KMS)；不過，「裝置建議程式」和「AWS IoT 無線」僅使用 AWS 擁有的金鑰來加密客戶資料。AWS Key Management Service

Device Advisor

使用 AWS API 時傳送至「裝置建議程式」的所有資料都會在靜態時加密。Device Advisor 會使用 [AWS Key Management Service](#) 中存放和管理的 KMS 金鑰，來加密所有靜態資料。裝置建議程式會使用來加密您的資料 AWS 擁有的金鑰。如需有關的更多資訊 AWS 擁有的金鑰，請參閱[AWS 擁有的金鑰](#)。

## 的身分識別與存取管理 AWS IoT

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有權限) 來使用 AWS IoT 資源。IAM 是您可以使用的 AWS 服務，無需額外付費。

主題

- [物件](#)
- [使用 IAM 身分來驗證](#)
- [使用政策管理存取權](#)
- [如何與 IAM AWS IoT 搭配使用](#)
- [AWS IoT 以識別為基礎的原則範例](#)
- [AWS 受管理的政策 AWS IoT](#)
- [疑難排解 AWS IoT 身分和存取](#)

## 物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在進行的工作 AWS IoT。

服務使用者 — 如果您使用 AWS IoT 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多 AWS IoT 功能來完成工作時，您可能需要其他權限。了解存取的管理方式可協助您向管理員請求正確的許可。若您無法存取 AWS IoT 中的某項功能，請參閱 [疑難排解 AWS IoT 身分和存取](#)。

服務管理員 — 如果您負責公司的 AWS IoT 資源，您可能擁有完整的存取權 AWS IoT。決定您的服務使用者應該存取哪些 AWS IoT 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步瞭解貴公司如何搭配使用 IAM AWS IoT，請參閱[如何與 IAM AWS IoT 搭配使用](#)。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 AWS IoT 存取權的詳細資訊。若要檢視可在 IAM 中使用的 AWS IoT 基於身分的政策範例，請參閱。[AWS IoT 以識別為基礎的原則範例](#)

## 使用 IAM 身分來驗證

AWS IoT 身分識別可以是裝置 (X.509) 憑證、Amazon Cognito 身分或 IAM 使用者或群組。本主題僅討論 IAM 身分。如需其他 AWS IoT 支援身分的詳細資訊，請參閱[用戶端身分驗證](#)。

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》AWS 帳戶中的[如何登入](#)您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的[簽署 AWS API 請求](#)。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的[多重要素驗證](#)和 IAM 使用者指南中的[在 AWS 中使用多重要素驗證 \(MFA\)](#)。

## AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的[需要根使用者憑證的任務](#)。

## IAM 使用者和群組

[IAM 使用者](#)是您內部的身份，具 AWS 帳戶有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 [IAM 使用者指南](#)中的為需要長期憑證的使用案例定期輪換存取金鑰。

[IAM 群組](#)是一種指定 IAM 使用者集合的身份。您無法以群組身份簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的[建立 IAM 使用者 \(而非角色\) 的時機](#)。

## IAM 角色

[IAM 角色](#)是您 AWS 帳戶 內部具有特定許可的身份。它類似 IAM 使用者，但不與特定的人員相關聯。您可以[切換角色，在中暫時擔任 IAM 角色](#)。AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的[使用 IAM 角色](#)。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身份使用者存取 — 如需向聯合身份指派許可，請建立角色，並為角色定義許可。當聯合身份進行身份驗證時，該身份會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 [IAM 使用者指南](#)中的為第三方身份提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身份驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的[許可集](#)。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源 (而不是使用角色作為代理)。若要了解跨帳戶存取角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 [IAM 中的跨帳戶資源存取](#)。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。

- 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱 [《轉發存取工作階段》](#)。
- 服務角色 – 服務角色是服務擔任的 [IAM 角色](#)，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的 [建立角色以委派許可給 AWS 服務服務](#)。
- 服務連結角色 — 服務連結角色是連結至 AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中，AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的 [利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式](#)。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的 [建立 IAM 角色 \(而非使用者\) 的時機](#)。

## 使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS，當與身分識別或資源相關聯時，會定義其權限。AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 [JSON 政策概觀](#)。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。



IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

## 身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的 [建立 IAM 政策](#)。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶。受管政策包括 AWS 受管政策和客戶管理的策略。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的 [在受管政策和內嵌政策間選擇](#)。

## 資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中 [指定主體](#)。主參與者可以包括帳戶、使用者、角色、同盟使用者或。AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

## 存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。AWS WAF 如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的 [存取控制清單 \(ACL\) 概觀](#)。

## 其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交

集。會在 Principal 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 [IAM 實體許可界限](#)。

- 服務控制策略 ( SCP ) — SCP 是 JSON 策略，用於指定中組織或組織單位 ( OU ) 的最大權限。AWS Organizations 是一種用於分組和集中管理您企業擁有的多個 AWS 帳戶的服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需 Organizations 和 SCP 的詳細資訊，請參閱 AWS Organizations 使用者指南中的 [SCP 運作方式](#)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的 [工作階段政策](#)。

## 多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的 [政策評估邏輯](#)。

## 如何與 IAM AWS IoT 搭配使用

在您使用 IAM 管理存取權限之前 AWS IoT，您應該瞭解哪些 IAM 功能可搭配使用 AWS IoT。若要深入瞭解如何 AWS IoT 和其他 AWS 服務如何使用 IAM，請參閱 IAM 使用者指南中的與 IAM 搭配使用的 [AWS 服務](#)。

### 主題

- [AWS IoT 身分型政策](#)
- [AWS IoT 資源型政策](#)
- [以 AWS IoT 標籤為基礎的授權](#)
- [AWS IoT IAM 角色](#)

## AWS IoT 身分型政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下會允許或拒絕動作。AWS IoT 支援特定動作、資源及條件索引鍵。若要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 [JSON 政策元素參考](#)。


## 動作

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

下表列出 IAM IoT 動作、相關聯的 AWS IoT API 以及動作所操控的資源。

政策動作	AWS IoT API	資源
物聯網:AcceptCertificate轉移	AcceptCertificate轉移	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> Note</p> <p>ARN 中 AWS 帳戶 指定的必須是傳輸憑證的目標帳戶。</p> </div>		
物聯網:AddThingToThing集團	AddThingToThing集團	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>  arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>
物聯網:AssociateTargetsWithJob	AssociateTargetsWithJob	無
物聯網:AttachPolicy	AttachPolicy	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>  或  arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>

政策動作	AWS IoT API	資源
物聯網:AttachPrincipal政策	AttachPrincipal政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網 : AttachSecurity設定檔	AttachSecurity設定檔	arn:aws:iot: <i>region</i> : <i>account-id</i> :securityprofile/ <i>security-profile-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
物聯網:AttachThing首席	AttachThing主要	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:CancelCertificate轉移	CancelCertificate轉移	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>  <div style="border: 1px solid #007bff; border-radius: 10px; padding: 10px; background-color: #e6f2ff;"> <p> Note ARN 中 AWS 帳戶 指定的必須是傳輸憑證的目標帳戶。</p> </div>
物聯網:CancelJob	CancelJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
物聯網:CancelJob執行	CancelJob執行	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:ClearDefault授權者	ClearDefault授權者	無
物聯網:CreateAuthorizer	CreateAuthorizer	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizer/ <i>authorizer-function-name</i>
物聯網:CreateCertificateFromCsr	CreateCertificateFromCsr	*

政策動作	AWS IoT API	資源
物聯網:CreateDimension	CreateDimension	arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
物聯網:CreateJob	CreateJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :jobtemplate/ <i>job-template-id</i>
物聯網:CreateJob模板	CreateJob模板	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :jobtemplate/ <i>job-template-id</i>
物聯網:CreateKeysAndCertificate	CreateKeysAndCertificate	*
物聯網:CreatePolicy	CreatePolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:CreatePolicy版本	CreatePolicy版本	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
<div style="border: 1px solid #add8e6; border-radius: 15px; padding: 10px; background-color: #e6f2ff;"> <p> Note 這必須是 AWS IoT 政策，而不是 IAM 政策。</p> </div>		
物聯網:CreateRole別名	CreateRole別名	(參數 : roleAlias)  arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealiases/ <i>role-alias-name</i>

政策動作	AWS IoT API	資源
物聯網 : CreateSecurity設定檔	CreateSecurity設定檔	arn:aws:iot: <i>region</i> : <i>account-id</i> :securityprofile/ <i>security-profile-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
物聯網:CreateThing	CreateThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:CreateThing組	CreateThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>  適用正在建立的群組和父群組 (若使用)
物聯網:CreateThing類型	CreateThing类型	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
物聯網:CreateTopic規則	CreateTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網:DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizer/ <i>authorizer-name</i>
iot:DeleteCACertificate	DeleteCACertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cacert/ <i>cert-id</i>
物聯網:DeleteCertificate	DeleteCertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:DeleteDimension	DeleteDimension	arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
物聯網:DeleteJob	DeleteJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
物聯網:DeleteJob模板	DeleteJob模板	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-template-id</i>

政策動作	AWS IoT API	資源
物聯網:DeleteJob執行	DeleteJob執行	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:DeletePolicy	DeletePolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:DeletePolicy版本	DeletePolicy版本	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:DeleteRegistration代碼	DeleteRegistration代碼	*
物聯網:DeleteRole別名	DeleteRole別名	arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealiases/ <i>role-alias-name</i>
物聯網 : DeleteSecurity設定檔	DeleteSecurity設定檔	arn:aws:iot: <i>region</i> : <i>account-id</i> :securityprofile/ <i>security-profile-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
物聯網:DeleteThing	DeleteThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:DeleteThing組	DeleteThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:DeleteThing類型	DeleteThing類型	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
物聯網:DeleteTopic規則	DeleteTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網:刪除 V2 LoggingLevel	刪除 V2 LoggingLevel	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>

政策動作	AWS IoT API	資源
物聯網:DeprecateThing類型	DeprecateThing 类型	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
物聯網:DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizer/ <i>authorizer-function-name</i>  (參數 : authorizerName) 無
iot:DescribeCACertificate	DescribeCACertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cacert/ <i>cert-id</i>
物聯網:DescribeCertificate	DescribeCertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:DescribeDefault授權者	DescribeDefault授權者	無
物聯網:DescribeEndpoint	DescribeEndpoint	*
物聯網 : DescribeEvent配置	DescribeEvent配置	無
物聯網:DescribeIndex	DescribeIndex	arn:aws:iot: <i>region</i> : <i>account-id</i> :index/ <i>index-name</i>
物聯網:DescribeJob	DescribeJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
物聯網:DescribeJob執行	DescribeJob執行	無
物聯網:DescribeJob模板	DescribeJob模板	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-template-id</i>
物聯網:DescribeRole別名	DescribeRole別名	arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealiases/ <i>role-alias-name</i>



政策動作	AWS IoT API	資源
物聯網:DescribeThing	DescribeThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:DescribeThing組	DescribeThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:DescribeThingRegistrationTask	DescribeThingRegistrationTask	無
物聯網:DescribeThing類型	DescribeThing類型	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
物聯網:DetachPolicy	DetachPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>  或  arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:DetachPrincipal政策	DetachPrincipal政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網 : DetachSecurity設定檔	DetachSecurity設定檔	arn:aws:iot: <i>region</i> : <i>account-id</i> :securityprofile/ <i>security-profile-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
物聯網:DetachThing首席	DetachThing主要	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:DisableTopic規則	DisableTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網:EnableTopic規則	EnableTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>

政策動作	AWS IoT API	資源
物聯網:GetEffective政策	GetEffective政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:GetIndexing配置	GetIndexing配置	無
物聯網:GetJob文件	GetJob文件	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
物聯網:GetLogging選項	GetLogging選項	*
物聯網:GetPolicy	GetPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:GetPolicy版本	GetPolicy版本	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:GetRegistration代碼	GetRegistration代碼	*
物聯網:GetTopic規則	GetTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網>ListAttached政策	ListAttached政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>  或  arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網>ListAuthorizers	ListAuthorizers	無
iot>ListCACertificates	ListCACertificates	*

政策動作	AWS IoT API	資源
物聯網:ListCertificates	ListCertificates	*
物聯網:ListCertificatesBYCA	ListCertificatesBYCA	*
物聯網:ListIndices	ListIndices	無
物聯網:ListJobExecutionsForJob	ListJobExecutionsForJob	無
物聯網:ListJobExecutionsFor事	ListJobExecutionsFor事	無
物聯網:ListJobs	ListJobs	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>  如果使用 thingGroupName 參數
物聯網:ListJob範本	ListJobs	無
物聯網:ListOutgoing證書	ListOutgoing證書	*
物聯網:ListPolicies	ListPolicies	*
物聯網:ListPolicy校長	ListPolicy校長	*
物聯網:ListPolicy版本	ListPolicy版本	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>

政策動作	AWS IoT API	資源
物聯網:ListPrincipal政策	ListPrincipal政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:ListPrincipal事情	ListPrincipal事情	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:ListRole別名	ListRole別名	無
物聯網:ListTargetsForPolicy	ListTargetsForPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:ListThing群組	ListThing群組	無
物聯網:ListThingGroupsFor事	ListThingGroupsFor事	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:ListThing校長	ListThing校長	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:ListThingRegistrationTask報告	ListThingRegistrationTask報告	無
物聯網:ListThingRegistrationTasks	ListThingRegistrationTasks	無
物聯網:ListThing類型	ListThing類型	*
物聯網:ListThings	ListThings	*

政策動作	AWS IoT API	資源
物聯網:ListThingsInThing集團	ListThingInThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:ListTopic規則	ListTopic規則	*
物聯網:清單 V2 LoggingLevels	列表 V2 LoggingLevels	無
iot:RegisterCACertificate	RegisterCACertificate	*
物聯網:RegisterCertificate	RegisterCertificate	*
物聯網:RegisterThing	RegisterThing	無
物聯網:RejectCertificate轉移	RejectCertificate轉移	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:RemoveThingFromThing集團	RemoveThingFromThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:ReplaceTopic規則	ReplaceTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網:SearchIndex	SearchIndex	arn:aws:iot: <i>region</i> : <i>account-id</i> :index/ <i>index-id</i>
物聯網:SetDefault授權者	SetDefault授權者	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizer/ <i>authorizer-function-name</i>

政策動作	AWS IoT API	資源
物聯網:SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:SetLogging選項	SetLogging選項	arn:aws:iot: <i>region</i> : <i>account-id</i> :role/ <i>role-name</i>
物聯網:設定 V2 LoggingLevel	第 2 集 LoggingLevel	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:設定 V2 LoggingOptions	第 2 集 LoggingOptions	arn:aws:iot: <i>region</i> : <i>account-id</i> :role/ <i>role-name</i>
物聯網:StartThingRegistrationTask	StartThingRegistrationTask	無
物聯網:StopThingRegistrationTask	StopThingRegistrationTask	無
物聯網:TestAuthorization	TestAuthorization	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:TestInvoke授權者	TestInvoke授權者	無
物聯網:TransferCertificate	TransferCertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizerfunction/ <i>authorizer-function-name</i>
iot:UpdateCACertificate	UpdateCACertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cacert/ <i>cert-id</i>

政策動作	AWS IoT API	資源
物聯網:UpdateCertificate	UpdateCertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:UpdateDimension	UpdateDimension	arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
物聯網 : UpdateEvent配置	UpdateEvent配置	無
物聯網:UpdateIndexing配置	UpdateIndexing配置	無
物聯網:UpdateRole別名	UpdateRole別名	arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealiases/ <i>role-alias-name</i>
物聯網 : UpdateSecurity設定檔	UpdateSecurity設定檔	arn:aws:iot: <i>region</i> : <i>account-id</i> :securityprofile/ <i>security-profile-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :dimension/ <i>dimension-name</i>
物聯網:UpdateThing	UpdateThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:UpdateThing組	UpdateThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:UpdateThingGroupsFor事	UpdateThingGroupsFor事	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>  arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>

中的策略動作在動作之前 AWS IoT 使用下列前置詞 : `iot:`。例如，若要授予某人列出在其 AWS 帳戶使用 `ListThings` API 註冊的所有 IoT 物件的權限，您可以將該 `iot>ListThings` 動作包含在他們的政策中。原則陳述式必須包含 `Action` 或 `NotAction` 元素。AWS IoT 定義了它自己的一組動作，描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
  "ec2:action1",
  "ec2:action2"
```

您也可以使用萬用字元 (\*) 來指定多個動作。例如，若要指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "iot:Describe*"
```

若要查看 AWS IoT 動作清單，請參閱《IAM 使用者指南》AWS IoT 中的 [「定義的動作」](#)。

### Device Advisor 動作

下表列出 IAM IoT Device Advisor 動作、相關的 AWS IoT Device Advisor API，以及動作所操控的資源。

政策動作	AWS IoT API	資源
物聯網設備 顧問：定義 CreateSuite	CreateSuite定義	無
物聯網設備 顧問：定義 DeleteSuite	DeleteSuite定義	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>
物聯網設備 顧問：定義 GetSuite	GetSuite定義	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>
物聯網設備 顧問：執行 GetSuite	GetSuite執行	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-run-id</i>
物聯網設備顧 問：GetSuiteRu nReport	GetSuiteR unReport	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/ <i>suite-definition-id</i> / <i>suite-run-id</i>



政策動作	AWS IoT API	資源
物聯網設備顧問：定義 ListSuite	ListSuite定義	無
物聯網設備顧問：執行 ListSuite	ListSuite執行	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>
物聯網設備顧問：ListTagsForResource	ListTagsForResource	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>  arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/suite-definition-id/ <i>suite-run-id</i>
物聯網設備顧問：執行 StartSuite	StartSuite執行	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>
物聯網設備顧問：TagResource	TagResource	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>  arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/suite-definition-id/ <i>suite-run-id</i>
物聯網設備顧問：UntagResource	UntagResource	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>  arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/suite-definition-id/ <i>suite-run-id</i>
物聯網設備顧問：定義 UpdateSuite	UpdateSuite定義	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suitedefinition/ <i>suite-definition-id</i>

政策動作	AWS IoT API	資源
物聯網設備顧問：執行 StopSuite	StopSuite執行	arn:aws:iotdeviceadvisor: <i>region</i> : <i>account-id</i> :suiterun/suite-definition-id/ <i>suite-run-id</i>

「AWS IoT 裝置建議程式」中的原則動作會在動作之前使用下列前置詞：iotdeviceadvisor:。例如，若要授 AWS 帳戶 與某人列出在 ListSuiteDefinitions API 中註冊的所有套件定義的權限，您可以將該iotdeviceadvisor:ListSuiteDefinitions動作包含在他們的政策中。

## 資源

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。


Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (\*) 來表示陳述式適用於所有資源。

```
"Resource": "*"

```

## AWS IoT 資源

政策動作	AWS IoT API	資源
物聯網:AcceptCertificate轉移	AcceptCertificate轉移	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin: 10px 0;"> <p> <b>Note</b></p> <p>ARN 中 AWS 帳戶 指定的必須是傳輸憑證的目標帳戶。</p> </div>		
物聯網:AddThingToThing集團	AddThingToThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>

政策動作	AWS IoT API	資源
		arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:AssociateTargetsWithJobs	AssociateTargetsWithJobs	無
物聯網:AttachPolicy	AttachPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i> 或 arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:AttachPrincipalPolicy	AttachPrincipalPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:AttachThingPrincipal	AttachThingPrincipal	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:CancelCertificateTransfer	CancelCertificateTransfer	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
		<div style="border: 1px solid #add8e6; border-radius: 15px; padding: 10px; background-color: #e6f2ff;"> <p> Note</p> <p>ARN 中 AWS 帳戶 指定的必須是傳輸憑證的目標帳戶。</p> </div>
物聯網:CancelJob	CancelJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
物聯網:CancelJobExecution	CancelJobExecution	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>

政策動作	AWS IoT API	資源
物聯網:Clear Default授權者	ClearDefault授權者	無
物聯網:CreateAuthorizer	CreateAuthorizer	arn:aws:iot: <i>region:account-id</i> :authorizer/ <i>authorizer-function-name</i>
物聯網:CreateCertificateFromCsr	CreateCertificateFromCsr	*
物聯網:CreateJob	CreateJob	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i> arn:aws:iot: <i>region:account-id</i> :jobtemplate/ <i>job-template-id</i>
物聯網:CreateJob模板	CreateJob模板	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i> arn:aws:iot: <i>region:account-id</i> :jobtemplate/ <i>job-template-id</i>
物聯網:CreateKeysAndCertificate	CreateKeysAndCertificate	*
物聯網:CreatePolicy	CreatePolicy	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>

政策動作	AWS IoT API	資源
CreatePolicy版本	物聯網:CreatePolicy版本	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"> <p> Note 這必須是 AWS IoT 政策，而不是 IAM 政策。</p> </div>		
物聯網:CreateRole別名	CreateRole別名	(參數 : roleAlias) arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealias/ <i>role-alias-name</i>
物聯網:CreateThing	CreateThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:CreateThing組	CreateThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i> 適用正在建立的群組和父群組 (若使用)
物聯網:CreateThing類型	CreateThing类型	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
物聯網:CreateTopic規則	CreateTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網>DeleteAuthorizer	DeleteAuthorizer	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizer/ <i>authorizer-name</i>
iot>DeleteCACertificate	DeleteCACertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cacert/ <i>cert-id</i>
物聯網>DeleteCertificate	DeleteCertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網>DeleteJob	DeleteJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>

政策動作	AWS IoT API	資源
物聯網:DeleteJob執行	DeleteJob執行	arn:aws:iot: <i>region:account-id</i> :job/ <i>job-id</i>  arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>
物聯網:DeleteJob模板	DeleteJob模板	arn:aws:iot: <i>region:account-id</i> :jobtemplate/ <i>job-template-id</i>
物聯網:DeletePolicy	DeletePolicy	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>
物聯網:DeletePolicy版本	DeletePolicy版本	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>
物聯網:DeleteRegistration代碼	DeleteRegistration代碼	*
物聯網:DeleteRole別名	DeleteRole別名	arn:aws:iot: <i>region:account-id</i> :rolealiases/ <i>role-alias-name</i>
物聯網:DeleteThing	DeleteThing	arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>
物聯網:DeleteThing組	DeleteThing集團	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:DeleteThing類型	DeleteThing类型	arn:aws:iot: <i>region:account-id</i> :thingtype/ <i>thing-type-name</i>
物聯網:DeleteTopic規則	DeleteTopic規則	arn:aws:iot: <i>region:account-id</i> :rule/ <i>rule-name</i>
物聯網:刪除 V2 LoggingLevel	刪除 V2 LoggingLevel	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:DeprecateThing類型	DeprecateThing类型	arn:aws:iot: <i>region:account-id</i> :thingtype/ <i>thing-type-name</i>

政策動作	AWS IoT API	資源
物聯網:DescribeAuthorizer	DescribeAuthorizer	arn:aws:iot: <i>region</i> : <i>account-id</i> :authorizer/ <i>authorizer-function-name</i>  (參數 : authorizerName) 無
iot:DescribeCACertificate	DescribeCACertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cacert/ <i>cert-id</i>
物聯網:DescribeCertificate	DescribeCertificate	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:DescribeDefault授權者	DescribeDefault授權者	無
物聯網:DescribeEndpoint	DescribeEndpoint	*
物聯網 : DescribeEvent配置	DescribeEvent配置	無
物聯網:DescribeIndex	DescribeIndex	arn:aws:iot: <i>region</i> : <i>account-id</i> :index/ <i>index-name</i>
物聯網:DescribeJob	DescribeJob	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>
物聯網:DescribeJob執行	DescribeJob執行	無
物聯網:DescribeJob模板	DescribeJob模板	arn:aws:iot: <i>region</i> : <i>account-id</i> :jobtemplate/ <i>job-template-id</i>
物聯網:DescribeRole別名	DescribeRole別名	arn:aws:iot: <i>region</i> : <i>account-id</i> :rolealiases/ <i>role-alias-name</i>
物聯網:DescribeThing	DescribeThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>

政策動作	AWS IoT API	資源
物聯網:DescribeThing組	DescribeThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:DescribeThingRegistrationTask	DescribeThingRegistrationTask	無
物聯網:DescribeThing類型	DescribeThing類型	arn:aws:iot: <i>region</i> : <i>account-id</i> :thingtype/ <i>thing-type-name</i>
物聯網:DetachPolicy	DetachPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i> 或 arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:DetachPrincipal政策	DetachPrincipal政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:DetachThing首席	DetachThing主要	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:DisableTopic規則	DisableTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網:EnableTopic規則	EnableTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網:GetEffective政策	GetEffective政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:GetIndexing配置	GetIndexing配置	無
物聯網:GetJob文件	GetJob文件	arn:aws:iot: <i>region</i> : <i>account-id</i> :job/ <i>job-id</i>



政策動作	AWS IoT API	資源
物聯網:GetLogging選項	GetLogging選項	*
物聯網:GetPolicy	GetPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:GetPolicy版本	GetPolicy版本	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:GetRegistration代碼	GetRegistration代碼	*
物聯網:GetTopic規則	GetTopic規則	arn:aws:iot: <i>region</i> : <i>account-id</i> :rule/ <i>rule-name</i>
物聯網>ListAttached政策	ListAttached政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>  或  arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網>ListAuthorizers	ListAuthorizers	無
iot>ListCACertificates	ListCACertificates	*
物聯網>ListCertificates	ListCertificates	*
物聯網>ListCertificatesBYCA	ListCertificatesBYCA	*
物聯網>ListIndices	ListIndices	無

政策動作	AWS IoT API	資源
物聯網:ListJobExecutionsForJobs	ListJobExecutionsForJobs	無
物聯網:ListJobExecutionsFor事	ListJobExecutionsFor事	無
物聯網:ListJobs	ListJobs	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>  如果使用 thingGroupName 參數
物聯網:ListJob範本	ListJob模板	無
物聯網:ListOutgoing證書	ListOutgoing證書	*
物聯網:ListPolicies	ListPolicies	*
物聯網:ListPolicy校長	ListPolicy校長	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:ListPolicy版本	ListPolicy版本	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:ListPrincipal政策	ListPrincipal政策	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:ListPrincipal事情	ListPrincipal事情	arn:aws:iot: <i>region</i> : <i>account-id</i> :cert/ <i>cert-id</i>
物聯網:ListRole別名	ListRole別名	無

政策動作	AWS IoT API	資源
物聯網:ListTargetsForPolicy	ListTargetsForPolicy	arn:aws:iot: <i>region</i> : <i>account-id</i> :policy/ <i>policy-name</i>
物聯網:ListThing群組	ListThing群組	無
物聯網:ListThingGroupsFor事	ListThingGroupsFor事	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:ListThing校長	ListThing校長	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:ListThingRegistrationTask報告	ListThingRegistrationTask報告	無
物聯網:ListThingRegistrationTasks	ListThingRegistrationTasks	無
物聯網:ListThing類型	ListThing類型	*
物聯網:ListThings	ListThings	*
物聯網:ListThingsInThing集團	ListThingsInThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:ListTopic規則	ListTopic規則	*
物聯網:清單 V2 LoggingLevels	列表 V2 LoggingLevels	無

政策動作	AWS IoT API	資源
iot:RegisterCACertificate	RegisterCACertificate	*
物聯網:RegisterCertificate	RegisterCertificate	*
物聯網:RegisterThing	RegisterThing	無
物聯網:RejectCertificate轉移	RejectCertificate轉移	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
物聯網:RemoveThingFromThingGroup	RemoveThingFromThingGroup	arn:aws:iot: <i>region:account-id</i> :thinggroup/ <i>thing-group-name</i> arn:aws:iot: <i>region:account-id</i> :thing/ <i>thing-name</i>
物聯網:ReplaceTopic規則	ReplaceTopic規則	arn:aws:iot: <i>region:account-id</i> :rule/ <i>rule-name</i>
物聯網:SearchIndex	SearchIndex	arn:aws:iot: <i>region:account-id</i> :index/ <i>index-id</i>
物聯網:SetDefault授權者	SetDefault授權者	arn:aws:iot: <i>region:account-id</i> :authorizer/ <i>authorizer-function-name</i>
物聯網:SetDefaultPolicyVersion	SetDefaultPolicyVersion	arn:aws:iot: <i>region:account-id</i> :policy/ <i>policy-name</i>
物聯網:SetLogging選項	SetLogging選項	*
物聯網:設定 V2 LoggingLevel	第 2 集 LoggingLevel	*

政策動作	AWS IoT API	資源
物聯網:設定 V2 LoggingOptions	第 2 集 LoggingOptions	*
物聯網:Start ThingRegistrationTask	StartThingRegistrationTask	無
物聯網:StopThingRegistrationTask	StopThingRegistrationTask	無
物聯網:TestAuthorization	TestAuthorization	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
物聯網:TestInvoke授權者	TestInvoke授權者	無
物聯網:TransferCertificate	TransferCertificate	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
物聯網:UpdateAuthorizer	UpdateAuthorizer	arn:aws:iot: <i>region:account-id</i> :authorizerfunction/ <i>authorizer-function-name</i>
iot:UpdateCACertificate	UpdateCACertificate	arn:aws:iot: <i>region:account-id</i> :cacert/ <i>cert-id</i>
物聯網:UpdateCertificate	UpdateCertificate	arn:aws:iot: <i>region:account-id</i> :cert/ <i>cert-id</i>
物聯網 : UpdateEvent配置	UpdateEvent配置	無
物聯網:UpdateIndexing配置	UpdateIndexing配置	無
物聯網:UpdateRole別名	UpdateRole別名	arn:aws:iot: <i>region:account-id</i> :rolealiases/ <i>role-alias-name</i>

政策動作	AWS IoT API	資源
物聯網:UpdateThing	UpdateThing	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>
物聯網:UpdateThing組	UpdateThing集團	arn:aws:iot: <i>region</i> : <i>account-id</i> :thinggroup/ <i>thing-group-name</i>
物聯網:UpdateThingGroupsFor事	UpdateThingGroupsFor事	arn:aws:iot: <i>region</i> : <i>account-id</i> :thing/ <i>thing-name</i>

如需 ARN 格式的詳細資訊，請參閱 [Amazon 資源名稱 \(ARN\)](#) 和 [AWS 服務命名空間](#)。

某些 AWS IoT 動作 (例如用來建立資源的動作) 無法在特定資源上執行。在這些情況下，您必須使用萬用字元 (\*)。

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

若要查看 AWS IoT 資源類型及其 ARN 的清單，請參閱《IAM 使用者指南》AWS IoT 中的「[定義資源](#)」。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 [AWS IoT 定義的動作](#)。

### Device Advisor 資源

若要定義 AWS IoT 裝置建議程式 IAM 政策的資源層級限制，請針對套件定義和套件執行使用下列資源 ARN 格式。

#### 套件定義資源 ARN 格式

```
arn:aws:iotdeviceadvisor:region:account-id:suitedefinition/suite-definition-id
```

#### 套件執行資源 ARN 格式

```
arn:aws:iotdeviceadvisor:region:account-id:suiterun/suite-definition-id/suite-run-id
```

## 條件索引鍵

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用[條件運算子](#)的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵，AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯 OR 運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的[IAM 政策元素：變數和標籤](#)。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的[AWS 全域條件內容金鑰](#)。

AWS IoT 定義了它自己的一組條件鍵，並且還支持使用一些全局條件鍵。若要查看所有 AWS 全域條件金鑰，請參閱 IAM 使用者指南中的[AWS 全域條件內容金鑰](#)。

### AWS IoT 條件鍵

AWS IoT 條件鍵	描述	Type
<code>aws:RequestTag/\${tag-key}</code>	在使用者對 AWS IoT 提出的請求中存在的標籤金鑰。	字串
<code>aws:ResourceTag/\${tag-key}</code>	貼附至 AWS IoT 資源之標籤的標籤關鍵元件。	字串
<code>aws:TagKeys</code>	與請求中資源相關聯的所有標籤鍵名稱清單。	字串

若要查看 AWS IoT 條件金鑰清單，請參閱《IAM 使用者指南》AWS IoT 中的[條件金鑰](#)。若要瞭解您可以使用條件索引鍵的動作和資源，請參閱[動作定義者 AWS IoT](#)。

## 範例

若要檢視以 AWS IoT 身為基礎的原則範例，請參閱。[AWS IoT 以識別為基礎的原則範例](#)

## AWS IoT 資源型政策

以資源為基礎的策略是 JSON 政策文件，指定指定的主體可以在 AWS IoT 資源上以及在何種情況下執行的動作。

AWS IoT 不支援以 IAM 資源為基礎的政策。但是，它確實支持 AWS IoT 以資源為基礎的策略。如需詳細資訊，請參閱[AWS IoT Core 政策](#)。

## 以 AWS IoT 標籤為基礎的授權

您可以將標籤附加至 AWS IoT 資源，或將要求中的標籤傳遞給 AWS IoT。如需根據標籤控制存取，請使用 `iot:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的[條件元素](#)中，提供標籤資訊。如需詳細資訊，請參閱[搭配 IAM 政策使用標籤](#)。如需標記 AWS IoT 資源的更多資訊，請參閱[標記您的 AWS IoT 資源](#)。

若要檢視身分型原則範例，以根據該資源上的標籤來限制存取資源，請參閱[根據標籤檢視 AWS IoT 資源](#)。

## AWS IoT IAM 角色

[IAM 角色](#)是您 AWS 帳戶 內部具有特定許可的實體。

### 使用臨時登入資料 AWS IoT

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫[AssumeRole](#)或[GetFederation權杖](#)等 AWS STS API 作業來取得臨時安全登入資料。

AWS IoT 支援使用臨時認證。

### 服務連結角色

[服務連結角色](#)可讓 AWS 服務存取其他服務中的資源，以代表您完成動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

AWS IoT 不支援服務連結角色。



## 服務角色

此功能可讓服務代表您擔任[服務角色](#)。此角色可讓服務存取其他服務中的資源，以代表您完成動作。服務角色會出現在您的 IAM 帳戶中，且由該帳戶所擁有。這表示 IAM 管理員可以變更此角色的許可。不過，這樣可能會破壞此服務的功能。

## AWS IoT 以識別為基礎的原則範例

根據預設，IAM 使用者和角色不具備建立或修改 AWS IoT 資源的許可。他們也無法使用 AWS Management Console AWS CLI、或 AWS API 執行工作。IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 操作的所需許可。管理員接著必須將這些政策連接至需要這些許可的使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱 IAM 使用者指南中的[在 JSON 索引標籤上建立政策](#)。

### 主題

- [政策最佳實務](#)
- [使用 AWS IoT 主控台](#)
- [允許使用者檢視他們自己的許可](#)
- [根據標籤檢視 AWS IoT 資源](#)
- [根據標記檢視 AWS IoT 裝置建議程式資源](#)

## 政策最佳實務

以身分識別為基礎的政策會決定某人是否可以建立、存取或刪除您帳戶中的 AWS IoT 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始將權限授與使用者和工作負載，請使用可授與許多常見使用案例權限的 AWS 受管理原則。它們可用在您的 AWS 帳戶。建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的[AWS 受管政策](#)或[任務職能的 AWS 受管政策](#)。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的[IAM 中的政策和許可](#)。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動

作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 [IAM JSON 政策元素：條件](#)。

- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 [IAM Access Analyzer 政策驗證](#)。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 [IAM 使用者指南](#) 中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 [IAM 安全最佳實務](#)。

## 使用 AWS IoT 主控台

若要存取 AWS IoT 主控台，您必須擁有最少一組權限。這些權限必須允許您列出和檢視有關 AWS 帳戶。AWS IoT 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

若要確保這些實體仍可使用 AWS IoT 主控台，請同時將下列 AWS 受管理的原則附加至實體：AWSIoTFullAccess 如需詳細資訊，請參閱《IAM 使用者指南》中的 [新增許可到使用者](#)。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。反之，只需允許存取符合您嘗試執行之 API 操作的動作就可以了。

## 允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",

```

```

        "iam:GetUser"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
},
{
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
    ],
    "Resource": "*"
}
]
}

```

## 根據標籤檢視 AWS IoT 資源

您可以在身分型政策中使用條件，以根據標籤控制存取 AWS IoT 資源。此範例會示範如何建立政策，允許檢視物件。但是，只有在物件標籤 Owner 的值是該使用者的使用者名稱時，才會授予該許可。此政策也會授予在主控台完成此動作的必要許可。

```

{
    "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}"}
    }
}
]
}

```

您可以將此政策連接到您帳戶中的 IAM 使用者。如果名為的使用者 richard-roe 嘗試檢視 AWS IoT 帳單群組，則必須標記帳單群組 Owner=richard-roe 或 owner=richard-roe。否則，他便會被拒絕存取。條件標籤鍵 Owner 符合 Owner 和 owner，因為條件索引鍵名稱不區分大小寫。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM JSON 政策元素：條件](#)。

## 根據標記檢視 AWS IoT 裝置建議程式資源

您可以在身分類型政策中使用條件，根據標籤來控制存取 AWS IoT Device Advisor 資源。以下範例顯示如何建立政策，允許檢視特定套件定義。不過，只在套件定義標籤已將 SuiteType 設定為值 MQTT 時，才會授予許可。此政策也會授予在主控台完成此動作的必要許可。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewSuiteDefinition",
      "Effect": "Allow",
      "Action": "iotdeviceadvisor:GetSuiteDefinition",
      "Resource": "arn:aws:iotdeviceadvisor:*:*:suitedefinition/*",
      "Condition": {
        "StringEquals": {"aws:ResourceTag/SuiteType": "MQTT"}
      }
    }
  ]
}

```

## AWS 受管理的政策 AWS IoT

若要新增使用者、群組和角色的權限，使用 AWS 受管理的原則比自己撰寫原則更容易。建立 [IAM 客戶受管政策](#) 需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。若要快速開始使用，您可以使用我們的 AWS 受管政策。這些政策涵蓋常見的使用案例，並可在您的 AWS 帳戶中使用。如需 AWS 受管政策的詳細資訊，請參閱 IAM 使用者指南中的 [AWS 受管政策](#)。

AWS 服務會維護和更新 AWS 受管理的策略。您無法變更 AWS 受管理原則中的權限。服務偶爾會在 AWS 受管政策中新增其他許可以支援新功能。此類型的更新會影響已連接政策的所有身分識別 (使用者、群組和角色)。當新功能啟動或新操作可用時，服務很可能會更新 AWS 受管政策。服務不會從 AWS 受管理的政策移除權限，因此政策更新不會破壞您現有的權限。

此外，還 AWS 支援跨多個服務之工作職能的受管理原則。例如，ReadOnly存取 AWS 管理原則會提供所有 AWS 服務和資源的唯讀存取權。當服務啟動新功能時，AWS 會為新的操作和資源新增唯讀許可。如需任務職能政策的清單和說明，請參閱 IAM 使用者指南中[有關任務職能的 AWS 受管政策](#)。

#### Note

AWS IoT 適用於 AWS IoT 和 IAM 政策。此主題僅討論 IAM 政策，此政策定義控制平面和資料平面 API 操作的政策動作。另請參閱[AWS IoT Core 政策](#)。

## AWS 受管理的策略：AWSIoTConfigAccess

您可將 AWSIoTConfigAccess 政策連接到 IAM 身分。

此政策授予相關的身分許可，允許存取所有 AWS IoT 組態操作。此政策會影響資料處理和儲存。若要在中檢視此原則 AWS Management Console，請參閱[AWSIoTConfigAccess](#)。

### 許可詳細資訊

此政策包含以下許可。

- `iot`— 擷取 AWS IoT 資料並執行 IoT 設定動作。

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

## AWS 受管理的策略：AWSIoTConfigReadOnlyAccess

您可將 AWSIoTConfigReadOnlyAccess 政策連接到 IAM 身分。

此政策授予相關的身分許可，允許以唯讀方式存取所有 AWS IoT 組態操作。若要在中檢視此原則 AWS Management Console，請參閱 [AWSIoTConfigReadOnlyAccess](#)。

### 許可詳細資訊

此政策包含以下許可。

- iot – 執行 IoT 組態操作的唯獨操作。

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

```

## AWS 受管理的策略：AWSIoTDataAccess

您可將 AWSIoTDataAccess 政策連接到 IAM 身分。

此原則會授與允許存取所有 AWS IoT 資料作業的相關身分識別權限。資料操作則可透過 MQTT 或 HTTP 通訊協定傳送資料。若要在 AWS Management Console 中檢視此政策，請參閱 [AWSIoTDataAccess](#)。

### 許可詳細資訊

此政策包含以下許可。

- `iot`— 檢索 AWS IoT 數據並允許對 AWS IoT 消息傳遞操作的完全訪問權限。

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

### AWS 受管理的策略：AWSIoTFullAccess

您可將 `AWSIoTFullAccess` 政策連接到 IAM 身分。

此政策授予相關的身分許可，允許完整存取所有 AWS IoT 組態和簡訊操作。若要在中檢視此原則 AWS Management Console，請參閱 [AWSIoTFullAccess](#)。

## 許可詳細資訊

此政策包含以下許可。

- `iot`-檢索 AWS IoT 數據並允許對 AWS IoT 配置和消息傳遞操作的完全訪問權限。
- `iotjobsdata`— 擷取 AWS IoT 工作資料，並允許完全存取 AWS IoT 工作資料平面 API 作業。

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

## AWS 受管理的策略：AWSIoTLogging

您可將 AWSIoTLogging 政策連接到 IAM 身分。

此政策授予關聯的身分許可，允許存取建立 Amazon CloudWatch 日誌群組並將日誌串流到群組。此原則會附加至您的 CloudWatch 記錄角色。若要在中檢視此原則 AWS Management Console，請參閱[AWSIoTLogging](#)。

## 許可詳細資訊

此政策包含以下許可。

- `logs`-檢索 CloudWatch 日誌。此外，還允許建立 CloudWatch 記錄群組，並將記錄串流至群組。

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

## AWS 受管理的策略：AWSIoTOTAUpdate

您可將 AWSIoTOTAUpdate 政策連接到 IAM 身分。

此原則會授與相關聯的身分權限，允許存取建立 AWS IoT 工作、AWS IoT 程式碼簽署工作，以及描述 AWS 程式碼簽署者工作。若要在中檢視此原則 AWS Management Console，請參閱 [AWSIoTOTAUpdate](#)。

### 許可詳細資訊

此政策包含以下許可。

- `iot`— 建立 AWS IoT 工作和程式碼簽章工作。
- `signer`— 執行程 AWS 式碼簽署者工作的建立。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iot:CreateJob",
      "signer:DescribeSigningJob"
    ],
    "Resource": "*"
  }
}
```

## AWS 受管理的策略：AWSIoTRuleActions

您可將 AWSIoTRuleActions 政策連接到 IAM 身分。

此原則會授與相關聯的身分權限，允許存取 AWS IoT 規則動作中支援的所有 AWS 服務身分識別權限。若要在中檢視此原則 AWS Management Console，請參閱[AWSIoTRuleActions](#)。

### 許可詳細資訊

此政策包含以下許可。

- `iot` - 執行發佈規則動作訊息的動作。
- `dynamodb` - 將訊息插入 DynamoDB 表格，或將訊息拆分至 DynamoDB 表格中多個欄。
- `s3` - 將物件存放在 Amazon S3 儲存貯體中。
- `kinesis` - 向 Amazon Kinesis 串流物件發送訊息。
- `firehose`-在 Firehose 串流物件中插入記錄。
- `cloudwatch`-更改 CloudWatch 警報狀態或將消息數據發送到 CloudWatch 指標。
- `sns` - 執行使用 Amazon SNS 發佈通知的操作。此作業的範圍是 AWS IoT SNS 主題。
- `sqs` - 插入要新增至 SQS 佇列的訊息。
- `es`-向 OpenSearch 服務服務發送消息。

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

## AWS 受管理的策略：AWSIoTThingsRegistration

您可將 AWSIoTThingsRegistration 政策連接到 IAM 身分。

此政策授予相關的身分許可，允許使用 StartThingRegistrationTask API 大量註冊物件。此政策會影響資料處理和儲存。若要在中檢視此原則 AWS Management Console，請參閱 [AWSIoTThingsRegistration](#)。

### 許可詳細資訊

此政策包含以下許可。

- `iot` - 進行大量註冊時，執行建立物件並附加政策和憑證的動作。

```
{
```



```
"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": [
      "*"
    ]
  }
]
```

## AWS IoTAWS 受管理策略的更新

檢視 AWS IoT 自此服務開始追蹤這些變更以來的 AWS 受管理策略更新詳細資料。如需有關此頁面變更的自動警示，請訂閱「AWS IoT 文件歷史記錄」頁面上的 RSS 摘要。

變更	描述	日期
<a href="#">AWSIoTFullAccess</a> – 更新現有政策	<p>AWS IoT 增加了新的權限，以允許用戶使用 HTTP 協議訪問 AWS IoT 作業數據平面 API 操作。</p> <p>新的 IAM 政策前綴可為您提供更精細的存取控制 <code>iotjobsdata:</code>，以存取 AWS IoT Jobs 資料平面端點。對於控制平面 API 操作，請依舊使用 <code>iot:</code> 字首。如需詳細資訊，請參閱 <a href="#">AWS IoT Core HTTPS 通訊協定的政策</a>。</p>	2022 年 5 月 11 日
AWS IoT 開始追蹤變更	AWS IoT 開始追蹤其 AWS 受管理策略的變更。	2022 年 5 月 11 日

## 疑難排解 AWS IoT 身分和存取

使用下列資訊可協助您診斷和修正使用和 IAM 時可能會遇到的 AWS IoT 常見問題。

### 主題

- [我沒有執行操作的授權 AWS IoT](#)
- [我沒有授權執行 iam : PassRole](#)
- [我想允許我以外的人訪 AWS 帳戶 問我的 AWS IoT 資源](#)

### 我沒有執行操作的授權 AWS IoT

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在 IAM 使用者 mateojackson 嘗試使用主控台檢視物件資源的詳細資訊，但並無 `iot:DescribeThing` 許可權限時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
iot:DescribeThing on resource: MyIoTThing
```

於此情況下，必須更新 mateojackson 使用者的政策，允許使用 `iot:DescribeThing` 動作存取物件資源。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

### 使用 AWS IoT 裝置建議

如果您使用 AWS IoT Device Advisor，當使用者 mateojackson 嘗試使用主控台來檢視有關套件定義的詳細資料，但沒有 `iotdeviceadvisor:GetSuiteDefinition` 權限時，就會發生下列範例錯誤。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
iotdeviceadvisor:GetSuiteDefinition on resource: MySuiteDefinition
```

於此情況下，必須更新 mateojackson 使用者的政策，允許使用 `iotdeviceadvisor:GetSuiteDefinition` 動作存取 `MySuiteDefinition` 資源。

### 我沒有授權執行 iam : PassRole

如果您收到錯誤，告知您未獲授權執行 `iam:PassRole` 動作，您的政策必須更新，允許您將角色傳遞給 AWS IoT。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

名為 marymajor 的 IAM 使用者嘗試使用主控台在 AWS IoT 中執行動作時，發生下列範例錯誤。但是，動作要求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

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

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

## 我想允許我以外的人訪 AWS 帳戶 問我的 AWS IoT 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要瞭解是否 AWS IoT 支援這些功能，請參閱[如何與 IAM AWS IoT 搭配使用](#)。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱 [IAM 使用者指南中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限](#)。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的[提供第三方 AWS 帳戶 擁有的存取權](#)。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的[將存取權提供給在外部進行身分驗證的使用者 \(聯合身分\)](#)。
- 若要了解跨帳戶存取使用角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 [IAM 中的跨帳戶資源存取](#)。

## 記錄和監控

監控是維持 AWS 解決方案的可靠性、可用性和效能的 AWS IoT 重要組成部分。您應該從 AWS 解決方案的所有部分收集監視資料，以便在發生多點失敗時更輕鬆地偵錯。如需記錄和監視程序的相關資訊，請參閱 [監控 AWS IoT](#)

### 監控工具

AWS 提供您可用來監視的工具 AWS IoT。您可設定部分這些工具以為您執行監控工作。部分工具將需要手動操作。建議您盡可能自動化監控任務。

### 自動化監控工具

您可以使用以下自動監視工具來觀看 AWS IoT 和報告出現問題時：

- Amazon A CloudWatch lar ps — 觀看您指定期間內的單一指標，並根據指定臨界值在多個時段內相對於指定閾值的指標值執行一或多個動作。動作是傳送至亞馬遜簡單通知服務 (Amazon SNS) 主題

或 Amazon EC2 Auto Scaling 政策的通知。CloudWatch 警報不會僅僅因為它們處於特定狀態而叫用動作。狀態必須已變更，且在指定的期間數內維持此狀態。如需詳細資訊，請參閱 [使用 Amazon 監控 AWS IoT 警示和指標 CloudWatch](#)。

- Amazon CloudWatch 日誌 — 監控、存放和存取來自 AWS CloudTrail 或其他來源的日誌檔。Amazon CloudWatch Logs 也可讓您查看 AWS IoT 裝置顧問測試案例採取的重要步驟、產生的事件和 MQTT 訊息從您的裝置傳送或測試執行 AWS IoT Core 期間。這些日誌可讓您在裝置上偵錯並採取更正動作。如需詳細資訊，請參閱 [AWS IoT 使用 CloudWatch 記錄監控](#) 如需使用 Amazon 的詳細資訊 CloudWatch，請參閱 Amazon 使用 CloudWatch 者指南中的 [監控日誌檔](#)。
- Amazon E CloudWatch vents — 匹配事件並將其路由到一個或多個目標函數或串流，以進行變更、擷取狀態資訊並採取糾正措施。有關更多信息，請參閱 [Amazon 用 CloudWatch 戶指南中的 Amazon CloudWatch 事件是什麼](#)。
- AWS CloudTrail 記錄監控 — 在帳戶之間共用記錄檔、即時監控記 CloudTrail 錄檔案，方法是將記錄檔傳送至 CloudWatch 記錄檔、使用 Java 撰寫記錄處理應用程式，以及驗證您的記錄檔在傳送之後未變更 CloudTrail。若要取得更多資訊，請參閱 [使用記錄 AWS IoT API 呼叫 AWS CloudTrail](#) 《使用指南》中的〈AWS CloudTrail 使用 CloudTrail [記錄檔](#)〉，另請參閱〈

## 手動監控工具

監視 AWS IoT 的另一個重要部分是手動監視 CloudWatch 警報未涵蓋的項目。AWS IoT CloudWatch、和其他 AWS 服務主控台儀表板可提供您 AWS 環境狀態的 at-a-glance 檢視。我們建議您同時檢查記錄檔 AWS IoT。

- AWS IoT 儀表板顯示：
  - 憑證授權機構憑證
  - 憑證
  - 政策
  - 規則
  - 實物
- CloudWatch 主頁顯示：
  - 目前警示與狀態。
  - 警示與資源的圖形。
  - 服務運作狀態。

您可以使用 CloudWatch 來執行下列動作：

- 建立 [自訂儀表板](#) 來監控您關心的服務。

- 用於疑難排解問題以及探索驅勢的圖形指標資料。
- 搜尋並瀏覽所有資 AWS 源指標。
- 建立與編輯要通知發生問題的警示。

## AWS IoT 核心的合規性驗證

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱[AWS 服務 遵循規範計劃](#)方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱[AWS 規範計劃](#)。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱[下載中的報告中的 AWS Artifact](#)。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- [安全性與合規性快速入門指南](#) — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- [在 Amazon Web Services 上架構 HIPAA 安全性與合規性](#) — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

### Note

並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 [HIPAA 資格服務參照](#)。

- [AWS 合規資源](#) — 此工作簿和指南集合可能適用於您的產業和所在地。
- [AWS 客戶合規指南](#) — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- [使用 AWS Config 開發人員指南中的規則評估資源](#) — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- [AWS Security Hub](#) — 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 [Security Hub controls reference](#)。
- [Amazon GuardDuty](#) — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。GuardDuty 可協助您因應各種合規性需求，例如 PCI DSS，滿足特定合規性架構所規定的入侵偵測需求。

- [AWS Audit Manager](#)— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

## AWS IoT 核心的韌性

AWS 全球基礎架構是圍繞 AWS 區域 s 和可用區域建立的。AWS 區域 s 提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域 s 和可用區域的詳細資訊，請參閱[AWS 全域基礎結構](#)。

AWS IoT Core 將您裝置的相關資訊儲存在裝置登錄中。它也會存放憑證授權機構憑證、裝置憑證和裝置影子資料。發生硬體或網路失敗時，此資料會自動跨可用區域複寫，但不會跨區域複寫。

AWS IoT Core 更新裝置登錄時，會發佈 MQTT 事件。您可以使用這些訊息來備份您的登錄資料，並將其儲存在某處，例如 DynamoDB 資料表。您必須負責儲存為您或您自己建立的憑證。AWS IoT Core 裝置陰影會儲存裝置的狀態資料，並可在裝置重新連線時重新傳送。AWS IoT 裝置建議程式會儲存測試套件組態的相關資訊。發生硬體或網路失敗時，此資料會自動複寫。

AWS IoT Core 資源是特定於區域的，除非您特別這樣做，AWS 區域 否則不會複製。

如需安全性最佳實務的相關資訊，請參閱 [安全性最佳做法 AWS IoT Core](#)。

## AWS IoT Core 與介面 VPC 端點搭配使用

您可以使用[介面 VPC 端點 AWS IoT Core](#)，在虛擬私有雲 (VPC) 中建立 IoT 資料端點。介面 VPC 私人雲端端點的支援是一種 AWS 技術 AWS PrivateLink，您可以使用私有 IP 位址存取執 AWS 行於上的服務。如需詳細資訊，請參閱 [Amazon Virtual Private Cloud](#)。

若要將遠端網路 (例如公司網路) 上的現場裝置連接到 Amazon VPC，請參閱網路到 [Amazon VPC 連線對照表](#) 中列出的選項。

### 目錄

- [為 AWS IoT Core 資料平面建立 VPC 端點](#)
- [建立 AWS IoT Core 認證提供者的 VPC 端點](#)
- [建立 Amazon VPC 界面端點](#)

- [配置私有託管區域](#)
- [控制對 VPC 端點 AWS IoT Core 的存取](#)
- [限制](#)
- [使用擴展 VPC 端點 AWS IoT Core](#)
- [搭配使用自訂網域與 VPC 端點](#)
- [下列項目的 VPC 端點的可用性 AWS IoT Core](#)

## 為 AWS IoT Core 資料平面建立 VPC 端點

您可以為 AWS IoT Core 資料平面 API 建立 VPC 端點，以將裝置連線至 AWS IoT 服務和其他 AWS 服務。若要開始使用 VPC 端點，請[建立介面 VPC 端點](#)並選取 AWS IoT Core 為服務。AWS 如果您正在使用 CLI，首先呼叫 [describe-vpc-端點服務](#)，以確保您選擇的是可用區域，其中存在於您的特定。AWS IoT Core AWS 區域例如，在 us-east-1 中，此命令將如下所示：

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.data
```

### Note

已停用自動建立 DNS 記錄的 VPC 功能。若要連接這些端點，您必須手動建立私有 DNS 記錄。如需有關私有 VPC DNS 記錄的詳細資訊，請參閱[介面端點的私有 DNS](#)。如需 AWS IoT Core VPC 限制的詳細資訊，請參閱[限制](#)。

如果要將 MQTT 用戶端連線到 VPC 端點介面：

- 您必須在附加至 VPC 的私有託管區域中手動建立 DNS 記錄。若要開始使用，請參閱[建立私有託管區域](#)。
- 在您的私有託管區域中，為 VPC 端點的每個 elastic network interface IP 建立別名記錄。如果多個 VPC 端點有多個網路介面 IP，請在所有加權記錄中建立具有相同權重的加權 DNS 記錄。在 [說明] 欄位中按 VPC 端點識別碼篩選時，可從[DescribeNetwork介面](#) API 呼叫取得這些 IP 位址。

請參閱下面的詳細說明，以[建立 Amazon VPC 介面端點](#)和為 AWS IoT Core 資料平面[設定私有託管區域](#)。



## 建立 AWS IoT Core 認證提供者的 VPC 端點

您可以為認證提供者建立 VPC 端點，以使用用戶端 AWS IoT Core 憑證型驗證連線裝置，並取得「[AWS 簽章版本 4](#)」格式的臨時認 AWS 證。若要開始使用 AWS IoT Core 認證提供者的 VPC 端點，請執行 [create-vpc-端點CLI 命令以建立介面 VPC 端點](#)，然後選取認證提供者做為服務。AWS IoT Core AWS 為了確保您選擇的 AWS IoT Core 是可用區域，其中存在於您的特定 AWS 區域，您第一次運行 [describe-vpc- endpoint 服務命令](#)。例如，在 us-east-1 中，此命令將如下所示：

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.iot.credentials
```

### Note

已停用自動建立 DNS 記錄的 VPC 功能。若要連接這些端點，您必須手動建立私有 DNS 記錄。如需有關私有 VPC DNS 記錄的詳細資訊，請參閱[介面端點的私有 DNS](#)。如需 AWS IoT Core VPC 限制的詳細資訊，請參閱[限制](#)。

如果要將 HTTP 用戶端連線到 VPC 端端點介面：

- 您必須在附加至 VPC 的私有託管區域中手動建立 DNS 記錄。若要開始使用，請參閱[建立私有託管區域](#)。
- 在您的私有託管區域中，為 VPC 端點的每個 elastic network interface IP 建立別名記錄。如果多個 VPC 端點有多個網路介面 IP，請在所有加權記錄中建立具有相同權重的加權 DNS 記錄。在 [說明] 欄位中按 VPC 端點識別碼篩選時，可從[DescribeNetwork介面](#) API 呼叫取得這些 IP 位址。

請參閱下面的詳細說明，以[建立 Amazon VPC 介面端點](#)和為 AWS IoT Core 登入資料提供者[設定私有託管區域](#)。

## 建立 Amazon VPC 介面端點

您可以建立介面 VPC 端點，以連線至提供支援的 AWS 服務。AWS PrivateLink使用下列程序建立連線至 AWS IoT Core 資料平面或 AWS IoT Core 認證提供者的介面 VPC 端點。如需詳細資訊，請參閱[使用介面 VPC 端點存取 AWS 服務](#)。

**Note**

為資料平面和 AWS IoT Core 登入 AWS IoT Core 資料提供者建立 Amazon VPC 界面端點的程序類似，但是您必須進行端點特定變更才能使連線正常運作。

使用 VPC 端點主控台建立介面 [VPC](#) 端點

1. 瀏覽至 [VPC](#) Endpoint 主控台，在左側功能表的「虛擬私有雲」下，選擇「端點」，然後選擇「建立端點」。
2. 在「建立端點」頁面中，指定下列資訊。

- 選擇服務類別的 AWS 服務。
- 針對 Service Name (服務名稱)，輸入關鍵字 `iot` 進行搜尋。在顯示的 `iot` 服務清單中，選擇端點。

如果您為 AWS IoT Core 資料平面建立 VPC 端點，請為您的區域選擇 AWS IoT Core 資料平面 API 端點。端點的格式為 `com.amazonaws.region.iot.data`。

如果您為 AWS IoT Core 認證提供者建立 VPC 端點，請為您的區域選擇 AWS IoT Core 認證提供者端點。端點的格式為 `com.amazonaws.region.iot.credentials`。

**Note**

中國地區資 AWS IoT Core 料平面的服務名稱格式為 `cn.com.amazonaws.region.iot.data`。中國地區不支援為 AWS IoT Core 憑證提供者建立 VPC 端點。

- 針對 VPC 和 Subnets (子網路)，選擇要在其中建立端點的 VPC，以及要在其中建立端點網路的可用區域 (AZ)。
  - 針對 Enable DNS name (啟用 DNS 名稱)，確定未選取 Enable for this endpoint (為此端點啟用)。AWS IoT Core 資料平面和 AWS IoT Core 憑證提供者都不支援私有 DNS 名稱。
  - 針對 Security group (安全群組)，選擇要與端點網路介面建立關聯的安全群組。
  - 您可以選擇性地新增或移除標籤。標籤是您用來與端點建立關聯的名稱值對。
3. 若要建立 VPC 端點，請選擇 Create endpoint (建立端點)。

建立端點後，在 AWS PrivateLink 端點的 [詳細資料] 索引標籤中，您會看到 DNS 名稱清單。您可以使用您在本節中建立的其中一個 DNS 名稱來[設定您的私有託管區域](#)。

## 配置私有託管區域

您可以使用您在上一節中建立的其中一個 DNS 名稱來設定您的私有託管區域。

對於 AWS IoT Core 資料平面

DNS 名稱必須是您的網域組態名稱或 IoT:Data-ATS 端點。範例 DNS 名稱可以是：`xxx-ats.data.iot.region.amazonaws.com`。

對於 AWS IoT Core 憑證提供者

DNS 名稱必須是您的 `iot:CredentialProvider` 端點。範例 DNS 名稱可以是：`xxxx.credentials.iot.region.amazonaws.com`。

### Note

為 AWS IoT Core 資料平面和 AWS IoT Core 認證提供者設定私有託管區域的程序類似，但是您必須進行端點特定變更才能使連線正常運作。

## 建立私有託管區域

若要使用 Route 53 主控台建立私有託管區域

1. 導覽至 [Route 53](#) Hosted zones (託管區域) 主控台，然後選擇 Create hosted zone (建立託管區域)。
2. 在 Create hosted zone (建立託管區域) 頁面中，指定下列資訊。
  - 在「網域名稱」中，輸入您 `iot:Data-ATS` 或端點的 `iot:CredentialProvider` 端點位址。下列 AWS CLI 命令顯示如何透過公用網路取得端點：`aws iot describe-endpoint --endpoint-type iot:Data-ATS`、或 `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`。

### Note

如果您使用自訂網域，請參閱[搭配使用自訂網域與 VPC 端點](#)。AWS IoT Core 認證提供者不支援自訂網域。

- 針對 Type (類型)，選擇 Private Hosted Zone (私有託管區域)。
  - 或者，您可以新增或移除要與託管區域建立關聯的標籤。
3. 若要建立私有託管區域，請選擇 Create hosted zone (建立託管區域)。

如需詳細資訊，請參閱[建立私有託管區域](#)。

## 建立記錄

在建立了私有託管區域之後，您可以建立記錄，告訴 DNS 您想要流量路由至該網域的方式。

### 若要建立記錄

1. 在顯示的託管區域清單，選擇您先前建立的私有託管區域，然後選擇 Create record (建立記錄)。
2. 使用精靈方法來建立記錄。如果主控台呈現 Quick create (快速建立) 方法，請選擇 Switch to wizard (切換至精靈)。
3. 為 Routing policy (路由政策) 選擇 Simple Routing (簡易路由)，然後選擇 Next (下一步)。
4. 在 Configure records (設定記錄) 頁面中，選擇 Define simple record (定義簡易記錄)。
5. 在 Define simple record (定義簡易記錄) 頁面中：
  - 在「記錄名稱」中，輸入 `iot:Data-ATS` 端點或 `iot:CredentialProvider` 端點。這必須與私有託管區域名稱相同。
  - 針對 Record type (紀錄類型)，將值保留為 `A - Routes traffic to an IPv4 address and some AWS resources`。
  - 針對 Value/Route traffic to (值/路由流量至)，選擇 Alias to VPC endpoint (VPC 端點的別名)。接著，選擇您的 Region (區域)，然後從顯示的端點清單中選擇您先前建立的端點，如[???](#)所述。
6. 選擇 Define simple record (定義簡易記錄) 來建立您的記錄。

## 控制對 VPC 端點 AWS IoT Core 的存取

您可以使用 VPC [條件內容](#) 金鑰，將裝置存取限制 AWS IoT Core 為僅允許透過 VPC 端點。AWS IoT Core 支援下列 VPC 相關內容金鑰：

- [SourceVpc](#)
- [SourceVpce](#)
- [VPC SourceIp](#)

**Note**

AWS IoT Core 不支援虛擬私人雲端端點 VPC 點政策。

例如，以下策略授與使用符合物名稱的 AWS IoT Core 用戶端 ID 連線的權限，以及發佈至以物件名稱為前置詞的任何主題，視連線至具有特定 VPC Endpoint ID 之 VPC 端點的裝置而定。此政策會拒絕與公有 IoT 資料端點的連線嘗試。

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

## 限制

VPC 端點目前僅支援[AWS IoT Core 資料端點](#)和[AWS IoT Core 認證提供者端點](#)。

### IoT 資料虛擬私人雲端端點的限制

本節涵蓋 IoT 資料虛擬私人雲端端點的限制。

- MQTT 持續作用期間限制為 230 秒。保持活躍時間超過此時間將自動減少到 230 秒。
- 每個 VPC 端點支援總共 100,000 個並行連網裝置。如果您需要更多連線，請參閱 [使用擴展 VPC 端點 AWS IoT Core](#)。
- VPC 端點僅支援 IPv4 流量。
- VPC 端點只會為 [ATS 憑證](#) 提供服務，但自訂網域除外。
- 不支援 [VPC 端點原則](#)。
- 針對為 AWS IoT Core 資料平面建立的 VPC 端點，AWS IoT Core 不支援使用區域或地區公用 DNS 記錄。

### 憑證提供者端點的限制

本節涵蓋認證提供者 VPC 端點的限制。

- VPC 端點僅支援 IPv4 流量。
- VPC 端點將僅提供 [ATS 憑證](#)。
- 不支援 [VPC 端點原則](#)。
- 認證提供者端點不支援自訂網域。
- 對於為 AWS IoT Core 認證提供者建立的 VPC 端點，AWS IoT Core 不支援使用區域或地區公用 DNS 記錄。

### 使用擴展 VPC 端點 AWS IoT Core

AWS IoT Core 透過單一介面端點，介面 VPC 端點限制為 100,000 個已連線裝置。如果您的使用案例要求更多並行連線到代理程式，則我們建議使用多個 VPC 端點，並跨您的介面端點手動路由裝置。建立私有 DNS 記錄以將流量路由到 VPC 端點時，請確保建立的加權記錄數量與 VPC 端點數量一樣多，以將流量分配到多個端點。

## 搭配使用自訂網域與 VPC 端點

如果您想要將自訂網域與 VPC 端點搭配使用，則必須在私有託管區域中建立自訂網域名稱記錄，並在 Route53 中建立路由記錄。如需詳細資訊，請參閱[建立私有託管區域](#)。

### Note

自訂網域僅支援資 AWS IoT Core 料端點。

## 下列項目的 VPC 端點的可用性 AWS IoT Core

AWS IoT Core 所有[AWS IoT Core 支援的區域](#)均提供介面 VPC 端點。AWS IoT Core 中國地區和 AWS IoT Core AWS GovCloud (US) Regions 認證提供者的介面 VPC 端點不支援。

## 基礎結構安全 AWS IoT

作為受管服務的集合，AWS IoT 受 [Amazon Web Services : 安 AWS 全流程概觀白皮書中所述的全球網路安全](#) 程序保護。

您可以使用 AWS 已發佈的 API 呼叫透 AWS IoT 過網路進行存取。用戶端必須支援 Transport Layer Security (TLS) 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。如需詳細資訊，請參閱 [中的運輸安全 AWS IoT Core](#)。

請求必須使用存取金鑰 ID 和與 IAM 委託人相關聯的私密存取金鑰來簽署。或者，您可以使用 [AWS Security Token Service](#) (AWS STS) 產生臨時安全憑證來簽署請求。

## 使用 Core 對生產車隊或設備進 AWS IoT 行安全監控

IoT 機群可由大量的裝置組成具有多樣化的功能、長時間在線上，並且散佈在多個地理位置。這些特點使機群的設定變得複雜且極易出錯。由於裝置通常受限於運算能力、記憶體和儲存功能，因此限制了裝置本身的加密和其他形式的安全性的使用。此外，裝置通常會使用具有已知漏洞的軟體。這些因素讓 IoT 機群成為對駭客極具吸引力的目標，且難以持續保護裝置機群。

AWS IoT Device Defender 通過提供工具來識別安全問題和與最佳實踐的偏差來解決這些挑戰。您可以使用 AWS IoT Device Defender 來分析、稽核和監視連線的裝置，以偵測異常行為並降低安全風險。AWS IoT Device Defender 可以稽核裝置叢集，確保它們遵守安全性最佳實務，並偵測裝置上的異常

行為。這可讓您在 AWS IoT 裝置叢集中強制執行一致的安全性原則，並在裝置遭到入侵時迅速做出回應。如需詳細資訊，請參閱 [AWS IoT Device Defender](#)。

AWS IoT 裝置建議程式會視需要推送更新並修補您的叢集。AWS IoT 裝置顧問會自動更新測試案例。您選取的測試案例一律搭配最新版本。如需詳細資訊，請參閱 [Device Advisor](#)。

## 安全性最佳做法 AWS IoT Core

本節包含的安全性最佳做法的相關資訊 AWS IoT Core。如需產業 IoT 解決方案安全規則的相關資訊，請參閱 [產業 IoT 解決方案的十大安全黃金法則](#)。

### 保護 MQTT 連接 AWS IoT

[AWS IoT Core](#) 是一種託管雲服務，使連接的設備可以輕鬆安全地與雲應用程式和其他設備進行交互。AWS IoT Core 支援 HTTP、和 [MQTT WebSocket](#)，這是一種專為容忍間歇性連線而設計的輕量型通訊協定。如果您要 AWS IoT 使用 MQTT 連線，則每個連線都必須與稱為用戶端 ID 的識別碼相關聯。MQTT 用戶端 ID 是獨一無二的，可識別 MQTT 連線。如果使用已針對其他連線宣告的用戶端 ID 建立新連線，則 AWS IoT 訊息代理程式會中斷舊的連線以允許新連線。每 AWS 帳戶 個用戶端 ID 必須是唯一的 AWS 區域。這意味著您不需要強制執行客戶端 ID 的全局唯一性，在您的 AWS 帳戶。AWS 帳戶

在您的裝置機群上捨棄 MQTT 連線的影響和嚴重性，取決於多個因素。其中包含：

- 您的使用案例 (例如，裝置傳送的資料 AWS IoT、資料量以及資料傳送的頻率)。
- 您的 MQTT 用戶端組態 (例如，自動重新連線設定、關聯的退避計時，以及 [MQTT 持久性工作階段](#) 的使用)。
- 裝置資源限制。
- 連線中斷的根本原因，以及其積極性和持久性。

為了避免用戶端 ID 衝突及其潛在的負面影響，請確定每個裝置或行動應用程式都有 AWS IoT 或 IAM 政策，以限制哪些用戶端 ID 可用於與訊息代理程式的 MQTT 連線。AWS IoT 例如，您可以使用 IAM 政策，藉由使用已在使用的用戶端 ID 來防止裝置意外關閉其他裝置的連線，如需詳細資訊，請參閱 [授權](#)。

叢集中的所有裝置都必須具有僅授權預期動作的權限的認證，其中包括 (但不限於) AWS IoT MQTT 動作，例如發佈訊息或訂閱具有特定範圍和內容的主題。特定的許可政策可能因您的使用案例而異。識別最符合您業務和安全性需求的許可政策。



為了簡化許可政策的建立和管理，您可以使用 [AWS IoT Core 策略變數](#) 和 [IAM 政策變數](#)。政策變數可以放置在政策中，當政策接受評估時，該變數可由來自裝置的請求值取代。透過使用政策變數，您可以建立一個單一政策，以授與許可給多個裝置。您可以根據 AWS IoT 帳戶組態、驗證機制和連線至 AWS IoT 訊息代理程式時使用的網路通訊協定，識別使用案例的相關原則變數。不過，若要撰寫最佳的許可政策，您需要考慮您使用案例的特定情形和 [威脅模型](#)。

例如，如果您在登 AWS IoT 錄中註冊裝置，則可以使用 [原則中 AWS IoT 的物件原則變數](#)，根據物件名稱、物類型及物屬性值等物內容授與或拒絕權限。物件名稱是從物件連線時傳送的 MQTT 連線訊息中的用戶端識別碼取得。AWS IoT [當物件使用 TLS 相互驗證或 MQTT 透過 WebSocket 通訊協定使用經 AWS IoT 過驗證的 Amazon Cognito 身分連線至透過 MQTT 時](#)，物件原則變數會被 [取代](#)。您可以使用 [AttachThing](#) 主體 API 將憑證和經過驗證的 Amazon Cognito 身分附加至物件。iot:Connection.Thing.ThingName 是強制執行用戶端 ID 限制的有用原則變數。下列範例 AWS IoT 原則需要使用已註冊物件的名稱作為 MQTT 連線至 AWS IoT 訊息代理程式的用戶端識別碼：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": [
        "arn:aws:iot:us-east-1:123456789012:client/${iot:Connection.Thing.ThingName}"
      ]
    }
  ]
}
```

如果您想要識別持續發生的用戶端 ID 衝突，您可以啟用和使用 [CloudWatch 記錄 AWS IoT](#)。對於 AWS IoT 訊息代理程式因為用戶端識別碼衝突而中斷連線的每個 MQTT 連線，會產生類似下列內容的記錄檔記錄：

```
{
  "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 記錄篩選器](#)，例如搜尋`{$.reason= "DUPLICATE_CLIENT_ID" }`用戶端 ID 衝突的執行個體，或設定[CloudWatch 指標篩選器](#)和對應 CloudWatch 警示，以便持續監視和報告。

您可以使用[AWS IoT 裝置防禦者](#)來識別過於寬鬆 AWS IoT 和 IAM 政策。AWS IoT Device Defender 還提供稽核檢查，通知您叢集中的多個裝置是否使用相同的用戶端 ID 連線到 AWS IoT 訊息代理程式。

您可以使用 AWS IoT 裝置建議程式來驗證您的裝置是否可靠地連線 AWS IoT Core 並遵循安全性最佳做法。

## 另請參閱

- [AWS IoT Core](#)
- [AWS IoT 的安全性功能](#)
- [AWS IoT Core 策略變數](#)
- [IAM 政策變數](#)
- [Amazon Cognito 身分](#)
- [AWS IoT 設備後衛](#)
- [CloudWatch 的記錄 AWS IoT](#)

## 讓裝置的時鐘保持同步

在裝置上保持準確的時間是很重要的。X.509 憑證具到期日期和時間。裝置上的時鐘用來驗證伺服器憑證是否仍然有效。如果您正在建置商用 IoT 裝置，請記住您的產品在售出前可能會長期存放。在這段期間，即時時鐘可能會漂移，電池可能會放電，因此在工廠設定時間並不足夠。

對大多數系統而言，這表示裝置的軟體必須包含網路時間通訊協定 (NTP) 用戶端。裝置應該等到與 NTP 伺服器同步時，才會嘗試連線到 AWS IoT Core。如果無法這麼做，系統應該提供方法讓使用者能夠設定裝置的時間，以便讓後續連線成功。

一旦裝置與 NTP 伺服器同步，就可以開啟與 AWS IoT Core 的連線。允許的時脈偏移會取決於您嘗試對連線執行的操作而定。

## 驗證伺服器憑證

設備與之交互的第一件事 AWS IoT 是打開安全連接。當您將設備連接到時 AWS IoT，請確保您正在與其他伺服器通話，AWS IoT 而不是模擬 AWS IoT 其他伺服器。每個 AWS IoT 伺服器都會使用針對 `iot.amazonaws.com` 網域發行的憑證佈建。此憑證是 AWS IoT 由可信任的憑證授權單位核發，該機構會驗證我們的網域身分和擁有權。

當設備連接時，首先要 AWS IoT Core 做的事情之一就是向設備發送伺服器證書。裝置可以驗證其預期連線至 `iot.amazonaws.com`，以及該連線端的伺服器是否擁有來自該網域之受信任授權單位的憑證。

TLS 憑證採用 X.509 格式，並包含各種資訊，例如組織的名稱、位置、網域名稱和有效期間。有效期間以稱為 `notBefore` 和 `notAfter` 的一對時間值來指定。諸如伺服器證書 AWS IoT Core 使用有限的有效期（例如，一年），並在舊證書過期之前開始提供新的服務。

## 使用每個裝置單一身分

使用每個用戶端單一身分。裝置通常使用 X.509 用戶端憑證。Web 與行動應用程式使用 Amazon Cognito 身分。這可讓您將精細的許可套用至您的裝置。

例如，您有由行動電話裝置組成的應用程式，該裝置從兩個不同的智慧型家用物件（燈泡和電熱器）接收狀態更新。燈泡會傳送其電池的電量狀態，恆溫器會傳送報告溫度的訊息。

AWS IoT 個別驗證裝置，並個別處理每個連線。您可以使用授權政策套用微調的存取控制。您可以定義電熱器的政策，以允許電熱器發佈至主題空間。您可以為燈泡定義個別的政策，以允許燈泡發佈至不同的主題空間。最後，您可以定義行動應用程式的政策，只允許它連線和訂閱電熱器和燈泡的主題，以接收來自這些裝置的訊息。

盡可能套用最低權限的政策，並盡可能降低每個裝置的許可範圍。所有裝置或使用者都應具有 AWS IoT 策略 `AWS IoT`，其中只允許其與已知的用戶端 ID 連線，以及發佈和訂閱已識別且已修正的主題集。

## 使用第二個 AWS 區域 作為備份

考慮在一秒鐘內存儲數據的副本 AWS 區域 作為備份。請注意，名為 [災難復原的 AWS 解決方案](#) AWS IoT 已不再可用。雖然關聯的程式 [GitHub 庫](#) 仍然可供存取，但在 2023 年 7 月將其 AWS 棄用，不再為其提供維護或支援。若要實作您自己的解決方案或探索其他支援選項，請造訪 [Cont act AWS](#)。如果您的帳戶有關聯的 AWS 技術客戶經理，請聯絡他們尋求協助。

## 使用及時佈建

手動建立和佈建每個裝置可能非常耗時。AWS IoT 提供一種定義範本的方法，以便在裝置首次連線時佈建裝置 AWS IoT。如需詳細資訊，請參閱 [Just-in-time 佈建](#)。

## 執行 AWS IoT 裝置建議程式測試的權限

下列政策範本顯示執行 AWS IoT 裝置顧問測試案例所需的最低許可和 IAM 實體。您需要將 *your-device-role-arn* 取代為在[先決條件](#)下建立的裝置角色 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
        "iot:Connect",
        "iot:CreateJob",
        "iot>DeleteJob",
        "iot:DescribeCertificate",
        "iot:DescribeEndpoint",
        "iot:DescribeJobExecution",
        "iot:DescribeJob",
        "iot:DescribeThing",
        "iot:GetPendingJobExecutions",
        "iot:GetPolicy",
        "iot:ListAttachedPolicies",
        "iot:ListCertificates",
        "Advisor console"
      ]
    }
  ]
}
```

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

## Device Advisor 跨服務預防混淆代理人

混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多權限的實體執行該動作。在中 AWS，跨服務模擬可能會導致混淆的副問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。為了防止這種情況發生，AWS 提供的工具可協助您透過已授予您帳戶中資源存取權的服務主體來保護所有服務的資料。

若要限制 Device Advisor 為資源提供另一項服務的許可，我們推薦在資源政策中使用 [aws:SourceArn](#) 和 [aws:SourceAccount](#) 全域條件內容索引鍵。如果同時使用全域條件內容索引鍵，則在相同政策陳述式中使用 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的帳戶時，必須使用相同的帳戶 ID。

`aws:SourceArn` 值必須是套件定義資源的 ARN。套件定義資源是指使用 Device Advisor 建立的測試套件。

防範混淆代理人問題最有效的方法，是使用 `aws:SourceArn` 全域條件內容金鑰，以及資源的完整 ARN。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 `aws:SourceArn` 全域條件內容金鑰，同時使用萬用字元 (\*) 表示 ARN 的未知部分。例如 `arn:aws:iotdeviceadvisor:*:account-id:suitedefinition/*`。

下列範例示範如何使用 Device Advisor 中的 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容金鑰，來預防混淆代理人問題。

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

## AWS 培訓和認證

參加以下課程，了解安全性的關鍵概念：AWS IoT [AWS IoT 安全入門](#)。

# 監控 AWS IoT

監控是維持 AWS 解決方案的可靠性、可用性和效能的 AWS IoT 重要組成部分。

我們強烈建議您從 AWS 解決方案的所有部分收集監控資料，以便在發生多點故障時更容易偵錯。請在一開始先建立可回答下列問題的監視計劃。如果您不確定如何回答這些問題，您仍然可以繼續[啟用記錄](#)並建立效能基準。

- 監控目標是什麼？
- 監控哪些資源？
- 監控這些資源的頻率為何？
- 將使用哪些監控工具？
- 誰將執行監控任務？
- 發生問題時應該通知誰？

您的下一個步驟是透過測量不同時間和不同負載條件下的 AWS IoT 效能，來[啟用記錄](#)並建立環境中正常效能的基準。監視時 AWS IoT，請保留歷史監視資料，以便您可以將其與目前的效能資料進行比較。這可協助您識別正常效能模式和效能異常情況，並策劃解決這些情況的方法。

若要建立您的基準線效能 AWS IoT，您應該監視這些測量結果以啟動。您可以隨時監控更多度量。

- [PublishIn.Success](#)
- [PublishOut.Success](#)
- [Subscribe.Success](#)
- [Ping.Success](#)
- [Connect.Success](#)
- [GetThingShadow.Accepted](#)
- [UpdateThingShadow.Accepted](#)
- [DeleteThingShadow.Accepted](#)
- [RulesExecuted](#)

本節中的主題可協助您開始記錄和監視 AWS IoT。

主題

- [設定 AWS IoT 記錄](#)
- [使用 Amazon 監控 AWS IoT 警示和指標 CloudWatch](#)
- [AWS IoT 使用 CloudWatch 記錄監控](#)
- [將設備端日誌上傳到 Amazon CloudWatch](#)
- [使用記錄 AWS IoT API 呼叫 AWS CloudTrail](#)

## 設定 AWS IoT 記錄

您必須先使用 AWS IoT 主控台、CLI 或 API 啟用記錄，才能監視和記錄 AWS IoT 活動。

您可以啟用所有 AWS IoT 或僅針對特定物件群組的記錄功能。您可以使用 AWS IoT 主控台、CLI 或 API 來設定 AWS IoT 記錄；不過，您必須使用 CLI 或 API 來設定特定物件群組的記錄。

在考慮如何設定 AWS IoT 記錄時，預設的記錄組態會決定如何記錄 AWS IoT 活動，除非另有指定。開始之後，您可能想要取得具有 INFO 的預設 [日誌層級](#) 或 DEBUG。檢閱初始日誌後，您可以將預設日誌層級變更為較不詳細的層級，例如 WARN 或 ERROR，並在可能需要更多注意的資源上設定更詳細的資源特定日誌層級。您可以隨時更改日誌層級。

本主題涵蓋雲端登入 AWS IoT。如需裝置端記錄和監控的相關資訊，請參閱 [將裝置端記錄上傳至 CloudWatch](#)

如需有關記錄和監控的資訊 AWS IoT Greengrass，請參閱 [中的記錄和監控 AWS IoT Greengrass](#)。自 2023 年 6 月 30 日起，AWS IoT Greengrass 核心軟體已移轉至 AWS IoT Greengrass Version 2。

## 設定記錄角色和政策

啟用登入之前 AWS IoT，您必須先建立 IAM 角色和政策，以提供代表您監控 AWS IoT 活動的 AWS 權限。您也可以根據 [AWS IoT 主控台的日誌區段](#) 所需政策產生 IAM 角色。

### Note

啟用 AWS IoT 記錄之前，請確定您瞭解 CloudWatch 錄檔存取權限。具有 CloudWatch 記錄存取權的使用者可以從您的裝置看到偵錯資訊。如需詳細資訊，請參閱 [Amazon CloudWatch 日誌的身分驗證和存取控制](#)。

如果您預期會 AWS IoT Core 因為負載測試而產生高流量模式，請考慮關閉 IoT 記錄以防止節流。若偵測到高流量，我們的服務可能會停用您的帳戶登入。



以下說明如何建立 AWS IoT Core 資源的記錄角色和原則。

## 建立記錄角色

若要建立記錄角色，請開啟 [IAM 主控台的角色中樞](#)，然後選擇 Create role (建立角色)。

1. 在 Select trusted entity (選取信任的實體) 下，選取 AWS service (服務)。然後在 Use case (使用案例) 下選擇 IoT。如果未顯示 IoT，請從 Use cases for other AWS services: (其他服務的使用案例：) 下拉式清單中輸入並搜尋 IoT。選取下一步。
2. 在 Add permissions (新增許可) 頁面上，您會看到自動連接到服務角色的政策。選擇下一步。
3. 在 Name, review, and create (命名、檢閱和建立) 頁面上，輸入角色的 Role name (角色名稱) 和 Role description (角色描述)，然後選擇 Create role (建立角色)。
4. 在 Roles (角色) 清單中，找到並開啟建立的角色，然後複製角色 ARN (*logging-role-arn*) 以於您 [在 AWS IoT \(主控台\) 中設定預設記錄](#) 時使用。

## 記錄角色政策

下列原則文件提供角色原則和信任原則，可 AWS IoT 讓您代表您提 CloudWatch 交記錄項目。如果您也允許 LoRa WAN AWS IoT Core 提交記錄項目，您會看到為您建立的原則文件，用於記錄這兩項活動。

### Note

這些文件是在您建立記錄角色時為您建立的。這些文件具有變數 (*\${partition}*、*\${region}* 和 *\${accountId}*)，您必須將其取代為您的值。

角色政策：

```
{
  "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:AWSIoTLogsV2:*"
  ]
}
]
}

```

信任策略僅記錄 AWS IoT Core 活動：

```

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

```

## 在 AWS IoT (主控台) 中設定預設記錄

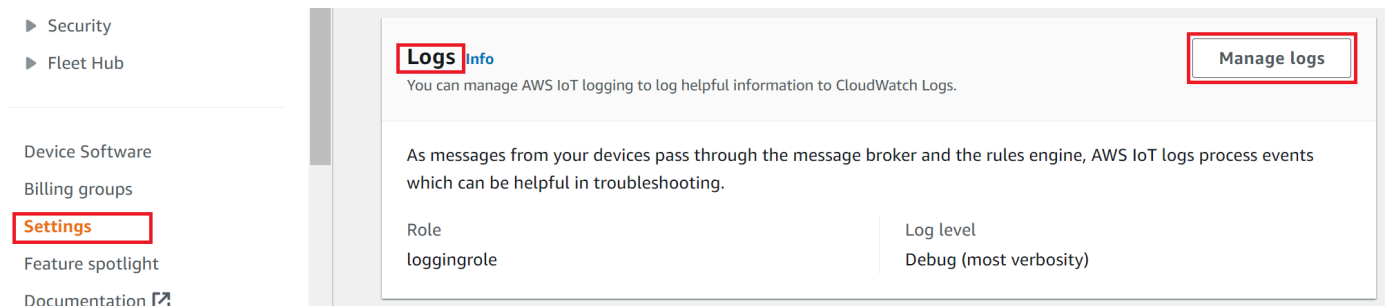
本節說明如何使用主 AWS IoT 控制台設定所有項目的記錄 AWS IoT。若要僅針對特定的事物群組設定記錄，您必須使用 CLI 或 API。如需設定特定事物群組記錄的相關資訊，請參閱 [在 AWS IoT 設定資源特定的記錄 \(CLI\)](#)。

使用 AWS IoT 主控台設定所有記錄的預設記錄 AWS IoT

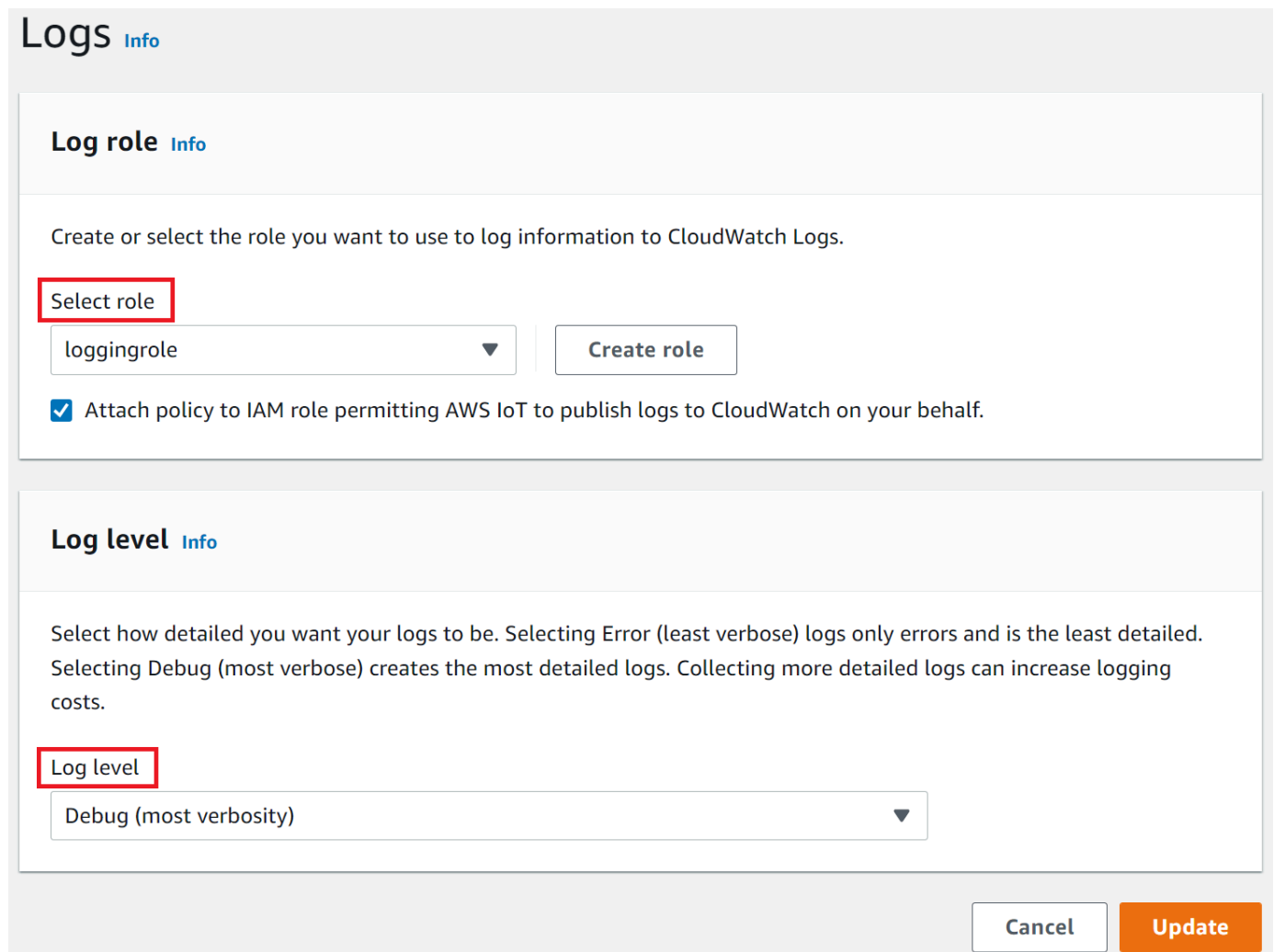
1. 登入 AWS IoT 主控台。如需詳細資訊，請參閱 [開啟主 AWS IoT 控制台](#)。

- 在左側的導覽窗格中，選擇設定。在 Settings (設定) 頁面的 Logs (記錄) 區段中，選擇 Manage logs (管理日誌)。

Logs (日誌) 頁面會顯示所有 AWS IoT 使用的記錄日誌角色和詳細資訊層級。



- 在 Logs (日誌) 頁面，選擇 Select role (選取角色) 以指定您先前建立於 [建立記錄角色](#) 的角色，或選擇 Create Role (建立角色) 以建立一個用於記錄的角色。



- 選擇說明您要顯示在記錄檔中之記錄項目 [詳細資訊層級](#) 的 CloudWatch 記錄層級。

## 5. 選擇 Update (更新) 以儲存您的設定。

啟用記錄之後，請造訪 [在 CloudWatch 主控台中檢視 AWS IoT 記錄](#) 以進一步了解檢視記錄項目。

## 設定預設登入 AWS IoT (CLI)

本節說明如何使用 CLI 設定全域記錄。AWS IoT

### Note

您需要您想要使用的角色的 Amazon Resource Name (ARN)。如果您需要建立用於記錄的角色，請在繼續前參閱 [建立記錄角色](#)。  
用於進行 API 呼叫的委託人必須擁有您記錄角色的 [傳遞角色權限](#)。

您也可以使用與此處顯示的 CLI 命令對應的 AWS API 中的方法，使用 API 執行此程序。

使用 CLI 設定下列項目的預設記錄 AWS IoT

### 1. 請使用 [set-v2-logging-options](#) 命令來設定您的帳戶的記錄選項。

```
aws iot set-v2-logging-options \  
  --role-arn logging-role-arn \  
  --default-log-level log-level
```

其中：

`--role-arn`

角色 ARN，授與在記錄檔中寫入記錄檔的 AWS IoT CloudWatch 權限。

`--default-log-level`

欲使用的 [日誌層級](#)。有效值為 ERROR、WARN、INFO、DEBUG 或 DISABLED。

`--no-disable-all-logs`

可啟用所有 AWS IoT 記錄的選用參數。使用此參數可在目前該參數停用時啟用記錄功能。

`--disable-all-logs`

停用所有 AWS IoT 記錄的選用參數。使用此參數可在該參數目前啟用時停用記錄功能。

2. 使用 [get-v2-logging-options](#) 命令取得目前的記錄選項。

```
aws iot get-v2-logging-options
```

啟用記錄之後，請造訪 [在 CloudWatch 主控台中檢視 AWS IoT 記錄](#) 以進一步了解檢視記錄項目。

#### Note

AWS IoT 繼續支持較舊的命令 ( [set-logging-options](#) 和 [get-logging-options](#) ) 來設置和獲取您的帳戶的全局日誌記錄。請注意，這些指令在使用時所產生的記錄包含純文字，而非 JSON 有效承載，且記錄延遲通常會更高。這些較舊指令的實施將不會進一步改進。我們建議您使用「v2」版本以設定您的記錄選項，並且，若可能的話，變更使用較舊版本的舊版應用程式。

## 在 AWS IoT 設定資源特定的記錄 (CLI)

本節說明如何使用 CLI 設定資源特定 AWS IoT 的記錄。資源特定的記錄允許您指定特定 [物件群組](#) 的日誌層級。

物件群組可以包含其他物件群組來建立階層關係。此程序說明如何設定單一物件群組的記錄。您可以將此程序套用至階層中的父項物件群組，以設定階層中所有物件群組的記錄。您也可以將此過程應用於子事件組，以覆蓋其父項的記錄組態。

除物件群組之外，您還可以記錄目標，如裝置的用戶端 ID、來源 IP 和主體 ID。

#### Note

您需要您想要使用的角色的 Amazon Resource Name (ARN)。如果您需要建立用於記錄的角色，請在繼續前參閱 [建立記錄角色](#)。

用於進行 API 呼叫的委託人必須擁有您記錄角色的 [傳遞角色權限](#)。

您也可以使用與此處顯示的 CLI 命令對應的 AWS API 中的方法，使用 API 執行此程序。

若要使用 CLI 設定資源特定的記錄 AWS IoT

1. 請使用 [set-v2-logging-options](#) 命令來設定您的帳戶的記錄選項。

```
aws iot set-v2-logging-options \
```

```
--role-arn logging-role-arn \  
--default-log-level log-level
```

其中：

`--role-arn`

角色 ARN，授與在記錄檔中寫入記錄檔的 AWS IoT CloudWatch 權限。

`--default-log-level`

欲使用的 [日誌層級](#)。有效值為 ERROR、WARN、INFO、DEBUG 或 DISABLED。

`--no-disable-all-logs`

可啟用所有 AWS IoT 記錄的選用參數。使用此參數可在目前該參數停用時啟用記錄功能。

`--disable-all-logs`

停用所有 AWS IoT 記錄的選用參數。使用此參數可在該參數目前啟用時停用記錄功能。

## 2. 使用 [set-v2-logging-level](#) 命令來設定物件群組的資源特定記錄。

```
aws iot set-v2-logging-level \  
    --log-target targetType=THING_GROUP,targetName=thing_group_name \  
    --log-level log_level
```

`--log-target`

您設定欲記錄的資源類型與名稱。target\_type 值必須是以下其中之一：THING\_GROUP | CLIENT\_ID | SOURCE\_IP | PRINCIPAL\_ID。log-target 參數值可以是文字，如前面的命令範例所示，也可以是 JSON 字串，例如下列範例。

```
aws iot set-v2-logging-level \  
    --log-target '{"targetType": "THING_GROUP","targetName":  
    "thing_group_name"}' \  
    --log-level log_level
```

`--log-level`

為指定資源產生日誌時所用的日誌層級。有效值

為：DEBUG、INFO、ERROR、WARN、DISABLED

```
aws iot set-v2-logging-level \  
    --log-target targetType=CLIENT_ID,targetName=ClientId1 \  
    --log-level DEBUG
```

3. 使用 [list-v2-logging-levels](#) 命令列出目前設定的日誌層級。

```
aws iot list-v2-logging-levels
```

4. 使用 [delete-v2-logging-level](#) 命令可刪除特定資源的日誌層級，如下列範例所示：

```
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 值必須是以下其中之一：THING\_GROUP | CLIENT\_ID | SOURCE\_IP | PRINCIPAL\_ID。

--targetName

要移除日誌層級之物件群組的名稱。

啟用記錄之後，請造訪 [在 CloudWatch 主控台中檢視 AWS IoT 記錄](#) 以進一步了解檢視記錄項目。

## 日誌層級

這些日誌層級會決定記錄的事件，並套用至預設和資源特定的日誌層級。

### ERROR

導致操作失敗的錯誤。

日誌內僅含 ERROR 資訊。

## WARN

會導致系統內出現不一致，但可能不會造成操作失敗的情況。

日誌內僅含 ERROR 和 WARN 資訊。

## INFO

物件流的高層級資訊。

日誌包含 INFO、ERROR、WARN 資訊。

## DEBUG

可能有助於偵錯問題的資訊。

日誌包含 DEBUG、INFO、ERROR、WARN 資訊。

## DISABLED

已停用所有記錄功能。

# 使用 Amazon 監控 AWS IoT 警示和指標 CloudWatch

您可以 AWS IoT 使用監視器 CloudWatch，它將原始數據收集並處理 AWS IoT 為可讀的近實時指標。這些統計資料會保存兩週的期間，以便您存取歷史資訊，並更清楚 web 應用程式或服務的執行方式。根據預設，AWS IoT 指標資料會以 CloudWatch 一分鐘的間隔自動傳送到。如需詳細資訊，請參閱[什麼是 Amazon CloudWatch、Amazon CloudWatch 活動和 Amazon CloudWatch 日誌？](#) 在 Amazon 用 CloudWatch 戶指南。

## 使用 AWS IoT 指標

報告的指標會提 AWS IoT 供您可以用不同方式分析的資訊。以下使用案例是根據案例，您有十個項目，每天連接到網際網路。每個天：

- 大致相同的時間連接到 AWS IoT 十件事。
- 每個真的訂閱主題篩選，然後等待一小時才能中斷。在此期間，事與其他應用程式通訊和進一步了解世界各地的狀態。
- 每個動作會根據其新發現有一些感知發佈資料 UpdateThingShadow。
- 每件事都斷開連接。AWS IoT

為了協助您開始使用，這些主題會探索可能遇到的一些問題。



- [如何每天都在我的物件連線失敗時收到通知？](#)
- [如何每天都在我的物件並未推送資料時收到通知？](#)
- [如何每天都在我的物件的影子更新遭拒時收到通知？](#)
- [如何為工作建立 CloudWatch 警示？](#)

更多關於 CloudWatch 警報和指標

- [建立要監視的 CloudWatch 警報 AWS IoT](#)
- [AWS IoT 量度和維度](#)

## 建立要監視的 CloudWatch 警報 AWS IoT

您可以建立 CloudWatch 警示，在警示狀態變更時傳送 Amazon SNS 訊息。警示會在您指定的期間，監看單一指標。當指標值在數個期間內超過指定的閾值時，會執行一或多個動作。此動作可為傳送至 Amazon SNS 主題或 Auto Scaling 政策的通知。警示只會針對持續的狀態變更觸發動作。CloudWatch 警示不會僅因為處於特定狀態而觸發動作；狀態必須已變更並維持指定數目的期間。

下列主題說明使用 CloudWatch 警示的一些範例。

- [如何每天都在我的物件連線失敗時收到通知？](#)
- [如何每天都在我的物件並未推送資料時收到通知？](#)
- [如何每天都在我的物件的影子更新遭拒時收到通知？](#)
- [如何為工作建立 CloudWatch 警示？](#)

您可以查看 CloudWatch 警報可以監控的所有指標[AWS IoT 量度和維度](#)。

### 如何每天都在我的物件連線失敗時收到通知？

1. 建立名為 `things-not-connecting-successfully` 的 Amazon SNS 主題，並記錄其 Amazon 資源名稱 (ARN)。此程序會將您主題的 ARN 稱為 *sns-topic-arn*。

如需如何建立 Amazon SNS 通知的詳細資訊，請參閱 [Amazon SNS 入門](#)。

2. 建立警示。

```
aws cloudwatch put-metric-alarm \  
  --alarm-name ConnectSuccessAlarm \  
  --alarm-description "Alarm when my Things don't connect successfully" \  
  --namespace AWS/IoT \  
  --metric-name ConnectSuccess \  
  --threshold 1 \  
  --evaluation-periods 1 \  
  --period 300 \  
  --dimensions Namespace=AWS/IoT,ThingName=MyThings \  
  --tags Environment=Production
```

```
--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. 測試警示。

```
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. 確認警示是否出現在[CloudWatch 主機](#)中。

## 如何每天都在我的物件並未推送資料時收到通知？

1. 建立名為 `things-not-publishing-data` 的 Amazon SNS 主題，並記錄其 Amazon 資源名稱 (ARN)。此程序會將您主題的 ARN 稱為 *sns-topic-arn*。

如需如何建立 Amazon SNS 通知的詳細資訊，請參閱 [Amazon SNS 入門](#)。

### 2. 建立警示。

```
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. 測試警示。

```
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. 確認警示是否出現在 [CloudWatch 主機](#) 中。

## 如何每天都在我的物件的影子更新遭拒時收到通知？

1. 建立名為 `things-shadow-updates-rejected` 的 Amazon SNS 主題，並記錄其 Amazon 資源名稱 (ARN)。此程序會將您主題的 ARN 稱為 *sns-topic-arn*。

如需如何建立 Amazon SNS 通知的詳細資訊，請參閱 [Amazon SNS 入門](#)。

2. 建立警示。

```
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. 測試警示。

```
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. 確認警示是否出現在 [CloudWatch 主機](#) 中。

### 如何為工作建立 CloudWatch 警示？

「工作」服務提供 CloudWatch 指標，供您監視工作。您可以創建 CloudWatch 警報以監視任何警報 [任務指標](#)。

下列命令會建立 CloudWatch 警示，以監視工作 *SampleOTAJob Job #####* *##### 20 ###* 執行失敗時通知您。警示每 300 秒會檢查報告的值，以監控任務指標 `FailedJobExecutionTotalCount`。當單一報告值大於 20 時，便會啟動警示，表示自任務啟動以來，失敗的任務執行次數超過 20 次。當警示關閉時，會傳送通知給所提供的 Amazon SNS 主題。

```
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-eexecutions
```

下列命令會建立 CloudWatch 警示，以監視指定期間內工作 *SampleOta Job* 失敗的工作執行次數。然後，當在此期間有超過五個任務執行失敗時，會通知您。警示每 3600 秒會檢查報告的值，以監控任務指標 `FailedJobExecutionCount`。當單一報告值大於 5 時，便會啟動警示，表示在過去一小時內，失敗的任務執行次數超過 5 次。當警示關閉時，會傳送通知給所提供的 Amazon SNS 主題。

```
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
```

```
--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-executions-per-hour
```

## AWS IoT 量度和維度

當您與之互動時 AWS IoT，服務會 CloudWatch 每分鐘傳送下列量度和維度。您可以使用下列程序來檢視 AWS IoT 的指標

若要檢視量度 (CloudWatch 主控台)

指標會先依服務命名空間分組，再依各命名空間內不同的維度組合分類。

1. 開啟 [CloudWatch 主控台](#)。
2. 在導覽窗格中，選擇 Metrics (指標)，然後選擇 All metrics (所有指標)。
3. 在「瀏覽」標籤中，搜尋 AWS IoT 以檢視測量結果清單。

檢視指標 (CLI)

- 在命令提示中，使用下列命令：

```
aws cloudwatch list-metrics --namespace "AWS/IoT"
```

CloudWatch 顯示下列測量結果群組 AWS IoT：

- [AWS IoT 度量](#)
- [AWS IoT Core 認證提供者度量](#)
- [身分驗證指標](#)
- [伺服器憑證 OCSP 裝訂測量結果](#)
- [規則指標](#)
- [規則動作指標](#)
- [HTTP 動作特定指標](#)

- [訊息代理程式指標](#)
- [Device Shadow 指標](#)
- [任務指標](#)
- [Device Defender Audit 指標](#)
- [Device Defender Detect 指標](#)
- [裝置佈建指標](#)
- [LoRaWAN 指標](#)
- [機群索引指標](#)
- [指標的維度](#)

## AWS IoT 度量

指標	描述
AddThingToDynamicThingGroupsFailed	與將物件新增至動態物件群組相關聯的失敗物件數量。此 DynamicThingGroupName 維度包含無法新增項目的動態群組名稱。
NumLogBatchesFailedToPublishThrottled	因調節錯誤而無法發佈的單批日誌事件。
NumLogEventsFailedToPublishThrottled	批次中因調節錯誤而無法發佈的日誌事件數。

## AWS IoT Core 認證提供者度量

指標	描述
CredentialExchangeSuccess	對 AWS IoT Core 憑證提供者提出成功的 AssumeRoleWithCertificate 請求數。

## 身分驗證指標

### Note

驗證度量會顯示在 CloudWatch 主控台下的「通訊協定測量結果」下。

指標	描述
Connect.AuthNError	因驗證失敗而 AWS IoT Core 拒絕的連線嘗試次數。此測量結果只會考慮傳送與您 AWS 帳戶的端點相符之伺服器名稱指示 (SNI) 字串的連線。此測量結果包括來自外部來源的連線嘗試，例如網際網路掃描工具或探查活動。Protocol 維度包含用於傳送連線嘗試的通訊協定。

## 伺服器憑證 OCSP 裝訂測量結果

指標	描述
檢索。成功 StapleData	OCSP 回應已收到並成功處理。此回應將包含在已設定網域的 TLS 交涉期間。DomainConfigurationName 維度包含已啟用伺服器憑證 OCSP 裝訂的已設定網域名稱。

## 規則指標

指標	描述
ParseError	JSON 剖析錯誤數目，發生於對規則所接聽主題發佈的訊息中。RuleName 維度會包含規則的名稱。
RuleMessageThrottled	由於惡意行為或訊息數量超過規則引擎調節限制，而被規則引擎調節的訊息數量。RuleName 維度包含欲觸發規則的名稱。

指標	描述
RuleNotFound	找不到欲觸發的規則。RuleName 維度會包含規則的名稱。
RulesExecuted	執行的 AWS IoT 規則數目。
TopicMatch	對規則所接聽主題發佈的傳入訊息數目。RuleName 維度會包含規則的名稱。

## 規則動作指標

指標	描述
Failure	失敗規則動作呼叫次數。RuleName 維度包含可指定動作之規則的名稱。ActionType 維度包含已呼叫動作的類型。
Success	成功規則動作呼叫次數。RuleName 維度包含可指定動作之規則的名稱。ActionType 維度包含已呼叫動作的類型。
ErrorActionFailure	失敗的錯誤動作數量。RuleName 維度包含可指定動作之規則的名稱。ActionType 維度包含已呼叫動作的類型。
ErrorActionSuccess	成功的錯誤動作數量。RuleName 維度包含可指定動作之規則的名稱。ActionType 維度包含已呼叫動作的類型。

## HTTP 動作特定指標

指標	描述
HttpCode_Other	如果來自下游 Web 服務/應用程式的回應狀態碼不是 2xx、4xx 或 5xx，則會產生此指標。



指標	描述
HttpCode_4XX	如果來自下游 Web 服務/應用程式的回應狀態碼介於 400 到 499 之間，則會產生此指標。
HttpCode_5XX	如果來自下游 Web 服務/應用程式的回應狀態碼介於 500 到 599 之間，則會產生此指標。
HttpInvalidUrl	在取代替換範本之後，如果端點 URL 未以 https:// 開頭，則會產生此指標。
HttpRequestTimeout	如果下游 Web 服務/應用程式未在請求逾時限制內傳回回應，則會產生此指標。如需詳細資訊，請參閱 <a href="#">Service Quotas</a> 。
HttpUnknownHost	如果 URL 有效，但服務不存在或無法連線，則會產生此指標。

## 訊息代理程式指標

### Note

訊息中介測量結果會顯示在 CloudWatch 主控台下的「通訊協定測量結果」

指標	描述
Connect.AuthError	訊息代理程式無法授權的連線要求數目。Protocol 維度包含用來傳送 CONNECT 訊息的協定。
Connect.ClientError	因 MQTT 訊息不符合 <a href="#">AWS IoT 配額</a> 中所定義的需求而拒絕的連線要求數。Protocol 維度包含用來傳送 CONNECT 訊息的協定。
Connect.ClientIDThrottle	由於用戶端超過特定用戶端 ID 所允許之連線要求率而調節的連線要求數目。Protocol 維度包含用來傳送 CONNECT 訊息的協定。

指標	描述
Connect.ServerError	因發生內部錯誤而失敗的連線要求數目。Protocol 維度包含用來傳送 CONNECT 訊息的協定。
Connect.Success	訊息代理程式成功的連線次數。Protocol 維度包含用來傳送 CONNECT 訊息的協定。
Connect.Throttle	因帳戶超過允許連線要求率而調節的連線要求數目。Protocol 維度包含用來傳送 CONNECT 訊息的協定。
Ping.Success	訊息代理程式收到的 ping 訊息數目。Protocol 維度包含用來傳送 ping 訊息的協定。
PublishIn.AuthError	訊息代理程式無法授權的發佈要求數目。Protocol 維度包含用來發佈訊息的協定。HTTP 發佈不支援此指標。
PublishIn.ClientError	因訊息不符合 <a href="#">AWS IoT 配額</a> 中所定義的需求，而遭訊息代理程式拒絕的發佈要求數。Protocol 維度包含用來發佈訊息的協定。HTTP 發佈不支援此指標。
PublishIn.ServerError	訊息代理程式因發生內部錯誤而無法處理的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。HTTP 發佈不支援此指標。
PublishIn.Success	訊息代理程式成功處理的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。
PublishIn.Throttle	因用戶端超過允許傳入訊息率而調節的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。HTTP 發佈不支援此指標。
PublishOut.AuthError	AWS IoT無法授權之訊息代理程式所提出的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。

指標	描述
<code>PublishOut.ClientError</code>	由訊息代理程式發送，但因訊息不符合 <a href="#">AWS IoT 配額</a> 中所定義的需求而拒絕的發佈要求數。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。
<code>PublishOut.Success</code>	訊息代理程式成功發出的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。
<code>PublishOut.Throttle</code>	因用戶端超過所允許之傳出訊息率而調節的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。
<code>PublishRetained.AuthError</code>	訊息代理程式無法授權帶 RETAIN 旗標集的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。
<code>PublishRetained.ServerError</code>	訊息代理程式因發生內部錯誤而無法處理的保留發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。
<code>PublishRetained.Success</code>	訊息代理程式成功處理帶 RETAIN 旗標集的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。
<code>PublishRetained.Throttle</code>	因用戶端超過允許傳入訊息率而調節的帶 RETAIN 旗標集的發佈要求數目。Protocol 維度包含用來傳送 PUBLISH 訊息的協定。
<code>Queued.Success</code>	訊息代理程式與其中斷持久性工作階段連線的用戶端成功處理的儲存訊息數目。具有持久性工作階段的用戶端中斷連線時，會儲存 QoS 為 1 的訊息。
<code>Queued.Throttle</code>	具有持久性工作階段的用戶端中斷連線時，無法儲存和限流的訊息數目。當用戶端超過 <a href="#">每個帳戶每秒的佇列訊息數</a> 限制時，就會發生這種情況。具有持久性工作階段的用戶端中斷連線時，會儲存 QoS 為 1 的訊息。

指標	描述
Queued.ServerError	因為內部錯誤而未針對持久性工作階段儲存的訊息數目。當具有持久性工作階段的用戶端中斷連線時，會儲存服務品質 (QoS) 為 1 的訊息。
Subscribe.AuthError	由用戶端提出且無法授權的訂閱要求數目。Protocol 維度包含用來傳送 SUBSCRIBE 訊息的協定。
Subscribe.ClientError	因 SUBSCRIBE 訊息不符合 <a href="#">AWS IoT 配額</a> 中所定義的要求，而拒絕的訂閱要求數。Protocol 維度包含用來傳送 SUBSCRIBE 訊息的協定。
Subscribe.ServerError	因發生內部錯誤而拒絕的訂閱要求數目。Protocol 維度包含用來傳送 SUBSCRIBE 訊息的協定。
Subscribe.Success	訊息代理程式成功處理的訂閱要求數目。Protocol 維度包含用來傳送 SUBSCRIBE 訊息的協定。
Subscribe.Throttle	限制的訂閱要求數目，因為超過了您的允許的訂閱要求速率限制。AWS 帳戶這些限制包括每個帳戶的每秒訂閱數、每個帳戶的訂閱數，以及 <a href="#">AWS IoT Core 訊息代理程式和通訊協定限制與配額</a> 中所述的每個連線的訂閱 Protocol 維度包含用來傳送 SUBSCRIBE 訊息的協定。
Throttle.Exceeded	CloudWatch 當 MQTT 用戶端在每個連線層級限制上的 <a href="#">每秒封包數限制上節流時，將會顯示此度量</a> 。此指標不適用於 HTTP 連線。
Unsubscribe.ClientError	因 UNSUBSCRIBE 訊息不符合 <a href="#">AWS IoT 配額</a> 中所定義的需求，而拒絕的取消訂閱要求數。Protocol 維度包含用來傳送 UNSUBSCRIBE 訊息的協定。
Unsubscribe.ServerError	因發生內部錯誤而拒絕的取消訂閱要求數目。Protocol 維度包含用來傳送 UNSUBSCRIBE 訊息的協定。

指標	描述
Unsubscribe.Success	訊息代理程式成功處理的取消訂閱要求數目。Protocol 維度包含用來傳送 UNSUBSCRIBE 訊息的協定。
Unsubscribe.Throttle	因用戶端超過允許取消訂閱要求率而拒絕的取消訂閱要求數目。Protocol 維度包含用來傳送 UNSUBSCRIBE 訊息的協定。

## Device Shadow 指標

### Note

裝置陰影度量會顯示在 CloudWatch 主控台下的「通訊協定指標」下。

指標	描述
DeleteThingShadow.Accepted	成功處理的 DeleteThingShadow 要求數目。Protocol 維度包含用來提出要求的協定。
GetThingShadow.Accepted	成功處理的 GetThingShadow 要求數目。Protocol 維度包含用來提出要求的協定。
ListThingShadow.Accepted	成功處理的 ListThingShadow 要求數目。Protocol 維度包含用來提出要求的協定。
UpdateThingShadow.Accepted	成功處理的 UpdateThingShadow 要求數目。Protocol 維度包含用來提出要求的協定。

## 任務指標

指標	描述
CanceledJobExecutionCount	在由 CloudWatch 決定的期間 CANCELED 內，其狀態已變更為的工作執行數目。如需有關 CloudWatch 指標的詳細資訊，請參閱 <a href="#">Amazon CloudWatch 指標</a> 。) JobId 維度包含任務的 ID。
CanceledJobExecutionTotalCount	針對指定任務，狀態為 CANCELED 的任務執行總數。JobId 維度包含任務的 ID。
ClientErrorCount	執行任務時產生的用戶端錯誤數目。JobId 維度包含任務的 ID。
FailedJobExecutionCount	在由 CloudWatch 決定的期間 FAILED 內，其狀態已變更為的工作執行數目。如需有關 CloudWatch 指標的詳細資訊，請參閱 <a href="#">Amazon CloudWatch 指標</a> 。) JobId 維度包含任務的 ID。
FailedJobExecutionTotalCount	針對指定任務，狀態為 FAILED 的任務執行總數。JobId 維度包含任務的 ID。
InProgressJobExecutionCount	在由 CloudWatch 決定的期間 IN_PROGRESS 內，其狀態已變更為的工作執行數目。如需有關 CloudWatch 指標的詳細資訊，請參閱 <a href="#">Amazon CloudWatch 指標</a> 。) JobId 維度包含任務的 ID。
InProgressJobExecutionTotalCount	針對指定任務，狀態為 IN_PROGRESS 的任務執行總數。JobId 維度包含任務的 ID。
RejectedJobExecutionTotalCount	針對指定任務，狀態為 REJECTED 的任務執行總數。JobId 維度包含任務的 ID。
RemovedJobExecutionTotalCount	針對指定任務，狀態為 REMOVED 的任務執行總數。JobId 維度包含任務的 ID。
QueuedJobExecutionCount	在由 CloudWatch 決定的期間 QUEUED 內，其狀態已變更為的工作執行數目。如需有關 CloudWatch 指標

指標	描述
	的詳細資訊，請參閱 <a href="#">Amazon CloudWatch 指標</a> 。) JobId 維度包含任務的 ID。
QueuedJobExecutionTotalCount	針對指定任務，狀態為 QUEUED 的任務執行總數。JobId 維度包含任務的 ID。
RejectedJobExecutionCount	在由 CloudWatch 決定的期間 REJECTED 內，其狀態已變更為的工作執行數目。如需有關 CloudWatch 指標的詳細資訊，請參閱 <a href="#">Amazon CloudWatch 指標</a> 。) JobId 維度包含任務的 ID。
RemovedJobExecutionCount	在由 CloudWatch 決定的期間 REMOVED 內，其狀態已變更為的工作執行數目。如需有關 CloudWatch 指標的詳細資訊，請參閱 <a href="#">Amazon CloudWatch 指標</a> 。) JobId 維度包含任務的 ID。
ServerErrorCount	執行任務時產生的伺服器錯誤數目。JobId 維度包含任務的 ID。
SucceededJobExecutionCount	在由 CloudWatch 決定的期間 SUCCESS 內，其狀態已變更為的工作執行數目。如需有關 CloudWatch 指標的詳細資訊，請參閱 <a href="#">Amazon CloudWatch 指標</a> 。) JobId 維度包含任務的 ID。
SucceededJobExecutionTotalCount	針對指定任務，狀態為 SUCCESS 的任務執行總數。JobId 維度包含任務的 ID。

## Device Defender Audit 指標

指標	描述
NonCompliantResources	在檢查中發現不相容的資源數量。在每次稽核執行的檢查時，系統都會回報不合規的資源數量。
ResourcesEvaluated	評估符合的資源數目。系統會為每次執行的稽核檢查報告評估的資源數量。

指標	描述
MisconfiguredDeviceDefenderNotification	<p>在的 SNS 組態設定錯誤時通知您。AWS IoT Device Defender</p> <p><a href="#">Dimensions (尺寸)</a></p>

## Device Defender Detect 指標

指標	描述
NumOfMetricsExported	為雲端、裝置端或自訂指標匯出的指標數目。系統會針對特定量度報告為帳戶匯出的量度數目。此指標僅適用於使用量度匯出的客戶。
NumOfMetricsSkipped	雲端、裝置端或自訂度量略過的測量結果數目。由於提供給「裝置防禦者偵測」的權限不足以發佈到 mqtt 主題，系統會報告特定度量針對該帳戶跳過的指標數量。此指標僅適用於使用量度匯出的客戶。
NumOfMetricsExceedingSizeLimit	由於大小超過 MQTT 郵件大小限制，因此針對雲端、裝置端或自訂量度跳過的匯出量度數目。由於大小超過 MQTT 郵件大小限制，系統會針對特定量度報告為帳戶跳過的匯出量度數目。此指標僅適用於使用量度匯出的客戶。
Violations	自從上次執行評估起，發現新違反安全性描述檔行為的數目。系統會回報帳戶、特定安全性設定檔，以及特定安全性設定檔特定行為的新違規次數。
ViolationsCleared	自上次評估執行以來已解決的安全性設定檔行為的違規次數。系統會報告已解決的帳戶、特定安全性設定檔以及特定安全性設定檔特定行為的違規次數。
ViolationsInvalidated	自上次執行評估起，資訊不再可用的安全性描述檔行為違反之數目 (因為報告裝置停止報告，或因為某些原因不再受到監控)。系統會報告整個帳戶，特定安全性



指標	描述
	設定檔以及特定安全性設定檔特定行為的無效違規次數。
MisconfiguredDeviceDefenderNotification	在的 SNS 組態設定錯誤時通知您。AWS IoT Device Defender  <a href="#">Dimensions (尺寸)</a>

## 裝置佈建指標

### AWS IoT 叢集佈建指標

指標	描述
ApproximateNumberOfThingsRegistered	已由機群佈建註冊的事項計數。  雖然計數通常相當準確，但 AWS IoT Core 的分配式架構將會很難保持已註冊事項的精確計數。  此指標使用的統計數字為： <ul style="list-style-type: none"> <li>Max (最大值)，報告已註冊的事項總數。如需在 CloudWatch 彙總視窗期間註冊的物件計數，請參閱 <code>RegisterThingFailed</code> 指標。</li> </ul> 尺寸： <a href="#">ClaimCertificate 識別碼</a>
CreateKeysAndCertificateFailed	呼叫 CreateKeysAndCertificate MQTT API 時發生的失敗次數。  在成功 (值 = 0) 和失敗 (值 = 1) 兩種情況下，皆會發出指標。此指標可用來追蹤在 CloudWatch 支援的彙總時段 (例如 5 分鐘或 1 小時) 期間建立和註冊的憑證數目。  此指標可用的統計數字為： <ul style="list-style-type: none"> <li>Sum (總和)，報告失敗的呼叫數量。</li> </ul>

指標	描述
	<ul style="list-style-type: none"> <li>• SampleCount報告成功和失敗的呼叫總數。</li> </ul>
CreateCertificateFromCsrFailed	<p>呼叫 CreateCertificateFromCsr MQTT API 時發生的失敗次數。</p> <p>在成功 (值 = 0) 和失敗 (值 = 1) 兩種情況下，皆會發出指標。此量度可用來追蹤在 CloudWatch 支援的彙總時段期間註冊的物件數目，例如 5 分鐘或 1 小時。</p> <p>此指標可用的統計數字為：</p> <ul style="list-style-type: none"> <li>• Sum (總和)，報告失敗的呼叫數量。</li> <li>• SampleCount報告成功和失敗的呼叫總數。</li> </ul>
RegisterThingFailed	<p>呼叫 RegisterThing MQTT API 時發生的失敗次數。</p> <p>在成功 (值 = 0) 和失敗 (值 = 1) 兩種情況下，皆會發出指標。此量度可用來追蹤在 CloudWatch 支援的彙總時段期間註冊的物件數目，例如 5 分鐘或 1 小時。如需註冊事項的總數，請參閱 ApproximateNumberOfThingsRegistered 指標。</p> <p>此指標可用的統計數字為：</p> <ul style="list-style-type: none"> <li>• Sum (總和)，報告失敗的呼叫數量。</li> <li>• SampleCount報告成功和失敗的呼叫總數。</li> </ul> <p>維度：<a href="#">TemplateName</a></p>

## Just-in-time 佈建指標

指標	描述
ProvisionThing.ClientError	裝置因用戶端錯誤而無法佈建的次數。例如，指定於範本中的政策不存在。

指標	描述
<code>ProvisionThing.ServerError</code>	裝置因伺服器錯誤而無法佈建的次數。客戶可在等待後重試佈建裝置，而若問題仍然存在，其可聯絡 AWS IoT。
<code>ProvisionThing.Success</code>	成功佈建的裝置的次數。

## LoRaWAN 指標

下表顯示 LoRa WAN 的 AWS IoT Core 測量結果。如需詳細資訊，請參閱 [AWS IoT Core 閱 LoRa WAN 測量結果](#)。

### AWS IoT Core 適用於 LoRa WAN 度量

指標	描述
作用中裝置/道	您帳戶中使用中 LoRa WAN 裝置和閘道的數量。
上行訊息計數	在指定的持續時間內，針對您 AWS 帳戶中的所有作用中閘道和裝置傳送的上行訊息數目。上行訊息是從裝置傳送至 LoRa WAN AWS IoT Core 的訊息。
下行鏈路消息計數	在指定的持續時間內，針對您 AWS 帳戶的所有作用中閘道和裝置傳送的下行連結訊息數目。下行鏈路消息是從 AWS IoT Core LoRa WAN 發送到您的設備的消息。
訊息遺失率	新增裝置並連線至 LoRa WAN 之後，您 AWS IoT Core 的裝置可以啟動上行訊息，開始與雲端交換訊息。您可以使用此量度追蹤遺失上行訊息的速率。
加入量度	新增裝置和閘道之後，您可以執行聯結程序，以便裝置可以傳送上行資料並與 LoRa WAN 進 AWS IoT Core 行通訊。您可以使用此指標來取得有關您中所有作用中裝置之聯結量度的資訊 AWS 帳戶。
平均接收訊號強度指示器 (RSSI)	您可以使用此度量監視指定持續時間內的平均 RSSI (接收訊號強度指示器)。RSSI 是一種測量結果，指出

指標	描述
	信號是否足以實現良好的無線連接。此值為負值，對於強式連接，必須接近零。
平均信噪比 (SNR)	您可以使用此測量結果監督指定持續時間內的平均 SNR (Signal-to-noise 比率)。SNR 是一種測量結果，指示接收到的訊號是否足夠強，相較於雜訊級別，以達到良好的無線連線。SNR 值為正值，且必須大於零，以表示訊號功率大於雜訊功率。
閘道可用性	您可以使用此測量結果，取得此閘道在指定期間內的使用狀態相關資訊。此測量結果顯示此閘道在指定期間內的 websocket 連線時間。

## Just-in-time 佈建指標

指標	描述
<code>ProvisionThing.ClientError</code>	裝置因用戶端錯誤而無法佈建的次數。例如，指定於範本中的政策不存在。
<code>ProvisionThing.ServerError</code>	裝置因伺服器錯誤而無法佈建的次數。客戶可在等待後重試佈建裝置，而若問題仍然存在，其可聯絡 AWS IoT。
<code>ProvisionThing.Success</code>	成功佈建的裝置的次數。

## 機群索引指標

### AWS IoT 叢集索引指標

指標	描述
<code>NamedShadowCountForDynamicGroupQueryLimitExceeded</code>	對於動態物件群組中並非特定於資料來源的查詢術語，每個物件最多處理 25 個已命名影子。當物件違反此限制時，會發出 <code>NamedShadowCountFo</code>

指標	描述
	rDynamicGroupQueryLimitExceeded 事件類型。

## 指標的維度

指標使用命名空間，並提供下列維度的指標。

維度	描述
ActionType	觸發請求，並由規則指定的 <a href="#">動作類型</a> 。
BehaviorName	被監控的 Device Defender Detect 安全性設定檔行為名稱。
ClaimCertificateId	用來佈建裝置的 certificateId 宣告。
CheckName	其結果正受到監控的 Device Defender 稽核檢查的名稱。
JobId	正在監控其進度或訊息連線成功/失敗的任務 ID。
Protocol	用來提出要求的協定。有效值為：MQTT 或 HTTP
RuleName	要求所觸發規則的名稱。
ScheduledAuditName	其檢查結果正受到監控之 Device Defender 排程稽核的名稱。若回報的結果是依隨需執行的稽核結果，則此值為 OnDemand。
SecurityProfileName	其行為正受到監控之 Device Defender Detect 安全性描述檔的名稱。
TemplateName	佈建範本的名稱。
SourceArn	指要偵測的安全性設定檔或要稽核的帳號 arn。
RoleArn	指的是裝置防禦者嘗試假設的角色。

維度	描述
TopicArn	指的是 SNS 主題裝置防禦者嘗試發佈至。
Error	<p>提供嘗試發佈至 SNS 主題時收到之錯誤的簡短描述。 可能值為：</p> <ul style="list-style-type: none"><li>• 「KeyNot找到 KMS」：表示該主題的 KMS 金鑰不存在。</li><li>• 「InvalidTopic名稱」：表示 SNS 主題無效。</li><li>• AccessDenied「KMS」：表示角色沒有主題 KMS 金鑰的權限。</li><li>• 「AuthorizationError」：表示提供的角色未授權裝置防禦者發佈至 SNS 主題。</li><li>• 「TopicNot找到 SNS」：表示提供的 SNS 主題不存在。</li><li>• 「FailureToAssumeRole」：表示所提供的角色未授權裝置防禦者擔任該角色。</li><li>• 「CrossRegionSNSToPic」：表示 SNS 主題存在於不同的地區。</li></ul>

## AWS IoT 使用 CloudWatch 記錄監控

[啟用AWS IoT 記錄時](#)，AWS IoT 會在每個訊息透過訊息代理程式和規則引擎從您的裝置傳送進度事件。在[CloudWatch 主控台](#)中，CloudWatch 記錄會顯示在名為的記錄群組中AWSIoTLogs。

如需 CloudWatch 記錄檔的詳細資訊，請參閱[CloudWatch 記錄檔](#)。如需有關支援 AWS IoT CloudWatch 記錄檔的資訊，請參閱[CloudWatch 記 AWS IoT 錄檔項目](#)。

### 在 CloudWatch 主控台中檢視 AWS IoT 記錄

#### Note

在下列情況下，AWSIoTLogsV2記錄群組才會顯示在 CloudWatch 主控台中：

- 您已啟用登入功能 AWS IoT。如需如何啟用登入功能的詳細資訊 AWS IoT，請參閱 [設定 AWS IoT 記錄](#)

- 某些記錄項目已由 AWS IoT 作業寫入。

在 CloudWatch 主控台中檢視您的 AWS IoT 記錄

1. 請瀏覽 <https://console.aws.amazon.com/cloudwatch/>。在導覽窗格中，選擇 Log groups (日誌群組)。
2. 在 Filter (篩選條件) 文字方塊中，輸入 **AWSIoTLogsV2**，然後按下 Enter 鍵。
3. 請按兩下 AWSIoTLogsV2 日誌群組。
4. 選擇 Search All (全部搜尋)。此時會顯示為您的帳戶產生的 AWS IoT 記錄的完整清單。
5. 選擇展開圖示可觀看個別的串流。

您也可以在 Filter events (篩選條件事件) 中輸入一個查詢。以下是一些可以試試看的有趣查詢：

- `{ $.logLevel = "INFO" }`

尋找日誌層級為 INFO 的所有日誌。

- `{ $.status = "Success" }`

尋找日誌狀態為 Success 的所有日誌。

- `{ $.status = "Success" && $.eventType = "GetThingShadow" }`

尋找狀態為 Success 且事件類型為 GetThingShadow 的所有日誌。

如需建立篩選器運算式的詳細資訊，請參閱[CloudWatch 記錄查詢](#)。

## CloudWatch 記 AWS IoT 錄檔項目

的每個元件都 AWS IoT 會產生自己的記錄項目。每個日誌項目都有 eventType 指定造成日誌項目產生的作業。本節旨在說明下列 AWS IoT 元件所產生的記錄項目。

### 主題

- [Message broker 日誌項目](#)
- [伺服器憑證 OCSP 記錄項目](#)
- [Device Shadow 日誌項目](#)
- [Rules engine 日誌項目](#)

- [任務日誌項目](#)
- [裝置佈建日誌項目](#)
- [動態物件群組日誌項目](#)
- [機群索引日誌項目](#)
- [一般 CloudWatch 記錄檔屬性](#)

## Message broker 日誌項目

AWS IoT 訊息代理程式會產生下列事件的記錄項目：

### 主題

- [Connect 記錄項目](#)
- [Publish-In 日誌項目](#)
- [GetRetainedMessage 記錄項目](#)
- [ListRetainedMessage 記錄項目](#)
- [Publish-In 日誌項目](#)
- [Publish-Out 日誌項目](#)
- [佇列日誌項目](#)
- [訂閱日誌項目](#)

### Connect 記錄項目

當 MQTT 用戶端連線 Connect 時，AWS IoT 訊息代理程式會產生一個記錄項目。eventType

### Connect 日誌項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，Connect 記錄項目包含下列屬性：

#### clientId

提出請求的用戶端 ID。

#### principalId

提出請求的委託人 ID。

#### protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

#### sourceIp

請求來源所在的 IP 地址。

#### sourcePort

請求來源所在的埠口。

#### Publish-In 日誌項目

當 MQTT 用戶端中斷連線Disconnect時，AWS IoT 訊息代理程式會產生一個記錄項目。eventType

#### Disconnect 日誌項目的連線範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，Disconnect 記錄項目包含下列屬性：

#### clientId

提出請求的用戶端 ID。

#### principalId

提出請求的委託人 ID。

#### protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

#### sourceIp

請求來源所在的 IP 地址。

#### sourcePort

請求來源所在的埠口。

#### reason

用戶端中斷連線的原因。

#### 詳細資訊

錯誤的簡要說明。

#### disconnectReason

用戶端中斷連線的原因。

### GetRetainedMessage 記錄項目

AWS IoT 訊息代理程式會產生一個記錄項目，其中包含呼 [GetRetainedMessage](#) 叫 event 的 GetRetainedMessage 時間。

### GetRetainedMessage 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，GetRetainedMessage 記錄項目包含下列屬性：

#### 最後 ModifiedDate

保留訊息儲存時的 Epoch 日期和時間 (以毫秒為單位)。AWS IoT

#### protocol

用來提出要求的協定。有效值：HTTP。

#### qos

發佈請求中使用的服務品質 (QoS) 等級。有效值為 0 或 1。

#### topicName

訂閱主題的名稱。

#### ListRetainedMessage 記錄項目

AWS IoT 訊息代理程式會產生一個記錄項目，其中包含呼 [ListRetainedMessages](#) 叫 eventType 的 ListRetainedMessage 時間。

#### ListRetainedMessage 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，ListRetainedMessage 日誌項目包含下列屬性：

#### protocol

用來提出要求的協定。有效值：HTTP。

#### Publish-In 日誌項目

當 AWS IoT 訊息代理程式收到 MQTT 訊息時，會產生含有的記錄項目。eventType Publish-In

#### Publish-In 日誌項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，Publish-In 記錄項目包含下列屬性：

#### clientId

提出請求的用戶端 ID。

#### principalId

提出請求的委託人 ID。

#### protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

## 保留

當「保留」旗標訊息的值設定為 `True` 時所使用的屬性。如果訊息沒有設定「保留」旗標，則此屬性不會顯示在日誌條目中。如需詳細資訊，請參閱 [MQTT 保留訊息](#)。

### sourceIp

請求來源所在的 IP 地址。

### sourcePort

請求來源所在的埠口。

### topicName

訂閱主題的名稱。

## Publish-Out 日誌項目

當訊息代理程式發佈 MQTT 訊息時，會產生 `eventType` 為 `Publish-Out` 的日誌項目。

### Publish-Out 日誌項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，`Publish-Out` 記錄項目包含下列屬性：

### clientId

接收有關該 MQTT 主題之訊息的訂閱用戶端的 ID。

## principalId

提出請求的委託人 ID。

## protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

## sourceIp

請求來源所在的 IP 地址。

## sourcePort

請求來源所在的埠口。

## topicName

訂閱主題的名稱。

## 佇列日誌項目

當具有持續工作階段的裝置中斷連線時，MQTT 訊息代理程式會儲存裝置的訊息，並 AWS IoT 產生具有 EventType 的記錄項目。Queued 如需 MQTT 持久性工作階段的相關資訊，請參閱 [MQTT 持久性工作階段](#)。

## 佇列的伺服器錯誤日誌項目範例

```
{
  "timestamp": "2022-08-10 15:39:30.961",
  "logLevel": "ERROR",
  "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
  "accountId": "123456789012",
  "topicName": "$aws/things/MyThing/get",
  "clientId": "123123123",
  "qos": "1",
  "protocol": "MQTT",
  "eventType": "Queued",
  "status": "Failure",
  "details": "Server Error"
}
```

除了 [一般 CloudWatch 記錄檔屬性](#)，Queued 伺服器錯誤日誌項目包含下列屬性：

## clientId

訊息排入佇列的用戶端 ID。

## 詳細資訊

### Server Error

伺服器錯誤會造成訊息無法儲存。

## protocol

用來提出要求的協定。這個值將永遠為 MQTT。

## qos

請求的服務品質 (QoS) 等級。該值將始終為 1，因為具有 QoS 為 0 的訊息不會儲存。

## topicName

訂閱主題的名稱。

## 佇列成功日誌項目範例

```
{
  "timestamp": "2022-08-10 15:39:30.961",
  "logLevel": "INFO",
  "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
  "accountId": "123456789012",
  "topicName": "$aws/things/MyThing/get",
  "clientId": "123123123",
  "qos": "1",
  "protocol": "MQTT",
  "eventType": "Queued",
  "status": "Success"
}
```

除了 [一般 CloudWatch 記錄檔屬性](#)，Queued 成功日誌項目包含下列屬性：

## clientId

訊息排入佇列的用戶端 ID。

## protocol

用來提出要求的協定。這個值將永遠為 MQTT。

## qos

請求的服務品質 (QoS) 等級。該值將始終為 1，因為具有 QoS 為 0 的訊息不會儲存。

## topicName

訂閱主題的名稱。

## 佇列限流日誌項目範例

```
{
  "timestamp": "2022-08-10 15:39:30.961",
  "logLevel": "ERROR",
  "traceId": "672ec480-31ce-fd8b-b5fb-22e3ac420699",
  "accountId": "123456789012",
  "topicName": "$aws/things/MyThing/get",
  "clientId": "123123123",
  "qos": "1",
  "protocol": "MQTT",
  "eventType": "Queued",
  "status": "Failure",
  "details": "Throttled while queueing offline message"
}
```

除了 [一般 CloudWatch 記錄檔屬性](#)，Queued 限流日誌項目包含下列屬性：

## clientId

訊息排入佇列的用戶端 ID。

## 詳細資訊

### **Throttled while queueing offline message**

用戶端超出 [Queued messages per second per account](#) 限制，因此訊息不會儲存。

## protocol

用來提出要求的協定。這個值將永遠為 MQTT。

## qos

請求的服務品質 (QoS) 等級。該值將始終為 1，因為具有 QoS 為 0 的訊息不會儲存。



## topicName

訂閱主題的名稱。

## 訂閱日誌項目

當 MQTT 用戶端訂閱某個eventType主題Subscribe時，AWS IoT 訊息代理程式會產生一個記錄項目，其中包含 in。

## MQTT 3 訂閱日誌項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，Subscribe 記錄項目包含下列屬性：

### clientId

提出請求的用戶端 ID。

### principalId

提出請求的委託人 ID。

### protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

### sourceIp

請求來源所在的 IP 地址。

## sourcePort

請求來源所在的埠口。

## topicName

訂閱主題的名稱。

## MQTT 5 訂閱日誌項目範例

```
{
  "timestamp": "2022-11-30 16:24:15.628",
  "logLevel": "INFO",
  "traceId": "7aa5c38d-1b49-3753-15dc-513ce4ab9fa6",
  "accountId": "123456789012",
  "status": "Success",
  "eventType": "Subscribe",
  "protocol": "MQTT",
  "topicName": "test/topic1,$invalid/reserved/topic",
  "subscriptions": [
    {
      "topicName": "test/topic1",
      "reasonCode": 1
    },
    {
      "topicName": "$invalid/reserved/topic",
      "reasonCode": 143
    }
  ],
  "clientId": "abf27092886e49a8a5c1922749736453",
  "principalId": "145179c40e2219e18a909d896a5340b74cf97a39641beec2fc3eeafc5a932167",
  "sourceIp": "205.251.233.181",
  "sourcePort": 13490
}
```

針對 MQTT 5 訂閱操作，除了 [一般 CloudWatch 記錄檔屬性](#) 和 [MQTT 3 訂閱日誌項目屬性](#) 之外，MQTT 5 Subscribe 日誌項目還包含下列屬性：

## 訂閱

「訂閱」請求中所要求主題與個別 MQTT 5 原因代碼之間的對應清單。如需更多詳細資訊，請參閱 [MQTT 原因代碼](#)。

## 伺服器憑證 OCSP 記錄項目

AWS IoT Core 會產生下列事件的記錄項目：

主題

- [擷取 OCSP StapleData 記錄項目](#)

### 擷取 OCSP StapleData 記錄項目

AWS IoT Core 當伺服器擷取 OCSP 裝訂資料RetrieveOCSPStapleData時，會產生一個記錄項目，eventType其中包含的。

### 擷取 OCSP StapleData 記錄項目範例

以下是的記錄項目範例Success。

```
{
  "timestamp": "2024-01-30 15:39:30.961",
  "logLevel": "INFO",
  "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
  "accountId": "123456789012",
  "status": "Success",
  "eventType": "RetrieveOCSPStapleData",
  "domainConfigName": "test-domain-config-name",
  "connectionDetails": {
    "httpStatusCode": "200",
    "ocspResponderUri": "http://ocsp.example.com",
    "sourceIp": "205.251.233.181",
    "targetIp": "250.15.5.3"
  },
  "ocspRequestDetails": {
    "requesterName": "iot.amazonaws.com",
    "requestCertId":
"30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:",
  },
  "ocspResponseDetails": {
    "responseCertId":
"30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:",
    "ocspResponseStatus": "successful",
    "certStatus": "good",
    "signature":
"4C:6F:63:61:6C:20:52:65:73:70:6F:6E:64:65:72:20:53:69:67:6E:61:74:75:72:65",
  }
}
```

```
"thisUpdateTime": "Jan 31 01:21:02 2024 UTC",
"nextUpdateTime": "Feb 02 00:21:02 2024 UTC",
"producedAtTime": "Jan 31 01:37:03 2024 UTC",
"stapledDataPayloadSize": "XXX"
}
}
```

以下是的記錄項目範例Failure。

```
{
  "timestamp": "2024-01-30 15:39:30.961",
  "logLevel": "ERROR",
  "traceId": "180532b7-0cc7-057b-687a-5ca1824838f5",
  "accountId": "123456789012",
  "status": "Failure",
  "reason": "A non 2xx HTTP response was received from the OCSP responder.",
  "eventType": "RetrieveOCSPStapleData",
  "domainConfigName": "test-domain-config-name",
  "connectionDetails": {
    "httpStatusCode": "444",
    "ocspResponderUri": "http://ocsp.example.com",
    "sourceIp": "205.251.233.181",
    "targetIp": "250.15.5.3"
  },
  "ocspRequestDetails": {
    "requesterName": "iot.amazonaws.com",
    "requestCertId":
    "30:3A:30:09:06:05:2B:0E:03:02:1A:05:00:04:14:9C:FF:90:A1:97:B0:4D:6C:01:B9:69:96:D8:3E:E7:A2:"
  }
}
```

對於RetrieveOCSPStaple作業，除了[一般 CloudWatch 記錄檔屬性](#)，記錄項目還包含下列屬性：

reason

操作失敗的原因。

網域 ConfigName

網域組態的名稱。

連接詳情

連接詳細信息的簡要說明。

- HTTP StatusCode

OCSP 回應程式針對用戶端對伺服器的要求而傳回的 HTTP 狀態碼。

- OCSP ResponderUri

從伺服器憑證 AWS IoT Core 擷取的 OCSP 回應程式 URI。

- sourceIp

伺服器的來源 IP 位 AWS IoT Core 址。

- 目標提示

OCSP 回應程式的目標 IP 位址。

### OCSP RequestDetails

OCSP 要求的詳細資料。

- 請求者名稱

傳送要求給 OCSP 回應程式之 AWS IoT Core 伺服器的識別碼。

- 請求 CertId

要求的憑證識別碼。這是要求 OCSP 回應的憑證識別碼。

### OCSP ResponseDetails

OCSP 回應的詳細資料。

- 回應 CertId

OCSP 回應的憑證識別碼。

- OCSP ResponseStatus

OCSP 回應的狀態。

- 認證狀態

憑證的狀態。

- signature

受信任實體套用至回應的簽章。

- 這 UpdateTime

正在指示的狀態已知正確的時間。

- 下一個 UpdateTime

提供有關憑證狀態之新資訊的時間或之前。

- 生產 AtTime

OCSP 回應程式簽署此回應的時間。

- 裝訂尺寸 DataPayload

裝訂資料的裝載大小。

## Device Shadow 日誌項目

AWS IoT Device Shadow 服務會針對下列事件產生記錄項目：

主題

- [DeleteThingShadow 記錄項目](#)
- [GetThingShadow 記錄項目](#)
- [UpdateThingShadow 記錄項目](#)

### DeleteThingShadow 記錄項目

當收到刪除 Device Shadow 的要求時，Device Shadow 服務會產生 eventType 為 DeleteThingShadow 的日誌項目。

### DeleteThingShadow 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，DeleteThingShadow 記錄項目包含下列屬性：

#### 裝置 ShadowName

要更新的影子名稱。

#### protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

#### topicName

請求所發布的主題名稱。

#### GetThingShadow 記錄項目

當收到影子的 get 要求時，Device Shadow 服務會產生 eventType 為 GetThingShadow 的日誌項目。

#### GetThingShadow 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，GetThingShadow 記錄項目包含下列屬性：

#### 裝置 ShadowName

所要求影子的名稱。

#### protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

## topicName

請求所發布的主題名稱。

## UpdateThingShadow 記錄項目

當收到更新 Device Shadow 的要求時，Device Shadow 服務會產生 eventType 為 UpdateThingShadow 的日誌項目。

## UpdateThingShadow 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，UpdateThingShadow 記錄項目包含下列屬性：

## 裝置 ShadowName

要更新的影子名稱。

## protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

## topicName

請求所發布的主題名稱。

## Rules engine 日誌項目

AWS IoT 規則引擎會產生下列事件的記錄檔：

## 主題



- [FunctionExecution 記錄項目](#)
- [RuleExecution 記錄項目](#)
- [RuleMatch 記錄項目](#)
- [RuleExecutionThrottled 記錄項目](#)
- [RuleNotFound 記錄項目](#)
- [StartingRuleExecution 記錄項目](#)

## FunctionExecution 記錄項目

當規則的 SQL 查詢呼叫外部函數時，規則引擎會產生 eventType 為 FunctionExecution 的日誌項目。當規則的動作向 AWS IoT 或另一個 Web 服務（例如，調用 get\_thing\_shadow 或 machinelearning\_predict）發出 HTTP 請求時，就會調用外部函數。

## FunctionExecution 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，FunctionExecution 記錄項目包含下列屬性：

### clientId

N/A 適用於 FunctionExecution 日誌。

### principalId

提出請求的委託人 ID。

## resources

該規則動作所使用的資源集合。

## ruleName

符合的規則名稱。

## topicName

訂閱主題的名稱。

## RuleExecution 記錄項目

AWS IoT 規則引擎觸發規則的動作時，會產生記RuleExecution錄項目。

## RuleExecution 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，RuleExecution 記錄項目包含下列屬性：

## clientId

提出請求的用戶端 ID。

## principalId

提出請求的委託人 ID。

## resources

該規則動作所使用的資源集合。

## ruleAction

受到觸發的動作名稱。

## ruleName

符合的規則名稱。

## topicName

訂閱主題的名稱。

## RuleMatch 記錄項目

當訊息代理程式收到符合 AWS IoT 規則eventType的訊息RuleMatch時，規則引擎會產生一個記錄項目，其中包含 in。

## RuleMatch 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，RuleMatch 記錄項目包含下列屬性：

## clientId

提出請求的用戶端 ID。

## principalId

提出請求的委託人 ID。

## ruleName

符合的規則名稱。

## topicName

訂閱主題的名稱。

## RuleExecutionThrottled 記錄項目

限制執行時，AWS IoT 規則引擎會產生一個記錄項目，其中包含 `eventType`

## RuleExecutionThrottled

## RuleExecutionThrottled 記錄項目範例

```
{
  "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": "Exection of Rule example_rule throttled"
}
```

除了 [一般 CloudWatch 記錄檔屬性](#)，RuleExecutionThrottled 記錄項目包含下列屬性：

## clientId

提出請求的用戶端 ID。

## 詳細資訊

錯誤的簡要說明。

## principalId

提出請求的委託人 ID。

## reason

字符串「RuleExecution節流」。

## ruleName

欲觸發的規則名稱。

## topicName

已發佈的主題名稱。

## RuleNotFound 記錄項目

當 AWS IoT 規則引擎找不到具有給定名稱的規則時，它會產生一個帶eventType有RuleNotFound.

## RuleNotFound 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，RuleNotFound 記錄項目包含下列屬性：

## clientId

提出請求的用戶端 ID。

## 詳細資訊

錯誤的簡要說明。

## principalId

提出請求的委託人 ID。

## reason

字符串「RuleNot找到」。

## ruleName

找不到的規則名稱。

## topicName

已發佈的主題名稱。

## StartingRuleExecution 記錄項目

當 AWS IoT 規則引擎開始觸發規則的動作時，它會產生一個帶eventType有StartingRuleExecution。

## StartingRuleExecution 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，rule- 記錄項目包含下列屬性：

## clientId

提出請求的用戶端 ID。

## principalId

提出請求的委託人 ID。

## ruleAction

受到觸發的動作名稱。

## ruleName

符合的規則名稱。

## topicName

訂閱主題的名稱。

## 任務日誌項目

AWS IoT Job 服務會產生下列事件的記錄項目。從裝置收到 MQTT 或 HTTP 請求時系統會產生日誌項目。

### 主題

- [DescribeJobExecution 記錄項目](#)
- [GetPendingJobExecution 記錄項目](#)
- [ReportFinalJobExecutionCount 記錄項目](#)
- [StartNextPendingJobExecution 記錄項目](#)
- [UpdateJobExecution 記錄項目](#)

### DescribeJobExecution 記錄項目

J AWS IoT obs 服務會在服務收到描述工作執行eventType的要求DescribeJobExecution時產生一個記錄項目，其中包含 a。

### DescribeJobExecution 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，GetJobExecution 記錄項目包含下列屬性：

#### clientId

提出請求的用戶端 ID。

#### clientToken

用來確保請求冪等的，是一個唯一、區分大小寫的識別符。如需詳細資訊，請參閱[如何確保冪等](#)。  
詳細資訊

任務裝置的其他資訊。

#### jobId

任務執行的任務 ID。

#### protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

#### topicName

用來提出請求的主題。

### GetPendingJobExecution 記錄項目

J AWS IoT obs 服務會在服務收到工作執行要求GetPendingJobExecution時產生一個記錄項目，eventType其中包含的。

### GetPendingJobExecution 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，GetPendingJobExecution 記錄項目包含下列屬性：

clientId

提出請求的用戶端 ID。

clientToken

用以確保請求冪等的，是一個唯一、區分大小寫的識別符。如需詳細資訊，請參閱[如何確保冪等](#)。

詳細資訊

任務裝置的其他資訊。

protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

topicName

訂閱主題的名稱。

ReportFinalJobExecutionCount 記錄項目

J AWS IoT obs 服務會在工作完成ReportFinalJobExecutionCount時產生一個記錄項目，entryType其中包含的。

ReportFinalJobExecutionCount 記錄項目範例

```
{  
  "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 [一般 CloudWatch 記錄檔屬性](#)，ReportFinalJobExecutionCount 記錄項目包含下列屬性：

#### 詳細資訊

任務裝置的其他資訊。

#### jobId

任務執行的任務 ID。

#### StartNextPendingJobExecution 記錄項目

當它收到啟動下一個擱置中工作執行的要求時，AWS IoT obs 服務會產生一個記錄項目，其中包eventTypes含StartNextPendingJobExecution。

#### StartNextPendingJobExecution 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，StartNextPendingJobExecution 記錄項目包含下列屬性：

#### clientId

提出請求的用戶端 ID。

#### clientToken

用以確保請求冪等的，是一個唯一、區分大小寫的識別符。如需詳細資訊，請參閱[如何確保冪等](#)。

#### 詳細資訊

任務裝置的其他資訊。

## protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

## topicName

用來提出請求的主題。

## UpdateJobExecution 記錄項目

J AWS IoT obs 服務會在服務收到更新工作執行eventType的要求UpdateJobExecution時，產生一個記錄項目，其中包含的。

## UpdateJobExecution 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#)，UpdateJobExecution 記錄項目包含下列屬性：

## clientId

提出請求的用戶端 ID。

## clientToken

用以確保請求冪等的，是一個唯一、區分大小寫的識別符。如需詳細資訊，請參閱[如何確保冪等](#)。

## 詳細資訊

任務裝置的其他資訊。

## jobId

任務執行的任務 ID。

## protocol

用來提出要求的協定。有效值為 MQTT 或 HTTP。

## topicName

用來提出請求的主題。

## versionNumber

任務執行的版本。

## 裝置佈建日誌項目

AWS IoT 裝置佈建服務會產生下列事件的記錄檔。

### 主題

- [GetDeviceCredentials 記錄項目](#)
- [ProvisionDevice 記錄項目](#)

### GetDeviceCredentials 記錄項目

AWS IoT 裝置佈建服務會在用戶端呼叫GetDeviceCredential時產生一個記錄項目，eventType其中包含 in GetDeviceCredential。

### GetDevice認證記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#) , GetDeviceCredentials 記錄項目包含下列屬性 :

#### 詳細資訊

錯誤的簡要說明。

裝置 CertificateId

裝置憑證的 ID。

#### ProvisionDevice 記錄項目

AWS IoT 裝置佈建服務會在用戶端呼叫ProvisionDevice時產生一個記錄項目 , eventType其中包含 in ProvisionDevice。

#### ProvisionDevice 記錄項目範例

```
{
  "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 [一般 CloudWatch 記錄檔屬性](#) , ProvisionDevice 記錄項目包含下列屬性 :

#### 詳細資訊

錯誤的簡要說明。

裝置 CertificateId

裝置憑證的 ID。

## 佈建 TemplateName

佈建範本的名稱。

## 動態物件群組日誌項目

AWS IoT 動態物件群組會針對下列事件產生記錄檔。

### 主題

- [AddThingToDynamicThingGroupsFailed 記錄項目](#)

### AddThingToDynamicThingGroupsFailed 記錄項目

AWS IoT 當無法將物件新增至指定的動態群組時，會產生具eventType有AddThingToDynamicThingGroupsFailed. 這會在物件符合要在動態物件群組中的準則時發生；不過，它無法加入至動態群組，或從動態群組中移除。發生這種情況的原因是：

- 事情已經屬於群組的最大數量。
- --override-dynamic-groups 選項用於將物件新增到靜態物件群組。它已經從動態物件群組中移除，使之成為可能。

如需詳細資訊，請參閱[動態物件群組限制與衝突](#)。

### AddThingToDynamicThingGroupsFailed 記錄項目範例

此範例顯示 AddThingToDynamicThingGroupsFailed 錯誤的日誌項目。在此範例中，TestThing符合中所列動態物群組中的條件dynamicThingGroupNames，但無法新增至這些動態群組，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 [一般 CloudWatch 記錄檔屬性](#)，AddThingToDynamicThingGroupsFailed 記錄項目包含下列屬性：

#### 動態ThingGroup名稱

無法新增物件的動態物件群組的陣列。

#### reason

物件無法新增到動態物件群組的原因。

#### thingName

無法新增至動態物件群組的物件名稱。

## 機群索引日誌項目

AWS IoT 叢集索引會針對下列事件產生記錄項目。

#### 主題

- [NamedShadowCountForDynamicGroupQueryLimitExceeded 記錄項目](#)

#### NamedShadowCountForDynamicGroupQueryLimitExceeded 記錄項目

對於動態群組中並非特定於資料來源的查詢術語，每個物件最多處理 25 個已命名影子。當物件違反此限制時，會發出 NamedShadowCountForDynamicGroupQueryLimitExceeded 事件類型。

#### NamedShadowCountForDynamicGroupQueryLimitExceeded 記錄項目範例

此範例顯示 NamedShadowCountForDynamicGroupQueryLimitExceeded 錯誤的日誌項目。在此範例中，基於 DynamicGroup 的所有值結果可能不正確，如 reason 欄位中所述。

```
{
  "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."
}
```

## 一般 CloudWatch 記錄檔屬性

所有 CloudWatch 記錄檔記錄項目都包含下列屬性：

accountId

你的 AWS 帳戶 身份證

eventType

產生日誌的事件類型。事件類型的值會根據產生該日誌項目的事件而定。每個日誌項目說明都包含該 eventType 日誌項目的值。

logLevel

使用的日誌層級。如需詳細資訊，請參閱 [the section called “日誌層級”](#)。

status

請求的狀態。

timestamp

用戶端連線至 AWS IoT 訊息代理程式時，人類看得懂的 UTC 時間戳記。

traceId

隨機產生的識別符，可用於建立某個特定請求的所有日誌的關聯性。

## 將設備端日誌上傳到 Amazon CloudWatch

您可以將歷史裝置端日誌上傳 CloudWatch 到 Amazon，以監控和分析現場設備的活動。設備端日誌可以包括系統、應用程式和裝置日誌檔案。[此程序會使用 CloudWatch Logs 規則動作參數，將裝置端記錄檔發佈至客戶定義的記錄群組。](#)



## 運作方式

該過程開始時，AWS IoT 設備將包含格式化日誌文件的 MQTT 消息發送到主題。AWS IoT 規則會監控訊息主題，並將記錄檔傳送至您定義的 CloudWatch 記錄群組。然後，您可以檢閱和分析資訊。

### 主題

- [MQTT 主題](#)
- [規則動作](#)

## MQTT 主題

選擇您將用來發佈日誌的 MQTT 主題名稱空間。我們建議將此格式用於共同主題空間 `$aws/rules/things/thing_name/logs`，並將此格式用於錯誤主題 `$aws/rules/things/thing_name/logs/errors`。建議使用日誌和錯誤主題的命名結構，但並非必要。如需詳細資訊，請參閱[設計 AWS IoT Core 的 MQTT 主題](#)。

透過使用建議的共同主題空間，您可以使用 AWS IoT 基本內嵌保留的主題。AWS IoT 基本內嵌可安全地將裝置資料傳送至 AWS IoT 規則動作支援的 AWS 服務。基本擷取會從擷取路徑移除發佈/訂閱訊息代理程式，因此更具成本效益。如需詳細資訊，請參閱[使用基本擷取減少簡訊費](#)。

如果您使用批次模式來上傳日誌檔案，您的訊息必須遵循包含 UNIX 時間戳記和訊息的特定格式。[如需詳細資訊，請參閱記錄檔規則動作中 CloudWatch 的批次模式 MQTT 訊息格式需求主題](#)。

## 規則動作

從用戶端裝置 AWS IoT 收到 MQTT 訊息時，AWS IoT 規則會監控客戶定義的主題，並將內容發佈到您定義的 CloudWatch 記錄群組中。此程序會使用「CloudWatch 記錄檔」規則動作來監視 MQTT 的記錄檔批次。如需詳細資訊，請參閱[CloudWatch 記錄檔](#) AWS IoT 規則處理行動。

### 批次模式

`batchMode` 是「AWS IoT CloudWatch 記錄檔」規則動作中的布林參數。這個參數是可選的，預設情況下是 `off (false)`。若要批次上傳裝置端記錄檔，您必須在建立規則 AWS IoT 時開啟 (`true`) 此參數。如需詳細資訊，請參閱[AWS IoT 規則動作](#) 區段中的 [CloudWatch 記錄檔](#)。

## 使用 AWS IoT 規則上傳裝置端日誌

您可以使用 AWS IoT 規則引擎，將現有裝置端日誌檔 (系統、應用程式和裝置用戶端日誌) 的日誌記錄上傳到 Amazon。CloudWatch 將裝置端記錄發佈至 MQTT 主題時，「記錄檔」規則動作會將訊息傳

送至「CloudWatch 記錄檔」。CloudWatch 此程序概述如何使用規則動作 `batchMode` 參數開啟 (設定為 `true`) 來批次上傳裝置日誌。

若要開始將裝置端記錄檔上傳到 CloudWatch，請完成下列先決條件。

## 必要條件

開始之前，請執行以下動作：

- 建立至少一個已註冊 AWS IoT Core 為物 AWS IoT 件的目標 IoT 裝置。如需詳細資訊，請參閱[建立物件](#)。
- 判斷用於擷取和錯誤的 MQTT 主題空間。如需 MQTT 主題和建議命名慣例的詳細資訊，請參閱將[裝置端日誌上傳](#)到 Amazon 中的 [MQTT 主題](#) [MQTT 主題](#) 一節。CloudWatch

如需這些必要條件的詳細資訊，請參閱[將裝置端記錄檔上傳](#)至。CloudWatch

## 建立記 CloudWatch 錄群組

若要建立 CloudWatch 記錄群組，請完成以下步驟。根據您是否偏好執行 AWS Management Console 或 AWS Command Line Interface (AWS CLI) 的步驟，選擇適當的索引標籤。

### AWS Management Console

若要使用建立 CloudWatch 日誌群組 AWS Management Console

1. 開啟 AWS Management Console 並導覽至[CloudWatch](#)。
2. 在導覽列上，選擇 Logs (日誌)，然後選擇 Log groups (日誌群組)。
3. 選擇 Create log group (建立日誌群組)。
4. 更新 Log group name (日誌群組名稱)，並選擇性地更新 Retention settings (保留設定) 欄位。
5. 選擇建立。

### AWS CLI

若要使用建立 CloudWatch 日誌群組 AWS CLI

1. 若要建立日誌群組，請執行以下命令。若要取得更多資訊，請參閱 AWS CLI v2 指令參考[create-log-group](#)中的 `<>`。

將範例 (`uploadLogsGroup`) 中的日誌群組名稱取代為您偏好的名稱。

```
aws logs create-log-group --log-group-name uploadLogsGroup
```

- 若要確認日誌群組已正確建立，請執行下列命令。

```
aws logs describe-log-groups --log-group-name-prefix uploadLogsGroup
```

輸出範例：

```
{
  "logGroups": [
    {
      "logGroupName": "uploadLogsGroup",
      "creationTime": 1674521804657,
      "metricFilterCount": 0,
      "arn": "arn:aws:logs:us-east-1:111122223333:log-
group:uploadLogsGroup:*",
      "storedBytes": 0
    }
  ]
}
```

## 建立主題規則

若要建立 AWS IoT 規則，請完成以下步驟。根據您是否偏好執行 AWS Management Console 或 AWS Command Line Interface (AWS CLI) 的步驟，選擇適當的索引標籤。

### AWS Management Console

若要使用建立主題規則 AWS Management Console

- 開啟規則中樞。
  - 開啟 AWS Management Console 並導覽至 [AWS IoT](#)。
  - 在導覽列上，選擇 Message routing (訊息路由)，然後選擇 Rules (規則)。
  - 選擇建立規則。
- 輸入規則屬性。
  - 輸入英數字元的 Rule name (規則名稱)。

- b. (選擇性) 輸入 Rule description (規則描述) 和 Tags (標籤)。
  - c. 選擇下一步。
3. 輸入 SQL 陳述式。

- a. 使用您為擷取定義的 MQTT 主題輸入 SQL 敘述句。

例如：`SELECT * FROM '$aws/rules/things/thing_name/logs'`

- b. 選擇下一步。
4. 輸入規則動作。

- a. 在「動作 1」功能表上，選擇「CloudWatch記錄檔」。
  - b. 選擇 Log group name (日誌群組名稱)，然後選擇您建立的日誌群組。
  - c. 選取 Use batch mode (使用批次模式)。
  - d. 指定規則的 IAM 角色。

如果您擁有規則的 IAM 角色，請執行下列動作。

1. 在 IAM role (IAM 角色) 選單上，選擇您的 IAM 角色。

如果您沒有規則的 IAM 角色，請執行下列動作。

1. 選擇 Create new role (建立新角色)。
2. 在 Role name (角色名稱) 中，輸入唯一的名稱，然後選擇 Create (建立)。
3. 確認 IAM role (IAM 角色) 欄位中的 IAM 角色 名稱是否正確。


- e. 選擇下一步。
5. 檢閱範本組態。
  - a. 檢閱 Job 範本的設定，以確認其正確無誤。
  - b. 完成時，選擇 Create (建立)。

## AWS CLI

若要建立 IAM 角色和主題規則，請使用 AWS CLI

1. 建立授與 AWS IoT 規則權限的 IAM 角色。
  - a. 建立 IAM 政策。

若要建立 IAM 政策，請執行下列命令。請務必更新 `policy-name` 參數值。若要取得更多資訊，請參閱 AWS CLI v2 指令參考 [create-policy](#) 中的 `<`。

 Note

如果您使用的是 Microsoft Windows 作業系統，您可能需要將行尾標記 (`\`) 取代為勾號 (`'`) 或其他字元。

```
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. 將政策 ARN 從輸出複製到文字編輯器中。

輸出範例：

```
{  
  "Policy": {  
    "PolicyName": "uploadLogsPolicy",  
    "PermissionsBoundaryUsageCount": 0,  
    "CreateDate": "2023-01-23T18:30:10Z",  
    "AttachmentCount": 0,  
    "IsAttachable": true,  
    "PolicyId": "AAABBBCCDDDEEEFFFGGG",  
    "DefaultVersionId": "v1",
```

```
    "Path": "/",
    "Arn": "arn:aws:iam::111122223333:policy/uploadLogsPolicy",
    "UpdateDate": "2023-01-23T18:30:10Z"
  }
}
```

c. 建立 IAM 角色和信任政策。

若要建立 IAM 政策，請執行下列命令。請務必更新 `role-name` 參數值。若要取得更多資訊，請參閱 AWS CLI v2 指令參考[create-role](#)中的 `<>`。

```
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. 將 IAM 政策連接至角色。

若要建立 IAM 政策，請執行下列命令。請務必更新 `role-name` 和 `policy-arn` 參數值。若要取得更多資訊，請參閱 AWS CLI v2 指令參考[attach-role-policy](#)中的 `<>`。

```
aws iam attach-role-policy \
--role-name uploadLogsRole \
--policy-arn arn:aws:iam::111122223333:policy/uploadLogsPolicy
```

e. 檢閱角色。

若要確認 IAM 角色已正確建立，請執行下列命令。請務必更新 `role-name` 參數值。若要取得更多資訊，請參閱 AWS CLI v2 指令參考[get-role](#)中的 `<>`。

```
aws iam get-role --role-name uploadLogsRole
```

輸出範例：

```
{
  "Role": {
    "Path": "/",
    "RoleName": "uploadLogsRole",
    "RoleId": "AAABBBCCDDDEEEFFFGGG",
    "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. 在中建立 AWS IoT 主題規則 AWS CLI。

- a. 若要建立 AWS IoT 主題規則，請執行下列命令。請務必更新 `--rule-name`、`sql` 陳述式、`description`、`roleARN`、和 `logGroupName` 參數值。如需詳細資訊，請參閱 v2 指令參考中的[建立主題規則](#)。AWS CLI

```
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. 若要確認規則已正確建立，請執行下列命令。請務必更新 `role-name` 參數值。如需詳細資訊，請參閱 [v2 命令參考中的取得主題規則](#)。AWS CLI

```
aws iot get-topic-rule --rule-name uploadLogsRule
```

輸出範例：

```
{
  "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"
  }
}
```



## 將裝置端日誌傳送至 AWS IoT

### 若要將裝置端記錄傳送至 AWS IoT

1. 若要將歷史記錄傳送至 AWS IoT，請與您的裝置通訊，以確保下列事項。
  - 日誌資訊會傳送至此程序的 Prerequisites (必要條件) 區段中所指定的正確主題命名空間。  
例如：`$aws/rules/things/thing_name/logs`
  - MQTT 訊息承載已正確格式化。如需 MQTT 主題和建議命名慣例的詳細資訊，請參閱 [將設備端日誌上傳到 Amazon CloudWatch](#) 中的 [MQTT 主題](#) 一節。
2. 確認 MQTT 訊息是否在 MQTT 用戶端內接 AWS IoT 收。
  - a. 開啟 AWS Management Console 並導覽至 [AWS IoT](#)。
  - b. 若要查看 MQTT 測試客戶端，請在導覽列上選擇 Test (測試)，MQTT test client (MQTT 測試客戶端)。
  - c. 對於 Subscribe to a topic (訂閱主題)、Topic filter (主題篩選條件)，請輸入 topic namespace (主題命名空間)。
  - d. 選擇 Subscribe (訂閱)。

MQTT 訊息會顯示在 Subscriptions (訂閱) 與 Topic (主題) 表格中，如下所示。這些訊息可能需要 5 分鐘才會顯示。

Subscribe to a topic
Publish to a topic

**Topic name**  
 The topic name identifies the message. The message payload will be published to this topic with a Quality of S

Q topic/test/

**Message payload**

▶ **Additional configuration**

Publish

Subscriptions

topic/test/

topic/test/	♥ ✕
-------------	-----

▼
topic/test/

```
[
  {
    "timestamp": 1673520691123,
    "message": "Test message 1"
  },
  {
    "timestamp": 1673520692321,
    "message": "Test message 2"
  },
  {
    "timestamp": 1673520693322,
    "message": "Test message 3"
  }
]
```

## 檢視日誌資料

若要檢閱記錄 CloudWatch 檔中的記錄

1. 開啟 AWS Management Console，然後瀏覽至[CloudWatch](#)。
2. 在導覽列上，選擇 Logs (日誌)，Log Insights (日誌深入解析)。
3. 在 [選取記錄群組] 功能表上，選擇您在 AWS IoT 規則中指定的記錄群組。
4. 在 Logs insights (日誌深入解析) 頁面上，選擇 Run query (執行查詢)。

## 使用記錄 AWS IoT API 呼叫 AWS CloudTrail

AWS IoT 與 (提供中的使用者 AWS CloudTrail、角色或服務所採取的動作記錄) 的 AWS 服務整合 AWS IoT。CloudTrail 擷取 AWS IoT 做為事件的所有 API 呼叫，包括來自 AWS IoT 主控台的呼叫，以及從 API 的程 AWS IoT 式碼呼叫。如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 AWS IoT。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷提出要求 AWS IoT、提出要求的 IP 位址、提出要求的人員、提出要求的時間以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使[AWS CloudTrail 用者指南](#)。

## AWS IoT 中的資訊 CloudTrail

CloudTrail 在您創建帳戶 AWS 帳戶 時啟用。當活動發生在中時 AWS IoT，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以查看，搜索和下載最近的事件 AWS 帳戶。如需詳細資訊，請參閱[檢視具有事 CloudTrail 件記錄的事件](#)。

如需您 AWS 帳戶帳戶中正在進行事件的記錄 (包含 AWS IoT 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台中建立追蹤時，追蹤會套用至所有 AWS 區域項目。追蹤記錄來自 AWS 分區中所有 AWS 區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。您可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱：

- [建立追蹤的概觀](#)
- [CloudTrail 支援的服務與整合](#)
- [設定 Amazon SNS 通知 CloudTrail](#)
- [從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail 記錄檔](#)

**Note**

AWS IoT 不會記錄資料平面動作 (裝置端) CloudTrail。用 CloudWatch 於監視這些動作。

一般而言，會記錄進行變更的 AWS IoT 控制平面動作 CloudTrail。呼叫，例如 `CreateThingCreateKeysAndCertificate`，和 `UpdateCertificate` 留下 CloudTrail 條目，而呼叫 (如 `ListThings` 和 `ListTopic規則`) 則不會。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根或 IAM 使用者憑證提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱 [CloudTrail 使用 userIdentity 元素](#)。

AWS IoT 動作記錄在 [AWS IoT API 參考](#) 中。AWS IoT 無線操作記錄在 [AWS IoT 無線 API 參考](#) 中。

## 瞭解 AWS IoT 記錄檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範 `AttachPolicy` 動作的 CloudTrail 記錄項目。

```
{
  "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",
"RequestParameters": {
  "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": ""
}

```

# 的規則 AWS IoT

規則可讓您的裝置進行互動 AWS 服務。規則的分析和動作的執行均是以 MQTT 的主題流為依歸。您可以使用規則來支援下列任務：

- 擴增或篩選從裝置接收的資料
- 將裝置接收的資料寫入 Amazon DynamoDB 資料庫。
- 將檔案儲存至 Amazon S3。
- 使用 Amazon SNS 將推送通知傳送給所有使用者。
- 將資料發佈至 Amazon SQS 佇列。
- 呼叫 Lambda 函數來擷取資料。
- 使用 Amazon Kinesis 來處理來自大量裝置的訊息。
- 將數據發送到 Amazon OpenSearch 服務。
- 擷取 CloudWatch 量度。
- 變更 CloudWatch 鬧鐘。
- 將 MQTT 訊息中的資料傳送 SageMaker 至 Amazon，以根據機器學習 (ML) 模型進行預測。
- 將訊息傳送至 Salesforce IoT 輸入串流。
- 將訊息資料傳送至 AWS IoT Analytics 頻道。
- 開始處理 Step Functions 狀態機器。
- 將消息數據發送到 AWS IoT Events 輸入。
- 將訊息資料傳送至 AWS IoT SiteWise 中的資產屬性。
- 將訊息資料傳送至 Web 應用程式或服務。

您的規則可以使用 MQTT 訊息，這些訊息會通過 [the section called “裝置通訊協定”](#) 支援的發佈/訂閱協定。您也可以使用 [Basic Ingest](#) 功能，將裝置資料安全地傳送到先前 AWS 服務 列出的資料，而不會產生 [訊息](#) 費用。[基本擷取](#) 功能會從擷取路徑移除發佈/訂閱訊息代理程式，使資料流程最佳化。這使得它具有成本效益，同時仍保持的安全性和數據處理功能 AWS IoT。

AWS IoT 在執行這些動作之前，您必須授與其代表您存取 AWS 資源的權限。執行動作時，您會產生您使用的 AWS 服務 標準費用。

## 目錄

- [授與 AWS IoT 規則所需的存取權](#)

- [傳遞角色權限](#)
- [建立規則](#)
- [檢視您的規則](#)
- [刪除規則](#)
- [AWS IoT 規則動作](#)
- [規則疑難排解](#)
- [使用規則存取跨帳戶資 AWS IoT 源](#)
- [錯誤處理 \(錯誤動作\)](#)
- [使用基本擷取減少簡訊費用](#)
- [AWS IoT SQL 參考資料](#)

## 授與 AWS IoT 規則所需的存取權

使用 IAM 角色來控制每個規則可存取的 AWS 資源。在建立規則之前，您必須建立具有允許存取所需 AWS 資源的政策 IAM 角色。AWS IoT 在實作規則時會擔任此角色。

完成以下步驟來建立 IAM 角色和 AWS IoT 政策，以授與 AWS IoT 規則所需的存取權限 (AWS CLI)。

1. 將下列信任原則文件儲存至名為的檔案，該文件會授與擔任角色的 AWS IoT 權限 `iot-role-trust.json`。

```
{
  "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:rule/  
rulename"
        }
      }
    }
  ]
}
```

```

    }
  }
]
}

```

使用 [create-role](#) 命令來建立用於指定 `iot-role-trust.json` 檔案的 IAM 角色：

```
aws iam create-role --role-name my-iot-role --assume-role-policy-document
file:///iot-role-trust.json
```

此命令的輸出結果如下所示：

```
{
  "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. 將下列 JSON 儲存至名為 `my-iot-policy.json` 的檔案。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "dynamodb:*",
      "Resource": "*"
    }
  ]
}
```

此 JSON 是一份範例政策文件，可授與 AWS IoT 管理員對 DynamoDB 的存取權。

使用 [create-policy](#) 命令在擔任該角色時授予對 AWS 資源的 AWS IoT 訪問權限，並傳入文件：`my-iot-policy.json`



```
aws iam create-policy --policy-name my-iot-policy --policy-document file://my-iot-policy.json
```

如需如何 AWS 服務 在原則中授與存取權的詳細資訊 AWS IoT，請參閱[建立規則](#)。

[create-policy](#) 命令的輸出結果含有該政策的 ARN。將政策連接至角色。

```
{
  "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. 請使用 [attach-role-policy](#) 命令將政策連接到您的角色：

```
aws iam attach-role-policy --role-name my-iot-role --policy-arn "arn:aws:iam::123456789012:policy/my-iot-policy"
```

## 傳遞角色權限

IAM 角色是規則定義的一部分，其授予指定於規則動作中資源的存取許可。當叫用規則的動作時，規則引擎會擔任該角色。角色的定義必須與規則 AWS 帳戶 相同。

當建立或替換某項規則時，實際上就是在將某個角色傳送至規則引擎。執行此操作需要 `iam:PassRole` 許可。若要驗證您是否具有此許可，請建立一項授予 `iam:PassRole` 許可的政策，並將其連接至您的 IAM 使用者。以下政策顯示了如何允許角色的 `iam:PassRole` 許可。

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

```

    "Sid": "Stmt1",
    "Effect": "Allow",
    "Action": [
      "iam:PassRole"
    ],
    "Resource": [
      "arn:aws:iam::123456789012:role/myRole"
    ]
  }
]
}

```

在此政策範例中，已將 `iam:PassRole` 許可授予 `myRole` 角色。該角色是使用角色的 ARN 來指定。將此政策連接至您的 IAM 使用者或是您使用者所屬的角色。如需詳細資訊，請參閱[處理受管政策的相關文章](#)。

### Note

Lambda 函數使用的是資源型政策，即是政策直接連接至 Lambda 函數本身。在您建立一個叫用 Lambda 函數的規則時，並不會傳遞角色，因此建立該規則的使用者不需要 `iam:PassRole` 許可。如需 Lambda 函數授權的詳細資訊，請參閱[使用資源政策授予許可](#)。

## 建立規則

您可以建立 AWS IoT 規則來路由連線物件中的資料，以便與其他 AWS 服務互動。AWS IoT 規則由下列元件組成：

### 規則的元件

元件	描述	必要或選擇性
規則名稱	規則的名稱。請注意，我們不建議您在規則名稱中使用可識別個人身份的資訊。	必要。
規則說明	該項規則的文字說明。請注意，我們不建議您在規則說明中使用個人身份信息。	選用。

元件	描述	必要或選擇性
SQL 陳述式	簡化的 SQL 語法，用於篩選 MQTT 主題所收到的訊息，並將資料推送到他處。如需詳細資訊，請參閱 <a href="#">AWS IoT SQL 參考資料</a> 。	必要。
SQL 版本	評估規則時所用的 SQL 規則引擎版本。雖然此項屬性為選用，但強烈建議您指定 SQL 的版本。AWS IoT Core 控制台默認情況下將此屬性 2016-03-23 設置為。如果未設定此性質 (例如在 AWS CLI 指令或樣 AWS CloudFormation 板中)，則會使用 2015-10-08 此性質。如需詳細資訊，請參閱 <a href="#">SQL 版本</a> 。	必要。
一個或多個動作	執行規則時 AWS IoT 會執行動作。例如，您可將資料插入 DynamoDB 表格，將資料寫入 Amazon S3 儲存貯體，發佈至 Amazon SNS 主題，或呼叫 Lambda 函數。	必要。
錯誤動作	當動作無法 AWS IoT 執行規則的動作時執行。	選用。

在建立 AWS IoT 規則之前，您必須建立具有允許存取所需 AWS 資源的政策的 IAM 角色。AWS IoT 在實作規則時會擔任此角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#) 和 [傳遞角色權限](#)。

建立規則時，請注意在主題上發佈的資料量。如果您建立了包含萬用字元主題模式的規則，這些規則可能會比對大部分的訊息。若是這種情況，您可能需要增加目標動作使用的 AWS 資源容量。此外，如果您建立了重新發佈規則，而其中包含萬用字元主題模式，最後可能會造成循環規則，導致無限迴圈。

#### Note

規則的建立和更新是屬於管理員層級的動作。任何擁有許可而能建立或更新規則的使用者，均能夠存取規則所處理的資料。

## 建立規則 (主控台)

### 建立規則 (AWS Management Console)

使用 [AWS Management Console](#) 指令建立規則：

1. 開啟 [AWS IoT 主控台](#)。
2. 在左側導航中，選擇消息路由低至管理部分。然後選擇「規則」。
3. 在 [規則] 頁面上，選擇 [建立規則]。
4. 在 [指定規則屬性] 頁面上，輸入規則的名稱。規則描述和標籤是可選的。選擇下一步。
5. 在「設定 SQL 敘述句」頁面上，選擇 SQL 版本並輸入 SQL 敘述句。一個例子 SQL 語句可以是 `SELECT temperature FROM 'iot/topic' WHERE temperature > 50`。如需詳細資訊，請參閱 [SQL 版本](#) 和 [AWS IoT SQL 參考資料](#)。
6. 在 [附加規則動作] 頁面上，新增規則動作以將資料路由至其他 AWS 服務。
  1. 在規則動作中，從下拉式清單中選取規則動作。例如，您可以選擇「Kinesis 串流」。如需有關規則動作的詳細資訊，請參閱 [AWS IoT 規則動作](#)。
  2. 根據您選擇的規則動作，輸入相關的組態詳細資訊。例如，如果您選擇 Kinesis Stream，則需要選擇或建立資料串流資源，並選擇性地輸入設定詳細資料 (例如分割金鑰)，該資料用於在蒸汽中依碎片分組資料。
  3. 在 IAM 角色中，選擇或建立角色以授與端點的 AWS IoT 存取權。請注意，系統 AWS IoT 會自動建立政策，其前置碼為選取的 IAM 角色 `aws-iot-rule` 下方。您可以選擇「檢視」，從 IAM 主控台檢視您的 IAM 角色和政策。錯誤動作是選擇性的。您可以在 [錯誤處理 \(錯誤動作\)](#) 中找到更多資訊。如需為規則建立 IAM 角色的詳細資訊，請參閱 [授予規則所需的存取權](#)。選擇下一步。
7. 在 [檢閱並建立] 頁面上，檢閱所有組態，並視需要進行編輯。選擇建立。

成功建立規則後，您會看到「規則」頁面上列出的規則。您可以選取規則來開啟「詳細資訊」頁面，您可以在其中檢視規則、編輯規則、停用規則以及刪除規則。

## 建立規則 (CLI)

### 建立規則 (AWS CLI)

請使用 [create-topic-rule](#) 命令來建立規則：

```
aws iot create-topic-rule --rule-name myrule --topic-rule-payload file://myrule.json
```

下列為一個範例承載檔案，其規則會將所有發送至 `iot/test` 主題的訊息插入指定的 DynamoDB 表格中。SQL 陳述式會篩選訊息，而角色 ARN 會授與寫入 DynamoDB 表格的 AWS IoT 權限。

```
{
  "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()}"
      }
    }
  ]
}
```

以下為範例承載檔案，含有一項規則，會將所有發送至 `iot/test` 主題的訊息插入指定的 S3 儲存貯體。SQL 陳述式會篩選訊息，而角色 ARN 會授與寫入 Amazon S3 儲存貯體的 AWS IoT 權限。

```
{
  "awsIotSqlVersion": "2016-03-23",
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "actions": [
    {
      "s3": {
        "roleArn": "arn:aws:iam::123456789012:role/aws_iam_s3",
        "bucketName": "my-bucket",
        "key": "myS3Key"
      }
    }
  ]
}
```

以下是有效負載檔案的範例，其中包含將資料推送至 Amazon OpenSearch 服務的規則：

```
{
```

```

"sql": "SELECT *, timestamp() as timestamp FROM 'iot/test'",
"ruleDisabled": false,
"awsIotSqlVersion": "2016-03-23",
"actions": [
  {
    "OpenSearch": {
      "roleArn": "arn:aws:iam::123456789012:role/aws_iot_es",
      "endpoint": "https://my-endpoint",
      "index": "my-index",
      "type": "my-type",
      "id": "${newuuid()}"
    }
  }
]
}

```

下列為範例承載檔案，具有會呼叫 Lambda 函數的規則：

```

{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "lambda": {
        "functionArn": "arn:aws:lambda:us-west-2:123456789012:function:my-lambda-function"
      }
    }
  ]
}

```

下列為範例承載檔案，具有會發佈至 Amazon SNS 主題的規則：

```

{
  "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"
      }
    }
  ]
}

```

```

}
]
}

```

以下為範例承載檔案，含有的規則會將資料重新發佈至不同的 MQTT 主題：

```

{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "republish": {
        "topic": "my-mqtt-topic",
        "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
      }
    }
  ]
}

```

以下是有效負載檔案的範例，其中包含將資料推送至 Amazon 資料 Firehose 串流的規則：

```

{
  "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"
      }
    }
  ]
}

```

以下是有效負載檔案的範例，其中包含規則，如果 MQTT 承載中的資料分類為 1，則會使用 Amazon SageMaker `machinelearning_predict` 函數重新發佈至主題。

```

{
  "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"
    }
  }
]
```

以下是範例承載檔案，含有的規則會將訊息發佈至 Salesforce IoT 雲端輸入串流：

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "salesforce": {
        "token": "ABCDEFGH123456789abcdefghi123456789",
        "url": "https://ingestion-cluster-id.my-env.sfdcnw.com/streams/stream-id/
connection-id/my-event"
      }
    }
  ]
}
```

以下是具有開始執行 Step Functions 狀態機器之規則的範例承載檔案。

```
{
  "sql": "expression",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "stepFunctions": {
        "stateMachineName": "myCoolStateMachine",
        "executionNamePrefix": "coolRunning",
        "roleArn": "arn:aws:iam::123456789012:role/my-iot-role"
      }
    }
  ]
}
```



```
]
}
```

## 標記您的規則

您可以套用標記，將另一層特殊性新增至新規則或現有規則。標記會利用規則中的索引鍵值配對，讓您更好地控制將規則套用至 AWS IoT 資源和服務的方式和位置。例如，您可以將規則範圍限制為僅適用於進行發行前測試的試用版環境 (Key=environment, Value=beta)，或擷取所有僅從特定端點傳送至 `iot/test` 的主題，並將這兩項資料存放在 Amazon S3 儲存貯體中。

### IAM 政策範例

對於示範如何授予規則標記許可的範例，請考慮使用者執行以下命令，以建立規則並將其標記為僅適用於試用版環境。

在該範例中，替換：

- *MyTopicRuleName* 與規則的名稱。
- *myrule.json* 為政策文件的名稱。

```
aws iot create-topic-rule
  --rule-name MyTopicRuleName
  --topic-rule-payload file://myrule.json
  --tags "environment=beta"
```

這個範例必須使用下列 IAM 政策：

```
{
  "Version": "2012-10-17",
  "Statement":
  {
    "Action": [ "iot:CreateTopicRule", "iot:TagResource" ],
    "Effect": "Allow",
    "Resource": [
      "arn:aws:iot:us-east-1:123456789012:rule/MyTopicRuleName"
    ]
  }
}
```

以上範例說明名為 `MyTopicRuleName` 並僅適用於試用版環境的新建規則。政策聲明中的 `iot:TagResource` 與 `MyTopicRuleName` 特別呼叫，允許在建立或更新 `MyTopicRuleName` 時予以標記。建立規則時使用的參數 `--tags "environment=beta"` 將 `MyTopicRuleName` 範圍限制為只有您的測試版環境。如果您移除參數 `--tags "environment=beta"`，`MyTopicRuleName` 將適用於所有環境。

如需根據 AWS IoT 規則建立 IAM 角色和政策的詳細資料，請參閱 [授與 AWS IoT 規則所需的存取權](#)

如需標記資源的相關資訊，請參閱 [標記您的 AWS IoT 資源](#)。

## 檢視您的規則

請使用 [list-topic-rules](#) 命令來列出規則：

```
aws iot list-topic-rules
```

請使用 [get-topic-rule](#) 命令來取得規則的相關資訊：

```
aws iot get-topic-rule --rule-name myrule
```

## 刪除規則

規則結束使用後即可刪除。

刪除規則 (AWS CLI)

請使用 [delete-topic-rule](#) 命令來刪除規則：

```
aws iot delete-topic-rule --rule-name myrule
```

## AWS IoT 規則動作

AWS IoT 規則動作會指定呼叫規則時要執行的動作。您可以定義動作以將資料傳送到 Amazon DynamoDB 資料庫、將資料傳送到 Amazon Kinesis Data Streams、叫用 AWS Lambda 函數等等。AWS IoT 支援動作服務 AWS 區域 可用的下列動作。

規則動作	描述	API 中的名稱
<a href="#">Apache Kafka</a>	將訊息傳送至 Apache Kafka 叢集。	kafka
<a href="#">CloudWatch 警報</a>	更改 Amazon CloudWatch 警報的狀態。	cloudwatchAlarm
<a href="#">CloudWatch 日誌</a>	傳送訊息至 Amazon CloudWatch 日誌。	cloudwatchLogs
<a href="#">CloudWatch 度量</a>	傳送訊息至 CloudWatch 量度。	cloudwatchMetric
<a href="#">DynamoDB</a>	將訊息傳送至 DynamoDB 表格。	dynamoDB
<a href="#">DynamoDBv2</a>	將訊息資料傳送至 DynamoDB 表格中的多個欄。	dynamoDBv2
<a href="#">Elasticsearch</a>	將消息發送到 OpenSearch 端點。	OpenSearch
<a href="#">HTTP</a>	將訊息發佈至 HTTPS 端點。	http
<a href="#">IoT Analytics</a>	將訊息傳送至 AWS IoT Analytics 頻道。	iotAnalytics
<a href="#">AWS IoT Events</a>	將消息發送到 AWS IoT Events 輸入。	iotEvents
<a href="#">AWS IoT SiteWise</a>	將訊息資料傳送至 AWS IoT SiteWise 資產屬性。	iotSiteWise
<a href="#">Firehose</a>	將訊息傳送至「Firehose」傳送串流。	firehose
<a href="#">Kinesis Data Streams</a>	將訊息傳送至 Kinesis 資料串流。	kinesis

規則動作	描述	API 中的名稱
<a href="#">Lambda</a>	以訊息資料作為輸入呼叫 Lambda 函數。	lambda
<a href="#">位置</a>	將位置資料傳送至 Amazon Location Service。	location
<a href="#">OpenSearch</a>	將訊息傳送至 Amazon OpenSearch 服務端點。	OpenSearch
<a href="#">Republish</a>	可在另一個 MQTT 主題上重新發佈訊息。	republish
<a href="#">S3</a>	將訊息存放於 Amazon Simple Storage Service (Amazon S3) 儲存貯體中。	s3
<a href="#">Salesforce IoT</a>	傳送訊息至 Salesforce IoT 輸入串流。	salesforce
<a href="#">SNS</a>	將訊息發佈為 Amazon Simple Notification Service (Amazon SNS) 推送通知。	sns
<a href="#">SQS</a>	將訊息發佈至 Amazon Simple Queue Service (Amazon SQS) 佇列。	sqs
<a href="#">Step Functions</a>	啟動 AWS Step Functions 狀態機。	stepFunctions
<a href="#">the section called “Timestream”</a>	將訊息傳送至 Amazon Timestream 資料庫表格。	timestream

### 備註

- 將規則定義 AWS 區域 為與其他服務的資源相同，以便規則動作可與該資源互動。

- 如果發生間歇性錯誤，AWS IoT 規則引擎可能會多次嘗試執行動作。如果所有嘗試都失敗，則會捨棄該訊息，並在您的 CloudWatch 記錄檔中提供該錯誤。您可以為失敗發生之後叫用的每一個規則指定一個錯誤動作。如需詳細資訊，請參閱 [錯誤處理 \(錯誤動作\)](#)。
- 某些規則動作會啟動服務中與 AWS Key Management Service (AWS KMS) 整合的動作，以支援靜態資料加密。如果您使用客戶管理 AWS KMS key (KMS 金鑰) 來加密靜態資料，則服務必須具有代表呼叫者使用 KMS 金鑰的權限。若要了解如何管理客戶受管 KMS 金鑰的許可，請參閱適當服務指南中的資料加密主題。如需客戶受管 KMS 金鑰的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [AWS Key Management Service 概念](#)。

## Apache Kafka

Apache 卡夫卡 (卡夫卡) 動作會將訊息直接傳送到您的 Amazon Managed Streaming for Apache Kafka (Amazon MSK)、由第三方供應商管理的 Apache 卡夫卡叢集 (例如 [Confluent 雲端](#)) 或自我管理的 Apache 卡夫卡叢集，以進行資料分析和視覺化。

### Note

本主題假設您熟悉 Apache Kafka 平台及相關概念。如需有關 Apache Kafka 的詳細資訊，請參閱 [Apache Kafka](#)。不支援 [MSK 無伺服器](#)。MSK 無伺服器叢集只能透過 IAM 身份驗證來完成，而 Apache Kafka 規則動作目前不支援該身份驗證。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執

行 `ec2:CreateNetworkInterface`、`ec2:DescribeNetworkInterfaces`、`ec2:CreateNetworkInterface` 的 IAM 角色。此角色會建立並管理您 Amazon Virtual Private Cloud 的彈性網路介面，以達到您的 Kafka 代理程式。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT Core 執行此規則動作的角色。

如需有關網路介面的詳細資訊，請參閱《Amazon EC2 使用者指南》中的 [彈性網路介面](#)。

連接至您指定之角色的政策應如下列範例所示：

```
{
  "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 AWS Secrets Manager 儲存連線到 Kafka 代理程式所需的登入資料，則必須建立 AWS IoT Core 可假設執行 `secretsmanager:GetSecretValue` 和 `secretsmanager:DescribeSecret` 作業的 IAM 角色。

連接至您指定之角色的政策應如下列範例所示：

```
{
  "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-*"
      ]
    }
  ]
}
```

```
]
}
```

- 您可在 Amazon Virtual Private Cloud (Amazon VPC) 內部執行您的 Apache Kafka 叢集。您必須建立 Amazon VPC 目的地，並使用子網路中的 NAT 閘道將訊息從 AWS IoT 轉寄到公用 Kafka 叢集。AWS IoT 規則引擎會在 VPC 目的地中列出的每個子網路中建立網路介面，直接路由傳送流量至 VPC。建立 VPC 目的地時，AWS IoT 規則引擎會自動建立 VPC 規則動作。如需有關 VPC 規則動作的詳細資訊，請參閱 [Virtual private cloud \(VPC\) 目的地](#)。
- 如果您使用客戶管理 AWS KMS key (KMS 金鑰) 來加密靜態資料，則服務必須具有代表呼叫者使用 KMS 金鑰的權限。如需詳細資訊，請參閱《Amazon Managed Streaming for Apache Kafka 開發人員指南》中的 [Amazon MSK 加密](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### destinationArn

VPC 目的地的 Amazon 資源名稱 (ARN)。如需有關建立 VPC 目的地的詳細資訊，請參閱 [Virtual private cloud \(VPC\) 目的地](#)。

### 主題

要傳送至 Kafka 代理程式之訊息的 Kafka 主題。

您可使用替代範本來替代此欄位。如需詳細資訊，請參閱 [the section called “替代範本”](#)。

### 金鑰 (選用)

Kafka 訊息金鑰。

您可使用替代範本來替代此欄位。如需詳細資訊，請參閱 [the section called “替代範本”](#)。

### 標頭 (選用)

您指定的 Kafka 標頭清單。每個標頭都是鍵值對，您可以在建立 Kafka 動作時指定。您可以使用這些標頭將資料從 IoT 用戶端路由到下游 Kafka 叢集，而不需修改訊息承載。

您可使用替代範本來替代此欄位。若要了解如何在 Kafka 動作標頭中傳遞內嵌規則的函數作為替換範本，請參閱 [範例](#)。如需詳細資訊，請參閱 [the section called “替代範本”](#)。

**Note**

不支援二進位格式的標頭。

**分割區 (選用)**

Kafka 訊息分割區。

您可使用替代範本來替代此欄位。如需詳細資訊，請參閱 [the section called “替代範本”](#)。

**clientProperties**

定義 Apache Kafka 生產者用戶端屬性的物件。

**acks (選用)**

生產者要求伺服器在考慮請求完成之前所收到的確認數目。

如果您指定 0 作為值，生產者將不會等待伺服器的任何確認。若伺服器並未收到訊息，生產者不會重試傳送訊息。

有效值：-1、0、1、all。預設值為 1。

**bootstrap.servers**

用來建立 Kafka 叢集初始連線的主機和連接埠配對清單 (例如 host1:port1、host2:port2)。

**compression.type (選用)**

生產者所產生所有資料的壓縮類型。

有效值：none、gzip、snappy、lz4、zstd。預設值為 none。

**security.protocol**

用來連接至您 Kafka 代理程式的安全通訊協定。

有效值：SSL、SASL\_SSL。預設值為 SSL。

**key.serializer**

指定如何將與 ProducerRecord 一起提供的金鑰物件轉換為位元組。

有效值：StringSerializer。



## value.serializer

指定如何將與 `ProducerRecord` 一起提供的值物件轉換為位元組。

有效值：`ByteBufferSerializer`。

## ssl.truststore

base64 格式的信任庫檔案或在 [AWS Secrets Manager](#) 中的信任庫檔案位置。若 Amazon 憑證授權機構 (CA) 信任您的信任存放區，則不需要此值。

此欄位支援替代範本。若您使用 Secrets Manager 儲存連線至 Kafka 代理程式所需的憑證，則可使用 `get_secret` SQL 函數來擷取此欄位的值。如需替代範本的詳細資訊，請參閱 [the section called “替代範本”](#)。如需 `get_secret` SQL 函數的詳細資訊，請參閱 [the section called “get\\_secret\(secretId, secretType, key, roleArn\)”](#)。若信任庫為檔案形式，請使用 `SecretBinary` 參數。若信任庫為字串形式，請使用 `SecretString` 參數。

此值的最大大小為 65 KB。

## ssl.truststore.password

信任庫的密碼。只有在您已建立信任庫的密碼時，才需要此值。

## ssl.keystore

金鑰存放區檔案。當您指定 SSL 為 `security.protocol` 的值時，則需要此值。

此欄位支援替代範本。使用 Secrets Manager 來存放連接至 Kafka 代理程式所需的憑證。若要擷取此欄位的值，請使用 `get_secret` SQL 函數。如需替代範本的詳細資訊，請參閱 [the section called “替代範本”](#)。如需 `get_secret` SQL 函數的詳細資訊，請參閱 [the section called “get\\_secret\(secretId, secretType, key, roleArn\)”](#)。使用 `SecretBinary` 參數。

## ssl.keystore.password

金鑰存放區檔案的存放區密碼。若指定 `ssl.keystore` 的值，則需要此值。

此欄位的值可以是純文字。此欄位也支援替代範本。使用 Secrets Manager 來存放連接至 Kafka 代理程式所需的憑證。若要擷取此欄位的值，請使用 `get_secret` SQL 函數。如需替代範本的詳細資訊，請參閱 [the section called “替代範本”](#)。如需 `get_secret` SQL 函數的詳細資訊，請參閱 [the section called “get\\_secret\(secretId, secretType, key, roleArn\)”](#)。使用 `SecretString` 參數。

## ssl.key.password

金鑰存放區檔案中私鑰的密碼。

此欄位支援替代範本。使用 Secrets Manager 來存放連接至 Kafka 代理程式所需的憑證。若要擷取此欄位的值，請使用 `get_secret` SQL 函數。如需替代範本的詳細資訊，請參閱 [the section called “替代範本”](#)。如需 `get_secret` SQL 函數的詳細資訊，請參閱 [the section called “get\\_secret\(secretId, secretType, key, roleArn\)”](#)。使用 `SecretString` 參數。

### sasl.mechanism

用來連接至 Kafka 代理程式的安全機制。指定 `security.protocol` 的 `SASL_SSL` 時，則需要此值。

有效值：PLAIN、SCRAM-SHA-512、GSSAPI。

#### Note

SCRAM-SHA-512是在 CN-北 -1、CN-西北 -1、-1 和 -1 區域中唯一支援的安全機制。  
us-gov-east us-gov-west

### sasl.plain.username

用來從 Secrets Manager 擷取秘密字串的使用者名稱。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `PLAIN` 時，則需要此值。

### sasl.plain.password

用於從 Secrets Manager 擷取秘密字串的密碼。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `PLAIN` 時，則需要此值。

### sasl.scram.username

用來從 Secrets Manager 擷取秘密字串的使用者名稱。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `SCRAM-SHA-512` 時，則需要此值。

### sasl.scram.password

用於從 Secrets Manager 擷取秘密字串的密碼。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `SCRAM-SHA-512` 時，則需要此值。

### sasl.kerberos.keytab

Secrets Manager 中 Kerberos 驗證的 keytab 檔案。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `GSSAPI` 時，則需要此值。

此欄位支援替代範本。使用 Secrets Manager 來存放連接至 Kafka 代理程式所需的憑證。若要擷取此欄位的值，請使用 `get_secret` SQL 函數。如需替代範本的詳細資訊，請參閱 [the section called “替代範本”](#)。如需 `get_secret` SQL 函數的詳細資訊，請參閱 [the section called “get\\_secret\(secretId, secretType, key, roleArn\)”](#)。使用 `SecretBinary` 參數。

`sasl.kerberos.service.name`

在 Apache Kafka 執行之下的 Kerberos 委託人名稱。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `GSSAPI` 時，則需要此值。

`sasl.kerberos.krb5.kdc`

您的 Apache Kafka 生產者用戶端連線至金鑰分配中心 (KDC) 的主機名稱。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `GSSAPI` 時，則需要此值。

`sasl.kerberos.krb5.realm`

您的 Apache Kafka 生產者用戶端連線的領域。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `GSSAPI` 時，則需要此值。

`sasl.kerberos.principal`

Kerberos 可指派票證來存取 Kerberos 感知服務的唯一 Kerberos 身分。指定 `security.protocol` 的 `SASL_SSL` 和 `sasl.mechanism` 的 `GSSAPI` 時，則需要此值。

## 範例

下列 JSON 範例會定義規則中的 Apache 卡夫卡動作 AWS IoT。以下範例會在 Kafka 動作標頭中傳遞 `sourceIp()` 內嵌函數作為 [替換範本](#)。

```
{
  "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"
  },
  "headers": [
    {
      "key": "static_header_key",
      "value": "static_header_value"
    },
    {
      "key": "substitutable_header_key",
      "value": "${value_from_payload}"
    },
    {
      "key": "source_ip",
      "value": "${sourceIp()}"
    }
  ]
}
]
}
}
}
}
}
}
}

```

### 關於您 Kerberos 設定的重要注意事項

- 您的金鑰分配中心 (KDC) 必須透過目標 VPC 內的私人網域名稱系統 (DNS) 進行解析。其中一種可能的方法是將 KDC DNS 項目新增至私有託管區域。如需此方法的詳細資訊，請參閱[使用私有託管區域](#)。
- 每個 VPC 皆必須啟用 DNS 解析。如需詳細資訊，請參閱[以 VPC 使用 DNS](#)。
- VPC 目的地中的網路介面安全群組和執行個體層級安全群組必須允許 VPC 內下列連接埠上的流量。
  - 引導代理程式接聽程式連接埠上的 TCP 流量 (通常是 9092，但必須在 9000-9100 範圍內)
  - KDC 連接埠 88 上的 TCP 和 UDP 流量

- SCRAM-SHA-512是在 CN-北 -1、CN-西北 -1、-1 和 -1 區域中唯一支援的安全機制。 us-gov-east us-gov-west

## Virtual private cloud (VPC) 目的地

Apache Kafka 規則動作將資料路由至 Amazon Virtual Private Cloud (Amazon VPC) 中的 Apache Kafka 叢集。在為規則動作指定 VPC 目的地時，系統會自動啟用 Apache Kafka 規則動作所使用的 VPC 組態。

VPC 目的地包含一份 VPC 內的子網路清單。規則引擎會在您於此清單中指定的每個子網路中建立彈性網路介面。如需有關網路介面的詳細資訊，請參閱《Amazon EC2 使用者指南》中的[彈性網路介面](#)。

### 需求和考量事項

- 如果您使用的是將由公有端點透過網際網路存取自我管理 Apache Kafka 叢集：
  - 為子網路中的執行個體建立 NAT 閘道。NAT 閘道具有可連接到網際網路的公有 IP 地址，允許規則引擎將您的訊息轉發到公有 Kafka 叢集。
  - 透過 VPC 目的地建立的彈性網路介面 (ENI) 分配彈性 IP 地址。您使用的安全群組必須配置為封鎖傳入流量。

#### Note

如果 VPC 目的地停用後重新啟用，必須將彈性 IP 與新的 ENI 重新關聯。

- 若 VPC 主題規則目的地連續 30 天未收到任何流量，該目的地將遭到停用。
- 如果 VPC 目的地使用的任何資源發生變更，則目的地會遭到停用且無法使用。
- 可停用 VPC 目的地的一些變更包括：刪除 VPC、子網路、安全群組或使用的角色；修改角色，使其不再具有必要許可；以及停用目的地。

### 定價

基於定價目的，除了資源位於您 VPC 中時將訊息傳送至資源之外，還會計量 VPC 規則動作。如需定價資訊，請參閱 [AWS IoT Core 定價](#)。

### 建立 Virtual Private Cloud (VPC) 主題規則目的地

您可以使用 [CreateTopicRuleDestination](#) API 或 AWS IoT Core 主控台建立虛擬私有雲 (VPC) 目的地。

建立 VPC 目的地時，請務必指定下列資訊：

## vpId

VPC 目的地的唯一 ID。

## subnetIds

規則引擎在其中建立彈性網路介面的子網路清單。規則引擎會為清單中的每個子網路配置一個單一網路介面。

## securityGroups (選用)

套用至網路介面的安全性群組清單。

## roleArn

有權代您建立網路介面的角色的 Amazon 資源名稱 (ARN)

此 ARN 應連接至一個看似於下列範例的政策。

```
{
  "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"
        }
      }
    }
  ]
}

```

## 使用建立 VPC 目的地 AWS CLI

下列範例會顯示如何使用 AWS CLI 建立 VPC 目的地。

```

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"}'

```

執行此命令後，VPC 目的地狀態為 IN\_PROGRESS。幾分鐘後，其狀態會變更為 ERROR (若命令不成功) 或 ENABLED。目的地狀態為 ENABLED 時，即可使用。

您可使用下列命令來取得 VPC 目的地的狀態。

```

aws --region region iot get-topic-rule-destination --arn "VPCDestinationARN"

```

## 使用主控台建立 VPC 目的 AWS IoT Core 地

下列步驟說明如何使用 AWS IoT Core 主控台建立 VPC 目的地。

1. 導覽至主 AWS IoT Core 控制台。在左窗格的動作索引標籤上，選擇目的地。

2. 請輸入下列欄位的值。
  - VPC ID
  - 子網路 ID
  - 安全群組
3. 選取具有建立網路介面所需許可的角色。上述範例政策包含這些許可。

VPC 目的地狀態為 ENABLED 時，即可使用。

## CloudWatch 警報

CloudWatch 警報 (cloudWatchAlarm) 動作會變更 Amazon CloudWatch 警示的狀態。您可以指定此呼叫狀態變更的原因和值。

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `cloudwatch:SetAlarmState` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

### 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

`alarmName`

CloudWatch 警示名稱。

支持 [替換模板](#)：API 和 AWS CLI 僅

`stateReason`

警示變更的原因。

支援 [替代範本](#)：是

`stateValue`

警示狀態的值。有效值：OK、ALARM、INSUFFICIENT\_DATA。



支援[替代範本](#)：是

roleArn

允許存取 CloudWatch 警示的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援[替代範本](#)：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的 CloudWatch 警示動作。

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

## 另請參閱

- [什麼是 Amazon CloudWatch？](#) 在 Amazon 用 CloudWatch 戶指南
- 在 [Amazon 用 CloudWatch 戶指南中使用 Amazon CloudWatch 警報](#)

## CloudWatch 日誌

CloudWatch 日誌 (cloudwatchLogs) 動作會將資料傳送到 Amazon CloudWatch 日誌。您可以使用 batchMode，以一則訊息上傳多個裝置日誌記錄並加上時間戳記。您也可以指定動作傳送資料的日誌群組。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `logs:CreateLogStream`、`logs:DescribeLogStreams` 和 `logs:PutLogEvents` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用客戶管理 AWS KMS key (KMS 金鑰) 來加密 CloudWatch 記錄檔中的記錄資料，則服務必須具有代表呼叫者使用 KMS 金鑰的權限。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的使用加密 CloudWatch 日誌 AWS KMS [中的日誌資料](#)。

## batchMode 的 MQTT 訊息格式需求

如果您在關 batchMode 閉的情況下使用「CloudWatch 記錄檔」規則動作，則不需要 MQTT 訊息格式設定。(注意：batchMode 參數的預設值為 false。)不過，如果您在 batchMode 開啟的情況下使用「CloudWatch 記錄檔」規則動作 (參數值為 true)，則包含裝置端記錄檔的 MQTT 訊息必須格式化為包含時間戳記和訊息承載。注意：timestamp 代表事件發生的時間，以 1970 年 1 月 1 日 00:00:00 UTC 起算的毫秒數表示。

以下是發佈格式的範例：

```
[
  {"timestamp": 1673520691093, "message": "Test message 1"},
  {"timestamp": 1673520692879, "message": "Test message 2"},
  {"timestamp": 1673520693442, "message": "Test message 3"}
]
```

根據裝置端日誌的產生方式，這些日誌可能需要先篩選並重新格式化才能傳送，以符合此需求。如需詳細資訊，請參閱 [MQTT 訊息承載](#)。

與 batchMode 參數無關，message 內容必須符合 AWS IoT 消息大小限制。如需詳細資訊，請參閱 [AWS IoT Core 端點和配額](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

## logGroupName

動作傳送資料的 CloudWatch 記錄群組。

支持[替換模板](#)：API 和 AWS CLI 僅

## roleArn

允許存取 CloudWatch 日誌群組的 IAM 角色。如需詳細資訊，請參閱[要求](#)。

支援[替代範本](#)：否

## (選用) batchSize

指出是否要擷取記錄檔記錄批次並將其上傳至中 CloudWatch。值包括 true 或 false (預設值)。如需詳細資訊，請參閱[要求](#)。

支援[替代範本](#)：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的「CloudWatch 記錄檔」動作。

```
{
  "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",
          "batchSize": false
        }
      }
    ]
  }
}
```

## 另請參閱

- [什麼是 Amazon CloudWatch 日誌？](#) 在 Amazon CloudWatch 日誌用戶指南

## CloudWatch 度量

指 CloudWatch 標 (cloudwatchMetric) 動作會擷取 Amazon CloudWatch 指標。您可以指定指標命名空間、名稱、值、單位、時間戳記。

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行cloudwatch:PutMetricData作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

### 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

metricName

CloudWatch 測量結果名稱。

支援[替代範本](#)：是

metricNamespace

CloudWatch 測量結果命名空間名稱。

支援[替代範本](#)：是

metricUnit

支援的公制單位 CloudWatch。

支援[替代範本](#)：是

metricValue

包含 CloudWatch 測量結果值的字串。

支援[替代範本](#)：是

metricTimestamp

(選用) 包含 Unix epoch 時間中時間戳記 (以秒為單位來表達) 的字串。預設為目前的 Unix epoch 時間。

支援替代範本：是

roleArn

允許存取 CloudWatch 指標的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援替代範本：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的 CloudWatch 量度動作。

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

下列 JSON 範例會在 AWS IoT 規則中使用替代範本定義 CloudWatch 量度動作。

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

## 另請參閱

- [什麼是 Amazon CloudWatch ?](#) 在 Amazon 用 CloudWatch 戶指南
- 在 [Amazon 用戶指南中使 CloudWatch 用 Amazon 指 CloudWatch 標](#)

## DynamoDB

DynamoDB (dynamoDB) 動作會將 MQTT 訊息的全部或部分寫入 Amazon DynamoDB 表格。

您可依循對您展示如何使用 DynamoDB 動作 來建立及測試規則的教學課程。如需詳細資訊，請參閱 [教學課程：將裝置資料儲存在 DynamoDB 表格中](#)。

### Note

此規則會以二進位資料形式將非 JSON 資料寫入 DynamoDB 中。DynamoDB 主控台會以 Base64 編碼文字來顯示資料。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `dynamodb:PutItem` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用客戶管理 AWS KMS key (KMS 金鑰) 來加密 DynamoDB 中的靜態資料，則服務必須具有代表呼叫者使用 KMS 金鑰的權限。如需詳細資訊，請參閱《Amazon DynamoDB 入門指南》中的 [客戶管理 KMS 金鑰](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### tableName

DynamoDB 資料表的名稱。

支持[替換模板](#)：API 和 AWS CLI 僅

### hashKeyField

雜湊索引鍵 (也稱為分割區索引鍵) 的名稱。

支持[替換模板](#)：API 和 AWS CLI 僅

### hashKeyType

(選用) 雜湊索引鍵 (也稱為分割區索引鍵) 的資料類型。有效值：STRING、NUMBER。

支持[替換模板](#)：API 和 AWS CLI 僅

### hashKeyValue

雜湊索引鍵的值。考慮使用替代範本，例如 `${topic()}` 或 `${timestamp()}`。

支援[替代範本](#)：是

### rangeKeyField

(選用) 範圍索引鍵 (亦稱為排序索引鍵) 的名稱。

支持[替換模板](#)：API 和 AWS CLI 僅

### rangeKeyType

(選用) 範圍索引鍵 (亦稱為排序索引鍵) 的資料類型。有效值：STRING、NUMBER。

支持[替換模板](#)：API 和 AWS CLI 僅

### rangeKeyValue

(選用) 範圍索引鍵的值。考慮使用替代範本，例如 `${topic()}` 或 `${timestamp()}`。

支援[替代範本](#)：是

### payloadField

(選用) 承載寫入的欄名稱。若省略此值，則會將承載寫入名為 `payload` 的欄。

支援替代範本：是

operation

(選用) 欲執行的作業類型。有效值：INSERT、UPDATE、DELETE。

支援替代範本：是

roleARN

允許存取 DynamoDB 資料表的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援替代範本：否

寫入 DynamoDB 表格的資料產自於規則的 SQL 陳述式。

## 範例

下列 JSON 範例會定義規則中的動作。AWS IoT

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

## 另請參閱

- 《Amazon DynamoDB 開發人員指南》中的 [什麼是 Amazon DynamoDB ?](#)



- 《Amazon DynamoDB 開發人員指南》中的 [DynamoDB 入門](#)
- [教學課程：將裝置資料儲存在 DynamoDB 表格中](#)

## DynamoDBv2

DynamoDBv2 (dynamoDBv2) 動作會將 MQTT 訊息的全部或部分寫入 Amazon DynamoDB 表格。承載中的每個屬性都會寫入 DynamoDB 資料庫中不同的欄。

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `dynamodb:PutItem` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- MQTT 訊息承載一定要包含一個與該表格主要分割區索引鍵相符的根層級索引鍵，以及一個與該表格的主要排序索引鍵相符的根層級索引鍵 (若有定義)。
- 如果您使用客戶管理 AWS KMS key (KMS 金鑰) 來加密 DynamoDB 中的靜態資料，則服務必須具有代表呼叫者使用 KMS 金鑰的權限。如需詳細資訊，請參閱《Amazon DynamoDB 入門指南》中的 [客戶管理 KMS 金鑰](#)。

### 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

#### `putItem`

指定訊息資料將寫入的 DynamoDB 表格物件。此物件必須包含下列資訊：

#### `tableName`

DynamoDB 資料表的名稱。

支持 [替換模板](#)：API 和 AWS CLI 僅

#### `roleARN`

允許存取 DynamoDB 資料表的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援[替代範本](#)：否

寫入 DynamoDB 表格的資料產自於規則的 SQL 陳述式。

## 範例

下列 JSON 範例會定義規則中的動作。AWS IoT

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

下列 JSON 範例會在規則中使用替代範本定義 DynamoDB 動作。AWS IoT

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

```
    ]  
  }  
}
```

## 另請參閱

- 《Amazon DynamoDB 開發人員指南》中的 [什麼是 Amazon DynamoDB ?](#)
- 《Amazon DynamoDB 開發人員指南》中的 [DynamoDB 入門](#)

## Elasticsearch

彈性搜尋 (elasticsearch) 動作會將 MQTT 訊息中的資料寫入 Amazon 服務網域。OpenSearch 然後，您可以使用 OpenSearch 儀表板等工具查詢和視覺化 OpenSearch 服務中的資料。

### Warning

Elasticsearch 動作只能由現有規則動作使用。若要建立新的規則動作或更新現有的規則動作，請改用 OpenSearch 規則動作。如需詳細資訊，請參閱 [OpenSearch](#)。

## 需求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `es:ESHttpPost` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用客戶管理 AWS KMS key (KMS 金鑰) 來加密中的靜態資料 OpenSearch，則服務必須具有代表呼叫者使用 KMS 金鑰的權限。如需詳細資訊，請參閱 Amazon 服務開發人員指南中的 Amazon OpenSearch 服務 [OpenSearch 靜態資料加密](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### endpoint

您的服務網域端點。

支持[替換模板](#)：API 和 AWS CLI 僅

index

您想儲存資料的索引。

支援[替代範本](#)：是

type

欲存放文件的類型。

支援[替代範本](#)：是

id

各文件的專屬識別符。

支援[替代範本](#)：是

roleARN

允許存取 OpenSearch 服務網域的 IAM 角色。如需詳細資訊，請參閱[需求](#)。

支援[替代範本](#)：否

## 範例

下面的 JSON 範例定義 AWS IoT 規則中的 Elasticsearch 動作，及您可如何指定 elasticsearch 動作的欄位。如需詳細資訊，請參閱[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"
        }
      }
    ]
  }
}
```

```

    }
  ]
}
}

```

下列 JSON 範例會在規則中定義具有替代範本的彈性搜尋動作。AWS IoT

```

{
  "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"
        }
      }
    ]
  }
}

```

## 另請參閱

- [OpenSearch](#)
- [什麼是 Amazon OpenSearch 服務？](#)

## HTTP

HTTPS (http) 動作會將資料從 MQTT 訊息傳送至 Web 應用程式或服務。

### 要求

此規則動作具有下列需求：

- 您必須先確認並啟用 HTTPS 端點，才可讓規則引擎使用它們。如需詳細資訊，請參閱 [使用 HTTP 主題規則目的地](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### url

使用 HTTP POST 方法傳送訊息的 HTTPS 端點。若您使用 IP 地址取代主機名稱，其必須為 IPv4 地址。不支援 IPv6 地址。

支援替代範本：是

### confirmationUrl

(選擇性) 如果有指定，AWS IoT 會使用確認 URL 來建立相符的主題規則目的地。您必須先啟用主題規則目的地，才能在 HTTP 動作中進行使用。如需詳細資訊，請參閱 [使用 HTTP 主題規則目的地](#)。如果您使用替代範本，則必須手動建立主題規則目的地，然後才能使用 http 動作。confirmationUrl 必須是 url 的字首。

url 與 confirmationUrl 之間的關係如下所述：

- 如果url是硬式編碼confirmationUrl且未提供，我們會隱含地將url欄位視為confirmationUrl AWS IoT 會建立的主題規則目標url。
- 如果url和confirmationUrl是硬編碼的，則url必須以confirmationUrl. AWS IoT 會建立的主題規則目標confirmationUrl。
- 如果 url 包含替代範本，則您必須指定 confirmationUrl 且 url 必須以 confirmationUrl 開頭。如果 confirmationUrl 包含替代範本，則您必須手動建立主題規則目的地，然後才能使用 http 動作。如果confirmationUrl不包含替代範本，請 AWS IoT 建立的主題規則目標confirmationUrl。

支援替代範本：是

### headers

(選用) 要包含於端點之 HTTP 請求中的標頭清單。每個標頭必須包含下列資訊：

#### key

標頭的金鑰。

支援替代範本：否

#### value

標頭的值。

支援替代範本：是

**Note**

當承載採用 JSON 格式時，預設內容類型為 application/json。否則，它是 application/octet-stream。您可以在標頭中使用重要的 content-type 指定確切的內容類型 (不區分大小寫) 來覆寫它。

## auth

(選用) 規則引擎用來連線至 url 引數中所指定端點 URL 的身分驗證。目前，Signature 第 4 版是唯一支援的身分驗證類型。如需詳細資訊，請參閱 [HTTP 授權](#)。

支援替代範本：否

## 範例

下列 JSON 範例會使用 HTTP 動作定義 AWS IoT 規則。

```
{
  "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 動作重試邏輯

AWS IoT 規則引擎會根據下列規則重試 HTTP 動作：

- 規則引擎會嘗試傳送訊息至少一次。
- 規則引擎最多重試兩次。嘗試次數上限為三次。
- 在下列情況下，規則引擎不會嘗試重試：
  - 上一次嘗試提供了大於 16,384 個位元組的回應。
  - 下游 Web 服務或應用程式在嘗試之後關閉 TCP 連線。
  - 完成請求的總時間超過請求逾時限制。
  - 請求傳回了 429、500-599 以外的 HTTP 狀態碼。

### Note

[標準資料傳輸成本](#)適用於重試。

## 另請參閱

- [使用 HTTP 主題規則目的地](#)
- 在 AWS 部落格上的物聯網中，[AWS IoT Core 將資料直接從您的網路服務路由傳送至您的 Web](#)

## 使用 HTTP 主題規則目的地

HTTP 主題規則目的地是規則引擎可以從主題規則路由資料的 Web 服務。資 AWS IoT Core 源描述的 Web 服務 AWS IoT。主題規則目標資源可以由不同的規則共用。

在 AWS IoT Core 可以將數據發送到另一個 Web 服務之前，它必須確認它可以訪問該服務的端點。

## HTTP 主題規則目的地概觀

HTTP 主題規則目的地是指支援確認 URL 和一個或多個資料收集 URL 的 Web 服務。HTTP 主題規則目的地資源包含 Web 服務的確切 URL。設定 HTTP 主題規則動作時，指定應接收資料的端點的實際



URL 以及 Web 服務的確認 URL。在確認了您的目的地之後，主題規則會將 SQL 陳述式的結果傳送至 HTTPS 端點 (而非確認 URL)。

HTTP 主題規則目的地可以處於下列其中一種狀態：

#### ENABLED

目的地已確認，且可由規則動作使用。目的地必須處於 ENABLED 狀態，才能在規則中使用它。您只能啟用處於 DISABLED 狀態的目的地。

#### DISABLED

目的地已確認，但無法由規則動作使用。如果您想暫時防止流量傳至端點，而不必再次進行確認程序，這會很有用。您只能停用處於 ENABLED 狀態的目的地。

#### IN\_PROGRESS

正在確認目的地。

#### ERROR

目的地確認逾時。

在確認並啟用 HTTP 主題規則目的地之後，即可在帳戶中搭配任何規則使用此目的地。

下列各節說明有關 HTTP 主題規則目的地的常見動作。

#### 建立和確認 HTTP 主題規則目的地

您可以呼叫 `CreateTopicRuleDestination` 操作或使用 AWS IoT 主控台來建立 HTTP 主題規則目的地。

建立目的地後，AWS IoT 會傳送確認要求至確認 URL。確認請求的格式如下：

```
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"
}
```

確認請求的內容包含下列資訊：

arn

要確認的主題規則目的地的 Amazon 資源名稱 (ARN)。

confirmationToken

由傳送の確認權杖 AWS IoT Core。範例中的字符會被截斷。您的字符將更長。您將需要此令牌來確認您的目的地 AWS IoT Core。

enableUrl

您瀏覽以確認主題規則目的地的 URL。

messageType

訊息的類型。

若要完成端點確認程序，您必須在確認 URL 收到確認請求後執行下列其中一項動作。

- 在確認請求中呼叫 `enableUrl`，然後呼叫 `UpdateTopicRuleDestination`，將主題規則的狀態設為 `ENABLED`。
- 呼叫 `ConfirmTopicRuleDestination` 操作並從確認請求傳遞 `confirmationToken`。
- 複製 `confirmationToken` 並將其粘貼到 AWS IoT 控制台中的目標確認對話框中。

傳送新的確認請求

若要啟動目的地的新確認訊息，請呼叫 `UpdateTopicRuleDestination`，並將主題規則目的地的狀態設為 `IN_PROGRESS`。

在傳送新的確認請求之後重複確認程序。

停用和刪除主題規則目的地

若要停用目的地，請呼叫 `UpdateTopicRuleDestination`，並將主題規則目的地的狀態設為 `DISABLED`。處於 `DISABLED` 狀態的主題規則可以再次啟用，無需傳送新的確認請求。

若要刪除主題規則目的地，請呼叫 `DeleteTopicRuleDestination`。

主題規則目的地中 HTTPS 端點支援的憑證授權機構

下列憑證授權機構受到主題規則目的地中的 HTTPS 端點所支援。您可以選擇下列其中一個支援的憑證授權機構。簽章僅供參考。請注意，您無法使用自我簽署憑證，因為它們無法運作。

 協助我們改善此主題

[讓我們了解您的想法。](#)

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:14:3`

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:7`

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:A3:8`

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:D0:6`

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:55:7
```

```
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:4
```

```
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:4
```

```
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:6
```

```
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:D
```

```
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:A
```

```
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:F

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:7

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:B3:4

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:E

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:42:16:1

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:B

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:C

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:8

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:F2:3

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:0

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:C7:01:6

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:C1:5

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:A

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:D7:60:6

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:25:89:70:9

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:A

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:B

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:98:19:6

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:9

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:A3:3
```

```
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:A
```

```
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:9
```

```
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:46:8
```

```
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:38:9
```

```
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:41:5
```

```
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:D

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:7

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:72:4

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:D

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:D

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:A4:7

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:C

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:7

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:32:31:3

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:F8:07:0

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:C

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:D

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:D8:81:21:3

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:D0:60:1

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:72:4

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:E1:8

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:02:8

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:9

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:9

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:C

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:F

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:3

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:B6:5

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:62:9

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:F

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:4

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:B

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:9

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:B2:4

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:A7:96:87:9

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:D

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:7

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:A1:2

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:E

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:F

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:D9:2

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:C4:40:8

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:F

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:89:2

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:22:8

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:C

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:B

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:D3:4

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:88:7

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:15:5
```

```
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:C
```

```
Alias name: usertrustsacertificationauthority
```

```
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:D
```

```
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:F3:3
```

```
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:83:86:0
```

```
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:3
```

```
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:F

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:3

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:B

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:B

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:F

Alias name: turktrustelektroniksertifikahizmetisaglayicisi5

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:7

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:C

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:B

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:3

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:B9:1

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:D

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:1

Alias name: tubitakkamussslkoksertifikasisurum1

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:86:79:90:64:87:1

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:B4:1

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:A

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:D

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:F

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:D3:48:8

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:B2:4

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:B

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:C8:99:8

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:18:4

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:D

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:D7:6

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:E

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:03:9

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:48:3

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:E5:8

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:B

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:17:05:25:5

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:A

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:A
```

```
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:F
```

```
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:C
```

```
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:F
```

```
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:4
```

```
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:6
```

```
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:A

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:C

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:B4:1

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:D4:53:1

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:F7:91:24:5

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:C1:8

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:F

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:B1:7

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:C9:6

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:B1:7

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:E2:57:5

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:E

Alias name: eecertificationcentrерootca

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:7

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:4

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:6

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:10:55:31:6

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:01:0

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:D

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:C

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:F5:9
```

```
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:3
```

```
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:7
```

```
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:8
```

```
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:D9:2
```

```
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:4
```

```
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:E

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:E

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:F3:3

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:7

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:A

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:A

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:C

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:A8:88:63:8

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:D

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:D8:14:36:2

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:A

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:9

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:3

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:C5:3

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:F4:2

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:B6:71:6

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:B

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:D6:77:07:3

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:E

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:E
```

```
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:84:88:82:14:2
```

```
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:F3:7
```

```
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:6
```

```
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:1
```

```
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:C7:43:3
```

```
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:F

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:4

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:14:03:0

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:C6:32:5

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:C5:3

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:9

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:B8:84:2

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:E

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:9

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:9

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:D1:2

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:D

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:A

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:5

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:D6:93:55:59:6

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:D0:6

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:9

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:A7:3

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:D

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:B8:0
```

```
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:B
```

```
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:B4:02:3
```

```
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:F
```

```
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:2
```

```
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:C
```

```
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:17:05:25:5

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:D1:04:38:41:1

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:C

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:6

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:C

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:F

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:6

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:9

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:D

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:C

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:D

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:D7:6

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:B

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:84:88:82:14:2

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:3

## IoT Analytics

AWS IoT Analytics (iotAnalytics) 動作會將 MQTT 訊息中的資料傳送至通道。AWS IoT Analytics

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `iotanalytics:BatchPutMessage` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

連接至您指定角色的政策應如下列範例所示：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iotanalytics:BatchPutMessage",
      "Resource": [
        "arn:aws:iotanalytics:us-west-2:account-id:channel/mychannel"
      ]
    }
  ]
}
```

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### batchMode

(選用) 是否將動作作為批次處理。預設值為 `false`。

當 `batchMode` 是 `true` 並且規則 SQL 語句評估為 Array 時，每個 Array 元素在傳遞給通 AWS IoT Analytics 道時會作為單獨 [BatchPutMessage](#) 的消息傳遞。產生的陣列不能含有超過 100 則訊息。

支援 [替代範本](#)：否

### channelName

要寫入資料的 AWS IoT Analytics 通道名稱。

支援 [替換模板](#)：API 和 AWS CLI 僅

### roleArn

允許存取 AWS IoT Analytics 頻道的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援 [替代範本](#)：否

## 範例

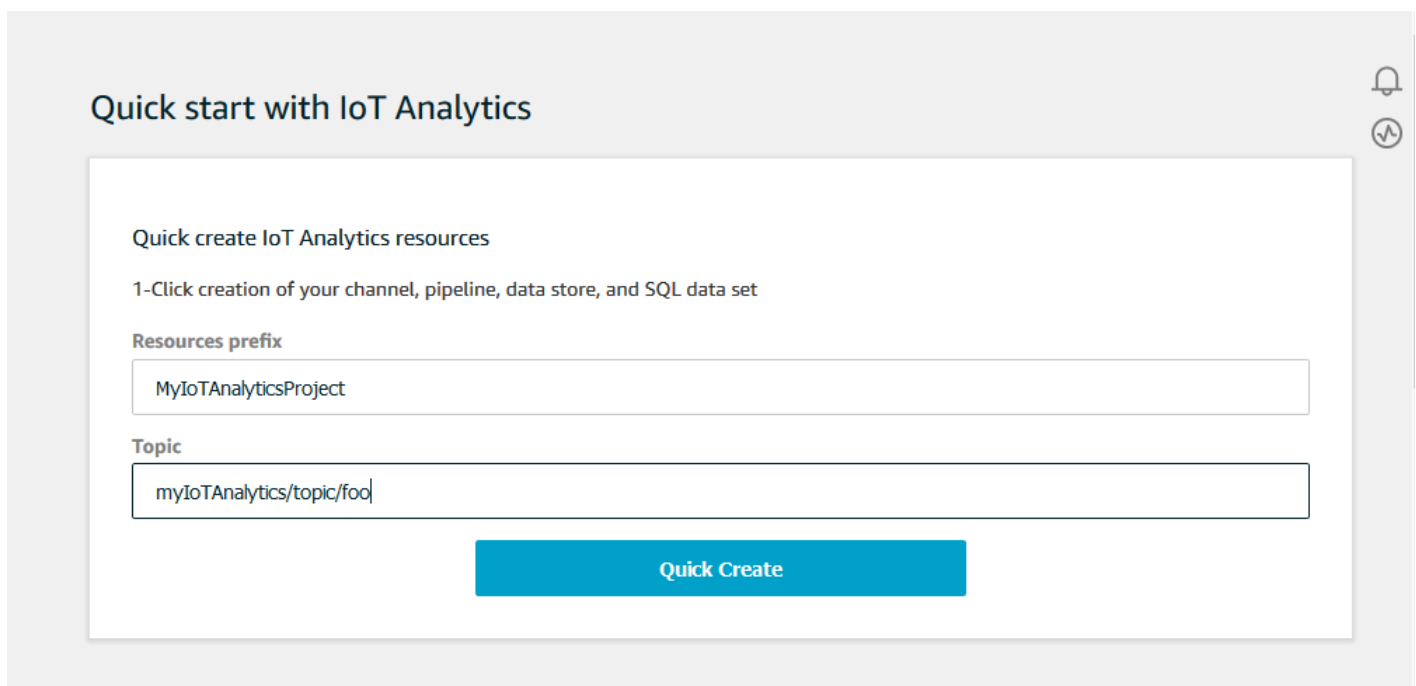
下列 JSON 範例會定義 AWS IoT 規則中的 AWS IoT Analytics 動作。

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "iotAnalytics": {
          "channelName": "mychannel",
          "roleArn": "arn:aws:iam::123456789012:role/analyticsRole",
        }
      }
    ]
  }
}
```

```
}
```

## 另請參閱

- [什麼是 AWS IoT Analytics ?](#) 《AWS IoT Analytics 使用者指南》中的
- 該 AWS IoT Analytics 控制台還具有快速啟動功能，可讓您按一下即可建立通道、資料存放區、管道和資料存放區。如需詳細資訊，請參閱《AWS IoT Analytics 使用者指南》中的 [AWS IoT Analytics 主控台快速入門指南](#)。



## AWS IoT Events

AWS IoT Events (iotEvents) 動作會將 MQTT 訊息中的資料傳送至輸入 AWS IoT Events 。

### **⚠ Important**

如果將裝載傳送到 AWS IoT Core 沒有 Input attribute Key，或者金鑰不在金鑰中指定的相同 JSON 路徑中，則會造成 IoT 規則失敗並顯示錯誤 Failed to send message to Iot Events。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `iotevents:BatchPutMessage` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### batchMode

(選用) 是否將事件動作作為批次處理。預設值為 `false`。

若 `batchMode` 為 `true` 且規則 SQL 陳述式評估為 `Array` 時，當每個 `Array` 元素在透過呼叫 [BatchPutMessage](#) 傳送至 AWS IoT 事件時，會將其視為單獨的訊息。產生的陣列不能含有超過 10 則訊息。

如果 `batchMode` 是 `true`，您無法指定 `messageId`。

支援 [替代範本](#)：否

### inputName

AWS IoT Events 輸入的名稱。

支持 [替換模板](#)：API 和 AWS CLI 僅

### messageId

(可選) 使用此選項可驗證 AWS IoT Events 檢測器僅處理具有給定 `messageId` 的一個輸入 (消息)。您可使用 `${newuuid()}` 替代範本，為每個請求產生一個唯一 ID。

當 `batchMode` 為 `true` 時，您無法指定 `messageId`：將會指派新的 UUID 值。

支援 [替代範本](#)：是

### roleArn

允許 AWS IoT 將輸入傳送至 AWS IoT Events 偵測器的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援 [替代範本](#)：否

## 範例

下面的 JSON 範例定義 AWS IoT 規則中的 IoT Events 動作。



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

## 另請參閱

- [什麼是 AWS IoT Events ?](#) 在AWS IoT Events 開發人員指南

## AWS IoT SiteWise

AWS IoT SiteWise (iotSiteWise) 動作會將 MQTT 訊息中的資料傳送至中的資產屬性。AWS IoT SiteWise

您可以按照教程進行操作，向您展示如何從 AWS IoT 事物中擷取數據。如需詳細資訊，請參閱《使用指南》中的 [〈AWS IoT SiteWise 從 AWS IoT 事物擷取資料至〉教學課程](#)或[使用 AWS IoT 核心規則擷取資料—AWS IoT SiteWise 節](#)。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行*iotSiteWise:BatchPutAssetPropertyValue*作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

您可連接下列範例信任政策至該角色。

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

```

    "Statement": [
      {
        "Effect": "Allow",
        "Action": "iotsitewise:BatchPutAssetPropertyValue",
        "Resource": "*"
      }
    ]
  }

```

若要改善安全Condition性，您可以在屬性中指定 AWS IoT SiteWise 資產階層路徑。下列範例是指定資產階層路徑的信任政策。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iotsitewise:assetHierarchyPath": [
            "/root node asset ID",
            "/root node asset ID/*"
          ]
        }
      }
    }
  ]
}

```

- 當您 AWS IoT SiteWise 使用此動作將資料傳送至時，您的資料必須符合BatchPutAssetPropertyValue作業的需求。如需詳細資訊，請參閱 AWS IoT SiteWise API 參考中的[BatchPutAssetPropertyValue](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

putAssetPropertyValueEntries

資產屬性值項目的清單，其中每個項目都包含下列資訊：

## propertyAlias

(選用) 與資產屬性相關聯的屬性別名。指定 `propertyAlias` 或同時指定 `assetId` 和 `propertyId`。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的[將工業資料串流映射至資產屬性](#)。

支援[替代範本](#)：是

## assetId

(選擇性) AWS IoT SiteWise 資產的 ID。指定 `propertyAlias` 或同時指定 `assetId` 和 `propertyId`。

支援[替代範本](#)：是

## propertyId

(選用) 該資產屬性的 ID。指定 `propertyAlias` 或同時指定 `assetId` 和 `propertyId`。

支援[替代範本](#)：是

## entryId

(選用) 此項目的唯一識別符。定義 `entryId`，以更好地追蹤在發生失敗時哪則訊息造成錯誤。預設為新的 UUID。

支援[替代範本](#)：是

## propertyValues

要插入的屬性值清單，其中每個屬性值包含以下格式的時間戳記、品質和值 (TQV)：

### timestamp

包含下列資訊的時間戳記結構：

### timeInSeconds

包含 Unix epoch 時間中時間 (以秒為單位) 的字串。如果您的消息承載沒有時間戳記，可以使用 `timestamp()`，返回目前時間 (以毫秒為單位)。若要將該時間轉換為以秒為單位，您可以使用下列替代範本：`${floor(timestamp()) / 1E3}`。

支援[替代範本](#)：是

## offsetInNanos

(選用) 包含從該時間 (以秒為單位) 開始的奈秒時間位移的字串。如果您的消息承載沒有時間戳記，可以使用 [timestamp\(\)](#)，返回目前時間 (以毫秒為單位)。若要計算從該時間開始的奈秒位移，可以使用下列替代範本：`${(timestamp() % 1E3) * 1E6}`。

支援[替代範本](#)：是

關於 Unix 紀元時間，僅 AWS IoT SiteWise 接受過去最多 7 分鐘的時間戳記的條目，最多 future 5 分鐘。

## quality

(選用) 說明數值品質的字串。有效值：GOOD、BAD、UNCERTAIN。

支援[替代範本](#)：是

## value

包含下列其中一個值欄位的值結構，視資產屬性的資料類型而定：

### booleanValue

(選用) 包含值項目之布林值的字串。

支援[替代範本](#)：是

### doubleValue

(選用) 包含值項目之雙值的字串。

支援[替代範本](#)：是

### integerValue

(選用) 包含值項目之整數值的字串。

支援[替代範本](#)：是

### stringValue

(選用) 值項目的字串值。

支援[替代範本](#)：是

## roleArn

IAM 角色的 ARN，AWS IoT 可授與向其傳送資產屬性值的權限。AWS IoT SiteWise 如需詳細資訊，請參閱 [要求](#)。

支援替代範本：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的基本 IoT SiteWise 動作。

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

下列 JSON 範例會定義 AWS IoT 規則中的 IoT SiteWise 動作。此範例使用此主題作為屬性別名和 `timestamp()` 函數。例如，如果您將資料發佈至 `/company/windfarm/3/turbine/7/rpm`，此動作會將資料傳送至屬性別名與您指定主題相同的資產屬性。

```
{
  "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"
    }
  }
]
}

```

## 另請參閱

- [《AWS IoT SiteWise 使用者指南》中的什麼是 AWS IoT SiteWise ？](#)
- [使用使用AWS IoT SiteWise 者指南中的 AWS IoT Core 規則擷取資料](#)
- [AWS IoT SiteWise 從AWS IoT SiteWise 使用者指南中的內 AWS IoT 容擷取資料](#)
- [疑難排解AWS IoT SiteWise 使用者指南中的 AWS IoT SiteWise 規則動作](#)

## Firehose

Firehose (firehose) 動作會將 MQTT 訊息中的資料傳送至 Amazon 資料 Firehose 串流。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `firehose:PutRecord` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用 Firehose 將資料傳送到 Amazon S3 儲存貯體，而且您使用受管的 AWS KMS 客戶 AWS KMS key 來加密 Amazon S3 中的靜態資料，Firehose 必須擁有儲存貯體的存取權，並允許代表呼叫者使用。AWS KMS key 如需詳細資訊，請參閱 [Amazon 資料 Firehose 開發人員指南中的授與 Firehose 存取 Amazon S3 目的地](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### batchMode

(選擇性) 是否使 [PutRecordBatch](#) 用批次傳送 Firehose 串流。預設值為 `false`。

若 `batchMode` 為 `true` 且規則的 SQL 陳述式評估為 `Array`，則每個 `Array` 元會在 `PutRecordBatch` 請求中形成一筆記錄。產生的陣列不能含有超過 500 條記錄。

支援 [替代範本](#)：否

### deliveryStreamName

要寫入訊息資料的「Firehose」串流。

支援 [替換模板](#)：API 和 AWS CLI 僅

### separator

(選擇性) 用來分隔寫入 Firehose 串流的記錄的字元分隔符號。若您略過此參數，則串流不會使用分隔符號。有效值：`,` (逗號)、`\t` (索引標籤)、`\n` (換行符號)，`\r\n` (Windows 換行)。

支援 [替代範本](#)：否

### roleArn

允許存取 Firehose 串流的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援 [替代範本](#)：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的「Firehose」動作。

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

下列 JSON 範例會在 AWS IoT 規則中定義具有替代範本的 Firehose 動作。

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

## 另請參閱

- [什麼是 Amazon 數據 Firehose？](#) 在 Amazon 數據 Firehose 開發人員指南



## Kinesis Data Streams

Kinesis Data Streams (kinesis) 動作 會將資料從 MQTT 訊息寫入 Amazon Kinesis Data Streams。

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `kinesis:PutRecord` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用 AWS KMS 客戶管理 AWS KMS key (KMS 金鑰) 來加密 Kinesis Data Streams 中的靜態資料，則該服務必須具有代表呼叫者使用 AWS KMS key 的權限。如需詳細資訊，請參閱《Amazon Kinesis Data Streams 開發人員指南》中 [使用使用者產生之 AWS KMS keys 的許可](#)。

### 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

`stream`

要寫入資料的 Kinesis 資料串流。

支持 [替換模板](#)：API 和 AWS CLI 僅

`partitionKey`

分割區索引鍵用來決定資料要寫入哪個碎片。分割區索引鍵通常是由表達式 (如 `${topic()}` 或 `${timestamp()}`) 組成。

支援 [替代範本](#)：是

`roleArn`

IAM 角色的 ARN，AWS IoT 可授與存取 Kinesis 資料串流的權限。如需詳細資訊，請參閱 [要求](#)。

支援 [替代範本](#)：否

### 範例

下列 JSON 範例會定義 AWS IoT 規則中的 Kinesis Data Streams 動作。

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

下列 JSON 範例會在 AWS IoT 規則中使用替代範本定義 Kinesis 動作。

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

## 另請參閱

- 《Amazon Kinesis Data Streams 開發人員指南》中的 [什麼是 Amazon Kinesis Data Streams ?](#)

# Lambda

Lambda (lambda) 動作會叫用 AWS Lambda 函式，傳入 MQTT 訊息。AWS IoT 以非同步方式叫用 Lambda 函數。

您可依循對您展示如何使用 Lambda 動作 來建立及測試規則的教學課程。如需詳細資訊，請參閱 [教學課程：使用 AWS Lambda 函數來格式化通知](#)。

## 需求

此規則動作具有下列需求：

- 若 AWS IoT 要叫用 Lambda 函數，您必須設定授與 `lambda:InvokeFunction` 權限的政策 AWS IoT。您只能叫用 Lambda 政策所在 AWS 區域 位置所定義的 Lambda 函數。Lambda 函數使用的是資源型政策，因此您必須將政策連接至 Lambda 函數本身。

使用下列 AWS CLI 命令附加授與 `lambda:InvokeFunction` 權限的原則。

```
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` 命令需要下列參數：

`--function-name`

Lambda 函數的名稱。您可以新增許可來更新函數的資源政策。

`--region`

函 AWS 區域 數的。

`--principal`

取得許可的委託人。這應該是 `iot.amazonaws.com` 為了允 AWS IoT 許調用 Lambda 函數的權限。

`--source-arn`

該項規則的 ARN。您可以使用 `get-topic-rule` AWS CLI 指令取得規則的 ARN。

`--source-account`

規則 AWS 帳戶 的定義位置。

--statement-id

專屬的陳述式識別符。

--action

您想要在此陳述式中允許的 Lambda 動作。若 AWS IoT 要允許叫用 Lambda 函數，請指定 `lambda:InvokeFunction`。

#### Important

如果您在未提供 `source-arn` 或的情況下為 AWS IoT 主體新增權限 `source-account`，則使 AWS 帳戶用 Lambda 動作建立規則的任何使用者都可以啟動規則以呼叫 Lambda 函數 AWS IoT。

如需詳細資訊，請參閱 [AWS Lambda 許可](#)。

- 如果您使用受管的 AWS KMS 客戶 AWS KMS key 來加密 Lambda 中的靜態資料，則該服務必須具有代表呼叫者使用的權限。AWS KMS key 如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [靜態加密](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

`functionArn`

要叫用的 Lambda 函數的 ARN。AWS IoT 必須具有調用函數的權限。如需詳細資訊，請參閱 [需求](#)。

如果您未針對您的 Lambda 函數指定版本或別名，則最新版的函數會關閉。如果想要關閉特定版本的 Lambda 函數，您可以指定版本或別名。如要指定版本或別名，請將版本或別名附加至 Lambda 函數的 ARN。

```
arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction:someAlias
```

如需版本控制與別名的詳細資訊，請參閱 [AWS Lambda 函數版本控制與別名](#)。

支持 [替換模板](#)：API 和 AWS CLI 僅

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的 Lambda 動作。

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "lambda": {
          "functionArn": "arn:aws:lambda:us-east-2:123456789012:function:myLambdaFunction"
        }
      }
    ]
  }
}
```

下列 JSON 範例會在 AWS IoT 規則中使用替代範本定義 Lambda 動作。

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "lambda": {
          "functionArn": "arn:aws:lambda:us-east-1:123456789012:function:
${topic()}"
        }
      }
    ]
  }
}
```

## 另請參閱

- [什麼是 AWS Lambda？](#) 在 AWS Lambda 開發人員指南
- [教學課程：使用 AWS Lambda 函數來格式化通知](#)

## 位置

Location (location) (位置) 動作會將您的地理位置資料傳送至 [Amazon Location Service](#)。

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `geo:BatchUpdateDevicePosition` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

### 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

`deviceId`

提供位置資料的裝置唯一 ID。如需詳細資訊，請參閱《Amazon Location Service API 參考》中的 [DeviceId](#)。

支援 [替代範本](#)：是

`latitude`

此字串評估為雙精確度值，該值代表裝置位置的緯度。

支援 [替代範本](#)：是

`longitude`

此字串評估為雙精確度值，該值代表裝置位置的經度。

支援 [替代範本](#)：是

`roleArn`

IAM 角色，允許存取 Amazon Location Service 網域。如需詳細資訊，請參閱 [要求](#)。

`timestamp`

對位置資料進行取樣的時間。預設值是處理 MQTT 訊息的時間。

timestamp 值包含以下兩個值：

- value：傳回一個很長的 Epoch 時間值的表達式。您可使用 [the section called "time\\_to\\_epoch\(String, String\)"](#) 函數，從訊息承載中傳遞的日期或時間建立有效的時間戳記。支援 [替代範本](#)：是。
- unit：(選擇性) value 中所述運算式產生的 timestamp 值的精確度。有效值：SECONDS | MILLISECONDS | MICROSECONDS | NANOSECONDS。預設值為 MILLISECONDS。支援 [替換模板](#)：API 和 AWS CLI 僅支持。

trackerName

在已更新位置的 Amazon Location 中的追蹤器資源名稱。如需詳細資訊，請參閱《Amazon Location Service 開發人員指南》中的 [追蹤器](#)。

支援 [替代範本](#)：API 且 AWS CLI 僅支援

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的「位置」動作。

```
{
  "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": "MILLISECONDS"
          },
          "latitude": "-12.3456",
          "longitude": "65.4321"
        }
      }
    ]
  }
}
```

下列 JSON 範例會在 AWS IoT 規則中定義具有替代範本的「位置」動作。

```
{
  "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": "${timestamp()}",
            "unit": "MILLISECONDS"
          },
          "latitude": "${get(position, 0)}",
          "longitude": "${get(position, 1)}"
        }
      }
    ]
  }
}
```

下列 MQTT 承載範例顯示上述範例中的替代範本如何存取資料。您可以使用 [get-device-position-history](#) CLI 命令來驗證 MQTT 承載資料是否已在您的位置追蹤器中進行傳送。

```
{
  "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"
}
]
```

## 另請參閱

- 在《Amazon Location Service 開發人員指南》中的[什麼是 Amazon Location Service ?](#)。

## OpenSearch

OpenSearch (openSearch) 動作會將 MQTT 訊息中的資料寫入 Amazon OpenSearch 服務網域。然後，您可以使用 OpenSearch 儀表板等工具查詢和視覺化 OpenSearch 服務中的資料。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `es:ESHttpPut` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用受管理的客戶 AWS KMS key 來加密 OpenSearch 服務中的靜態資料，則該服務必須具有代表呼叫者使用 KMS 金鑰的權限。如需詳細資訊，請參閱 Amazon 服務開發人員指南中的 Amazon OpenSearch 服務 [靜態資料加密](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### endpoint

您的 Amazon OpenSearch 服務域的端點。

支援[替代範本](#)：API 且 AWS CLI 僅支援

index

要存儲 OpenSearch 數據的索引。

支援[替代範本](#)：是

type

欲存放文件的類型。

**Note**

對於 1.0 之後的 OpenSearch 版本，type 參數的值必須是 `_doc`。如需詳細資訊，請參閱 [OpenSearch 文件](#)。

支援[替代範本](#)：是

id

各文件的專屬識別符。

支援[替代範本](#)：是

roleARN

允許存取 OpenSearch 服務網域的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援[替代範本](#)：否

## 限制

OpenSearch (openSearch) 動作無法用於將資料傳遞至 VPC 彈性搜尋叢集。

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的 OpenSearch 動作，以及如何指定 OpenSearch 動作的欄位。如需詳細資訊，請參閱 [OpenSearch 動作](#)。

```
{
  "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": "_doc",
      "id": "${newuuid()}",
      "roleArn": "arn:aws:iam::123456789012:role/aws_iot_os"
    }
  }
]
}

```

下列 JSON 範例會在 AWS IoT 規則中定義具有替代範本的 OpenSearch 動作。

```

{
  "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"
        }
      }
    ]
  }
}

```

### Note

取代的 type 欄位適用於 1.0 OpenSearch 版。對於 1.0 之後的任何版本，的值 type 必須是 `_doc`。

## 另請參閱

[什麼是 Amazon OpenSearch 服務？](#) 在 Amazon 服務 OpenSearch 開發人員指南

## Republish

重新發佈 (republish) 動作將 MQTT 訊息重新發佈至另一個 MQTT 主題。

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `iot:Publish` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

### 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

#### headers

MQTT 5.0 版標頭資訊。

如需詳細資訊，請參閱 AWS API 參考 [MqttHeaders](#) 中的 [RepublishAction](#) 和。

#### topic

要將訊息重新發佈至該 MQTT 主題。

如要重新發佈至以 \$ 開頭的預留主題，請改用 \$\$。例如，如要重新發佈至裝置影子主題 `$aws/things/MyThing/shadow/update`，請將主題指定為 `$$aws/things/MyThing/shadow/update`。

#### Note

重新發佈至 [預留任務主題](#) 不受支援。

AWS IoT Device Defender 保留主題不支援 HTTP 發佈。

支援替代範本：是

qos

(選用) 重新發佈訊息時要使用的服務品質 (QoS) 層級。有效值：0、1。預設值為 0。如需 MQTT QoS 的詳細資訊，請參閱 [MQTT](#)。

支援替代範本：否

roleArn

允許發佈 AWS IoT 至 MQTT 主題的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援替代範本：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的重新發佈動作。

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

下列 JSON 範例會在 AWS IoT 規則中定義具有替代範本的重新發佈動作。

```
{
  "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"
        }
      }
    ]
  }
}

```

下面的 JSON 範例定義 AWS IoT 規則中使用 headers 的重新發佈動作。

```

{
  "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"
              }
            ]
          }
        }
      }
    ]
  }
}

```

**Note**

原始來源 IP 不會透過[重新發布動作](#)傳遞。

## S3

S3 (s3) 動作會從 MQTT 訊息將資料寫入 Amazon Simple Storage Service (Amazon S3) 儲存貯體。

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `s3:PutObject` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用 AWS KMS 受管的客戶 AWS KMS key 來加密 Amazon S3 中的靜態資料，則該服務必須具有代表呼叫者使用的權限。AWS KMS key 如需詳細資訊，請參閱 [AWS Amazon 簡單儲存服務開發人員指南 AWS KMS keys 中的受管 AWS KMS keys 和客戶管理](#)。

### 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

`bucket`

要寫入資料的 Amazon S3 儲存貯體。

支援[替代範本](#)：API 且 AWS CLI 僅支援

`cannedacl`

(選用) Amazon S3 固定的 ACL，負責控制物件金鑰辨識出之物件的存取權限。如需詳細資訊 (包含允許的值)，請參閱[固定 ACL](#)。

支援[替代範本](#)：否

`key`

寫入資料的檔案路徑。

考慮一個範例，其中此參數為 `${topic()}/${timestamp()}`，且規則會收到主題為 `some/topic` 訊息。若目前的時間戳記為 `1460685389`，則此動作會將資料寫入在 S3 儲存貯體 `some/topic` 資料夾中名為 `1460685389` 的檔案。

#### Note

如果您使用靜態索引鍵，則會在每次規則呼叫時 AWS IoT 覆寫單一檔案。我們建議您使用訊息時間戳記或另一個唯一的訊息識別符，以便每一個收到的訊息都會將新檔案儲存於 Amazon S3 中。

支援替代範本：是

roleArn

允許存取 Amazon S3 儲存貯體的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援替代範本：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的 S3 動作。

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



## 另請參閱

- 《Amazon Simple Storage Service 使用者指南》中的[什麼是 Amazon S3 ?](#)

## Salesforce IoT

Salesforce IoT (salesforce) 動作將資料從觸發規則的 MQTT 訊息傳送至 Salesforce IoT 輸入串流。

### 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

#### url

該 URL 會透過 Salesforce IoT 輸入串流公開。在您建立輸入串流時，便可從 Salesforce IoT 平台使用 URL。若需詳細資訊，請參閱 Salesforce IoT 文件。

支援[替代範本](#)：否

#### token

用於驗證特定 Salesforce IoT 輸入串流存取的權杖。在您建立輸入串流時，您便可在 Salesforce IoT 平台使用權杖。若需詳細資訊，請參閱 Salesforce IoT 文件。

支援[替代範本](#)：否

## 範例

下面的 JSON 範例定義 AWS IoT 規則中的 Salesforce IoT 動作。

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "salesforce": {
          "token": "ABCDEFGHII123456789abcdefghi123456789",
          "url": "https://ingestion-cluster-id.my-env.sfdcnw.com/streams/stream-id/connection-id/my-event"
        }
      }
    ]
  }
}
```

```
}  
  }  
] }  
} }
```

## SNS

使用 SNS (sns) 動作，將來自 MQTT 訊息的資料作為 Amazon Simple Notification Service (Amazon SNS) 推送通知進行傳送。

您可依循對您展示如何使用 SNS 動作 來建立及測試規則的教學課程。如需詳細資訊，請參閱 [教學課程：傳送 Amazon SNS 通知](#)。

### Note

SNS 動作不支援 [Amazon SNS FIFO \(先進先出\) 主題](#)。因為規則引擎是一項完全分配式的服務，因此在呼叫 SNS 動作時無法保證訊息順序。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行sns:Publish作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用 AWS KMS 客戶管理的受管 AWS KMS key 來加密 Amazon SNS 中的靜態資料，則該服務必須具有代表呼叫者使用 AWS KMS key 的權限。如需詳細資訊，請參閱《Amazon Simple Notification Service 開發人員指南》中的 [金鑰管理](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

targetArn

推送通知傳送到的 SNS 主題或個別裝置。

支援[替代範本](#)：API 且 AWS CLI 僅支援

messageFormat

(選用) 訊息格式。Amazon SNS 會使用此設定來判斷是否應剖析承載，以及是否應擷取承載與特定平台相關的部分。有效值：JSON、RAW。預設為 RAW。

支援[替代範本](#)：否

roleArn

允許存取 SNS 的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援[替代範本](#)：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的 SNS 動作。

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

下列 JSON 範例會在 AWS IoT 規則中定義具有替代範本的 SNS 動作。

```
{
  "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"
  }
}
```

## 另請參閱

- 《Amazon Simple Notification Service 開發人員指南》中的[什麼是 Amazon Simple Notification Service ?](#)
- [教學課程：傳送 Amazon SNS 通知](#)

## SQS

使用 SQS (sqs) 動作，將資料從 MQTT 訊息傳送至 Amazon Simple Queue Service (Amazon SQS) 佇列。

### Note

SQS 動作不支援 [Amazon SQS FIFO \(先進先出\) 佇列](#)。因為規則引擎是一項完全分配式的服務，因此在觸發 SQS 動作時無法保證訊息順序。

## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `sqs:SendMessage` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

- 如果您使用受管的 AWS KMS 客戶 AWS KMS key 來加密 Amazon SQS 中的靜態資料，則該服務必須具有代表呼叫者使用的權限。AWS KMS key 如需詳細資訊，請參閱《Amazon Simple Queue Service 開發人員指南》中的[金鑰管理](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### queueUrl

要寫入資料的 Amazon SQS 佇列 URL。此 URL 中的區域不需要與您的 [AWS IoT 規則 AWS 區域](#) 相同。

#### Note

使用 SQS 規則作業，交叉資料傳輸可能會產生額外費 AWS 區域 用。如需詳細資訊，請參閱 [Amazon SQS 定價](#)。

支援 [替代範本](#)：API 且 AWS CLI 僅支援

### useBase64

設定此參數為 true 以配置規則動作，在資料寫入 Amazon SQS 佇列之前，對訊息資料進行 base64 編碼。預設為 false。

支援 [替代範本](#)：否

### roleArn

允許存取 Amazon SQS 佇列的 IAM 角色。如需詳細資訊，請參閱 [要求](#)。

支援 [替代範本](#)：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的 SQS 動作。

```
{
  "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"
        }
    ]
}

```

下列 JSON 範例會在 AWS IoT 規則中定義具有替代範本的 SQS 動作。

```

{
  "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"
        }
      }
    ]
  }
}

```

## 另請參閱

- 《Amazon Simple Queue Service 開發人員指南》中的 [什麼是 Amazon Simple Queue Service ?](#)

## Step Functions

Step Functions ( stepFunctions ) 動作啟動 AWS Step Functions 狀態機。

### 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `states:StartExecution` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇或建立允許 AWS IoT 執行此規則動作的角色。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### `stateMachineName`

要啟動的 Step Functions 狀態機器名稱。

支援 [替代範本](#)：API 且 AWS CLI 僅支援

### `executionNamePrefix`

(選用) 狀態機器執行的名稱，包含此字首後接 UUID。若未提供名稱，Step Functions 會為每個狀態機器建立一個唯一的名稱。

支援 [替代範本](#)：是

### `roleArn`

授與啟動狀態機器 AWS IoT 權限之角色的 ARN。如需詳細資訊，請參閱 [要求](#)。

支援 [替代範本](#)：否

## 範例

下列 JSON 範例會定義 AWS IoT 規則中的「Step Functions 式」動作。

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

```
        }  
    ]  
}  
}
```

## 另請參閱

- [什麼是 AWS Step Functions ?](#) 在 AWS Step Functions 開發人員指南

## Timestream

Timestream 規則動作將屬性 (測量) 從 MQTT 訊息寫入 Amazon Timestream 表格中。如需 Amazon Timestream 的詳細資訊，請參閱[什麼是 Amazon Timestream ?](#)。

### Note

Amazon Timestream 並不是在所有 AWS 區域 s 可用。若 Amazon Timestream 無法在您的區域中使用，則不會顯示於規則動作清單中。

此規則儲存於 Timestream 資料庫中的屬性是規則查詢陳述式所產生的屬性。剖析查詢陳述式結果中每個屬性值，以推斷其資料類型 (如 [the section called “DynamoDBv2”](#) 動作中所示)。每個屬性值都會寫入其在 Timestream 表格中的記錄。如要指定或變更屬性的資料類型，請使用查詢陳述式中的 [cast\(\)](#) 函數。如需每個 Timestream 記錄內容的詳細資訊，請參閱 [the section called “Timestream 記錄內容”](#)。

### Note

在 SQL V2 (2016-03-23) 中，作為完整數的數值，例如 10.0，將轉換為其整數表示形式 (10)。將其明確轉換為 Decimal 值，例如使用 [cast\(\)](#) 函數不能阻止此種行為，結果仍是一個 Integer 值。這可能會造成類型不相符的錯誤，導致資料無法記錄於 Timestream 資料庫中。若要將整數數值當成 Decimal 值處理，請針對規則查詢陳述式使用 SQL V1 (2015-10-08)。

### Note

可以寫入 Amazon Timestream 表格的最大 Timestream 規則動作值為 100。如需詳細資訊，請參閱 [Amazon Timestream 配額參考](#)。



## 要求

此規則動作具有下列需求：

- AWS IoT 可假設執行 `timestream:DescribeEndpoints` 和 `timestream:WriteRecords` 作業的 IAM 角色。如需詳細資訊，請參閱 [授與 AWS IoT 規則所需的存取權](#)。

在 AWS IoT 主控台中，您可以選擇、更新或建立角色，以 AWS IoT 允許執行此規則動作。

- 如果您使用客戶 AWS KMS 來加密 Timestream 中的靜態資料，則該服務必須具有代表呼叫者使用的權限。AWS KMS key 如需詳細資訊，請參閱 [AWS 服務如何使用 AWS KMS](#)。

## 參數

使用此動作建立 AWS IoT 規則時，必須指定下列資訊：

### **databaseName**

Amazon Timestream 資料庫的名稱，其中包含用於接收此動作所建立之記錄的資料表。另請參閱 **tableName**。

支援 [替代範本](#)：API 且 AWS CLI 僅支援

### **dimensions**

寫入每個量測記錄的時間序列的中繼資料屬性。例如，EC2 執行個體的名稱和可用區域，或風力發電機製造商的名稱即為維度。

#### **name**

中繼資料維度名稱。這是資料庫資料表記錄中的欄的名稱。

無法命名的維度：`measure_name`、`measure_value`，或 `time`。這些為預留名稱。維度名稱不可以 `ts_` 或 `measure_value` 開頭，且其不可包含冒號 (:) 字元。

支援 [替代範本](#)：否

#### **value**

要寫入資料庫記錄的該資料欄的值。

支援 [替代範本](#)：是

## roleArn

授予 AWS IoT 許可之角色的 Amazon 資源名稱 (ARN)，可寫入 Timestream 資料庫表格。如需詳細資訊，請參閱 [要求](#)。

支援[替代範本](#)：否

## tableName

要寫入測量記錄的資料庫表格名稱。另請參閱 `databaseName`。

支援[替代範本](#)：API 且 AWS CLI 僅支援

## timestamp

要用於項目的 timestamp 的值。若為空白，則會使用處理項目的時間。

unit

value 中所述運算式產生的 timestamp 值的精確度。

有效值：SECONDS | MILLISECONDS | MICROSECONDS | NANoseconds。預設值為 MILLISECONDS。

value

傳回一個很長的 Epoch 時間值的表達式。

您可使用 [the section called “time\\_to\\_epoch\(String, String\)”](#) 函數，從訊息承載中傳遞的日期或時間建立有效的時間戳記。

## Timestream 記錄內容

此動作寫入 Amazon Timestream 表格的資料包括時間戳記、來自 Timestream 規則動作的中繼資料，及規則查詢陳述式的結果。

對於查詢陳述式結果中的每個屬性 (測量)，此規則動作會將記錄寫入具有這些欄的指定 Timestream 表格。

欄名稱	屬性類型	Value	說明
<i>dimension-name</i>	DIMENSION	指定於 Timestream 規則動作項目中的值。	指定於規則動作項目中的每個維度都會於

欄名稱	屬性類型	Value	說明
			Timestream 資料庫中建立一個具維度名稱的欄。
measure_name	MEASURE_NAME	屬性名稱	查詢陳述式結果中的屬性名稱，其值指定於 <code>measure_value::data-type</code> 欄中。
measure_value:: <i>data-type</i>	MEASURE_VALUE	查詢陳述式結果中的屬性值。屬性名稱位於 <code>measure_name</code> 欄中。	此值會進行解譯 * 並轉換為最適合下列的比對項目： <code>bigint</code> 、 <code>boolean</code> 、 <code>double</code> 或 <code>varchar</code> 。Amazon Timestream 為每種資料類型建立一個個別的欄。訊息中的值可使用規則查詢陳述式中的 <a href="#">cast()</a> 函數轉換為另一個資料。
time	TIMESTAMP	資料庫中記錄的日期和時間。	此值是由規則引擎或 <code>timestamp</code> 屬性來指派 (若已定義)。

\* 從訊息承載讀取的屬性值解譯如下。請參閱 [the section called “範例”](#)，以取得每個案例的說明。

- 將 `true` 或 `false` 的未加引號值解釋為 `boolean` 類型。
- 將十進位數字解釋為 `double` 類型。
- 將未帶小數點的十進位數值解釋為 `bigint` 類型。
- 將帶引號的字串解釋為 `varchar` 類型。
- 將物件和陣列數值轉換為 JSON 字串並儲存為 `varchar` 類型。

## 範例

下列 JSON 範例會使用規則中的替代範本定義時間串流 AWS IoT 規則動作。

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

上一個範例中定義的 Timestream 主題規則動作與下列訊息承載搭配使用會導致 Amazon Timestream 記錄寫入後續表格中。

```
{
  "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"]
}
}

```

下表會顯示使用指定主題規則動作處理先前訊息承載建立的資料庫欄和記錄。device\_firmware\_sku 和 device\_id 欄為定義於主題規則動作中的 DIMENSIONS (維度)。Timestream 主題規則動作會建立 time 欄及 measure\_name 和 measure\_value::\* 欄，其中其會填入主題規則動作查詢陳述式結果的值。

device_firmware_sku	device_id	measure_name	measure_value::bigint	measure_value::varchar	measure_value::double	measure_value::boolean	time
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	complex_value	-	{"simple_element": 42,"array_of_integers":[23,36,56,72], "字串陣列":["紅","綠","藍"]}	-	-	2020-08-26 22:42:16.423000000
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	integer_value_as_string	-	123456789012	-	-	2020-08-26 22:42:16.423000000
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	boolean_value	-	-	-	TRUE	2020-08-26 22:42:16.423000000

device_firmware_sku	device_id	measure_name	measure_value::bigint	measure_value::varchar	measure_value::double	measure_value::boolean	time
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	integer_value	123456789012	-	-	-	2020-08-26 22:42:16.423000000
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	string_value	-	字串值	-	-	2020-08-26 22:42:16.423000000
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	array_of_integers	-	[23,36,56,72]	-	-	2020-08-26 22:42:16.423000000
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	字串陣列	-	["紅","綠","藍"]	-	-	2020-08-26 22:42:16.423000000
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	boolean_value_as_string	-	TRUE	-	-	2020-08-26 22:42:16.423000000
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	double_value	-	-	123.456789012	-	2020-08-26 22:42:16.423000000
我的靜態中繼資料	iotconsole-159EXAMPLE738-0	double_value_as_string	-	123.45679	-	-	2020-08-26 22:42:16.423000000

## 規則疑難排解

如果您的規則有問題，我們建議您啟用 CloudWatch 記錄檔。您可分析您的記錄，即可判斷該問題是否在於授權，或者是否為 WHERE 子句條件不相符之類的問題。如需詳細資訊，請參閱[設定 CloudWatch 記錄檔](#)。

## 使用規則存取跨帳戶資 AWS IoT 源

您可以設定跨帳戶存取的 AWS IoT 規則，以便將一個帳戶 MQTT 主題上擷取的資料路由到另一個帳戶的 AWS 服務 (例如 Amazon SQS 和 Lambda)。以下說明如何設定跨帳戶資料擷取的 AWS IoT 規則，從一個帳戶中的 MQTT 主題到另一個帳戶中的目的地。

跨帳戶規則可使用[資源型許可](#)，在目的地資源上進行配至。因此，只有 AWS IoT 支援以資源為基礎的權限的目的地才能透過規則啟用跨帳戶存取。支援的目的地包括 Amazon SQS、Amazon SNS、Amazon S3 和 AWS Lambda。

### Note

對於支援的目的地，Amazon SQS 除外，您必須將規則定義 AWS 區域 為與其他服務的資源相同，以便規則動作可以與該資源互動。如需有關 AWS IoT 規則動作的詳細資訊，請參閱[AWS IoT 規則動作](#)。如需規則之 SQS 動作的詳細資訊，請參閱[???](#)。

## 必要條件

- 熟悉 [AWS IoT 規則](#)
- 了解 IAM [使用者](#)、[角色](#) 和 [資源型許可](#)
- 安裝了 [AWS CLI](#)

## Amazon SQS 的跨帳戶設定

案例：帳戶 A 將資料從 MQTT 訊息傳送至帳戶 B 的 Amazon SQS 佇列。

AWS 帳戶	帳戶稱為	描述
<b>1111-1111-1111</b>	帳戶 A	規則動作：sqs:SendMessage
<b>2222-2222-2222</b>	帳戶 B	Amazon SQS 佇列 <ul style="list-style-type: none"> <li>• ARN : <b>arn:aws:sqs:region:2222-222-2222:ExampleQueue</b></li> </ul>

AWS 帳戶	帳戶稱為	描述
		<ul style="list-style-type: none"> <li>URL : <i>https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue</i></li> </ul>

### Note

您的目的地 Amazon SQS 佇列不一定要與您的[AWS IoT 規則 AWS 區域](#)相同。如需規則之 SQS 動作的詳細資訊，請參閱[???](#)。

## 執行帳戶 A 任務

### 注意

如要執行下列命令，您的 IAM 使用者應具有以規則的 Amazon 資源名稱 (ARN) 作為資源的 `iot:CreateTopicRule` 許可，及以資源作為角色之 ARN 的 `iam:PassRole` 動作許可。

1. 使用帳戶 A 的 IAM 使用者[配置 AWS CLI](#)。
2. 建立信任 AWS IoT 規則引擎的 IAM 角色，並附加允許存取帳戶 B 之 Amazon SQS 佇列的政策。請參閱[授與所需存取權中 AWS IoT 的範例命令和政策文件](#)。
3. 若要建立附加至主題的規則，請執行[create-topic-rule 命令](#)。

```
aws iot create-topic-rule --rule-name myRule --topic-rule-payload file:///./my-rule.json
```

下列為範例承載檔案，具有一個會將傳送至 `iot/test` 主題之所有訊息插入指定 Amazon SQS 佇列的規則。SQL 陳述式會篩選訊息，而角色 ARN 授予 AWS IoT 新增 Amazon SQS 佇列之訊息的許可。

```
{
  "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
  }
}
]
```

如需如何在 AWS IoT 規則中定義 Amazon SQS 動作的詳細資訊，請參閱[AWS IoT 規則動作-Amazon SQS](#)。

### 執行帳戶 B 任務

1. 使用帳戶 B 的 IAM 使用者[配置 AWS CLI](#)。
2. 如要將 Amazon SQS 佇列資源的許可授予帳戶 A，請執行 [add-permission 命令](#)。

```
aws sqs add-permission --queue-url https://sqs.region.amazonaws.com/2222-2222-2222/ExampleQueue --label SendMessageToMyQueue --aws-account-ids 1111-1111-1111 --actions SendMessage
```

## Amazon SNS 的跨帳戶設定

案例：帳戶 A 將資料從 MQTT 訊息傳送至帳戶 B 的 Amazon SNS 主題。

AWS 帳戶	帳戶稱為	描述
<i>1111-1111-1111</i>	帳戶 A	規則動作：sns:Publish
<i>2222-2222-2222</i>	帳戶 B	Amazon SNS 主題 ARN： <i>arn:aws:sns:region:2222-2222-2222:ExampleTopic</i>

## 執行帳戶 A 任務

### 備註

如要執行下列命令，您的 IAM 使用者應具有以規則 ARN 作為資源的 `iot:CreateTopicRule` 許可，以及資源作為角色 ARN 的 `iam:PassRole` 動作許可。

1. 使用帳戶 A 的 IAM 使用者 [配置 AWS CLI](#)。
2. 建立信任 AWS IoT 規則引擎的 IAM 角色，並附加允許存取帳戶 B Amazon SNS 主題的政策。如需命令和原則文件 [AWS IoT 的範例](#)，請參閱 [授與必要存取權](#)。
3. 若要建立附加至主題的規則，請執行 [create-topic-rule 命令](#)。

```
aws iot create-topic-rule --rule-name myRule --topic-rule-payload file://./my-rule.json
```

下列為範例承載檔案，具有一個會將傳送至 `iot/test` 主題之所有訊息插入指定 Amazon SNS 主題的規則。SQL 陳述式會篩選訊息，而角色 ARN 會授與將訊息傳送至 Amazon SNS 主題的 AWS IoT 權限。

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

如需如何在 AWS IoT 規則中定義 Amazon SNS 動作的詳細資訊，請參閱 [AWS IoT 規則動作-Amazon SNS](#)。

## 執行帳戶 B 任務

1. 使用帳戶 B 的 IAM 使用者[配置 AWS CLI](#)。
2. 如要將 Amazon SNS 主題資源的許可授予帳戶 A，請執行 [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 的跨帳戶設定

案例：帳戶 A 將資料從 MQTT 訊息傳送至帳戶 B 的 Amazon S3 儲存貯體。

AWS 帳戶	帳戶稱為	描述
<i>1111-1111-1111</i>	帳戶 A	規則動作：s3:PutObject
<i>2222-2222-2222</i>	帳戶 B	Amazon S3 儲存貯體 ARN： <i>arn:aws:s3:::ExampleBucket</i>

## 執行帳戶 A 任務

### 注意

如要執行下列命令，您的 IAM 使用者應具有以規則 ARN 作為資源的 `iot:CreateTopicRule` 許可，以及資源作為角色 ARN 的 `iam:PassRole` 動作許可。

1. 使用帳戶 A 的 IAM 使用者[配置 AWS CLI](#)。
2. 建立可信任 AWS IoT 規則引擎的 IAM 角色，並附加允許存取帳戶 B Amazon S3 儲存貯體的政策。如需命令和原則文件[AWS IoT 的範例](#)，請參閱[授與必要存取權](#)。
3. 若要建立附加至目標 S3 儲存貯體的規則，請執行[create-topic-rule 命令](#)。

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file:///./my-rule.json
```

下列為範例承載檔案，具有一個會將傳送至 `iot/test` 主題之所有訊息插入指定 Amazon S3 儲存貯體的規則。SQL 陳述式會篩選訊息，而角色 ARN 授予 AWS IoT 將訊息新增至 Amazon S3 儲存貯體的許可。

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

如需如何在 AWS IoT 規則中定義 Amazon S3 動作的詳細資訊，請參閱[AWS IoT 規則動作-Amazon S3](#)。

## 執行帳戶 B 任務

1. 使用帳戶 B 的 IAM 使用者[配置 AWS CLI](#)。
2. 建立一個信任帳戶 A 之委託人的儲存貯體政策。

下列為範例承載檔案，其會定義信任另一個帳戶之委託人的儲存貯體政策。

```
{
  "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/*"
  }
]
}

```

如需詳細資訊，請參閱[儲存貯體政策範例](#)。

- 若要將值區政策附加至指定的值區，請執行[put-bucket-policy 命令](#)。

```
aws s3api put-bucket-policy --bucket ExampleBucket --policy file:///./my-bucket-policy.json
```

- 如要讓跨帳戶存取運作，請確定您具有正確的封鎖所有公有存取設定。如需詳細資訊，請參閱[Amazon S3 安全最佳實務](#)。

## 跨帳戶設定 AWS Lambda

案例：科目 A 呼叫科目 B 的 AWS Lambda 函數，並傳入 MQTT 訊息。

AWS 帳戶	帳戶稱為	描述
<i>1111-1111-1111</i>	帳戶 A	規則動作：lambda:InvokeFunction
<i>2222-2222-2222</i>	帳戶 B	Lambda 函數 ARN： <i>arn:aws:lambda:region:2222-2222-2222:function:example-function</i>

### 執行帳戶 A 任務

#### 備註

如要執行下列命令，您的 IAM 使用者應具有以規則 ARN 作為資源的 `iot:CreateTopicRule` 許可，以及資源作為角色 ARN 的 `iam:PassRole` 動作許可。

- 使用帳戶 A 的 IAM 使用者[配置 AWS CLI](#)。

2. 執行命 `create-topic-rule` 令以建立定義對帳戶 B Lambda 函數的跨帳戶存取權限的規則。

```
aws iot create-topic-rule --rule-name my-rule --topic-rule-payload file://./my-rule.json
```

下列為範例承載檔案，具有一個會將傳送至 `iot/test` 主題之所有訊息插入指定 Lambda 函數的規則。SQL 陳述式會篩選訊息，而角色 ARN 會 AWS IoT 授與將資料傳遞至 Lambda 函數的權限。

```
{
  "sql": "SELECT * FROM 'iot/test'",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "lambda": {
        "functionArn": "arn:aws:lambda:region:2222-2222-2222:function:example-function"
      }
    }
  ]
}
```

如需如何在 AWS IoT 規則中定義 AWS Lambda 動作的詳細資訊，請參閱 [AWS IoT 規則動作-Lambda](#)。

## 執行帳戶 B 任務

1. 使用帳戶 B 的 IAM 使用者 [配置 AWS CLI](#)。
2. 執行 [Lambda 的新增權限命令](#)，授予 AWS IoT 規則啟動 Lambda 函數的權限。如要執行下列命令，您的 IAM 使用者應具有 `lambda:AddPermission` 動作的權限。

```
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"
```

選項：

`--principal`

此欄位授予 AWS IoT (以表示 `iot.amazonaws.com`) 呼叫 Lambda 函數的權限。

`--source-arn`

此欄位確認僅 AWS IoT 中的 `arn:aws:iot:region:1111-1111-1111:rule/example-rule` 會觸發此 Lambda 函數，且相同或不同帳戶中的任何其他規則皆無法啟動此 Lambda 函數。

`--source-account`

此欄位確認僅代表 1111-1111-1111 帳戶 AWS IoT 啟用此 Lambda 函數。

### 備註

若您看到來自 AWS Lambda 函數主控台 Configuration (組態) 之下的錯誤訊息「找不到規則」，請略過錯誤訊息並繼續測試連線。

## 錯誤處理 (錯誤動作)

當 AWS IoT 收到來自設備的消息時，規則引擎會檢查消息是否與規則匹配。若是如此，則會對規則的查詢陳述式進行評估，並啟動規則的動作，傳送查詢陳述式的結果。

如果在啟動動作時發生問題，則若針對規則指定了錯誤動作，規則引擎便會啟動該動作。這可能在以下情況時發生：

- 規則並未擁有存取 Amazon S3 儲存貯體的許可。
- 使用者錯誤會造成 DynamoDB 佈建的傳輸量超量。

### Note

本主題所述的錯誤處理適用於[規則動作](#)。若要偵錯 SQL 問題 (包括外部函數)，您可以設定 AWS IoT 記錄。如需詳細資訊，請參閱[???](#)。

## 錯誤動作訊息格式

每一項規則和訊息會產生單條訊息。例如，如果相同規則中的兩個規則動作失敗，則錯誤動作會收到包含那兩個錯誤的訊息。

錯誤動作訊息看似下列範例。

```
{
  "ruleName": "TestAction",
  "topic": "testme/action",
  "cloudwatchTraceId": "7e146a2c-95b5-6caf-98b9-50e3969734c7",
  "clientId": "iotconsole-1511213971966-0",
  "base64OriginalPayload":
  "ewogICJtZXNzYWdlIjogIkhkbGxvIHZyb20gQVdTIElvVCBjb25zb2xlIgp9",
  "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:
yMah1cwPhqTH267QLPhTKeVPKJB8B05ndBHz0mWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y=).
Message arrived on: error/action, Action: s3, Bucket: us-
east-1-s3-verify-user, Key: \"aaa\". Value of x-amz-id-2:
yMah1cwPhqTH267QLPhTKeVPKJB8B05ndBHz0mWtxLTM6uAvwYYuqieAKyb6qRPTxP1tHXCoR4Y="
    }
  ]
}
```

### ruleName

觸發錯誤動作的規則名稱。

### 主題

收到原始訊息的主題。

### 雲觀察 TraceId

參照錯誤記錄的唯一身份 CloudWatch。

### clientId

訊息發佈者的用戶端 ID。



## Base64 OriginalPayload

原始訊息承載 Base64 編碼。

### failures

#### failedAction

無法完成的動作名稱 (例如, 「S3Action」)。

#### failedResource

資源名稱 (例如, S3 儲存貯體名稱)。

#### errorMessage

對該項錯誤的敘述和說明。

## 錯誤動作範例

以下範例為有加入錯誤動作的規則。下列規則中有個動作會將訊息資料寫入 DynamoDB 表格, 還有一個錯誤動作, 會將資料寫入 Amazon S3 儲存貯體:

```
{
  "sql": "SELECT * FROM ..."
  "actions": [{
    "dynamoDB": {
      "table": "PoorlyConfiguredTable",
      "hashKeyField": "AConstantString",
      "hashKeyValue": "AHashKey"}}
  ],
  "errorAction": {
    "s3": {
      "roleArn": "arn:aws:iam::123456789012:role/aws_iam_s3",
      "bucketName": "message-processing-errors",
      "key": "${replace(topic(), '/', '-') + '-' + timestamp() + '-' +
newuuid()}"
    }
  }
}
```

您可以在錯誤動作的 SQL 陳述式中使用任何[函數](#)或[替代範本](#), 包括外部函

數: [aws\\_lambda\(\)](#)、[get\\_dynamodb\(\)](#)、[get\\_thing\\_shadow\(\)](#)、[get\\_secret\(\)](#)、[machinelearning\\_p](#)和[decode\(\)](#)。如果錯誤作業需要呼叫外部函數, 則呼叫錯誤作業可能會產生外部函數的額外帳單。

下列外部函數的計費方式相當於規則動作的費用：[aws\\_lambda\\_get\\_dynamodb\(\)](#)、[get\\_thing\\_shadow\(\)](#)。只有當您將 Protobuf 訊息解碼為 JSON 時，您才會收取該 [decode\(\)](#) 函數的費用。如需詳細資訊，請參閱定 [AWS IoT Core 價頁面](#)。

如需規則及如何指定錯誤動作的相關資訊，請參閱 [建立規 AWS IoT 則](#)。

如需使用 CloudWatch 來監視規則成功或失敗的詳細資訊，請參閱 [AWS IoT 量度和維度](#)。

## 使用基本擷取減少簡訊費用

您可以使用 Basic Ingest，將裝置資料安全地傳送至 AWS 服務支援的 [AWS IoT 規則動作](#)，而不會產生 [訊息](#) 費用。基本擷取會從擷取路徑移除發佈/訂閱訊息代理程式，讓資料流程最佳化。

基本擷取可從您的裝置或應用程式傳送訊息。訊息的前三個層級具有以 `$aws/rules/rule_name` 開頭的主題名稱，其中 *rule\_name* 是您要叫用的 AWS IoT 規則的名稱。

您可將基本擷取字首 (`$aws/rules/rule_name`) 新增至您用來叫用規則的訊息主題，即可搭配基本擷取使用現有的規則。例如，如果您有一個名為 BuildingManager 規則會透過 Buildings/Building5/Floor2/Room201/Lights ("sql": "SELECT \* FROM 'Buildings/#'") 之類的訊息主題叫用，則您可以透過傳送主題為 `$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights` 的訊息，使用基本擷取叫用相同的規則。

請注意：

- 您的裝置和規則無法訂閱基本擷取預留主題。如需詳細資訊，請參閱 [預留主題](#)。
- 如果需要發佈/訂閱 Broker 才能將訊息分發給多位訂閱者 (例如，將訊息傳遞給其他裝置和規則引擎)，您應該繼續使用 AWS IoT 訊息代理程式來處理訊息發佈。不過，請確認使用基本擷取主題以外的主題來發佈您的訊息。

## 使用基本擷取

在使用基本擷取之前，請驗證您的裝置或應用程式是否正在使用對 `$aws/rules/*` 具有發佈許可的 [政策](#)。或者，您可以 `$aws/rules/rule_name/*` 在策略中為單個規則指定權限。除此之外，您的裝置和應用程式可以繼續使用他們與 AWS IoT Core 的現有連線。

當訊息抵達規則引擎時，從基本擷取叫用的規則和透過訊息代理程式訂閱叫用的規則，在實作或錯誤處理上就沒有差別。

您可以建立用於基本擷取的規則。請謹記以下幾點：

- 基本擷取主題的初始字首 (`$aws/rules/rule_name`) 無法用於 `topic(Decimal)` 函數。
- 如果您定義了只能由基本擷取叫用的規則，`rule` 定義中 `sql` 欄位的 `FROM` 子句則為選用。如果規則也需要由必須透過訊息代理程式傳送的其他訊息呼叫 (例如，因為那些其他訊息必須分配到多個訂閱者)，則仍然會需要此項。如需詳細資訊，請參閱 [AWS IoT SQL 參考資料](#)。
- 首三個層級的基本擷取主題 (`$aws/rules/rule_name`) 不會算在主題的 8 區段長度或 256 個總字元限制中。除此之外，適用其他相同限制，如 [AWS IoT 限制](#) 所述。
- 如果收到的訊息包含「基本導入」主題，該主題指定了非作用中規則或不存在的規則，則會在 Amazon 日誌中建立錯誤 CloudWatch 日誌以協助您進行除錯。如需詳細資訊，請參閱 [Rules engine 日誌項目](#)。系統會指出 `RuleNotFound` 指標，讓您可以為此指標建立警示。如需詳細資訊，請參閱 [規則指標](#) 中的規則指標。
- 您仍可使用 QoS 1 發佈基本擷取主題。訊息成功交付至規則引擎後，您就會收到 `PUBACK`。收到 `PUBACK` 不表示您的規則已經成功完成動作。您可以設定錯誤動作，以在動作執行時處理錯誤。如需更多詳細資訊，請參閱 [錯誤處理 \(錯誤動作\)](#)。

## AWS IoT SQL 參考資料

在中 AWS IoT，規則是使用類似 SQL 的語法來定義。SQL 陳述式是由三種類型的子句所組成：

### SELECT

(必要) 從傳入的訊息承載擷取資訊並對資訊執行轉換。要使用的訊息是由 `FROM` 子句中指定的 [主題篩選條件](#) 所識別。

`SELECT` 子句支援 [資料類型](#)、[運算子](#)、[函數](#)、[文字](#)、[案例陳述式](#)、[JSON Extensions](#)、[替代範本巢狀物件查詢](#) 和 [二進位承載](#)。

### FROM

MQTT 訊息 [主題篩選條件](#)，可識別要從中擷取資料的訊息。系統會為每則傳送到符合此處指定的主題篩選條件之 MQTT 主題的訊息啟動該規則。對於透過訊息啟動的規則是必要的，而這些訊息是透過訊息代理程式傳遞的。若為僅使用 [基本擷取](#) 功能啟動的規則，則為選用。

### WHERE

(選用) 新增條件邏輯，其會判斷是否執行某規則指定的動作。

`WHERE` 子句支援 [資料類型](#)、[運算子](#)、[函數](#)、[文字](#)、[案例陳述式](#)、[JSON Extensions](#)、[替代範本](#) 和 [巢狀物件查詢](#)。

SQL 陳述式範例如下所示：

```
SELECT color AS rgb FROM 'topic/subtopic' WHERE temperature > 50
```

MQTT 訊息 (也稱作傳入承載) 的範例如下所示：

```
{
  "color": "red",
  "temperature": 100
}
```

如果此訊息是發佈在 'topic/subtopic' 主題上，便會觸發規則，SQL 陳述式會受到評估。如果 color 屬性大於 50，SQL 陳述式會擷取 "temperature" 屬性的值。WHERE 子句會指定條件 temperature > 50。AS 關鍵字將 "color" 屬性重新命名為 "rgb"。結果 (也稱作傳出承載) 會如下所示：

```
{
  "rgb": "red"
}
```

該資料接下來會轉送給該規則的動作，並發送資料以進行更多處理作業。如需規則動作的詳細資訊，請參閱 [AWS IoT 規則動作](#)。

#### Note

AWS IoT SQL 語法目前不支援註解。

包含空格的屬性名稱不能用作 SQL 陳述式中的欄位名稱。雖然傳入的承載可以包含含有空格的屬性名稱，但這些名稱不能在 SQL 陳述式中使用。但是，如果您使用萬用字元 (\*) 欄位名稱規範，這些名稱會被傳遞到傳出承載。

## SELECT 子句

SELECT AWS IoT Core 子句基本上與 ANSI SQL SELECT 子句相同，但有一些細微的差異。

SELECT 子句支援 [資料類型](#)、[運算子](#)、[函數](#)、[文字](#)、[案例陳述式](#)、[JSON Extensions](#)、[替代範本](#) [巢狀物件查詢](#) 和 [二進位承載](#)。

以使用 SELECT 子句來擷取傳入的 MQTT 訊息內的資訊。您也可以使用 SELECT \* 來擷取整個傳入訊息的承載。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}
```

如果承載為 JSON 物件，您可以參考物件中的鍵。您的傳出承載會包含鍵值組。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 關鍵字重新命名索引鍵。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"}
```

您可以選擇多個項目，以逗號分隔即可。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}
```

您可以選擇多個項目，包括以「\*」新增項目至來年的承載。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" 關鍵字產生傳出的承載不 JSON 物件。使用 SQL 版本 2015-10-08，您只能選取一個項目。使用 SQL 版本 2016-03-23 或更新版本，您也可以選取要作為頂層物件輸出的陣列。

## 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 '.' 語法來深入巢狀 JSON 物件傳入的承載。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}
```

如需如何使用 JSON 物件和包含保留字元 (例如數字或連字號 (減號) 字元) 之屬性名稱的相關資訊，請參閱 [JSON Extensions](#)

您可以使用函數 (請參閱 [函數](#))，將傳入的承載。您可以使用括號進行分組。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 子句可讓[主題](#)或[主題篩選條件](#)訂閱您的規則。以單引號 (') 括住主題或主題篩選條件。系統會為每則傳送到符合此處指定的主題篩選條件之 MQTT 主題的訊息觸發該規則。您可以使用主題篩選條件訂閱類似主題的群組。

範例：

發佈在主題 'topic/subtopic' 的傳入承載：{temperature: 50}

發佈在主題 'topic/subtopic-2' 的傳入承載：{temperature: 50}

SQL: "SELECT temperature AS t FROM 'topic/subtopic'".

規則已訂閱至 'topic/subtopic'，因此傳入的承載會傳遞至該規則。傳送至規則動作的外寄承載為：{t: 50}。規則並未給 'topic/subtopic-2' 訂閱，因此發佈在 'topic/subtopic-2' 的訊息不會觸發該規則。

# 萬用字元範例：

您可以使用 '#'(多層級) 萬用字元，以比對一或多個特定路徑元素：

發佈在主題 'topic/subtopic' 的傳入承載：{temperature: 50}。

發佈在主題 'topic/subtopic-2' 的傳入承載：{temperature: 60}。

發佈在主題 'topic/subtopic-3/details' 的傳入承載：{temperature: 70}。

發佈在主題 'topic-2/subtopic-x' 的傳入承載：{temperature: 80}。

```
SQL: "SELECT temperature AS t FROM 'topic/#'"。
```

已將該規則訂閱至開頭為 'topic' 的任何主題，因此該規則會執行三次，將傳出的 {t: 50} (表示主題/子主題)、{t: 60} (表示主題/子主題 2) 和 {t: 70} (表示主題/子主題 3/詳細資訊) 的承載傳送給其動作。它沒有訂閱 'topic-2/subtopic-x'，因此不會觸發 {temperature: 80} 訊息的規則。

+ 萬用字元範例：

您可以使用 '+' (單層級) 萬用字元，以比對任何一個特定路徑元素：

發佈在主題 'topic/subtopic' 的傳入承載：{temperature: 50}。

發佈在主題 'topic/subtopic-2' 的傳入承載：{temperature: 60}。

發佈在主題 'topic/subtopic-3/details' 的傳入承載：{temperature: 70}。

發佈在主題 'topic-2/subtopic-x' 的傳入承載：{temperature: 80}。

```
SQL: "SELECT temperature AS t FROM 'topic/+'"。
```

所有主題訂閱的規則有兩個路徑元素，第一個為：'topic'。規則是針對傳送至 'topic/subtopic' 和 'topic/subtopic-2' 的訊息執行，但不是針對 'topic/subtopic-3/details' (它具有比主題過濾條件更多的層級) 或 'topic-2/subtopic-x' (它不是以 topic 開頭) 執行。

## WHERE 子句

WHERE 子句會判斷是否執行某規則指定的動作。如果 WHERE 子句判斷值為 true，則規則動作已執行。否則，規則動作就不會執行。

WHERE 子句支援 [資料類型](#)、[運算子](#)、[函數](#)、[文字](#)、[案例陳述式](#)、[JSON Extensions](#)、[替代範本](#) 和 [巢狀物件查詢](#)。

範例：

發佈在 topic/subtopic 的傳入承載：{"color": "red", "temperature": 40}。

```
SQL: SELECT color AS my_color FROM 'topic/subtopic' WHERE temperature > 50 AND color <> 'red'。
```

在此情況下，該規則會觸發，但不會執行由規則指定的動作。將不會出現傳出承載。

您可以在 WHERE 子句中使用函數和運算子。不過您無法參考任何在 SELECT 中以 AS 關鍵字建立的別名。(系統會優先評估 WHERE 子句，以判定 SELECT 是否受到評估。)

非 JSON 承載的範例：

在「主題/副主題」上發佈的傳入非 JSON 承載：`80`


```
SQL:`SELECT decode(encode(*, 'base64'), 'base64') AS value FROM 'topic/subtopic' WHERE decode(encode(*, 'base64'), 'base64') > 50`
```

在此情況下，該規則會觸發，並且執行由規則指定的動作。傳出承載將由 SELECT 子句轉換為 JSON 承載 {"value":80}。

## 資料類型

AWS IoT 規則引擎支援所有 JSON 資料類型。

支援的資料類型

Type	意義
Int	離散的 Int。最多 34 位數。
Decimal	<p>精度為 34 位的 Decimal，最小非零量級為 1E-999，最大量級為 9.999...E999。</p> <div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> <b>Note</b></p> <p>有些函數傳回的 Decimal 值為雙精度，而非 34 位數的精度。使用 SQL V2 (2016-03-23) 時，數值若為整數 (例如 10.0) 會被當成 Int 值 (10) 處理，而不是預期的 Decimal 值 (10.0)。若要可靠地將整數數值當成 Decimal 值處理，請針對規則查詢陳述式使用 SQL V1 (2015-10-08)。</p> </div>
Boolean	True 或 False
String	UTF-8 字串。



Type	意義
Array	一系列類型無需相同的值。
Object	由鍵和值組成的 JSON 值。鍵必須為字串。值可以為任何類型。
Null	Null 是由 JSON 定義。此為用來代表缺少某個值的實際值。您可以在 SQL 陳述式中使用 Null 的關鍵字來明確建立 Null 的值。例如："SELECT NULL AS n FROM 'topic/su btopic'"
Undefined	<p>並非值。無法以 JSON 明確顯示，除非是省略該值。舉例而言，在物件 {"foo": null} 中，「foo」鍵會傳回 NULL，但「bar」鍵傳回 Undefined。SQL 語言在內部將 Undefined 視為一個值，但無法以 JSON 顯示，因此序列化為 JSON 時，結果會是 Undefined。</p> <pre>{"foo":null, "bar":undefined}</pre> <p>序列化為 JSON 如下：</p> <pre>{"foo":null}</pre> <p>同樣地，Undefined 由自身序列化時，會轉換為空白字串。以無效引數 (例如，錯誤類型、錯誤的引數數量等等) 呼叫的函數會傳回 Undefined。</p>

## 轉換

下表列出的是當某個類型的值轉換為另一種類型 (給函數指定錯誤類型的值) 所得的結果。舉例而言，如果絕對值的函數「abs」(預期為 Int 或 Decimal) 得到的是 String，其會嘗試按照下列規則將 String 轉換為 Decimal。此例中，'abs("-5.123")' 會視為 'abs(-5.123)'。

**Note**

不會企圖轉換為 Array、Object、Null、Undefined。

## To Decimal

引數類型	結果
Int	無小數點的 Decimal。
Decimal	來源值。
Boolean	Undefined 。(您可以明確使用 Cast 函數轉換 true = 1.0、false = 0.0。)
String	SQL 引擎會嘗試將字串剖析為 Decimal。AWS IoT 嘗試解析匹配正則表達式的字符串： <code>^-?\d+(\.\d+)?((?i)E-?\d+)?\$</code> 。 「0」、「-1.2」、「5E-12」均為自動轉換為 Decimal 的範例字串。
Array	Undefined 。
物件	Undefined 。
Null	Null。
未定義	Undefined 。

## To Int

引數類型	結果
Int	來源值。
Decimal	四捨五入到最接近 Int 的來源值。
Boolean	Undefined 。(您可以明確使用 Cast 函數轉換 true = 1.0、false = 0.0。)

引數類型	結果
String	SQL 引擎會嘗試將字串剖析為Decimal. AWS IoT 嘗試解析匹配正則表達式的字符串： <code>^-?\d+(\.\d+)?((?i)E-?\d+)?\$</code> 。 「0」，「-1.2」，「5E-12」都是自動轉換為Decimals 的字符串示例。AWS IoT 嘗試將其轉換為aDecimal，然後截斷該String字符串的小數位以創建。Decimal Int
Array	Undefined .
物件	Undefined .
Null	Null.
未定義	Undefined .

## To Boolean

引數類型	結果
Int	Undefined。(您可以明確使用 cast 函數轉換 0 = False、any_nonzero_value = True。)
Decimal	Undefined。(您可以明確使用 Cast 函數轉換 0 = False、any_nonzero_value = True。)
Boolean	原始的值。
String	「true」 = True 而「false」 = False (不區分大小寫)。其他字符串值為 Undefined。
Array	Undefined .
物件	Undefined .
Null	Undefined .

引數類型	結果
未定義	Undefined .

## To String

引數類型	結果
Int	以標準表示法表示的 Int 的字串顯示方式。
Decimal	代表 Decimal 值的字串，可能是採取科學表示法。
Boolean	「true」或「false」。所有小寫。
String	原始的值。
陣列	序列化為 JSON 的 Array。該結果字串為以逗號分隔的清單，並包含在方括弧內。String 在括號中。Decimal、Int、Boolean 和 Null 則不是。
物件	序列化為 JSON 的物件。該結果字串為以逗號分隔的鍵值組清單，並以大括號為開頭和結尾。String 在括號中。Decimal、Int、Boolean 和 Null 則不是。
Null	Undefined .
未定義	未定義。

## 運算子

下列運算子可用於 SELECT 和 WHERE 子句。

## AND 運算子

傳回 Boolean 結果。執行邏輯 AND 運算。如果左右運算元為 true，即傳回 true。否則會傳回 false。需要 Boolean 運算元或不區分大小寫的「true」或「false」字串運算元。

語法：*expression* AND *expression*。

### AND 運算子

左運算元	右運算元	輸出
Boolean	Boolean	Boolean。如果兩個運算元皆為 true 即為 true。否則為 false。
String/Boolean	String/Boolean	如果所有字串均為「true」或「false」(不區分大小寫)，他們會轉換為 Boolean，並以 <i>boolean</i> AND <i>boolean</i> 的方式正常處理。
其他值	其他值	Undefined .

## OR 運算子

傳回 Boolean 結果。執行邏輯 OR 運算。如果右運算元或左運算元有一個為 true 即傳回 true。否則會傳回 false。需要 Boolean 運算元或不區分大小寫的「true」或「false」字串運算元。

語法：*expression* OR *expression*。

### OR 運算子

左運算元	右運算元	輸出
Boolean	Boolean	Boolean。如果有一個運算元為 true 即為 true。否則為 false。
String/Boolean	String/Boolean	如果所有字串均為「true」或「false」(不區分大小寫)，它們會轉換為布林值，並以 <i>boolean</i> OR <i>boolean</i> 的方式正常處理。
其他值	其他值	Undefined .

## NOT 運算子

傳回 Boolean 結果。執行邏輯 NOT 運算。如果運算元為 false 即傳回 true。否則即傳回 true。需要 Boolean 運算元或不區分大寫的「true」或「false」字串運算元。

語法：NOT *expression*。

### NOT 運算子

運算元	輸出
Boolean	Boolean。如果運算元為 false 即為 true。否則為 true。
String	如果字串為「true」或「false」(不區分大小寫)，則會轉換為對應的布林值，並傳回相反的值。
其他值	Undefined .

## IN 運算子

傳回 Boolean 結果。您可以在 WHERE 子句中使用 IN 運算子來檢查值是否與陣列中的任何值相符。如果找到匹配，則返回 true，否則返回 false。

語法：*expression* IN *expression*。

### IN 運算子

左運算元	右運算元	輸出
Int/Decimal/String/A	Array	如果在數組中找到 Integer Decimal StringArray/Object 元素，則為真。否則為 false。

範例：

```
SQL: "select * from 'a/b' where 3 in arr"
```

```
JSON: {"arr": [1, 2, 3, "three", 5.7, null]}
```

在此範例中，條件子句 `where 3 in arr` 將評估為 `true`，因為 3 存在於名為的陣列中 `arr`。因此，在 SQL 語句中，`select * from 'a/b'` 將執行。這個範例也顯示陣列可以是異質的。

## 存在運算符

傳回 Boolean 結果。您可以在條件子句中使用 EXISTS 運算子來測試子查詢中是否存在元素。如果子查詢返回一個或多個元素，如果子查詢返回沒有元素返回 `false` 它返回 `true`。

語法：*expression*。

範例：

```
SQL: "select * from 'a/b' where exists (select * from arr as a where a = 3)"
```

```
JSON: {"arr":[1, 2, 3]}
```

在此範例中，條件子句 `where exists (select * from arr as a where a = 3)` 將評估為 `true`，因為 3 存在於名為的陣列中 `arr`。因此，在 SQL 語句中，`select * from 'a/b'` 將執行。

範例：

```
SQL: select * from 'a/b' where exists (select * from e as e where foo = 2)
```

```
JSON: {"foo":4,"bar":5,"e":[{"foo":1},{"foo":2}]}
```

在此範例中，因為 JSON 物件 `e` 內的陣列包含物件，因此條件子句 `where exists (select * from e as e where foo = 2)` 會評估為 `true {"foo":2}`。因此，在 SQL 語句中，`select * from 'a/b'` 將執行。

## > 運算子

傳回 Boolean 結果。如果左運算元大於右運算元即傳回 `true`。兩個運算元均轉換為 `Decimal`，再做比較。

語法：*expression* > *expression*。

> 運算子

左運算元	右運算元	輸出
Int/Decimal	Int/Decimal	Boolean。如果左運算元大於右運算元即為 <code>true</code> 。否則為 <code>false</code> 。

左運算元	右運算元	輸出
String/Int/Deci	String/Int/Deci	如果所有字串均可轉換為 Decimal，即 Boolean。如果左運算元大於右運算元即傳回 true。否則為 false。
其他值	Undefined .	Undefined .

## >= 運算子

傳回 Boolean 結果。如果左運算元大於或等於右運算元，即傳回 true。兩個運算元均轉換為 Decimal，再做比較。

語法：*expression* >= *expression*。

## >= 運算子

左運算元	右運算元	輸出
Int/Decimal	Int/Decimal	Boolean。如果左運算元大於或等於右運算元，即為 true。否則為 false。
String/Int/Deci	String/Int/Deci	如果所有字串均可轉換為 Decimal，即 Boolean。如果左運算元大於或等於右運算元，即傳回 true。否則為 false。
其他值	Undefined .	Undefined .

## < 運算子

傳回 Boolean 結果。如果左側運算元少於右運算元。兩個運算元均轉換為 Decimal，再做比較。

語法：*expression* < *expression*。

## < 運算子

左運算元	右運算元	輸出
Int/Decimal	Int/Decimal	Boolean。如果左運算元小於右運算元即為 true。否則為 false。



左運算元	右運算元	輸出
String/Int/Deci	String/Int/Deci	如果所有字串均可轉換為 Decimal，即 Boolean。如果左側運算元少於右運算元。否則為 false。
其他值	Undefined	Undefined

## <= 運算子

傳回 Boolean 結果。如果左運算元小於或等於右運算元，即傳回 true。兩個運算元均轉換為 Decimal，再做比較。

語法：*expression* <= *expression*。

## <= 運算子

左運算元	右運算元	輸出
Int/Decimal	Int/Decimal	Boolean。如果左運算元小於或等於右運算元，即為 true。否則為 false。
String/Int/Deci	String/Int/Deci	如果所有字串均可轉換為 Decimal，即 Boolean。如果左運算元小於或等於右運算元，即傳回 true。否則為 false。
其他值	Undefined	Undefined

## <> 運算子

傳回 Boolean 結果。如果左右運算元不相等，即傳回 true。否則即傳回 false。

語法：*expression* <> *expression*。

## <> 運算子

左運算元	右運算元	輸出
Int	Int	如果左運算元不等於右運算元即為 true。否則為 false。

左運算元	右運算元	輸出
Decimal	Decimal	如果左運算元不等於右運算元即為 true。否則為 false。在做比較前，先將 Int 轉換為 Decimal。
String	String	如果左運算元不等於右運算元即為 true。否則為 false。
陣列	陣列	如果各運算元的項目不相等且順序不同，即為 true。否則為 false
物件	物件	如果各運算元的鍵和值不相同，即為 true。否則為 false。鍵/值的順序不重要。
Null	Null	False。
任何值	Undefined	未定義。
Undefined	任何值	未定義。
類型不符合	類型不符合	True。

## = 運算子

傳回 Boolean 結果。如果兩個左右運算元相同，即傳回 true。否則即傳回 false。

語法：*expression* = *expression*。

## = 運算子

左運算元	右運算元	輸出
Int	Int	如果左運算元等於右運算元即為 true。否則為 false。
Decimal	Decimal	如果左運算元等於右運算元即為 true。否則為 false。在做比較前，先將 Int 轉換為 Decimal。
String	String	如果左運算元等於右運算元即為 true。否則為 false。
陣列	陣列	如果各運算元的項目相等且順序相同，即為 true。否則為 false。

左運算元	右運算元	輸出
物件	物件	如果各運算元的鍵和值相同，即為 true。否則為 false。鍵/值的順序不重要。
任何值	Undefined	Undefined .
Undefined	任何值	Undefined .
類型不符合	類型不符合	False。

## + 運算子

「+」是過載的運算子。可以用來連接或新增字串。

語法：*expression* + *expression*。

## + 運算子

左運算元	右運算元	輸出
String	任何值	將右運算元轉換為字串，並連接至左運算元的尾端。
任何值	String	將左運算元轉換為字串，並將右運算元連接至轉換後的左運算元的尾端。
Int	Int	Int 值。將運算元相加。
Int/Decimal	Int/Decimal	Decimal 值。將運算元相加。
其他值	其他值	Undefined .

## - 運算子

從左運算元中減去右運算元。

語法：*expression* - *expression*。

## - 運算子

左運算元	右運算元	輸出
Int	Int	Int 值。從左運算元中減去右運算元。
Int/Decimal	Int/Decimal	Decimal 值。從左運算元中減去右運算元。
String/Int/Decimal	String/Int/Decimal	如果所有字串都正確轉換為小數，則會傳回 Decimal 值。從左運算元中減去右運算元。如果不是，則傳回 Undefined 。
其他值	其他值	Undefined 。
其他值	其他值	Undefined 。

## \* 運算子

將左運算元乘以右運算元。

語法： *expression* \* *expression*。

## \* 運算子

左運算元	右運算元	輸出
Int	Int	Int 值。將左運算元乘以右運算元。
Int/Decimal	Int/Decimal	Decimal 值。將左運算元乘以右運算元。
String/Int/Decimal	String/Int/Decimal	如果所有字串都正確轉換為小數，則會傳回 Decimal 值。將左運算元乘以右運算元。如果不是，則傳回 Undefined 。
其他值	其他值	Undefined 。

## / 運算子

將左運算元除以右運算元。

語法： *expression* / *expression*。

## / 運算子

左運算元	右運算元	輸出
Int	Int	Int 值。將左運算元除以右運算元。
Int/Decimal	Int/Decimal	Decimal 值。將左運算元除以右運算元。
String/Int/Decimal	String/Int/Decimal	如果所有字串都正確轉換為小數，則會傳回 Decimal 值。將左運算元除以右運算元。如果不是，則傳回 Undefined 。
其他值	其他值	Undefined 。

## % 運算子

傳回左運算元除以右運算元的餘數。

語法：*expression* % *expression*。

## % 運算子

左運算元	右運算元	輸出
Int	Int	Int 值。傳回左運算元除以右運算元的餘數。
String/Int/Decimal	String/Int/Decimal	如果所有字串都正確轉換為小數，則會傳回 Decimal 值。傳回左運算元除以右運算元的餘數。否則為 Undefined 。
其他值	其他值	Undefined 。

## 函數

您可以在 SQL 表達式的 SELECT 或 WHERE 子句中使用下列內建函數。

## abs(Decimal)

傳回某個數字的絕對值。受 SQL 版本 2015-10-08 和更新版本支援。

範例：abs(-5) 傳回 5。

引數類型	結果
Int	Int , 引數的絕對值。
Decimal	Decimal , 引數的絕對值。
Boolean	Undefined .
String	Decimal。結果為引數的絕對值。如果字串無法轉換，則結果為 Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

### accountid()

將擁有此規則的帳戶 ID 傳回為 String。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
accountid() = "123456789012"
```

### acos(Decimal)

以弧度傳回數字的反餘弦值。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例： $\text{acos}(0) = 1.5707963267948966$

引數類型	結果
Int	Decimal (使用雙精度) , 引數的反向餘弦值。傳回的虛數結果為 Undefined 。

引數類型	結果
Decimal	Decimal (使用雙精度), 引數的反向餘弦值。傳回的虛數結果為 Undefined 。
Boolean	Undefined 。
String	Decimal, 引數的反向餘弦值。如果字串無法轉換, 則結果為 Undefined 。
Array	Undefined 。
物件	Undefined 。
Null	Undefined 。
未定義	Undefined 。

## asin(Decimal)

以弧度傳回數字的反正弦值。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例 :  $\text{asin}(0) = 0.0$

引數類型	結果
Int	Decimal (使用雙精度), 引數的反向正弦值。傳回的虛數結果為 Undefined 。
Decimal	Decimal (使用雙精度), 引數的反向正弦值。傳回的虛數結果為 Undefined 。
Boolean	Undefined 。
String	Decimal (使用雙精度), 引數的反向正弦值。如果字串無法轉換, 則結果

引數類型	結果
	為 Undefined 。傳回的虛數結果為 Undefined 。
Array	Undefined 。
物件	Undefined 。
Null	Undefined 。
未定義	Undefined 。

### atan(Decimal)

以弧度傳回數字的反正切值。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例： $\text{atan}(0) = 0.0$

引數類型	結果
Int	Decimal (使用雙精度)，引數的反向正切值。傳回的虛數結果為 Undefined 。
Decimal	Decimal (使用雙精度)，引數的反向正切值。傳回的虛數結果為 Undefined 。
Boolean	Undefined 。
String	Decimal，引數的反向正切值。如果字串無法轉換，則結果為 Undefined 。傳回的虛數結果為 Undefined 。
Array	Undefined 。
物件	Undefined 。
Null	Undefined 。



引數類型	結果
未定義	Undefined .

## atan2(Decimal, Decimal)

以弧度傳回 x 軸正軸和兩個引數所定義的 (x, y) 點之間的角度。逆時針角度為正值 (上半象限,  $y > 0$ )，順時鐘的角度為負值 (下半象限  $y < 0$ )。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例： $\text{atan2}(1, 0) = 1.5707963267948966$

引數類型	引數類型	結果
Int/Decimal	Int/Decimal	Decimal (使用雙精度), x 軸與點之間的角度。
Int/Decimal/String	Int/Decimal/String	Decimal, 所述之點的反向正切串無法轉換, 則結果為 Undefined .
其他值	其他值	Undefined .

## aws\_lambda (functionArn、inputJson)

呼叫指定的 Lambda 函數，其會將 inputJson 傳送至 Lambda 函數，並傳回 Lambda 函數產生的 JSON。

### 引數

引數	描述
functionArn	Lambda 函數呼叫的 ARN。Lambda 函數必須傳回 JSON 資料。
inputJson	傳遞到 Lambda 函數的 JSON 輸入。若要傳遞巢狀物件查詢和文字，您必須使用 SQL 版本 2016-03-23。

您必須授與 AWS IoT `lambda:InvokeFunction` 權限才能叫用指定的 Lambda 函數。下列範例顯示了如何使用 AWS CLI 授與 `lambda:InvokeFunction` 的許可：

```
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"
```

以下為 add-permission 命令的引數：

#### --function-name

Lambda 函數的名稱。您可以新增許可來更新函數的資源政策。

#### --region

您 AWS 區域 的帳戶。

#### --principal

取得許可的委託人。這應該是 `iot.amazonaws.com` 為了允 AWS IoT 許調用 Lambda 函數的權限。

#### --source-arn

該項規則的 ARN。您可以使用 `get-topic-rule` AWS CLI 指令取得規則的 ARN。

#### --source-account

規則 AWS 帳戶 的定義位置。

#### --statement-id

專屬的陳述式識別符。

#### --action

您想要在此陳述式中允許的 Lambda 動作。若要允許 AWS IoT 叫用 Lambda 函數，請指定 `lambda:InvokeFunction`。

#### Important

如果您在未提供 `source-arn` 或的情況下為 AWS IoT 主體新增權限 `source-account`，則使 AWS 帳戶 用 Lambda 動作建立規則的任何使用者都可以觸發規則以呼叫 Lambda 函數 AWS IoT。如需詳細資訊，請參閱 [Lambda 許可模型](#)。

指定的 JSON 訊息承載，如以下所示：

```
{
  "attribute1": 21,
  "attribute2": "value"
}
```

aws\_lambda 函數可用於呼叫 Lambda 函數，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'
```

如果您想要傳遞完整的 MQTT 訊息承載，您可以使用「\*」來指定 JSON 承載，如下列範例所示。

```
SELECT
aws_lambda("arn:aws:lambda:us-east-1:account_id:function:lambda_function", *) as output
FROM 'topic-filter'
```

payload.inner.element 會從發佈在「主題/子主題」主題上的訊息選取資料。

some.value 會從 Lambda 函數產生的輸出結果中選取資料。

#### Note

該規則引擎會限制 Lambda 函數的執行期間。規則的 Lambda 函數呼叫應該會在 2000 毫秒內完成。

## bitand (Int, Int)

在兩個 Int (已轉換) 引數的位元表現上執行按位元的 AND 運算。受 SQL 版本 2015-10-08 和更新版本支援。

範例：bitand(13, 5) = 5

引數類型	引數類型	結果
Int	Int	Int，兩個引數按位元的 AND 運算結果

引數類型	引數類型	結果
Int/Decimal	Int/Decimal	Int，兩個引數按位元的 AND 運算。Int 的數字會無條件捨去至最接近的 Int (整數)。如果轉換失敗，則結果為 Undefined。
Int/Decimal/String	Int/Decimal/String	Int，兩個引數按位元的 AND 運算。字串都會轉換為小數，並會無條件捨去至最接近的 Int (整數)。如果轉換失敗，則結果為 Undefined。
其他值	其他值	Undefined。

### bitor(Int, Int)

在兩個引數的位元表現上執行按位元的 OR 運算。受 SQL 版本 2015-10-08 和更新版本支援。

範例：`bitor(8, 5) = 13`

引數類型	引數類型	結果
Int	Int	Int，兩個引數按位元的 OR 運算。
Int/Decimal	Int/Decimal	Int，兩個引數按位元的 OR 運算。Int 的數字會無條件捨去至最接近的 Int (整數)。如果轉換失敗，則結果為 Undefined。
Int/Decimal/String	Int/Decimal/String	Int，兩個引數按位元的 OR 運算。字串都會轉換為小數，並會無條件捨去至最接近的 Int (整數)。如果轉換失敗，則結果為 Undefined。
其他值	其他值	Undefined。

## bitxor(Int, Int)

在兩個 Int (已轉換) 引數的位元表現上執行按位元的 XOR 運算。受 SQL 版本 2015-10-08 和更新版本支援。

範例：`bitxor(13, 5) = 8`

引數類型	引數類型	結果
Int	Int	Int，兩個引數按位元的 XOR 運算。
Int/Decimal	Int/Decimal	Int，兩個引數按位元的 XOR 運算。Int 的數字會無條件捨去至最接近 Int 的數字。
Int/Decimal/String	Int/Decimal/String	Int 是在兩個引數上的位元 XOR 運算。字串會轉換為小數，並無條件捨去至最接近 Int 的數字。如果任何轉換失敗，則結果為 Undefined。
其他值	其他值	Undefined。

## bitnot(Int)

在 Int (已轉換) 引數的位元表現上執行按位元的 NOT 運算。受 SQL 版本 2015-10-08 和更新版本支援。

範例：`bitnot(13) = 2`

引數類型	結果
Int	Int，引數按位元的 NOT 運算。
Decimal	Int，引數按位元的 NOT 運算。Decimal 值會無條件捨去至最接近 Int 的值。
String	Int，引數按位元的 NOT 運算。字串會轉換為小數，並會無條件捨去至最接近 Int 的值。

引數類型	結果
	Int (整數)。如果任何轉換失敗，則結果為 Undefined 。
其他值	其他值。

## cast()

將一個值從某個資料類型轉換至另一個類型。除了增加了將數字和布林值相互轉換的能力外，轉換行為大致如同標準轉換。如果 AWS IoT 無法確定如何將一種類型轉換為另一種類型，則結果為 Undefined。受 SQL 版本 2015-10-08 和更新版本支援。格式：`cast(value as type)`。

範例：

```
cast(true as Int) = 1
```

在呼叫 cast 時，以下關鍵字可能出現在「as」之後：

對於 SQL 版本 2015-10-08 與 2016-03-23

關鍵字	結果
String	將值轉換為 String。
Nvarchar	將值轉換為 String。
文字	將值轉換為 String。
Ntext	將值轉換為 String。
varchar	將值轉換為 String。
Int	將值轉換為 Int。
Integer	將值轉換為 Int。
Double	將值轉換為 Decimal (使用雙精度)。

此外，對於 SQL 版本 2016-03-23

關鍵字	結果
Decimal	將值轉換為 Decimal。
Bool	將值轉換為 Boolean。
Boolean	將值轉換為 Boolean。

轉換規則：

轉換為小數

引數類型	結果
Int	無小數點的 Decimal。
Decimal	來源值。  <div data-bbox="511 976 1136 1438" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"> <p> Note</p> <p>使用 SQL V2 (2016-03-23) 時，數值若為整數 (例如 10.0) 會傳回 Int 值 (10)，而不是預期的 Decimal 值 (10.0)。若要可靠地將整數數值當成 Decimal 值處理，請針對規則查詢陳述式使用 SQL V1 (2015-10-08)。</p> </div>
Boolean	true = 1.0、false = 0.0。
String	嘗試將字串剖析為 Decimal。AWS IoT 會嘗試剖析字串以符合正規表示式： <code>^-?\d+(\.\d+)?((?i)E-?\d+)?\$</code> 。「0」、「-1.2」、「5E-12」均為自動轉換為 Decimal 的範例字串。
Array	Undefined .

引數類型	結果
物件	Undefined .
Null	Undefined .
未定義	Undefined .

### 轉換為 Int

引數類型	結果
Int	來源值。
Decimal	無條件捨去至最接近 Int 的來源值。
Boolean	true = 1.0、false = 0.0。
String	嘗試將字串剖析為 Decimal。AWS IoT 會嘗試剖析字串以符合正規表示式： <code>^-?\d+(\.\d+)?((?i)E-?\d+)?\$</code> 。「0」、「-1.2」、「5E-12」均為自動轉換為 Decimal 的範例字串。AWS IoT 會嘗試將字串轉換為 Decimal，並無條件捨去至最接近的 Int。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

### 轉換到 Boolean

引數類型	結果
Int	0 = False , any_nonzero_value = True。



引數類型	結果
Decimal	0 = False , any_nonzero_value = True。
Boolean	來源值。
String	「true」 = True 而 「false」 = False (不區分大小寫)。其他字串值 = Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

## 轉換為字串

引數類型	結果
Int	以標準表示法表示的 Int 的字串顯示方式。
Decimal	代表 Decimal 值的字串，可能是採取科學表示法。
Boolean	「true」或「false」，均為小寫。
String	"true"=True 及 "false"=False (區分大小寫)。其他字串值 = Undefined 。
陣列	序列化為 JSON 的陣列。結果字串是以方括號括住，並以逗號分隔的清單。String 在括號中。Decimal、Int 和 Boolean 則不是。
物件	序列化為 JSON 的物件。JSON 字串是首尾以大括號括住，並以逗號分隔的鍵值組清單。String 在括號

引數類型	結果
	中。Decimal、Int、Boolean 和 Null 則不是。
Null	Undefined .
未定義	Undefined .

## ceil(Decimal)

將指定的 Decimal 無條件進位至最近的 Int。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
ceil(1.2) = 2
```

```
ceil(-1.2) = -1
```

引數類型	結果
Int	Int , 引數值。
Decimal	Int , 無條件進位至最接近的 Decimal 的 Int 值。
String	Int。此字串會轉換成 Decimal 並四捨五入至最接近 Int。如果字串無法轉換為 Decimal , 則結果為 Undefined 。
其他值	Undefined .

## chr(String)

傳回指定的 Int 引數對應到的 ASCII 字元。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
chr(65) = "A"。
```

`chr(49) = "1"`。

引數類型	結果
Int	對應到指定的 ASCII 值的字元。如果引數並非有效的 ASCII 值，結果會為 Undefined 。
Decimal	對應到指定的 ASCII 值的字元。Decimal 的引數值會無條件捨去至最接近 Int 的值。如果引數並非有效的 ASCII 值，結果會為 Undefined 。
Boolean	Undefined 。
String	如果 String 可以轉換為 Decimal，要無條件捨去到最接近的 Int。如果引數並非有效的 ASCII 值，結果會為 Undefined 。
Array	Undefined 。
物件	Undefined 。
Null	Undefined 。
其他值	Undefined 。

## clientid()

傳回傳送訊息的 MQTT 用戶端的 ID，如果訊息不是透過 MQTT 傳送，則為 n/a。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
clientid() = "123456789012"
```

## concat()

連接陣列或字串。此函數會接受任意數量的引數，並傳回 String 或 Array。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
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]
```

引數數量	結果
0	Undefined .
1	傳回的引數未經修改。
2+	如果任一引數為 Array，則結果會是包含所有引數的單一陣列。若沒有列出引數，且至少有一個引數是 String，則結果為所有引數 String 表示法的串聯。使用之前列出的標準轉換將引數轉換為字串。

## cos(Decimal)

以弧度傳回數字的餘弦值。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
cos(0) = 1。
```

引數類型	結果
Int	Decimal (使用雙精度), 引數的餘弦值。傳回的虛數結果為 Undefined 。
Decimal	Decimal (使用雙精度), 引數的餘弦值。傳回的虛數結果為 Undefined 。
Boolean	Undefined 。
String	Decimal (使用雙精度), 引數的餘弦值。如果字串無法轉換為 Decimal, 則結果為 Undefined 。
Array	Undefined 。
物件	Undefined 。
Null	Undefined 。
未定義	Undefined 。

## cosh(Decimal)

以弧度傳回數字的雙曲餘弦值。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例： $\cosh(2.3) = 5.037220649268761$ 。

引數類型	結果
Int	Decimal (使用雙精度), 引數的雙曲餘弦值。傳回的虛數結果為 Undefined 。
Decimal	Decimal (使用雙精度), 引數的雙曲餘弦值。傳回的虛數結果為 Undefined 。
Boolean	Undefined 。

引數類型	結果
String	Decimal (使用雙精度), 引數的雙曲餘弦值。如果字串無法轉換為 Decimal, 則結果為 Undefined。傳回的虛數結果為 Undefined。
Array	Undefined。
物件	Undefined。
Null	Undefined。
未定義	Undefined。

## decode(value, decodingScheme)

使用 decode 函數來解碼已編碼值。如果解碼字串為 JSON 文件, 則會傳回可定址物件。否則, 解碼字串會當成字串傳回。如果字串無法解碼, 函數會傳回 NULL。此功能支援解碼 base64 編碼字串及協定緩衝區 (protobuf) 訊息格式。

受 SQL 版本 2016-03-23 和更新版本支援。

### value

字串值或任何有效的表達式 (如 [AWS IoT SQL 參考資料](#) 所定義), 其會傳回字串。

### decodingScheme

代表用來解碼值之結構描述的文字字串。目前僅支援 'base64' 和 'proto'。

### 對 base64 編碼字串進行解碼

在此範例中, 訊息承載包含編碼值。

```
{
  encoded_temp: "eyAidGVtcGVyYXR1cmUiOiAzMyB9Cg=="
}
```

SQL 陳述式中的 decode 函數會解碼訊息承載中的值。

```
SELECT decode(encoded_temp,"base64").temperature AS temp from 'topic/subtopic'
```

解碼 `encoded_temp` 值會產生下列有效的 JSON 文件，允許 `SELECT` 陳述式讀取溫度值。

```
{ "temperature": 33 }
```

這裡顯示此範例中 `SELECT` 陳述式的結果。

```
{ "temp": 33 }
```

如果解碼值不是有效的 JSON 文件，解碼值將以字串形式傳回。

### 對 protobuf 訊息承載進行解碼

您可以使用解碼 SQL 函數來設定可對 protobuf 訊息承載進行解碼的規則。如需詳細資訊，請參閱[對 protobuf 訊息承載進行解碼](#)。

功能簽章外觀如下：

```
decode(<ENCODED DATA>, 'proto', '<S3 BUCKET NAME>', '<S3 OBJECT KEY>', '<PROTO NAME>', '<MESSAGE TYPE>')
```

### ENCODED DATA

指定要解碼的 protobuf 編碼資料。如果傳送到規則的整個訊息是 protobuf 編碼資料，則可以使用 \* 參考原始二進位傳入承載。否則，此欄位必須是 base-64 編碼的 JSON 字串，而且可以直接傳入字串的參考。

1) 要解碼原始二進位 protobuf 傳入承載：

```
decode(*, 'proto', ...)
```

2) 要解碼由 base64 編碼字串 'a.b' 表示的 protobuf 編碼訊息：

```
decode(a.b, 'proto', ...)
```

### proto

指定要以 protobuf 訊息格式解碼的資料。如果您指定 base64 而不是 proto，此函數會將 base64 編碼字串解碼為 JSON。

## S3\_BUCKET\_NAME

您用來上傳 FileDescriptorSet 檔案的 Amazon S3 儲存貯體名稱。

## S3\_OBJECT\_KEY

用來在 Amazon S3 儲存貯體中指定 FileDescriptorSet 檔案的物件索引鍵。

## PROTO\_NAME

從中產生 FileDescriptorSet 檔案的 .proto 檔案名稱 (不含副檔名)。

## MESSAGE\_TYPE

待解碼資料在 FileDescriptorSet 檔案中所應符合的 Protobuf 訊息結構名稱。

使用解碼 SQL 函數的 SQL 運算式可能顯示如下：

```
SELECT VALUE decode(*, 'proto', 's3-bucket', 'messageformat.desc', 'myproto',  
'messagetype') FROM 'some/topic'
```

- \*

表示二進位傳入承載，符合名為 mymessagetype 的 Protobuf 訊息類型。

- messageformat.desc

存放在名為 s3-bucket 的 Amazon S3 儲存貯體中的 FileDescriptorSet 檔案。

- myproto

此原始 .proto 檔案的用途是產生名為 myproto.proto 的 FileDescriptorSet 檔案。

- messagetype

如 myproto.proto 中所定義名為 messagetype 的訊息類型(以及任何匯入的相依性)。

## encode(value, encodingScheme)

根據編碼機制，使用 encode 函數將承載 (可能並非 JSON 資料) 編碼為字串表現形式。受 SQL 版本 2016-03-23 和更新版本支援。



## value

任何有效的表達式，如 [AWS IoT SQL 參考資料](#) 的定義。無論承載是否為 JSON 格式，您都可以指定 \* 以編碼整個承載。若您提供了運算式，則在編碼之前，評估結果將轉換為字串。

## encodingScheme

代表您想要使用的編碼機制的文字字串。目前僅支援 'base64'。

## endswith(String, String)

傳回 Boolean，指出第一個 String 引數是否以第二個 String 引數結尾。如果引數為 Null 或 Undefined，則結果為 Undefined。受 SQL 版本 2015-10-08 和更新版本支援。

範例：endswith("cat", "at") = true。

引數類型 1	引數類型 2	結果
String	String	如果第一個引數以第二個引數結尾，則結果為 true。否則為 false。
其他值	其他值	所有引數都將使用標準轉換規則。如果第一個引數以第二個引數結尾，則結果為 true。否則為 false。如果引數為 Undefined，則結果為 Undefined。

## exp(Decimal)

傳回 e 次方的 Decimal 引數。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例：exp(1) = e。

引數類型	結果
Int	Decimal (使用雙精度)，e ^ 引數。
Decimal	Decimal (使用雙精度)，e ^ 引數。

引數類型	結果
String	Decimal (使用雙精度), e ^ 引數。如果 String 無法轉換為 Decimal , 則結果為 Undefined 。
其他值	Undefined 。

## floor(Decimal)

將指定的 Decimal 無條件進位至最接近的 Int。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
floor(1.2) = 1
```

```
floor(-1.2) = -2
```

引數類型	結果
Int	Int , 引數值。
Decimal	Int , Decimal 值會無條件捨去至最接近的 Int。
String	Int。此字串會轉換成 Decimal , 並無條件捨去至最接近的 Int。如果字串無法轉換為 Decimal , 則結果為 Undefined 。
其他值	Undefined 。

## get

從集合類型 (陣列、字串、物件) 擷取值。第一個引數不會套用任何轉換。轉換會按表格中的記錄套用至第二個引數。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
get(["a", "b", "c"], 1) = "b"
```

```
get({"a":"b"}, "a") = "b"
```

```
get("abc", 0) = "a"
```

引數類型 1	引數類型 2	結果
陣列	任何類型 (轉換為 Int)	第二個引數 (轉換為 Array) 所指定從零開始的索引的項目。如果轉換為負值，則結果為 Undefined。如果索引在 Array 的邊界之外 (負值或 $\geq$ array.length)，則結果為 Undefined。
字串	任何類型 (轉換為 Int)	第二個引數 (轉換為 Int) 所提供從零開始的索引的字元。如果轉換為負值，則結果為 Undefined。如果索引在字串的邊界之外 (負值或 $\geq$ string.length)，則結果為 Undefined。
物件	String (未套用轉換)	對應至第二個引數所提供的字串索引物件所儲存的值。
其他值	任何值	Undefined。

獲取動態 ( tableName 稱,,,,, partitionKeyName partitionKeyValue sortKeyName sortKeyValue)

從 DynamoDB 資料表擷取資料。get\_dynamodb() 允許您在評估規則時查詢 DynamoDB 資料表。您可以使用從 DynamoDB 中擷取的資料來篩選或增強訊息承載。受 SQL 版本 2016-03-23 和更新版本支援。

get\_dynamodb() 接受下列參數：

tableName

所要查詢 DynamoDB 資料表的名稱。

partitionKeyName

分割區索引鍵的名稱。如需詳細資訊，請參閱 [DynamoDB 索引鍵](#)。

## partitionKeyValue

用來識別記錄的分割區索引鍵值。如需詳細資訊，請參閱 [DynamoDB 索引鍵](#)。

## sortKeyName

(選用) 排序索引鍵的名稱。只有在查詢的 DynamoDB 資料表使用複合索引鍵時，才需要此參數。如需詳細資訊，請參閱 [DynamoDB 索引鍵](#)。

## sortKeyValue

(選用) 排序索引鍵的值。只有在查詢的 DynamoDB 資料表使用複合索引鍵時，才需要此參數。如需詳細資訊，請參閱 [DynamoDB 索引鍵](#)。

## roleArn

授予 DynamoDB 資料表存取權限的 IAM 角色 ARN。規則引擎會假設此角色代表您存取 DynamoDB 資料表。請避免使用過多許可的角色。僅授與角色規則所需的許可。以下是授與一個 DynamoDB 資料表存取權的範例政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "dynamodb:GetItem",
      "Resource": "arn:aws:dynamodb:aws-region:account-id:table/table-name"
    }
  ]
}
```

舉例說明如何使用 `get_dynamodb()`，假設您有一個 DynamoDB 資料表，其中包含所有連接至 AWS IoT 之裝置的裝置 ID 和位置資訊。下列 SELECT 陳述式使用 `get_dynamodb()` 函數來擷取指定裝置 ID 的位置：

```
SELECT *, get_dynamodb("InServiceDevices", "deviceId", id,
"arn:aws:iam::12345678910:role/getdynamo").location AS location FROM 'some/
topic'
```

**Note**

- 每個 SQL 陳述式最多可以呼叫一次 `get_dynamodb()`。在單一 SQL 陳述式中多次呼叫 `get_dynamodb()` 會導致規則終止，而不會呼叫任何動作。
- 如果 `get_dynamodb()` 傳回的資料超過 8 KB，則可能不會叫用規則的動作。

## `get_mqtt_property`(名稱)

參考下列任一 MQTT5 標頭：`contentType`、`payloadFormatIndicator`、`responseTopic`、和 `correlationData`。此函數會接受下列任何常值字串做為引數：`content_type`、`format_indicator`、`response_topic` 和 `correlation_data`。如需詳細資訊，請參閱下列函數引數表。

### ContentType

字串：描述發佈訊息內容的 UTF-8 編碼字串。

### payloadFormat指示器

字串：列舉字串值，用於表示承載是否已格式化為 UTF-8。有效值為 `UNSPECIFIED_BYTES` 和 `UTF8_DATA`。

### responseTopic

字串：UTF-8 編碼字串，用來當作回應訊息的主題名稱。回應主題是用來描述要在請求-回應流程中作為接收者發佈目標的主題。主題不得包含萬用字元。

### correlationData

字串：請求訊息的傳送者使用 base64 編碼的二進位資料，用以在收到回應訊息時識別其所對應的請求。

下表列出可接受的函數引數及其相關聯的 `get_mqtt_property` 函數傳回類型：

## 函數引數

SQL	傳回資料類型 (如果存在)	傳回資料類型 (如果不存在)
<code>get_mqtt_property("format_indicator")</code>	字串 (UNSPECIFIED_BYTES 或 UTF8_DATA)	字串 (NSPECIFIED_BYTES)
<code>get_mqtt_property("content_type")</code>	字串	未定義
<code>get_mqtt_property("response_topic")</code>	字串	未定義
<code>get_mqtt_property("correlation_data")</code>	base64 編碼字串	未定義
<code>get_mqtt_property("some_invalid_name")</code>	未定義	未定義

下列範例規則 SQL 會參考下列任一 MQTT5 標

頭：`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)`

擷取所加密 `SecretString` 或 `SecretBinary` 欄位的值，此欄位是 [AWS Secrets Manager](#) 中目前秘密版本的欄位。如需建立和維護密碼的詳細資訊 [CreateSecret](#)，請參閱 [UpdateSecret](#)、和 [PutSecretValue](#)。

`get_secret()` 接受下列參數：

## secretId

字串：要擷取之秘密的 Amazon 資源名稱 (ARN) 或易記名稱。

## secretType

字串：秘密類型。有效值：SecretString | SecretBinary。

### SecretString

- 對於透過使用 API、或 AWS Secrets Manager 主控台建立為 JSON 物件的密碼：AWS CLI
  - 如果您指定 key 參數的值，此函數會傳回所指定金鑰的值。
  - 如果未指定 key 參數的值，此函數會傳回整個 JSON 物件。
- 對於您使用 API 或 AWS CLI 建立為非 JSON 物件的秘密：
  - 如果您指定 key 參數的值，此函數會失敗並顯示例外狀況。
  - 如果未指定 key 參數的值，此函數會傳回秘密的內容。

### SecretBinary

- 如果您指定 key 參數的值，此函數會失敗並顯示例外狀況。
- 如果未指定 key 參數的值，此函數會以 Base64 編碼的 UTF-8 字串形式傳回秘密值。

## key

(選用) 字串：JSON 物件內的金鑰名稱，此物件存放在秘密的 SecretString 欄位中。當您只想要擷取存放在秘密中的秘密值，而不是整個 JSON 物件時，請使用此值。

如果您指定此參數的值，而且秘密並未在其 SecretString 欄位內包含 JSON 物件，此函數會失敗並顯示例外狀況。

## roleArn

字串：擁有 secretsmanager:GetSecretValue 和 secretsmanager:DescribeSecret 許可的角色 ARN。

### Note

此函數一律傳回目前版本的秘密 (標籤為 AWSCURRENT 的版本)。AWS IoT 規則引擎會快取每個密碼長達 15 分鐘。因此，規則引擎最多可能需要 15 分鐘來更新秘密。這意味著，如果您在更新後最多 15 分鐘內檢索密碼 AWS Secrets Manager，則此函數可能會返回以前的版本。

此功能未計量，但 AWS Secrets Manager 需要收費。由於秘密快取機制，規則引擎偶爾會呼叫 AWS Secrets Manager。因為規則引擎是完全分散式服務，您可能會在 15 分鐘快取時間範圍期間，看到多個來自規則引擎的 Secrets Manager API 呼叫。

範例：

您可以在 HTTPS 規則動作的身分驗證標題中使用 `get_secret` 函式，如下列 API 金鑰身分驗證範例所示。

```
"API_KEY": "${get_secret('API_KEY', 'SecretString', 'API_KEY_VALUE',  
'arn:aws:iam::12345678910:role/getsecret')}}"
```

如需 HTTPS 規則動作的詳細資訊，請參閱 [the section called “HTTP”](#)。

`get_thing_shadow(thingName, shadowName, roleARN)`

傳回指定物件的影子。受 SQL 版本 2016-03-23 和更新版本支援。

`thingName`

字串：想要擷取影子的物件名稱。

`shadowName`

(選用) 字串：影子的名稱。只有在參考已命名的影子時，才需要此參數。

`roleArn`

字串：擁有 `iot:GetThingShadow` 許可的角色 ARN。

範例：

與已命名的影子搭配使用時，請提供 `shadowName` 參數。

```
SELECT * from 'topic/subtopic'  
WHERE  
  get_thing_shadow("MyThing", "MyThingShadow", "arn:aws:iam::123456789012:role/  
AllowsThingShadowAccess")
```



```
.state.reported.alarm = 'ON'
```

與未命名影子搭配使用時，請省略 shadowName 參數。

```
SELECT * from 'topic/subtopic'
WHERE
  get_thing_shadow("MyThing","arn:aws:iam::123456789012:role/
AllowsThingShadowAccess")
  .state.reported.alarm = 'ON'
```

## 獲取用戶屬性 ( ) userPropertyKey

參考使用者屬性，這是 MQTT5 支援的一種屬性標題類型。

### userProperty

字串：使用者屬性是索引鍵/值對。此函數將索引鍵當作參數，並傳回所有符合關聯索引鍵的值陣列。

### 函數引數

對於下列位於訊息標頭中的使用者屬性：

金鑰	值
部分索引鍵	部分值
不同的索引鍵	不同的值
部分索引鍵	具有重複索引鍵的值

下表顯示預期的 SQL 行為：

SQL	傳回資料類型。	傳回資料類型。
get_user_properties(「部分索引鍵」)	字串陣列	['some value', 'value with duplicate key']

SQL	傳回資料類型。	傳回資料類型。
<code>get_user_properties(「其他索引鍵」)</code>	字串陣列	<code>['a different value']</code>
<code>get_user_properties()</code>	索引鍵/值對物件的陣列	<code>[{"some key": "some value"}, {"other key": "a different value"}, {"some key": "value with duplicate key"}]</code>
<code>get_user_properties(「不存在的索引鍵」)</code>	未定義	

下列範例規則 SQL 會將使用者屬性 (MQTT5 屬性標頭類型) 參考到承載中：

```
SELECT *, get_user_properties('user defined property key') as userProperty
FROM 'some/topic'
```

## 雜湊函數

AWS IoT 提供下列雜湊函數：

- md2
- md5
- sha1
- sha224
- sha256
- sha384
- sha512

所有雜湊函數都預期有一個字串引數。結果為該字串的雜湊值。標準字串轉換會套用至非字串的引數。SQL 版本 2015-10-08 和更新版本可支援所有雜湊函數。

範例：

```
md2("hello") = "a9046c73e00331af68917d3804f70655"
```

```
md5("hello") = "5d41402abc4b2a76b9719d911017c592"
```

## indexOf(String, String)

傳回第二個引數的第一個索引 (從 0 開始)，作為第一個引數的子字串。預期兩個引數均為字串。非字串的引數需遵守標準字串轉換規則。此函數不能套用在陣列上，僅可套用到字串上。受 SQL 版本 2016-03-23 和更新版本支援。

範例：

```
indexOf("abcd", "bc") = 1
```

## isNull()

如果引數是 Null 值，則傳回 true 受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
isNull(5) = false。
```

```
isNull(Null) = true。
```

引數類型	結果
Int	false
Decimal	false
Boolean	false
String	false
Array	false
Object	false
Null	true
Undefined	false

## isUndefined()

如果引數是 Undefined，則傳回 true。受 SQL 版本 2016-03-23 和更新版本支援。

範例：

```
isUndefined(5) = false。
```

```
isUndefined(floor([1,2,3])) = true。
```

引數類型	結果
Int	false
Decimal	false
Boolean	false
String	false
Array	false
Object	false
Null	false
Undefined	true

## length(String)

傳回所提供的字串內的字元數。標準轉換規則會套用至非 String 的引數。受 SQL 版本 2016-03-23 和更新版本支援。

範例：

```
length("hi") = 2
```

```
length(false) = 5
```

## ln(Decimal)

傳回引數的自然對數。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例： $\ln(e) = 1$ 。

引數類型	結果
Int	Decimal (使用雙精度)，引數的自然對數。
Decimal	Decimal (使用雙精度)，引數的自然對數。
Boolean	Undefined .
String	Decimal (使用雙精度)，引數的自然對數。 如果字串無法轉換為 Decimal，則結果為 Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

## log(Decimal)

傳回引數以 10 為底的對數。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例： $\log(100) = 2.0$ 。

引數類型	結果
Int	Decimal (使用雙精度)，引數以 10 為底的對數。

引數類型	結果
Decimal	Decimal (使用雙精度), 引數以 10 為底的對數。
Boolean	Undefined .
String	Decimal (使用雙精度), 引數以 10 為底的對數。如果 String 無法轉換為 Decimal, 則結果為 Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

## lower(String)

傳回小寫版本的特定 String。使用標準轉換規則將非字串引數轉換為字串。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
lower("HELLO") = 「hello」。
```

```
lower(["HELLO"]) = ["hello"]。
```

## lpad(String, Int)

傳回 String 引數，左側填入第二個引數所指定的空格數。Int 引數必須介於 0 到 1000 之間。如果提供的值超出此有效範圍，則引數將設為最接近的有效值 (0 或 1000)。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
lpad("hello", 2) = "  hello".
```

```
lpad(1, 3) = "  1"
```

引數類型 1	引數類型 2	結果
String	Int	String，所給的 String 左側填給的 Int 的數量的空格。
String	Decimal	Decimal 引數將無條件捨去至最接近的 Int，而 String 會在左側填充數字。
String	String	第二個引數會轉換為 Decimal 並捨去至最接近的 Int，而 String 會填入指定的空格數。如果第二個引數為 Int，則結果為 Undefined。
其他值	Int/Decimal/String	第一個值會使用標準轉換轉為 String，然後 LPAD 函數會套用至該 String。如果無法轉換，則結果為 Undefined。
任何值	其他值	Undefined。

## ltrim(String)

移除所給的 String 前方所有空格 (tab 與空格)。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
Ltrim(" h i ") = "hi"。
```

引數類型	結果
Int	移除 Int 前方所有空格的 String 顯示方式。
Decimal	移除 Decimal 前方所有空格的 String 顯示方式。
Boolean	移除布林值 (「true」或「false」) 前方所有空格的 String 顯示方式。

引數類型	結果
String	移除前方所有空格的引數。
陣列	String (使用標準轉換規則) 的 Array 顯示方式，且移除所有前置空格。
物件	Object (使用標準轉換規則) 的 String 顯示方式，且移除所有前置空格。
Null	Undefined .
未定義	Undefined .

### machinelearning\_predict(modelId, roleArn, record)

使用該machinelearning\_predict函數可根據 Amazon SageMaker 模型使用 MQTT 訊息中的資料進行預測。受 SQL 版本 2015-10-08 和更新版本支援。machinelearning\_predict 函數的引數如下：

#### modelId

要對其執行預測的模型 ID。必須啟用該模型的即時端點。

#### roleArn

IAM 角色具有擁有 machinelearning:Predict 和 machinelearning:GetMLModel 許可的政策，且允許存取要對其執行預測的模型。

#### record

要傳入 SageMaker 預測 API 的資料。此應顯示為單層 JSON 物件。若該記錄為多層級 JSON 物件，則記錄會透過值的序列化來扁平化。例如，以下 JSON：

```
{ "key1": {"innerKey1": "value1"}, "key2": 0}
```

會變成：

```
{ "key1": "{\"innerKey1\": \"value1\"}", "key2": 0}
```



該函數會傳回具有以下欄位的 JSON 物件：

#### predictedLabel

根據模型的輸入分類。

#### 詳細資訊

包含下列屬性：

#### PredictiveModelType

模型類型。有效值為 REGRESSION、BINARY、MULTICLASS。

#### 演算法

用 SageMaker 來進行預測的演算法。該值必須為 SGD。

#### predictedScores

包含對應至每個標籤的原始分類分數。

#### predictedValue

由預測的值 SageMaker。

### mod(Decimal, Decimal)

傳回第一個引數除以第二的引數的餘數。等同於 [remainder\(Decimal, Decimal\)](#)。也可以使用「%」當做同樣的模除功能的 infix 運算子。受 SQL 版本 2015-10-08 和更新版本支援。

範例：mod(8, 3) = 2。

左運算元	右運算元	輸出
Int	Int	Int，第一個引數模除第二個引數
Int/Decimal	Int/Decimal	Decimal，第一個引數模除第二個引數
String/Int/Decimal	String/Int/Decimal	如果所有的字串都轉換為小數，第一個引數以第二個引數為模。Undefined。

左運算元	右運算元	輸出
其他值	其他值	Undefined .

## 楠弗爾 (,) AnyValue AnyValue

若第一個引數為有效的 Decimal，即傳回。否則便傳回第二個引數。受 SQL 版本 2015-10-08 和更新版本支援。

範例：Nanvl(8, 3) = 8。

引數類型 1	引數類型 2	輸出
未定義	任何值	第二個參數。
Null	任何值	第二個參數。
Decimal (NaN)	任何值	第二個參數。
Decimal (非 NaN)	任何值	第一個參數。
其他值	任何值	第一個參數。

## newuuid()

傳回隨機的 16 位元組 UUID。受 SQL 版本 2015-10-08 和更新版本支援。

範例：newuuid() = 123a4567-b89c-12d3-e456-789012345000

## numbytes(String)

傳回所給字串 UTF-8 編碼中的位元組數。標準轉換規則會套用至非 String 的引數。受 SQL 版本 2016-03-23 和更新版本支援。

範例：

numbytes("hi") = 2

numbytes("€") = 3

## parse\_time(String, Long[, String])

使用 `parse_time` 函數來設定時間戳記格式，變成人類易懂的日期/時間格式。受 SQL 版本 2016-03-23 和更新版本支援。若要將時間戳記字串轉換為毫秒，請參閱 [time\\_to\\_epoch\(String, String\)](#)。

`parse_time` 函數預期下列引數：

pattern

(字串) 遵循 [Joda-Time 格式](#) 的日期/時間模式。

timestamp

(Long) 自 Unix epoch 起以毫秒單位設定格式的時間。請參閱函數 [timestamp\(\)](#)。

timezone

(字串) 設定好格式的日期/時間的時區。預設值為「UTC」。該函數支援 [Joda-Time 時區](#)。此為選用引數。

範例：

在此訊息發佈至主題「A/B」時，承載 `{"ts": "1970.01.01 AD at 21:46:40 CST"}` 會傳送至 S3 儲存貯體：

```
{
  "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:role/ROLE_NAME",
          "bucketName": "BUCKET_NAME",
          "key": "KEY_NAME"
        }
      }
    ]
  },
}
```

```

    "ruleName": "RULE_NAME"
  }
}

```

在此訊息發佈至主題「A/B」時，與 {"ts": "2017.06.09 AD at 17:19:46 UTC"} 相似，但包含目前日期/時間的承載會傳送至 S3 儲存貯體：

```

{
  "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:role/ROLE_NAME",
          "bucketName": "BUCKET_NAME",
          "key": "KEY_NAME"
        }
      }
    ],
    "ruleName": "RULE_NAME"
  }
}

```

parse\_time() 也可以用作替代範本。例如，當此訊息發佈至主題「A/B」時，該承載會傳送至金鑰 = 「2017」的 S3 儲存貯體：

```

{
  "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:role/ROLE_NAME",
        "bucketName": "BUCKET_NAME",
        "key": "${parse_time('yyyy', timestamp(), 'UTC')}}"
      }
    ]
  }
}

```

```

    }],
    "ruleName": "RULE_NAME"
  }
}

```

## power(Decimal, Decimal)

傳回第一個引數次方的第二個引數值。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。受 SQL 版本 2015-10-08 和更新版本支援。

範例：power(2, 5) = 32.0。

引數類型 1	引數類型 2	輸出
Int/Decimal	Int/Decimal	Decimal (使用雙精度)，第一個第二個引數值。
Int/Decimal/String	Int/Decimal/String	Decimal (使用雙精度)，第一個第二個引數值。任何轉換為小數。如果任何 String 無法轉換為 Dec，結果為 Undefined。
其他值	其他值	Undefined。

## principal()

根據發佈觸發訊息的方式，傳回裝置用於身分驗證的委託人。下表說明為各發佈方法和通訊協定傳回的委託人。

訊息發佈方式	通訊協定	憑證類型
MQTT 用戶端	MQTT	X.509 裝置憑證
AWS IoT 控制台 MQTT 客戶端	MQTT	IAM 使用者或角色
AWS CLI	HTTP	IAM 使用者或角色
AWS IoT 裝置 SDK	MQTT	X.509 裝置憑證

訊息發佈方式	通訊協定	憑證類型
AWS IoT 裝置 SDK	MQTT 超過 WebSocket	IAM 使用者或角色

以下範例顯示 `principal()` 可以傳回哪些不同類型的值：

- X.509 憑證指  
紋：ba67293af50bf2506f5f93469686da660c7c844e7b3950bfb16813e0d31e9373
- IAM 角色 ID 和工作階段名稱：ABCD1EFG3HIJK2LMNOP5:my-session-name
- 傳回使用者 ID：ABCD1EFG3HIJK2LMNOP5

## rand()

傳回虛擬亂數，平均分散在 0.0 和 1.0。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
rand() = 0.8231909191640703
```

## regexp\_matches(String, String)

如果字串 (第一個引數) 包含規則表達式的相符項目 (第二個引數)，則傳回 true。如果您在規則運算式中使用，請搭配使用 `()`。

範例：

```
regexp_matches("aaaa", "a{2,}") = true。
```

```
regexp_matches("aaaa", "b") = false。
```

```
regexp_matches("aaa", "(aaa|bbb)") = true。
```

```
regexp_matches("bbb", "(aaa|bbb)") = true。
```

```
regexp_matches("ccc", "(aaa|bbb)") = false。
```

第一個引數：

引數類型	結果
Int	Int 的 String 顯示方式。

引數類型	結果
Decimal	Decimal 的 String 顯示方式。
Boolean	布林值 (「true」或「false」) 的 String 顯示方式。
String	String。
陣列	Array (使用標準轉換規則) 的 String 顯示方式。
物件	Object (使用標準轉換規則) 的 String 顯示方式。
Null	Undefined .
未定義	Undefined .

第二個引數：

必須為有效的 regex 表達式。非字串類型都會使用標準轉換規則轉換為 String。根據類型，結果字串可能不是有效的正規運算式。如果 (轉換的) 引數並非有效的 regex 值，結果會為 Undefined。

`regexp_replace(String, String, String)`

以第三個引數取代所有在第一個引數中出現的第二個參數 (一般表達式)。請以「\$」參考擷取群組。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
regexp_replace("abcd", "bc", "x") = "axd"。
```

```
regexp_replace("abcd", "b(.*)d", "$1") = "ac"。
```

第一個引數：

引數類型	結果
Int	Int 的 String 顯示方式。

引數類型	結果
Decimal	Decimal 的 String 顯示方式。
Boolean	布林值 (「true」或「false」) 的 String 顯示方式。
String	來源值。
陣列	Array (使用標準轉換規則) 的 String 顯示方式。
物件	Object (使用標準轉換規則) 的 String 顯示方式。
Null	Undefined .
未定義	Undefined .

第二個引數：

必須為有效的 regex 表達式。非字串類型都會使用標準轉換規則轉換為 String。根據類型，結果字串可能不是有效的正規運算式。如果 (轉換的) 引數並非有效的 regex 表達式，則結果為 Undefined。

第三個引數：

必須為有效的 regex 替換字串。(可以參考擷取群組)。非字串類型都會使用標準轉換規則轉換為 String。如果 (轉換的) 引數並非有效的 regex 替換字串，則結果為 Undefined。

`regexp_substr(String, String)`

在第一個參數中尋找第一個符合第二個參數 (regex) 的值。請以「\$」參考擷取群組。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
regexp_substr("hihihello", "hi") = "hi"
```

```
regexp_substr("hihihello", "(hi)*") = "hihi"
```



## 第一個引數：

引數類型	結果
Int	Int 的 String 顯示方式。
Decimal	Decimal 的 String 顯示方式。
Boolean	布林值 (「true」或「false」) 的 String 顯示方式。
String	String 引數。
陣列	Array (使用標準轉換規則) 的 String 顯示方式。
物件	Object (使用標準轉換規則) 的 String 顯示方式。
Null	Undefined .
未定義	Undefined .

## 第二個引數：

必須為有效的 regex 表達式。非字串類型都會使用標準轉換規則轉換為 String。根據類型，結果字串可能不是有效的正規運算式。如果 (轉換的) 引數並非有效的 regex 表達式，則結果為 Undefined。

## remainder(Decimal, Decimal)

傳回第一個引數除以第二的引數的餘數。等同於 [mod\(Decimal, Decimal\)](#)。也可以使用「%」當做同樣的模除功能的 infix 運算子。受 SQL 版本 2015-10-08 和更新版本支援。

範例：remainder(8, 3) = 2。

左運算元	右運算元	輸出
Int	Int	Int，第一個引數模除第二個引數
Int/Decimal	Int/Decimal	Decimal，第一個引數模除第二個引數

左運算元	右運算元	輸出
String/Int/Decimal	String/Int/Decimal	如果所有的字串都轉換為小數，第一個引數以第二個引數為模。 Undefined 。
其他值	其他值	Undefined 。

## replace(String, String, String)

以第三個引數取代所有在第一個引數中出現的第二個引數。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
replace("abcd", "bc", "x") = "axd".
```

```
replace("abcdabcd", "b", "x") = "axcdaxcd".
```

所有引數

引數類型	結果
Int	Int 的 String 顯示方式。
Decimal	Decimal 的 String 顯示方式。
Boolean	布林值 (「true」或「false」) 的 String 顯示方式。
String	來源值。
陣列	Array (使用標準轉換規則) 的 String 顯示方式。
物件	Object (使用標準轉換規則) 的 String 顯示方式。
Null	Undefined 。
未定義	Undefined 。

## rpad(String, Int)

傳回字串引數，右側填入第二個引數所指定的空格數。Int 引數必須介於 0 到 1000 之間。如果提供的值超出此有效範圍，則引數將設為最接近的有效值 (0 或 1000)。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
rpad("hello", 2) = "hello  ".
```

```
rpad(1, 3) = "1   ".
```

引數類型 1	引數類型 2	結果
String	Int	String 在右側填入，且空格數量等同所給的 Int。
String	Decimal	Decimal 引數會無條件捨去至最接近的 Int，並且該字串將以提供的 Int 相同的空格數填入右側。
String	String	第二個引數將轉換為 Decimal，並無條件捨去至最接近的 Int。String 在右側填入，且空格數量等同 Int 的值。
其他值	Int/Decimal/String	第一個值會使用標準轉換來轉換為 String，而 rpad 函數將套用到該 String 上。如果其無法轉換，則結果為 Undefined。
任何值	其他值	Undefined。

## round(Decimal)

將指定的 Decimal 無條件進位至最接近的 Int。如果 Decimal 與兩個 Int 值 (例如, 0.5) 等距, 則 Decimal 會無條件進位。受 SQL 版本 2015-10-08 和更新版本支援。

範例 : Round(1.2) = 1。

Round(1.5) = 2。

Round(1.7) = 2。

Round(-1.1) = -1。

Round(-1.5) = -2。

引數類型	結果
Int	引數。
Decimal	Decimal 是要向下捨入到最接近的 Int。
String	Decimal 是要向下捨入到最接近的 Int。 如果字串無法轉換為 Decimal, 則結果為 Undefined。
其他值	Undefined。

## rtrim(String)

移除所給的 String 後方所有空格 (tab 與空格)。受 SQL 版本 2015-10-08 和更新版本支援。

範例 :

```
rtrim(" h i ") = "hi"
```

引數類型	結果
Int	Int 的 String 顯示方式。
Decimal	Decimal 的 String 顯示方式。

引數類型	結果
Boolean	布林值 (「true」或「false」) 的 String 顯示方式。
陣列	Array (使用標準轉換規則) 的 String 顯示方式。
物件	Object (使用標準轉換規則) 的 String 顯示方式。
Null	Undefined .
未定義	Undefined

## sign(Decimal)

傳回所給數字的符號。當引數的符號為正值時，傳回 1。當引數的符號為負值時，傳回 -1。如果引數為 0，傳回 0。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

$\text{sign}(-7) = -1$ 。

$\text{sign}(0) = 0$ 。

$\text{sign}(13) = 1$ 。

引數類型	結果
Int	Int , Int 值的符號。
Decimal	Int , Decimal 值的符號。
String	Int , Decimal 值的符號。此字串會轉換為 Decimal 值，並傳回 Decimal 值的符號。如果 String 無法轉換為 Decimal，則結果為 Undefined。受 SQL 版本 2015-10-08 和更新版本支援。

引數類型	結果
其他值	Undefined .

## sin(Decimal)

以弧度傳回數字的正弦值。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例：sin(0) = 0.0

引數類型	結果
Int	Decimal (使用雙精度)，引數的正弦值。
Decimal	Decimal (使用雙精度)，引數的正弦值。
Boolean	Undefined .
String	Decimal (使用雙精度)，引數的正弦值。 如果字串無法轉換為 Decimal，則結果為 Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
Undefined	Undefined .

## sinh(Decimal)

以弧度傳回數字的雙曲正弦值。Decimal 值在套用函數前會四捨五入至雙精度。結果為雙精度的 Decimal 值。受 SQL 版本 2015-10-08 和更新版本支援。

範例：sinh(2.3) = 4.936961805545957

引數類型	結果
Int	Decimal (使用雙精度), 引數的雙曲正弦值。
Decimal	Decimal (使用雙精度), 引數的雙曲正弦值。
Boolean	Undefined .
String	Decimal (使用雙精度), 引數的雙曲正弦值。如果字串無法轉換為 Decimal, 則結果為 Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

## sourceip()

擷取裝置或與其連線之路由器的 IP 地址。如果您的裝置直接連線至網際網路，則此函數會傳回裝置的來源 IP 地址。如果您的裝置是連線到連線網際網路的路由器，則此函數會傳回路由器的來源 IP 地址。SQL 2016-03-23 版可支援。sourceip() 不會採用任何參數。

### Important

裝置的公有來源 IP 地址通常是最後一個網路位址轉譯 (NAT) 閘道的 IP 地址，例如，您的網際網路服務供應商的路由器或有線數據機。

範例：

```
sourceip()="192.158.1.38"
```

```
sourceip()="1.102.103.104"
```

```
sourceip()="2001:db8:ff00::12ab:34cd"
```

SQL 範例：

```
SELECT *, sourceip() as deviceIp FROM 'some/topic'
```

如何在 AWS IoT Core 規則動作中使用 sourceip () 函數的範例：

### 範例 1

下列範例顯示如何在 [DynamoDB 動作](#) 中呼叫 () 函數作為 [替換範本](#)。

```
{
  "topicRulePayload": {
    "sql": "SELECT * AS message FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
        "dynamoDB": {
          "tableName": "my_ddb_table",
          "hashKeyField": "key",
          "hashKeyValue": "${sourceip()}",
          "rangeKeyField": "timestamp",
          "rangeKeyValue": "${timestamp()}",
          "roleArn": "arn:aws:iam::123456789012:role/aws_iam_dynamoDB"
        }
      }
    ]
  }
}
```

### 範例 2

下列範例顯示如何使用 [替換範本](#) 新增 sourceip() 函數作為 MQTT 使用者屬性。

```
{
  "topicRulePayload": {
    "sql": "SELECT * FROM 'some/topic'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
      {
```



```
"republish": {
  "topic": "${topic()}/republish",
  "roleArn": "arn:aws:iam::123456789012:role/aws_iot_republish",
  "headers": {
    "payloadFormatIndicator": "UTF8_DATA",
    "contentType": "rule/contentType",
    "correlationData": "cnVsZSBjb3JyZWxhdGlvbiBkYXRh",
    "userProperties": [
      {
        "key": "ruleKey1",
        "value": "ruleValue1"
      },
      {
        "key": "sourceip",
        "value": "${sourceip()}"
      }
    ]
  }
}
```

您可以從訊息代理程式和[基本內嵌路徑傳遞至 AWS IoT Core 規則的訊息擷取](#)來源 IP 位址。您也可以擷取 IPv4 和 IPv6 訊息的來源 IP。來源 IP 顯示如下：

IPv6 : yyyy:yyyy:yyyy::yyyy:yyyy

IPv4 : xxx.xxx.xxx.xxx

#### Note

原始來源 IP 不會透過[重新發布動作](#)傳遞。

## substring(String, Int[, Int])

預期 String 後有一或兩個 Int 值。若為 String 和單個 Int 引數，此函數會從所給的 String 索引 (從 0 開始，包含 0) 到 Int 的結尾傳回所給的 String 的子字串。若為 String 和兩個 Int 引數，該函數會傳回從第一個 String 索引引數 (從 0 開始，包含 0) 到第二個 Int 索引引數 (從 0 開始，不含 0) 所提供的所給 Int 的子字串。少於零的索引將設為零。比 String 長度還長的索引會設

為 String 長度。至於第三個引數版本，如果第一個索引大於 (或等於) 第二個索引，則結果為空的 String。

如果提供的引數不是 (##、##) 或 (##、##、##)，則標準轉換會套用至該引數，以嘗試將那些引數轉換為正確類型。如果類型無法轉換，函數的結果即為 Undefined。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
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) = "true"。
```

```
substring(false, -2.411E247) = "false"。
```

```
substring("012345", 1, 3) = "12"。
```

```
substring("012345", -50, 50) = "012345"。
```

```
substring("012345", 3, 1) = ""。
```

### sql\_version()

傳回此規則中指定的 SQL 版本。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
sql_version() = "2016-03-23"
```

### sqrt(Decimal)

傳回數字的平方根。Decimal 引數在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例：`sqrt(9) = 3.0`。

引數類型	結果
Int	引數的平方根。
Decimal	引數的平方根。
Boolean	Undefined .
String	引數的平方根。如果字串無法轉換為 Decimal，則結果為 Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

### startswith(String, String)

傳回 Boolean，無論第一個字串引數的開頭是否為第二個引數。如果引數為 Null 或 Undefined，則結果為 Undefined。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
startswith("ranger", "ran") = true
```

引數類型 1	引數類型 2	結果
String	String	無論第一個字串的開頭是否為第
其他值	其他值	所有引數都將使用標準轉換規則字串。如果第一個字串的開頭是字串，則傳回 true。如果引數為 Undefined，則結果為 Undef

## tan(Decimal)

以弧度傳回數字的正切值。Decimal 值在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例： $\tan(3) = -0.1425465430742778$

引數類型	結果
Int	Decimal (使用雙精度), 引數的正切值。
Decimal	Decimal (使用雙精度), 引數的正切值。
Boolean	Undefined .
String	Decimal (使用雙精度), 引數的正切值。 如果字串無法轉換為 Decimal, 則結果為 Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

## tanh(Decimal)

以弧度傳回數字的雙曲正切值。Decimal 值在套用函數前會四捨五入至雙精度。受 SQL 版本 2015-10-08 和更新版本支援。

範例： $\tanh(2.3) = 0.9800963962661914$

引數類型	結果
Int	Decimal (使用雙精度), 引數的雙曲正切值。

引數類型	結果
Decimal	Decimal (使用雙精度), 引數的雙曲正切值。
Boolean	Undefined .
String	Decimal (使用雙精度), 引數的雙曲正切值。如果字串無法轉換為 Decimal, 則結果為 Undefined 。
Array	Undefined .
物件	Undefined .
Null	Undefined .
未定義	Undefined .

## time\_to\_epoch(String, String)

使用 `time_to_epoch` 函數將時間戳字串轉換為 Unix epoch 時間中的毫秒數。受 SQL 版本 2016-03-23 和更新版本支援。若要將毫秒轉換為格式化的時間戳記字串, 請參閱 [parse\\_time\(String, Long\[, String\]\)](#)。

`time_to_epoch` 函數預期下列引數 :

timestamp

(字串) 自 Unix epoch 起要轉換為毫秒的時間戳記字串。如果時間戳記字串未指定時區, 函數會使用 UTC 時區。

pattern

(字串) 遵循 [JDK11 時間格式](#) 的日期/時間模式。

範例 :

```
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()

根據 AWS IoT 規則引擎觀察，從 1970 年 1 月 1 日星期四 00:00:00 開始，傳回目前的時間戳記 (以毫秒為單位)。受 SQL 版本 2015-10-08 和更新版本支援。

範例：`timestamp() = 1481825251155`

## topic(Decimal)

傳回觸發該規則的訊息要傳送至的主題。如果未指定參數，則傳回整個主題。Decimal 參數用於指定特定主題區段，並會使用 1 指定第一個區段。對於 `topic foo/bar/baz`，`topic(1)` 會傳回 `foo`，`topic(2)` 會傳回 `bar`，依此類推。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
topic() = "things/myThings/thingOne"
```

```
topic(1) = "things"
```

使用[基本擷取](#)時，`topic()` 函數無法使用主題的初始字首 (`$aws/rules/rule-name`)。例如，假定主題為：

```
$aws/rules/BuildingManager/Buildings/Building5/Floor2/Room201/Lights
```

```
topic() = "Buildings/Building5/Floor2/Room201/Lights"
```

```
topic(3) = "Floor2"
```

## traceid()

傳回 MQTT 的追蹤 ID (UUID)，如果訊息不是透過 MQTT 傳送，則為 `Undefined`。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
traceid() = "12345678-1234-1234-1234-123456789012"
```

## transform(String, Object, Array)

傳回物件的陣列，其中包含 Array 參數上 Object 參數的所指定轉換結果。

受 SQL 版本 2016-03-23 和更新版本支援。

### 字串

要使用的轉換模式。請參閱下表以取得支援的轉換模式，以及了解它們如何從 Object 和 Array 參數建立 Result。

### 物件

包含要套用至每個 Array 元素之屬性的物件。

### 陣列

Object 屬性套用至其中的物件陣列。

此陣列中的每個物件都對應於函數回應中的物件。函數回應中的每個物件都包含存在於原始物件的屬性，以及 Object 所提供的屬性，這是由 String 中指定的轉換模式所決定。

String 參數	Object 參數	Array 參數	結果
enrichArray	物件	物件的陣列	物件的陣列，其中每個物件都包含來自 Array 參數的元素屬性，以及 Object 參數的屬性。
任何其他值	任何值	任何值	未定義

#### Note

此函數傳回的陣列限制為 128 KiB。

### 轉換函數範例 1

此範例顯示 transform() 函數如何從一個資料物件和一個陣列產生單一物件陣列。

在此範例中，下列訊息會發佈至 MQTT 主題 A/B。

```
{
  "attributes": {
    "data1": 1,
    "data2": 2
  },
  "values": [
    {
      "a": 3
    },
    {
      "b": 4
    },
    {
      "c": 5
    }
  ]
}
```

主題規則動作的這個 SQL 陳述式會使用 transform() 函數與 enrichArray 的 String 值搭配。在此範例中，Object 是來自訊息承載的 attributes 屬性，而 Array 是 values 陣列，其中包含三個物件。

```
select value transform("enrichArray", attributes, values) from 'A/B'
```

在收到訊息承載時，SQL 陳述式會評估為下列回應。

```
[
  {
    "a": 3,
    "data1": 1,
    "data2": 2
  },
  {
    "b": 4,
    "data1": 1,
    "data2": 2
  },
  {
    "c": 5,
    "data1": 1,

```



```
    "data2": 2
  }
]
```

## 轉換函數範例 2

此範例顯示 transform() 函數如何使用文字值，以包含並重新命名來自訊息承載的個別屬性。

在此範例中，下列訊息會發佈至 MQTT 主題 A/B。這是用於 [the section called “轉換函數範例 1”](#) 的同一則訊息。

```
{
  "attributes": {
    "data1": 1,
    "data2": 2
  },
  "values": [
    {
      "a": 3
    },
    {
      "b": 4
    },
    {
      "c": 5
    }
  ]
}
```

主題規則動作的這個 SQL 陳述式會使用 transform() 函數與 enrichArray 的 String 值搭配。transform() 函數中的 Object 在訊息承載中有一個名為 key 且值為 attributes.data1 的單一屬性，而 Array 是 values 陣列，其中包含上述範例中使用的三個物件。

```
select value transform("enrichArray", {"key": attributes.data1}, values) from 'A/B'
```

在收到訊息承載時，此 SQL 陳述式會評估為下列回應。請注意，data1 屬性在回應中名為 key。

```
[
  {
    "a": 3,
    "key": 1
  },
]
```

```
{
  "b": 4,
  "key": 1
},
{
  "c": 5,
  "key": 1
}
]
```

### 轉換函數範例 3

此範例顯示 transform() 函數如何用於巢狀 SELECT 子句中，以選取多個屬性並建立新的物件，以供後續處理。

在此範例中，下列訊息會發佈至 MQTT 主題 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
      }
    ]
  }
}
```

此轉換函數的 Object 是 SELECT 陳述式傳回的物件，其中包含訊息 data2 物件的 a 和 b 元素。Array 參數由來自原始訊息中 data2.c 陣列的兩個物件組成。

```
select value transform('enrichArray', (select a, b from data2), (select value c from data2)) from 'A/B'
```

在收到上述訊息時，SQL 陳述式會評估為下列回應。

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

此回應中傳回的陣列可與支援 `batchMode` 的主題規則動作搭配使用。

## trim(String)

移除所給的 String 前方和後方所有空格。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
Trim(" hi ") = "hi"
```

引數類型	結果
Int	移除 Int 前方和後方所有空格的 String 顯示方式。

引數類型	結果
Decimal	移除 Decimal 前方和後方所有空格的 String 顯示方式。
Boolean	移除 Boolean (「true」或「false」) 前方和後方所有空格的 String 顯示方式。
String	移除前方和後方所有空格的 String。
陣列	Array (使用標準轉換規則) 的 String 顯示方式。
物件	Object (使用標準轉換規則) 的 String 顯示方式。
Null	Undefined .
未定義	Undefined .

## trunc(Decimal, Int)

將第一個引數截為第二的引數所指定的 Decimal 位數。若第二個引數少於零，則它會設為零。若第二個引數大於 34，則它會設為 34。結果的結尾去掉了零。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

`trunc(2.3, 0) = 2。`

`trunc(2.3123, 2) = 2.31。`

`trunc(2.888, 2) = 2.88。`

`trunc(2.00, 5) = 2。`

引數類型 1	引數類型 2	結果
Int	Int	來源值。

引數類型 1	引數類型 2	結果
Int/Decimal	Int/Decimal	第一個引數會截短為第二個引數度。第二個引數如果不是 Int，捨去至最接近的 Int。
Int/Decimal/String	Int/Decimal	第一個引數會截短為第二個引數度。第二個引數如果不是 Int，件捨去至最接近的 Int。String 值轉換為 Decimal 值。如果字串轉換失敗則 Undefined。
其他值		Undefined。

## upper(String)

傳回大寫版本的特定 String。非 String 引數都會使用標準轉換規則轉換為 String。受 SQL 版本 2015-10-08 和更新版本支援。

範例：

```
upper("hello") = "HELLO"
```

```
upper(["hello"]) = ["HELLO"]
```

## 文字

可以直接在規則 SQL 的 SELECT 和 WHERE 子句中指定文字物件，很適合用於傳遞資訊。

### Note

只有在使用 2016-03-23 版本或更高版本的 SQL 時才能使用文字。

會使用 JSON 物件語法 (金鑰/值對，以逗號分隔，索引鍵為字串而值為 JSON 值，以大括號 {} 括起)。例如：

```
發佈在主題 topic/subtopic 的傳入承載：{"lat_long": [47.606, -122.332]}
```

```
SQL 陳述式 : SELECT {'latitude': get(lat_long, 0), 'longitude': get(lat_long, 1)} as lat_long FROM 'topic/subtopic'
```

產生的傳出承載為 : {"lat\_long":{"latitude":47.606,"longitude":-122.332}}。

您也可以直接在規則 SQL 的 SELECT 和 WHERE 子句中指定陣列，即可將資訊分組。會使用 JSON 語法 (將逗號分隔項目用方括號 [] 括起，以建立陣列文字)。例如：

發佈在主題 topic/subtopic 的傳入承載 : {"lat": 47.696, "long": -122.332}

```
SQL 陳述式 : SELECT [lat,long] as lat_long FROM 'topic/subtopic'
```

產生的輸出承載為 : {"lat\_long": [47.606, -122.332]}。

## 案例陳述式

可用於分支執行的案例陳述式，例如切換陳述式。

語法：

```
CASE v WHEN t[1] THEN r[1]
      WHEN t[2] THEN r[2] ...
      WHEN t[n] THEN r[n]
      ELSE r[e] END
```

表達式 *v* 會受到評估並對每個 WHEN 子句的 *t[i]* 值配對是否相等。如果找到匹配的結果，相對應的 *r[i]* 表達式會成為該 CASE 陳述式的結果。此 WHEN 子句會按順序進行評估，以便如果有多個符合子句，則第一個符合子句的結果會成為 CASE 陳述式的結果。如果沒有相符項目，ELSE 子句的 *r[e]* 即為結果。如果沒有相符項目，也沒有 ELSE 子句，則結果即為 Undefined。

CASE 陳述式會要求至少一個 WHEN 子句。ELSE 子句是選用的。

例如：

發佈在主題 topic/subtopic 的傳入承載：

```
{
  "color":"yellow"
}
```

## SQL 陳述式：

```
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'
```

## 產生的輸出承載為：

```
{
  "instructions":"caution"
}
```

### Note

如果 `v` 是 `Undefined`，則案例陳述式的結果為 `Undefined`。

## JSON Extensions

您可以使用下列 ANSI SQL 的延伸模組，協助使用巢狀 JSON 物件。

### "." 運算子

此運算子會存取嵌入式 JSON 物件中的成員，而函式與 ANSI SQL 和 JavaScript。例如：

```
SELECT foo.bar AS bar.baz FROM 'topic/subtopic'
```

從傳送至 `topic/subtopic` 主題的下列訊息承載中選取 `foo` 物件中 `bar` 屬性的值。

```
{
  "foo": {
    "bar": "RED",
    "bar1": "GREEN",
    "bar2": "BLUE"
  }
}
```

如果 JSON 屬性名稱包含連字元或數字字元，「點」標記法將無法運作。您必須改用 [get 函數](#) 來擷取屬性的值。

在此範例中，下列訊息會傳送至 `iot/rules` 主題。

```
{
  "mydata": {
    "item2": {
      "0": {
        "my-key": "myValue"
      }
    }
  }
}
```

正常情況下，將識別 `my-key` 的值，如在此查詢中一般。

```
SELECT * from iot/rules WHERE mydata.item2.0.my-key= "myValue"
```

不過，由於屬性名稱 `my-key` 包含連字號，且 `item2` 包含數字字元，所以 [get 函數](#) 必須如下列查詢所示一般使用。

```
SELECT * from 'iot/rules' WHERE get(get(get(mydata,"item2"),"0"),"my-key") = "myValue"
```

### \* 運算子

運作方式與 ANSI SQL 的 \* 萬用字元相同。此僅會使用在 SELECT 子句，且會建立包含訊息資料的新 JSON 物件。如果訊息承載並非 JSON 格式，\* 會以原始位元組的形式傳回整個訊息承載。例如：

```
SELECT * FROM 'topic/subtopic'
```

將函數套用在屬性值上

以下為可能為由裝置發佈的範例 JSON 承載：

```
{
  "deviceid" : "iot123",
  "temp" : 54.98,
  "humidity" : 32.43,
  "coords" : {
```



```
    "latitude" : 47.615694,  
    "longitude" : -122.3359976  
  }  
}
```

以下範例會將函數套用在 JSON 承載內的屬性值上：

```
SELECT temp, md5(deviceid) AS hashed_id FROM topic/#
```

此次查詢的結果為下列 JSON 物件：

```
{  
  "temp": 54.98,  
  "hashed_id": "e37f81fb397e595c4aeb5645b8cbbbd1"  
}
```

## 替代範本

您可以使用替代範本來擴充規則觸發並 AWS IoT 執行動作時傳回的 JSON 資料。替代範本的語法是 `{運算式}`，其 AWS IoT 中運算式可以是 SELECT 子句、WHERE 子句和支援的任何運算式 [AWS IoT 規則動作](#)。您可以將此表達式插入規則的動作欄位中，以便動態設定動作。實際上，此功能會取代動作中的資訊片段。這包含了在原始訊息中呈現的函數、運算子和資訊。

### Important

因為替換範本中的運算式與「SELECT ...」陳述式是分開計算的，所以不能參考使用 AS 子句建立的別名。您只能參考原始承載、[函數](#)和[運算子](#)中呈現的資訊。

如需支援的表達式的詳細資訊，請參閱 [AWS IoT SQL 參考資料](#)。

下列規則動作支援替代範本。每個動作都支援可以取代的不同欄位。

- [Apache Kafka](#)
- [CloudWatch 警報](#)
- [CloudWatch 日誌](#)
- [CloudWatch 度量](#)
- [DynamoDB](#)

- [DynamoDBv2](#)
- [Elasticsearch](#)
- [HTTP](#)
- [IoT Analytics](#)
- [AWS IoT Events](#)
- [AWS IoT SiteWise](#)
- [Kinesis Data Streams](#)
- [Firehose](#)
- [Lambda](#)
- [位置](#)
- [OpenSearch](#)
- [Republish](#)
- [S3](#)
- [SNS](#)
- [SQS](#)
- [Step Functions](#)
- [Timestream](#)

替代範本會顯示在規則內的動作參數中：

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

如果這個規則是由下列發佈至 my/iot/topic 的 JSON 所觸發：

```
{
```

```
"deviceid": "iot123",
"temp": 54.98,
"humidity": 32.43,
"coords": {
  "latitude": 47.615694,
  "longitude": -122.3359976
}
}
```

然後，此規則會將下列 JSON 發佈至 `my/iot/topic/republish`，AWS IoT 取代自 `${topic()}/republish`：

```
{
  "deviceid": "iot123",
  "temp": 54.98,
  "humidity": 32.43,
  "coords": {
    "latitude": 47.615694,
    "longitude": -122.3359976
  },
  "timestamp": 1579637878451
}
```

## 巢狀物件查詢

您可以使用巢狀 SELECT 子句來查詢陣列和內部 JSON 物件中的屬性。受 SQL 版本 2016-03-23 和更新版本支援。

請考慮下列 MQTT 訊息：

```
{
  "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

您可以使用以下規則將值轉換為新的陣列。

```
SELECT (SELECT VALUE n FROM e) as sensors FROM 'my/topic'
```

該規則會產生以下輸出。

```
{
  "sensors": [
    "temperature",
    "light",
    "acidity"
  ]
}
```

### Example

使用相同的 MQTT 訊息，您也可以使用下列規則查詢巢狀物件中的特定值。

```
SELECT (SELECT v FROM e WHERE n = 'temperature') as temperature FROM 'my/topic'
```

該規則會產生以下輸出。

```
{
  "temperature": [
    {
      "v": 22.5
    }
  ]
}
```

### Example

您也可以使用更複雜的規則來平面化輸出。

```
SELECT get((SELECT v FROM e WHERE n = 'temperature'), 0).v as temperature FROM 'topic'
```

該規則會產生以下輸出。

```
{
  "temperature": 22.5
}
```

## 使用二進位承載

若要將訊息承載做為原始二進位資料 (而不是 JSON 物件) 處理，可以使用 \* 運算子在 SELECT 子句中參考它。

在本主題中：

- [二進位承載範例](#)
- [對 Protobuf 訊息承載進行解碼](#)

### 二進位承載範例

使用 \* 參考作為原始二進位資料的訊息承載時，您可以將資料新增至規則。如果您有空白或 JSON 承載，則產生的承載可以使用規則新增資料。下列顯示支援 SELECT 子句的範例。

- 對於二進位承載，您可以使用下方僅具有一個 \* 的 SELECT 子句。

```
SELECT * FROM 'topic/subtopic'
```

```
SELECT * FROM 'topic/subtopic' WHERE timestamp() % 12 = 0
```

- 您也可以新增資料並使用下方 SELECT 子句。

```
SELECT *, principal() as principal, timestamp() as time FROM 'topic/subtopic'
```

```
SELECT encode(*, 'base64') AS data, timestamp() AS ts FROM 'topic/subtopic'
```

- 您也可以使用這些 SELECT 子句搭配二進位承載使用。

- 下方項目是指 WHERE 子句中的 device\_type。

```
SELECT * FROM 'topic/subtopic' WHERE device_type = 'thermostat'
```

- 也支援下方項目。

```
{
  "sql": "SELECT * FROM 'topic/subtopic'",
  "actions": [
    {
      "republish": {
        "topic": "device/${device_id}"
      }
    }
  ]
}
```

```
}  
]  
}
```

下方規則動作不支援二進位承載，因此您必須將它們解碼。

- 對於某些不支援二進位承載輸入 (例如 [Lambda 動作](#)) 的規則，您必須解碼二進位承載。如果 Lambda 規則動作是 base64 編碼且在 JSON 承載中，則可以接收二進位資料。您可以將規則變更如下，以此執行此項操作。

```
SELECT encode(*, 'base64') AS data FROM 'my_topic'
```

- SQL 陳述式不支援將字串作為輸入。若要將字串輸入轉換為 JSON，您可以執行下列命令。

```
SELECT decode(encode(*, 'base64'), 'base64') AS payload FROM 'topic'
```

## 對 Protobuf 訊息承載進行解碼

[協定緩衝區 \(protobuf\)](#) 是一種開放原始碼資料格式，用於將結構化資料序列化為壓縮二進位形式。其可用於透過網路傳輸資料或將其儲存在檔案中。Protobuf 允許您以較小的數據包大小並以比其他消息傳遞格式更快的速度發送數據。AWS IoT Core 規則通過提供[解碼 \(值, 解碼方案\) SQL 函數](#)來支持 [protobuf](#)，該函數允許您將原型編碼的消息有效載荷解碼為 JSON 格式並將其路由到下游服務。本節詳細介紹了在規則中配置原生解碼的 step-by-step AWS IoT Core 過程。

在本節中：

- [必要條件](#)
- [建立描述項檔案](#)
- [將描述項檔案上傳至 S3 儲存貯體](#)
- [在規則中設定 protobuf 解碼](#)
- [限制](#)
- [最佳實務](#)

### 必要條件

- [協定緩衝區 \(protobuf\)](#) 的基本知識
- 該 [.proto 檔案](#) 定義了訊息類型及相關的相依性

- 在您的系統上安裝 [Protobuf 編譯器 \(protoc\)](#)

## 建立描述項檔案

如果您已有描述項檔案，則可以略過此步驟。描述項檔案 (.desc) 是 .proto 檔案的編譯版本，屬於文字檔案，用於定義在 protobuf 序列化中使用的資料結構和訊息類型。要產生描述項檔案，您必須定義 .proto 檔案並使用 [protoc](#) 編譯器對其進行編譯。

1. 建立用於定義訊息類型的 .proto 檔案。範例 .proto 檔案看起來可能與以下內容相似：

```
syntax = "proto3";

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;
}
```

在此範例 .proto 檔案中，您使用 proto3 語法並定義訊息類型 Person。Person 訊息定義會指定三個欄位 (名稱、ID 和電子郵件)。如需有關 .proto 檔案訊息格式的詳細資訊，請參閱 [語言指南 \(proto3\)](#)。

2. 使用 [protoc](#) 編譯器編譯 .proto 檔案並產生描述項檔案。用於建立描述項 (.desc) 檔案的範例命令如下所示：

```
protoc --descriptor_set_out=<FILENAME>.desc \
  --proto_path=<PATH_TO_IMPORTS_DIRECTORY> \
  --include_imports \
  <PROTO_FILENAME>.proto
```

這個示例命令生成一個描述符文件 <FILENAME>.desc，AWS IoT Core Rules 可以用它來解碼符合中定義的數據結構的 protobuf 有效載荷。<PROTO\_FILENAME>.proto

- --descriptor\_set\_out

指定所要產生的描述項檔案 (<FILENAME>.desc) 名稱。

- --proto\_path

指定編譯檔案所參考的任何已匯入 .proto 檔案的位置。如果您有多項已匯入的 .proto 檔案位於不同位置，則可以多次指定旗標。

- `--include_imports`

指定任何已匯入的 `.proto` 檔案也應加以編譯，並納入 `<FILENAME>.desc` 描述項檔案中。

- `<PROTO_FILENAME>.proto`

指定要編譯的 `.proto` 檔案名稱。

如需 `protoc` 參考的詳細資訊，請參閱 [API 參考](#)。

## 將描述項檔案上傳至 S3 儲存貯體

建立描述元檔案後 `<FILENAME>.desc`，`<FILENAME>.desc` 請使用 AWS API、AWS 開發套件或 AWS Management Console。

## 重要考量

- 請確定您將描述元檔案上傳到您想要設定規則 AWS 帳戶 的相同 AWS 區域 位置的 Amazon S3 儲存貯體。
- 確保您授予 `FileDescriptorSet` 從 S3 讀 AWS IoT Core 取的訪問權限。如果 S3 儲存貯體已停用伺服器端的加密 (SSE)，或者 S3 儲存貯體是使用 Amazon S3 受管金鑰 (SSE-S3) 進行加密，則不需要其他政策組態。下列範例儲存貯體政策可實現此目的：

```
{
  "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 儲存貯體使用金 AWS Key Management Service 鑰 (SSE-KMS) 加密，請確保在存取 S3 儲存貯體時授 AWS IoT Core 予使用金鑰的權限。作法為您可以將下列陳述式新增至金鑰政策：



```
{
  "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"
}
```

在規則中設定 protobuf 解碼

將描述項檔案上傳到 Amazon S3 儲存貯體後，請設定一項規則以供使用 [decode\(value, decodingScheme\)](#) SQL 函數來解碼 protobuf 訊息承載格式。詳細的函數簽章和範例可參見 AWS IoT SQL 參考的 [decode\(value, decodingScheme\)](#) SQL 函數。

以下是使用 [decode\(value, decodingScheme\)](#) 函數的 SQL 表達式範例：

```
SELECT VALUE decode(*, 'proto', '<BUCKET NAME>', '<FILENAME>.desc', '<PROTO_FILENAME>',
'<PROTO_MESSAGE_TYPE>') FROM '<MY_TOPIC>'
```

在此範例表達式中：

- 您可以使用 [decode\(value, decodingScheme\)](#) SQL 函數，來解碼 \* 參考的二進位訊息承載。這可以是二進位 protobuf 編碼承載，也可以是代表 base64 編碼 protobuf 承載的 JSON 字串。
- 所提供的訊息承載會使用 PROTO\_FILENAME.proto 中定義的 Person 訊息類型進行編碼。
- 名為 BUCKET NAME 的 Amazon S3 儲存貯體含有從 PROTO\_FILENAME.proto 產生的 FILENAME.desc。

完成設定後，請針對訂閱規則 AWS IoT Core 的主題發佈訊息至。

## 限制

AWS IoT Core 規則支援原生成，但有下列限制：

- 不支援在[替代範本](#)中解碼 protobuf 訊息承載。
- 在解碼 protobuf 訊息承載時，您最多可以在單一 SQL 表達式中使用[解碼 SQL 函數](#) 2 次。
- 傳入承載大小上限為 128 KiB (1KiB = 1024 位元組)，傳出承載大小上限為 128 KiB，而儲存在 Amazon S3 儲存貯體中的 FileDescriptorSet 物件大小上限為 32 KiB。
- 不支援使用 SSE-C 加密功能對 Amazon S3 儲存貯體進行加密。

## 最佳實務

以下是一些最佳實務和疑難排解提示。

- 在 Amazon S3 儲存貯體中備份原型檔案。

理想的做法是備份 proto 檔案以防患未然。例如，如果在執行 protoc 時錯誤地修改了沒有備份的原型檔案，這可能會導致生產堆疊發生問題。有多種方法在 Amazon S3 儲存貯體中備份檔案 例如，您可以在[S3 儲存貯體中使用版本控制](#)。如需有關如何備份 Amazon S3 儲存貯體中檔案的詳細資訊，請參閱[Amazon S3 開發人員指南](#)。

- 設定 AWS IoT 記錄以檢視記錄項目。

配置日誌 AWS IoT 記錄是一個很好的做法，以便您可以在中檢查帳戶的 AWS IoT 日誌 CloudWatch。當規則的 SQL 查詢呼叫外部函數時，Ru AWS IoT Core les 會產生一個eventType記錄項目FunctionExecution，其中包含可協助您疑難排解失敗問題的 reason 欄位。可能的錯誤包括找不到 Amazon S3 物件，或是 protobuf 檔案描述項無效。如需進一步了解如何設定 AWS IoT 記錄及查看日誌項目，請參閱[設定 AWS IoT 記錄](#)和[規則引擎日誌項目](#)。

- 使用新的物件索引鍵來更新 FileDescriptorSet 以及更新規則中的物件索引鍵。

您可以將更新後的描述項檔案上傳到 Amazon S3 儲存貯體來更新 FileDescriptorSet。系統可能需要最多 15 分鐘的時間來反映 FileDescriptorSet 的更新作業。為了避免這種延遲，理想做法是使用新的物件索引鍵上傳更新後的 FileDescriptorSet，並在規則中更新物件索引鍵。

## SQL 版本

AWS IoT 規則引擎使用類似 SQL 的語法從 MQTT 訊息中選取資料。SQL 陳述式會以指定的 SQL 版本來解釋，版本由描述規則的 JSON 文件的 awsIotSqlVersion 屬性指定。如需更多關於 JSON 規

則文件結構的資訊，請參閱[建立規則的相關文章](#)。此 `awsIotSqlVersion` 屬性可讓您指定要使用的 AWS IoT SQL 規則引擎版本。部署新版本時，可以繼續使用較早版本，或變更規則以使用新版本。目前的規則會繼續使用建立時使用的版本。

下列 JSON 範例顯示了如何使用 `awsIotSqlVersion` 屬性來指定 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"
    }
  ]
}
```

AWS IoT 目前支援下列 SQL 版本：

- 2016-03-23：建立於 2016 年 3 月 23 日的 SQL 版本（推薦）。
- 2015-10-08：建立於 2015 年 10 月 8 日的原始 SQL 版本。
- beta：最新的 SQL 版本。此版本可能導致規則受到中段變更。

## 2016 年 3 月 23 日 SQL 規則引擎版本的新功能

- 修正巢狀 JSON 物件的選取問題。
- 修正陣列查詢的問題。
- 支援物件內部查詢。如需詳細資訊，請參閱 [巢狀物件查詢](#)。
- 支援將陣列輸出為最上層物件。
- 除了 `encode(value, encodingScheme)` 函數外，它也可以應用於 JSON 和非 JSON 格式資料。如需詳細資訊，請參閱 [編碼函數](#)。

輸出一個 **Array** 作為最上層物件

此功能可讓規則將陣列作為最上層物件傳回。舉例而言，若為以下的 MQTT 訊息：

```
{
```

```
"a": {"b": "c"},  
"arr": [1, 2, 3, 4]  
}
```

此外以下規則：

```
SELECT VALUE arr FROM 'topic'
```

該規則會產生以下輸出。

```
[1, 2, 3, 4]
```

# AWS IoT Device Shadow 服務

AWS IoT Device Shadow 服務會將陰影新增至 AWS IoT 物件物件。無論裝置是否連線，陰影都可讓應用程式和其他服務使用裝置 AWS IoT 的狀態。AWS IoT 物件物件可以有許多命名陰影，因此您的 IoT 解決方案有更多選項可將裝置連接至其他應用程式和服務。

AWS IoT 在明確建立物件之前，物件物件沒有任何陰影。您可以使用 AWS IoT 控制台建立、更新和刪除陰影。裝置、其他 Web 用戶端和服務可使用 MQTT 及 [預留的 MQTT 主題](#)、使用 [Device Shadow REST API](#) 的 HTTP，以及 [AWS IoT 的 AWS CLI](#)。由於陰影是由 AWS 雲端儲存，因此無論裝置是否連線，它們都可以從應用程式和其他雲端服務收集和報告裝置狀態資料。

## 使用影子

影子會提供可靠的資料存放區，讓裝置、應用程式和其他雲端服務共用資料。它們可讓裝置、應用程式和其他雲端服務連線和中斷連線，而不會遺失裝置的狀態。

當裝置、應用程式和其他雲端服務連線到時 AWS IoT，裝置可透過陰影存取和控制裝置的目前狀態。例如，應用程式可以透過更新陰影來要求變更裝置的狀態。AWS IoT 會發佈訊息，指出裝置的變更。裝置會接收此訊息、更新其狀態以符合，並發佈具有更新狀態的訊息。Device Shadow 服務會在對應的影子中反映此更新的狀態。應用程式可以訂閱影子的更新，也可以查詢影子的目前狀態。

當設備離線時，應用程式仍然可以 AWS IoT 與設備的陰影進行通信。當裝置重新連線時，它會收到其影子的目前狀態，以便更新其狀態以符合其影子的狀態，然後發佈具有更新狀態的訊息。同樣地，當應用程式離線且裝置狀態在離線時變更時，裝置會保持影子更新，以便在重新連線時，應用程式可以查詢其目前狀態的影子。

如果您的裝置頻繁離線，而您希望將裝置設定為在重新連線後接收增量訊息，則可以使用持續工作階段功能。如需持續工作階段有效期限的詳細資訊，請參閱 [持續工作階段有效期限](#)。

## 選擇使用已命名或未命名的影子

Device Shadow 服務支援已命名、未命名或傳統的影子。一個物件可以有許多已命名的影子，並且不超過一個未命名的影子。物件也可以具有預留已命名影子，其運作方式與已命名影子類似，只是您無法更新其名稱。如需詳細資訊，請參閱 [預留已命名影子](#)。

一個物件物件可以同時具有已命名和未命名的影子；但是用於存取每個影子的 API 略有不同，所以決定最適合您解決方案的影子類型並僅使用該類型可能會更有效率。如需存取影子 API 的詳細資訊，請參閱 [影子主題](#)。

使用已命名的影子，您可以建立物件物件狀態的不同檢視。例如，您可以將具有許多屬性的物件物件分割成具有邏輯屬性群組的影子，每個屬性都用其影子名稱加以識別。您也可以透過將屬性分組成不同的影子，並使用原則來控制存取來限制屬性的存取權限。如需與裝置影子搭配使用之政策的詳細資訊，請參閱[適用於 AWS IoT 的動作、資源及條件索引鍵](#)及 [AWS IoT Core 政策](#)。

典型、未命名的影子比已命名的影子更簡單，但限制比已命名的影子多。每個物 AWS IoT 件物件只能有一個未命名的陰影。如果您預期 IoT 解決方案對影子資料的需求有限，這可能就是您要開始使用影子的契機。但是，如果您認為未來可能想要增加其他影子，請考慮從一開始就使用已命名的影子。

機群索引以不同方式支援未命名的影子和已命名的影子。如需詳細資訊，請參閱[管理機群索引](#)。

## 存取影子

每個影子都有預留的 [MQTT 主題](#) 和 [HTTP URL](#)，其支援在影子上的 get、update 和 delete 動作。

影子會使用 [JSON 影子文件](#) 來儲存和擷取資料。影子的文件包含狀態屬性，描述裝置狀態的下列層面：

- desired

應用程式會透過更新 desired 物件來指定裝置屬性的所需狀態。

- reported

裝置會報告 reported 物件中的目前狀態。

- delta

AWS IoT 報告物件中所需狀態與報告狀態之間的差delta異。

儲存在影子中的資料會由更新動作訊息內文的狀態屬性決定。後續的更新動作可以修改現有資料物件的值，也可以在影子的狀態物件中新增和刪除金鑰和其他元素。如需存取影子的詳細資訊，請參閱 [在裝置中使用影子](#) 和 [在應用程式和服務中使用影子](#)。

### Important

提出更新要求的權限應限於受信任的應用程式和裝置。這可以防止影子的狀態屬性遭意外地變更；否則，使用影子的裝置和應用程式的設計應預期狀態屬性中的金鑰會變更。

## 在裝置、應用程式和其他雲端服務中使用影子

在裝置、應用程式和其他雲端服務中使用影子需要一致性和協調性。AWS IoT Device Shadow 服務會儲存陰影狀態、在陰影狀態變更時傳送訊息，以及回應變更其狀態的訊息。IoT 解決方案中的裝置、應用程式和其他雲端服務必須管理其狀態，並使其與 Device Shadow 狀態保持一致。

影子狀態資料是動態的，可由裝置、應用程式和其他具有存取影子權限的雲端服務進行變更。基於這個原因，請務必考慮每個裝置、應用程式和其他雲端服務如何與影子互動。例如：

- 將狀態資料傳到影子時，設備應該只會寫入影子狀態的 `reported` 屬性。
- 透過影子將狀態變更要求傳給裝置時，應用程式和其他雲端服務應該只會寫入 `desired` 屬性。

### Important

包含在影子資料物件中的資料是獨立於其他影子和其他物件物件屬性的資料，例如物件的屬性和物件物件的裝置可能發佈的 MQTT 訊息內容。但如有必要，裝置可以在不同的 MQTT 主題和影子中報告相同的資料。

支援多重影子的裝置必須維持它在不同影子中報告之資料的一致性。

## 訊息順序

不保證來自 AWS IoT 服務的消息將以任何特定順序到達設備。下列情境說明在此情況下會發生的情況。

初始狀態文件：

```
{
  "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
}
```

最終狀態文件 :

```
{
  "state": {
    "reported": {
      "color": "GREEN"
    }
  },
  "version": 12,
  "timestamp": 123456779
}
```

這樣會產生兩個增量訊息 :

```
{
  "state": {
    "color": "RED"
  },
  "version": 11,
```



```
"timestamp": 123456778
}
```

```
{
  "state": {
    "color": "GREEN"
  },
  "version": 12,
  "timestamp": 123456779
}
```

裝置可能不會依此順序收到這些訊息。由於這些訊息中的狀態是累積的，因此裝置可以安全捨棄比追蹤中版本編號更舊的所有訊息。如果裝置在接收到版本 11 的增量之前收到了版本 12 的增量，則可安全捨棄版本 11 的訊息。

## 裁剪影子訊息

若要減少傳送至您裝置的影子訊息大小，請定義只選擇了裝置所需之欄位的規則，然後將訊息重新發佈至裝置目前監聽的 MQTT 主題。

規則是在 JSON 中指定，並應如下所示：

```
{
  "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 陳述會決定將訊息的哪些欄位重新發佈至指定主題。使用 "+" 萬用字元表示所有影子名稱。規則會指定所有符合的訊息都應重新發佈至指定主題。在這種情況下，該 "topic()" 函數是用於指定要重新發佈的主題。topic(3) 會評估原始主題中的物件名稱。如需關於建立規則的詳細資訊，請參閱 [的規則 AWS IoT](#)。

## 在裝置中使用影子

本節說明使用 MQTT 訊息進行陰影的裝置通訊，MQTT 訊息是裝置與 Dev AWS IoT Core Shadow 服務通訊的偏好方法。

影子通訊功能會模擬使用 MQTT 的發佈/訂閱通訊模型的請求/回應模型。每個影子動作包括一個請求主題，一個成功的回應主題 (accepted) 和一個錯誤響應主題 (rejected)。

如果您希望應用程式和服務能夠判斷裝置是否已連線，請參閱 [偵測裝置已連線](#)。

### Important

因 MQTT 會使用發佈/訂閱通訊模型，因此您應先訂閱回應主題，再發佈請求主題。若您並未這麼做，則將不會收到發佈請求的回應。

若您使用 [AWS IoT Device SDK](#) 呼叫 Device Shadow 服務 API，則會為您處理。

本節中的範例使用主題的縮寫形式，其中 *ShadowTopicPrefix* 可參考具名或未命名的陰影，如本表所述。

影子可以為已命名或未命名 (典型)。各影子所使用的主題只有在主題字首中有所不同。此表格會顯示每種影子類型所使用的主題字首。

<i>ShadowTopicPrefix</i> 值	影子類型
<code>\$aws/things/ <i>thingName</i> /shadow</code>	未命名 (經典) 影子
<code>\$aws/things/ <i>thingName</i> /shadow/name/ <i>shadowName</i></code>	已命名影子

### Important

確保您的應用程式或服務對影子的使用符合一致性，並且受您裝置中對應的實作支援。例如，請考慮如何建立、更新和刪除影子。也請考慮如何在裝置中處理更新，以及透過影子存取裝置的應用程式或服務。您的設計應該清楚了解裝置狀態如何更新和報告，以及您的應用程式和服務如何與裝置及其影子互動。

若要建立完整的主題，請在 *ShadowTopicPrefix* 選取您要參照的影子類型、用對應值取代 *thingName* 及 *shadowName* (如適用) 的影子類型，然後將該類型附加至主題 stub，如下表所示。請記住，主題會區分大小寫。

如需有關影子保留主題的詳細資訊，請參閱 [影子主題](#)。

## 在第一次連接時初始化設備 AWS IoT

在設備註冊後 AWS IoT，它應該訂閱這些 MQTT 消息以獲取其支持的陰影。

主題	意義	收到此主題時裝置應採取的動作
<i>ShadowTopicPrefix</i> / delete/accepted	請delete求被接受並 AWS IoT 刪除了陰影。	適應已刪除影子所需的動作，例如停止發佈更新。
<i>ShadowTopicPrefix</i> / delete/rejected	請delete求已被拒絕，而 AWS IoT 且陰影未刪除。訊息內文包含錯誤資訊。	在訊息內文中回應錯誤訊息。
<i>ShadowTopicPrefix</i> / get/accepted	要get求已被接受 AWS IoT，且郵件內文包含目前的陰影文件。	處理訊息內文中的狀態文件所需的動作。
<i>ShadowTopicPrefix</i> / get/rejected	要get求已被拒絕 AWS IoT，且郵件內文包含錯誤資訊。	在訊息內文中回應錯誤訊息。
<i>ShadowTopicPrefix</i> / update/accepted	要update求已被接受 AWS IoT，且郵件內文包含目前的陰影文件。	確認訊息內文中的更新資料符合裝置狀態。
<i>ShadowTopicPrefix</i> / update/rejected	要update求已被拒絕 AWS IoT，且郵件內文包含錯誤資訊。	在訊息內文中回應錯誤訊息。
<i>ShadowTopicPrefix</i> / update/delta	陰影文件已由要求更新 AWS IoT，且郵件內文包含要求的變更。	更新裝置的狀態，以符合訊息內文中所需的狀態。

主題	意義	收到此主題時裝置應採取的動作
<code>ShadowTopicPrefix / update/documents</code>	最近已完成影子的更新，而訊息內文包含目前的影子文件。	確認訊息內文中的更新狀態符合裝置的狀態。

訂閱上表中每個影子的訊息之後，裝置應該測試，看看它支援的影子是否已經透過將 `/get` 主題發佈到每個影子來建立。如果收到 `/get/accepted` 訊息，訊息內文會包含影子文件，裝置可用來初始化其狀態。如果收到 `/get/rejected` 訊息，應該透過發佈具有目前裝置狀態的 `/update` 訊息來建立影子。

例如，假設您有一個沒有任何經典或已命名影子的物件 `My_IoT_Thing`。如果您現在於保留主題 `$aws/things/My_IoT_Thing/shadow/get` 上發佈 `/get` 請求，則會在 `$aws/things/My_IoT_Thing/shadow/get/rejected` 主題上傳回錯誤，因為物件沒有任何影子。若要解決此錯誤，請先使用 `$aws/things/My_IoT_Thing/shadow/update` 主題發佈 `/update` 訊息與目前裝置狀態 (如以下承載)。

```
{
  "state": {
    "reported": {
      "welcome": "aws-iot",
      "color": "yellow"
    }
  }
}
```

現在會為物件建立一個經典影子，並將訊息發佈至 `$aws/things/My_IoT_Thing/shadow/update/accepted` 主題。如果發佈至主題 `$aws/things/My_IoT_Thing/shadow/get`，則會將回應與裝置狀態傳回給 `$aws/things/My_IoT_Thing/shadow/get/accepted` 主題。

對於已命名的影子，在使用 `get` 請求之前，必須先建立已命名的影子或發佈含有影子名稱的更新。例如，若要建立已命名的影子 `namedShadow1`，請先將裝置狀態資訊發佈至主題 `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/update`。若要擷取狀態資訊，請對已命名的影子 `$aws/things/My_IoT_Thing/shadow/name/namedShadow1/get` 使用 `/get` 請求。

## 在設備連接到時處理消息 AWS IoT

當設備連接到時 AWS IoT，它可以接收 `/update/delta` 消息，並應通過以下方式使設備狀態與其陰影中的更改保持匹配：

1. 讀取接收到的所有 `/update/delta` 訊息，並同步處理裝置狀態以進行比對。
2. 只要裝置狀態發生變化時，便使用具有裝置目前狀態的 `reported` 訊息內文發佈 `/update` 訊息。

當裝置連線時，它應該在指示時發佈這些訊息。

指示	主題	承載
裝置的狀態已變更。	<code>ShadowTopicPrefix / update</code>	具有 <code>reported</code> 屬性的影子文件。
裝置可能不會與影子同步。	<code>ShadowTopicPrefix /get</code>	(空白)
裝置上的動作表示裝置將不再支援影子，例如移除或取代裝置時。	<code>ShadowTopicPrefix / delete</code>	(空白)

## 裝置重新連線時處理訊息 AWS IoT

當具有一個或多個陰影的設備連接到時 AWS IoT，它應該通過以下方式將其狀態與其支持的所有陰影的狀態同步：

1. 讀取接收到的所有 `/update/delta` 訊息，並同步處理裝置狀態以進行比對。
2. 使用具有裝置目前狀態的 `reported` 訊息內文發佈 `/update` 訊息。

## 在應用程式和服務中使用影子

本節說明應用程式或服務如何與 AWS IoT Device Shadow 服務互動。此範例假設應用程式或服務只透過影子與該裝置互動。此範例不包含任何管理動作，例如建立或刪除影子。

此範例使用 AWS IoT Device Shadow 服務的 REST API 與陰影互動。與用於 [在裝置中使用影子](#) 中的範例 (使用發佈/訂閱通訊模型) 不同，此範例使用 REST API 的請求/回應通訊模型。這表示應用程式或服務必須先提出要求，才能收到來自的回應 AWS IoT。不過，此模型的缺點是它不支援通知。如果您的應用程式或服務需要及時通知裝置狀態變更，請考慮透過 WSS 通訊協定支援發佈/訂閱通訊模型的 MQTT 或 MQTT，如中所述 [在裝置中使用影子](#)。

### ⚠ Important

請確定您的應用程式或服務對影子的使用與裝置中的對應實作一致，並支援這些功能。例如，請考慮如何建立、更新和刪除影子，以及如何在裝置和存取影子的應用程式或服務中處理更新。您的設計應該清楚指定裝置狀態的更新和報告方式，以及您的應用程式和服務如何與裝置及其影子互動。

REST API 的已命名影子 URL 是：

```
https://endpoint/things/thingName/shadow?name=shadowName
```

和一個未命名的影子：

```
https://endpoint/things/thingName/shadow
```

其中：

端點

CLI 命令傳回的端點：

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

thingName

影子所屬物件物件的名稱

shadowName

已命名影子的名稱。此參數不會與未命名影子搭配使用。

## 在連接時初始化應用程序或服務 AWS IoT

當應用程序第一次連接到時 AWS IoT，它應該向它使用的陰影的 URL 發送 HTTP GET 請求，以獲取它正在使用的陰影的當前狀態。這可讓它將應用程式或服務同步到影子。

## 在應用程式或服務連線時處理狀態變更 AWS IoT

當應用程式或服務連線到時 AWS IoT，它可以透過在所使用陰影的 URL 上傳送 HTTP GET 要求，定期查詢目前的狀態。

當使用者與應用程式或服務互動以變更裝置狀態時，應用程式或服務可以將 HTTP POST 請求傳送至用來更新 desired 影子狀態的影子 URL。此請求會傳回已接受的變更，但是您可能必須透過提出 HTTP GET 請求來輪詢影子，直到裝置已更新其新狀態的影子。

### 偵測裝置已連線

若要判斷裝置目前是否已連線，請在影子文件中包含 `connected` 屬性，並使用 MQTT Last Will 和 Testament (LWT) 訊息以將 `connected` 屬性設為 `false` (若裝置因錯誤而中斷連線)。

#### Note

AWS IoT Device Shadow 服務會忽略傳送至 AWS IoT 保留主題 (以 \$ 開頭的主題) 的 MQTT LWT 訊息。但是，它們由訂閱的用戶端和 AWS IoT 規則引擎處理，因此您需要建立傳送至非保留主題的 LWT 訊息，以及將 MQTT LWT 訊息重新發佈為陰影保留更新主題的陰影更新訊息的規則。 *ShadowTopicPrefix*/update

### 傳送 LWT 訊息 Device Shadow 服務

1. 建立重新發佈保留主題上的 MQTT LWT 訊息的規則。下列範例規則會接聽有關 `my/things/myLightBulb/update` 主題的訊息，並將其重新發佈至 `$aws/things/myLightBulb/shadow/update`。

```
{
  "rule": {
    "ruleDisabled": false,
    "sql": "SELECT * FROM 'my/things/myLightBulb/update'",
    "description": "Turn my/things/ into $aws/things/",
    "actions": [
      {
        "republsh": {
          "topic": "$aws/things/myLightBulb/shadow/update",
          "roleArn": "arn:aws:iam:123456789012:role/aws_iot_republsh"
        }
      }
    ]
  }
}
```

```
    ]  
  }  
}
```

2. 當設備連接到時 AWS IoT，它會將 LWT 消息註冊到非保留主題，以便重新發布規則進行識別。在此範例中，該主題是 `my/things/myLightBulb/update`，它會將連線的內容設定為 `false`。

```
{  
  "state": {  
    "reported": {  
      "connected": "false"  
    }  
  }  
}
```

3. 連線後，裝置會在其影子更新主題上發佈訊息 `$aws/things/myLightBulb/shadow/update`，並報告其目前狀態，其中包括將其 `connected` 屬性設定為 `true`。

```
{  
  "state": {  
    "reported": {  
      "connected": "true"  
    }  
  }  
}
```

4. 在裝置正常中斷連線之前，它會發佈其影子更新主題的訊息 `$aws/things/myLightBulb/shadow/update`，以報告其最新狀態，其中包括將其 `connected` 屬性設定為 `false`。

```
{  
  "state": {  
    "reported": {  
      "connected": "false"  
    }  
  }  
}
```

5. 如果裝置因錯誤而中斷連線，則 AWS IoT 訊息代理程式會代表裝置發佈裝置的 LWT 訊息。重新發布規則會偵測到此訊息，並發佈影子更新訊息，以更新 Device Shadow 的 `connected` 屬性。



# 模擬 Device Shadow 服務通訊

本主題示範如何將 Device Shadow 服務如何作為中繼點，並允許裝置和應用程式使用影子來更新、儲存和擷取裝置的狀態。

若要示範本主題所描述的互動，並進一步探索它，您需要一個 AWS 帳戶 可以在其上執行 AWS CLI。如果您沒有這些項目，您仍然可以在程式碼範例中看到互動。

在此範例中，主 AWS IoT 控制台代表裝置。AWS CLI 代表透過陰影方式存取裝置的應用程式或服務。該 AWS CLI 界面與應用程序可能用於通信的 API 非常相似 AWS IoT。此範例中的裝置是智慧型燈泡，應用程式會顯示燈泡的狀態，而且可以變更燈泡的狀態。

## 準備模擬

以下程序會開啟模擬裝置的 [AWS IoT 主控台](#)，以及模擬應用程式的命令列視窗來初始化模擬。

### 準備模擬環境的步驟

1. 您需 AWS 帳戶 要一個自行運行此主題中的示例。如果您沒有 AWS 帳戶，請建立一個，如中所述 [設定您的 AWS 帳戶](#)。
2. 開啟 [AWS IoT 主控台](#)，然後在左側選單中選擇 Test (測試) 以開啟 MQTT client (MQTT 用戶端)。
3. 在另一個視窗中，在已安裝 AWS CLI 的系統上開啟終端機視窗。

您應該打開兩個窗口：一個在「測試」頁面上顯示 AWS IoT 控制台，另一個帶有命令行提示符。

## 初始化裝置

在這個模擬中，我們將使用一個名為 mySimulatedThing，和它的陰影命名為 SimShadow 1 的物件。

### 建立物件及其 IoT 政策

如要建立物件物件，請於 AWS IoT 主控台中：

1. 選擇 Manage (管理)，然後選擇 Things (物件)。
2. 如果已列出項目，請按一下「建立」按鈕，否則請按一下「註冊單一物件」以建立單一 AWS IoT 物件。
3. 輸入名稱 mySimulatedThing，將其他設定保留為預設值，然後按一下 Next (下一步)。

4. 使用單鍵憑證建立來產生憑證，其會驗證裝置到 AWS IoT 的連線。按一下 Activate (啟用)，啟用憑證。
5. 您可連接政策 My\_IoT\_Policy，其將授與裝置發佈和訂閱 MQTT 預留主題的許可。如需如何建立 AWS IoT 物件以及如何建立此原則的詳細步驟，請參閱[建立物件](#)。

### 為物件建立已命名的影子

您可以向主題 `$aws/things/mySimulatedThing/shadow/name/simShadow1/update` 發佈更新請求來為物件建立已命名的影子，如下方所述。

或者，要創建一個命名的陰影：

1. 在 AWS IoT Console (主控台) 中，在顯示的物件清單中選擇您的物件，然後選擇 Shadows (影子)。
2. 選擇 Add a shadow (新增影子)，輸入名稱 `simShadow1`，接著選擇 Create (建立)，以新增命名的影子。

### 訂閱並發佈至預留 MQTT 主題

於主控台中，訂閱預留 MQTT 影子主題。這些主題是對 `get`、`update` 和 `delete` 動作的回應，以便您的裝置在發佈動作之後就可以接收回應。

### 訂閱 MQTT 用戶端中的 MQTT 主題

1. 於 MQTT client (MQTT 用戶端) 中，選擇 Subscribe to a topic (訂閱主題)。
2. 輸入 `get`、`update` 和 `delete` 主題以進行訂閱。從下方清單一次複製一個主題，將其貼到 Topic filter (主題篩選條件) 欄位，再按一下 Subscribe (訂閱)。您應會看到顯示於 Subscriptions (訂閱) 之下的主題。

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/delete/accepted`
- `$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`

此時，您的模擬裝置已準備好接收由 AWS IoT 發佈的主題。

## 如要訂閱 MQTT 用戶端中的 MQTT 主題

在裝置自行初始化並訂閱回應主題之後，它會查詢所支援的影子。此模擬僅支援一個陰影，即支援名為 SimShadow 1 的物件物件的陰影。mySimulatedThing

從 MQTT 用戶端取得目前影子狀態

1. 於 MQTT 用戶端中，選擇 Publish to a topic (發佈至主題)。
2. 在 Publish (發佈) 之下，輸入下列主題，然後刪除下方主題訊息內文視窗的任何內容，您可在其中輸入要取得的主題。接著，您可選擇 Publish to topic (發佈至主題)，發佈請求。`$aws/things/mySimulatedThing/shadow/name/simShadow1/get`。

若您尚未建立命名的影子 `simShadow1`，您收到了 `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/rejected` 主題中的一則訊息，且 code 為 404，(如此範例中所示)，則系統尚未建立該影子，因此我們接下來將其建立。

```
{
  "code": 404,
  "message": "No shadow exists with name: 'simShadow1'"
}
```

## 建立具有裝置目前狀態的影子

1. 在 MQTT 用戶端中，選擇 Publish to a topic (發佈到主題)，然後輸入此主題：

```
$aws/things/mySimulatedThing/shadow/name/simShadow1/update
```

2. 在您輸入主題的下方訊息內文視窗中，輸入此影子文件以顯示裝置正在報告其 ID 及其目前顏色的 RGB 值。選擇 Publish (發佈)，發佈請求。

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
```

```
        128,  
        128  
    ]  
  }  
},  
"clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"  
}
```

若您在主題中收到一則訊息：

- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/accepted`：這表示已建立影子，而訊息內文包含目前的影子文件。
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/update/rejected`：檢閱訊息內文中的錯誤。
- `$aws/things/mySimulatedThing/shadow/name/simShadow1/get/accepted`：影子已存在，且訊息內文具有目前的影子狀態，例如在此範例中。有了這個，您可以設置您的裝置或確認它符合影子狀態。

```
{  
  "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"
}
```

## 從應用程式傳送更新

本節使用示 AWS CLI 範應用程式如何與陰影互動。

若要取得陰影的目前狀態，請使用 AWS CLI

在命令列輸入此命令：

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 /dev/stdout
```

在 Windows 平台上，您可使用 con，而非 /dev/stdout。

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name simShadow1 con
```

因為影子已存在，而且已由裝置初始化以反映其目前狀態，所以它會傳回下列的影子文件。

```
{
  "state": {
    "reported": {
      "ID": "SmartLamp21",
      "ColorRGB": [
        128,
        128,
        128
      ]
    }
  },
  "metadata": {
    "reported": {
```

```
"ID": {
  "timestamp": 1591140517
},
"ColorRGB": [
  {
    "timestamp": 1591140517
  },
  {
    "timestamp": 1591140517
  },
  {
    "timestamp": 1591140517
  }
]
},
"version": 3,
"timestamp": 1591141111
}
```

應用程式可以使用此回應來初始化對其裝置狀態的表示法。

如果應用程式更新狀態，例如當使用者將智慧型燈泡的顏色變更為黃色時，應用程式會傳送 `update-thing-shadow` 命令。此命令會對應 `UpdateThingShadow` REST API。

從應用程式更新影子

在命令列輸入此命令：

AWS 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
```

AWS 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，這個命令應該會傳回下列影子文件。

```
{
  "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"
}
```

## 回應裝置中的更新

返回到 AWS 控制台中的 MQTT 客戶端，您應該看到發 AWS IoT 布以反映上一節中發布的更新命令的消息。

在 MQTT 用戶端中檢視更新訊息

在 MQTT 客戶端中，在訂閱列中選擇 \$ aws/事物//影子/名mySimulatedThing稱/模擬影子1/更新/delta。如果主題名稱被截斷，您可以暫停以查看完整主題。在本主題的主題記錄中，您應會看到類似此訊息的 /delta 訊息。

```
{
  "version": 4,
  "timestamp": 1591141596,
  "state": {
    "ColorRGB": [
      255,
      255,
      0
    ]
  },
  "metadata": {
    "ColorRGB": [
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      },
      {
        "timestamp": 1591141596
      }
    ]
  },
  "clientToken": "21b21b21-bfd2-4279-8c65-e2f697ff4fab"
}
```

您的裝置會處理此訊息的內容，以將裝置狀態設定為符合訊息中的 `desired` 狀態。

裝置更新狀態以符合訊息中的 `desired` 狀態之後，必須 AWS IoT 透過發佈更新訊息將新報告的狀態傳送回。此程序會在 MQTT 用戶端中模擬這個部分。

### 從裝置更新影子

1. 於 MQTT 用戶端 中，選擇 `Publish to a topic` (發佈至主題)。
2. 在訊息內文視窗的訊息內文視窗上方的主題欄位中，輸入影子的主題，接著輸入 `/update` 動作：`$aws/things/mySimulatedThing/shadow/name/simShadow1/update`，並在訊息內文中輸入此更新的影子文件，其中說明裝置的目前狀態。選擇 `Publish` (發佈)，發佈更新的裝置狀態。

```
{
  "state": {
    "reported": {
```



```
    "ColorRGB": [255,255,0]
  }
},
"clientToken": "a4dc2227-9213-4c6a-a6a5-053304f60258"
}
```

如果訊息已成功接收 AWS IoT，您應該會在 MQTT 用戶端中看到具有陰影目前狀態的 MQTT 用戶端中的 \$aws/things/ mySimulatedThing 陰影 /名稱/Simshadow1/更新/接受的訊息記錄檔中的新回應，例如本範例。

```
{
  "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"
}
```

成功更新裝置的報告狀態也會導致 AWS IoT 將訊息中陰影狀態的完整描述傳送至主題，例如此訊息內文是由裝置在上述程序中執行陰影更新所造成的。

```
{
  "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"
}
```

## 觀察應用程式中的更新

應用程式現在可以查詢裝置報告的目前狀態的影子。

若要取得陰影的目前狀態，請使用 AWS CLI

1. 在命令列輸入此命令：

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
simShadow1 /dev/stdout
```

在 Windows 平台上，您可使用 con，而非 /dev/stdout。

```
aws iot-data get-thing-shadow --thing-name mySimulatedThing --shadow-name
simShadow1 con
```

2. 因為裝置剛剛更新影子，以反映其目前的狀態，所以應該傳回下列影子文件。

```
{
```

```
"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
}
```

## 超越模擬

試驗 AWS CLI (代表應用程式) 與主控台 (代表裝置) 之間的互動，以建立 IoT 解決方案的模型。

## 與影子互動

本主題說明與 AWS IoT 提供使用影子的三種方法中的每一種相關聯的訊息。這些方法包括下列各項：

### UPDATE

建立一個影子，如果它不存在，或者使用訊息正文中提供的狀態訊息更新現有影子的內容。AWS IoT 會記錄每次更新的時間戳記，以指出上次更新狀態的時間。當陰影的狀態變更時，AWS IoT 會傳送/delta訊息給所有 MQTT 訂閱者，desired且狀態與狀態之間的差異。reported接收 / delta 訊息的裝置或應用程式會根據差異執行動作。例如，裝置可以更新其狀態至所需的狀態，或者應用程式可以更新 UI 以反映裝置狀態的改變。

### GET

擷取包含影子完整狀態的目前影子文件，包括中繼資料。

### DELETE

刪除裝置影子及其內容。

您無法還原已刪除的裝置影子文件，但可以使用已刪除的裝置影子文件的名稱建立新的裝置影子。如果您建立的裝置影子文件名稱與過去 48 小時內刪除的文件名稱相同，則新裝置影子文件的版本號碼會跟隨已刪除影子的版本號碼。如果裝置影子文件已刪除超過 48 小時，則使用相同名稱的新裝置影子文件的版本號碼將會是 0。

## 通訊協定支援

AWS IoT 通過 HTTPS 協議支持 [MQTT](#) 和 REST API 以與陰影進行交互。AWS IoT 為 MQTT 發行和訂閱動作提供一組保留的請求與回應主題。裝置和應用程式應先訂閱回應主題，然後再發佈要求主

題，以取得有關如何 AWS IoT 處理要求的資訊。如需詳細資訊，請參閱 [Device Shadow MQTT 主題](#) 和 [Device Shadow REST API](#)。

## 請求和報告狀態

使用 AWS IoT 和陰影設計 IoT 解決方案時，您應該決定將要求變更的應用程式或裝置，以及將實作這些變更的應用程式或裝置。一般而言，裝置會實作並報告變更回影子，而應用程式和服務會回應並請求影子中的變更。您的解決方案可能不同，但本主題中的範例假設用戶端應用程式或服務請求變更影子中，而裝置會執行變更，並將它們回報給影子。

## 更新影子

您的應用程式或服務可以使用 [UpdateThingShadow](#) API 或發佈至 [/update](#) 主題來更新影子的狀態。更新只會影響請求中所指定的欄位。

### 當用戶端請求狀態變更時更新影子

當用戶端使用 MQTT 通訊協定請求影子中的狀態變更

1. 用戶端應具有目前的影子文件，以便識別要變更的屬性。請參閱 [/get](#) 動作，了解如何取得目前的影子文件。
2. 用戶端會訂閱下列 MQTT 主題：
  - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
  - `$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. 用戶端會發佈 `$aws/things/thingName/shadow/name/shadowName/update` 請求主題，其中包含所需的影子狀態的狀態文件。只有要變更的屬性需要包含在文件中。這是具有所需狀態的文件的範例。

```
{
  "state": {
    "desired": {
      "color": {
        "r": 10
      },
      "engine": "ON"
    }
  }
}
```

```
}
}
```

4. 如果更新要求有效，請在陰影中 AWS IoT 更新所需的狀態，並發佈下列主題的訊息：

- \$aws/things/*thingName*/shadow/name/*shadowName*/update/accepted
- \$aws/things/*thingName*/shadow/name/*shadowName*/update/delta

/update/accepted 訊息包含 [/accepted response state document](#) 影子文件，而 /update/delta 訊息包含 [/delta response state document](#) 影子文件。

5. 如果更新要求無效，會 AWS IoT 發佈一則訊息，其中包含描述錯誤的[錯誤回應文件](#)陰影文件的 \$aws/things/*thingName*/shadow/name/*shadowName*/update/rejected 主題。

當用戶端請求在影子中使用 API 的狀態變更

1. 用戶端呼叫 [UpdateThingShadow](#) API，包含作為其訊息內文的 [請求狀態文件](#) 狀態文件。
2. 如果要求有效，會 AWS IoT 傳回 HTTP 成功回應碼和 [/accepted response state document](#) 陰影文件做為其回應訊息內文。

AWS IoT 也會針對任何訂閱的裝置或用戶端，將 MQTT 訊息以及 [/delta response state document](#) 陰影文件的 \$aws/things/*thingName*/shadow/name/*shadowName*/update/delta 主題發佈至主題。

3. 如果要求無效，會 AWS IoT 傳回 HTTP 錯誤回應碼 a [錯誤回應文件](#) 做為其回應訊息內文。

當裝置收到有關 /update/delta 主題的 /desired 狀態時，它會在裝置中進行所需的變更。然後，它會將訊息傳送至 /update 主題，向影子報告其目前狀態。

當裝置回報其目前狀態時更新影子

當裝置使用 MQTT 通訊協定向影子報告其目前的狀態

1. 裝置應該在更新影子之前訂閱這些 MQTT 主題：
  - \$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. 裝置會將訊息發佈至報告目前狀態的 `$aws/things/thingName/shadow/name/shadowName/update` 主題，例如在此範例中，以報告其目前狀態。

```
{
  "state": {
    "reported" : {
      "color" : { "r" : 10 },
      "engine" : "ON"
    }
  }
}
```

3. 如果 AWS IoT 接受更新，則會將訊息發佈至具有 [/accepted response state document](#) 陰影文件的 `$aws/things/thingName/shadow/name/shadowName/update/accepted` 主題。
4. 如果更新要求無效，會 AWS IoT 發佈一則訊息，其中包含描述錯誤的 [錯誤回應文件](#) 陰影文件的 `$aws/things/thingName/shadow/name/shadowName/update/rejected` 主題。

當裝置藉由使用 API 向影子報告其目前的狀態

1. 裝置會呼叫 [UpdateThingShadow](#) API，包含 [請求狀態文件](#) 狀態文件作為其訊息正文。
2. 如果要求有效，請 AWS IoT 更新陰影，並傳回含有 [/accepted response state document](#) 陰影文件做為其回應訊息內文的 HTTP 成功回應碼。

AWS IoT 也會針對任何訂閱的裝置或用戶端，將 MQTT 訊息以及 [/delta response state document](#) 陰影文件的 `$aws/things/thingName/shadow/name/shadowName/update/delta` 主題發佈至主題。

3. 如果要求無效，會 AWS IoT 傳回 HTTP 錯誤回應碼 a [錯誤回應文件](#) 做為其回應訊息內文。

## 樂觀鎖定

您可以使用狀態文件版本，確認您正在更新的裝置影子文件為最新版本。當您為更新請求提供版本，若狀態文件的目前版本與提供的版本不符，則服務會拒絕請求並顯示 HTTP 409 衝突回應代碼。衝突回應代碼也會出現在任何修改 ThingShadow 的 API 上，包括 DeleteThingShadow。

例如：

初始文件：

```
{
```

```
"state": {
  "desired": {
    "colors": [
      "RED",
      "GREEN",
      "BLUE"
    ]
  }
},
"version": 10
}
```

更新：(版本不符合；系統將拒絕請求)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 9
}
```

結果：

```
{
  "code": 409,
  "message": "Version conflict",
  "clientToken": "426bfd96-e720-46d3-95cd-014e3ef12bb6"
}
```

更新：(版本符合；將接受請求)

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  }
}
```

```
  },
  "version": 10
}
```

最終狀態：

```
{
  "state": {
    "desired": {
      "colors": [
        "BLUE"
      ]
    }
  },
  "version": 11
}
```

## 擷取影子文件

您可以使用 [GetThingShadow](#) API 或訂閱並發行至 `/get` 主題來擷取影子文件。這會擷取完整的影子文件，包括 `desired` 和 `reported` 狀態之間的任何差異。無論裝置或用戶端正在提出請求，此工作的程序都是相同的。

使用 MQTT 通訊協定擷取影子文件

1. 裝置或用戶端應該在更新影子之前訂閱這些 MQTT 主題：
  - `$aws/things/thingName/shadow/name/shadowName/get/accepted`
  - `$aws/things/thingName/shadow/name/shadowName/get/rejected`
2. 裝置或用戶端會將訊息發佈至具有空白訊息內文的 `$aws/things/thingName/shadow/name/shadowName/get` 主題。
3. 如果要求成功，則會將訊息 AWS IoT 發佈至訊息內文 [accepted response state document](#) 中的 `$aws/things/thingName/shadow/name/shadowName/get/accepted` 主題。
4. 如果要求無效，請將訊息 AWS IoT 發佈至訊息內文 [錯誤回應文件](#) 中的 `$aws/things/thingName/shadow/name/shadowName/get/rejected` 主題。

使用 REST API 擷取影子文件

1. 裝置或用戶端會使用空的訊息內文呼叫 [GetThingShadow](#) API。

2. 如果要求有效，會 AWS IoT 傳回含有 [/accepted response state document](#) 陰影文件做為其回應訊息內文的 HTTP 成功回應碼。
3. 如果要求無效，會 AWS IoT 傳回 HTTP 錯誤回應碼 a [錯誤回應文件](#) 做為其回應訊息內文。

## 刪除影子資料

有兩種方法可以刪除影子資料：您可以刪除影子文件中的特定屬性，也可以完全刪除影子。

- 若要從影子中刪除特定屬性，請更新影子；不過，請將您要刪除的屬性值設定為 null。值為 null 的欄位會從影子文件中移除。
- 若要刪除整個影子，請使用 [DeleteThingShadow](#) API 或發佈至 [/delete](#) 主題。

### Note

刪除陰影不會一次將其版本號碼重設為零。其將於 48 小時後重設為零。

## 從影子文件中刪除屬性

使用 MQTT 通訊協定從影子刪除屬性

1. 裝置或用戶端應該有目前的影子文件，以便識別要變更的屬性。如需如何取得目前影子文件的詳細資訊，請參閱 [擷取影子文件](#)。
2. 裝置或用戶端會訂閱下列 MQTT 主題：
  - `$aws/things/thingName/shadow/name/shadowName/update/accepted`
  - `$aws/things/thingName/shadow/name/shadowName/update/rejected`
3. 裝置或用戶端會發佈 `$aws/things/thingName/shadow/name/shadowName/update` 請求主題，其中包含狀態文件，該文件會將 null 值指派給要刪除的影子屬性。只有要變更的屬性需要包含在文件中。這是刪除 engine 屬性的文件範例。

```
{
  "state": {
    "desired": {
      "engine": null
    }
  }
}
```

```
}

```

4. 如果更新要求有效，AWS IoT 會刪除陰影中指定的屬性，並在郵件內文中發行含有 [/accepted response state document](#) 陰影文件 `$aws/things/thingName/shadow/name/shadowName/update/accepted` 主題的郵件。
5. 如果更新要求無效，會 AWS IoT 發佈一則訊息，其中包含描述錯誤的 [錯誤回應文件](#) 陰影文件的 `$aws/things/thingName/shadow/name/shadowName/update/rejected` 主題。

## 使用 REST API 從影子刪除屬性

1. 裝置或用戶端會呼叫 [UpdateThingShadow](#) API，其具有將 null 值指派給要刪除之影子屬性的 [請求狀態文件](#)。只包含您要在文件中刪除的屬性。這是刪除 engine 屬性的文件範例。

```
{
  "state": {
    "desired": {
      "engine": null
    }
  }
}
```

2. 如果要求有效，會 AWS IoT 傳回 HTTP 成功回應碼和 [/accepted response state document](#) 陰影文件做為其回應訊息內文。
3. 如果要求無效，會 AWS IoT 傳回 HTTP 錯誤回應碼 a [錯誤回應文件](#) 做為其回應訊息內文。

## 刪除影子

以下是刪除裝置影子時的一些考量事項。

- 將裝置的影子狀態設定為 null 不會刪除影子。影子版本將在下一次更新時遞增。
- 刪除裝置的影子並不會刪除物件物件。刪除物件物件也不會刪除對應的裝置影子。
- 刪除陰影不會一次將其版本號碼重設為零。其將於 48 小時後重設為零。

## 使用 MQTT 通訊協定刪除影子

1. 裝置或用戶端會訂閱下列 MQTT 主題：
  - `$aws/things/thingName/shadow/name/shadowName/delete/accepted`

- `$aws/things/thingName/shadow/name/shadowName/delete/rejected`
2. 裝置或用戶端會發佈具有空白訊息緩衝區的 `$aws/things/thingName/shadow/name/shadowName/delete`。
  3. 如果刪除要求有效，請刪 AWS IoT 除陰影，並在郵件內文中發佈包含 `$aws/things/thingName/shadow/name/shadowName/delete/accepted` 主題和縮寫 [/accepted response state document](#) 陰影文件的郵件。以下是接受的刪除訊息範例：

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

4. 如果更新要求無效，會 AWS IoT 發佈一則訊息，其中包含描述錯誤的 [錯誤回應文件](#) 陰影文件的 `$aws/things/thingName/shadow/name/shadowName/delete/rejected` 主題。

## 使用 REST API 刪除影子

1. 裝置或用戶端會使用空的訊息緩衝區呼叫 [DeleteThingShadow](#) API。
2. 如果要求有效，會 AWS IoT 傳回 HTTP 成功回應碼，以 [/accepted response state document](#) 及郵件內文中縮寫的 [/accepted response state document](#) 陰影文件。以下是接受的刪除訊息範例：

```
{
  "version": 4,
  "timestamp": 1591057529
}
```

3. 如果要求無效，會 AWS IoT 傳回 HTTP 錯誤回應碼 a [錯誤回應文件](#) 做為其回應訊息內文。

## Device Shadow REST API

影子會公開下列 URI 以更新狀態資訊：

```
https://account-specific-prefix-ats.iot.region.amazonaws.com/things/thingName/shadow
```

端點是特定於您的 AWS 帳戶。如要尋找您的端點，您可以：

- 使用來自 AWS CLI 的 [describe-endpoint](#) 命令。
- 使用主 AWS IoT 控制台設定。於 Settings (設定) 中，端點會列於 Custom endpoint (自訂端點) 之下

- 使用 AWS IoT 主控台物件詳細資訊頁面。在主控台中：
  1. 開啟 Manage (管理)，在 Manage (管理) 之下，選擇 Things (物件)。
  2. 於物件清單中，選擇要取得端點 URI 的物件。
  3. 選擇 Device Shadows 索引標籤，選擇您的影子。您可在 Device Shadow 詳細資訊頁面的 Device Shadow URL 區段中檢視端點 URI。

端點格式如下：

```
identifier.iot.region.amazonaws.com
```

影子 REST API 會遵循相同的 HTTPS 通訊協定/連接埠映射，如 [裝置通訊協定](#) 中所述。

#### Note

若要使用 API，您必須使用 `iotdevicegateway` 作為身分驗證的服務名稱。如需詳細資訊，請參閱 [IoT DataPlane](#)。

## API 動作

- [GetThingShadow](#)
- [UpdateThingShadow](#)
- [DeleteThingShadow](#)
- [ListNamedShadowsForThing](#)

您也可以透過提供 `name=shadowName` 作為 API 的查詢參數的一部分，使用 API 建立已命名的影子。

## GetThingShadow

取得特定物件的影子。

回應狀態文件包括 `desired` 和 `reported` 狀態之間的差量。

請求

此請求包括標準 HTTP 標頭加上以下 URI：

```
HTTP GET https://endpoint/things/thingName/shadow?name=shadowName
```

```
Request body: (none)
```

未命名 (典型) 影子不需要 name 查詢參數。

## 回應

成功時，回應會包括標準 HTTP 標頭以及下列程式碼與本文：

```
HTTP 200
Response Body: response state document
```

如需詳細資訊，請參閱[回應狀態文件範例](#)。

## 授權

擷取影子需要一項允許呼叫者可執行 `iot:GetThingShadow` 動作的政策。Device Shadow 服務接受兩種形式的身分驗證：使用 IAM 憑證的 Signature 第 4 版，或使用用戶端憑證的 TLS 交互身分驗證。

下列為允許呼叫者擷取裝置影子的政策範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:GetThingShadow",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

## UpdateThingShadow

更新指定之物件的影子。

更新僅會影響請求狀態文件中所指定的欄位。值為 `null` 的任何欄位，皆會從裝置影子中移除。

## 請求

此請求包含標準 HTTP 標頭和內文加上以下 URI 與本文：



```
HTTP POST https://endpoint/things/thingName/shadow?name=shadowName  
Request body: request state document
```

未命名 (典型) 影子不需要 name 查詢參數。

如需詳細資訊，請參閱[請求狀態文件範例](#)。

## 回應

成功時，回應會包括標準 HTTP 標頭以及下列程式碼與本文：

```
HTTP 200  
Response body: response state document
```

如需詳細資訊，請參閱[回應狀態文件範例](#)。

## 授權

更新影子需要一項允許呼叫者執行 `iot:UpdateThingShadow` 動作的政策。Device Shadow 服務接受兩種形式的身分驗證：使用 IAM 憑證的 Signature 第 4 版，或使用用戶端憑證的 TLS 交互身分驗證。

下列為允許呼叫者更新裝置影子的政策範例：

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "iot:UpdateThingShadow",  
      "Resource": [  
        "arn:aws:iot:region:account:thing/thing"  
      ]  
    }  
  ]  
}
```

## DeleteThingShadow

刪除特定物件的影子。

## 請求

此請求包括標準 HTTP 標頭加上以下 URI：

```
HTTP DELETE https://endpoint/things/thingName/shadow?name=shadowName
Request body: (none)
```

未命名 (典型) 影子不需要 name 查詢參數。

## 回應

成功時，回應會包括標準 HTTP 標頭以及下列程式碼與本文：

```
HTTP 200
Response body: Empty response state document
```

請注意，刪除影子並不會將其版本號碼重設為 0。

## 授權

刪除裝置影子需要一項允許呼叫者執行 `iot:DeleteThingShadow` 動作的政策。Device Shadow 服務接受兩種形式的身分驗證：使用 IAM 憑證的 Signature 第 4 版，或使用用戶端憑證的 TLS 交互身分驗證。

下列為允許呼叫者刪除裝置影子的政策範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:DeleteThingShadow",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

## ListNamedShadowsForThing

列出指定物件的影子。

## 請求

此請求包括標準 HTTP 標頭加上以下 URI：

```
HTTP GET /api/things/shadow/ListNamedShadowsForThing/thingName?  
nextToken=nextToken&pageSize=pageSize  
Request body: (none)
```

### nextToken

用來擷取下一組結果的符記。

此值會在分頁結果上傳回，並在傳回下一頁的呼叫中使用。

### pageSize

影子名稱在每個呼叫中傳回的數目。另請參閱 nextToken。

### thingName

要列出命名影子的物件的名稱。

## 回應

成功時，回應會包括標準 HTTP 標頭以及下列回應程式碼與 [影子名稱清單回應文件](#)。

### Note

未命名 (傳統) 影子不會出現在此清單中。如果您只有一個經典的影子，或者您指定的 thingName 不存在，則回應將是一個空清單。

```
HTTP 200  
Response body: Shadow name list document
```

## 授權

列出裝置影子需要一項允許呼叫者執行 `iot:ListNamedShadowsForThing` 動作的政策。Device Shadow 服務接受兩種形式的身分驗證：使用 IAM 憑證的 Signature 第 4 版，或使用用戶端憑證的 TLS 交互身分驗證。

以下為允許呼叫者更新物件已命名影子的範例政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:ListNamedShadowsForThing",
      "Resource": [
        "arn:aws:iot:region:account:thing/thing"
      ]
    }
  ]
}
```

## Device Shadow MQTT 主題

Device Shadow 服務會使用預留的 MQTT 主題，讓應用程式與裝置可取得、更新或刪除裝置 (影子) 的狀態資訊。

在影子主題上發佈和訂閱需要主題為基礎的授權。AWS IoT 有權新增主題到現有的主題結構。因此，我們建議您避免使用萬用字元訂閱影子主題。例如，避免訂閱類似的主題篩選器，`$aws/things/thingName/shadow/#` 因為符合此主題篩選條件的主題數量可能會隨著 AWS IoT 新的陰影主題而增加。關於發佈至這些主題的訊息範例，請參閱 [與影子互動](#)。

影子可以為已命名或未命名 (典型)。各影子所使用的主題只有在主題字首中有所不同。此表格會顯示每種影子類型所使用的主題字首。

<i>ShadowTopicPrefix</i> 值	影子類型
<code>\$aws/things/ <i>thingName</i> /shadow</code>	未命名 (經典) 影子
<code>\$aws/things/ <i>thingName</i> /shadow/name/ <i>shadowName</i></code>	已命名影子

若要建立完整的主題，請在 *ShadowTopicPrefix* 選取您要參照的影子類型、用對應值取代 *thingName* 及 *shadowName* (如適用) 的影子類型，然後將該類型附加至主題 stub，如下節所示。

下列為用來與影子互動的 MQTT 主題。

## 主題

- [/get](#)
- [/get/accepted](#)
- [/get/rejected](#)
- [/update](#)
- [/update/delta](#)
- [/update/accepted](#)
- [/update/documents](#)
- [/update/rejected](#)
- [/delete](#)
- [/delete/accepted](#)
- [/delete/rejected](#)

## /get

發佈空白訊息至此主題可取得裝置的影子：

```
ShadowTopicPrefix/get
```

AWS IoT 透過發佈至[/get/accepted](#)或來回應[/get/rejected](#)。

## 範例 政策

以下為所需政策的範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/get"
      ]
    }
  ]
}
```

```
    ]
  }
]
}
```

## /get/accepted

AWS IoT 返回設備的陰影時，將響應陰影文檔發布到此主題：

```
ShadowTopicPrefix/get/accepted
```

如需詳細資訊，請參閱 [回應狀態文件](#)。

## 範例 政策

以下為所需政策的範例：

```
{
  "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

AWS IoT 當無法返回設備的陰影時，將錯誤響應文檔發布到此主題：

```
ShadowTopicPrefix/get/rejected
```

如需詳細資訊，請參閱 [錯誤回應文件](#)。

### 範例 政策

以下為所需政策的範例：

```
{
  "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

對此主題發佈要求狀態文件，以更新裝置的影子：

```
ShadowTopicPrefix/update
```

訊息內文包含[部分請求狀態文件](#)。

嘗試更新裝置狀態的用戶端會發送一個 JSON 請求狀態文件，其 `desired` 屬性如下：

```
{
  "state": {
    "desired": {
      "color": "red",
      "power": "on"
    }
  }
}
```

更新其影子的裝置將發送一個 JSON 請求狀態文件與該 `reported` 屬性，例如：

```
{
  "state": {
    "reported": {
      "color": "red",
      "power": "on"
    }
  }
}
```

AWS IoT 透過發佈至[/update/accepted](#)或來回應[/update/rejected](#)。

## 範例 政策

以下為所需政策的範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/things/thingName/shadow/update"
      ]
    }
  ]
}
```



```
]
}
```

## /update/delta

AWS IoT 當它接受設備陰影的更改時，發布響應狀態文檔到此主題，並且請求狀態文檔包含不同的值 `desired` 和 `reported` 狀態：

```
ShadowTopicPrefix/update/delta
```

訊息緩衝區包含 [/delta response state document](#)。

### 訊息內文詳細資料

- 發佈於 `update/delta` 的訊息，僅包括 `desired` 和 `reported` 部分之間不同的所需屬性。這些屬性全都包含在其中，無論這些屬性是否原本就包含於目前的更新訊息中或是已經存放於 AWS IoT。在 `desired` 和 `reported` 部分之間沒有差異的屬性則不會包含在內。
- 如果某個屬性出現在 `reported` 部分，但在 `desired` 部分卻沒有，則此屬性不會包含在內。
- 如果某個屬性出現在 `desired` 部分，但在 `reported` 部分卻沒有，則此屬性會包含在內。
- 如果某個屬性在 `reported` 部分被刪除，但仍存在於 `desired` 部分，則此屬性會包含在內。

### 範例 政策

以下為所需政策的範例：

```
{
  "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

AWS IoT 當接受設備陰影的更改時，將響應狀態文檔發布到此主題：

```
ShadowTopicPrefix/update/accepted
```

訊息緩衝區包含 [/accepted response state document](#)。

## 範例 政策

以下為所需政策的範例：

```
{
  "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

AWS IoT 每當陰影的更新成功執行時，都會將狀態文件發佈至此主題：

```
ShadowTopicPrefix/update/documents
```

訊息內文包含 [/documents response state document](#)。

## 範例 政策

以下為所需政策的範例：

```
{
  "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

AWS IoT 當拒絕裝置陰影的變更時，會針對此主題發佈錯誤回應文件：

```
ShadowTopicPrefix/update/rejected
```

訊息內文包含 [錯誤回應文件](#)。

## 範例 政策

以下為所需政策的範例：

```
{
  "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

若要刪除裝置的影子，發佈空白訊息即可刪除主題：

```
ShadowTopicPrefix/delete
```

訊息的內容會被忽略。

請注意，刪除影子並不會將其版本號碼重設為 0。

AWS IoT 透過發佈至 [/delete/accepted](#) 或來回應 [/delete/rejected](#)。

## 範例 政策

以下為所需政策的範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic:$aws/things/thingName/shadow/delete"
      ]
    }
  ]
}
```

## /delete/accepted

AWS IoT 刪除裝置的陰影時，會將訊息發佈至此主題：

```
ShadowTopicPrefix/delete/accepted
```

## 範例 政策

以下為所需政策的範例：

```
{
  "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

AWS IoT 當無法刪除設備的陰影時，將錯誤響應文檔發佈到此主題：

```
ShadowTopicPrefix/delete/rejected
```

訊息內文包含 [錯誤回應文件](#)。

## 範例 政策

以下為所需政策的範例：

```
{
  "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 服務文件

Device Shadow 服務遵守 JSON 規格的所有規則。值、物件與陣列均存放於裝置的影子文件中。

### 目錄

- [影子文件範例](#)
- [文件屬性](#)
- [差量狀態](#)
- [版本控制影子文件](#)
- [影子文件中的用戶端字符](#)
- [空白影子文件屬性](#)
- [影子文件中的陣列值](#)

## 影子文件範例

在使用 [REST API](#) 或 [MQTT 發佈/訂閱訊息](#) 的 UPDATE、GET、DELETE 操作中，Device Shadow 服務會使用下列文件。

### 範例

- [請求狀態文件](#)

- [回應狀態文件](#)
- [錯誤回應文件](#)
- [影子名稱清單回應文件](#)

## 請求狀態文件

請求狀態文件具有以下格式：

```
{
  "state": {
    "desired": {
      "attribute1": integer2,
      "attribute2": "string2",
      ...
      "attributeN": boolean2
    },
    "reported": {
      "attribute1": integer1,
      "attribute2": "string1",
      ...
      "attributeN": boolean1
    }
  },
  "clientToken": "token",
  "version": version
}
```

- `state`：更新僅會影響所指定的欄位。一般而言，您會使用 `desired` 或 `reported` 屬性，但不會在同一個要求中使用兩者。
  - `desired`：請求在裝置中更新的狀態屬性和值。
  - `reported`：裝置回報的狀態屬性和值。
- `clientToken`：如已使用，您可透過用戶端字符來比對請求和相應的回應。
- `version`：若使用此項目，Device Shadow 服務只有在指定版本與其擁有的最新版本相符時，才會處理更新。

## 回應狀態文件

根據回應類型，回應狀態文件的格式如下。



## /accepted response state document

```
{
  "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 response state document

```
{
  "state": {
    "attribute1": integer2,
    "attribute2": "string2",
    ...
    "attributeN": boolean2
  },
  "metadata": {
    "attribute1": {
      "timestamp": timestamp
    },
  },
}
```

```

    "attribute2": {
      "timestamp": timestamp
    },
    ...
    "attributeN": {
      "timestamp": timestamp
    }
  },
  "timestamp": timestamp,
  "clientToken": "token",
  "version": version
}

```

/documents response state document

```

{
  "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"
}
```

## 回應狀態文件屬性

- `previous` : 成功更新之後，會在更新前包含物件的 `state`。
- `current` : 成功更新之後，會在更新後包含物件的 `state`。
- `state`
  - `reported` : 僅當物件於 `reported` 區段回報任何資料，且包含僅位於請求狀態文件中的欄位時才出現。
  - `desired` : 僅當裝置於 `desired` 區段回報任何資料，且包含僅位於請求狀態文件中的欄位時才出現。
  - `delta` : 僅當 `desired` 資料與影子目前 `reported` 資料不同時才會存在。
- `metadata` : 包含 `desired` 和 `reported` 部分中每個屬性的時間戳記，因此您可確定狀態的更新時間。
- `timestamp` - 響應生成的時代日期和時間。AWS IoT
- `clientToken` : 僅在發佈有效 JSON 至 `/update` 主題並使用用戶端符記時才會出現。
- `version` : AWS IoT 中目前共用之裝置影子的文件版本。此版本會在先前的文件版本編號加一。

## 錯誤回應文件

錯誤回應文件的格式如下：

```
{
  "code": error-code,
  "message": "error-message",
  "timestamp": timestamp,
  "clientToken": "token"
}
```

- `code`：用於指明錯誤類型的 HTTP 回應碼。
- `message`：可提供額外資訊的文字訊息。
- `timestamp`— 生成響應的日期和時間 AWS IoT。此屬性不會出現在所有錯誤回應文件中。
- `clientToken`：僅於已發佈的訊息中使用用戶端字符時才會出現。

如需詳細資訊，請參閱 [Device Shadow 錯誤訊息](#)。

## 影子名稱清單回應文件

影子名稱清單回應文件的格式如下：

```
{
  "results": [
    "shadowName-1",
    "shadowName-2",
    "shadowName-3",
    "shadowName-n"
  ],
  "nextToken": "nextToken",
  "timestamp": timestamp
}
```

- `results`：影子名稱的陣列。
- `nextToken`：在分頁請求中使用的字符值，以取得序列中的下一頁。當沒有更多的影子名稱返回時，這個屬性不存在。
- `timestamp`— 生成響應的日期和時間 AWS IoT。

## 文件屬性

裝置的影子文件具有下列屬性：

## state

### desired

裝置所需的狀態。應用程式可將此部分編寫入文件，在無需直接連線的狀況下更新裝置狀態。

### reported

裝置回報的狀態。裝置會在文件中編寫此部分以回報其新狀態。應用程式會讀取文件的這部分，以判斷裝置上次報告的狀態。

## metadata

存放於文件 state 部分之資料的相關資訊。其中包括針對 state 部分中各屬性的時間戳記 (Epoch 時間)，可讓您確定資料是在何時更新的。

### Note

中繼資料不會影響服務限制或定價的文件大小。如需詳細資訊，請參閱 [AWS IoT 服務限制](#)。

## timestamp

指出訊息的傳送時間 AWS IoT。藉由使用訊息中的時間戳記和 desired 或 reported 區段中的個別屬性的時間戳記，裝置可以判斷屬性的保留天數，即使裝置沒有內部時鐘也是如此。

## clientToken

裝置特有的字串能夠讓您在 MQTT 環境中將回應與請求建立關聯。

## version

文件版本。每次文件更新時，版本編號便會自動增加。如此可確保更新的文件為最新版本。

如需詳細資訊，請參閱 [影子文件範例](#)。

## 差量狀態

差量 (delta) 狀態是一種虛擬類型的狀態，包含了 desired 和 reported 之間的差異。在 desired 部分的欄位若未出現於 reported 部分，則會納入差量之中。在 reported 部分的欄位若未出現於 desired 部分，則不會納入差量之中。差量包含中繼資料，且其值與 desired 欄位裡的中繼資料相同。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
}
```

```
}  
}
```

當巢狀物件不同，差量則會包含根路徑。

```
{  
  "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 服務會重複查看 `desired` 狀態內的所有欄位，然後將其與 `reported` 狀態相比較，藉此計算出差量。

陣列的處理和值相同。如果 `desired` 部分的某個陣列與 `reported` 部分的陣列不符，則整組所需陣列都會複製到差量中。



## 版本控制影子文件

Device Shadow 服務支援每個更新訊息的版本控制，包括請求和回應。這表示，隨著影子的每次更新，JSON 文件的版本都會遞增。如此可確保兩種情況：

- 如果用戶端嘗試以較舊的版本編號覆寫影子，就會收到錯誤訊息。通知用戶端必須先重新同步裝置的影子，才可進行更新。
- 如果此訊息的版本低於存放於用戶端的訊息版本，則用戶端可以決定不採取行動。

用戶端可以透過不在影子文件中包含版本來略過版本比對。

## 影子文件中的用戶端字符

您可以使用用戶端符記收發以 MQTT 為基礎的訊息，藉此驗證請求與針對請求的回應中是否包含了同樣的用戶端符記。如此可確保回應與請求之間具有關聯。

### Note

用戶端字符不能超過 64 位元組。長於 64 位元組的用戶端字符會導致 400 (Bad Request) 回應以及一個 Invalid clientToken 錯誤訊息。

## 空白影子文件屬性

當影子文件中的 reported 和 desired 屬性不適用於目前影子狀態時，它們可以是空的，您也可以省略它們。例如，影子文件只有在具有所需狀態時才會包含 desired 屬性。以下是沒有 desired 屬性的狀態文件的有效範例：

```
{
  "reported" : { "temp": 55 }
}
```

該 reported 屬性也可以是空白，例如，如果影子尚未由裝置更新：

```
{
  "desired" : { "color" : "RED" }
}
```

如果更新導致 `desired` 或 `reported` 屬性變為 `null`，則會從文件中移除。下面顯示了如何透過將屬性設定為 `null` 來刪除 `desired`。例如，當裝置更新其狀態時，您可能會這麼做。

```
{
  "state": {
    "reported": {
      "color": "red"
    },
    "desired": null
  }
}
```

影子文件也可以具有 `desired` 或 `reported` 屬性，使影子文件變成空白。這是一個空白但有效的影子文件範例。

```
{
}
```

## 影子文件中的陣列值

影子支援陣列，但會將其視為一般值，即更新某陣列會取代整組陣列。您無法只更新陣列中的某部分。

初始狀態：

```
{
  "desired" : { "colors" : ["RED", "GREEN", "BLUE" ] }
}
```

更新：

```
{
  "desired" : { "colors" : ["RED" ] }
}
```

最終狀態：

```
{
  "desired" : { "colors" : ["RED" ] }
}
```

陣列不能有 Null 值。例如，以下陣列無效並且會被拒絕。

```
{
  "desired" : {
    "colors" : [ null, "RED", "GREEN" ]
  }
}
```

## Device Shadow 錯誤訊息

當嘗試變更狀態文件失敗時，Device Shadow 服務會透過 MQTT 發佈訊息至錯誤主題。僅當對其中一個預留 \$aws 主題的發佈請求回應時，此訊息才會進行發送。如果用戶端使用 REST API 更新文件，則會收到 HTTP 錯誤代碼作為部分回應，而 MQTT 錯誤訊息也不會發出。

HTTP 錯誤代碼	錯誤訊息
400 (錯誤的請求)	<ul style="list-style-type: none"> <li>無效的 JSON</li> <li>缺少必要的節點：狀態</li> <li>狀態節點必須是物件</li> <li>所需節點必須是物件</li> <li>報告節點必須是物件</li> <li>版本無效</li> <li>clientToken 無效</li> </ul> <div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px; margin-top: 10px;"> <p> <b>Note</b> 用戶端字符超過 64 位元組將會導致這個回應。</p> </div> <ul style="list-style-type: none"> <li>JSON 包含太多巢狀層級；最大數量為 6</li> <li>狀態包含無效的節點</li> </ul>
401 (未經授權)	<ul style="list-style-type: none"> <li>未經授權</li> </ul>
403 (禁止)	<ul style="list-style-type: none"> <li>禁止</li> </ul>
404 (未找到)	<ul style="list-style-type: none"> <li>找不到物件</li> <li>不存在具有名稱的影子：<i>shadowName</i></li> </ul>

HTTP 錯誤代碼	錯誤訊息
409 (衝突)	<ul style="list-style-type: none"><li>• 版本衝突</li></ul>
413 (承載過大)	<ul style="list-style-type: none"><li>• 承載超過允許的最大值</li></ul>
415 (不支援的媒體類型)	<ul style="list-style-type: none"><li>• 不支援的文件編碼；支援的編碼為 UTF-8</li></ul>
429 (太多請求)	<ul style="list-style-type: none"><li>• 當單一連接上的請求超過 10 則時，Device Shadow 服務就會產生此錯誤訊息。進行中的請求是已啟動但尚未完成的進行中請求。</li></ul>
500 (內部伺服器錯誤)	<ul style="list-style-type: none"><li>• 內部服務失敗</li></ul>

# 任務

使用 AWS IoT 工作可定義一組遠端操作，這些操作可以傳送到一個或多個連接到的設備上並在其上執行 AWS IoT。例如，您可以定義一個任務，指示一組裝置下載並安裝應用程式、執行韌體更新、重新啟動、輪換憑證，或者執行遠端故障排除操作。

## 存取 AWS IoT 工作

您可以使用主控台或 AWS IoT Core API 開始使用 AWS IoT 工作。

### 使用主控台

登入 AWS Management Console，然後前往 AWS IoT 主控台。在導覽窗格中，選擇 Manage (管理)，然後選擇 Jobs (任務)。您可以從此區段建立和管理任務。如果您想建立和管理任務範本，請在導覽窗格中選擇 Job templates (任務範本)。如需詳細資訊，請參閱 [使用 AWS Management Console 建立和管理任務](#)。

### 使用 API 或 CLI

您可以使用 AWS IoT Core API 操作開始使用。如需詳細資訊，請參閱 [AWS IoT API 參考](#)。SDK AWS IoT Core 支援建置 AWS IoT 工作的 AWS API。如需詳細資訊，請參閱 [AWS 開發套件與工具組](#)。

您可以使用執行指令 AWS CLI 來建立和管理工作和工作樣板。如需詳細資訊，請參閱 [AWS IoT CLI 參考](#)。

## AWS IoT 工作區域和端點

AWS IoT 作業支援特定於您的控制平面和資料平面 API 端點 AWS 區域。資料平面 API 端點專屬於您的 AWS 帳戶和 AWS 區域。如需有關「AWS IoT 工作」端點的詳細資訊，請參閱 [AWS 一般參考中的 AWS IoT Device Management - 工作資料端點](#)。

## 什麼是遠程操作？

遠端操作是您可以在實體裝置、虛擬裝置或端點上執行的任何更新或動作，而無需操作員或技術人員即可遠端完成。遠程操作是使用 over-the-air (OTA) 更新執行的，因此您的設備不必物理存在。在中管理您的裝置叢集 AWS 雲端可讓您在登錄裝置時，在裝置上執行遠端操作 AWS IoT Core。

AWS IoT Device Management Jobs 提供了一種可擴展的方法，用於在註冊的設備上執行遠端操作 AWS IoT Core。在中建立工作，AWS 雲端 並透過 MQTT 或 HTTP 通訊協定使用 OTA 更新推送至所有目標裝置。

AWS IoT Device Management 作業可讓您以安全、可擴充且更具成本效益的方式執行遠端操作，例如恢復原廠設定、裝置重新開機，以及軟體 OTA 更新。

如需詳細資訊 AWS IoT Core，請參閱[什麼是 AWS IoT ?](#)。

如需「AWS IoT Device Management 工作」的詳細資訊，請參閱[什麼是 AWS IoT 工作 ?](#)。

## 將工作用於遠端 AWS IoT Device Management 作業的好處

使用 AWS IoT Device Management 工作執行遠端作業可簡化裝置叢集的管理。下列清單強調使用「工單」執行遠端 AWS IoT Device Management 作業的一些主要優點：

- 與其他產品無縫整合 AWS 服務
  - AWS IoT Device Management 工作機會與以下增值 AWS 服務 和功能緊密整合：
    - Amazon S3：將遠端操作指示存放在安全的 Amazon S3 儲存貯體中，您可以在其中控制該內容的存取許可。使用 Amazon S3 儲存貯體可提供可擴展且耐用的儲存解決方案，以原生方式與 AWS IoT 裝置管理軟體 Package 件目錄互相整合，讓任 AWS IoT Device Management 務能夠參考更新指示並取代。如需詳細資訊，請參閱[什麼是 Amazon S3 ?](#)。
    - Amazon CloudWatch：監控和記錄每個裝置的任務執行的遠端作業實作狀態，以及其他裝置活動，以追蹤和分析任務中的整體任 AWS IoT Device Management 務效能。如需詳細資訊，請參閱[什麼是 Amazon CloudWatch ?](#) 監控作業記錄並擷取歷史資料以進行疑難排解。它是如何與工作的工作。
    - AWS IoT Device Shadow 服務：使用 AWS IoT Device Management 作業透過裝置陰影維持 AWS IoT 物件的數位表示，以便無論裝置連線為何，應用程式和其他服務都能使用裝置的狀態。如需詳細資訊，請參閱[AWS IoT Device Shadow 服務](#)。
    - 用於 AWS IoT 裝置管理的叢集中樞：建置獨立的 Web 應用程式，以監控裝置叢集的健康狀況。如需詳細資訊，請參閱[什麼是用於 AWS IoT 裝置管理的叢集中樞 ?](#)。
- 安全性最佳做法
  - 權限控制：使用 Amazon S3 控制遠端操作指示的存取許可，並確定哪些 IAM 使用者可以使用 AWS IoT 政策和 IAM 使用者角色將遠端操作指示部署到您的裝置叢集。
    - 如需 AWS IoT 策略的詳細資訊，請參閱[建立 AWS IoT 策略](#)。
    - 如需 IAM 使用者角色的詳細資訊，請參閱[身分識別與存取管理 AWS IoT](#)。
- 擴充性

- **目標工作部署**：建立工作時，使用在工作文件中輸入的特定裝置分組條件，控制哪些裝置會從具有目標工作部署的工作接收工作文件。為每個裝置建立 AWS IoT 物件並將該資訊儲存在 AWS IoT 登錄中，可讓您使用叢集索引來執行目標搜尋。您可以根據叢集索引搜尋結果建立自訂群組，以支援目標工作部署。如需詳細資訊，請參閱 [管理裝置 AWS IoT](#)。使用作業執行快照與連續作業。
- **Job 狀態**：除了每個設備上工作文件的個別實施狀態之外，還可以追蹤裝置叢集的作業文件發佈狀態，以及裝置叢集層級的整體作業狀態。如需詳細資訊，請參閱 [任務和任務執行狀態](#)。
- **新裝置可擴充性**：透過連續工作將工作文件新增至使用叢集索引建立的現有自訂群組，輕鬆將工作文件部署至新裝置。這將節省您不必分別將作業文檔部署到每個新設備的時間。或者，您可以透過將工作文件部署到預先決定的裝置群組，然後完成工作，藉此使用快照拍攝的更具針對性的方法。
- **彈性**
  - **Job 組態**：使用選擇性的工作組態推出、排程、中止、逾時和重試來自訂您的工作和工作文件，以滿足您的特定需求。如需詳細資訊，請參閱 [任務 組態](#)。
- **具成本效益**
  - **利用 AWS IoT Device Management Jobs 部署重要更新並執行例行維護工作**，引入更有效率的成本結構來維護您的裝置機群。維護裝置機群的 do-it-yourself (DIY) 解決方案包括經常性的可變成本，例如託管和管理 DIY 解決方案所需的基礎架構、開發、維護和擴展 DIY 解決方案的人力成本，以及資料傳輸成本。利用作 AWS IoT Device Management 業的透明、固定成本結構，您可以確切知道裝置的每個作業執行所需的成本，以及加速工作文件發佈至裝置叢集並追蹤每個裝置的作業執行狀態所需的資料傳輸成本。如需詳細資訊，請參閱 [AWS IoT Core 定價](#)。

## 什麼是 AWS IoT 工作？

使用 AWS IoT 工作可定義一組遠端操作，這些操作可以傳送到一個或多個連接到的設備上並在其上執行 AWS IoT。

若要建立任務，請先定義任務文件，其中包含說明裝置必須從遠端執行之操作的指示清單。若要執行這些操作，請指定目標清單，其為個別物件、[物件群組](#)，或兩者皆是。任務文件和目標一起構成部署。

每個部署都可以有其他組態：

- **推展**：此組態定義每分鐘接收任務文件的裝置數量。
- **中止**：如果有特定數量的裝置未接收到任務通知，請使用此組態取消任務。這樣可避免將錯誤的更新傳送至整個機群。
- **逾時**：如果在特定期間內未接收到任務目標的回應，則任務可能會失敗。您可以追蹤在這些裝置上執行的任務。

- **重試**：如果裝置報告失敗或工作逾時，您可以使用 AWS IoT 工作將工作文件自動重新傳送至裝置。
- **排程**：此組態可讓您排定未來日期與時間的任務。它也可讓您建立週期性維護時段，在預先定義的低流量期間更新裝置。

AWS IoT 工作會傳送訊息，通知目標有工作可用。目標會透過下載工作文件、執行指定的作業，並向其報告進度來啟動工作的執行 AWS IoT。您可以執行「工作」提供的命令，來追蹤特定目標或所有目標的 AWS IoT 工作進度。任務開始時，它的狀態為 `In progress` (進行中)。然後，裝置會在顯示此狀態的同時報告增量更新，直到任務成功、失敗或逾時為止。

以下主題介紹任務的一些關鍵概念，以及任務和任務執行的生命週期。

## 主題

- [任務的重要概念](#)
- [任務和任務執行狀態](#)

## 任務的重要概念

下列概念提供有關 AWS IoT 作業以及如何建立和部署工作以在裝置上執行遠端作業的詳細資料。

### 基本概念

以下是使用「AWS IoT 工作」時必須了解的基本概念。

### 任務

任務為傳送至連接至 AWS IoT 的一或多個裝置，並在其上執行的遠端作業。例如，您可以定義一個任務，指示一組裝置下載並安裝應用程式或執行韌體更新、重新啟動、輪換憑證，或者執行遠端故障排除操作。

### 任務文件

若要建立任務，您必須先建立任務文件，而其為裝置要執行之遠端操作的描述。

任務文件為 UTF-8 編碼的 JSON 文件，其中包含您裝置執行任務所需的資訊。任務文件包含一個或多個 URL，讓裝置可以下載更新或其他資料。任務文件可以存放於 Amazon S3 儲存貯體，或者內嵌於建立任務的命令。



**i** Tip

如需工作文件範例，請參閱的 AWS IoT SDK 中的 [jobs-agent.js](#) 範例 JavaScript。

## 目標

當您建立任務時，要指定目標的清單，也就是要執行操作的裝置。目標可以是物件或物件群組，或同時為兩者。AWS IoT obs 服務會傳送訊息給每個目標，通知它有可用的工作。

## 部署

提供任務文件並指定目標清單來建立任務後，隨後將任務文件部署至要為其執行更新的遠端目標裝置。若為快照任務，任務將在部署至目標裝置之後完成。若為持續任務，任務會在裝置新增至群組時部署到裝置群組。

## 任務執行

任務執行即目標裝置上任務的執行個體。目標會下載任務文件，藉此開始執行任務。然後它會執行文件中指定的作業，並將其進度報告給 AWS IoT。執行編號是特定目標上不會重複的任務執行之識別符。AWS IoT obs 服務提供命令，用來追蹤目標上工作執行的進度，以及所有目標間工作的進度。

## 任務類型概念

下列概念可協助您進一步瞭解可使用「工作」建立的不同 AWS IoT 工作類型。

### 快照任務

在預設情況下，當您建立任務，該任務會傳送至您指定的所有目標。這些目標完成任務之後（或者回報無法完成任務），任務即視為完成。

### 持續任務

當您建立任務，持續任務會傳送至您指定的所有目標。它會繼續執行並傳送任何新的裝置（物件）到已新增的目標群組。例如：持續任務加入群組時，可用於部署或更新裝置。您可以在建立任務時設定選用參數，藉此讓任務可以持續。

**Note**

使用動態物件群組定位 IoT 機群時，我們建議您使用連續任務而不是快照任務。使用連續任務，加入群組的裝置即使在任務建立之後，也會收到任務執行。

## 預先簽章的 URL

為了以安全、限時的方式存取任務文件以外的資料，您可以使用預先簽章的 Amazon S3 URL。將資料置於 Amazon S3 儲存貯體，並在任務作文件中的資料新增一個預留位置連結。當任 AWS IoT 務收到任務文件的請求時，會尋找預留位置連結來剖析工作文件，然後以預先簽署的 Amazon S3 URL 取代連結。

預留位置連結的格式如下：

```
{aws:iot:s3-presigned-url:https://s3.amazonaws.com/bucket/key}
```

其中，*bucket* 是您的儲存貯體名稱，而 *key* 是您所連結在儲存貯體中的物件。

在北京和寧夏區域中，僅當資源擁有者具有 ICP (網際網路內容提供者) 授權時，預先簽章的 URL 才能運作。如需詳細資訊，請參閱中國[服務入門說明文件中的 Amazon 簡易儲存 AWS 服務](#)。

## 任務組態概念

以下概念可幫助您瞭解如何設定任務。

### 推展

您可以指定何時通知目標有一項待執行的任務。這樣可以建立一個分階段推展的任務，讓您更輕易管理更新、重新啟動以及其他操作。您可以使用靜態推展率或指數推展率來建立推展組態。若要指定每分鐘通知任務目標的數量上限，請使用靜態推展率。

如需設定推展率的範例以及設定任務推展的更多資訊，請參閱 [任務推展、排程和中止組態](#)。

### 排程

任務排程可讓您針對連續和快照任務，排定將任務文件推展至目標群組中所有裝置的時間範圍。此外，您可以建立選用的維護時段，其中包含任務將任務文件推展到目標組中所有裝置的特定日期和時間。維護時段是週期性執行個體，其頻率為每日、每週、每月或在初始任務或任務範本建立期間選取的自訂日期與時間。只有連續任務才能排定在維護時段期間執行推展。

任務排程是專屬於您特定任務的功能。個別任務執行無法進行排程。如需詳細資訊，請參閱 [任務推展、排程和中止組態](#)。

## 中止

您可以建立一組條件，能夠在滿足指定條件時取消推展。如需詳細資訊，請參閱 [任務推展、排程和中止組態](#)。

## 逾時

當任務部署長時間非預期卡在 IN\_PROGRESS 狀態時，任務逾時即會通知您。計時器有兩種類型：進行中計時器和步驟計時器。當任務處於 IN\_PROGRESS 狀態時，您可以監控和追蹤任務部署的進度。

推展和中止組態與任務相關，逾時組態則與任務部署相關。如需詳細資訊，請參閱 [任務執行逾時和重試組態](#)。

## 重試

任務重試可以在任務失敗、逾時，或兩者同時發生時重試任務執行。您最多可以對任務執行嘗試 10 次重試。您可以監控和追蹤重試嘗試的進度，以及任務執行是否成功。

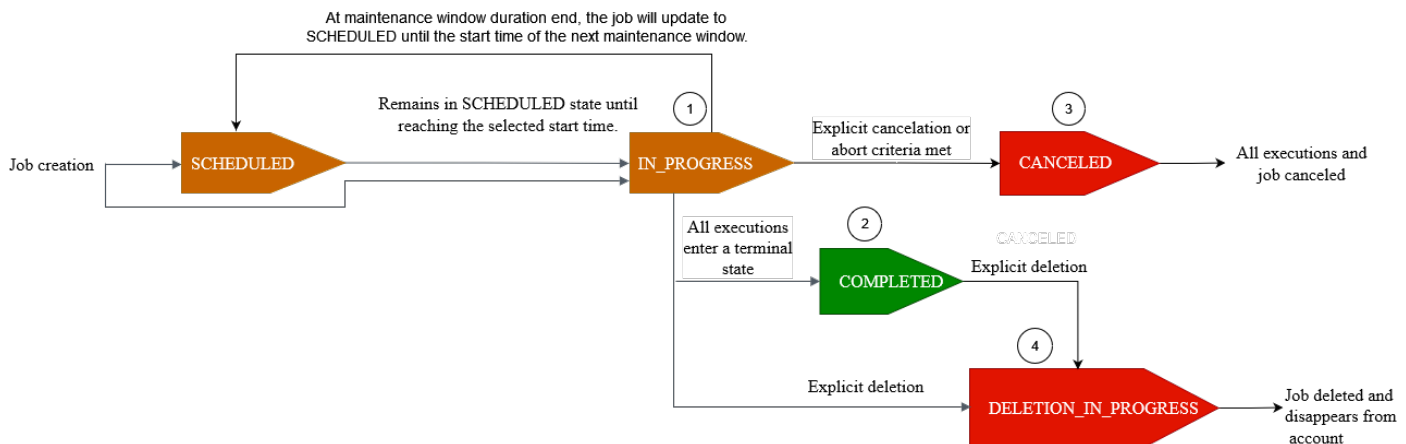
推展和中止組態與任務相關，逾時和重試組態則與任務執行相關。如需詳細資訊，請參閱 [任務執行逾時和重試組態](#)。

## 任務和任務執行狀態

下列各節說明工作的生命週期以及 AWS IoT 工作執行的生命週期。

### 任務狀態

下圖顯示工 AWS IoT 作的不同狀態。



您使用「工作」建立的 AWS IoT 工作可能處於下列其中一種狀態：

- 已排程

在使用 AWS IoT 主控台、[CreateJob](#)API 或 API 建立初始作業或[CreateJobTemplate](#)工作範本期間，您可以在 AWS IoT 主控台或 [CreateJob](#)API 或 [CreateJobTemplate](#)API 中選取選用的排程設定。SchedulingConfig 當您啟動包含特定 `startTime`、`endTime` 和 `endBehaviour` 的排程任務時，任務狀態會更新為 SCHEDULED。當任務達到您選取的 `startTime` 或下一個維護時段的 `startTime` (如果您選取了在維護時段推展任務)，狀態會從 SCHEDULED 更新為 IN\_PROGRESS，並開始將任務文件推展至目標群組中的所有裝置。

- IN\_PROGRESS

使用 AWS IoT 主控台或 [CreateJob](#)API 建立工作時，工作狀態會更新為 IN\_PROGRESS。在建立任務期間，AWS IoT 任務開始將任務執行推展到目標群組中的裝置。在所有任務執行推展完畢之後，AWS IoT 任務會等待裝置完成遠端動作。

如需套用至進行中任務的並行和限制的相關資訊，請參閱 [任務限制](#)。

**Note**

當 IN\_PROGRESS 任務到達目前維護時段結束時，任務文件的展開就會停止。任務將更新為 SCHEDULED，直到下一個維護時段的 `startTime`。

- COMPLETED (已完成)

會以下列其中一種方式處理連續任務：

- 對於未選取選用排程組態的連續任務，它會始終在進行中，持續為新增至目標群組的任何新裝置執行。它永遠不會達到 COMPLETED 的狀態。
- 對於具有選用排程組態的連續任務，則會發生下列情況：
  - 如果有提供 `endTime`，則連續任務將在超過 `endTime` 且所有任務執行都達到結束狀態時進入 COMPLETED 狀態。
  - 如果選用排程組態中未提供 `endTime`，則連續任務會繼續執行任務文件推展。

對於快照任務，任務狀態會在所有任務執行進入終端狀態 (例如 SUCCEEDED、FAILED、TIMED\_OUT、REMOVED 或 CANCELED) 時變更為 COMPLETED。

- CANCELED

當您使用 AWS IoT 主控台、[CancelJob](#) API 或取消工作時，工作狀態會變更為 CANCELED。[任務中止組態](#) 在工作取消期間，AWS IoT 工作會開始取消先前建立的工作執行。

如需套用至取消中任務的並行和限制的相關資訊，請參閱 [任務限制](#)。

- DELETION\_IN\_PROGRESS

使用 AWS IoT 主控台或 [DeleteJob](#) API 刪除工作時，工作狀態會變更為 DELETION\_IN\_PROGRESS。在工作刪除期間，AWS IoT 工作會開始刪除先前建立的工作執行項目。刪除所有工作執行後，工作會從您的 AWS 帳戶中消失。

## 任務執行狀態

下表顯示 AWS IoT 工作執行的不同狀態，以及狀態變更是由設備還是由 AWS IoT 工作啟動。

### 任務執行狀態與來源

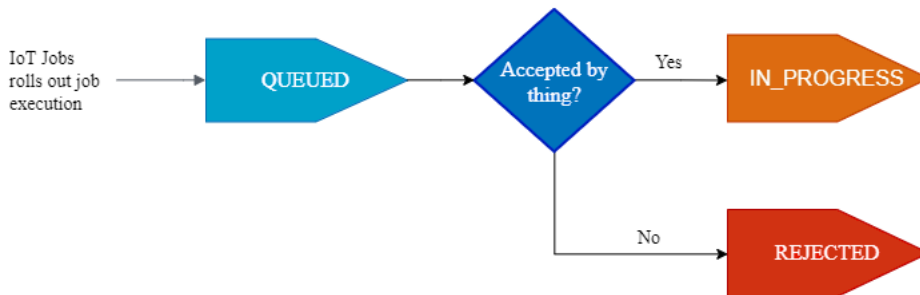
任務執行狀態	由裝置起始？	由 AWS IoT 工作啟動？	結束狀態？	可否重試？
QUEUED	否	是	否	不適用
IN_PROGRESS	是	否	否	不適用
SUCCEEDED	是	否	是	不適用
FAILED	是	否	是	是
TIMED_OUT	否	是	是	是
REJECTED	是	否	是	否
REMOVED	否	是	是	否
CANCELED	否	是	是	否

以下段落說明當您使用「工作」建立工作時，所推出的 AWS IoT 工作執行狀態的詳細資訊。

- QUEUED

當 AWS IoT 工作針對目標裝置推出工作執行時，工作執行狀態會設定為 QUEUED。任務執行會維持在 QUEUED 狀態，直到：

- 您的裝置接收任務執行，並叫用任務 API 操作，將狀態報告為 IN\_PROGRESS。
- 您取消任務或任務執行，或在符合您指定的中止條件時，狀態會變更為 CANCELED。
- 您的裝置已從目標群組中移除，且狀態變更為 REMOVED。



### • IN\_PROGRESS

如果您的 IoT 裝置訂閱了保留的 `$notify-next`，[任務主題](#) `$notify` 且您的裝置呼叫狀態為的 `StartNextPendingJobExecution` API 或 `UpdateJobExecution` API `IN_PROGRESS`，則作業會將 AWS IoT 作業執行狀態設定為 `IN_PROGRESS`

此 `UpdateJobExecution` API 可多次叫用，且狀態為 `IN_PROGRESS`。您可以使用 `statusDetails` 物件來指定關於執行步驟的其他詳細資訊。

#### **Note**

如果您為每個設備創建多個任 AWS IoT 務，則作業和 MQTT 協議不保證交付順序。

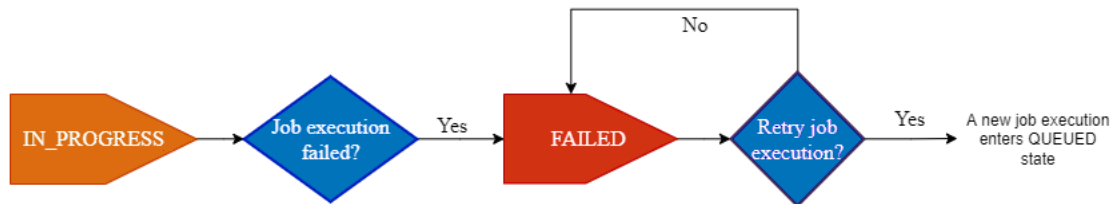
### • SUCCEEDED (成功)

當您的裝置成功完成遠端作業時，裝置必須叫用狀態 `SUCCEEDED` 為的 `UpdateJobExecution` API，以指出工作執行成功。AWS IoT 工作接著會更新，並將工作執行狀態傳回為 `SUCCEEDED`。



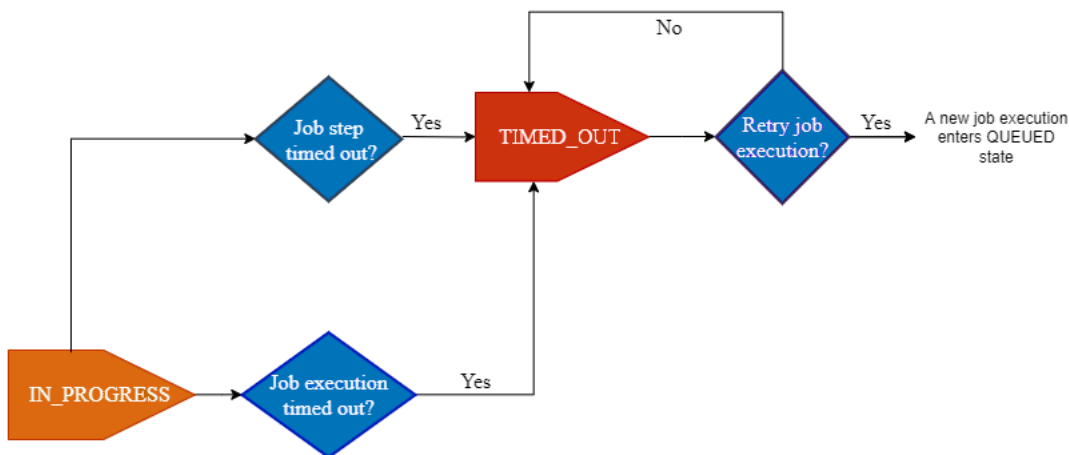
### • 失敗

當您的設備無法完成遠端操作時，設備必須調用狀態Failed為的 UpdateJobExecution API，以指示作業執行失敗。AWS IoT 工作接著會更新，並將工作執行狀態傳回為Failed。您可以使用 [任務執行重試組態](#) 未裝置重試此任務執行。



#### • TIMED\_OUT

當裝置無法完成作業步驟時，狀態為IN\_PROGRESS，或在進行中計時器的逾時持續時間內無法完成遠端作業時，AWS IoT obs 會將TIMED\_OUT作業執行狀態設定為。您也可以為進行中任務的每個任務步驟設定一個步驟計時器，且僅適用於任務執行。此進行中計時器期間使用 [任務執行逾時組態](#) 的 inProgressTimeoutInMinutes 屬性指定。您可以使用 [任務執行重試組態](#) 未裝置重試此任務執行。



#### • REJECTED

當您的裝置收到無效或不相容的要求時，裝置必須調用狀態為的 UpdateJobExecution API REJECTED。AWS IoT 工作接著會更新，並將工作執行狀態傳回為REJECTED。

#### • REMOVED

當您的裝置不再是任務執行的有效目標時，例如當它與動態物件群組分離時，AWS IoT 任務會將任務執行狀態設置為 REMOVED。您可以將物件重新附加到目標群組，然後重新啟動裝置的任務執行。

#### • CANCELED

當您使用主控台或 `CancelJobExecution` API 取消工作執行，或符合使用指定的中止準則時，AWS IoT 工作會取 [任務中止組態](#) 取消工作並將工作執行狀態設定為 `CancelJob CANCELED`。

## 管理任務

使用任務通知裝置軟體或韌體更新。您可以使用 [AWS IoT 主控台](#)、[任務管理和控制 API 操作](#)、[AWS Command Line Interface](#) 或 [AWSSDK](#) 來建立和管理任務。

## 任務的程式碼簽署

傳送程式碼至裝置時，若要讓裝置偵測程式碼在傳輸過程中是否遭到修改，我們建議您使用 AWS CLI 對程式碼檔案進行簽署。如需相關說明，請參閱 [使用 AWS CLI 建立和管理任務](#)。

如需詳細資訊，請參閱 [什麼是 AWS IoT 程式碼簽署？](#)

## 任務文件

建立任務之前，您必須建立任務文件。如果使用 AWS IoT 的程式碼簽署，您必須將任務文件上傳到已使用版本控制的 Amazon S3 儲存貯體。如需建立 Amazon S3 儲存貯體並將檔案上傳至其中的詳細資訊，請參閱《Amazon S3 入門指南》中的 [Amazon Simple Storage Service 入門](#)。

### Tip

如需任務文件範例，請參閱適用於 JavaScript 的 AWS IoT SDK 中的 [jobs-agent.js](#) 範例。

## 預先簽章的 URL

您的任務文件可以包含指向程式碼檔 (或其他檔案) 的預先簽章 Amazon S3 URL。預先簽章的 Amazon S3 URL 僅在有限的時間內有效，因此在裝置請求任務文件之後才會產生。由於建立任務文件時未建立預先簽章的 URL，所以請在任務文件中使用預留位置 URL。預留位置 URL 如下所示：

```
${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/<bucket>/<code file>}
```

其中：

- *bucket* 為含有程式碼檔案的 Amazon S3 儲存貯體。
- *code file* 為程式碼檔案的 Amazon S3 金鑰。



當裝置請求任務文件時，AWS IoT 會產生預先簽章的 URL，並以預先簽章的 URL 取代預留位置 URL。您的任務文件接著會傳送到裝置。

### IAM 角色授予從 S3 下載檔案的許可

當建立一個使用預先簽章 Amazon S3 URL 的任務時，您必須提供 IAM 角色。此角色必須授予自資料或更新儲存所在 Amazon S3 儲存貯體中下載檔案的許可。此角色也必須授與 AWS IoT 許可來擔任此角色。

您可指定預先簽章 URL 的選擇性逾時值。如需詳細資訊，請參閱 [CreateJob](#)。

### 授予 AWS IoT 任務擔任您角色的許可

1. 前往 [IAM 主控台的角色中心](#)，然後選擇您的角色。
2. 在 Trust Relationships (信任關係) 索引標籤中，選擇 Edit Trust Relationship (編輯信任關係)，並將政策文件取代為以下 JSON。選擇 Update Trust Policy (更新信任政策)。

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

3. 建議您新增全域條件內容金鑰 [aws:SourceArn](#) 和 [aws:SourceAccount](#) 至政策，保護自己免受混淆代理人問題的困擾。

#### Important

`aws:SourceArn` 必須符合以下格式：`arn:aws:iot:region:account-id:*`。確定 *region* 與 AWS IoT 區域相符，而 *account-id* 與客戶帳戶 ID 相符。如需詳細資訊，請參閱 [預防跨服務混淆代理人](#)。

```
{
  "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. 如果您的任務使用本身為 Amazon S3 物件的任務文件，請選擇許可並使用下列 JSON。這會新增一個政策，授予從 Amazon S3 儲存貯體下載檔案的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::your_S3_bucket/*"
    }
  ]
}
```

## 主題

- [使用 AWS Management Console 建立和管理任務。](#)
- [使用 AWS CLI 建立和管理任務。](#)

## 使用 AWS Management Console 建立和管理任務。

### 建立任務

1. 請登入 AWS Management Console 並登入 AWS IoT 主控台。
2. 在左側導覽窗格上，於「管理」區段下選擇遠端動作，然後選擇任務。
3. 在 Jobs (任務) 對話方塊的 Jobs (任務) 頁面上，選擇 Create job (建立任務)。
4. 您可以根據使用的裝置建立自訂任務、FreeRTOS OTA 更新任務或 AWS IoT Greengrass 任務。在此範例中，請選擇 Create a custom job (建立自訂任務)。選擇 Next (下一步)。
5. 在 Custom job properties(自訂任務屬性) 頁面的 Job properties (任務屬性) 對話方塊中，輸入下列各欄資訊：
  - Name (姓名)：輸入任務的唯一英數名稱。
  - Description - optional (說明 - 選擇性)：輸入有關任務的選擇性說明。
  - 標籤 – 選用：

#### Note

建議您不要在任務 ID 和描述中使用個人識別資訊。

選擇 Next (下一步)。

6. 在 Job targets (任務目標) 對話方塊的 File configuration (檔案組態) 頁面上，選取要執行此任務的 Things (物件) 或 Thing groups (物件群組)。

在 Job document (任務文件) 對話方塊中選取下列其中一個選項：

- From file (從檔案)：您先前上傳至 Amazon S3 儲存貯體的 JSON 任務檔案。
- 程式碼簽署

在 Amazon S3 URL 中的任務文件中，`${aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}` 必須做為預留位置，直到使用程式碼簽署設定檔將其取代為已簽署的程式碼檔案路徑為止。新簽署的程式碼檔案一開始會出現在 Amazon S3 來源儲存貯體的 SignedImages 資料夾中。系統會建立包含 Codesigned\_ 字首的新任務文件，其中包含已簽署的程式碼檔案路徑取代程式碼簽署預留位置，且放置在 Amazon S3 URL 中以建立新任務。

- 預先簽署資源 URL

在預先簽署角色下拉式清單中，選擇您在[預先簽署 URL](#)中建立的 IAM 角色。對於從 Amazon S3 下載物件的裝置，對位於 Amazon S3 中的物件使用 `${aws:iot:s3-presigned-url:預先簽署 URL}` 是最佳安全實務。

如果您想將預先簽署的 URL 用於程式碼簽署預留位置，請使用下列範例範本：

```
${aws:iot:s3-presigned-url:${aws:iot:code-sign-signature:<S3 URL>}
```

- From template (從範本)：包含任務文件和任務組態的任務範本。任務範本可以是您建立的自訂任務範本或 AWS 受管範本。

如果要建立任務來執行常用遠端動作 (如重新啟動裝置)，則可以使用 AWS 受管範本。這些範本已預先設定，可供使用。如需詳細資訊，請參閱[建立自訂任務範本](#)及[從受管範本建立自訂任務範本](#)。

7. 在 Job configuration (任務組態) 對話方塊的 Job configuration (任務組態) 頁面上，選取下列其中一種任務類型：

- 快照任務：快照任務在目標裝置和群組上完成執行後即完成。
- 連續任務：連續任務適用於物件群組，並在稍後新增至所指定目標群組的任何裝置上執行。

8. 在 Additional configurations - optional (其他組態 - 選擇性) 對話方塊中，檢閱下列選擇性任務組態，並視需要進行選擇：

- 推展組態
- Scheduling configuration (排程組態)
- 任務執行逾時組態
- Job executions retry configuration - new (任務執行重試組態 - 新增)
- 中止組態

如需有關任務組態的其他資訊，請參閱下列各節：

- [任務推展、排程和中止組態](#)
- [任務執行逾時和重試組態](#)

檢閱所有任務選項，然後選擇 Submit (提交) 以建立任務。

建立任務後，主控台會產生 JSON 簽章，並將其放置在您的任務文件中。您可以使用 [AWS IoT 主控台](#) 來檢視狀態，或取消、刪除任務。若要管理任務，請前往 [主控台的任務中心](#)。

## 使用 AWS CLI 建立和管理任務。

本節說明如何建立和管理任務。

### 建立任務

若要建立 AWS IoT 任務，請使用 `CreateJob` 命令。任務會排入所指定目標 (物件或物件群組) 的執行佇列。若要建立 AWS IoT 任務，您需要可以加入請求本文中或作為 Amazon S3 文件連結的任務文件。如果任務包含使用預先簽章的 Amazon S3 URL 下載檔案，您就需要 IAM 角色 Amazon 資源名稱 (ARN) 來得到檔案下載許可，並授予 AWS IoT 任務服務許可來擔任此角色。

如需使用 API 命令或 AWS CLI 輸入日期和時間的語法詳細資訊，請參閱 [時間戳記](#)。

### 使用任務進程式碼簽署

如果使用 AWS IoT 程式碼簽署，您必須開始程式碼簽署任務，並將輸出包含在任務文件中。這會取代任務文件中的程式碼簽署簽章預留位置，在使用程式碼簽署設定檔將其替換為已簽署程式碼檔案路徑之前，這個預留位置是必要的。程式碼簽署簽章預留位置將如下所示：

```
{aws:iot:code-sign-signature:s3://region.bucket/code-file@code-file-version-id}
```

使用 [start-signing-job](#) 命令來建立程式碼簽署任務。`start-signing-job` 會傳回任務 ID。若要取得存放簽章的 Amazon S3 位置，請使用 `describe-signing-job` 命令。然後，您可以從 Amazon S3 下載簽章。如需程式碼簽署任務的詳細資訊，請參閱 [適用於 AWS IoT 的程式碼簽署](#)。

您的任務文件必須包含程式碼檔的預先簽章 URL 預留位置，以及使用在 `start-signing-job` 命令放置在 Amazon S3 儲存貯體的 JSON 簽章輸出：

```
{
  "presign": "${aws:iot:s3-presigned-url:https://s3.region.amazonaws.com/bucket/
image}",
}
```

### 使用任務文件建立任務

下列命令顯示如何使用 Amazon S3 儲存貯體 (*jobBucket*) 中存放的任務文件 (*job-document.json*) 來建立任務，以及建立有權從 Amazon S3 下載檔案的角色 (*S3DownloadRole*)。

```
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}"
```

任務在 *thingOne* 執行。

選用的 `timeout-config` 參數，會指定每個裝置必須完成其任務執行的時間。任務執行狀態設定為 `IN_PROGRESS` 時，計時器即會開始。在時間過期之前，如果任務執行狀態未設定為其他終止狀態，就會將其設定為 `TIMED_OUT`。

進行中的計時器無法更新，並會套用到任務的所有任務執行。每當任務執行維持在 `IN_PROGRESS` 狀態的時間超過此間隔時，其就會失敗，並切換到結束 `TIMED_OUT` 狀態。AWS IoT 也會發佈 MQTT 通知。

如需建立任務推展和中止組態的詳細資訊，請參閱[任務推展和中止組態](#)。

#### Note

指定為 Amazon S3 檔案的任務文件會在您建立任務時擷取。如果您在建立任務文件後變更了作為任務文件來源的 Amazon S3 檔案的內容，則傳送到任務目標的內容不會變更。

## 更新任務

若要更新任務，請使用 `UpdateJob` 命令。您可以更新任務的 `description`、`presignedUrlConfig`、`jobExecutionsRolloutConfig`、`abortConfig` 和 `timeoutConfig` 欄位。

```
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}"
```

如需詳細資訊，請參閱[任務推展和中止組態](#)。

## 取消任務

若要取消任務，請使用 `CancelJob` 命令。取消任務會使 AWS IoT 停止為任務部署新的任務執行。它還會取消狀態為 `QUEUED` 的任何任務執行。AWS IoT 會使處於結束狀態的任務執行維持不變，因為裝置已完成任務。如果任務執行的狀態為 `IN_PROGRESS`，則該任務執行也將維持不變，除非您使用選用的 `--force` 參數。

以下命令說明如何使用 ID 010 來取消任務。

```
aws iot cancel-job --job-id 010
```

該命令會顯示下列輸出：

```
{  
  "jobArn": "string",  
  "jobId": "string",  
  "description": "string"  
}
```

當您取消任務時，狀態為 `QUEUED` 的任務執行將會遭到取消。處於 `IN_PROGRESS` 狀態的任務執行會遭到取消，前提是您指定了可選 `--force` 參數。處於結束狀態的任務執行不會取消。

**⚠ Warning**

取消狀態為 `IN_PROGRESS` (透過設定 `--force` 參數) 的任務將取消任何進行中的任務執行，並導致正在執行任務的裝置無法更新任務執行狀態。請謹慎執行，並確認每個裝置所執行的已取消任務可以復原至有效狀態。

取消的任務或其中一項任務執行的狀態具有最終一致性，AWS IoT 會盡快停止安排新的任務執行，並且不將該任務的 `QUEUED` 任務執行提供給裝置。將任務執行的狀態變更為 `CANCELED` (取消) 可能會需要一些時間，視裝置數量及其他因素而定。

如果任務由於符合 `AbortConfig` 物件定義的條件而被取消，則服務新增自動填入 `comment` 和 `reasonCode` 欄位的值。當任務取消是由使用者驅動時，您可以建立自己的 `reasonCode` 值。

## 取消任務執行

若要取消裝置上的任務執行，請使用 `CancelJobExecution` 命令。這麼做會取消狀態為 `QUEUED` 的任務執行。如果想取消進行中的任務執行，您必須使用 `--force` 參數。

以下命令說明如何從 `myThing` 上執行的任務 `010` 取消任務執行。

```
aws iot cancel-job-execution --job-id 010 --thing-name myThing
```

此命令不會顯示輸出。

系統會取消狀態為 `QUEUED` 的任務執行。處於 `IN_PROGRESS` 狀態的任務執行會遭到取消，前提是您指定了可選 `--force` 參數。處於結束狀態的任務執行無法被取消。

**⚠ Warning**

取消狀態為 `IN_PROGRESS` 的任務執行時，將導致裝置無法更新任務執行狀態。請謹慎使用並確保裝置可以還原為有效狀態。

如果任務執行處於終端狀態，或如果任務執行處於 `IN_PROGRESS` 狀態且 `--force` 參數未設定為 `true`，此命令會導致 `InvalidStateTransitionException`。

取消的任務執行之狀態最終會保持一致。將任務執行的狀態變更為 `CANCELED` 可能會需要一些時間，依不同因素而定。



## 刪除任務

若要刪除任務及其任務執行，請使用 `DeleteJob` 命令。依預設，您只能刪除處於結束狀態 (SUCCEEDED 或 CANCELED) 的任務。否則會發生例外狀況。不過，只在 `IN_PROGRESS` 參數設定為 `force` 時，您才能刪除處於 `true` 狀態的任務。

若要刪除任務，請執行下列命令：

```
aws iot delete-job --job-id 010 --force|--no-force
```

此命令不會顯示輸出。

### Warning

刪除狀態為 `IN_PROGRESS` 的任務時，正在部署任務的裝置將無法存取任務資訊或更新任務執行狀態。請謹慎執行，並確保每個裝置所部署的已刪除任務可以復原至有效狀態。

刪除任務可能會需要一些時間，根據為任務建立的任務執行數量與其他因素而定。在刪除任務期間，任務狀態會顯示為「`DELETION_IN_PROGRESS`」。如果嘗試刪除或取消狀態已為 `DELETION_IN_PROGRESS` 的任務將會導致錯誤。

只有 10 個任務可以同時具有 `DELETION_IN_PROGRESS` 狀態。否則會發生 `LimitExceededException`。

## 取得任務文件

若要從任務中擷取任務文件，請使用 `GetJobDocument` 命令。任務文件為由裝置執行遠端操作的描述。

若要取得取得任務文件，請執行下列命令：

```
aws iot get-job-document --job-id 010
```

此命令會傳回指定任務的任務文件：

```
{
  "document": "{\n\t\"operation\": \"install\",\n\t\"url\": \"http://amazon.com/firmWareUdate-01\",\n\t\"data\": \"${aws:iot:s3-presigned-url:https://s3.amazonaws.com/job-test-bucket/datafile}\"\n}"
```

```
}
```

### Note

在使用此命令擷取任務文件時，預留位置 URL 並不會被預先簽章的 Amazon S3 URL 取代。當裝置呼叫 [GetPendingJobExecutions](#) API 操作時，預留位置 URL 才會被任務文件中的預先簽章 Amazon S3 URL 取代。

## 列出任務

若要取得 AWS 帳戶中所有任務的清單，請使用 `ListJobs` 命令。任務資料和任務執行資料會保留一段**有限的時間**。執行下列命令以列出 AWS 帳戶中的所有任務：

```
aws iot list-jobs
```

此命令會傳回您帳戶中所有任務，並依任務狀態排序：

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

## 描述任務

若要取得任務的狀態，請執行 DescribeJob 命令。以下命令說明如何描述任務：

```
$ aws iot describe-job --job-id 010
```

此命令會傳回指定任務的狀態。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
}
}
```

## 列出任務的執行

任務執行物件代表在特定裝置上運作的任務。執行 `ListJobExecutionsForJob` 命令以列出任務中的所有任務執行。以下顯示如何列出任務執行：

```
aws iot list-job-executions-for-job --job-id 010
```

此命令會傳回一份任務執行清單：

```
{
  "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
    }
  }
]
```

## 列出物件的任務執行

執行 `ListJobExecutionsForThing` 命令以列出在物件上運作的所有任務執行。以下顯示如何列出物件的任務執行：

```
aws iot list-job-executions-for-thing --thing-name thingOne
```

此命令會傳回一份清單，列出特定物件上正在運作或已經運作的任務執行：

```
{
  "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"
  }
]
```

## 描述任務執行

執行 `DescribeJobExecution` 命令以取得任務執行的狀態。您必須指定任務 ID 及物件名稱，或者可選擇是否指定執行編號來識別任務執行。以下顯示如何描述任務執行：

```
aws iot describe-job-execution --job-id 017 --thing-name thingOne
```

此命令會傳回 [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"
    }
  }
}
```

## 刪除任務執行

執行 `DeleteJobExecution` 命令以刪除任務執行。必須指定任務 ID (物件名稱) 和執行編號來識別任務執行。以下顯示如何刪除任務執行：

```
aws iot delete-job-execution --job-id 017 --thing-name thingOne --execution-number
1234567890 --force|--no-force
```

此命令不會顯示輸出。

根據預設，任務執行的狀態必須為 `QUEUED` 或處於終端狀態 (`SUCCEEDED`、`FAILED`、`REJECTED`、`TIMED_OUT`、`REMOVED` 或 `CANCELED`)。否則會發生錯誤。若要刪除狀態為 `IN_PROGRESS` 的任務執行，您可以將 `force` 參數設為 `true`。

### Warning

當您刪除狀態為 `IN_PROGRESS` 的任務執行時，正在執行任務的裝置將無法存取任務資訊或更新任務執行狀態。請謹慎使用並確保裝置可以還原為有效狀態。

## 任務範本

使用任務範本預先設定任務，以便將它們部署至多組目標裝置。若要將經常執行的遠端動作 (例如重新啟動或安裝應用程式) 部署至裝置，您可以使用範本來定義標準組態。若要執行部署安全性修補程式和錯誤修正等操作，您可以從現有任務建立範本。

建立任務範本時，請指定下列其他組態和資源。

- 任務屬性
- 任務文件和目標
- 推展、排程和取消條件
- 逾時和重試條件

## 自訂和 AWS 受管理的範本

根據您要執行的遠端動作，您可以建立自訂工作範本或使用 AWS 受管理的範本。使用自訂工作範本提供您自己的自訂工作文件，並建立可重複使用的工作以部署至您的裝置。AWS 受管理的範本是由 AWS IoT 工作提供給常用動作的工作範本。這些範本具有適用於某些遠端動作的預先定義任務文件，無需您建立自己的任務文件。受管範本會協助您建立可重複使用的任務，以便更快速地啟動至裝置。

### 主題

- [使用 AWS 受管理的範本部署常見的遠端作業](#)
- [建立自訂任務範本](#)

## 使用 AWS 受管理的範本部署常見的遠端作業

AWS 受管理的範本是由提供的工作範本 AWS。它們用於經常執行的遠端動作，例如重新啟動、下載檔案或在裝置上安裝應用程式。這些範本具有適用於每個遠端動作的預先定義任務文件，無需您建立自己的任務文件。

您可以從一組預先定義的組態中進行選擇，並使用這些範本建立任務，而無需撰寫任何其他程式碼。使用受管範本，您可以檢視部署至機群的任務文件。您可以使用這些範本來建立任務，並建立可重複使用於遠端操作的自訂任務範本。

### 受管範本包含哪些內容？

每個 AWS 受管理的範本包含：



- 在任務文件中執行命令的環境。
- 指定操作名稱及其參數的任務文件。例如，如果使用 Download file (下載檔案) 範本，操作名稱為 Download file 並且參數可以是：
  - 您要下載至裝置的檔案 URL。這可以是網際網路資源，也可以是公有或預先簽章的 Amazon Simple Storage Service (Amazon S3) URL。
  - 裝置上用來存放下載檔案的本機檔案路徑。

如需有關任務文件及其參數的詳細資訊，請參閱 [受管範本遠端動作和任務文件](#)。

## 必要條件

若要讓裝置執行受管範本任務文件指定的遠端操作，必須：

- 在裝置上安裝特定軟體

使用您自己的裝置軟體和工作處理常式，或 AWS IoT 裝置用戶端。根據您的業務案例，您也可以同時執行它們，以便它們執行不同的功能。

- 使用自有裝置軟體和任務處理常式

您可以使用 AWS IoT Device SDK 及其支援遠端操作的處理常式程式庫，為裝置撰寫自己的程式碼。若要部署和執行任務，請驗證裝置代理程式程式庫是否已正確安裝，且正在裝置上執行。

您也可以選擇使用自己的處理常式來支援遠端操作。如需詳細資訊，請參閱 AWS IoT 裝置用戶端 GitHub 儲存庫中的 [範例工作處理常式](#)。

- 使用 AWS IoT 裝置用戶端

或者，您可以在裝置上安裝並執行裝 AWS IoT 置用戶端，因為它預設支援直接從主控台使用所有受管理的範本。

裝置用戶端是以 C++ 撰寫的開放原始碼軟體，可供您在內嵌 Linux 型 IoT 裝置上編譯並安裝。裝置用戶端具有基礎用戶端和分散的用戶端功能。基礎用戶端可建立透 AWS IoT 過 MQTT 通訊協定的連線能力，並可連接不同的用戶端功能。

若要在裝置上執行遠端操作，請使用裝置用戶端的用戶端任務功能。此功能包含剖析器，可接收實作任務文件中指定之遠端動作的任務文件和任務處理常式。如需有關裝置用戶端及其功能的詳細資訊，請參閱 [AWS IoT 裝置用戶端](#)。

在裝置上執行時，裝置用戶端會接收任務文件，並具有用於在文件中執行命令的平台特定實作。如需有關設定裝置用戶端和使用任務功能的詳細資訊，請參閱 [AWS IoT 教學課程](#)。

- 使用受支援的環境

針對每個受管範本，您會找到可用來執行遠端動作之環境的相關資訊。建議依照範本所述，使用支援 Linux 環境的範本。使用 AWS IoT 裝置用戶端來執行受管範本遠端動作，因為它支援常見的微處理器和 Linux 環境，例如 Debian 和 Ubuntu。

## 受管範本遠端動作和任務文件

下節列出 AWS IoT 工作的不同 AWS 受管理範本，並說明可在設備上執行的遠端動作。下節具有任務文件的相關資訊，以及每個遠端動作之任務文件參數的描述。您的裝置端軟體使用範本名稱和參數執行遠端動作。

AWS 受管理的範本會接受您在使用範本建立工作時指定值的輸入參數。所有受管範本都有兩個共同的選用輸入參數：runAsUser 和 pathToHandler。除了 AWS-Reboot 範本之外，範本需要額外的輸入參數，您必須在使用範本建立任務時為其指定值。這些所需輸入參數取決於您選擇的範本。例如，如果您選擇 AWS-Download-File 範本，您必須指定要安裝的套件清單，以及要從中下載檔案的 URL。

指定使用 AWS IoT 主控台時的輸入參數值，或指定 AWS Command Line Interface (AWS CLI) 建立使用受管理範本的工作。使用 CLI 時，請使用 document-parameters 物件提供這些值。如需詳細資訊，請參閱 [documentParameters](#)。

### Note

只在從 AWS 受管範本建立任務時，才會使用 document-parameters。這個參數不能與自訂任務範本一起使用，也不能用於從自訂任務範本建立任務。

以下為常見選用輸入參數的敘述。您會在下節中看到每個受管範本所需之其他輸入參數的描述。

### runAsUser

此參數指定是否以其他使用者身分執行任務處理常式。如果未在建立任務期間指定，則任務處理常式會以與裝置用戶端相同的使用者身分執行。在以其他使用者身分執行任務處理常式時，請指定不超過 256 個字元的字串值。

## pathToHandler

此為在裝置上執行的任務處理常式的路徑。如果未在任務建立期間指定，則裝置用戶端會使用目前的工作目錄。

下文顯示不同的遠端動作、其任務文件及其接受的參數。這些範本都支援 Linux 環境，以便在裝置上執行遠端操作。

### AWS-Download-File

#### 範本名稱

#### AWS-Download-File

#### 範本描述

由下載檔案所提供 AWS 的受管理範本。

#### 輸入參數

此範本具有下列必要參數。您也可以指定選用參數 `runAsUser` 和 `pathToHandler`。

#### downloadUrl

要從中下載檔案的 URL。這可以是網際網路資源、Amazon S3 中可以公開存取的物件，或者 Amazon S3 中只能由您裝置使用預先簽章 URL 存取的物件。如需有關使用預先簽章的 URL 和授予許可的詳細資訊，請參閱 [預先簽章的 URL](#)。

#### filePath

本機檔案路徑，其中會顯示裝置中存放下載檔案的位置。

#### 裝置行為

裝置會從指定的位置下載檔案，確認下載已完成，並將其存放在本機。

#### 任務文件

下文顯示任務文件及其最新版本。範本會顯示任務處理常式和 Shell 指令碼的路徑 (`download-file.sh`)；必須執行任務處理常式，才能下載檔案。並說明必要參數 `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}"
    }
  }
]
```

## AWS-Install-Application

### 範本名稱

### AWS-Install-Application

### 範本描述

由安裝一或多個應用程式所提供 AWS 的受管理範本。

### 輸入參數

此範本具有下列必要參數 `packages`。您也可以指定選用參數 `runAsUser` 和 `pathToHandler`。

### `packages`

此為要安裝的一或多個應用程式的空格分隔清單。

### 裝置行為

裝置會依照任務文件所述安裝應用程式。

### 任務文件

下文顯示任務文件及其最新版本。範本會顯示任務處理常式和 Shell 指令碼的路徑 (install-packages.sh)；必須執行任務處理常式，才能下載檔案。其也會展示必要參數 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}"
      }
    }
  ]
}
```

## AWS-Reboot

### 範本名稱

### AWS-Reboot

### 範本描述

由提供的受管理範本，用 AWS 於重新啟動裝置。

### 輸入參數

此範本沒有必要參數。您可以指定選用參數 runAsUser 和 pathToHandler。

### 裝置行為

裝置重新啟動成功。

### 任務文件

下文顯示任務文件及其最新版本。範本會顯示任務處理常式和 Shell 指令碼的路徑 (reboot.sh)；必須執行任務處理常式，才能重新啟動裝置。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

## AWS-Remove-Application

### 範本名稱

### AWS-Remove-Application

### 範本描述

由解除安裝一或多個應用程式所提供 AWS 的受管理範本。

### 輸入參數

此範本具有下列必要參數 packages。您也可以指定選用參數 runAsUser 和 pathToHandler。

### packages

此為要解除安裝的一或多個應用程式的空格分隔清單。

### 裝置行為

裝置會依照任務文件所述解除安裝應用程式。

### 任務文件

下文顯示任務文件及其最新版本。範本會顯示任務處理常式和 Shell 指令碼的路徑 (remove-packages.sh)；必須執行任務處理常式，才能下載檔案。其也會展示必要參數 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}"
      }
    }
  ]
}
```

## AWS-Restart-Application

### 範本名稱

### AWS-Restart-Application

### 範本描述

由提供 AWS 用於停止和重新啟動一或多個服務的受管理範本。

### 輸入參數

此範本具有下列必要參數 services。您也可以指定選用參數 runAsUser 和 pathToHandler。

### 服務

此為要重新啟動的一或多個應用程式的空格分隔清單。

### 裝置行為

指定應用程式會停止，然後在裝置上重新啟動。

## 任務文件

下文顯示任務文件及其最新版本。範本會顯示任務處理常式和 Shell 指令碼的路徑 (restart-services.sh)；必須執行任務處理常式，才能重新啟動系統服務。其也會展示必要參數 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}"
      }
    }
  ]
}
```

## AWS-Start-Application

### 範本名稱

AWS-Start-Application

### 範本描述

由提供 AWS 用於啟動一或多個服務的受管理範本。

### 輸入參數

此範本具有下列必要參數 services。您也可以指定選用參數 runAsUser 和 pathToHandler。

services



此為要啟動的一或多個應用程式的空格分隔清單。

## 裝置行為

指定應用程式會在裝置上開始執行。

## 任務文件

下文顯示任務文件及其最新版本。範本會顯示任務處理常式和 Shell 指令碼的路徑 (start-services.sh)；必須執行任務處理常式，才能啟動系統服務。其也會展示必要參數 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}"
      }
    }
  ]
}
```

## AWS-Stop-Application

### 範本名稱

### AWS-Stop-Application

### 範本描述

由停止一或多個服務所提供 AWS 的受管理範本。

### 輸入參數

此範本具有下列必要參數 `services`。您也可以指定選用參數 `runAsUser` 和 `pathToHandler`。

## services

此為要停止的一或多個應用程式的空格分隔清單。

## 裝置行為

指定應用程式會在裝置上停止執行。

## 任務文件

下文顯示任務文件及其最新版本。範本會顯示任務處理常式和 Shell 指令碼的路徑 (`stop-services.sh`)；必須執行任務處理常式，才能停止系統服務。其也會展示必要參數 `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}"
      }
    }
  ]
}
```

## AWS-Run-Command

### 範本名稱

### AWS-Run-Command

### 範本描述

由執行 shell 命令所提供 AWS 的受管理範本。

## 輸入參數

此範本具有下列必要參數 `command`。您也可以指定選用參數 `runAsUser`。

## command

以逗號分隔的命令字串。命令本身中包含的任何逗號都必須逸出。

## 裝置行為

裝置會依照任務文件指示執行 Shell 命令。

## 任務文件

下文顯示任務文件及其最新版本。範本會顯示任務命令路徑，以及您所提供將由裝置執行的命令。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Run-Command",
        "type": "runCommand",
        "input": {
          "command": "${aws:iot:parameter:command}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

## 主題

- [使用從 AWS 受管理的範本建立工作 AWS Management Console](#)
- [使用從 AWS 受管理的範本建立工作 AWS CLI](#)

## 使用從 AWS 受管理的範本建立工作 AWS Management Console

使用取 AWS Management Console 得有關 AWS 受管理範本的資訊，並使用這些範本建立工作。然後可以將建立的任務儲存為自己的自訂範本。

## 取得受管範本的詳細資訊

您可以從主控台取得可使用之不同受管理範本的相關資 AWS IoT 訊。

1. 若要查看可用的受管理範本，請移至 [AWS IoT 主控台的 \[Job 範本\] 中心](#)，然後選擇 [受管理的範本] 索引標籤。
2. 若要檢視詳細資訊，請選擇受管範本。

詳細資訊頁面包含以下資訊：

- 受管範本的名稱、說明和 Amazon 資源名稱 (ARN)。
- 可執行遠端操作的環境，如 Linux。
- JSON 任務文件，指定任務處理常式的路徑以及要在裝置上執行的命令。例如，下文顯示 AWS-  
Reboot 範本的範例任務文件。範本會顯示任務處理常式和 Shell 指令碼的路徑 (reboot.sh)；必須執行任務處理常式，才能重新啟動裝置。

```
{
  "version": "1.0",
  "steps": [
    {
      "action": {
        "name": "Reboot",
        "type": "runHandler",
        "input": {
          "handler": "reboot.sh",
          "path": "${aws:iot:parameter:pathToHandler}"
        },
        "runAsUser": "${aws:iot:parameter:runAsUser}"
      }
    }
  ]
}
```

如需有關任務文件以及各種遠端操作參數的詳細資訊，請參閱 [受管範本遠端動作和任務文件](#)。

- 任務文件的最新版本。

## 使用受管範本建立任務

您可以使用 AWS 管理主控台選擇要用來建立工作的 AWS 受管理範本。本節將告訴您如何做到。

您也可以啟動工作建立工作流程，然後選擇建立工作時要使用的 AWS 受管理範本。如需有關此工作流程的詳細資訊，請參閱 [使用 AWS Management Console 建立和管理任務](#)。

## 1. 選擇您的 AWS 受管理範本

移至 [AWS IoT 主控台的 \[Job 範本\] 中心](#)，選擇 [受管理的範本] 索引標籤，然後選擇您的範本。

## 2. 使用受管範本建立任務

### 1. 在範本的詳細資訊頁面中，選擇 Create job (建立任務)。

主控台會切換至已新增範本組態的 Create job (建立任務) 工作流程的 Custom job properties (自訂任務屬性) 步驟。

### 2. 輸入唯一的英數字元任務名稱，以及選用說明和標籤，然後選擇 Next (下一步)。

### 3. 選擇要在此任務中執行的物件或物件群組作為任務目標。

### 4. 在 Job document (任務文件) 區段中，範本會顯示其組態設定和輸入參數。輸入所選範本的輸入參數值。例如，如果您選擇 AWS-Download-File 範本：

- downloadUrl 中，輸入要下載的檔案 URL，例如：`https://example.com/index.html`。
- filePath 中，輸入裝置上存放下載檔案的路徑，例如：`path/to/file`。

您也可以選擇輸入參數 runAsUser 和 pathToHandler 的值。如需有關各個範本輸入參數的詳細資訊，請參閱 [受管範本遠端動作和任務文件](#)。

### 5. 在 Job configuration (任務組態) 頁面上，選擇任務類型為連續或快照任務。快照任務在目標裝置和群組上完成執行後即完成。連續任務適用於物件群組，並在新增至所指定目標群組的任何裝置上執行。

### 6. 繼續為任務新增任何其他組態，然後檢閱並建立任務。如需其他組態的資訊，請參閱：

- [任務推展、排程和中止組態](#)
- [任務執行逾時和重試組態](#)

## 從受管範本建立自訂任務範本

您可以使用 AWS 受管理的範本和自訂工作作為建立自訂工作範本的起點。若要建立自訂工作範本，請先依照上一節所述，從 AWS 受管理的範本建立工作。

然後將自訂任務另存為範本，建立您的自訂任務範本。若要另存為範本：

### 1. 移至 [AWS IoT 主控台的「Job 中心」](#)，然後選擇包含受管理範本的工作。

2. 選擇 Save as a job template (另存為任務範本)，然後建立自訂任務範本。如需有關建立自訂任務範本的詳細資訊，請參閱 [從現有任務建立任務範本](#)。

## 使用從 AWS 受管理的範本建立工作 AWS CLI

使用取 AWS CLI 得有關 AWS 受管理範本的資訊，並使用這些範本建立工作。您便可以將任務另存為範本，然後建立自己的自訂範本。

### 列出受管範本

此[list-managed-job-templates](#) AWS CLI 指令會列出您的 AWS 帳戶。

```
aws iot list-managed-job-templates
```

依預設，執行此命令會顯示所有可用的 AWS 受管理範本及其詳細資料。

```
{
  "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"
    }
]
}

```

如需詳細資訊，請參閱[ListManagedJobTemplates](#)。

### 取得受管範本的詳細資訊

該[describe-managed-job-template](#) AWS CLI 命令獲取有關指定作業模板的詳細信息。指定任務範本名稱和選用範本版本。若未指定範本版本，則會傳回預先定義的預設版本。以下說明一個範例，執行命令以取得有關 `AWS-Download-File` 範本的詳細資訊。

```

aws iot describe-managed-job-template \
    --template-name AWS-Download-File

```

這個命令顯示範本詳細資訊和 ARN、其任務文件以及 `documentParameters` 參數，該參數為範本輸入參數的鍵值對清單。如需有關不同範本和輸入參數的詳細資訊，請參閱 [受管範本遠端動作和任務文件](#)。

**Note**

使用此 API 時傳回的 `documentParameters` 物件只能在從 AWS 受管理範本建立工作時使用。這個物件不能用於自訂任務範本。如需示範如何使用這個參數的範例，請參閱 [使用受管範本建立任務](#)。

```
{
  "templateName": "AWS-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}\"}}]}\"
}

```

如需詳細資訊，請參閱[DescribeManagedJobTemplate](#)。

### 使用受管範本建立任務

此指[create-job](#) AWS CLI 令可用來從工作樣板建立工作。這會將名為 `thingOne` 的裝置設為目標，並指定受管範本的 Amazon 資源名稱 (ARN)，以用作任務的基礎。您可以覆寫進階組態，例如逾時和取消組態，方法是傳遞 `create-job` 命令的相關聯參數。

這個範例說明如何使用 `AWS-Download-File` 範本建立任務。它也說明了如何使用 `document-parameters` 參數指定範本的輸入參數。

#### Note

只能將 `document-parameters` 物件與 AWS 受管理的範本搭配使用。這個對象不能與自訂任務範本搭配使用。

```

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：

- `##` 是 AWS 區域。
- `account-id` 是唯一的 AWS 帳戶數字。

- *thingOne* 是任務所針對的 IoT 物件名稱。
- *AWS-Download-File:1.0* 是受管範本的名稱。
- <https://example.com/index.html> 是要下載檔案的 URL。
- <https://pathto/file/index> 是裝置上用來存放下載檔案的路徑。

執行以下命令來建立範本 *AWS-Download-File* 的任務。

```
{
  "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."
}
```

從受管範本建立自訂任務範本

1. 使用受管範本建立任務，如前一節中所述。
2. 使用您建立的任務的 ARN 建立自訂任務範本。如需詳細資訊，請參閱 [從現有任務建立任務範本](#)。

## 建立自訂任務範本

您可以使用 AWS CLI 和 AWS IoT 主控台建立工作樣板。您也可以使用 AWS IoT 裝置管理 Web 應用程式的 AWS CLI AWS IoT 主控台和叢集中樞，從作業範本建立作業。如需有關在 Fleet Hub 應用程式中使用作業範本的詳細資訊，請參閱 [在叢集中樞中使用作業範本進行 AWS IoT 裝置管理](#)。

### Note

任務文件中替代模式總數應小於或等於 10。

### 主題

- [使用 AWS Management Console 建立自訂任務範本](#)
- [使用 AWS CLI 建立自訂任務範本](#)

## 使用 AWS Management Console 建立自訂任務範本

本主題說明如何使用主控台建立、刪除及檢視工作樣板的詳細資訊。AWS IoT 訊。

## 建立自訂任務範本

可以建立原始自訂任務範本或從現有任務建立任務範本。您也可以從使用 AWS 受管理範本建立的現有工作建立自訂工作範本。如需詳細資訊，請參閱 [從受管範本建立自訂任務範本](#)。

### 建立原始任務範本

#### 1. 開始建立任務範本

1. 移至 [AWS IoT 主控台的 \[Job 範本\] 中心](#)，然後選擇 [自訂範本] 索引標籤。
2. 選擇 Create job template (建立任務範本)。

#### Note

您也可以從 Fleet Hub (機群中樞) 下的 Related services (相關服務) 頁面導覽至 Job templates (任務範本) 頁面。

#### 2. 指定任務範本屬性

在 Create job template (建立任務範本) 頁面中，輸入任務名稱的英數字元識別碼和英數字元說明，提供範本的其他詳細資訊。

#### Note

不建議在任務 ID 或描述中使用個人識別資訊。

#### 3. 提供任務文件

提供存放在 S3 儲存貯體中或作為任務內指定的內嵌任務文件的 JSON 任務檔案。使用此範本建立任務時，此任務檔案會成為任務文件。

若任務檔案存放於 S3 儲存貯體中，請輸入 S3 URL 或選擇 Browse S3 (瀏覽 S3)，然後導覽至您的任務文件並加以選取。

#### Note

您只能選取目前區域中的 S3 儲存貯體。

4. 繼續為任務新增任何其他組態，然後檢閱並建立任務。如需其他選用組態的詳細資訊，請參閱下列項目：

- [任務推展、排程和中止組態](#)
- [任務執行逾時和重試組態](#)

從現有任務建立任務範本

1. 選擇您的任務。

1. 移至 [AWS IoT 主控台的「Job 中心」](#)，然後選擇您要用來做為工作範本基礎的工作。
2. 選擇 Save as a job template (另存為任務範本)。

**Note**

也可以選擇不同的任務文件或編輯原始任務中的進階組態，然後選擇 Create job template (建立任務範本)。您的新任務範本會出現在 Job templates (任務範本) 頁面上。

2. 指定任務範本屬性

在 Create job template (建立任務範本) 頁面中，輸入任務名稱的英數字元識別碼和英數字元說明，提供範本的其他詳細資訊。

**Note**

任務文件是您在建立範本時指定的任務檔案。如果在任務內而非 S3 位置指定任務文件，則可在此任務的詳細資訊頁面中看到任務文件。

3. 繼續為任務新增任何其他組態，然後檢閱並建立任務。如需其他組態的資訊，請參閱：

- [任務推展、排程和中止組態](#)
- [任務執行逾時和重試組態](#)

從自訂任務範本建立任務

您可以前往任務範本的詳細資訊頁面，從自訂任務範本建立任務，如本主題所述。您也可以建立任務，或透過選擇執行任務建立工作流程時要使用的任務範本來建立任務。如需詳細資訊，請參閱 [使用 AWS Management Console 建立和管理任務](#)。

本主題顯示如何從自訂任務範本的詳細資訊頁面建立任務。您也可以從 AWS 受管理的範本建立工作。如需詳細資訊，請參閱 [使用受管範本建立任務](#)。

## 1. 選擇自訂任務範本

移至 [AWS IoT 主控台的 \[Job 範本\] 中心](#)，然後選擇 [自訂範本] 索引標籤，然後選擇您的範本。

## 2. 使用自訂範本建立任務

若要建立任務：

1. 在範本的詳細資訊頁面中，選擇 Create job (建立任務)。

主控台會切換至已新增範本組態的 Create job (建立任務) 工作流程的 Custom job properties (自訂任務屬性) 步驟。

2. 輸入唯一的英數字元任務名稱，以及選用說明和標籤，然後選擇 Next (下一步)。
3. 選擇要在此任務中執行的物件或物件群組作為任務目標。

在 Job document (任務文件) 區段中，範本會顯示其組態設定。如果您想要使用不同的任務文件，請選擇 Browse (瀏覽)，然後選取不同的儲存貯體和文件。選擇 Next (下一步)。

4. 在 Job configuration (任務組態) 頁面上，選擇任務類型為連續或快照任務。快照任務在目標裝置和群組上完成執行後即完成。連續任務適用於物件群組，並在新增至所指定目標群組的任何裝置上執行。
5. 繼續為任務新增任何其他組態，然後檢閱並建立任務。如需其他組態的資訊，請參閱：
  - [任務推展、排程和中止組態](#)
  - [任務執行逾時和重試組態](#)

### Note

當從任務範本建立的任務更新任務範本提供的現有參數時，這些更新的參數將會覆寫該任務之任務範本所提供的現有參數。

您也可以使用 Fleet Hub Web 應用程式，從任務範本建立任務。如需在叢集中樞中建立工作的相關資訊，請參閱 [在叢集中樞中使用作業範本以進行 AWS IoT 裝置管理](#)。

## 刪除任務範本

若要刪除 Job 範本，請先移至 [AWS IoT 主控台的 \[工作範本\] 中心](#)，然後選擇 [自訂範本] 索引標籤。然後，選擇要刪除的任務範本，並選擇刪除。

### Note

此為永久性刪除，且任務範本不會再顯示在 Custom templates (自訂範本) 索引標籤上。

## 使用 AWS CLI 建立自訂任務範本

本主題說明如何使用 AWS CLI 建立、刪除和擷取任務範本的詳細資訊。

### 從頭建立任務範本

```
## AWS CLI ##### S3 ##### (##-## .json) ##### Amazon S3 (S3) ###  
#####DownloadRole
```

```
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}"
```

選用的 `timeout-config` 參數會指定每個裝置必須完成任務執行的時間量。任務執行狀態設定為 `IN_PROGRESS` 時，計時器即會開始。在時間過期之前，如果任務執行狀態未設定為其他終止狀態，就會將其設定為 `TIMED_OUT`。

進行中的計時器無法更新，並會套用到任務的所有任務啟動。每當工作啟動保持狀 `IN_PROGRESS` 態超過此間隔時間，工作啟動就會失敗，並切換到終端機 `TIMED_OUT` 狀態。AWS IoT 同時也會發佈 MQTT 通知。

如需有關建立任務推展和中止組態的詳細資訊，請參閱[任務推展和中止組態](#)。

#### Note

指定為 Amazon S3 檔案的任務文件會在您建立任務時擷取。如果您在建立任務後變更作為任務文件來源的 Amazon S3 檔案的內容，則傳送到任務目標的內容不會變更。

### 從現有任務建立任務範本

下列 AWS CLI 命令透過指定現有任務的 Amazon 資源名稱 (ARN) 來建立任務範本。新的任務範本會使用任務中指定的所有組態。您可以選擇性地使用任何選用參數，來變更現有任務中的任何組態。

```
aws iot create-job-template \  
  --job-arn arn:aws:iot:region:123456789012:job/job-name \  
  --timeout-config inProgressTimeoutInMinutes=100
```

### 取得任務範本的詳細資訊

下列 AWS CLI 命令會取得有關指定工作樣板的詳細資料。

```
aws iot describe-job-template \  
  --job-template-id template-id
```

該命令會顯示下列輸出。

```
{  
  "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
  }
}
```

## 列出任務範本

下列 AWS CLI 命令會列出您的所有工作樣板 AWS 帳戶。

```
aws iot list-job-templates
```

該命令會顯示下列輸出。

```
{
  "jobTemplates": [
    {
```



```
    "createdAt": number,  
    "description": "string",  
    "jobTemplateArn": "string",  
    "jobTemplateId": "string"  
  }  
],  
"nextToken": "string"  
}
```

若要擷取其他結果頁面，請使用 `nextToken` 欄位的值。

## 刪除任務範本

下列 AWS CLI 命令會刪除指定的工作樣板。

```
aws iot delete-job-template \  
  --job-template-id template-id
```

此命令不會顯示輸出。

## 從自訂任務範本建立任務

下列 AWS CLI 指令會從自訂工作範本建立工作。它會將名為 `thingOne` 的裝置設為目標，並指定任務範本的 Amazon 資源名稱 (ARN)，以用作任務的基礎。您可以覆寫進階組態，例如逾時和取消組態，方法是傳遞 `create-job` 命令的相關聯參數。

### Warning

從 AWS 受管範本建立任務時，`document-parameters` 物件只能與 `create-job` 命令搭配使用。這個對象不能與自訂任務範本搭配使用。如需示範如何使用這個參數建立任務的範例，請參閱 [使用受管範本建立任務](#)。

```
aws iot create-job \  
  --targets arn:aws:iot:region:123456789012:thing/thingOne \  
  --job-template-arn arn:aws:iot:region:123456789012:jobtemplate/template-id
```

## 任務組態

對於部署至指定目標的每個任務，您可以擁有下列其他組態。

- 推展：定義每分鐘接收任務文件的裝置數量。
- 排程：除了使用週期性維護時段之外，還排程在未來日期和時間執行任務。
- 中止：在某些裝置未接收到任務通知或您的裝置報告其任務執行失敗等情況下取消任務。
- 逾時：如果在其任務執行開始後的特定期間內未接收到任務目標的回應，則任務可能會失敗。
- 重試：如果您的裝置在嘗試完成任務執行時報告失敗，或如果任務執行逾時，重試任務執行。

透過使用這些組態，您可以監控任務執行的狀態，並避免將錯誤的更新傳送至整個機群。

### 主題

- [任務組態的運作方式](#)
- [指定其他組態](#)

## 任務組態的運作方式

在部署任務時使用推展和中止組態，在任務執行時則使用逾時和重試組態。下列各節說明有關這些組態運作方式的詳細資訊。

### 主題

- [任務推展、排程和中止組態](#)
- [任務執行逾時和重試組態](#)

## 任務推展、排程和中止組態

您可以使用任務推展、排程和中止組態，來定義接收任務文件的裝置數量、排程任務推展，以及決定取消任務的條件。

### 任務推展組態

您可以指定何時通知目標有一項待執行的任務。您亦可建立一個分階段推展的任務，來管理更新、重新啟動以及其他操作。若要指定通知目標的方式，請使用任務推展率。

## 任務推展率

您可以使用恆定推展率或指數推展率來建立推展組態。若要指定每分鐘通知任務目標的數量上限，請使用恆定推展率。

AWS IoT 工作可以使用指數發佈率部署，因為符合各種準則和閾值。如果失敗任務數量與您指定的一組條件相符，則可取消任務推展。您可以使用 [JobExecutionsRolloutConfig](#) 物件，在建立任務時設定任務推展率條件。您還可以使用 [AbortConfig](#) 物件在建立任務時設定任務中止條件。

下列範例說明推展率的運作方式。例如，基本速率為每分鐘 50 個、增量因數為 2、已通知和成功裝置的數量各為 1000 的任務推展率，將依以下方式運作：任務將以每分鐘 50 個任務執行的速率開始，並以該速率繼續，直到 1000 個物件均收到任務執行通知或發生 1000 次成功的任務執行。

下表說明推展如何在前四個增量中繼續進行。

每分鐘推展率	50	100	200	400
滿足速率提高的已通知裝置或成功執行任務的數量	1,000	2,000	3,000	4,000

### Note

如果最大並行任務限制為 500 個任務 (`isConcurrent = True`)，則所有作用中的任務將保持狀態 IN-PROGRESS 且不會推展任何新的任務執行作業，直到並行任務數量達到 499 以下 (`isConcurrent = False`)。此原則適用於連續任務和快照任務。

如果 `isConcurrent = True`，表示此任務目前正在將任務執行推展到目標群組中的所有裝置。如果 `isConcurrent = False`，任務已完成將所有任務執行推展至目標群組中的所有裝置。它會在目標群組中的所有裝置一旦達到終端狀態或目標群組的臨界值百分比後，就會更新其狀態 (如果您選取了任務中止組態)。 `isConcurrent = True` 與 `isConcurrent = False` 的任務層級狀態皆為 IN\_PROGRESS。

如需作用中任務與並行任務限制的詳細資訊，請參閱 [作用中和並行任務限制](#)。

## 使用動態物件群組進行連續任務的任務推展率

當您使用連續工作在叢集上部署遠端操作時，AWS IoT 作業會針對目標物件群組中的裝置推出工作執行。對於新增至動態物件群組中的新裝置，這些任務執行會繼續推展到那些裝置，即使建立了任務之後也是如此。

建立任務之前，推展組態只能控制加入群組之裝置的推展率。建立任務後，對於任何新裝置，只要裝置加入目標群組，就會以近乎即時的速度建立任務執行。

## 任務排程組態

您可以使用預定的開始時間、結束時間以及各項任務執行作業達到結束時間後所應發生的結束行為，藉此預先排定最多 1 年以後的連續或快照任務。此外，您還可以為連續任務建立具有彈性頻率、開始時間和持續時間的選用週期性維護時段，以便將任務文件推展到目標群組中的所有裝置。

## 任務排程組態

### 開始時間

排程任務的開始時間是任務開始向目標群組中所有裝置推展任務文件的未來日期和時間。排程任務的開始時間會套用至連續任務和快照任務。最初建立排程任務時，它會維持為 SCHEDULED 的狀態。到達您選取的 `startTime` 後，它會更新為 IN\_PROGRESS 並開始任務文件推展。`startTime` 必須少於或等於從您建立排程任務初始日期和時間起算的一年。

如需有關使用 API 命令或的語法的詳細資訊 AWS CLI，請參閱 [startTime 時間戳記](#)。

對於在遵守日光節約時間 (DST) 的位置中的週期性維護時段期間進行選用排程組態的任務，從 DST 切換到標準時間以及從標準時間切換到 DST 時，時間會變動一小時。

### Note

顯示在中的時區 AWS Management Console 是您目前的系統時區。但是，這些時區將在系統中轉換為 UTC。

### 結束時間

排程任務的結束時間是任務停止向目標群組中所有裝置推展任務文件的未來日期和時間。排程任務的結束時間會套用至連續任務和快照任務。在排程任務達到選取的 `endTime` 且所有任務執行都達到了終端狀態之後，它會將其狀態從 IN\_PROGRESS 更新為 COMPLETED。`endTime` 必須少於或等於從您建立排程任務初始日期和時間起算的二年。`startTime` 與 `endTime` 之間的最短持續時間為 30 分鐘。繼續嘗試任務執行重試，直到任務達到 `endTime` 為止，然後 `endBehavior` 將指示如何繼續進行。

如需有關使用 API 命令或的語法的詳細資訊 AWS CLI，請參閱 [endTime 時間戳記](#)。

對於在遵守日光節約時間 (DST) 的位置中的週期性維護時段期間進行選用排程組態的任務，從 DST 切換到標準時間以及從標準時間切換到 DST 時，時間會變動一小時。

**Note**

顯示在中的時區 AWS Management Console 是您目前的系統時區。但是，這些時區將在系統中轉換為 UTC。

## 結束行為

排程任務的結束行為會決定任務到達選定的 `endTime` 時，該任務和所有未完成的任務執行項目應有的狀態。

以下列出建立任務或任務範本時可供選取的結束行為：

- `STOP_ROLLOUT`
  - `STOP_ROLLOUT` 會停止將任務文件推展至任務目標群組中的所有剩餘裝置。此外，所有 `QUEUED` 和 `IN_PROGRESS` 任務執行都會繼續進行，直到達到終端狀態為止。除非您選取 `CANCEL` 或 `FORCE_CANCEL`，否則這是預設的結束行為。
- `CANCEL`
  - `CANCEL` 會停止將任務文件推展至任務目標群組中的所有剩餘裝置。此外，所有 `QUEUED` 任務執行項目都會取消，而所有 `IN_PROGRESS` 任務執行項目會繼續進行，直到達到終端狀態為止。
- `FORCE_CANCEL`
  - `FORCE_CANCEL` 會停止將任務文件推展至任務目標群組中的所有剩餘裝置。此外，所有 `QUEUED` 和 `IN_PROGRESS` 任務執行項目都會取消。

**Note**

若要選取 `endbehavior`，您必須選取 `endtime`

## 最長持續期間

不論 `startTime` 和 `endTime` 為何，排程任務的最長持續時間都必須小於或等於兩年。

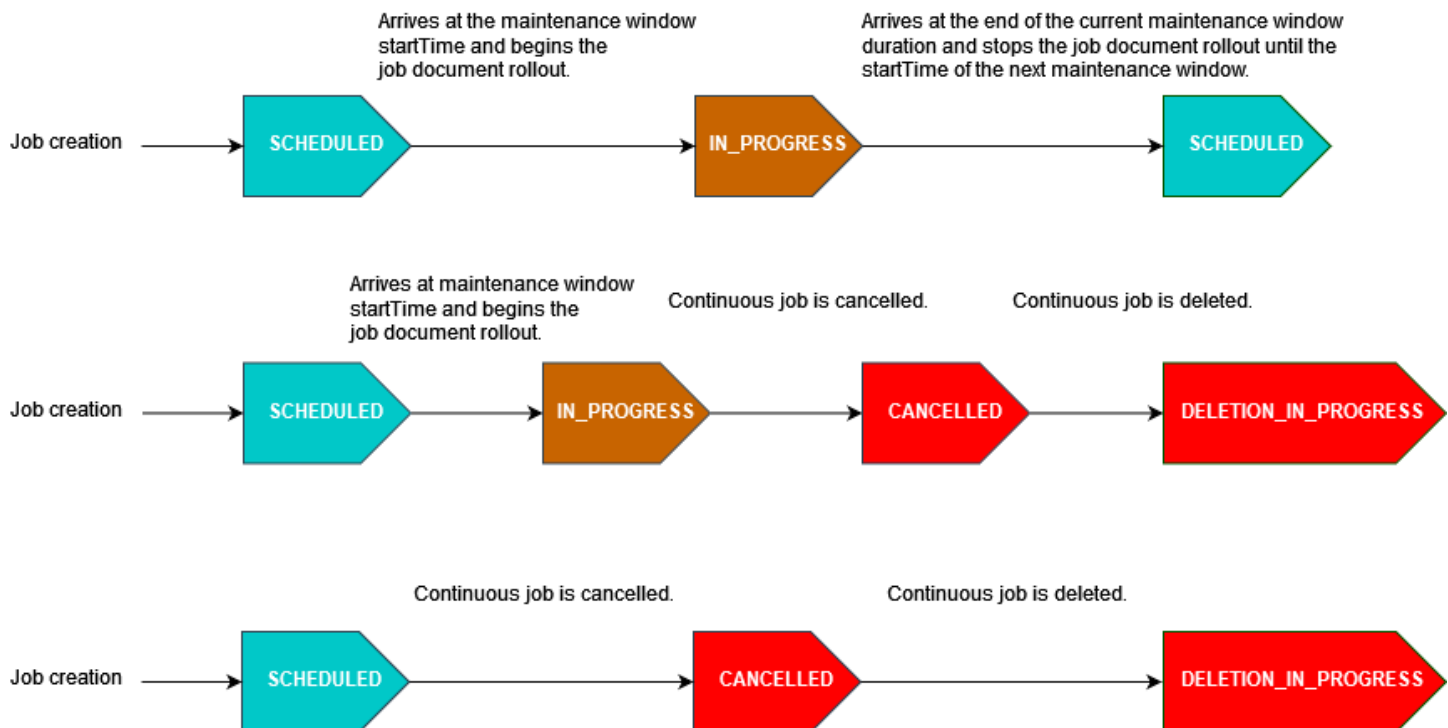
下表列出排程任務持續時間的常見情境：

排程任務範例編號	startTime	endTime	最長持續期間
1	在初始任務建立後立即執行。	初始任務建立一年後。	一年
2	初始任務建立後一個月。	初始建立任務後的 13 個月。	一年
3	初始任務建立一年後。	初始任務建立兩年後。	一年
4	在初始任務建立後立即執行。	初始任務建立兩年後。	兩年

### 週期性維護時段

維護時段是和 CreateJobTemplate API 內部的排程組態 SchedulingConfig 中 AWS Management Console 的選用 CreateJob 組態。您可以使用預先決定的開始時間、持續時間和頻率 (每日、每週或每月) 來設定週期性維護時段。維護時段僅適用於連續任務。週期性維護時段的最長持續時間為 23 小時 50 分鐘。

下圖說明具有選用維護時段之各種排程任務情境的任務狀態：



如需任務狀態的詳細資訊，請參閱 [任務和任務執行狀態](#)。

### Note

如果任務在維護時段到達 `endTime`，它將從 `IN_PROGRESS` 更新為 `COMPLETED`。此外，任何剩餘的任務執行都會遵循任務的 `endBehavior`。

## Cron 表達式

對於在維護時段期間以自訂頻率推出任務文件的排定任務，自訂頻率會使用 cron 表達式輸入。Cron 表達式有六個必要欄位，以空格隔開。

## 語法

```
cron(fields)
```

欄位	Values (數值)	萬用字元
分鐘	0-59	, - * /
小時	0-23	, - * /
D ay-of-month	1-31	, - * ? / L W
月	1-12 或 JAN-DEC	, - * /
D ay-of-week	1-7 或 SUN-SAT	, - * ? L #
年	1970-2199	, - * /

## 萬用字元

- , (逗號) 萬用字元包含額外的值。在 Month (月) 欄位，JAN、FEB、MAR 包括 January (一月)、February (二月) 與 March (三月)。
- - (破折號) 萬用字元用於指定範圍。在 Day (日) 欄位，1-15 包含指定月份的 1 至 15 號。
- \* (星號) 包含欄位中所有的值。在 Hours (小時) 欄位，\* 包含每個小時。您不能在 D ay-of-month 和 D ay-of-week 欄位中使用 \*。若您在其中一個欄位使用它，您必須在另一個欄位使用 ?。

- / (斜線) 萬用字元用於指定增量。在 Minutes (分鐘) 欄位，您可以輸入 1/10 指定每十分鐘的間隔，從小時的第一分鐘開始 (例如第 11、第 21、第 31 分鐘等)。
- ? (問號) 萬用字元用於表示不限定任何一個。在 Day-of-month 字段中，您可以輸入 7，如果您不在乎第 7 週的哪一天，可以輸入 ? 在 Day-of-week 段中。
- Day-of-month 或 Day-of-week 欄位中的 L 萬用字元會指定月份或週的最後一天。
- Day-of-month 欄位中的 W 萬用字元指定工作日。在 Day-of-month 欄位中，3W 指定最接近月份第三天的星期幾。
- Day-of-week 欄位中的 # 萬用字元會指定一個月內星期中指定日期的特定執行個體。例如，3#2 代表則該月的第二個星期二：3 是指星期二，因為它是每週的第三天，2 指的是一個月內該類型的第二天。

#### Note

如果您使用 '#' 字元，則只能在 day-of-week 欄位中定義一個運算式。例如："3#1,6#3" 無效，因為它被轉譯為兩個表達式。

## 限制

- 您無法在相同的 cron 運算式中指定 Day-of-month 和 Day-of-week 欄位。如果您在其中一個欄位指定了一值 (或 \*)，則必須在另一個欄位中使用 ?。

## 範例

針對週期性維護時段的 startTime 使用 cron 表達式時，請參閱下列範例 cron 字串。

分鐘	小時	月中的日	月	週中的日	年	意義
0	10	*	*	?	*	在每天上午 10:00 (UTC) 執行
15	12	*	*	?	*	在每天下午 12:15 (UTC) 執行



分鐘	小時	月中的日	月	週中的日	年	意義
0	18	?	*	MON-FRI	*	在每週一至週五下午 6:00 (UTC) 執行
0	8	1	*	?	*	在每個月第一天上午 8:00 (UTC) 執行

### 週期性維護時段持續時間結束邏輯

當維護時段期間的任務推展到達目前維護時段結束時，將會發生下列動作：

- 任務將停止向目標群組中的所有剩餘裝置推展任務文件。它將在下一個維護時段的 `startTime` 繼續。
- 狀態為 `QUEUED` 的所有工作執行都會保留在 `QUEUED` 中，直到下一個維護時段的 `startTime` 發生為止。在下一個時段中，它們可以在裝置準備好開始執行任務文件中指定的操作時切換至 `IN_PROGRESS`。
- 所有 `IN_PROGRESS` 狀態為的任務執行都會繼續執行任務文件中指定的動作，直到達到結束狀態為止。`JobExecutionsRetryConfig` 中指定的任何重試嘗試都會在下一個維護時段的 `startTime` 開始進行。

### 任務中止組態

達到閾值百分比的裝置滿足該條件時，即可使用此組態來建立取消任務的條件。例如，在以下情況下，您可以使用此組態來取消任務：

- 在有達到閾值百分比的裝置未接收到任務執行通知時，例如您的裝置與空中 (OTA) 更新不相容時。在這種情況下，您的裝置可報告 `REJECTED` 狀態。
- 在有達到閾值百分比的裝置報告其任務執行失敗時，例如當您的裝置在嘗試從 Amazon S3 URL 下載任務文件時發生連線中斷時。在這種情況下，您的裝置必須以程式設計方式向 AWS IoT 報告 `FAILURE` 狀態。
- 在任務執行開始後有達到閾值百分比的裝置任務執行逾時，因此報告 `TIMED_OUT` 狀態時。

- 出現多次重試失敗時。新增重試組態時，每次重試嘗試均會對您的 AWS 帳戶產生額外費用。在這種情況下，取消任務可以取消已排入佇列的任務執行，並避免這些執行的重試嘗試。如需重試組態以及將其與中止組態一起使用的詳細資訊，請參閱 [任務執行逾時和重試組態](#)。

您可以使用 AWS IoT 主控台或工作 API 來設定 AWS IoT 工作中止條件。

## 任務執行逾時和重試組態

當任務執行的進行時間超過設定的持續時間時，使用任務執行逾時組態向您傳送 [任務通知](#)。使用任務執行重試組態在任務失敗或逾時時重試執行。

### 任務執行逾時組態

使用任務執行逾時組態，當任務執行長時間非預期卡在 IN\_PROGRESS 狀態時即通知您。當任務為 IN\_PROGRESS 時，您可以監控任務執行的進度。

### 任務逾時的計時器

計時器有兩種類型：進行中計時器和步驟計時器。

#### 進行中計時器

建立任務或任務範本時，您可為進行中計時器指定一個介於 1 分鐘到 7 天之間的值。您可以更新此計時器的值，直到任務執行開始。計時器啟動後即無法更新，計時器值會套用於任務的所有任務執行。每當工作執行的 IN\_PROGRESS 狀態保持超過此間隔時間，工作執行就會失敗，並切換到終端機 TIMED\_OUT 狀態。AWS IoT 同時也會發佈 MQTT 通知。

#### 步驟計時器

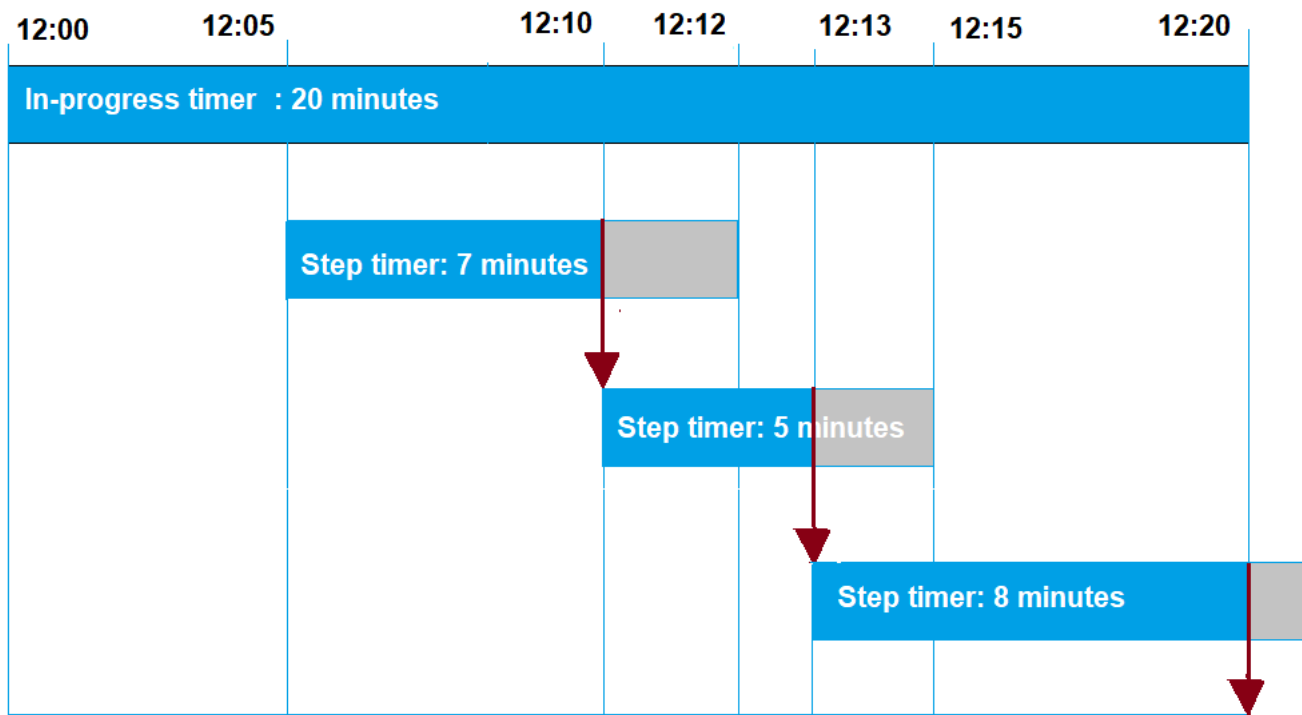
您還可以設定僅適用於要更新的任務執行的步驟計時器。此計時器對進行中計時器沒有任何影響。每次更新任務執行時，您可以為步驟計時器設定新值。在啟動物件的下一個待處理任務執行時，您也可以建立新的步驟計時器。如果任務執行維持在 IN\_PROGRESS 狀態的時間超過步驟計時器隔時，任務執行就會失敗，並切換到終止的 TIMED\_OUT 狀態。

#### Note

您可以使用 AWS IoT 主控台或 AWS IoT 工作 API 來設定進行中的計時器。若要指定步驟計時器，請使用 API。

## 計時器如何處理任務逾時

下圖說明在 20 分鐘逾時時間內進行中逾時和步驟逾時相互互動方式。



以下顯示不同步驟：

## 1. 12:00

建立新任務，並在建立任務時啟動 20 分鐘的進行中計時器。進行中計時器開始執行，且任務執行切換為 IN\_PROGRESS 狀態。

## 2. 12:05 PM

建立一個值為 7 分鐘的新步驟計時器。任務執行現在將在 12:12 PM 逾時。

## 3. 12:10 PM

建立一個值為 5 分鐘的新步驟計時器。當建立新步驟計時器時，會捨棄前一個步驟計時器，且任務執行現在將會在 12:15 PM 逾時。

## 4. 12:13 PM

建立一個值為 9 分鐘的新步驟計時器。捨棄前一個步驟計時器，且任務執行現在將在 12:20 PM 逾時，因為進行中計時器會在 12:20 PM 逾時。步驟計時器無法超過進行中計時器的絕對界限。

## 任務執行重試組態

當滿足某一組條件時，您可以使用重試組態來重試任務執行。當任務逾時或裝置失敗時，可以嘗試重試。若要因逾時失敗而重試執行，必須啟用逾時組態。

### 如何使用重試組態

使用下列步驟重試組態：

1. 確定是否針對 FAILED、TIMED\_OUT 或兩者的失敗條件使用重試組態。對於狀TIMED\_OUT態，在報告狀態後，AWS IoT 作業會自動重試設備的工作執行。
2. 若為 FAILED 狀態，請檢查您的任務執行失敗是否可以重試。如果裝置可重試，請將您的裝置以程式設計方式向 AWS IoT報告 FAILURE 狀態。以下部分介紹了有關可重試和不可重試失敗的詳細資訊。
3. 使用上述資訊來指定要用於每種失敗類型的重試次數。針對單一裝置，您可為兩種失敗類型組合指定最多 10 次重試。當執行成功或達到指定的嘗試次數時，重試嘗試即會自動停止。
4. 新增中止組態以在重複重試失敗的情況下取消任務，進而避免在發生大量重試嘗試時產生額外費用。

#### Note

當任務到達週期性維護時段結束時，所有 IN\_PROGRESS 任務執行會繼續執行任務文件中識別的動作，直到達到結束狀態為止。如果任務執行在維護時段之外到達 FAILED 或 TIMED\_OUT 的終端狀態，若重試次數未用完，則會在下一個維護時段進行重試。在下一次維護時段的 startTime 時，會建立新的任務執行，並進入 QUEUED 狀態，直到裝置準備好開始為止。

### 重試和中止組態

每次嘗試重試都會向您產生額外的 AWS 帳戶費用。為避免重複重試失敗而產生額外費用，我們建議新增中止組態。如需定價的詳細資訊，請參閱 [AWS IoT Device Management 定價](#)。

在裝置的高閾值百分比逾時或報告失敗時，您可能會遇到多次重試失敗。在這種情況下，您可以使用中止組態來取消任務，並避免任何已排入佇列的任務執行或進一步重試嘗試。

**Note**

當滿足取消任務執行的中止條件時，僅會取消 QUEUED 任務。將不會嘗試對裝置進行任何佇列重試。但是，系統不會取消目前處於 IN\_PROGRESS 狀態的任務執行。

在重試失敗的任務執行之前，我們亦建議您檢查任務執行失敗是否可重試，如下段所述。

**重試失敗類型 FAILED**

若要嘗試重試失敗類型 FAILED，則裝置必須以程式設計方式向 AWS IoT 報告任務執行失敗的 FAILURE 狀態。使用重試 FAILED 任務執行的條件來設定重試組態，並指定要執行的重試次數。當 AWS IoT 工作偵測到 FAILURE 狀態時，它會自動嘗試重試裝置的工作執行。重試將繼續進行，直到任務執行成功或達到最大重試次數。

您可以追蹤每次重試作業的嘗試情形，以及在這些裝置上執行的任務。透過追蹤執行狀態，在嘗試指定的重試次數後，您可以使用裝置報告失敗並啟動另一次重試嘗試。

**可重試和不可重試的失敗**

您的任務執行失敗可能是可重試或不可重試的失敗。每次重試嘗試都會對您的 AWS 帳戶產生費用。為避免多次重試嘗試產生額外費用，請先考慮檢查任務執行失敗是否可重試。可重試失敗的範例包括在嘗試從 Amazon S3 URL 下載任務文件時您的裝置所遇到的連線錯誤。如果您的任務執行失敗可重試，請使用程式將您的裝置設計為若任務執行失敗，則報告 FAILURE。然後，設定重試組態以重試 FAILED 執行。

如果執行無法重試，則若要避免重試和可能會對您的帳戶產生額外費用，我們建議使用程式將裝置設計為向 AWS IoT 報告 REJECTED 狀態。不可重試失敗的範例包括：裝置與接收任務更新不相容，或者在執行任務時遇到記憶體錯誤。在這些情況下，AWS IoT 作業不會重試工作執行，因為它只會在偵測到 FAILED 或 TIMED\_OUT 狀態時重試工作執行。

確定可重試任務執行失敗後，如果重試嘗試仍失敗，請考慮檢查裝置日誌。

**Note**

在具有選用排程組態的任務達到其 endTime 時，選取的 endBehavior 會停止將任務文件推展至目標群組中所有剩餘的裝置，並指定如何繼續其餘的任務執行。如果透過重試組態選取，則會重試嘗試。

## 重試失敗類型 TIMEOUT

如果您在建立 AWS IoT 工作時啟用逾時，當狀態從變更為時，工作將嘗試重試裝置的工作。當 IN\_PROGRESS 行 TIMED\_OUT。當進行中計時器逾時，或者您指定的步驟計時器為 IN\_PROGRESS 然後逾時，即可能會發生此狀態變更。重試會繼續進行，直到任務執行成功或達到此失敗類型的重試次數上限。

### 連續的任務和物件群組成員資格更新

對於任務狀態為 IN\_PROGRESS 的連續任務，則當物件的群組成員資格更新時，重試嘗試次數將重設為零。例如，假定您指定了五次重試嘗試，並且已執行了三次重試。如果現已從物件群組中移除物件，然後重新連結群組 (例如動態物件群組)，重試嘗試次數將重設為零。現在，您可以對物件群組執行五次重試嘗試，而不是剩餘的兩次嘗試。此外，當物件從物件群組中移除時，系統將取消其他重試嘗試。

## 指定其他組態

當您建立任務或任務範本時，您可以指定這些其他組態。下文顯示您何時可以指定這些組態。

- 建立自訂任務範本時。從範本建立任務時，系統將會儲存您指定的其他組態設定。
- 使用任務檔案建立自訂任務時。任務檔案可以是上傳至 S3 儲存貯體中的 JSON 檔案。
- 使用自訂任務範本建立自訂任務時。如果範本已指定了這些設定，則可以重新使用這些設定，或是指定新的組態設定覆寫這些設定。
- 使用 AWS 受管理的範本建立自訂工作時。

### 主題

- [使用 AWS Management Console 來指定任務組態](#)
- [使用 AWS IoT 任務 API 來指定任務組態](#)

## 使用 AWS Management Console 來指定任務組態

您可以使用 AWS IoT 主控台為工作新增不同的組態。建立任務後，您可以在任務詳細資訊頁面上看到任務組態的狀態詳細資訊。如需不同組態及其運作方式的詳細資訊，請參閱 [任務組態的運作方式](#)。

建立任務或任務範本時新增任務組態。

建立自訂任務範本時

建立自訂任務範本時指定推展組態

1. 移至 [AWS IoT 主控台的 \[Job 範本\] 中樞](#)，然後選擇 [建立工作範本]。
2. 指定任務範本屬性，提供任務文件，展開要新增的組態，然後指定組態參數。

## 建立自訂任務時

### 建立自訂任務時指定推展組態

1. 移至 [AWS IoT 主控台的 \[Job 中心\]](#)，然後選擇 [建立工作]。
2. 選擇 Create a custom job (建立自訂任務) 並指定任務屬性、目標，以及是否針對任務文件使用任務檔案或任務範本。您可以使用自訂範本或 AWS 受管理的範本。
3. 選擇任務組態，然後展開 Rollout configuration (推展組態) 指定是否使用 Constant rate (恆定速率) 或 Exponential rate (指數速率)。然後，指定組態參數。

下一節將顯示您可以為每個組態指定的參數。

## 推展組態

您可以指定使用恆定推展率或指數推展率。

- 設定恆定推展率

若要設定任務執行的恆定速率，請選擇恆定速率，然後針對速率的上限指定每分鐘最大值。此值為選用值，範圍從 1 到 1000。如果未設定，則將使用 1000 作為預設值。

- 設定指數推展率

若要設定指數速率，請選擇 Exponential rate (指數速率)，然後指定這些參數：

- 每分鐘的基本速率

在達到速率提高標準的已通知的裝置數量或成功的裝置數量閾值之前的任務執行速率。

- 增量因數

在 Number of notified devices (已通知裝置數量) 或 Number of succeeded devices (成功裝置數量) 閾值滿足 Rate increase criteria (速率提高標準) 後推展率增加的指數因數。

- 速率提高標準

Number of notified devices (已通知的裝置數量) 或 Number of succeeded devices (成功的裝置數量) 的閾值。

## 中止組態

選擇 Add new configuration (新增組態) 並為每個組態指定下列參數：

- 失敗類型

指定啟動任務中止的失敗類型。其中包括 FAILED (失敗)、REJECTED (拒絕)、TIMED\_OUT (逾時) 或 ALL (所有)。

- 增量因數

指定在符合任務中止條件之前，必須發生的任務執行完成數量。

- 閾值百分比

指定啟動任務中止的已執行物件總數。

## 排程組態

每項任務都可以在初始建立時立即開始、或排定在之後的日期和時間開始，或者在週期性維護時段進行。

選擇 Add new configuration (新增組態) 並為每個組態指定下列參數：

- 任務開始

指定任務開始的日期和時間。

- 週期性維護時段

週期性維護時段用於定義任務可將任務文件部署到任務中目標裝置的特定日期和時間。維護時段可以在每天、每週、每月或自訂日期和時間重複。

- 任務結束

指定任務開始的日期和時間。

- 任務結束行為

為所有未完成的任務執行項目選取當任務完成時應有的結束行為。



**Note**

在具有選用排程組態和所選結束時間的任務達到結束時間時，任務會停止推展至目標群組中所有剩餘的裝置。它也會利用選取的結束行為，此行為有關如何根據重試組態繼續進行剩餘的工作執行及其重試嘗試。

## 逾時組態

根據預設不會逾時，且系統已取消或刪除您的任務執行。若要使用逾時，請選擇啟用逾時，然後指定一個介於 1 分鐘到 7 天之間的逾時值。

## 重試組態

**Note**

建立任務後，即無法更新重試次數。您只能移除所有失敗類型的重試組態。建立任務時，請考慮用於組態的適當重試次數。為避免因潛在的重試失敗而產生超額成本，請新增中止組態。

選擇 Add new configuration (新增組態) 並為每個組態指定下列參數：

- 失敗類型

指定應觸發任務執行重試的失敗類型。其中包括 Failed (失敗)、Timeout (逾時) 及 All (全部)。

- 重試次數

指定選定的 Failure type (失敗類型) 的重試次數。對於兩種失敗類型的組合，最多可嘗試 10 次重試。

## 使用 AWS IoT 任務 API 來指定任務組態

您可以使用 [CreateJob](#) 或 [CreateJobTemplate](#) API 來指定不同的工作組態。下列各節描述如何新增這些組態。新增組態後，您可以使用 [JobExecutionSummary](#) 和 [JobExecutionSummaryForJob](#) 檢視其狀態。

如需不同組態及其運作方式的詳細資訊，請參閱 [任務組態的運作方式](#)。

## 推展組態

您可以為推展組態指定恆定推展率或指數推展率。

- 設定恆定推展率

若要設定恆定推展率，請使用 [JobExecutionsRolloutConfig](#) 物件將 `maximumPerMinute` 參數新增至 `CreateJob` 請求。此參數指定可以發生任務執行的速率上限。此值為選用值，範圍從 1 到 1000。如果未設定該值，則將使用 1000 作為預設值。

```
"jobExecutionsRolloutConfig": {
  "maximumPerMinute": 1000
}
```

- 設定指數推展率

若要設定可變任務推展率，請使用 [JobExecutionsRolloutConfig](#) 物件。您可以在執行 `CreateJob` API 操作時設定 `ExponentialRolloutRate` 屬性。以下範例使用 `exponentialRate` 參數來設定指數推展率。如需這些參數的詳細資訊，請參閱 [ExponentialRolloutRate](#)。

```
{
  ...
  "jobExecutionsRolloutConfig": {
    "exponentialRate": {
      "baseRatePerMinute": 50,
      "incrementFactor": 2,
      "rateIncreaseCriteria": {
        "numberOfNotifiedThings": 1000,
        "numberOfSucceededThings": 1000
      },
      "maximumPerMinute": 1000
    }
  }
  ...
}
```

其中參數：

`baseRatePer分鐘`

指定達到 `numberOfNotifiedThings` 或 `numberOfSucceededThings` 閾值之前的任務執行速率。

## incrementFactor

指定推展率在達到 `numberOfNotifiedThings` 或 `numberOfSucceededThings` 閾值之後提高的指數因數。

## rateIncreaseCriteria

指定 `numberOfNotifiedThings` 或 `numberOfSucceededThings` 閾值。

## 中止組態

若要使用 API 新增此組態，請在執行 [CreateJob](#) 或 [CreateJobTemplate](#) API 操作時指定 [AbortConfig](#) 參數。以下範例顯示任務推展的中止組態，該組態經歷了多次失敗的執行，如 [CreateJob](#) API 操作所指定。

### Note

刪除任務執行會影響已完成執行總數的計算值。當任務中止時，服務會建立自動化的 `comment` 和 `reasonCode`，以區分使用者驅動的取消和任務中止的取消。

```
"abortConfig": {
  "criteriaList": [
    {
      "action": "CANCEL",
      "failureType": "FAILED",
      "minNumberOfExecutedThings": 100,
      "thresholdPercentage": 20
    },
    {
      "action": "CANCEL",
      "failureType": "TIMED_OUT",
      "minNumberOfExecutedThings": 200,
      "thresholdPercentage": 50
    }
  ]
}
```

其中參數：

## 動作

指定符合中止條件時所要採取的動作。此為必要參數，且 CANCEL 是唯一有效的值。

## failureType

指定應啟動任務中止的失敗類型。有效值為 FAILED、REJECTED、TIMED\_OUT、ALL。

## minNumberOfExecutedThings

指定在符合任務中止條件之前，必須發生的任務執行完成數量。在此範例中，AWS IoT 不會檢查是否在至少 100 個裝置完成任務執行之前發生任務中止。

## thresholdPercentage

針對可以啟動任務中止的已執行任務指定物件總數。在此範例中，如果符合臨界值百分比，則會依序 AWS IoT 檢查並啟動工作中止。如果在 100 次執行完成後至少有 20% 的完成執行失敗，則會取消任務推展。如果不符合此條件，AWS IoT 則檢查 200 次執行完成後，是否至少有 50% 的已完成執行超時。如果發生情況，即會取消任務推展。

## Scheduling configuration (排程組態)

若要使用 API 新增此組態，請在執行 [CreateJob](#) 或 [CreateJobTemplate](#) API 操作時指定選用的 [SchedulingConfig](#)。

```
"SchedulingConfig": {
  "endBehavior": string
  "endTime": string
  "maintenanceWindows": string
  "startTime": string
}
```

其中參數：

### startTime

指定任務開始的日期和時間。

### endTime

指定任務開始的日期和時間。

## maintenanceWindows

指定是否為排程任務選取了選用維護時段，以便將任務文件推展至目標群組中的所有裝置。maintenanceWindow 的字串格式是日期為 YYYY/MM/DD，而時間則為 hh:mm。

## endBehavior

指定排程任務到達 endTime 時的任務行為。

### Note

可在 [DescribeJob](#) 和 [DescribeJobTemplate](#) API 中檢視任務的選擇性 SchedulingConfig。

## 逾時組態

若要使用 API 新增此組態，請在執行 [CreateJob](#) 或 [CreateJobTemplate](#) API 操作時指定 [TimeoutConfig](#) 參數。

### 使用逾時組態

1. 若要在建立工作或工作範本時設定進行中計時器，請為選擇性 [TimeoutConfig](#) 物件的 inProgressTimeoutInMinutes 屬性設定值。

```
"timeoutConfig": {  
  "inProgressTimeoutInMinutes": number  
}
```

2. 若要指定工作執行的步驟計時器，請設定呼叫 stepTimeoutInMinutes 時的值 [UpdateJobExecution](#)。步驟計時器僅適用於您更新的任務執行。您可以在每次更新任務執行時，為此計時器設定新值。

### Note

UpdateJobExecution 可以透過建立值為 -1 的新步驟計時器，來捨棄已建立的步驟計時器。

```
{
```

```

...
  "statusDetails": {
    "string" : "string"
  },
  "stepTimeoutInMinutes": number
}

```

3. 若要建立新的步驟計時器，您也可以呼叫 [StartNextPendingJobExecution](#) API 作業。

## 重試組態

### Note

建立任務時，請考慮用於組態的適當重試次數。為避免因潛在的重試失敗而產生超額成本，請新增中止組態。建立任務後，即無法更新重試次數。您只能使用 [UpdateJob](#) API 作業將重試次數設定為 0。

若要使用 API 新增此組態，請在執行 [CreateJob](#) 或 [CreateJobTemplate](#) API 操作時指定 [jobExecutionsRetryConfig](#) 參數。

```

{
  ...
  "jobExecutionsRetryConfig": {
    "criteriaList": [
      {
        "failureType": "string",
        "numberOfRetries": number
      }
    ]
  }
  ...
}

```

其中 `criteriaList` 是一個陣列，用於指定條件清單，可確定任務的每種失敗類型所允許的重試次數。

## 裝置和任務

裝置可以使用 MQTT、HTTP 簽章版本 4 或 HTTP TLS 與 AWS IoT 工作進行通訊。若要在裝置與 AWS IoT 工作通訊時決定要使用的端點，請執行 `DescribeEndpoint` 命令。例如，如果您執行此命令：

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

您會得到類似以下的結果：

```
{
  "endpointAddress": "a1b2c3d4e5f6g7-ats.iot.us-west-2.amazonaws.com"
}
```

## 使用 MQTT 通訊協定

設備可以使用 MQTT AWS IoT 協議與作業進行通信。裝置訂閱 MQTT 主題，接收新工作的通知，並接收來自「AWS IoT 工作」服務的回應。裝置發佈於 MQTT 主題來查詢或更新任務啟動的狀態。每項裝置皆有其一般性 MQTT 主題。如需發佈與訂閱 MQTT 主題的詳細資訊，請參閱 [裝置通訊協定](#)。

使用此通訊方法，您的裝置會使用其裝置特定的憑證和私密金鑰來透 AWS IoT 過工作進行驗證。

裝置可以訂閱下列主題。thing-name 是與裝置關聯的事物名稱。

- **`$aws/things/thing-name/jobs/notify`**

訂閱此主題，即可在待定任務啟動清單中新增或移除任務啟動時通知您。

- **`$aws/things/thing-name/jobs/notify-next`**

訂閱此主題，即可在下一個待處理任務執行項目發生變更時接收通知。

- **`$aws/things/thing-name/jobs/request-name/accepted`**

「AWS IoT 工作」服務會針對 MQTT 主題發佈成功與失敗訊息。主題是透過將 `accepted` 或 `rejected` 附加到用於發出請求的主題來形成。在這裡，`request-name` 是請求的名稱，例如 `Get`，主題可以是：`$aws/things/myThing/jobs/get`。AWS IoT 然後，工作會發佈有關該 `$aws/things/myThing/jobs/get/accepted` 主題的成功訊息。

- **`$aws/things/thing-name/jobs/request-name/rejected`**

在此例中，`request-name` 是請求的名稱，例如 `Get`。如果請求失敗，AWS IoT 作業會發佈關於該 `$aws/things/myThing/jobs/get/rejected` 主題的失敗訊息。

您也可以使用下列 HTTPS API 操作：

- 藉由呼叫 [UpdateJobExecution](#) API 來更新任務執行的狀態。
- 藉由呼叫 [DescribeJobExecution](#) API 來查詢任務執行的狀態。

- 藉由呼叫 [GetPendingJobExecutions](#) API 來擷取待定任務執行的清單。
- 呼叫 [DescribeJobExecution](#) API (其中 jobId 作為 \$next) 來擷取下一個待定任務執行。
- 藉由呼叫 [StartNextPendingJobExecution](#) API 來取得並開始下一個待定任務執行。

## 使用 HTTP Signature 第 4 版

裝置可以使用連接埠 443 上的 HTTP 簽章版本 4 與 AWS IoT 工作進行通訊。此為 AWS SDK 與 CLI 使用的方法。如需這些工具的詳細資訊，請參閱[AWS CLI 命令參考：iot-jobs-data](#)或 [AWS SDK 和工具](#)，並參閱您偏好語言的 `iotJobsDataPlane` 章節。

使用這種通訊方法，您的裝置會使用 IAM 登入資料透過 AWS IoT 工作進行驗證。

使用此方法可運用下列命令：

- 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

裝置可以使用支援此通訊協定的協力廠商軟體用戶端，在連接埠 8443 上使用 HTTP TLS 與 AWS IoT 工作進行通訊。

若使用此方法，您的裝置會使用 X.509 憑證型身分驗證 (例如，其裝置專屬的憑證與私有金鑰)。

使用此方法可運用下列命令：

- DescribeJobExecution
- GetPendingJobExecutions
- StartNextPendingJobExecution



- UpdateJobExecution

## 設定裝置與任務合作

本節中的範例使用 MQTT 說明裝置與 AWS IoT 任務服務之間如何運作。或者，您可以使用對應的 API 或 CLI 命令。在這些範例中，我們會假設名為 MyThing 的裝置將訂閱下列 MQTT 主題：

- \$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

如果您使用程式碼簽章 AWS IoT，您的裝置程式碼必須驗證程式碼檔案的簽章。該簽章是包含任務文件中的 codesign 屬性。如需驗證程式碼檔簽章的詳細資訊，請參閱[裝置代理程式範例](#)。

### 主題

- [裝置工作流程](#)
- [任務工作流程](#)
- [任務通知](#)

## 裝置工作流程

裝置可以使用下列其中一種方式來處理其執行的任務。

- 取得下一個任務
  1. 裝置首次上線時，應訂閱裝置的 notify-next 主題。
  2. 以 jobId \$next 呼叫 [DescribeJobExecution](#) MQTT API 來取得下一個任務、任務文件及其他詳細資訊，包括任何儲存於 statusDetails 的狀態。如果任務文件具有程式碼檔簽章，則必須先驗證簽章，然後再繼續處理任務請求。
  3. 呼叫 [UpdateJobExecution](#) MQTT API 以更新工作狀態。或者，裝置可以呼叫 [StartNextPendingJobExecution](#)，將此步驟與前一步驟結合至同一個呼叫裡。
  4. (選用) 當您呼叫 [UpdateJobExecution](#) 或 [StartNextPendingJobExecution](#) 時，您可以透過設定 stepTimeoutInMinutes 的值來新增步驟計時器。

5. 執行由使用 [UpdateJobExecution](#) MQTT API 之工作文件所指定的動作，以回報工作進度。
6. 使用此 jobId 來呼叫 [DescribeJobExecution](#) MQTT API，即可繼續監控任務執行。如果刪除任務執行，[DescribeJobExecution](#) 會傳回 `ResourceNotFoundException`。

如果任務執行在裝置執行任務時遭到取消或刪除，裝置應可還原為有效狀態。

7. 任務完成時呼叫 [UpdateJobExecution](#) MQTT API 以更新任務狀態並回報成功或失敗。
8. 由於此任務的執行狀態已變更為結束狀態，因此下一個可執行的任務 (若有) 將會變更。裝置會收到下一個待定任務執行已變更的通知。此時，裝置應繼續如步驟 2 所述運作。

如果裝置仍在線上，它會繼續收到下一個待定任務執行的通知。這包括其任務執行資料、它何時完成任務或何時新增待定任務執行。在此情況下，裝置應繼續如步驟 2 所述運作。

- 選取可用的任務

1. 裝置首次上線時，應訂閱物件的 `notify` 主題。
2. 呼叫 [GetPendingJobExecutions](#) MQTT API 來取得待定任務執行清單。
3. 如果清單包含了一或多個任務執行項目，請選取其中一項。
4. 呼叫 [DescribeJobExecution](#) MQTT API 來取得任務文件及其他詳細資訊，包括儲存於 `statusDetails` 的任何狀態。
5. 呼叫 [UpdateJobExecution](#) MQTT API 以更新工作狀態。如果 `includeJobDocument` 欄位在此命令中設定為 `true`，則裝置可以略過先前的步驟並擷取當前的任務文件。
6. 或者，當您呼叫 [UpdateJobExecution](#) 時，您可以透過設定 `stepTimeoutInMinutes` 的值來新增步驟計時器。
7. 執行由使用 [UpdateJobExecution](#) MQTT API 之工作文件所指定的動作，以回報工作進度。
8. 使用此 jobId 來呼叫 [DescribeJobExecution](#) MQTT API，即可繼續監控任務執行。如果任務執行在裝置執行任務時遭取消或刪除，裝置應可還原為有效狀態。
9. 任務完成時呼叫 [UpdateJobExecution](#) MQTT API 以更新任務狀態並回報成功或失敗。

如果裝置維持在線上，只要有新的待定任務執行出現，裝置就會收到所有待定任務執行的通知。在此情況下，裝置可繼續如步驟 2 所述運作。

如果裝置無法實施任務，則應呼叫 [UpdateJobExecution](#) MQTT API 將任務狀態更新為 `REJECTED`。

## 任務工作流程

以下顯示了任務工作流程中從開始新任務到報告任務執行完成狀態的各個步驟。

### 開始新任務

建立新工作後，AWS IoT Core 會針對每個目標裝置發佈有關 `$aws/things/thing-name/jobs/notify` 主題的訊息。

訊息包含下列資訊：

```
{
  "timestamp":1476214217017,
  "jobs":{
    "QUEUED":[
      {
        "jobId":"0001",
        "queuedAt":1476214216981,
        "lastUpdatedAt":1476214216981,
        "versionNumber" : 1
      }
    ]
  }
}
```

當任務執行排入佇列時，裝置就會收到這個以 `'$aws/things/thingName/jobs/notify'` 為主題的訊息。

#### Note

對於設有選擇性 `SchedulingConfig` 的任務，任務會維持 `SCHEDULED` 的初始狀態。當任務達到選定的 `startTime` 時，會發生下列情形：

- 任務狀態將更新為 `IN_PROGRESS`。
- 任務會開始將任務文件推展至目標群組中的所有裝置。

### 取得任務資訊

若需取得更多有關任務執行的資訊，該裝置會呼叫 [DescribeJobExecution](#) MQTT API，且 `includeJobDocument` 欄位將設為 `true` (此為預設)。

如果要求成功，AWS IoT Jobs 服務會針對下列 `$aws/things/MyThing/jobs/0023/get/accepted` 主題發佈訊息：

```
{
  "clientToken" : "client-001",
  "timestamp" : 1489097434407,
  "execution" : {
    "approximateSecondsBeforeTimedOut": number,
    "jobId" : "023",
    "status" : "QUEUED",
    "queuedAt" : 1489097374841,
    "lastUpdatedAt" : 1489097374841,
    "versionNumber" : 1,
    "jobDocument" : {
      < contents of job document >
    }
  }
}
```

如果要求失敗，AWS IoT Jobs 服務會發佈關於該 `$aws/things/MyThing/jobs/0023/get/rejected` 主題的訊息。

裝置現在具有可用於執行任務遠端操作的任務文件。如果任務文件包含 Amazon S3 預先簽章的 URL，則裝置可以使用此 URL 下載任務需要的所有檔案。

## 報告任務執行狀態

裝置執行工作時，可以呼叫 [UpdateJobExecutionMQTT API](#) 以更新工作執行的狀態。

例如，裝置可以藉由以 `IN_PROGRESS` 主題發佈下列訊息而將任務執行狀態更新為 `$aws/things/MyThing/jobs/0023/update`：

```
{
  "status": "IN_PROGRESS",
  "statusDetails": {
    "progress": "50%"
  },
  "expectedVersion": "1",
  "clientToken": "client001"
}
```

任務會透過將訊息發佈至 `$aws/things/MyThing/jobs/0023/update/accepted` 或 `$aws/things/MyThing/jobs/0023/update/rejected` 主題進行回應：

```
{
  "clientToken":"client001",
  "timestamp":1476289222841
}
```

裝置可以透過呼叫 [StartNextPendingJobExecution](#) 來合併前兩個請求。這會取得並開始下一個待定任務執行，並能讓裝置更新任務執行狀態。此請求也會在任務執行待定时傳回任務文件。

如果工作包含 [TimeoutConfig](#)，則進行中的計時器會開始執行。您也可以設定呼叫時的值，以設定工作執行的步驟計 `stepTimeoutInMinutes` 計時器 [UpdateJobExecution](#)。步驟計時器僅適用於您更新的任務執行。您可以在每次更新任務執行時，為此計時器設定新值。您也可以在建通話時建立步數計時器 [StartNextPendingJobExecution](#)。如果任務執行維持在 `IN_PROGRESS` 狀態的時間超過步驟計時器隔時，任務執行就會失敗，並切換到終止的 `TIMED_OUT` 狀態。當您建立任務時，步驟計時器不會影響您設定的進行中計時器。

`status` 欄位可以設定為 `IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。若任務執行已經處於結束狀態，您就無法更新其狀態。

## 報告執行完成

裝置完成執行工作時，會呼叫 [UpdateJobExecution](#) MQTT API。如果任務成功，則將 `status` 設定為 `SUCCEEDED`，並在訊息酬載的 `statusDetails` 中以名稱值對的方式加入任務的其他相關資訊。任務執行完成時，執行中與步驟計時器會結束。

例如：

```
{
  "status":"SUCCEEDED",
  "statusDetails": {
    "progress":"100%"
  },
  "expectedVersion":"2",
  "clientToken":"client-001"
}
```

如果任務不成功，則將 `status` 設定為 `FAILED`，並在 `statusDetails` 加入所發生錯誤的資訊：

```
{
```

```
"status":"FAILED",
"statusDetails": {
  "errorCode":"101",
  "errorMsg":"Unable to install update"
},
"expectedVersion":"2",
"clientToken":"client-001"
}
```

### Note

`statusDetails` 屬性可以包含任意數量的名稱值對。

J AWS IoT obs 服務收到此更新時，會發佈有關 `$aws/things/MyThing/jobs/notify` 主題的訊息，指出工作執行已完成：

```
{
  "timestamp":1476290692776,
  "jobs":{}
}
```

## 額外任務

如果裝置還有其他待定的任務執行，則會包含在發佈至 `$aws/things/MyThing/jobs/notify` 的訊息中。

例如：

```
{
  "timestamp":1476290692776,
  "jobs":{
    "QUEUED":[{
      "jobId":"0002",
      "queuedAt":1476290646230,
      "lastUpdatedAt":1476290646230
    }],
    "IN_PROGRESS":[{
      "jobId":"0003",
      "queuedAt":1476290646230,
      "lastUpdatedAt":1476290646230
    }],
  }
}
```

```

    }
  }
}

```

## 任務通知

當 AWS IoT 工作擱置中或清單中的第一個工作執行變更時，Jobs 服務會將 MQTT 訊息發佈至保留主題。裝置可訂閱這些主題以追蹤待定的任務。

### 任務通知類型

任務通知會作為 JSON 承載發佈到 MQTT 主題。通知有兩種：

#### ListNotification

ListNotification 包含不超過 15 個待定任務執行的清單。它們會依狀態排序 (IN\_PROGRESS 任務執行先於 QUEUED 任務執行)，然後依其佇列的時間排序。

每當符合以下其中一個條件，就發佈 ListNotification。

- 新的任務執行會排入佇列或變更為非終結狀態 (IN\_PROGRESS 或 QUEUED)。
- 舊任務執行會變更為終止狀態 (FAILED、SUCCEEDED、CANCELED、TIMED\_OUT、REJECTED 或 REMOVED)。

清單通知 (位於 **QUEUED** 或 **IN\_PROGRESS** 中的最多 15 個待定任務執行)

不具選用排程組態和週期性維護時段

(最多 10 個任務執行)

永遠顯示在中 ListNotification。

具有選用排程組態和週期性維護時段

(最多 5 個任務執行)

只有在維護 ListNotification  
時段中才會顯示在中。

#### NextNotification

- NextNotification 包含佇列中下一個任務執行的摘要資訊。

當清單中第一個任務執行變更時，就發佈 NextNotification。

- 新的任務執行會新增到清單中作為 QUEUED，而且會是清單中的第一個。

- 清單中不是第一個的現有任務執行的狀態，從 QUEUED 變更為 IN\_PROGRESS，並且成為清單中的第一個。(清單中沒有其他 IN\_PROGRESS 任務執行時，或者其狀態從 QUEUED 變更為 IN\_PROGRESS 的任務執行比清單中任何其他 IN\_PROGRESS 任務執行較早排入佇列時，就會發生這種情況)。
- 任務執行的狀態，即清單中第一個變更為終結狀態並從清單中移除。

如需發佈與訂閱 MQTT 主題的詳細資訊，請參閱 [the section called “裝置通訊協定”](#)。

### Note

當您使用 HTTP Signature 第 4 版或 HTTP TLS 來與任務通訊時，通知無法使用。

## 任務待定

當 AWS IoT 工作新增至物件的擱置工作執行清單或清單中的第一個工作執行項目變更時，Jobs 服務會在 MQTT 主題上發佈訊息：

- `$aws/things/thingName/jobs/notify`
- `$aws/things/thingName/jobs/notify-next`

訊息包含的承載範例如下：

`$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
  } ]
}
}

```

如果名為 `this-job` 的任務執行項目源於已選取選用排程組態的任務，且任務文件推出排定在維護時段期間執行，則它只會顯示在週期性維護時段期間。在維護時段之外，名為 `this-job` 的任務會從待定任務執行清單中排除，如下列範例所示。

```

{
  "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"}
  }
}

```

如果名為 `other-job` 的任務執行項目源於已選取選用排程組態的任務，且任務文件推出排定在維護時段期間執行，則它只會顯示在週期性維護時段期間。在維護時段之外，名為 `other-job` 的任務不會列為下一個任務執行，如下列範例所示。

```
{ } //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"
```

可能的任務執行狀態為

QUEUED、IN\_PROGRESS、FAILED、SUCCEEDED、CANCELED、TIMED\_OUT、REJECTED 及 REMOVED。

下列一連串範例顯示在建立任務執行，且其從一個狀態變為另一個狀態時，發佈到各主題的通知。

首先，建立一個任務，稱為 job1。此通知會發佈到 jobs/notify 主題：

```
{
  "timestamp": 1517016948,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

此通知會發佈到 jobs/notify-next 主題：

```
{
  "timestamp": 1517016948,
```

```
"execution": {
  "jobId": "job1",
  "status": "QUEUED",
  "queuedAt": 1517016947,
  "lastUpdatedAt": 1517016947,
  "versionNumber": 1,
  "executionNumber": 1,
  "jobDocument": {
    "operation": "test"
  }
}
```

建立另一個任務 (job2) 時，此通知會發佈至 jobs/notify 主題：

```
{
  "timestamp": 1517017192,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job1",
        "queuedAt": 1517016947,
        "lastUpdatedAt": 1517016947,
        "executionNumber": 1,
        "versionNumber": 1
      },
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
        "lastUpdatedAt": 1517017191,
        "executionNumber": 1,
        "versionNumber": 1
      }
    ]
  }
}
```

通知不會發佈至 jobs/notify-next 主題，因為佇列中的下一個任務 (job1) 尚未變更。當 job1 開始執行時，它的狀態變更為 IN\_PROGRESS。不會發佈通知，因為任務清單與佇列中下一個任務尚未變更。

新增第三個任務 (job3) 時，此通知會發佈至 jobs/notify 主題：

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

通知不會發佈至 `jobs/notify-next` 主題，因為佇列中的下一個任務仍然是 (job1)。

當 job1 完成時，它的狀態會變更為 `SUCCEEDED`，而且此通知會發佈至 `jobs/notify` 主題：

```
{
  "timestamp": 1517186269,
  "jobs": {
    "QUEUED": [
      {
        "jobId": "job2",
        "queuedAt": 1517017191,
```

```

    "lastUpdatedAt": 1517017191,
    "executionNumber": 1,
    "versionNumber": 1
  },
  {
    "jobId": "job3",
    "queuedAt": 1517017905,
    "lastUpdatedAt": 1517017905,
    "executionNumber": 1,
    "versionNumber": 1
  }
]
}
}

```

此時，系統會從佇列中移除 job1，而下一個要執行的任務為 job2。此通知會發佈到 jobs/notify-next 主題：

```

{
  "timestamp": 1517186269,
  "execution": {
    "jobId": "job2",
    "status": "QUEUED",
    "queuedAt": 1517017191,
    "lastUpdatedAt": 1517017191,
    "versionNumber": 1,
    "executionNumber": 1,
    "jobDocument": {
      "operation": "test"
    }
  }
}
}

```

如果 job3 必須在 job2 前開始執行 (不建議這麼做)，可將 job3 的狀態變更為 IN\_PROGRESS。這種情況下，job2 將不會是佇列中的下一個任務，而此通知會發佈至 jobs/notify-next 主題：

```

{
  "timestamp": 1517186779,
  "execution": {
    "jobId": "job3",
    "status": "IN_PROGRESS",
    "queuedAt": 1517017905,

```

```
"startedAt": 1517186779,  
"lastUpdatedAt": 1517186779,  
"versionNumber": 2,  
"executionNumber": 1,  
"jobDocument": {  
  "operation": "test"  
}  
}  
}
```

不會將通知發佈到 `jobs/notify` 主題，因為未新增或移除任何任務。

如果裝置拒絕 `job2` 並將其狀態更新為 `REJECTED`，此通知會發佈至 `jobs/notify` 主題：

```
{  
  "timestamp": 1517189392,  
  "jobs": {  
    "IN_PROGRESS": [  
      {  
        "jobId": "job3",  
        "queuedAt": 1517017905,  
        "lastUpdatedAt": 1517186779,  
        "startedAt": 1517186779,  
        "executionNumber": 1,  
        "versionNumber": 2  
      }  
    ]  
  }  
}
```

如果 `job3` (仍在進行中) 是強制刪除，此通知會發佈到 `jobs/notify` 主題：

```
{  
  "timestamp": 1517189551,  
  "jobs": {}  
}
```

此時，佇列是空的。此通知會發佈到 `jobs/notify-next` 主題：

```
{  
  "timestamp": 1517189551  
}
```

# AWS IoT 任務 API 操作

AWS IoT 任務 API 可用於以下任一類別：

- 管理任務，例如任務的管理和控制。這是控制平面。
- 實施這些任務的裝置。這是資料平面，可讓您傳送和接收資料。

任務管理與控制會使用 HTTPS 通訊協定的 API。裝置可以使用 MQTT 或 HTTPS 通訊協定的 API。控制平面 API 適用於少量呼叫，特別是在建立與追蹤任務時使用。其通常會為單一請求開啟連線，並在接收到回應之後關閉連線。資料平面 HTTPS 和 MQTT API 允許長輪詢。這些 API 操作適用於可擴展至數百萬個裝置的大量流量。

每個 AWS IoT 任務 HTTPS API 都有一項對應的命令，可讓您從 AWS Command Line Interface (AWS CLI) 呼叫 API。命令為小寫字母，且在組成 API 名稱的單字之間有連字號。例如，您可以輸入以下內容來呼叫 CLI 上的 CreateJob API：

```
aws iot create-job ...
```

如果操作期間發生錯誤，您會收到包含錯誤相關資訊的錯誤回應。

## ErrorResponse

包含在 AWS IoT 任務服務操作期間所發生的錯誤資訊。

以下範例顯示此操作的語法：

```
{
  "code": "ErrorCode",
  "message": "string",
  "clientToken": "string",
  "timestamp": timestamp,
  "executionState": JobExecutionState
}
```

以下是 ErrorResponse 的說明：

code

ErrorCode 可設定為：

## InvalidTopic

請求已傳送至 AWS IoT 任務命名空間中未對應至任何 API 操作的主題。

## InvalidJson

請求的內容無法解譯為有效的 UTF-8 編碼 JSON。

## InvalidRequest

請求的內容無效。例如，當 UpdateJobExecution 請求包含了無效的狀態詳細資訊，此代碼就會傳回。訊息包含錯誤的詳細資訊。

## InvalidStateTransition

已嘗試更新，將任務執行變更為由於任務執行目前狀態而無效的狀態。例如，嘗試將狀態為 SUCCEEDED 的請求變更為狀態 IN\_PROGRESS。在此情況下，錯誤訊息的本文也會包含 executionState 欄位。

## ResourceNotFound

請求主題指定的 JobExecution 不存在。

## VersionMismatch

請求所指定的預期版本不符合 AWS IoT 任務服務中的任務執行版本。在此情況下，錯誤訊息的本文也會包含 executionState 欄位。

## InternalError

處理請求時發生內部錯誤。

## RequestThrottled

請求受到調節。

## TerminalStateReached

在處於結束狀態的任務上執行描述任務的命令時發生。

## message

錯誤訊息字串。

## clientToken

用於將請求與回覆建立關聯的任意字串。

## timestamp

Epoch 時間，以秒為單位。



## executionState

[JobExecutionState](#) 物件。只有當 code 欄位有 `InvalidStateTransition` 或 `VersionMismatch` 值，此欄位才會包含在內。在這些情況下，就不必另外執行 `DescribeJobExecution` 請求以獲得任務執行狀態資料。

以下列出任務 API 操作和資料類型。

- [任務管理、控制 API 和控制資料類型](#)
- [任務裝置 MQTT、HTTPS API 操作和資料類型](#)

## 任務管理、控制 API 和控制資料類型

以下命令可透過 HTTPS 通訊協定在 CLI 中用於任務管理和控制。

- [任務管理和控制資料類型](#)
- [任務管理和控制 API 操作](#)

執行這個命令，為 CLI 命令確定 `endpoint-url` 參數。

```
aws iot describe-endpoint --endpoint-type=iot:Jobs
```

此命令會傳回下列輸出。

```
{
  "endpointAddress": "account-specific-prefix.jobs.iot.aws-region.amazonaws.com"
}
```

### Note

任務端點不支援 ALPN `z-amzn-http-ca`。

## 任務管理和控制資料類型

管理和控制應用程式會使用下列資料類型與 AWS IoT 任務通訊。

## 任務

Job物件包含了工作的詳細資訊。語法如下列範例所示。

```
{
  "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
}
}
```

如需詳細資訊，請參閱[Job](#)或[job](#)。

## JobSummary

JobSummary 物件包含了工作摘要。語法如下列範例所示。

```
{
  "jobArn": "string",
  "jobId": "string",
  "status": "IN_PROGRESS|CANCELED|SUCCEEDED|SCHEDULED",
  "targetSelection": "CONTINUOUS|SNAPSHOT",
  "thingGroupId": "string",
  "createdAt": timestamp,
  "lastUpdatedAt": timestamp,
  "completedAt": timestamp
}
```

如需詳細資訊，請參閱[JobSummary](#)或[job-summary](#)。

## JobExecution

JobExecution 物件代表在裝置上執行任務。語法如下列範例所示。

**Note**

當您使用控制平面 API 操作時，JobExecution 資料類型不包含 JobDocument 欄位。若要取得此資訊，您可以使用 [GetJobDocument](#) API 操作或 [get-job-document](#) CLI 命令。

```
{
  "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
}
```

如需詳細資訊，請參閱[JobExecution](#)或[job-execution](#)。

### JobExecutionSummary

JobExecutionSummary 物件包含了任務執行摘要的資訊。語法如下列範例所示。

```
{
  "executionNumber": 1234567890,
  "queuedAt": timestamp,
  "lastUpdatedAt": timestamp,
  "startedAt": timestamp,
  "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|REMOVED"
}
```

如需詳細資訊，請參閱[JobExecutionSummary](#)或[job-execution-summary](#)。

## JobExecutionSummaryForJob

JobExecutionSummaryForJob物件包含了關於特定工作之工作執行的資訊摘要。語法如下列範例所示。

```
{
  "executionSummaries": [
    {
      "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyThing",
      "jobExecutionSummary": {
        "status": "IN_PROGRESS",
        "lastUpdatedAt": 1549395301.389,
        "queuedAt": 1541526002.609,
        "executionNumber": 1
      }
    },
    ...
  ]
}
```

如需詳細資訊，請參閱[JobExecutionSummaryForJob](#)或[job-execution-summary-for-job](#)。

## JobExecutionSummaryForThing

JobExecutionSummaryForThing物件包含了關於特定物件之工作執行的資訊摘要。語法如下列範例所示：

```
{
  "executionSummaries": [
    {
      "jobExecutionSummary": {
        "status": "IN_PROGRESS",
        "lastUpdatedAt": 1549395301.389,
        "queuedAt": 1541526002.609,
        "executionNumber": 1
      },
      "jobId": "MyThingJob"
    },
    ...
  ]
}
```

如需詳細資訊，請參閱 [JobExecutionSummaryForThing](#) 或 [job-execution-summary-for-thing](#)。

## 任務管理和控制 API 操作

使用下列 API 操作或 CLI 命令：

### AssociateTargetsWithJob

將群組與持續性任務建立關聯。必須符合以下條件：

- 建立任務時必須將 `targetSelection` 欄位設定為「CONTINUOUS」。
- 任務狀態目前必須是「IN\_PROGRESS」。
- 與工作關聯的目標總數不得超過 100 個。

### HTTPS request

```
POST /jobs/jobId/targets

{
  "targets": [ "string" ],
  "comment": "string"
}
```

如需更多詳細資訊，請參閱 [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"  
}
```

如需更多詳細資訊，請參閱 [associate-targets-with-job](#)。

## CancelJob

取消任務。

### HTTPS request

```
PUT /jobs/jobId/cancel  
  
{  
  "force": boolean,  
  "comment": "string",  
  "reasonCode": "string"  
}
```

如需更多詳細資訊，請參閱 [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"  
}
```

如需更多詳細資訊，請參閱 [cancel-job](#)。

## CancelJobExecution

取消裝置上的任務執行。

### HTTPS request

```
PUT /things/thingName/jobs/jobId/cancel

{
  "force": boolean,
  "expectedVersion": "string",
  "statusDetails": {
    "string": "string"
    ...
  }
}
```

如需更多詳細資訊，請參閱 [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"
  }
}
```

如需更多詳細資訊，請參閱 [cancel-job-execution](#)。



## CreateJob

建立任務。您可以將 Amazon S3 儲存貯體 (`documentSource` 參數) 或請求本文 (`document` 參數) 中的檔案連結提供給任務文件。

您可以將選用 `targetSelection` 參數設定為 `CONTINUOUS` (預設為 `SNAPSHOT`) 以讓任務「持續」執行。持續任務可以用來在新增至群組時加入或升級裝置，因為它會持續執行，並在新增的項目上啟動。即使在建立任務時群組中的物件已完成任務之後，仍可能發生這種情況。

任務可以有一個選用 [TimeoutConfig](#)，其會設定進行中計時器的值。進行中計時器無法更新，並會套用到任務的所有執行。

以下驗證會在 `CreateJobAPI` 的引數上執行：

- `targets` 引數必須是有效物件或物件群組 ARN 的清單。所有物件和物件群組都必須在您的 AWS 帳戶中。
- `documentSource` 引數必須是連結至任務文件的有效 Amazon S3 URL。Amazon S3 URL 的格式為：`https://s3.amazonaws.com/bucketName/objectName`。
- 由 `documentSource` 引數指定存放文件的 URL 必須是 UTF-8 編碼的 JSON 文件。
- 由於 MQTT 訊息大小限制 (128 KB) 與加密因素，因此工作文件的大小限制為 32 KB。
- 在您的 AWS 帳戶中，`jobId` 必須是唯一的。

## 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
}
}

```

如需更多詳細資訊，請參閱 [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"
}
}
```

如需更多詳細資訊，請參閱 [create-job](#)。

## DeleteJob

刪除任務及其相關的任務執行。

刪除任務可能會需要時間，根據為任務與各種其他因素所建立的任務執行數量而定。雖然任務被刪除，任務狀態將顯示為「DELETION\_IN\_PROGRESS」。嘗試刪除或取消狀態已為「DELETION\_IN\_PROGRESS」的任務將會導致錯誤。

## HTTPS request

```
DELETE /jobs/jobId?force=force
```

如需更多詳細資訊，請參閱 [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
}
```

如需更多詳細資訊，請參閱 [delete-job](#)。

## DeleteJobExecution

刪除任務執行。

HTTPS request

```
DELETE /things/thingName/jobs/jobId/executionNumber/executionNumber?force=force
```

如需更多詳細資訊，請參閱 [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  
}
```

如需更多詳細資訊，請參閱 [delete-job-execution](#)。

## DescribeJob

取得任務執行的詳細資訊。

HTTPS request

```
GET /jobs/jobId
```

如需更多詳細資訊，請參閱 [DescribeJob](#)。

## CLI syntax

```
aws iot describe-job \  
--job-id <value> \  
[--cli-input-json <value>] \  
[--generate-cli-skeleton]
```

cli-input-json 格式：

```
{  
  "jobId": "string"  
}
```

如需更多詳細資訊，請參閱 [describe-job](#)。

## DescribeJobExecution

取得工作執行的詳細資訊。任務的執行狀態必須為 SUCCEEDED 或 FAILED。

## HTTPS request

```
GET /things/thingName/jobs/jobId?executionNumber=executionNumber
```

如需更多詳細資訊，請參閱 [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
```

```
}
```

如需更多詳細資訊，請參閱 [describe-job-execution](#)。

## GetJobDocument

取得工作的工作文件。

### Note

在傳回的文件中，預留位置 URL 不會被預先簽章的 Amazon S3 URL 取代。預先簽章的 URL 只有在 AWS IoT 任務服務透過 MQTT 收到請求時才會產生。

## HTTPS request

```
GET /jobs/jobId/job-document
```

如需更多詳細資訊，請參閱 [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"  
}
```

如需更多詳細資訊，請參閱 [get-job-document](#)。

## ListJobExecutionsForJob

取得工作的工作執行清單。

## HTTPS request

```
GET /jobs/jobId/things?status=status&maxResults=maxResults&nextToken=nextToken
```

如需更多詳細資訊，請參閱 [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"  
}
```

如需更多詳細資訊，請參閱 [list-job-executions-for-job](#)。

## ListJobExecutionsForThing

取得物件的工作執行清單。

## HTTPS request

```
GET /things/thingName/jobs?status=status&maxResults=maxResults&nextToken=nextToken
```

如需更多詳細資訊，請參閱 [ListJobExecutionsForThing](#)。

## CLI syntax

```
aws iot list-job-executions-for-thing \  
--thing-name <value> \  
[--generate-cli-skeleton]
```



```
[--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"  
}
```

如需更多詳細資訊，請參閱 [list-job-executions-for-thing](#)。

## ListJobs

取得您 AWS 帳戶中的任務清單。

### HTTPS request

```
GET /jobs?  
status=status&targetSelection=targetSelection&thingGroupName=thingGroupName&thingGroupId=thing
```

如需更多詳細資訊，請參閱 [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"
}
```

如需更多詳細資訊，請參閱 [list-jobs](#)。

## UpdateJob

更新指定任務的支援欄位。timeoutConfig 的更新值僅對新進行中的啟動生效。目前，進行中的啟動會繼續以先前的逾時組態啟動。

## 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
}
}

```

如需更多詳細資訊，請參閱 [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
}
}
```

如需更多詳細資訊，請參閱 [update-job](#)。

## 任務裝置 MQTT、HTTPS API 操作和資料類型

以下命令可透過 MQTT 和 HTTPS 通訊協定執行。在執行任務之裝置的資料平面上使用這些 API 操作。

### 任務裝置 MQTT 和 HTTPS 資料類型

與 AWS IoT 任務服務通訊時會使用以下資料類型搭配 MQTT 和 HTTPS 通訊協定。

#### JobExecution

JobExecution 物件代表在裝置上執行任務。語法如下列範例所示。

#### Note

當您使用 MQTT 和 HTTP 資料平面 API 操作時，JobExecution 資料類型會包含 JobDocument 欄位。您的裝置可以使用此資訊從任務執行擷取任務文件。

```
{
  "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
}
```

如需詳細資訊，請參閱[JobExecution](#)或[job-execution](#)。

## JobExecutionState

JobExecutionState 包含有關任務執行狀態的資訊。語法如下列範例所示。

```
{
  "status": "QUEUED|IN_PROGRESS|FAILED|SUCCEEDED|CANCELED|TIMED_OUT|REJECTED|
REMOVED",
  "statusDetails": {
    "string": "string"
    ...
  }
  "versionNumber": "number"
}
```

如需詳細資訊，請參閱[JobExecutionState](#)或[job-execution-state](#)。

## JobExecutionSummary

包含關於工作執行的部分資訊。語法如下列範例所示。

```
{
  "jobId": "string",
  "queuedAt": timestamp,
  "startedAt": timestamp,
  "lastUpdatedAt": timestamp,
  "versionNumber": "number",
  "executionNumber": long
}
```

如需詳細資訊，請參閱[JobExecutionSummary](#)或[job-execution-summary](#)。

請參閱以下部分，了解更多有關 MQTT 和 HTTPS API 操作的資訊：

- [任務裝置 MQTT API 操作](#)
- [任務裝置 HTTP API](#)

## 任務裝置 MQTT API 操作

您可以發出任務裝置命令，方法為將 MQTT 訊息發佈至[用於任務命令的保留主題](#)。

您的裝置端用戶端必須訂閱這些命令的回應訊息主題。如果您使用 AWS IoT 裝置用戶端，您的裝置會自動訂閱回應主題。這表示，無論您的用戶端是否已訂閱回應訊息主題，訊息代理程式都會將回應訊息主題發佈至發佈命令訊息的用戶端。這些回應訊息不會透過訊息代理程式傳遞，而且其他用戶端或規則無法訂閱這些訊息。

訂閱機群監控解決方案的任務和 `jobExecution` 事件主題時，首先啟用[任務和任務執行事件](#)來接收雲端的任何事件。透過訊息代理程式處理並可由 AWS IoT 規則使用的任務進度訊息會發佈為[任務事件](#)。由於訊息代理程式會發佈回應訊息，即使沒有明確地訂閱它們，您的用戶端也必須設定為接收並識別其所接收的訊息。您的用戶端也必須確認，在用戶端對訊息採取動作之前，內送訊息主題中的 `thingName` 套用至用戶端的物件名稱。

### Note

AWS IoT 若傳送訊息，以回應 MQTT 任務 API 命令訊息，無論您是否明確地訂閱這些訊息，都會向您的帳戶收費。

以下說明 MQTT API 操作及其請求和回應語法。所有 MQTT API 操作均有下列參數：

`clientToken`

用來關聯請求和回應的選用用戶端字符。在此輸入任意值，就會在回應中反映出來。

`timestamp`

訊息傳送的 epoch 時間，以秒為單位。

`GetPendingJobExecutions`

針對特定物件，取得其所有未處於終止狀態之任務的清單。

欲呼叫此 API，請發佈訊息於 `$aws/things/thingName/jobs/get`。

請求酬載：

```
{ "clientToken": "string" }
```

訊息代理程式將發佈 `$aws/things/thingName/jobs/get/accepted` 和 `$aws/things/thingName/jobs/get/rejected`，即使沒有具體訂閱它們也一樣。不過，為了讓您的用戶端接收訊息，它必須正在接聽它們。如需詳細資訊，請參閱[關於任務 API 訊息的注意事項](#)。

回應酬載：

```
{
  "InProgressJobs" : [ JobExecutionSummary ... ],
  "queuedJobs" : [ JobExecutionSummary ... ],
  "timestamp" : 1489096425069,
  "clientToken" : "client-001"
}
```

`InProgressJobs` 和 `queuedJobs` 回傳狀態為 `IN_PROGRESS` 或 `QUEUED` 的 [JobExecutionSummary](#) 物件清單。

`StartNextPendingJobExecution`

取得並啟動物件的下一個待定任務執行 (狀態 `IN_PROGRESS` 或 `QUEUED`)。

- 狀態為 `IN_PROGRESS` 的任務執行會先傳回。
- 任務執行會依照其排入佇列的順序傳回。當您任務的目標群組中新增或移除物件時，請確認所有新任務執行相較於現有任務執行的推展順序。
- 如果下一個待定任務執行為 `QUEUED`，則其狀態會變更為 `IN_PROGRESS`，而任務執行狀態詳細資訊也會依指定設定。
- 如果下一個待定任務執行已經為 `IN_PROGRESS`，則其狀態詳細資訊不會變更。
- 如果沒有工作執行為待定，則回應不會包括 `execution` 欄位。
- 您也可以選擇透過設定 `stepTimeoutInMinutes` 的屬性來新增步驟計時器。如果您未透過執行 `UpdateJobExecution` 來更新此屬性的值，則步驟計時器逾期時，任務執行將逾時。

欲呼叫此 API，請發佈訊息於 `$aws/things/thingName/jobs/start-next`。

請求酬載：

```
{
  "statusDetails": {
```

```
    "string": "job-execution-state"
    ...
  },
  "stepTimeoutInMinutes": long,
  "clientToken": "string"
}
```

## statusDetails

名稱值對的集合，能夠描述任務執行的狀態。如果未指定，則 statusDetails 不會改變。

## stepTimeoutInMinutes

指定此裝置必須完成這項任務執行的時間。在此計時器到期或計時器重設之前 (藉由呼叫 UpdateJobExecution，將狀態設為 IN\_PROGRESS，並在 stepTimeoutInMinutes 欄位中指定新的逾時值)，如果任務執行狀態未設定為結束狀態，任務執行狀態就會設為 TIMED\_OUT。設定此逾時不會影響任務執行逾時，該任務執行逾時可能已在建立任務時加以指定 (使用欄位 timeoutConfig 的 CreateJob)。

訊息代理程式將發佈 \$aws/things/*thingName*/jobs/start-next/accepted 和 \$aws/things/*thingName*/jobs/start-next/rejected，即使沒有具體訂閱它們也一樣。不過，為了讓您的用戶端接收訊息，它必須正在接聽它們。如需詳細資訊，請參閱[關於任務 API 訊息的注意事項](#)。

回應酬載：

```
{
  "execution" : JobExecutionData,
  "timestamp" : timestamp,
  "clientToken" : "string"
}
```

execution 為 [JobExecution](#) 物件。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

取得工作執行的詳細資訊。

您可以將 `jobId` 設定為 `$next`，傳回物件 (狀態為 `IN_PROGRESS` 或 `QUEUED`) 的下一個待定任務執行。

欲呼叫此 API，請發佈訊息於 `$aws/things/thingName/jobs/jobId/get`。

請求酬載：

```
{
"jobId" : "022",
"thingName" : "MyThing",
"executionNumber": long,
"includeJobDocument": boolean,
"clientToken": "string"
}
```

### thingName

與裝置關聯的物件名稱。

### jobId

此工作在建立時被指派的獨特識別符。

或者使用 `$next` 傳回物件 (狀態為 `IN_PROGRESS` 或 `QUEUED`) 的下一個待定任務執行。在此情況下，狀態為 `IN_PROGRESS` 的任務執行會先傳回。工作執行會依照其建立的順序傳回。

### executionNumber

(選用) 可識別在裝置上之任務執行的編號。如果未指定，則會傳回最新的工作執行。

### includeJobDocument

(選用) 除非設為 `false`，否則回應會包含任務文件。預設值為 `true`。

訊息代理程式將發佈 `$aws/things/thingName/jobs/jobId/get/accepted` 和 `$aws/things/thingName/jobs/jobId/get/rejected`，即使沒有具體訂閱它們也一樣。不過，為了讓您的用戶端接收訊息，它必須正在接聽它們。如需詳細資訊，請參閱[關於任務 API 訊息的注意事項](#)。

回應酬載：

```
{
  "execution" : JobExecutionData,
  "timestamp": "timestamp",
  "clientToken": "string"
}
```

`execution` 為 [JobExecution](#) 物件。

### UpdateJobExecution

更新工作執行的狀態。您可以選擇透過設定 `stepTimeoutInMinutes` 的屬性來新增步驟計時器。如果您未透過再次執行 `UpdateJobExecution` 來更新此屬性的值，則步驟計時器逾期時，任務執行將逾時。

欲呼叫此 API，請發佈訊息於 `$aws/things/thingName/jobs/jobId/update`。

請求酬載：

```
{
  "status": "job-execution-state",
  "statusDetails": {
    "string": "string"
    ...
  },
  "expectedVersion": "number",
  "executionNumber": long,
  "includeJobExecutionState": boolean,
  "includeJobDocument": boolean,
  "stepTimeoutInMinutes": long,
  "clientToken": "string"
}
```

### status

任務執行的新狀態 (IN\_PROGRESS、FAILED、SUCCEEDED 或 REJECTED)。這必須在每次更新時指定。

## statusDetails

名稱值對的集合，能夠描述任務執行的狀態。如果未指定，則 statusDetails 不會改變。

## expectedVersion

預期的目前工作執行版本。每次您更新工作執行，其版本編號就會增加。如果 AWS IoT 任務服務中儲存的任務執行版本不符，則會拒絕更新並顯示 VersionMismatch 錯誤。也會傳回 [ErrorResponse](#)，其中包含目前任務執行狀態資料。(這樣就不必另外執行 DescribeJobExecution 請求以獲得任務執行狀態資料。)

## executionNumber

(選用) 可識別在裝置上之任務執行的編號。如果未指定，則會使用最新的工作執行。

## includeJobExecutionState

(選用) 若被包括在內並設定為 true，則回應會包含 JobExecutionState 欄位。預設值為 false。

## includeJobDocument

(選用) 若被包括在內並設定為 true，則回應會包含 JobDocument。預設值為 false。

## stepTimeoutInMinutes

指定此裝置必須完成這項任務執行的時間。如果任務執行狀態未在此計時器逾期之前，或未在計時器重設之前設定為終止狀態，任務執行狀態會設定為 TIMED\_OUT。設定或重設此逾時不會影響任務執行逾時，該任務執行逾時可能已在建立任務時加以指定。

訊息代理程式將發佈 `$aws/things/thingName/jobs/jobId/update/accepted` 和 `$aws/things/thingName/jobs/jobId/update/rejected`，即使沒有具體訂閱它們也一樣。不過，為了讓您的用戶端接收訊息，它必須正在接聽它們。如需詳細資訊，請參閱[關於任務 API 訊息的注意事項](#)。

回應酬載：

```
{
  "executionState": JobExecutionState,
  "jobDocument": "string",
  "timestamp": timestamp,
  "clientToken": "string"
}
```

## executionState

[JobExecutionState](#) 物件。

## jobDocument

[工作文件](#) 物件。

## timestamp

訊息傳送的 epoch 時間，以秒為單位。

## clientToken

用於將請求和回應建立關聯的用戶端符記。

使用 MQTT 通訊協定時，也可以執行下列更新：

## JobExecutionsChanged

每當物件的待定任務執行清單新增或移除任務執行時就會傳送。

使用主題：

`$aws/things/thingName/jobs/notify`

訊息酬載：

```
{
  "jobs" : {
    "JobExecutionState": [ JobExecutionSummary ... ]
  },
  "timestamp": timestamp
}
```

## NextJobExecutionChanged

每當物件待定任務清單上的下一個任務執行變更就會傳送，亦即定義 [DescribeJobExecution](#) 時加上 `jobId $next`。當下一個任務的執行詳細資訊變更，此訊息並不會傳送，只有在加上 `jobId $next` 之 [DescribeJobExecution](#) 傳回的下一個任務變更時才會傳送。考量狀態為 QUEUED 的任務執行 J1 與 J2。J1 是下一個待定任務執行。如果 J2 的狀態變更為 IN\_PROGRESS 而 J1 的狀態保持不變，則會傳送此通知並包含 J2 的詳細資訊。

使用主題：

```
$aws/things/thingName/jobs/notify-next
```

訊息酬載：

```
{  
  "execution" : JobExecution,  
  "timestamp": timestamp,  
}
```

## 任務裝置 HTTP API

裝置可以在連接埠 443 上使用 HTTP Signature 第 4 版與 AWS IoT 任務通訊。此為 AWS SDK 與 CLI 使用的方法。如需這些工具詳細資訊，請參閱 [AWS CLI 命令參考：iot-jobs-data](#) 或 [AWS SDK 與工具](#)。

以下命令可用於執行任務的裝置。如需如何使用 API 操作搭配 MQTT 通訊協定的詳細資訊，請參閱 [任務裝置 MQTT API 操作](#)。

### GetPendingJobExecutions

針對特定物件，取得其所有未處於終止狀態之任務的清單。

#### HTTPS request

```
GET /things/thingName/jobs
```

回應：

```
{  
  "InProgressJobs" : [ JobExecutionSummary ... ],  
  "queuedJobs" : [ JobExecutionSummary ... ]  
}
```

如需更多詳細資訊，請參閱 [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"  
}
```

如需更多詳細資訊，請參閱 [get-pending-job-executions](#)。

## StartNextPendingJobExecution

取得並啟動物件的下一個待定任務執行 (狀態 IN\_PROGRESS 或 QUEUED)。

- 狀態為 IN\_PROGRESS 的任務執行會先傳回。
- 工作執行會依照其建立的順序傳回。
- 如果下一個待定任務執行為 QUEUED，則其狀態會變更為 IN\_PROGRESS，而任務執行狀態詳細資訊也會依指定設定。
- 如果下一個待定任務執行已經為 IN\_PROGRESS，則其狀態詳細資訊不會變更。
- 如果沒有工作執行為待定，則回應不會包括 execution 欄位。
- 您也可以選擇透過設定 stepTimeoutInMinutes 的屬性來新增步驟計時器。如果您未透過執行 UpdateJobExecution 來更新此屬性的值，則步驟計時器逾期時，任務執行將逾時。

## HTTPS request

請求語法如下列範例所示：

```
PUT /things/thingName/jobs/$next  
{  
  "statusDetails": {  
    "string": "string"  
    ...  
  },  
  "stepTimeoutInMinutes": long  
}
```

如需更多詳細資訊，請參閱 [StartNextPendingJobExecution](#)。

## CLI syntax

概要：

```
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  
}
```

如需更多詳細資訊，請參閱 [start-next-pending-job-execution](#)。

## DescribeJobExecution

取得工作執行的詳細資訊。

您可以將 `jobId` 設定為 `$next`，傳回物件的下一個待定任務執行。任務的執行狀態必須為 `QUEUED` 或 `IN_PROGRESS`。

## HTTPS request

要求：

```
GET /things/thingName/jobs/jobId?  
executionNumber=executionNumber&includeJobDocument=includeJobDocument
```

回應：

```
{  
  "execution" : JobExecution,
```

```
}
```

如需更多詳細資訊，請參閱 [DescribeJobExecution](#)。

## CLI syntax

概要：

```
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  
}
```

如需更多詳細資訊，請參閱 [describe-job-execution](#)。

## UpdateJobExecution

更新工作執行的狀態。您也可以選擇透過設定 `stepTimeoutInMinutes` 的屬性來新增步驟計時器。如果您未透過再次執行 `UpdateJobExecution` 來更新此屬性的值，則步驟計時器逾期時，任務執行將逾時。

## HTTPS request

要求：

```
POST /things/thingName/jobs/jobId  
{  
  "status": "job-execution-state",  
  "statusDetails": {
```



```

    "string": "string"
    ...
  },
  "expectedVersion": "number",
  "includeJobExecutionState": boolean,
  "includeJobDocument": boolean,
  "stepTimeoutInMinutes": long,
  "executionNumber": long
}

```

如需更多詳細資訊，請參閱 [UpdateJobExecution](#)。

## CLI syntax

### 概要：

```

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
}

```

}

如需更多詳細資訊，請參閱 [update-job-execution](#)。

## 使用 AWS IoT 工作保護使用者和裝置

若要授權使用者在其裝置上使用 AWS IoT 工作，您必須使用 IAM 政策授予他們許可。然後必須使用 AWS IoT Core 原則來授權裝置，以安全地連線 AWS IoT、接收工作執行，以及更新執行狀態。

### 工作的必要原則 AWS IoT 類型

下表說明您必須用於授權的不同政策類型。如需所需使用政策的詳細資訊，請參閱 [授權](#)。

#### 所需政策類型

使用案例	通訊協定	身分驗證	控制平面/資料平面	身分類型	所需政策類型
授權管理員、操作員或雲端服務安全使用任務	HTTPS	AWS 簽章版本 4 驗證 (連接埠 443)	控制平面和資料平面兩者	Amazon Cognito 身分、IAM 或聯合身分使用者	IAM 政策
授權您的 IoT 裝置安全使用任務	MQTT/HTTP S	TCP 或 TLS 相互驗證 (連接埠 8883 或 443)	資料平面	X.509 憑證	AWS IoT Core 政策

若要授權可在控制平面和資料平面上執行的 AWS IoT 作業作業，您必須使用 IAM 政策。身分必須已透過 AWS IoT 驗證才能執行這些作業，必須是 [Amazon Cognito 身分](#) 或 [IAM 使用者、群組和角色](#)。如需身分驗證的相關詳細資訊，請參閱 [身分驗證](#)。

裝置現在必須在資料層上獲得授權，方法是使用 AWS IoT Core 原則安全地連線至裝置閘道。裝置閘道可讓裝置安全地通訊 AWS IoT、接收工作執行，以及更新工作執行狀態。裝置通訊透過使用加密 [MQTT](#) 或 [HTTPS](#) 通訊協定進行加密。這些通訊協定會使用提供的 [X.509 用戶端憑證](#) 來驗證裝置連線。

以下說明您如何授權使用者、大聲服務和裝置使用 AWS IoT 工作。如需控制平面和資料平面 API 操作的資訊，請參閱 [AWS IoT 任務 API 操作](#)。

## 主題

- [授權使用者和雲端服務使用 AWS IoT 任務](#)
- [授權您的裝置安全地使用資料層上的 AWS IoT 工作](#)

## 授權使用者和雲端服務使用 AWS IoT 任務

若要授權使用者和雲端任務，您必須在控制平面和資料平面上都使用 IAM 政策。這些政策必須與 HTTPS 通訊協定一起使用，並且必須使用 AWS Signature 第 4 版身分驗證 (連接埠 443) 以驗證使用者。

### Note

AWS IoT Core 不得在控制平面上使用策略。IAM 政策僅用於授權使用者或雲端服務。如需使用所需政策類型的詳細資訊，請參閱 [工作的必要原則 AWS IoT 類型](#)。

IAM 政策是包含政策陳述式的 JSON 文件。政策聲明使用效果、動作以及資源元素來指定資源、允許或拒絕動作，以及在何種條件下允許或拒絕動作。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM JSON 政策元素參考](#)。

### Warning

建議您不要使用萬用字元許可，例如 IAM 政策或政 AWS IoT Core 策 "Action": ["iot:\*"] 中。使用萬用字元許可不是推薦的安全最佳實務。如需詳細資訊，請參閱 [AWS IoT 政策過於寬容](#)。

## 控制平面上的 IAM 政策

在控制平面上，IAM 政策使用 `iot:` 字首，以授權對應的任務 API 作業。例如 `iot:CreateJob` 政策動作會授予使用者許可使用 [CreateJob](#) API。

### 政策動作

下表顯示使用 API 動作的 IAM 政策動作和許可列表。如需有關資源類型的資訊，請參閱 [由定義的資源類型 AWS IoT](#)。如需有關 AWS IoT 動作的詳細資訊，請參閱 [由定義的動作 AWS IoT](#)。

## 控制平面上的 IAM 政策動作

政策動作	API 操作	資源類型	描述
<code>iot:AssociateTargetsWithJob</code>	<a href="#">AssociateTargetsWithJob</a>	<ul style="list-style-type: none"> <li>• job</li> <li>• 物件</li> <li>• thinggroup</li> </ul>	代表將群組與連續任務關聯的許可。每次發出關聯目標的要求時，就會檢查一次 <code>iot:AssociateTargetsWithJob</code> 許可。
<code>iot:CancelJob</code>	<a href="#">CancelJob</a>	job	代表取消任務的許可。每次發出取消任務的要求時，就會檢查一次 <code>iot:CancelJob</code> 許可。
<code>iot:CancelJobExecution</code>	<a href="#">CancelJobExecution</a>	<ul style="list-style-type: none"> <li>• job</li> <li>• 物件</li> </ul>	代表可取消任務執行的許可。每次發出取消任務執行的要求時，就會檢查一次 <code>iot:CancelJobExecution</code> 許可。
<code>iot&gt;CreateJob</code>	<a href="#">CreateJob</a>	<ul style="list-style-type: none"> <li>• job</li> <li>• 物件</li> <li>• thinggroup</li> <li>• jobtemplate</li> <li>• package</li> </ul>	代表建立任務的許可。每次發出建立任務的要求時，就會檢查一次 <code>iot:CreateJob</code> 許可。
<code>iot&gt;CreateJobTemplate</code>	<a href="#">CreateJobTemplate</a>	<ul style="list-style-type: none"> <li>• job</li> <li>• jobtemplate</li> <li>• package</li> </ul>	代表建立任務範本的許可。每次發出建立任務範本的要求時，就會檢查一次 <code>iot:CreateJobTemplate</code> 許可。
<code>iot&gt;DeleteJob</code>	<a href="#">DeleteJob</a>	job	代表刪除任務的許可。每次發出刪除任務的要求時，就會檢查一次 <code>iot&gt;DeleteJob</code> 許可。

政策動作	API 操作	資源類型	描述
iot:DeleteJobTemplate	<a href="#">DeleteJobTemplate</a>	jobtemplate	代表刪除任務範本的許可。每次發出刪除任務範本的要求時，就會檢查一次 iot:CreateJobTemplate 許可。
iot:DeleteJobExecution	<a href="#">DeleteJobTemplate</a>	<ul style="list-style-type: none"> <li>job</li> <li>物件</li> </ul>	代表可刪除任務執行的許可。每次發出刪除任務執行的要求時，就會檢查一次 iot:DeleteJobExecution 許可。
iot:DescribeJob	<a href="#">DescribeJob</a>	job	代表描述任務的許可。每次發出描述任務的要求時，就會檢查一次 iot:DescribeJob 許可。
iot:DescribeJobExecution	<a href="#">DescribeJobExecution</a>	<ul style="list-style-type: none"> <li>job</li> <li>物件</li> </ul>	代表描述任務執行的許可。每次發出描述任務執行的要求時，就會檢查一次 iot:DescribeJobExecution 許可。
iot:DescribeJobTemplate	<a href="#">DescribeJobTemplate</a>	jobtemplate	代表描述任務範本的許可。每次發出描述任務範本的要求時，就會檢查一次 iot:DescribeJobTemplate 許可。
iot:DescribeManagedJobTemplate	<a href="#">DescribeManagedJobTemplate</a>	jobtemplate	代表描述受管任務範本的許可。每次發出描述受管任務範本的要求時，就會檢查一次 iot:DescribeManagedJobTemplate 許可。
iot:GetJobDocument	<a href="#">GetJobDocument</a>	job	代表取得任務之任務文件的許可。每次發出取得任務文件的要求時，就會檢查一次 iot:GetJobDocument 許可。
iot:ListJobExecutionsForJob	<a href="#">ListJobExecutionsForJob</a>	job	代表列出任務之任務執行的許可。每次發出列出任務之任務執行的要求時，就會檢查一次 iot:ListJobExecutionsForJob 許可。

政策動作	API 操作	資源類型	描述
iot:ListJobExecutionsForThing	<a href="#">ListJobExecutionsForThing</a>	物件	代表列出任務之任務執行的許可。每次發出列出物件之任務執行的要求時，就會檢查一次 iot:ListJobExecutionsForThing 許可。
iot:ListJobs	<a href="#">ListJobs</a>	無	代表可列出任務的許可。每次發出列出任務的要求時，就會檢查一次 iot:ListJobs 許可。
iot:ListJobTemplates	<a href="#">ListJobTemplates</a>	無	代表可列出任務範本的許可。每次發出列出任務範本的要求時，就會檢查一次 iot:ListJobTemplates 許可。
iot:ListManagedJobTemplates	<a href="#">ListManagedJobTemplates</a>	無	代表可列出受管任務範本的許可。每次發出列出受管任務範本的要求時，就會檢查一次 iot:ListManagedJobTemplates 許可。
iot:UpdateJob	<a href="#">UpdateJob</a>	job	代表可更新任務的許可。每次發出更新任務的要求時，就會檢查一次 iot:UpdateJob 許可。
iot:TagResource	<a href="#">TagResource</a>	<ul style="list-style-type: none"> <li>• job</li> <li>• jobtemplate</li> <li>• 物件</li> </ul>	准許標記指定的資源。
iot:UntagResource	<a href="#">UntagResource</a>	<ul style="list-style-type: none"> <li>• job</li> <li>• jobtemplate</li> <li>• 物件</li> </ul>	准許取消標記指定的資源。

## 基本 IAM 政策範例

以下範例顯示一個 IAM 政策，該政策允許使用者對您的 IoT 物件和物件群組執行以下動作。

在該範例中，替換：

- 與您的 **##** AWS 區域，例如 `us-east-1`。
- 帶有您的 AWS 帳戶號碼的 **## ID**，例如。57EXAMPLE833
- ***thing-group-name*** 使用您要鎖定工作目標的 IoT 物件群組名稱，例如 `FirmwareUpdateGroup`。
- ***thing-name*** 換成您目標任務的 IoT 物件名稱，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/*"
      ]
    }
  ]
}
```

```

    ]
  }
]
}

```

## 以 IP 為基礎的授權的 IAM 政策範例

您可以限制 principals (主體) 從特定 IP 地址對您的控制平面端點進行 API 呼叫。若要指定允許的 IP 地址，請在 IAM 政策的 Condition 元素中使用 [aws:SourceIp](#) 全域條件金鑰。

使用此條件金鑰也可以拒絕其他 AWS 服務人的存取，以代表您進行這些 API 呼叫，例如 AWS CloudFormation。若要允許存取這些服務，請搭配 aws: key 使用 [aws:ViaAWSService](#) 全域條件 SourceIp 金鑰。這可確保來源 IP 地址存取限制僅適用於主體的直接請求。如需詳細資訊，請參閱 [AWS：AWS 根據來源 IP 拒絕存取](#)。

下列範例顯示如何僅允許可對控制平面端點進行 API 呼叫的特定 IP 地址。此 `aws:ViaAWSService` 金鑰為 `true`，允許其他服務代表您進行 API 呼叫。

```

{
  "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"}
    }
  ],
}

```



## 資料平面上的 IAM 政策

資料平面上的 IAM 政策使用 `iotjobsdata:` 字首來授權使用者可以執行的任務 API 操作。在資料平面上，您可以使用 `iotjobsdata:DescribeJobExecution` 政策動作授予使用者使用 [DescribeJobExecution](#) API 的許可。

### Warning

鎖定裝置的 AWS IoT 任務時，不推薦使用在資料平面上使用 IAM 政策。建議您在控制平面上使用 IAM 政策，以便使用者建立和管理任務。在資料平面上，若要授權裝置擷取任務執行並更新執行狀態，請使用 [AWS IoT Core HTTPS 通訊協定的政策](#)。

### 基本 IAM 政策範例

必須授權的 API 操作通常由您輸入 CLI 命令來執行。以下說明使用者執行 `DescribeJobExecution` 操作的範例。

在該範例中，替換：

- 與您的 `##` AWS 區域，例如 `us-east-1`。
- 帶有您的 AWS 帳戶號碼的 `## ID`，例如 `57EXAMPLE833`
- `thing-name` 換成您目標任務的 IoT 物件名稱，例如 `myRegisteredThing`。
- `job-id` 是使用 API 鎖定之任務的唯一識別符。

```
aws iot-jobs-data describe-job-execution \  
  --endpoint-url "https://account-id.jobs.iot.region.amazonaws.com" \  
  --job-id jobID --thing-name thing-name
```

以下顯示授權這個動作的 IAM 政策範例：

```
{  
  "Version": "2012-10-17",  
  "Statement":  
  {  
    "Action": ["iotjobsdata:DescribeJobExecution"],  
    "Effect": "Allow",
```

```

    "Resource": "arn:aws:iot:region:account-id:thing/thing-name",
  }
}

```

### 以 IP 為基礎的授權的 IAM 政策範例

您可以限制 principal (主體) 從特定 IP 地址對資料平面端點進行 API 呼叫。若要指定允許的 IP 地址，請在 IAM 政策的 Condition 元素中使用 [aws:SourceIp](#) 全域條件金鑰。

使用此條件金鑰也可以拒絕其他 AWS 服務人的存取，以代表您進行這些 API 呼叫，例如 AWS CloudFormation。若要允許存取這些服務，請使用 [aws:ViaAWSService](#) 有 `aws:SourceIp` 條件金鑰的全域條件金鑰。這可確保 IP 地址存取限制僅適用於主體直接發出的要求。如需詳細資訊，請參閱 [AWS：AWS 根據來源 IP 拒絕存取](#)。

下列範例顯示如何僅允許可對資料平面端點進行 API 呼叫的特定 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"}
    }
  ],
}

```

下列範例顯示如何限制特定 IP 地址或地址範圍，以便對資料平面端點進行 API 呼叫。

```

{
  "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": ["*"],
  }
],
}

```

### 控制平面和資料平面的 IAM 政策範例

如果您同時在控制平面和資料平面上執行 API 操作，控制平面政策動作必須使用 `iot:` 字首，而且您的資料平面政策動作必須使用 `iotjobsdata:` 字首。

例如 `DescribeJobExecution` API 可以同時在控制平面和資料平面上使用。在控制平面上，[DescribeJobExecution](#) API 用於描述作業執行。在資料平面上，[DescribeJobExecution](#) API 用於取得作業執行的詳細資訊。

以下 IAM 政策授權使用者同時在控制平面和資料平面上使用 `DescribeJobExecution` API。

在該範例中，替換：

- 與您的 `##` AWS 區域，例如 `us-east-1`。
- 帶有您的 AWS 帳戶號碼的 `## ID`，例如 `57EXAMPLE833`。
- `thing-name` 換成您目標任務的 IoT 物件名稱，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/*"
    ]
  }
]
```

## 授權對 IoT 資源進行標記

為了更妥善控制您可以建立、修改或使用的任務和任務範本，您可以將標籤連接到任務或任務範本。標籤也可以幫助您識別所有權，並將其放在計費群組中並附加標籤來指派和分配成本。

當使用者想要標記他們使用或建立的工作 AWS Management Console 或工作範本時 AWS CLI，您的 IAM 政策必須授與使用者權限來標記他們。要授予權限，您的 IAM 政策必須使用 `iot:TagResource` 動作。

### Note

如果您的 IAM 政策不包含 `iot:TagResource` 動作，則任何 [CreateJob](#) 或 [CreateJobTemplate](#) 帶有標籤的內容都會傳回 `AccessDeniedException` 錯誤。

當您想要標記使用或建立的任務或工作範本時 AWS CLI，您的 AWS Management Console IAM 政策必須授予權限來標記它們。要授予權限，您的 IAM 政策必須使用 `iot:TagResource` 動作。

如需標記資源的相關資訊，請參閱 [標記您的 AWS IoT 資源](#)。

## IAM 政策範例

請參閱下列授與標記權限的 IAM 政策範例：

### 範例 1

執行下列命令以建立工作並將其標記至特定環境的使用者。

在此範例中，請取代：

- 與您的## AWS 區域，例如us-east-1。
- 帶有您的 AWS 帳戶號碼的## ID，例如。57EXAMPLE833
- *thing-name*換成您目標任務的 IoT 物件名稱，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
```

這個範例必須使用下列 IAM 政策：

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

## 授權您的裝置安全地使用資料層上的 AWS IoT 工作

若要授權您的裝置與資料平面上的 AWS IoT 工作安全互動，您必須使用 AWS IoT Core 原則。AWS IoT Core 工作的策略是包含策略陳述式的 JSON 文件。這些政策也使用效果、動作以及資源元素，並遵循與 IAM 政策類似的慣例。如需這些元素的詳細資訊，請參閱《IAM 使用者指南》中的 [IAM JSON 政策元素參考](#)。

這些政策可以與 MQTT 和 HTTPS 通訊協定搭配使用，並且必須使用 TCP 或 TLS 相互驗證對裝置進行身分驗證。以下介紹如何跨不同的通訊協定使用這些政策。

**⚠ Warning**

建議您不要使用萬用字元許可，例如 IAM 政策或政 AWS IoT Core 策 "Action": ["iot:\*"] 中。使用萬用字元許可不是推薦的安全最佳實務。如需詳細資訊，請參閱 [AWS IoT 政策過於寬容](#)。

## AWS IoT Core MQTT 通訊協定的政策

AWS IoT Core MQTT 通訊協定的原則授予您使用作業裝置 MQTT API 動作的權限。MQTT API 操作用於處理為任務命令保留的 MQTT 主題。如需有關這些 API 操作的詳細資訊，請參閱 [任務裝置 MQTT API 操作](#)。

MQTT 政策使用政策動作 (例如 `iot:Connect`、`iot:Publish`、`iot:Subscribe` 以及 `iot:Receive`) 以處理任務主題。這些政策可讓您連線至訊息代理程式、訂閱任務 MQTT 主題，以及在您的裝置和雲端之間傳送和接收 MQTT 訊息。如需這些動作的詳細資訊，請參閱《[AWS IoT Core 政策動作](#)》。

如需「AWS IoT 工作」主題的相關資訊，請參閱 [任務主題](#)。

### 基本 MQTT 政策範例

以下範例說明如何使用 `iot:Publish` 和 `iot:Subscribe` 發佈和訂閱任務和任務執行。

在該範例中，替換：

- 與您的 **##** AWS 區域，例如 `us-east-1`。
- 帶有您的 AWS 帳戶號碼的 **## ID**，例如 `57EXAMPLE833`
- *thing-name* 換成您目標任務的 IoT 物件名稱，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"
}
```

## AWS IoT Core HTTPS 通訊協定的政策

AWS IoT Core 資料平面上的原則也可以使用 HTTPS 通訊協定搭配 TLS 驗證機制來授權您的裝置。資料平面上的政策使用 `iotjobsdata:` 字首來授權您的裝置可以執行的任務 API 操作。例如 `iotjobsdata:DescribeJobExecution` 政策動作會授予使用者許可使用 [DescribeJobExecution](#) API。

### Note

資料平面政策動作必須使用 `iotjobsdata:` 字首。在控制平面上，動作必須使用 `iot:` 字首。有關同時使用控制平面和資料平面政策動作時的 IAM 政策範例，請參閱 [控制平面和資料平面的 IAM 政策範例](#)。

## 政策動作

下表顯示授權裝置使用 API 動作的 AWS IoT Core 政策動作和權限清單。如需您可以在資料平面中執行的 API 操作清單，請參閱 [任務裝置 HTTP API](#)。

### Note

任務執行政策動作僅適用於 HTTP TLS 端點。如果您使用 MQTT 端點，必須使用前述所定義的 MQTT 政策動作。

## AWS IoT Core 資料平面上的原則動作

政策動作	API 操作	資源類型	描述
iotjobsdata:DescribeJobExecution	<a href="#">DescribeJobExecution</a>	<ul style="list-style-type: none"> <li>job</li> <li>物件</li> </ul>	代表可擷取任務執行的許可。每次發出擷取任務執行的要求時，就會檢查一次 iotjobsdata:DescribeJobExecution 許可。
iotjobsdata:GetPendingJobExecutions	<a href="#">GetPendingJobExecutions</a>	物件	代表可擷取任務 (對物件來說並非結束狀態) 之清單的許可。每次發出擷取清單的要求時，就會檢查一次 iotjobsdata:GetPendingJobExecutions 許可。
iotjobsdata:StartNextPendingJobExecution	<a href="#">StartNextPendingJobExecution</a>	物件	代表可取得並啟動物件之下一個待定任務執行的許可。也就是將任務執行狀態由 QUEUED 更新為 IN_PROGRESS。每次發出啟動下一個待定任務執行的要求時，就會檢查一次 iot:StartNextPendingJobExecution 許可。
iotjobsdata:UpdateJobExecution	<a href="#">UpdateJobExecution</a>	物件	代表可更新任務執行的許可。每次發出更新任務執行狀態的要求時，就會檢查一次 iot:UpdateJobExecution 許可。

## 基本政策範例

以下顯示原則範例，該 AWS IoT Core 原則授與對任何資源的資料平面 API 作業執行動作的權限。您可以將政策範圍限定為特定資源，例如 IoT 物件。在您的範例中，替換：

- **##**與你 AWS 區域 如 us-east-1.
- 帶有您的 AWS 帳戶號碼的 **## ID**，例如。57EXAMPLE833
- **thing-name** 換成 IoT 物件名稱，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"
    }
  ]
}
```

對於必須使用這些政策的時機，一個範例是當 IoT 裝置使用 AWS IoT Core 政策來存取這些 API 操作的其中之一時，例如以下的 DescribeJobExecution API 範例：

```
GET /things/thingName/jobs/jobId?
executionNumber=executionNumber&includeJobDocument=includeJobDocument&namespaceId=namespaceId
HTTP/1.1
```

## 任務限制

AWS IoT 工作具有與您的服務資源或作業數目上限相對應的服務配額或限制 AWS 帳戶。

### 作用中和並行任務限制

本區段將幫助您深入瞭解作用中和並行任務，以及適用於這些任務的限制。

#### 作用中任務和作用中任務限制

使用 AWS IoT 主控台或 CreateJob API 建立工作時，工作狀態會變更為 IN\_PROGRESS。所有進行中任務均為作用中任務，計入作用中任務限制。這包括正在推展新任務執行的任務，或正在等待裝置完成任務執行的任務。此限制適用於連續任務和快照任務。

#### 並行任務和任務並行限制

正在推出新工作執行項目的進行中工作，或取消先前建立的工作執行項目的工作都是並行工作，而且會計入工作並行限制。AWS IoT 工作可以以每分鐘 1000 部裝置的速度快速推出和取消工作執行。每個

任務為 `concurrent` 且只會短時間計入任務並行限制。在任務執行完成推展或取消後，任務不再是並行任務，不計入任務並行限制。您可以使用任務並行建立大量任務，同時等待裝置完成任務執行。

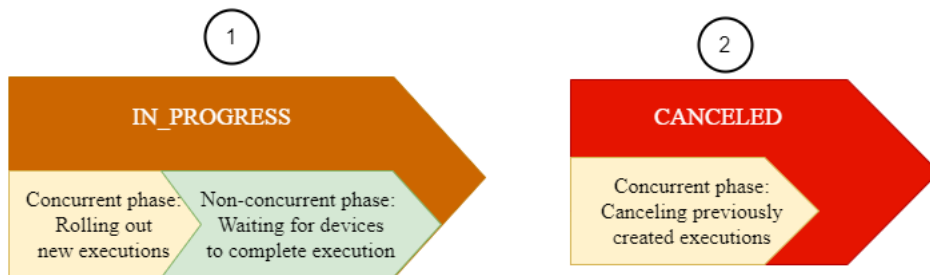
### Note

如果具有選用排程組態的任務和排定在維護時段期間執行的任務文件推展達到選取的 `startTime`，但您達到最大任務並行限制，則該排定的任務將會移至 `CANCELED` 的狀態。

若要判斷工作是否同時進行，您可以從 AWS IoT 主控台或使用 `DescribeJob` 或 `ListJob` API 使用工作的 `IsConcurrent` 屬性。此限制適用於連續任務和快照任務。

若要檢視您的作用中工作和 AWS IoT 工作並行限制以 AWS 帳戶 及其他工作配額，並要求提高限 [AWS IoT 制](#)，請參閱 [AWS 一般參考](#)

下圖顯示如何將任務並行套用至進行中的任務以及取消中的任務。




### Note

具有選擇性 `SchedulingConfig` 的新任務將維持 `SCHEDULED` 的初始狀態，並在達到選取的 `startTime` 時更新為 `IN_PROGRESS`。在具有選擇性 `SchedulingConfig` 的新任務達到所選 `startTime` 並更新為 `IN_PROGRESS` 之後，它會計入作用中任務限制以及任務並行限制。狀態為 `SCHEDULED` 的任務將計入作用中任務限制，但不會計入任務並行限制。

下表顯示套用至作用中和並行任務的限制，以及任務狀態的並行和非並行階段。

## 作用中和並行任務限制

任務狀態	Phase (階段)	作用中任務限制	任務並行限制
SCHEDULED	非並行階段：AWS IoT 工作會等待排定 <code>startTime</code> 的工作，開始傳送至您裝置的工作執行通知。此階段中的任務僅計入作用中任務限制，且 <code>IsConcurrent</code> 屬性設置為 <code>false</code> 。	適用	不適用
IN_PROGRESS	並行階段：AWS IoT 工作接受建立工作的請求，並開始向您的裝置推出工作執行通知。此階段中的任務為並行任務，如同 <code>IsConcurrent</code> 屬性設置為 <code>true</code> 所示，並計入作用中任務和任務並行限制。	適用	適用
	非並行階段：AWS IoT 作業會等待裝置回報其工作執行的結果。此階段中的任務僅計入作用中任務限制，且 <code>IsConcurrent</code> 屬性設置為 <code>false</code> 。	適用	不適用
Canceled	並行階段：AWS IoT 工作接受取消工作的請求，並開始取消先前為您的裝置建立的工作執行。此階段中的任務為並行任務，且 <code>IsConcurrent</code> 屬性設置為 <code>true</code> 。一旦取消任務和任務執行，任務將不再是並行任務，且不計入任務並行限制。	不適用	適用

 Note

週期性維護時段的最長持續時間為 23 小時 50 分鐘。

# AWS IoT 安全隧道

在遠端網站的受限制防火牆後面部署裝置時，您需要一種方法來存取這些裝置，進行疑難排解、組態更新及其他作業任務。使用安全通道，透過由管理的安全連線，與遠端裝置建立雙向通訊。AWS IoT安全通道不需要更新現有的輸入防火牆規則，因此您可以在遠端網站保留防火牆規則所提供的相同安全層級。

例如，位於幾百英里外工廠的感應器裝置在測量工廠溫度時遇到麻煩。您可以使用安全通道，開啟並快速啟動該感應器裝置的工作階段。找出問題之後（例如，不當的組態檔），您可以重設檔案並透過相同的工作階段重新啟動感應器裝置。相較於傳統的故障診斷（例如，派遣技術人員到工廠調查感應器裝置），安全通道可減少事故回應和回復時間，以及操作成本。

## 什麼是安全通道？

使用安全通道存取部署在遠端站點連接埠受限防火牆後的裝置。您可使用 AWS 雲端，從作為來源裝置的筆記型電腦或桌上型電腦連線至目的地裝置。來源和目的地會使用在每個裝置上執行的開放原始碼本機代理進行通訊。本機 Proxy 會使用防火牆允許的 AWS 雲端 開放連接埠（通常為 443）與通訊。透過通道傳輸的資料會使用 Transport Layer Security (TLS) 加密。

### 主題

- [安全通道概念](#)
- [安全通道運作方式](#)
- [安全通道生命週期](#)

## 安全通道概念

與遠端裝置建立通訊時，安全通道使用以下術語。如需安全通道運作方式的資訊，請參閱 [安全通道運作方式](#)。

### 用戶端存取字符 (CAT)

建立新通道時，由安全通道產生的一對字符。CAT 是由來源和目標裝置用來連線至安全通道服務。CAT 只能用於連接通道一次。若要重新連線至通道，請使用 [RotateTunnelAccessToken](#) API 作業或 [rotate-tunnel-access-token](#) CLI 命令輪換用戶端存取權杖。

## 用戶端字符

用戶端產生的唯一值，AWS IoT 安全通道可用於所有後續重試連線至相同通道的安全通道。此欄位為選用欄位。如果未提供用戶端字符，則用戶端存取字符 (CAT) 只能用於同一通道一次。使用相同 CAT 的後續連線嘗試會被拒絕。如需有關使用用戶端權杖的詳細資訊，請參閱中的[本機 Proxy 參考實作 GitHub](#)。

## 目標應用程式

在目標裝置上執行的應用程式。例如，目標應用程式可以是一個 SSH 協助程式，用於使用安全通道建立 SSH 工作階段。

## 目標裝置

您要存取的遠端裝置。

## 裝置代理程式

連接至 AWS IoT 裝置閘道並透過 MQTT 接聽新通道通知的 IoT 應用程式。如需詳細資訊，請參閱[IoT Agent Snippet](#)。

## 本機代理

在來源和目標裝置上執行，並在安全通道與裝置應用程式之間轉送資料串流的軟體代理。本機代理可以在源模式或目標模式下執行。如需詳細資訊，請參閱[本機代理](#)。

## 來源裝置

此裝置通常是筆記型電腦或桌上型電腦，操作人員用來啟動目標裝置的工作階段。

## 通道

一種邏輯通路 AWS IoT，可在來源裝置與目標裝置之間進行雙向通訊。

## 安全通道運作方式

下面顯示了安全通道如何在來源裝置和目的地裝置之間建立連線。如需用戶端存取字符 (CAT) 等不同術語的資訊，請參閱[安全通道概念](#)。

### 1. 開啟通道

[若要開啟通道以啟動遠端目標裝置的工作階段，您可以使用開AWS CLI 放隧道命令或 API。AWS Management Console OpenTunnel](#)

### 2. 下載用戶端存取字符對

開啟通道後，您可以下載來源和目的地的用戶端存取字符 (CAT)，並將其儲存在來源裝置上。在啟動本機代理之前，必須擷取 CAT 並立即儲存。

### 3. 以目的地模式啟動本機代理

已安裝並正在目的地裝置上執行的 IoT 代理將訂閱保留的 MQTT 主題 `$aws/things/thing-name/tunnels/notify` 並將收到 CAT。在這裡，*Thing-name* 是您為目的地創建的 AWS IoT 事物的名稱。如需詳細資訊，請參閱 [安全通道主題](#)。

然後，IoT 代理使用 CAT 在目的地模式下啟動本機代理，並在通道的目的地端設定連線。如需詳細資訊，請參閱 [IoT Agent Snippet](#)。

### 4. 以來源模式啟動本機代理

通道開啟之後，AWS IoT Device Management 提供您可以在來源裝置上下載的來源 CAT。您可以使用 CAT 在來源模式下啟動本機代理，然後連線通道的來源端。如需本機代理的詳細資訊，請參閱 [本機代理](#)。

### 5. 開啟 SSH 工作階段

當通道兩端都連線時，您可以使用來源端的本機代理啟動 SSH 工作階段。

如需如何使用開啟通道和啟 AWS Management Console 動 SSH 工作階段的詳細資訊，請參閱 [開啟通道並啟動遠端裝置的 SSH 工作階段](#)。

下列影片介紹了安全通道的運作方式，並引導您完成設定與 Raspberry Pi 裝置的 SSH 工作階段的程序。

## 安全通道生命週期

通道可以具有 OPEN 或 CLOSED 狀態。與通道的連線狀態可為 CONNECTED 或 DISCONNECTED。下面顯示了不同的通道和連線狀態的運作方式。

1. 當您開啟通道時，它的狀態為 OPEN。通道的來源和目標連線狀態會設定為 DISCONNECTED。
2. 當裝置 (來源或目標) 連線至通道時，對應的連線狀態會變更為 CONNECTED。
3. 當裝置與通道中斷連線而通道狀態維持為 OPEN 時，對應的連線狀態會變回 DISCONNECTED。只要通道仍然保持 OPEN，裝置就可以重複連線至通道，以及與通道中斷連線。

**Note**

用戶端存取字符 (CAT) 只能用於連接通道一次。若要重新連線至通道，請使用 [RotateTunnelAccessToken](#) API 作業或 [rotate-tunnel-access-token](#) CLI 命令輪換用戶端存取權杖。如需範例，請參閱 [透過旋轉用戶端存取權杖來解決 AWS IoT 安全通道連線問題](#)。

- 當您呼叫 `CloseTunnel` 或通道維持 OPEN 的時間超過 `MaxLifetimeTimeout` 值時，通道的狀態就會變成 CLOSED。您可以在呼叫 `OpenTunnel` 時設定 `MaxLifetimeTimeout`。如果您未指定一值，則 `MaxLifetimeTimeout` 會預設為 12 小時。

**Note**

狀態為 CLOSED 時，無法重新開啟通道。

- 通道顯示時，您可以呼叫 `DescribeTunnel` 和 `ListTunnels` 來檢視通道中繼資料。在刪除通道之前，至少可以在 AWS IoT 控制台中看到三個小時。

## AWS IoT 安全隧道教程

AWS IoT 安全通道可協助客戶透過管理的安全連線，與位於防火牆後方的遠端裝置建立雙向通訊。  
AWS IoT

若要示範 AWS IoT 安全通道，請在上使用我們的 [AWS IoT 安全通道](#) 示範。GitHub

下列教學課程將會協助您了解如何開始及使用安全通道。您將學到如何：

- 使用快速設定和手動設定方法建立安全通道以存取遠端裝置。
- 在使用手動設定方法時設定本機代理，並連線至通道以存取目的地裝置。
- 無需設定本機代理即可透過瀏覽器使用 SSH 進入遠端裝置。
- 轉換使用 AWS CLI 或使用手動設定方法建立的通道，以使用快速設定方法。

### 本節中的教學課程

本節中的教學課程著重於使用 AWS Management Console 和 AWS IoT API 參考建立通道。在 AWS IoT 主控台中，您可以從「[通道中樞](#)」[頁面](#)或您建立之物件的[詳細資訊頁面](#)建立通道。如需詳細資訊，請參閱 [AWS IoT 控制台中的隧道創建方法](#)。

本節教學課程如下所示：

- [開啟通道並使用瀏覽器型 SSH 來存取遠端裝置](#)

本教學課程說明如何使用快速設定方法從 [Tunnels hub](#) (通道中樞) 頁面開啟通道。您還將學習如何使用基於瀏覽器的 SSH 使用控制台中的上下文命令行界面訪問遠程設備。AWS IoT

- [使用手動設定開啟通道並連線至遠端裝置](#)

本教學課程說明如何使用手動設定方法從 [Tunnels hub](#) (通道中樞) 頁面開啟通道。您也將了解如何從來源裝置中的終端設定和啟動本機代理並連接到通道。

- [開啟遠端裝置的通道並使用瀏覽器型 SSH](#)

本教學課程將介紹如何從您所建立物件的詳細資訊頁面開啟通道。您將了解如何建立新的通道和使用現有通道。現有通道會對應至為裝置建立的最新開放通道。您也可以使用瀏覽器型 SSH 來存取遠端裝置。

## AWS IoT 安全隧道教程

- [開啟通道並啟動遠端裝置的 SSH 工作階段](#)
- [開啟遠端裝置的通道並使用瀏覽器型 SSH](#)

## 開啟通道並啟動遠端裝置的 SSH 工作階段

在這些教學課程中，您將了解如何從遠端存取防火牆後方的裝置。您無法在裝置中啟動直接 SSH 工作階段，因為防火牆會封鎖所有傳入流量。教學課程會示範如何開啟通道，然後使用該通道啟動遠端裝置的 SSH 工作階段。

### 教學課程的先決條件

執行教學課程的先決條件可能不盡相同，取決於您是使用手動或是快速設定方法來開啟通道和存取遠端裝置。

#### Note

對於這兩種設定方法，您都必須允許連接埠 443 的傳出流量。

- 如需了解快速設定方法自學課程之先決條件，請參閱 [快速設定方法的先決條件](#)。



- 如需了解手動設定方法自學課程之先決條件，請參閱 [手動設定方法的先決條件](#)。如果您使用此設定方法，則必須在來源裝置上設定本機代理。若要下載本機 Proxy 原始程式碼，請參閱 [上的本機 Proxy 參考實作 GitHub](#)。

## 通道設定方法

在這些教學課程中，您將了解如何以手動和快速設定方法開啟通道和連線至遠端裝置。下表顯示設定方法之間的差異。建立通道之後，您就可以使用瀏覽器內的命令列介面，透過 SSH 存取遠端裝置。如果誤置權杖或通道連線中斷，您可以傳送新的存取權杖以重新連線至通道。

### 快速和手動設定方法

條件	快速設定	手動設定
通道建立	使用預設、可編輯的組態來建立新的通道。若要存取遠端裝置，您只能使用 SSH 作為目的地服務。	透過手動指定通道組態來建立通道。您可以使用此方法，透過 SSH 以外的服務連線至遠端裝置。
存取權杖	若在建立通道時指定物件名稱，目的地存取權杖會自動傳遞至 <a href="#">保留 MQTT 主題</a> 上的裝置。您無需透過來源裝置下載或管理權杖。	您必須透過來源裝置手動下載和管理權杖。若在建立通道時指定物件名稱，目的地存取權杖會自動傳遞至 <a href="#">保留 MQTT 主題</a> 上的遠端裝置。
本機代理	系統會自動為您設定 Web 型本機代理，以便與裝置進行互動。您無需手動設定本機代理。	您必須手動設定並啟動本機代理。若要設定本機 Proxy，您可以使用 AWS IoT 裝置用戶端，或在 <a href="#">上下載本機 Proxy 參考實作 GitHub</a> 。

## AWS IoT 控制台中的隧道創建方法

本節中的教學課程將說明如何使用 AWS Management Console 和 [OpenTunnelAPI](#) 建立通道。如果您在建立通道時設定目的地，AWS IoT 安全通道會透過 MQTT 將目的地用戶端存取權杖傳遞至遠端裝置，以及保留的 MQTT 主題)。\$aws/things/RemoteDeviceA/tunnels/notify在收到 MQTT 訊息後，遠端裝置上的 IoT 代理程式會以目的地模式啟動本機代理。如需詳細資訊，請參閱 [預留主題](#)。

**Note**

如果您想要透過其他方法將目標用戶端存取字符交付給遠端裝置，則可以省略目標組態。如需詳細資訊，請參閱 [設定遠端裝置並使用 IoT 代理程式](#)。

在 AWS IoT 主控台中，您可以使用下列其中一種方法建立通道。如需更多有關使用這些方法建立通道的輔助教學課程資訊，請參閱 [本節中的教學課程](#)。

- [通道中樞](#)

建立通道時，您可以指定要使用快速設定或手動設定方法來建立通道，並提供選擇性的通道組態詳細資訊。組態詳細資訊也包括目的地裝置的名稱，以及您要用來連線到裝置的服務。建立通道後，您可以在瀏覽器中使用 SSH 或在 AWS IoT 主控台外開啟終端機以存取遠端裝置。

- 物件詳細資訊頁面

建立通道時，除了選擇設定方法並提供任何選擇性的通道組態詳細資訊之外，您還可以指定是否要使用最新開放通道或為裝置建立新的通道。您無法編輯現有通道的組態詳細資訊。您可以使用快速設定方法輪換存取權杖，並在瀏覽器中使用 SSH 進入遠端裝置。若要使用此方法開啟通道，您必須在 AWS IoT 登錄中建立 IoT 物件 (例如 RemoteDeviceA)。如需詳細資訊，請參閱 [在登 AWS IoT 錄中註冊裝置](#)。

### 本節中的教學課程

- [開啟通道並使用瀏覽器型 SSH 來存取遠端裝置](#)
- [使用手動設定開啟通道並連線至遠端裝置](#)

## 開啟通道並使用瀏覽器型 SSH 來存取遠端裝置

您可以使用快速設定或手動設定方法來建立通道。本教學課程說明如何使用快速設定方法開啟通道，並使用瀏覽器型 SSH 連線至遠端裝置。如需查看如何使用手動設定方法來開啟通道的範例，請參閱 [使用手動設定開啟通道並連線至遠端裝置](#)。

您可以透過快速設定方法，使用可編輯的預設組態建立新的通道。已為您設定 Web 型本機代理，並使用 MQTT 將存取權杖自動傳遞到您的遠端目的地裝置。建立通道後，您可以使用主控台內的命令列介面開始與遠端裝置互動。

若採用快速設定方法，您必須使用 SSH 作為目的地服務才能存取遠端裝置。如需不同設定方法的詳細資訊，請參閱 [通道設定方法](#)。

## 快速設定方法的先決條件

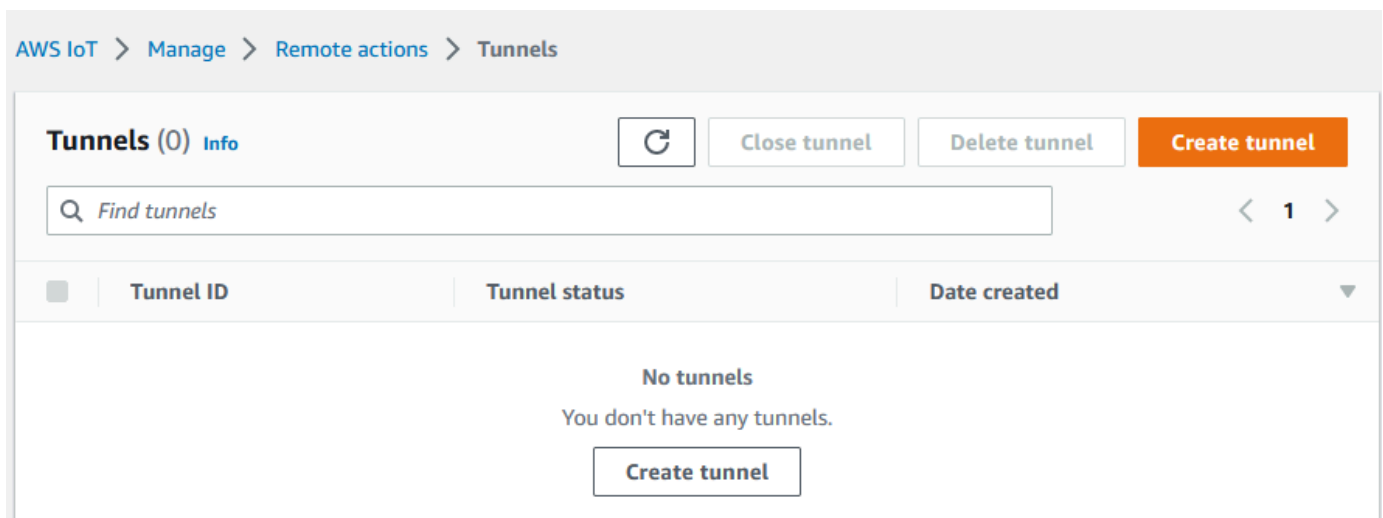
- 遠端裝置所在的防火牆必須允許連接埠 443 上的傳出流量。您建立的通道將使用此通訊埠連線至遠端裝置。
- 您有一個 IoT 裝置代理程式 (請參閱[IoT Agent Snippet](#)) 在連線至裝置閘道的遠端 AWS IoT 裝置上執行，且已設定 MQTT 主題訂閱。如需詳細資訊，請參閱[將裝置連線至 AWS IoT 裝置閘道](#)。
- 您必須具有一個在遠端裝置上執行的 SSH 常駐程式。

## 開啟通道

您可以使用 AWS Management Console AWS IoT API 參考或 AWS CLI。您可以選擇設定目的地名稱，但這並非本教學課程的必要步驟。如果您設定目的地，安全通道會使用 MQTT 自動將存取權杖傳遞至遠端裝置。如需詳細資訊，請參閱[AWS IoT 控制台中的隧道創建方法](#)。

若要使用主控台中開啟通道

1. 前往 [AWS IoT 主控台的 Tunnels hub \(通道中樞\)](#)，然後選擇 Create job (建立通道)。



2. 在本教學課程中，請選擇 Quick setup (快速設定) 作為通道建立方法，然後選擇 Next (下一步)。

### Note

如果您從建立物件的詳細資訊頁面建立安全通道，您可以選擇要建立新通道或使用現有通道。如需詳細資訊，請參閱[開啟遠端裝置的通道並使用瀏覽器型 SSH](#)。

### Setup method

Quick setup (SSH)

Manual setup

#### Quick setup (SSH)

Use quick setup to create a new tunnel with default, editable tunnel configurations. When you use quick setup:

- A web-based local proxy will be automatically configured for you to SSH into the remote device.
- The destination access token will be automatically delivered to your device on the [reserved MQTT topic](#), if a thing name is specified.

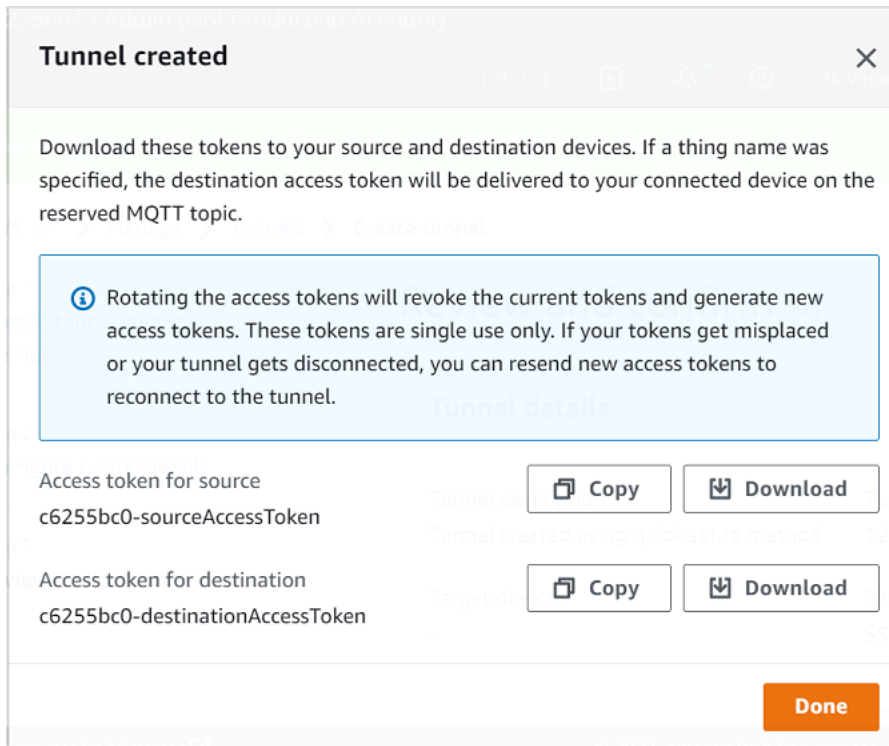
3. 檢閱並確認通道組態詳細資訊。若要建立通道，請選擇 **Confirm and create (確認並建立)**。如果您要編輯這些詳細資訊，請選擇 **Previous (上一步)** 返回上一頁，然後確認並建立通道。

#### Note

使用快速設定時，將無法編輯服務名稱。您必須使用 SSH 作為服務。

4. 若要建立通道，請選擇 **Done (完成)**。

在本教學課程中，您無需下載來源或目的地存取權杖。這些權杖只能用於連接通道一次。如果您的通道中斷連線，您可以產生新權杖並將其傳送到遠端裝置，以重新連線至通道。如需詳細資訊，請參閱 [重新傳送通道存取權杖](#)。



若要使用主控台中開啟通道

若要開啟新的通道，您可以使用 [OpenTunnelAPI](#) 作業。

#### Note

您只能從 AWS IoT 主控台使用快速設定方法建立通道。當您使用 AWS IoT API 參考 API 或時 AWS CLI，它將使用手動設定方法。您可以開啟您所建立的現有通道，然後變更通道的設定方法，即可使用快速設定。如需詳細資訊，請參閱 [開啟現有的通道並使用瀏覽器型 SSH](#)。

關於如何執行此 API 操作的範例如下所示。如果您要指定物件名稱和目的地服務，也可以選擇使用 `DestinationConfig` 參數。如需示範如何使用這個參數的範例，請參閱 [為遠端裝置開啟新通道](#)。

```
aws iotsecuretunneling open-tunnel
```

執行此命令會建立新的通道，並提供您來源和目的地存取權杖。

```
{  
  "tunnelId": "01234567-89ab-0123-4c56-789a01234bcd",
```

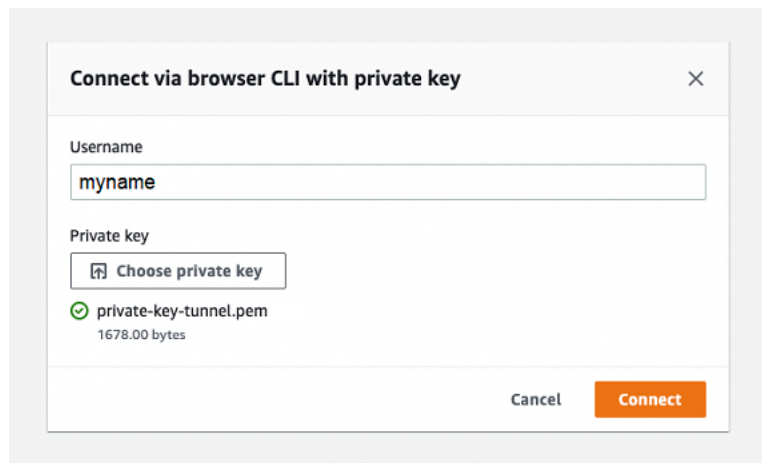
```
"tunnelArn": "arn:aws:iot:us-east-1:123456789012:tunnel/01234567-89ab-0123-4c56-789a01234bcd",
"sourceAccessToken": "<SOURCE_ACCESS_TOKEN>",
"destinationAccessToken": "<DESTINATION_ACCESS_TOKEN>"
}
```

## 使用瀏覽器型 SSH

在使用快速設定方法建立通道並且目的地裝置已連線至通道後，您就可以使用瀏覽器型 SSH 存取遠端裝置。透過瀏覽器型 SSH，您可以在主控台內的上下文命令列介面中輸入命令，即可直接與遠端裝置進行通訊。此功能可讓您更輕鬆地與遠端裝置互動，因為您不必在主控台外開啟終端或設定本機代理。

### 若要使用瀏覽器型 SSH

1. 前往 [AWS IoT 主控台的 Tunnels hub \(通道中樞\)](#)，然後選擇您建立的通道以檢視其詳細資訊。
2. 展開 Secure Shell (SSH) 區段，然後選擇 Connect (連線)。
3. 選擇是否要提供您的使用者名稱和密碼來驗證 SSH 連線，或者如需更安全的驗證，則可以使用裝置的私有金鑰。如果您使用私密金鑰進行驗證，您可以使用 PEM (PKCS#1, PKCS#8) 和 OpenSSH 格式的 RSA、DSA、ECDSA (nistp-\*) 和 ED25519 金鑰類型。
  - 若要使用使用者名稱和密碼進行連線，請選擇 Use password (使用密碼)。然後，您可以輸入使用者名稱和密碼，並開始使用瀏覽器內的 CLI。
  - 若要使用目的地裝置的私有金鑰進行連線，請選擇 Use private key (使用私有金鑰)。指定您的使用者名稱並上傳裝置的私有金鑰檔案，然後選擇 Connect (連線) 以開始使用瀏覽器內 CLI。



在通過 SSH 連接驗證後，您即可迅速開始使用瀏覽器 CLI 輸入命令並與裝置進行互動，因為系統已為您設定本機代理。

### ▼ Comand line interface [Info](#)

```
const [preferences, setPreferences] = React.useState(
  undefined
);
const [loading, setLoading] = React.useState(false);
return (
  <CodeEditor
    ace={ace}
    language="javascript"
    value="const pi = 3.14;"
    preferences={preferences}
    onPreferencesChange={e => setPreferences(e.detail)}
    loading={loading}
  />
);
```

如果瀏覽器 CLI 在通道持續時間之後保持開啟狀態，則可能會發生逾時，導致命令列介面中斷連線。您可以複製通道並啟動另一工作階段，以便在主控台本身內部與遠端裝置互動。

### 解決使用瀏覽器型 SSH 時發生的問題

以下介紹如何解決使用瀏覽器型 SSH 時可能發生的一些問題。

- 系統顯示錯誤，而不是命令列介面

您可能會看到錯誤，原因是目的地裝置中斷連線。您可以選擇 **Generate new access tokens** (產生新的存取權杖) 來產生新的存取權杖，並使用 MQTT 將權杖傳送到遠端裝置。新的權杖可以用來重新連線至通道。重新連線至通道會清除歷程記錄並重新整理命令列工作階段。

- 使用私有金鑰進行驗證時，您會看到通道中斷連線的錯誤

您可能會看到錯誤訊息，原因可能是目的地裝置尚未接受您的私有金鑰。若要解決此錯誤，請檢查您上傳用於進行驗證的私有金鑰檔案。如果仍然看到錯誤，請檢查您的裝置日誌。您也可以向遠端裝置發送新的存取權杖，藉此嘗試重新連線至通道。

- 您的通道已在使用工作階段時關閉

如果您的通道因保持開啟超過指定持續時間而關閉，則您的命令列工作階段可能會中斷連線。通道一旦關閉就不能重新開放。若要重新連線，您必須開啟另一個連往裝置的通道。

您可以複製通道，以建立與已關閉通道相同組態的新通道。您可以從 AWS IoT 主控台複製封閉的通道。若要複製通道，請選擇已關閉的通道以檢視其詳細資訊，然後選擇 Duplicate tunnel (複製通道)。指定您要使用的通道持續時間，然後建立新通道。

## 清除

### • 關閉通道

我們建議您在使用完通道後關閉通道。如果開放時間超過指定的通道持續時間，通道也可能會關閉。通道一旦關閉就不能重新開放。您仍然可以選擇已關閉的通道，然後選擇 Duplicate tunnel (複製通道) 來複製通道。指定您要使用的通道持續時間，然後建立新通道。

- 若要從 AWS IoT 主控台關閉個別通道或多個通道，請移至 [Tunnels hub](#) (通道中樞)，選擇您要關閉的通道，然後選擇 Close tunnel (關閉通道)。
- 若要使用 AWS IoT API 參考 API 關閉個別通道或多個通道，請使用 [CloseTunnelAPI](#)。

```
aws iotsecuretunneling close-tunnel \  
  --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

### • 刪除通道

您可以從您的 AWS 帳戶。

#### Warning

刪除動作為永久性動作，且無法還原。

- 若要從 AWS IoT 主控台刪除個別通道或多個通道，請移至 [Tunnels hub](#) (通道中樞)，選擇您要刪除的通道，然後選擇 Delete tunnel (刪除通道)。
- 若要使用 AWS IoT API 參考 API 刪除個別通道或多個通道，請使用 [CloseTunnelAPI](#)。使用 API 時，請將 delete 旗標設定為 true。

```
aws iotsecuretunneling close-tunnel \  
  --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd" \  
  --delete true
```



## 使用手動設定開啟通道並連線至遠端裝置

開啟通道時，您可以選擇以快速設定或手動設定方法來開啟通道進入遠端裝置。本教學課程說明如何使用手動設定方法開啟通道，並設定及啟動本機代理以連線至遠端裝置。

若使用手動設定方法，則必須在建立通道時手動指定通道組態。創建隧道後，您可以在瀏覽器中使用 SSH 或在 AWS IoT 控制台外打開終端。本教學課程說明如何使用主控台外部的終端存取遠端裝置。您也將了解如何設定本機代理，然後連線至本機代理以與遠端裝置進行互動。若要連線至本機代理，您必須在建立通道時下載來源存取權杖。

在此設定方法中，您可以使用 SSH 以外的服務，例如透過 FTP 連線至遠端裝置。如需不同設定方法的詳細資訊，請參閱 [通道設定方法](#)。

### 手動設定方法的先決條件

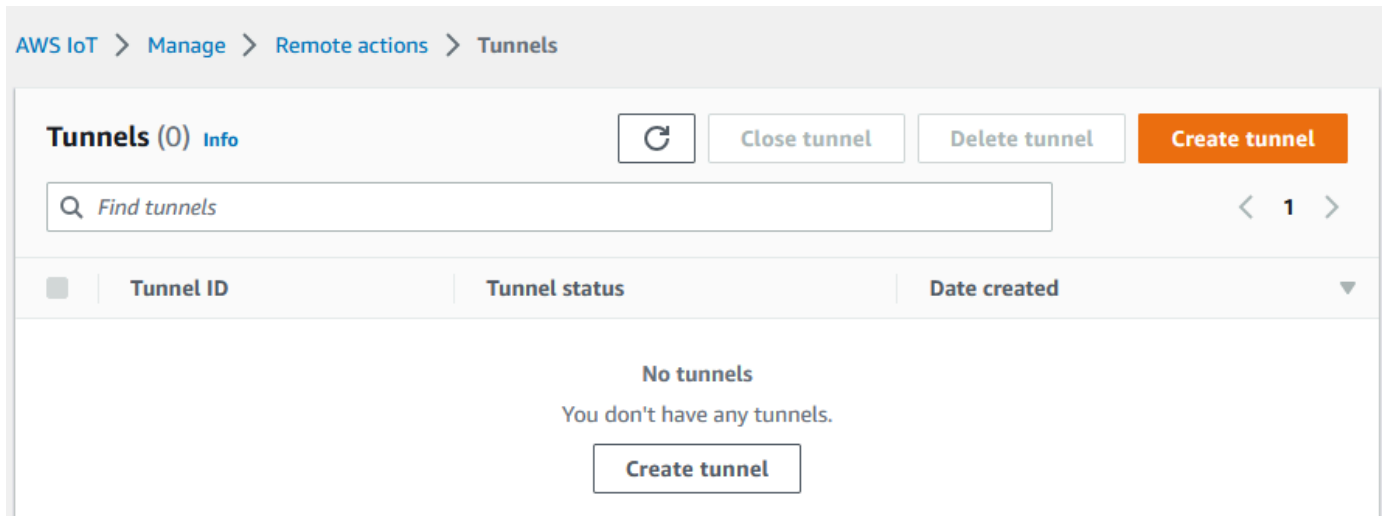
- 遠端裝置所在的防火牆必須允許連接埠 443 上的傳出流量。您建立的通道將使用此通訊埠連線至遠端裝置。
- 您有一個 IoT 裝置代理程式 (請參閱 [IoT Agent Snippet](#)) 在連線至裝置開道的遠端 AWS IoT 裝置上執行，且已設定 MQTT 主題訂閱。如需詳細資訊，請參閱 [將裝置連線至 AWS IoT 裝置開道](#)。
- 您必須具有一個在遠端裝置上執行的 SSH 常駐程式。
- 您已經從中下載了本地代理源代碼，[GitHub](#) 並為您選擇的平台構建了它。在本教學課程中，我們將建置的本機代理可執行檔案稱作 localproxy。

### 開啟通道

您可以使用 AWS Management Console AWS IoT API 參考或 AWS CLI。您可以選擇設定目的地名稱，但這並非本教學課程的必要步驟。如果您設定目的地，安全通道會使用 MQTT 自動將存取權杖傳遞至遠端裝置。如需詳細資訊，請參閱 [AWS IoT 控制台中的隧道創建方法](#)。

### 如要在主控台中開啟通道

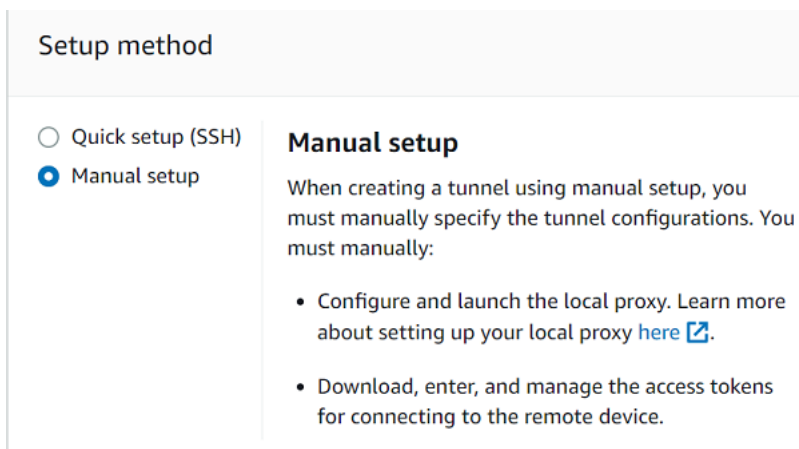
1. 前往 [AWS IoT 主控台的 Tunnels hub \(通道中樞\)](#)，然後選擇 Create job (建立通道)。



2. 在本教學課程中，請選擇 Manual setup (手動設定) 作為通道建立方法，然後選擇 Next (下一步)。如需了解如何使用快速設定方法建立通道，請參閱 [開啟通道並使用瀏覽器型 SSH 來存取遠端裝置](#)。

#### Note

如果從建立物件的詳細資訊頁面建立安全通道，您可以選擇要建立新通道或使用現有通道。如需詳細資訊，請參閱 [開啟遠端裝置的通道並使用瀏覽器型 SSH](#)。



3. (選擇性) 輸入通道的組態設定。您也可以略過此步驟，繼續進行下一個步驟來建立通道。

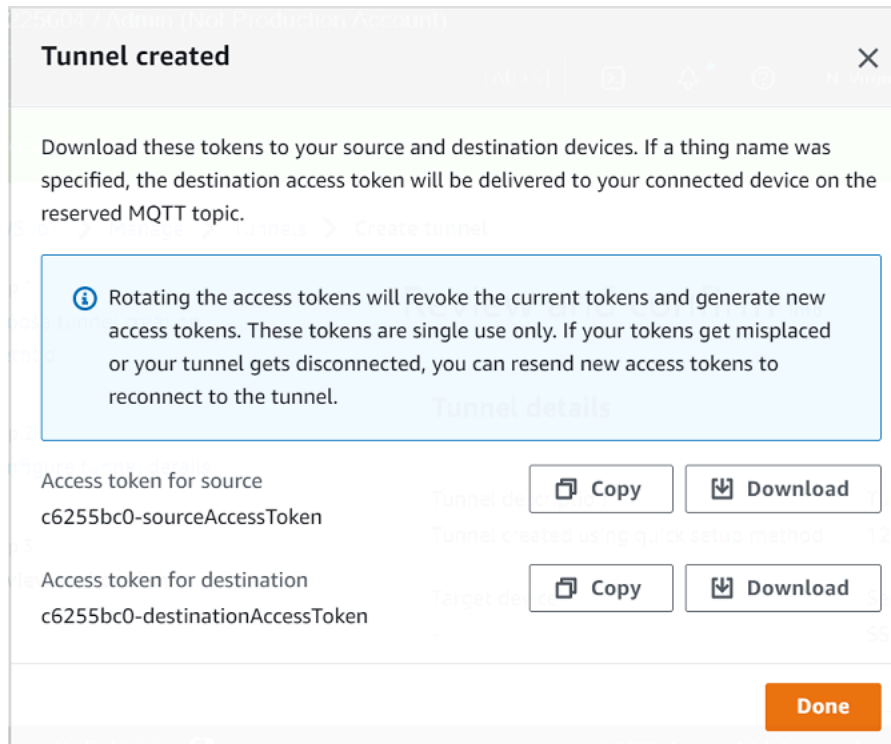
輸入通道描述、通道逾時持續時間和資源標籤作為索引鍵值對，以協助您識別資源。在本教學課程中，您可以略過目的地組態。

**Note**

費用不會依據通道保持開放的持續時間進行計價。只有在建立新通道時會產生費用。如需定價資訊，請參閱 [AWS IoT Device Management 定價](#) 中的安全通道。

4. 下載用戶端存取權杖，然後選擇 Done (完成)。選擇 Done (完成) 後，字符將無法下載。

這些權杖只能用於連接通道一次。如果您誤置權杖或通道中斷連線，您可以產生新權杖並將其傳送到遠端裝置，以重新連線至通道。



若要使用主控台中開啟通道

若要開啟新的通道，您可以使用 [OpenTunnel](#) API 作業。您也可以使用 API 指定其他組態，例如通道持續時間和目的地組態。

```
aws iotsecuretunneling open-tunnel \  
  --region us-east-1 \  
  --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
```

執行此命令會建立新的通道，並提供您來源和目的地存取權杖。

```
{
  "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>"
}
```

## 重新傳送通道存取權杖

建立通道時取得的權杖，只能用來連接通道一次。如果您放錯了訪問令牌或隧道斷開連接，則可以使用 MQTT 重新發送新的訪問令牌到遠程設備，無需額外付費。AWS IoT 安全隧道將撤銷當前令牌並返回用於重新連接到隧道的新訪問令牌。

### 從主控台輪換權杖

1. 前往 [AWS IoT 主控台的「通道」中樞](#)，然後選擇您建立的通道。
2. 在通道詳細資訊頁面中，選擇 Generate new access tokens (產生新存取權杖)，然後選擇 Next (下一步)。
3. 請為您的通道下載新的存取權杖，然後選擇 Done (完成)。這些權杖只能使用一次。如果誤置這些權杖或通道連線中斷，您可以重新傳送新的存取權杖。

**Tokens rotated** ×

Download these tokens to your source and destination devices. If a thing name was specified, the destination access token will be delivered to your connected device on the reserved MQTT topic.

**ⓘ** Rotating the access tokens will revoke the current tokens and generate new access tokens. These tokens are single use only. If your tokens get misplaced or your tunnel gets disconnected, you can resend new access tokens to reconnect to the tunnel.

Access token for source Source Copy Download  
c6255bc0-sourceAccessToken

Access token for destination Destination Copy Download  
c6255bc0-destinationAccessToken

Services Tunnel status Done

## 使用 API 輪換存取權杖

要旋轉隧道訪問令牌，您可以使用 [RotateTunnelAccessToken](#) API 操作撤銷當前令牌並返回新的訪問令牌以重新連接到隧道。例如，下列指令會輪換目的地裝置的存取權杖：*RemoteThing1*。

```
aws iotsecuretunneling rotate-tunnel-access-token \  
  --tunnel-id <tunnel-id> \  
  --client-mode DESTINATION \  
  --destination-config thingName=<RemoteThing1>,services=SSH \  
  --region <region>
```

執行此命令會產生新的存取權杖，如下所示。如果裝置代理程式設定正確，會使用 MQTT 將權杖傳遞到裝置以連接通道。

```
{  
  "destinationAccessToken": "destination-access-token",  
  "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"  
}
```

如需說明如何及何時輪換存取權杖的範例，請參閱 [透過旋轉用戶端存取權杖來解決 AWS IoT 安全通道連線問題](#)。

## 配置並啟動本機代理

若要連線至遠端裝置，請在筆記型電腦上開啟終端，然後設定並啟動本機代理。本機 Proxy 會透過安全連線使用安全通道，傳輸來源裝置上執行的應用程式所傳送的資料。WebSocket 您可以從下載本地代理源代理 [GitHub](#)。

設定完成本機代理後，請複製來源用戶端存取權杖，並使用它在來源模式下啟動本機代理。以下展示使用範例命令來啟動本機代理的步驟。在下列命令中，本機代理會設定為在連接埠 5555 上接聽新連線。在此命令中：

- `-r` 指定 AWS 區域，此區域必須與建立隧道的相同區域。
- `-s` 指定代理應該連線的連接埠。
- `-t` 指定用戶端權杖文字。

```
./localproxy -r us-east-1 -s 5555 -t source-client-access-token
```

若執行此命令，系統會以來源模式啟動本機代理。如果在執行命令後收到下列錯誤，請設定 CA 路徑。如需詳細資訊，請參閱[開啟安全通道本機 Proxy](#)。GitHub

```
Could not perform SSL handshake with proxy server: certificate verify failed
```

下列會顯示以 source 模式執行本機代理的範例輸出。

```
...
...

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.secure tunneling-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
```

## 啟動 SSH 工作階段

開啟另一個終端機並使用以下命令，透過連線至連接埠 5555 上的本機代理來啟動新的 SSH 工作階段。

```
ssh username@localhost -p 5555
```

系統可能會提示您輸入 SSH 工作階段的密碼。當您完成了 SSH 工作階段時，請輸入 **exit** 以關閉工作階段。

## 清除

### • 關閉通道

我們建議您在使用完通道後關閉通道。如果開放時間超過指定的通道持續時間，通道也可能會關閉。通道一旦關閉就不能重新開放。您仍然可以開啟已關閉的通道，然後選擇 Duplicate tunnel (複製通道) 來複製通道。指定您要使用的通道持續時間，然後建立新通道。

- 若要從 AWS IoT 主控台關閉個別通道或多個通道，請移至 [Tunnels hub](#) (通道中樞)，選擇您要關閉的通道，然後選擇 Close tunnel (關閉通道)。
- 若要使用 AWS IoT API 參考 API 關閉個別通道或多個通道，請使用 [CloseTunnel](#) API 作業。

```
aws iotsecuretunneling close-tunnel \  
  --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

### • 刪除通道

您可以從您的 AWS 帳戶。

#### Warning

刪除動作為永久性動作，且無法還原。

- 若要從 AWS IoT 主控台刪除個別通道或多個通道，請移至 [Tunnels hub](#) (通道中樞)，選擇您要刪除的通道，然後選擇 Delete tunnel (刪除通道)。
- 若要使用 AWS IoT API 參考 API 刪除個別通道或多個通道，請使用 [CloseTunnel](#) API 作業。使用 API 時，請將 delete 旗標設定為 true。

```
aws iotsecuretunneling close-tunnel \  
  --delete true
```

```
--tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"  
--delete true
```

## 開啟遠端裝置的通道並使用瀏覽器型 SSH

從主 AWS IoT 控制台，您可以從「通道」中樞或從您建立之 IoT 物件的詳細資料頁面建立通道。當您從 Tunnel hub (通道中樞) 建立通道，可以指定要使用快速設定或手動設定來建立通道。如需教學課程範例，請參閱[開啟通道並啟動遠端裝置的 SSH 工作階段](#)。

當您從 AWS IoT 主控台的物件詳細資訊頁面建立通道時，您也可以指定是要建立新通道或開啟該物件的現有通道，如本教學課程所示。如果您選擇現有通道，則可以存取您為此裝置建立的最新開放通道。然後，您可以使用終端中的命令行介面 SSH 進入設備。

### 必要條件

- 遠端裝置所在的防火牆必須允許連接埠 443 上的傳出流量。您建立的通道將使用此通訊埠連線至遠端裝置。
- 您已在 AWS IoT 登錄中建立 IoT 物件 (例如 RemoteDevice1)。此物件對應於您遠端裝置在雲端中的表示法。如需詳細資訊，請參閱[在 AWS IoT 登錄檔中註冊裝置](#)。
- 您有一個 IoT 裝置代理程式 (請參閱[IoT Agent Snippet](#)) 在連線至裝置開道的遠端 AWS IoT 裝置上執行，且已設定 MQTT 主題訂閱。如需詳細資訊，請參閱[將裝置連線至 AWS IoT 裝置開道](#)。
- 您必須具有一個在遠端裝置上執行的 SSH 常駐程式。

### 為遠端裝置開啟新通道

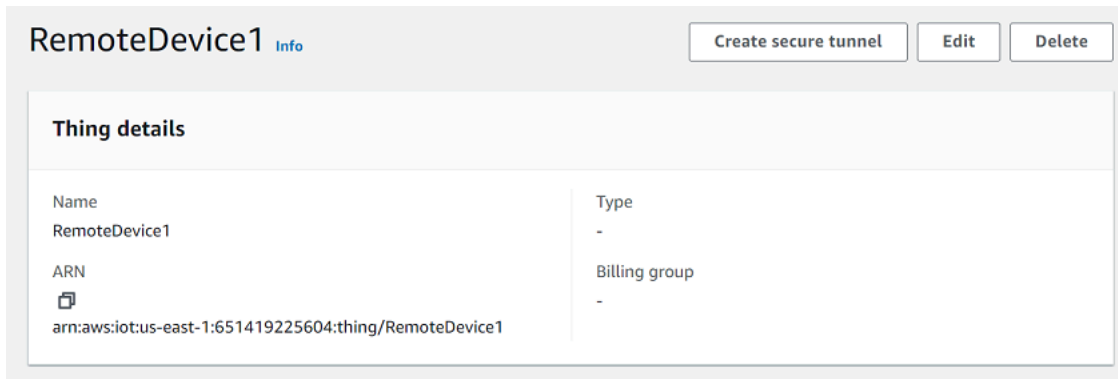
假設您要開啟通道進入遠端裝置：RemoteDevice1。首先，在 AWS IoT 登錄檔中建立名為 RemoteDevice1 的 IoT 物件。然後 AWS Management Console，您可以使用 AWS IoT API 參考 API 或 AWS CLI。

如果您在鍵例通道時設定目的地，安全通道服務會透過 MQTT 和保留的 MQTT 主題 (\$aws/things/RemoteDeviceA/tunnels/notify)，將目的地用戶端存取權杖傳遞給遠端裝置。如需詳細資訊，請參閱[AWS IoT 控制台中的隧道創建方法](#)。

若要從主控台建立遠端裝置的通道

- 選擇物件 RemoteDevice1 以檢視其詳細資訊，然後選擇 Create secure tunnel (建立安全通道)。





2. 選擇要建立新通道或是開啟現有通道。若要建立新通道，請選擇 **Create new tunnel** (建立新通道)。然後，您可以選擇要用快速設定或手動設定方法來建立通道。如需詳細資訊，請參閱 [使用手動設定開啟通道並連線至遠端裝置](#) 及 [開啟通道並使用瀏覽器型 SSH 來存取遠端裝置](#)。

若要使用 API 為遠端裝置建立通道

若要開啟新的通道，您可以使用 [OpenTunnelAPI](#) 作業。以下程式碼顯示執行此命令的範例。

```
aws iotsecuretunneling open-tunnel \
  --region us-east-1 \
  --endpoint https://api.us-east-1.tunneling.iot.amazonaws.com
  --cli-input-json file://input.json
```

下列顯示 `input.json` 檔案的內容。您可以使用 `destinationConfig` 參數來指定目的地裝置的名稱 (例如 `RemoteDevice1`) 以及要用來存取目的地裝置的服務，例如 `SSH`。您亦可選擇指定其他參數，如通道描述和標籤。

`input.json` 的內容

```
{
  "description": "Tunnel to remote device1",
  "destinationConfig": {
    "services": [ "SSH" ],
    "thingName": "RemoteDevice1"
  }
}
```

執行此命令會建立新的通道，並提供您來源和目的地存取權杖。

```
{
  "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>"
}
```

## 開啟現有的通道並使用瀏覽器型 SSH

假設您使用手動設置方法或使用 AWS IoT API 參考 API 為遠端設備創建了隧道。RemoteDevice1 此時您即可開啟裝置的現有通道，並選擇 Quick setup (快速設定) 以使用瀏覽器型 SSH 功能。現有通道的組態無法編輯，因此您無法使用手動設定方法。

若要使用瀏覽器型 SSH 功能，您不必下載來源存取權杖或設定本機代理。Web 型本機代理將為您自動設定，以便您開始與遠端裝置進行互動。

若要使用快速設定方法和瀏覽器型 SSH

1. 前往您所建立物件 RemoteDevice1 的詳細資訊頁面，然後 Create secure tunnel (建立安全通道)。
2. 選擇 Use existing tunnel (使用現有通道)，開啟您為遠端裝置所建立的最新開放通道。無法編輯通道組態，因此您無法對通道使用手動設定方法。若要使用快速設定方法，請選擇 Quick setup (快速設定)。
3. 繼續檢閱，確認通道組態詳細資訊，並建立通道。通道組態不可編輯。

當您建立通道時，安全通道會使用 [Rotate Tunnel Access Token](#) API 作業來撤銷原始存取權杖，並產生新的存取權杖。如果您的遠端裝置使用 MQTT，這些權杖會自動傳遞至其所訂閱 MQTT 主題的遠端裝置。您也可以選擇手動將這些權杖下載到來源裝置。

建立通道之後，您可以使用瀏覽器型 SSH，透過上下文命令列介面，直接從主控台與遠端裝置互動。若要使用此命令行介面，請選擇您所建立物件的通道，然後在詳細資訊頁面中展開 Command-line interface (命令行介面) 區段。由於已為您配置本機代理，因此您可以開始輸入命令以迅速開始存取遠端裝置並與其進行互動 RemoteDevice1。

如需進一步了解快速設定方法和如何使用瀏覽器型 SSH，請參閱 [開啟通道並使用瀏覽器型 SSH 來存取遠端裝置](#)。

## 清除

- 關閉通道

我們建議您在使用完通道後關閉通道。如果開放時間超過指定的通道持續時間，通道也可能會關閉。通道一旦關閉就不能重新開放。您仍然可以開啟已關閉的通道，然後選擇 Duplicate tunnel (複製通道) 來複製通道。指定您要使用的通道持續時間，然後建立新通道。

- 若要從 AWS IoT 主控台關閉個別通道或多個通道，請移至 [Tunnels hub](#) (通道中樞)，選擇您要關閉的通道，然後選擇 Close tunnel (關閉通道)。
- 若要使用 AWS IoT API 參考 API 關閉個別通道或多個通道，請使用 [CloseTunnel](#) API 作業。

```
aws iotsecuretunneling close-tunnel \  
  --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd"
```

## • 刪除通道

您可以從您的 AWS 帳戶。

### Warning

刪除動作作為永久性動作，且無法還原。

- 若要從 AWS IoT 主控台刪除個別通道或多個通道，請移至 [Tunnels hub](#) (通道中樞)，選擇您要刪除的通道，然後選擇 Delete tunnel (刪除通道)。
- 若要使用 AWS IoT API 參考 API 刪除個別通道或多個通道，請使用 [CloseTunnel](#) API 作業。使用 API 時，請將 delete 旗標設定為 true。

```
aws iotsecuretunneling close-tunnel \  
  --tunnel-id "01234567-89ab-0123-4c56-789a01234bcd" \  
  --delete true
```

## 本機代理

本機 Proxy 會透過安全連線使用安全通道，傳輸來源裝置上執行的應用程式所傳送的資料。

WebSocket 您可以從下載本地代理源代理 [GitHub](#)。

本機代理可以在兩種模式下執行：source 或 destination。在來源模式中，本機代理會在與啟動 TCP 連線的用戶端應用程式相同的裝置或網路上執行。在目的地模式中，本機代理會與目的地應用程式一起在遠端裝置上執行。單一通道可以使用通道多工，一次最多支援三個資料串流。對於每個資料串

流，安全通道會使用多個 TCP 連線，進而降低逾時的可能性。如需詳細資訊，請參閱 [多工處理資料串流](#)，並在安全通道中使用同步 TCP 連線。

## 如何使用本機代理

您可以在來源裝置和目的地裝置上執行本機代理，以將資料傳輸至安全通道端點。若您的裝置位於使用 Web 代理的網路中，Web 代理可在將連線轉送至網際網路之前加以攔截。在這種情況下，您需要將本機代理設定為使用 Web 代理。如需詳細資訊，請參閱 [為使用 Web 代理的裝置配置本機代理](#)。

## 本機代理工作流程

下列步驟顯示了本機代理如何在來源和目的地裝置上執行。

### 1. 將本機代理連線至安全通道

本機代理必須先建立安全通道的連線。當您啟動本機代理時，請使用下列引數：

- 指定 AWS 區域 在其中開啟通道的 `-r` 引數。
- `-t` 引數傳遞從 `OpenTunnel` 傳回的來源或目標用戶端存取字符。

#### Note

使用相同用戶端存取字符值的兩個本機代理不能同時連線。

### 2. 執行來源或目的地動作

建立 WebSocket 連線之後，本機 Proxy 會根據其組態執行來源模式或目標模式動作。

根據預設，如果發生任何輸入/輸出 (I/O) 錯誤或連線意外關閉，本機 Proxy 會嘗試重新 WebSocket 連線至安全通道。這會導致 TCP 連線關閉。如果發生任何 TCP 通訊端錯誤，本機代理會透過通道傳送訊息，通知對方關閉其 TCP 連線。根據預設，本機代理一律使用 SSL 通訊。

### 3. 停用本機代理

使用通道之後，可以安全地停止本機代理程序。建議您呼叫 `CloseTunnel` 來明確地關閉通道。活動隧道客戶端可能不會在調用後立即關閉 `CloseTunnel`。

如需如何使用開啟通道和啟 AWS Management Console 動 SSH 工作階段的詳細資訊，請參閱 [開啟通道並啟動遠端裝置的 SSH 工作階段](#)。

## 本機代理最佳實務

執行本機代理時，請遵循下列最佳實務：

- 避免使用 `-t` 本機代理引數傳入存取字符。建議您使用 `AWSIOT_TUNNEL_ACCESS_TOKEN` 環境變數來設定本機代理的存取字符。
- 在作業系統或環境中以最低權限執行本機代理可執行檔。
  - 避免在 Windows 上以系統管理員身分執行本機代理。
  - 避免在 Linux 和 macOS 上以 root 身分執行本機代理。
- 考慮在個別的主機、容器、沙箱，chroot jail 或虛擬化環境上執行本機代理。
- 使用相關的安全旗標建置本地代理，具體取決於您的工具鏈。
- 在具有多個網路界面的裝置上，使用 `-b` 引數將 TCP 通訊端繫結至用來與目標應用程式通訊的網路界面。

## 命令和輸出範例

下列會顯示您執行的命令範例和對應的輸出。此範例顯示如何在 `source` 和 `destination` 模式下設定本機 Proxy。本機 Proxy 會將 HTTPS 通訊協定升級 WebSockets 為建立長期連線，然後開始透過連線傳輸資料至安全通道裝置端點。

在您執行這些命令之前：

您必須先開啟通道並取得來源和目的地的用戶端存取字符。您亦須依照先前所述建置本機代理。要構建本地代理，請在 GitHub 存儲庫中打開[本地代理源代碼](#)，然後按照構建和安裝本地代理的說明進行操作。

### Note

用於範例中的下列命令會使用 `verbosity` 標記，說明執行本機代理後先前所述之不同步驟的概觀。建議您僅將此標記用於測試目的。

## 以來源模式執行本機代理

下列命令顯示如何以來源模式執行本機代理。

### Linux/macOS

在 Linux 或 macOS 中，請在終端機中執行下列命令，以配置並啟動您來源上的本機代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -s 5555 -v 5 -r us-west-2
```

其中：

- `-s` 是來源接聽連接埠，會以來源模式啟動本機代理。
- `-v` 是輸出的詳細程度，可以是 0 到 6 之間的值。
- `-r` 是開啟通道的端點區域。

如需有關這些參數的詳細資訊，請參閱[使用命令列引數設定的選項](#)。

## Windows

在 Windows 中，您可將本機代理配置為類似於您處理 Linux 或 macOS 的方式，但定義環境變數的方式則與其他平台不同。在 cmd 視窗中執行下列命令，以配置並啟動您來源上的本機代理。

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -s 5555 -v 5 -r us-west-2
```

其中：

- `-s` 是來源接聽連接埠，會以來源模式啟動本機代理。
- `-v` 是輸出的詳細程度，可以是 0 到 6 之間的值。
- `-r` 是開啟通道的端點區域。

如需有關這些參數的詳細資訊，請參閱[使用命令列引數設定的選項](#)。

下列會顯示以 `source` 模式執行本機代理的範例輸出。

```
...
...
```

### 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
```

## 以目的地模式執行本機代理

下列命令顯示如何以目的地模式執行本機代理。

### Linux/macOS

在 Linux 或 macOS 中，請在終端機中執行下列命令，以配置並啟動您目的地上的本機代理。

```
export AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
./localproxy -d 22 -v 5 -r us-west-2
```

其中：

- `-d` 是以目的地模式啟動本機代理的目的地應用程式。
- `-v` 是輸出的詳細程度，可以是 0 到 6 之間的值。
- `-r` 是開啟通道的端點區域。

如需有關這些參數的詳細資訊，請參閱[使用命令列引數設定的選項](#)。

## Windows

在 Windows 中，您可將本機代理配置為類似於您處理 Linux 或 macOS 的方式，但定義環境變數的方式則與其他平台不同。在 cmd 視窗中執行下列命令，以配置並啟動您目的地上的本機代理。

```
set AWSIOT_TUNNEL_ACCESS_TOKEN=${access_token}
.\localproxy -d 22 -v 5 -r us-west-2
```

其中：

- -d 是以目的地模式啟動本機代理的目的地應用程式。
- -v 是輸出的詳細程度，可以是 0 到 6 之間的值。
- -r 是開啟通道的端點區域。

如需有關這些參數的詳細資訊，請參閱[使用命令列引數設定的選項](#)。

下列會顯示以 destination 模式執行本機代理的範例輸出。

```
...
...
```

### 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...
```

## 為使用 Web 代理的裝置配置本機代理

您可以在 AWS IoT 裝置上使用本機 Proxy 與 AWS IoT 安全通道 API 進行通訊。本機 Proxy 會透過安全連線，使用安全通道傳輸裝置應用程式傳送的資料。WebSocket 本機代理可在 source 或 destination 模式中運作。在 source 模式中，其會在啟動 TCP 連線的相同裝置或網路上執行。在 destination 模式中，本機代理會與目的地應用程式一起在遠端裝置上執行。如需詳細資訊，請參閱 [本機代理](#)。

本機 Proxy 必須直接連線至網際網路，才能使用 AWS IoT 安全通道。對於具有安全通道的長期 TCP 連線，本機 Proxy 會升級 HTTPS 要求，以建立與其中一個 [安全通道](#) 裝置 WebSockets 連線端點的連線。

若您的裝置位於使用 Web 代理的網路中，Web 代理可在將連線轉送至網際網路之前加以攔截。如要建立與安全通道裝置連線端點的長期連線，請配置本機代理使用 Web 代理，如 [WebSocket 規格](#) 中所述。

### Note

[AWS IoT 裝置用戶端](#) 不支援使用 Web 代理的裝置。如要使用 Web 代理，您需要使用本機代理，並配置其與 Web 代理搭配使用，如下所述。

下列步驟顯示了本機代理如何與 Web 代理搭配使用。

1. 本機代理傳送 HTTP CONNECT 請求至 Web 代理，其包含安全通道服務的遠端地址及 Web 代理身分驗證資訊。
2. Web 代理隨後會建立遠端安全通道端點的長期連線。
3. TCP 連線已建立，本機代理現在可以來源和目的地模式運作，進行資料傳輸。

如要完成此程序，請執行下列步驟：

- [建置本機代理](#)
- [配置您的 Web 代理](#)
- [配置並啟動本機代理](#)

## 建置本機代理

打開 GitHub 存儲庫中的[本地代理源代碼](#)，然後按照構建和安裝本地代理的說明進行操作。

## 配置您的 Web 代理

本機代理依賴 [HTTP/1.1 規格](#) 所說明的 HTTP 通道機制。如要符合規格，您的 Web 代理必須允許裝置可以使用 CONNECT 方法。

您配置 Web 代理的方式依您使用的 Web 代理及 Web 代理版本而定。如要確保您正確地配置 Web 代理，請查看您的 Web 代理文件。

如要配置您的 Web 代理，請先識別您的 Web 代理 URL，並確認您的 Web 代理是否支援 HTTP 通道。當您配置並啟動本機代理時，Web 代理 URL 將於稍後使用。

### 1. 識別您的 Web 代理 URL

您的 Web 代理 URL 將會採用下列格式。

```
protocol://web_proxy_host_domain:web_proxy_port
```

AWS IoT 安全通道僅支援 Web Proxy 的基本驗證。如要使用基本身分驗證，您必須指定 **username** 和 **password** 以做為 Web 代理 URL 的一部分。Web 代理 URL 將會採用下列格式。

```
protocol://username:password@web_proxy_host_domain:web_proxy_port
```

- **####** 可為 http 或 https。建議您使用 https。
- **web\_proxy\_host\_domain** 是您 Web 代理的 IP 地址，或可解析 Web 代理 IP 地址的 DNS 名稱。
- **web\_proxy\_port** 是 Web 代理接聽的連接埠。
- web 代理會使用此 **username** 和 **password** 來驗證請求。

### 2. 測試您的 Web 代理 URL

如要確認您的 Web 代理是否支援 TCP 通道，請使用 curl 命令並確保您取得 2xx 或 3xx 回應。

例如，若您的 Web 代理 URL 是 `https://server.com:1235`，請使用具 `curl` 命令的 `proxy-insecure` 旗標，因為 Web 代理可能會依賴自簽憑證。

```
export HTTPS_PROXY=https://server.com:1235
curl -I https://aws.amazon.com --proxy-insecure
```

若您的 Web 代理 URL 具有 `http` 連接埠 (例如，`http://server.com:1234`)，則您無須使用 `proxy-insecure` 旗標。

```
export HTTPS_PROXY=http://server.com:1234
curl -I https://aws.amazon.com
```

## 配置並啟動本機代理

如要將本機代理配置為使用 Web 代理，您必須以 Web 代理使用的 DNS 網域名稱或 IP 地址和連接埠號來配置 `HTTPS_PROXY` 環境變數。

在您配置本機代理之後，您可以使用本機代理，如 [讀我檔案](#) 文件中所述。

### Note

您的環境變數宣告區分大小寫。我們建議您使用全部大寫或全部小寫字母來定義每個變數一次。下列範例顯示環境變數以全部大寫字母進行宣告。若同時使用大寫和小寫字母指定相同的變數，則使用小寫字母指定的變數優先。

下列命令顯示如何將執行於目的地上的本機代理配置為使用 Web 代理並啟動本機代理。

- `AWSIoT_TUNNEL_ACCESS_TOKEN`：此變數會保存目的地的用戶端存取字符 (CAT)。
- `HTTPS_PROXY`：此變數會保留 Web 代理 URL 或 IP 地址，以配置本機代理。

下列範例中顯示的命令取決於您使用的作業系統，及 Web 代理是否在 HTTP 或 HTTPS 連接埠上進行接聽。

### 在 HTTP 連接埠上接聽 Web 代理

若您的 Web 代理正在接聽 HTTP 連接埠，您可提供 `HTTPS_PROXY` 變數的 Web 代理 URL 或 IP 地址。

## Linux/macOS

在 Linux 或 macOS 中，請在終端機中執行下列命令，以配置並啟動您目的地上的本機代理，以使用接聽 HTTP 連接埠的 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
```

若您必須使用代理進行驗證，則必須指定 **username** 和 **password** 以作為 HTTPS\_PROXY 變數的一部分。

```
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，您可將本機代理配置為類似於您處理 Linux 或 macOS 的方式，但定義環境變數的方式則與其他平台不同。在 cmd 視窗中執行下列命令，以配置並啟動您目的地上的本機代理，以使用接聽 HTTP 連接埠的 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
```

若您必須使用代理進行驗證，則必須指定 **username** 和 **password** 以作為 HTTPS\_PROXY 變數的一部分。

```
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
```

## 在 HTTPS 連接埠上接聽 Web 代理

如果您的 Web 代理正在接聽 HTTPS 連接埠，請執行下列命令。

**Note**

如果您使用的是自簽名證書的 Web 代理，或者如果您在沒有原生 OpenSSL 支持和默認配置的操作系統上運行本地代理，則必須按照存儲庫中的[證書設置部分中的說明設置 Web 代理證書](#)。GitHub

下列命令與您為 HTTP 代理配置 Web 代理的方式類似，不同之處在於您還需要指定之前所述所安裝憑證檔案的路徑。

**Linux/macOS**

在 Linux 或 macOS 中，請在終端機中執行下列命令，以配置執行於您目的地上的本機代理，以使用接聽 HTTPS 連接埠的 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
```

若您必須使用代理進行驗證，則必須指定 **username** 和 **password** 以作為 HTTPS\_PROXY 變數的一部分。

```
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，執行 cmd 視窗中的下列命令，可配置並啟動執行於目的地上的本機代理，以使用接聽 HTTP 連接埠的 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
```

若您必須使用代理進行驗證，則必須指定 **username** 和 **password** 以作為 HTTPS\_PROXY 變數的一部分。

```
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
```

## 命令和輸出範例

下列會顯示您執行於 Linux 作業系統上的命令範例，及相對應的輸出。該範例顯示接聽 HTTP 連接埠上的 Web 代理，及如何配置本機代理以 `source` 和 `destination` 兩種模式來使用 Web 代理。在執行這些命令之前，您必須先開啟通道並取得來源和目的地的用戶端存取字符。您還必須依照先前所述建置本機代理並配置 Web 代理。

此處概述您啟動本機代理後的步驟。本機代理：

- 識別 Web 代理 URL，以便其可使用 URL 連線至代理伺服器。
- 建立與 Web 代理的 TCP 連線。
- 將 HTTP CONNECT 請求傳送至 Web 代理，並等待 HTTP/1.1 200 回應，表示已建立連線。
- 將 HTTPS 通訊協定升級 WebSockets 為建立長期連線。
- 透過與安全通道裝置端點的連接，開始傳輸資料。

### Note

用於範例中的下列命令會使用 `verbosity` 標記，說明執行本機代理後先前所述之不同步驟的概觀。建議您僅將此標記用於測試目的。

## 以來源模式執行本機代理

下列命令顯示如何以來源模式執行本機代理。

```
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
```

下列會顯示以 `source` 模式執行本機代理的範例輸出。

...

```

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.secure tunneling-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
```

## 以目的地模式執行本機代理

下列命令顯示如何以目的地模式執行本機代理。

```

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

```

下列會顯示以 destination 模式執行本機代理的範例輸出。

```
...
```

```
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: 06717bffffed3fd05-00001355-000b8315-da3109a85da804dd-24c3d10d
```

```
Web socket subprotocol selected: aws.iot.secure tunneling-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...
```

## 多工處理資料串流，並在安全通道中使用同步 TCP 連線

您可使用安全通道多工功能，在每個通道中使用多個資料串流。透過多工處理，您可以使用多個資料串流對裝置進行疑難排解。您還可透過消除建置、部署及啟動多個本機裝置或為相同裝置開啟多個通道的需求來減少作業負載。例如，對於需要傳送多個 HTTP 和 SSH 資料串流的 Web 瀏覽器，可使用多工處理。

對於每個資料串流，AWS IoT 安全通道支援同時 TCP 連線。使用同時連線可減少用戶端發出多個請求時逾時的可能性。例如，它可以減少遠端存取目標裝置本機的 Web 伺服器時的載入時間。

以下各節將詳細說明多工處理和使用同時 TCP 連線，以及它們的不同使用案例。

### 主題

- [多工處理安全通道中的多個資料串流](#)



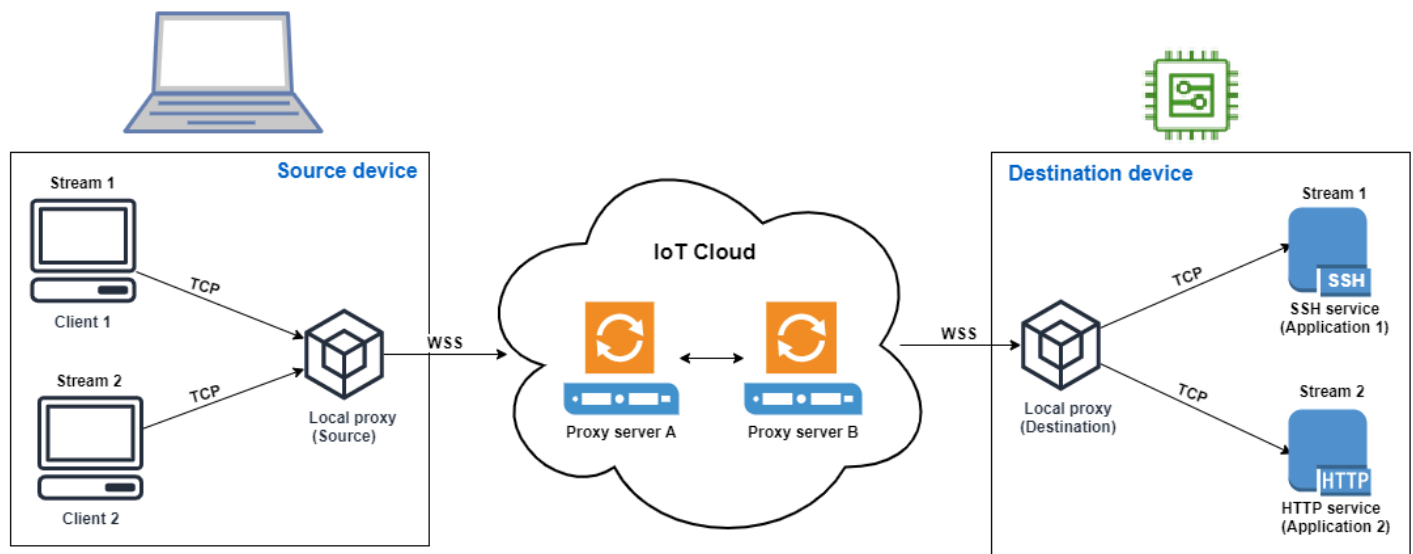
- [在安全通道中使用 TCP 連線](#)

## 多工處理安全通道中的多個資料串流

對於使用多個連接或連接埠的裝置，您可以使用多工功能。當您需要與遠端裝置進行多個連線以疑難排解任何問題時，也可以使用多工處理。例如，可使用於需要傳送多個 HTTP 和 SSH 資料串流的 Web 瀏覽器。來自兩個串流的應用程式資料會透過多工通道同時傳輸。

### 範例使用案例

例如，您可能需要連接至裝置上的 Web 應用程式以更改某些連網參數，同時透過終端機發出 shell 命令，驗證裝置是否在使用新的連網參數時仍然正常運作。在這種情況下，您可能需要透過 HTTP 和 SSH 連接到設備，並傳輸兩個 parallel 數據流以同時訪問 Web 應用程式和終端機。利用多工功能，這兩個獨立串流可在同一時間透過同一個通道進行傳輸。



### 如何設定多工通道

下列程序將逐步引導您如何使用需要連線至多個連接埠的應用程式，來設定多工通道，以便對裝置進行疑難排解。您會設定一個具有兩個多工串流的通道：一個 HTTP 串流和一個 SSH 串流。

#### 1. (選擇性) 建立組態檔

您可以選擇性地使用組態檔來設定來源和目的地裝置。如果您的連接埠映射可能經常變更，請使用組態檔。如果您喜歡使用 CLI 指定連接埠映射，或無需在指定的接聽連接埠上啟動本機 Proxy，則可略過此步驟。如需有關如何使用組態檔的詳細資訊，請參閱中的[透過--config 設定的選項](#)。

GitHub

1. 在來源裝置上，在要執行本機 Proxy 的資料夾中，建立名為 Config 的組態資料夾。在此資料夾中建立一個名為 SSHSource.ini 的檔案，其內容如下：

```
HTTP1 = 5555
SSH1 = 3333
```

2. 在目的地裝置上，在要執行本機 Proxy 的資料夾中，建立名為 Config 的組態資料夾。在此資料夾中建立一個名為 SSHDestination.ini 的檔案，其內容如下：

```
HTTP1 = 80
SSH1 = 22
```

## 2. 開啟通道

使用 OpenTunnel API 操作或 open-tunnel CLI 命令開啟通道。透過將 SSH1 和指定 HTTP1 為與遠端裝置對應的物件的服務以及 AWS IoT 物件名稱來設定目的地。您的 SSH 和 HTTP 應用程式正在此遠端裝置上執行。您必須已在 AWS IoT 登錄中建立 IoT 物件。如需詳細資訊，請參閱 [如何使用登錄檔管理物件](#)。

```
aws iotsecuretunneling open-tunnel \  
--destination-config thingName=RemoteDevice1,services=HTTP1,SSH1
```

執行此命令會產生您將用來執行本機 Proxy 的來源和目的地存取字符。

```
{  
  "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. 配置並啟動本機代理

在執行本機 Proxy 之前，請先設定 AWS IoT 裝置用戶端，或從下載本機 Proxy 原始程式碼，[GitHub](#) 然後針對您選擇的平台進行建置。然後，您可以啟動目的地和來源本機 Proxy 以連接到安全通道。如需設定和使用本機 Proxy 的詳細資訊，請參閱 [如何使用本機代理](#)。

**Note**

在來源裝置上，如果您不使用任何組態檔或使用 CLI 指定連接埠映射，您仍然可以使用相同的命令來執行本機 Proxy。本機 Proxy 將會自動選取可用的連接埠供您使用並管理映射。

## Start local proxy using configuration files

執行下列命令，以使用組態檔在來源和目的地模式下執行本機 Proxy。

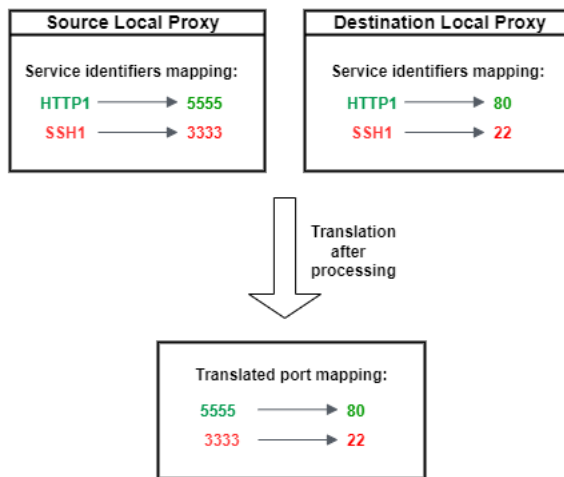
```
// ----- Start the destination local proxy -----  
./localproxy -r us-east-1 -m dst -t destination_client_access_token  
  
// ----- Start the source local proxy -----  
// You also run the same command below if you want the local proxy to  
// choose the mappings for you instead of using configuration files.  
./localproxy -r us-east-1 -m src -t source_client_access_token
```

## Start local proxy using CLI port mapping

使用 CLI 明確指定連接埠映射，以執行下列命令，在來源和目的地模式中執行本機 Proxy。

```
// ----- 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
```

來自 SSH 和 HTTP 連線的應用程式資料現在可以透過多工通道同時進行傳輸。從下方對應圖中可看出，服務識別符的作用是一種可讀取格式，用來轉換來源和目的地裝置之間的連接埠映射。透過此組態，安全通道會將來源裝置上連接埠 *5555* 的任何傳入 HTTP 流量轉送至目的地裝置上的連接埠 *80*，以及從連接埠 *3333* 傳入的 SSH 流量轉送至目的地裝置上的連接埠 *22*。



## 在安全通道中使用 TCP 連線

AWS IoT 安全通道為每個資料串流同時支援一個以上的 TCP 連線。當您需要同時連線至遠端裝置時，可以使用此功能。如果有多個請求來自用戶端，使用同時 TCP 連線可以降低逾時的可能性。例如，當存取具有執行多個元件的 Web 伺服器時，同時 TCP 連線可減少載入網站所需的時間。

### Note

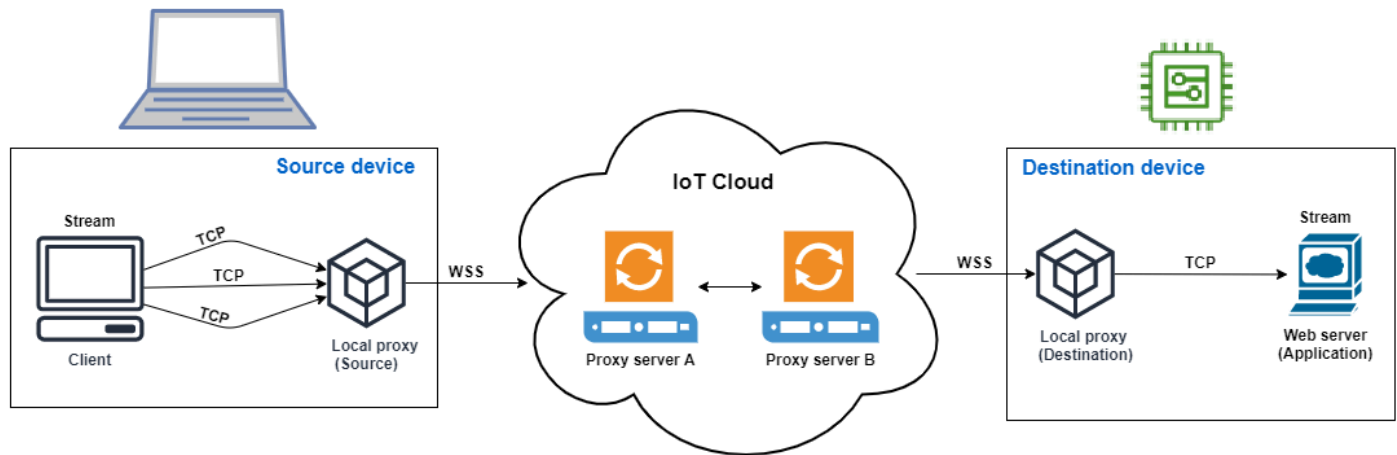
同時 TCP 連線的頻寬限制為每秒 800 KB。AWS 帳戶 AWS IoT 安全通道可以根據傳入的請求數量為您配置此限制。

## 範例使用案例

假設您需要遠端存取位於目的地裝置本機上，且其上有多個元件運行的 Web 服務器。若使用單一 TCP 連線，嘗試存取 Web 伺服器時，循序載入可能會增加載入網站上資源所需的時間。同時 TCP 連線可透過滿足網站的資源需求來減少加載時間，從而縮短存取時間。下圖顯示對於通往遠端裝置上執行之 Web 伺服器應用程式的資料串流如何支援同時 TCP 連線。

### Note

如果您想要使用通道存取遠端裝置上執行的多個應用程式，您可以使用通道多工處理。如需詳細資訊，請參閱 [多工處理安全通道中的多個資料串流](#)。



## 如何使用同時 TCP 連線

下列程序將逐步引導您如何使用同時 TCP 連線來存取遠端裝置上的 Web 瀏覽器。當用戶端有多個要求時，AWS IoT 安全通道會自動設定同時 TCP 連線來處理要求，進而減少載入時間。

### 1. 開啟通道

使用 `OpenTunnel` API 操作或 `open-tunnel` CLI 命令開啟通道。將 HTTP 指定為與遠端裝置對應的服務和 AWS IoT 物件名稱，以設定目的地。您的 Web 伺服器應用程式正在此遠端裝置上執行。您必須已在 AWS IoT 登錄中建立 IoT 物件。如需詳細資訊，請參閱 [如何使用登錄檔管理物件](#)。

```
aws iotsecuretunneling open-tunnel \
  --destination-config thingName=RemoteDevice1,services=HTTP
```

執行此命令會產生您將用來執行本機 Proxy 的來源和目的地存取字符。

```
{
  "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. 配置並啟動本機代理

在執行本機 Proxy 之前，請先從下載本機 Proxy 原始程式碼，[GitHub](#)並針對您選擇的平台建置。然後，您可以啟動目的地和來源本機 Proxy，以連線到安全通道，並開始使用遠端 Web 伺服器應用程式。

### Note

若要使用同時 TCP 連線的 AWS IoT 安全通道，您必須升級至最新版本的本機 Proxy。如果您使用 AWS IoT 裝置用戶端設定本機 Proxy，則無法使用此功能。

```
// 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
```

如需設定和使用本機 Proxy 的詳細資訊，請參閱 [如何使用本機代理](#)。

您現在可以使用通道來存取 Web 伺服器應用程式。AWS IoT 當客戶端有多個請求時，安全通道將自動設置並處理同時 TCP 連接。

## 設定遠端裝置並使用 IoT 代理程式

IoT 代理程式用來接收包含用戶端存取字符的 MQTT 訊息，並在遠端裝置上啟動本機代理。如果您想要安全通道使用 MQTT 交付用戶端存取字符，則必須在遠端裝置上安裝並執行 IoT 代理程式。IoT 代理程式必須訂閱下列保留的 IoT MQTT 主題：

### Note

如果您想要透過訂閱保留的 MQTT 主題以外的方法將目的地用戶端存取字符傳遞給遠端裝置，則可能需要目的地用戶端存取字符 (CAT) 接聽程式和本機 Proxy。CAT 接聽程式必須使用您選擇的用戶端存取字符傳遞機制，並且能夠以目的地模式啟動本機 Proxy。

## IoT Agent Snippet

IoT 代理程式必須訂閱下列保留的 IoT MQTT 主題，才能接收 MQTT 訊息並啟動本機 Proxy：

```
$aws/things/thing-name/tunnels/notify
```

哪裡 *thing-name* 是與遠端裝置相 AWS IoT 關聯的物件名稱。

以下是 MQTT 訊息酬載的範例：

```
{
  "clientAccessToken": "destination-client-access-token",
  "clientMode": "destination",
  "region": "aws-region",
  "services": ["destination-service"]
}
```

在收到 MQTT 訊息之後，IoT 代理程式必須使用適當的參數，在遠端裝置上啟動本機代理。

下列 Java 程式碼示範如何使用 [AWS IoT 裝置 SDK](#) 和 Java 程式庫 [ProcessBuilder](#) 來建置簡單的 IoT 代理程式以使用安全通道。

```
// Find the IoT device endpoint for your AWS ##
final String endpoint = iotClient.describeEndpoint(new
    DescribeEndpointRequest().withEndpointType("iot:Data-ATS")).getEndpointAddress();

// Instantiate the IoT Agent with your AWS credentials
final String thingName = "RemoteDeviceA";
final String tunnelNotificationTopic = String.format("$aws/things/%s/tunnels/notify",
    thingName);
final AWSTunnelMqttClient mqttClient = new AWSTunnelMqttClient(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 Kibana 的存取

安全通道提供服務特有的動作、資源和條件內容金鑰，可用於 IAM 許可政策。

## 通道存取先決條件

- 了解如何使用 [IAM 政策](#) 保護 AWS 資源。
- 了解如何建立和評估 [IAM 條件](#)。
- 了解如何使用 AWS 資源 [標籤保護資源](#)。

## 通道存取政策

您必須使用以下政策授予許可，才能使用安全通道 API。如需 AWS IoT 安全性的詳細資訊，請參閱 [的身分識別與存取管理 AWS IoT](#)。

物聯網:OpenTunnel

iot:OpenTunnel 原則動作會授予主體呼叫權限 [OpenTunnel](#)。

在 IAM 政策陳述式中的 Resource 元素：

- 指定萬用字元通道 ARN：

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

- 指定一個物件 ARN 來管理特定 IoT 物件的 OpenTunnel 許可：

```
arn:aws:iot:aws-region:aws-account-id:thing/thing-name
```

例如，以下政策陳述式允許您開啟一個通道，前往名為 TestDevice 的 IoT 物件。

```
{
  "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 政策動作支援下列條件金鑰：

- iot:ThingGroupArn
- iot:TunnelDestinationService
- aws:RequestTag/*tag-key*
- aws:SecureTransport
- aws:TagKeys

下列政策陳述式可讓您對事物開啟通道，前提是該事物屬於名稱開頭為 TestGroup 的事物群組，並且通道上配置的目的地服務為 SSH。

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

您也可以使用資源標籤來控制開啟通道的許可。例如，如果標籤索引鍵 Owner 的值為 Admin 且未指定其他標籤，則下列政策陳述式允許開啟通道。如需使用標籤的詳細資訊，請參閱 [標記您的 AWS IoT 資源](#)。

```
{
  "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"
      }
    }
  }
}

```

## 物聯網:RotateTunnelAccessToken

iot:RotateTunnelAccessToken 原則動作會授予主體呼叫權限 [RotateTunnelAccessToken](#)。

在 IAM 政策陳述式中的 Resource 元素：

- 指定完全的合格通道 ARN：

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

您也可以使用萬用字元通道 ARN：

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

- 指定一個物件 ARN 來管理特定 IoT 物件的 RotateTunnelAccessToken 許可：

```
arn:aws:iot:aws-region:aws-account-id:thing/thing-name
```

例如，以下政策陳述式允許您輪換通道的來源存取字符，或用於名為 TestDevice 的 IoT 物件之用戶端目的地存取字符。

```

{
  "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 政策動作支援下列條件金鑰：

- `iot:ThingGroupArn`
- `iot:TunnelDestinationService`
- `iot:ClientMode`
- `aws:SecureTransport`

下列政策陳述式可讓您將目的地存取字符輪換至物件，前提是該事物屬於名稱開頭為 `TestGroup` 的物件群組，並且通道上配置的目的地服務為 SSH，用戶端狀態為 `DESTINATION`。

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

## 物聯網:DescribeTunnel

`iot:DescribeTunnel` 原則動作會授予主體呼叫權限 [DescribeTunnel](#)。

在 IAM 政策陳述式的 `Resource` 元素中指定完全合格的通道 ARN：

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

您也可以使用萬用字元 ARN：

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

`iot:DescribeTunnel` 政策動作支援下列條件金鑰：

- `aws:ResourceTag/tag-key`
- `aws:SecureTransport`

如果使用值為 Admin 的金鑰 Owner 標記所請求的通道，下列政策陳述式可讓您呼叫 DescribeTunnel。

```
{
  "Effect": "Allow",
  "Action": "iot:DescribeTunnel",
  "Resource": [
    "arn:aws:iot:aws-region:aws-account-id:tunnel/*"
  ],
  "Condition": {
    "StringEquals": {
      "aws:ResourceTag/Owner": "Admin"
    }
  }
}
```

### 物聯網:ListTunnels

`iot:ListTunnels` 原則動作會授予主體呼叫權限 [ListTunnels](#)。

在 IAM 政策陳述式中的 Resource 元素：

- 指定萬用字元通道 ARN：

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

- 指定一個物件 ARN 來管理所選 IoT 物件的 ListTunnels 許可：

```
arn:aws:iot:aws-region:aws-account-id:thing/thing-name
```

`iot:ListTunnels` 政策動作支援條件金鑰 `aws:SecureTransport`。

以下政策陳述式允許您列出名為 TestDevice 之物件的通道。

```
{
  "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"
  ]
}

```

### 物聯網:ListTagsForResource

iot:ListTagsForResource 政策動作可授與委託人呼叫 ListTagsForResource 的許可。

在 IAM 政策陳述式的 Resource 元素中指定完全合格的通道 ARN：

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

您也可以使用萬用字元通道 ARN：

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

iot:ListTagsForResource 政策動作支援條件金鑰 aws:SecureTransport。

### 物聯網:CloseTunnel

iot:CloseTunnel 原則動作會授予主體呼叫權限 [CloseTunnel](#)。

在 IAM 政策陳述式的 Resource 元素中指定完全合格的通道 ARN：

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

您也可以使用萬用字元通道 ARN：

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

iot:CloseTunnel 政策動作支援下列條件金鑰：

- iot:Delete
- aws:ResourceTag/*tag-key*
- aws:SecureTransport

如果請求的 Delete 參數為 false，並使用值為 QATeam 的金鑰 Owner 標記所請求的通道，下列政策陳述式可讓您呼叫 CloseTunnel。

```

{
  "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"
    }
  }
}
```

### 物聯網:TagResource

iot:TagResource 政策動作可授與委託人呼叫 TagResource 的許可。

在 IAM 政策陳述式的 Resource 元素中指定完全合格的通道 ARN：

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

您也可以使用萬用字元通道 ARN：

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

iot:TagResource 政策動作支援條件金鑰 aws:SecureTransport。

### 物聯網:UntagResource

iot:UntagResource 政策動作可授與委託人呼叫 UntagResource 的許可。

在 IAM 政策陳述式的 Resource 元素中指定完全合格的通道 ARN：

```
arn:aws:iot:aws-region: aws-account-id:tunnel/tunnel-id
```

您也可以使用萬用字元通道 ARN：

```
arn:aws:iot:aws-region:aws-account-id:tunnel/*
```

iot:UntagResource 政策動作支援條件金鑰 aws:SecureTransport。

## 透過旋轉用戶端存取權杖來解決 AWS IoT 安全通道連線問題

使用 AWS IoT 安全通道時，即使通道處於開啟狀態，也可能會遇到連線問題。以下部分介紹了一些可能的問題，以及如何透過輪換用戶端存取字符解決這些問題。若要輪替用戶端存取權杖 (CAT)，請使用

[RotateTunnelAccessToken](#) API 或 [rotate-tunnel-access-token](#) AWS CLI。根據在來源模式或目的地模式下使用用戶端時遇到錯誤，可以在來源模式或目的地模式下輪換 CAT，也可以同時輪換。

#### Note

- 如果您不確定是否需要在來源或目的地輪換 CAT，可以在使用 `RotateTunnelAccessToken` API 時將 `ClientMode` 設為 `ALL`，同時輪換來源和目的地的 CAT。
- 輪換 CAT 不會延長通道持續時間。例如，假設通道持續時間為 12 小時，且通道已開放 4 小時。當您輪換存取字符，產生的新字符只能在剩下 8 小時內使用。

#### 主題

- [無效的用戶端存取字符錯誤](#)
- [用戶端字符不相符錯誤](#)
- [遠端裝置連線問題](#)

## 無效的用戶端存取字符錯誤

使用 AWS IoT 安全通道時，使用相同的用戶端存取權杖 (CAT) 重新連線至相同的通道時，可能會遇到連線錯誤。在這種情況下，本機代理無法連線到安全通道代理伺服器。如果您使用來源模式的用戶端，可能會看到以下錯誤訊息：

```
Invalid access token: The access token was previously used and cannot be used again
```

出現這個錯誤的原因是用戶端存取字符 (CAT) 只能由本機代理使用一次，之後就無效。如果要解決這個錯誤，請使用 `SOURCE` 模式輪換用戶端存取字符，為來源產生新的 CAT。如需顯示如何輪換來源 CAT 的範例，請參閱 [輪換來源 CAT 範例](#)。

## 用戶端字符不相符錯誤

#### Note

不推薦以用戶端字符重複使用 CAT。建議您使用 `RotateTunnelAccessToken` API 來輪換用戶端存取字符以重新連線到通道。



如果您使用的是用戶端字符，可以重複使用 CAT 重新連線到通道。要重複使用 CAT，必須在首次連線到安全通道時為用戶端字符提供 CAT。安全通道存放用戶端字符，因此對於使用相同字符的後續連線嘗試，也必須提供用戶端字符。如需有關使用用戶端權杖的詳細資訊，請參閱中的[本機 Proxy 參考實作 GitHub](#)。

使用用戶端字符時，如果您使用來源模式的用戶端，可能會看到以下錯誤：

```
Invalid client token: The provided client token does not match the client token
that was previously set.
```

出現這個錯誤的原因，是提供的用戶端字符與存取通道時隨 CAT 提供的用戶端字符不相符。若要解決這個錯誤，請使用 SOURCE 模式輪換 CAT，為來源產生新的 CAT。如下列範例所示：

### 輪換來源 CAT 範例

以下說明如何以 SOURCE 模式執行 RotateTunnelAccessToken API，為來源產生一個新的 CAT：

```
aws iotsecuretunneling rotate-tunnel-access-token \
  --region <region> \
  --tunnel-id <tunnel-id> \
  --client-mode SOURCE
```

執行這個命令會產生新的來源存取字符，並回傳通道的 ARN。

```
{
  "sourceAccessToken": "<source-access-token>",
  "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/<tunnel-id>"
}
```

您現在可以使用新的來源字符以來源模式與本機代理連線。

```
export AWSIoT_TUNNEL_ACCESS_TOKEN=<source-access-token>
./localproxy -r <region> -s <port>
```

下列會顯示執行本機代理的範例輸出：

```
...
[info] Starting proxy in source mode
...
```

```
[info] Successfully established websocket connection with proxy server ...
[info] Listening for new connection on port <port>
...
```

## 遠端裝置連線問題

使用 AWS IoT 安全通道時，即使通道已開啟，裝置也可能會意外中斷連線。若要識別裝置是否仍連線至通道，您可以使用 [DescribeTunnel](#) API 或 [描述](#) AWS CLI 通道。

裝置可能會因多種原因而中斷連線。要解決連線問題，如果裝置由於以下可能原因而中斷連線，可以在輪換目的地的 CAT：

- 目的地的 CAT 變為無效。
- 字符未透過安全通道保留的 MQTT 主題傳遞到裝置：

```
$aws/things/<thing-name>/tunnels/notify
```

以下範例說明如何解決這個問題：

### 輪換目標 CAT 範例

考慮使用遠端裝置 *<RemoteThing1>*。要開放該物件的通道，可以使用以下命令：

```
aws iotsecuretunneling open-tunnel \
  --region <region> \
  --destination-config thingName=<RemoteThing1>,services=SSH
```

執行這個命令會產生來源和目的地的通道詳細資訊和 CAT。

```
{
  "sourceAccessToken": "<source-access-token>",
  "destinationAccessToken": "<destination-access-token>",
  "tunnelId": "<tunnel-id>",
  "tunnelArn": "arn:aws:iot:<region>:<account-id>:tunnel/<tunnel-id>"
}
```

但是，當您使用 [DescribeTunnel](#) API 時，輸出會指示裝置已中斷連線，如下所示：

```
aws iotsecuretunneling describe-tunnel \
  --tunnel-id <tunnel-id> \
```

```
--region <region>
```

執行這個命令會顯示裝置仍未連線。

```
{
  "tunnel": {
    ...
    "destinationConnectionState": {
      "status": "DISCONNECTED"
    },
    ...
  }
}
```

若要解決這個錯誤，請以 DESTINATION 模式的用戶端和目的地的組態執行 RotateTunnelAccessToken API。執行這個命令會撤銷舊的存取字符，產生新字符，並將這個字符重新傳送到 MQTT 主題：

```
$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>
```

執行這個命令會產生新的存取字符，如下所示。如果裝置代理程式設定正確，會將字符傳遞到裝置以連接通道。

```
{
  "destinationAccessToken": "destination-access-token",
  "tunnelArn": "arn:aws:iot:region:account-id:tunnel/tunnel-id"
}
```

## 裝置佈建

AWS 提供數種不同的方式來佈建裝置，並在其上安裝唯一的用戶端憑證。本節描述每一種方式，以及如何選取最適合您 IoT 解決方案的方式。這些選項會在標題名稱為 [Device Manufacturing and Provisioning with X.509 Certificates in AWS IoT Core](#) 的白皮書中詳細描述。

### 選取最適合您情況的選項

- 您可以在 IoT 裝置上安裝憑證，然後再交付這些憑證

如果您可以在 IoT 裝置上安全地安裝唯一的用戶端憑證，然後再交付這些憑證，以供最終使用者使用，則您想要使用 [即時佈建 \(JITP\)](#) 或 [即時註冊 \(JITR\)](#)。

使用 JITP 和 JITR 時，用於簽署裝置憑證的憑證授權單位 (CA) 會在裝置第一次連線時進行註冊，AWS IoT 並可在裝置第一次連線 AWS IoT 時辨識。該設備在 AWS IoT 其第一次連接時使用其佈建模板的詳細信息進行佈建。

如需有關單一物件、JITP、JITR，以及大量佈建具有唯一憑證之裝置的詳細資訊，請參閱 [the section called “佈建具有裝置憑證的裝置”](#)。

- 最終使用者或安裝程式可以使用應用程式，在其 IoT 裝置上安裝憑證

如果您無法在 IoT 裝置上安全地安裝唯一的用戶端憑證，然後再將這些憑證交付給最終使用者，但最終使用者或安裝程式可以使用應用程式，來註冊裝置並安裝唯一的裝置憑證，則您想要使用 [透過信任的使用者佈建](#) 程序。

使用信任的使用者 (例如具有已知帳戶的最終使用者或安裝程式) 可以簡化裝置製造程序。裝置不是唯一的用戶端憑證，而是具有可讓裝置僅連線 5 分鐘 AWS IoT 的臨時憑證。在這 5 分鐘的時間範圍期間，信任的使用者會取得連線時間更長的唯一用戶端憑證，並將其安裝在裝置上。宣告憑證的有限連線時間可將憑證洩露的風險降至最低。

如需詳細資訊，請參閱 [the section called “由信任的使用者佈建”](#)。

- 最終使用者無法使用應用程式，在其 IoT 裝置上安裝憑證

如果先前的選項都無法在您的 IoT 解決方案中運作，[透過要求佈建](#) 程序是一個選項。透過此程序，您的 IoT 裝置會擁有由機群中其他裝置共用的宣告憑證。裝置首次連線至宣告憑證時，會使用其佈建範本 AWS IoT 註冊該裝置，並發出其唯一用戶端憑證以供後續存取 AWS IoT。

此選項可在裝置連線時自動佈建裝置 AWS IoT，但如果宣告憑證遭到破壞，則可能會產生較大的風險。如果宣告憑證洩露，您可以停用該憑證。停用宣告憑證可防止未來不得註冊具有該宣告憑證的所有裝置。不過，停用宣告憑證並不會封鎖已佈建的裝置。

如需詳細資訊，請參閱 [the section called “透過要求佈建”](#)。

## 佈建 AWS IoT 中的裝置

當您使用佈建裝置時 AWS IoT，您必須建立資源，讓您的裝置 AWS IoT 能夠安全地通訊。可以建立其他資源來協助您管理裝置機群。可以在佈建過程中建立下列資源：

- IoT 物件。

IoT 的東西是 AWS IoT 設備註冊表中的條目。每個物件都有一個唯一名稱和一組屬性，並與實體裝置相關聯。物件可以使用物件類型來定義，也可以分組為物件群組。如需詳細資訊，請參閱 [管理裝置 AWS IoT](#)。

雖然不一定要建立物件，但建立物件可讓您依物件類型、物件群組和物件屬性搜尋裝置，更有效地管理您的裝置機群。如需詳細資訊，請參閱 [機群索引](#)。

### Note

欲為物件的連線狀態資料建立索引，請佈建您的物件並加以配置，使物件名稱符合「Connect 線」要求中使用的用戶端 ID。

- X.509 憑證。

裝置使用 X.509 憑證與執行相互驗證。AWS IoT 您可以註冊現有憑證，也可以為您 AWS IoT 產生並註冊新憑證。您可以將憑證附加至代表裝置的物件，將該憑證與裝置產生關聯。您也必須將憑證和關聯的私有金鑰複製到裝置。連線至時，裝置會顯示憑證 AWS IoT。如需詳細資訊，請參閱 [身分驗證](#)。

- IoT 政策。

IoT 政策定義了裝置可在 AWS IoT 中執行哪些操作。IoT 政策會附加至裝置憑證。當設備向其提供憑證時 AWS IoT，系統會將在規則中指定的權限授與該設備。如需詳細資訊，請參閱 [授權](#)。每個裝置都需要憑證才能與 AWS IoT 進行通訊。

AWS IoT 支援使用佈建範本的自動化叢集佈建。佈建範本說明佈建裝置所 AWS IoT 需的資源。範本包含的變數可讓您使用一個範本來佈建多個裝置。佈建裝置時，您可以使用字典或 map，為裝置特定的變數指定值。若要佈建另一個裝置，請在字典中指定新值。

無論您的裝置是否具有唯一憑證 (及其關聯的私有金鑰)，您都可以使用自動佈建。

## 機群佈建 API

機群佈建中使用的 API 有數種類別：

- 這些控制平面 API 可建立和管理機群佈建範本，以及設定受信任的使用者政策。
  - [CreateProvisioning](#)。模板
  - [CreateProvisioningTemplateVersion](#)
  - [DeleteProvisioning](#)。模板
  - [DeleteProvisioningTemplateVersion](#)
  - [DescribeProvisioning](#)。模板
  - [DescribeProvisioningTemplateVersion](#)
  - [ListProvisioning](#)模板
  - [ListProvisioningTemplateVersions](#)
  - [UpdateProvisioning](#)。模板
- 受信任的使用者可以使用此控制平面功能來產生暫時上線要求。此暫時要求會在 Wi-Fi 組態或類似方法期間傳遞至裝置。
  - [CreateProvisioning](#)索償
- 裝置在佈建過程中使用的 MQTT API，具有內嵌在裝置中的佈建宣告憑證，或是由受信任使用者傳遞給它的佈建要求。
  - [the section called “CreateCertificateFromCsr”](#)
  - [the section called “CreateKeysAndCertificate”](#)
  - [the section called “RegisterThing”](#)

## 使用機群佈建來佈建沒有裝置憑證的裝置

透過使用 AWS IoT 叢集佈建，AWS IoT 可以在裝置首次連線時產生裝置憑證和私密金鑰，並將其安全地傳遞至 AWS IoT 裝置。AWS IoT 提供由 Amazon 根憑證授權單位 (CA) 簽署的用戶端憑證。

使用機群佈建的方式有兩種：

- [透過要求佈建](#)
- [由信任的使用者佈建](#)

## 透過要求佈建

裝置可以使用內嵌的佈建宣告憑證和私有金鑰 (這是特殊用途的憑證憑證) 來製造。如果這些憑證是在註冊的 AWS IoT，則服務可以將它們交換為裝置可用於一般作業的唯一裝置憑證。此程序包含以下步驟：

在您交付裝置之前

1. 呼叫 [CreateProvisioningTemplate](#) 來建立佈建範本。這個 API 傳回範本 ARN。如需詳細資訊，請參閱 [裝置佈建 MQTT API](#)。

您也可以直接在 AWS IoT 主控台中建立叢集佈建範本。

- a. 從導覽窗格中，選擇 Connect (連接)，然後選擇 Fleet provisioning templates (機群佈建範本)。
  - b. 選擇 Create template (建立範本)，並依照提示進行。
2. 建立要用作佈建宣告憑證的憑證和關聯私有金鑰。
  3. 將這些憑證註冊到限制憑證使用的 IoT 原則，AWS IoT 並將其關聯起來。下列範例 IoT 政策會限制使用與此政策相關聯的憑證來佈建裝置。

```
{
  "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. 在佈建裝置時，授予 AWS IoT 服務權限，以建立或更新您帳戶中的物件和憑證等 IoT 資源。透過將 AWS IoT Things Registration 受管政策附加到信任 AWS IoT 服務主體的 IAM 角色 (稱為佈建角色) 來執行此操作。
5. 製造裝置，並安全地將佈建宣告憑證內嵌其中。

裝置現已準備就緒，可以傳送到將安裝的位置以供使用。

#### Important

佈建要求私有金鑰應該始終加以保護，包括在裝置上。我們建議您使用 AWS IoT CloudWatch 指標和記錄來監控濫用的跡象。如果您偵測到誤用，請關閉佈建宣告憑證，使其無法用於裝置佈建。

#### 初始化裝置以供使用

1. 裝置會使用連線 [AWS IoT 裝置 SDK](#)、[行動 SDK](#) 和 [AWS IoT 裝置用戶端](#) 到裝置上安裝的佈建宣告憑證，並 AWS IoT 使用進行驗證。

#### Note

為了安全起見，由 [CreateCertificateFromCsr](#) 和 [CreateKeysAndCertificate](#) 傳回的 `certificateOwnershipToken` 會在一小時後過期。必須在 `certificateOwnershipToken` 過期之前呼叫 [RegisterThing](#)。如果由 [CreateCertificateFromCsr](#) 或者 [CreateKeysAndCertificate](#) 建立的憑證尚未啟用，並且在權杖過期時，尚未附加到政策或物件，憑證會遭到刪除。如果字符過期，裝



置可以再次呼叫 [CreateCertificateFromCsr](#) 或 [CreateKeysAndCertificate](#) 來產生新憑證。

2. 裝置會使用其中一個選項取得永久憑證和私密金鑰。裝置將使用憑證和金鑰進行所有 future 的驗證 AWS IoT。
  - a. 呼叫 [CreateKeysAndCertificate](#) 以使用憑證授權單位建立新的 AWS 憑證和私密金鑰。  
或
  - b. 呼叫 [CreateCertificateFromCsr](#) 以從保持私有金鑰安全的憑證簽署要求產生憑證。
3. 從裝置呼叫 [RegisterThing](#)，以向 AWS IoT 註冊裝置，並建立雲端資源。

同時，機群佈建服務會使用佈建範本來定義並建立雲端資源，例如 IoT 物件。範本可以指定物件所屬的屬性與群組。必須先有物件群組，才能將新物件加入其中。

4. 在裝置上儲存永久憑證後，裝置必須中斷與佈建宣告憑證初始化的工作階段連線，然後使用永久憑證重新連線。

裝置現在已準備就緒，可以正常通訊 AWS IoT。

## 由信任的使用者佈建

在許多情況下，當受信任的使 AWS IoT 用者 (例如使用者或安裝技術人員) 使用行動應用程式在其部署位置設定裝置時，裝置會首次連線至裝置。

### Important

您必須管理信任之使用者的存取權和許可，才能執行此程序。其中一種方法是為信任的使用者提供和維護帳戶，以驗證他們，並授予他們存取執行此程序所需的 AWS IoT 功能和 API 操作。

在您交付裝置之前

1. 呼叫 [CreateProvisioningTemplate](#) 以建立佈建範本，並傳回其 *templateArn* 和 *templateName*。
2. 建立受信任使用者用來啟動佈建程序的 IAM 角色。佈建範本只允許該使用者佈建裝置。例如：

```
{
```

```
"Effect": "Allow",
"Action": [
    "iot:CreateProvisioningClaim"
],
"Resource": [
    "arn:aws:iot:aws-region:aws-account-id:provisioningtemplate/templateName"
]
}
```

3. 在佈建裝置時，授予 AWS IoT 服務權限以建立或更新 IoT 資源，例如帳戶中的項目和憑證。您可以透過將 `AWSIoTThingsRegistration` 受管政策附加到信任 AWS IoT 服務主體的 IAM 角色 (稱為佈建角色) 來達到此目的。
4. 提供識別受信任使用者的方法，例如向他們提供可驗證使用者的帳戶，並授權他們與註冊其裝置所需的 AWS API 作業互動。

### 初始化裝置以供使用

1. 信任的使用者會登入您的佈建行動應用程式或 Web 服務。
2. 行動應用程式或 Web 應用程式會使用 IAM 角色並呼叫 [CreateProvisioningClaim](#)，以從 AWS IoT 取得暫時佈建宣告憑證。

#### Note

為了安全起見，`CreateProvisioningClaim` 傳回的暫時佈建宣告憑證會在五分鐘後過期。在暫時佈建宣告憑證到期之前，下列步驟必須成功傳回有效的憑證。暫時佈建宣告憑證不會出現在您帳戶的憑證清單中。

3. 行動應用程式或 Web 應用程式會將暫時佈建宣告憑證以及任何必要的設定資訊 (例如 Wi-Fi 認證) 提供給裝置。
4. 裝置會使用暫時佈建宣告憑證，使用 AWS IoT 來連線至 [AWS IoT 裝置 SDK](#)、[行動 SDK](#) 和 [AWS IoT 裝置用戶端](#)。
5. 裝置會在連線至 AWS IoT 暫時佈建宣告憑證後在五分鐘內，使用其中一個選項取得永久憑證和私密金鑰。設備將使用這些選項返回的證書和密鑰，以便將 future 進行所有身份驗證 AWS IoT。
  - a. 呼叫 [CreateKeysAndCertificate](#) 以使用憑證授權單位建立新的 AWS 憑證和私密金鑰。  
或
  - b. 呼叫 [CreateCertificateFromCsrf](#) 以從保持私有金鑰安全的憑證簽署要求產生憑證。

**Note**

請記住 [CreateKeysAndCertificate](#) 或 [CreateCertificateFromCsr](#) 必須在連接到 AWS IoT 臨時佈建聲明證書的五分鐘內返回有效證書。

6. 裝置會呼叫 [RegisterThing](#) 以向裝置註冊 AWS IoT 並建立雲端資源。

同時，機群佈建服務會使用佈建範本來定義並建立雲端資源，例如 IoT 物件。範本可以指定物件所屬的屬性與群組。必須先有物件群組，才能將新物件加入其中。

7. 在裝置上儲存永久憑證之後，裝置必須中斷與使用暫時佈建宣告憑證起始的工作階段的連線，然後使用永久憑證重新連線。

裝置現在已準備就緒，可以正常通訊 AWS IoT。

## 將預先佈建掛接與 AWS CLI 搭配使用

下列程序會建立具有預先佈建掛接的佈建範本。這裡使用的 Lambda 函數是一個可修改的範例。

若要建立預先佈建掛接，並將其套用至佈建範本

1. 建立具有所定義輸入和輸出的 Lambda 函數。Lambda 函數是可高度自訂的。需有 `allowProvisioning` 和 `parameterOverrides` 才能建立預先佈建的掛接。如需建立 Lambda 函數的詳細資訊，請參閱 [AWS Lambda 搭配命 AWS 令列介面使用](#)。

以下是 Lambda 函數輸出的範例：

```
{
  "allowProvisioning": True,
  "parameterOverrides": {
    "incomingKey0": "incomingValue0",
    "incomingKey1": "incomingValue1"
  }
}
```

2. AWS IoT 使用以資源為基礎的政策來呼叫 Lambda，因此您必須 AWS IoT 授予呼叫 Lambda 函數的權限。

**⚠ Important**

請務必在連接到 Lambda 動作的政策的全域條件內容金鑰中包含 `source-arn` 或 `source-account`，以防止許可操作。如需此項目的詳細資訊，請參閱[預防跨服務混淆代理人](#)。

以下是使用 [add-permission](#) 將 IoT 許可授予您 Lambda 的範例。

```
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，將預先啟動設定掛接新增至樣板。

下列 CLI 範例使用 [create-provisioning-template](#) 建立具有預先啟動設定掛接的佈建範本：

```
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
```

此命令的輸出結果如下所示：

```
{  
  "templateArn": "arn:aws:iot:us-east-1:1234564789012:provisioningtemplate/myTemplate",  
  "defaultVersionId": 1,  
  "templateName": myTemplate  
}
```

您也可以從檔案載入參數，而不必一一輸入所有命令列參數值，以節省時間。如需詳細資訊，請參閱[從檔案載入 AWS CLI 參數](#)。以下顯示了擴展 JSON 格式的 `template` 參數：

```
{
```

```

"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"]}
    }
}

```

以下顯示了擴展 JSON 格式的 pre-provisioning-hook 參數：

```

{
    "targetArn" : "arn:aws:lambda:us-
east-1:765219403047:function:pre_provisioning_test",
    "payloadVersion" : "2020-04-01"
}

```

## 佈建具有裝置憑證的裝置

AWS IoT 當裝置上已有裝置憑證 (及相關私密金鑰) 時，提供三種佈建方式：

- 含佈建範本的單一物件佈建。如果您一次只需要佈建一部裝置，適合使用此選項。
- Just-in-time 佈建 (JITP)，其範本會在裝置首次連線時佈建該裝置。AWS IoT 如果您需要註冊大量裝置，但您沒有裝置的相關資訊，無法組合到大量佈建清單內，適合使用此選項。
- 大量註冊 此選項可讓您指定儲存在 S3 儲存貯體內的檔案中的單一物件佈建範本值清單。如果您有大量已知裝置且您可將其所需特性組合到清單內，適合使用此選項。

### 主題

- [單一物件佈建](#)

- [Just-in-time 佈建](#)
- [大量註冊](#)

## 單一物件佈建

若要佈建物件，請使用 [RegisterThing](#) API 或 `register-thing` CLI 命令。`register-thing` CLI 命令會使用以下引數：

`--template-body`

該佈建範本。

`--parameters`

佈建範本使用的名稱/值對參數清單 (JSON 格式) (例如，`{"ThingName" : "MyProvisionedThing", "CSR" : "csr-text"}`)。

請參閱 [佈建範本](#)。

[RegisterThing](#) 或 `register-thing` 返回資源的 ARN 和它創建的證書的文本：

```
{
  "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"
  }
}
```

若字典省略參數，則會使用預設值。若未指定預設值，則參數不會替換成值。

## Just-in-time 佈建

您可以使用 just-in-time 佈建 (JITP) 在裝置第一次嘗試連線時佈建裝置。AWS IoT 若要佈建裝置，您必須啟用自動註冊，而且您在佈建裝置時，其裝置憑證簽署所用的憑證授權機構憑證，必須與佈建範本建立關聯。佈建成功和錯誤記錄為 [裝置佈建指標](#) 在 Amazon CloudWatch 中。

主題

- [JITP 概觀](#)
- [使用佈建範本註冊 CA](#)
- [使用佈建立範本名稱來註冊 CA](#)

## JITP 概觀

當裝置嘗試使用已註冊 CA 憑證簽署的憑證進行連線時，會從 CA 憑證 AWS IoT 載入範本並使用該範本進行呼叫 [RegisterThing](#)。AWS IoT JITP 工作流程會先註冊狀態值為 PENDING\_ACTIVATION 的憑證。裝置佈建完成時，憑證狀態將變為 ACTIVE。

AWS IoT 定義下列可在佈建範本中宣告及參照的參數：

- `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`

這些佈建範本參數的值僅限於 JITP 會自佈建中的裝置憑證的主旨欄位擷取的值。憑證必須包含範本主體中所有參數的值。`AWS::IoT::Certificate::Id` 參數代表的是內部產生的 ID，而不是憑證內包含的 ID。您可以使用 AWS IoT 規則內的 `principal()` 函數取得此 ID 的值。

### Note

您可以使用 AWS IoT Core just-in-time 佈建 (JITP) 功能佈建裝置，而不必在裝置的第一個連線上傳送整個信任鏈。AWS IoT Core 雖不要求顯示憑證授權機構憑證，但需要裝置在連線至 AWS IoT Core 時傳送 [伺服器名稱指示 \(SNI\)](#) 延伸。

## 範例範本內文

以下 JSON 檔案為完整 JITP 範本的範例範本內文。



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

此範例範本會宣告擷取自憑證並使用在 Resources 部分的

AWS::IoT::Certificate::CommonName、AWS::IoT::Certificate::SerialNumber、AWS::IoT::Certificate::Id 佈建參數的值。JITP 工作流程接著使用此範本執行下列動作：

- 註冊憑證並將其狀態設為 PENDING\_ACTIVE。
- 建立一個物件資源。
- 建立一個政策資源。
- 將政策連接至憑證。
- 將憑證連接至物件。
- 更新憑證狀態為 ACTIVE。

如果憑證沒有 Parameters 的 templateBody。例如，如果 AWS::IoT::Certificate::Country 併入在範本中，但憑證沒有 Country 屬性，裝置佈建會失敗。

您也可以使用 CloudTrail 來疑難排解 JITP 範本的問題。如需 Amazon 中記錄的指標的相關資訊 CloudWatch，請參閱[裝置佈建指標](#)。如需佈建範本的詳細資訊，請參閱[佈建範本](#)。

**Note**

在佈建程序期間，just-in-time 佈建 (JITP) 會呼叫其他 AWS IoT 控制平面 API 作業。這些呼叫可能會超過針對您帳戶設定的 [AWS IoT 調節配額](#)，並導致調節呼叫。如有必要，請聯絡 [AWS 客戶支援](#)，以提高您的調節配額。

## 使用佈建範本註冊 CA

若要使用完整的佈建範本註冊 CA，請依照下列步驟執行：

1. 將您的佈建範本和角色 ARN 資訊 (如下列範例) 儲存為 JSON 檔案：

```
{
  "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
            \"AttributePayload\" : {\r\n
              \"version\" : \"v1\",\r\n
              \"serialNumber\" : {\r\n
                \"Ref\" : \"AWS::IoT::Certificate::SerialNumber\",\r\n
                \"ThingTypeName\" : \"lightBulb-versionA\",\r\n
                \"ThingGroups\" : [\r\n
                  {\r\n
                    \"Ref\" : \"AWS::IoT::Certificate::Country\",\r\n
                    \"AttributePayload\" : \"MERGE\",\r\n
                    \"ThingTypeName\" : \"REPLACE\",\r\n
                    \"ThingGroups\" : [\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
                              \"Status\" : \"ACTIVE\",\r\n
                              \"OverrideSettings\" : {\r\n
                                \"Status\" : \"DO_NOTHING\",\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"
}

```

在此範例中，`templateBody` 欄位的值必須是指定為溢出字串的 JSON 物件，且只能使用 [前述清單](#) 中的值。您可以使用各種工具來建立必要的 JSON 輸出，例如 `json.dumps` (Python) 或 `JSON.stringify` (節點)。 `roleARN` 欄位的值必須為連接 `AWSIoTThingsRegistration` 之角色的 ARN。此外，您的範本可以使用現有的 `PolicyName`，而不是範例中的內嵌 `PolicyDocument`。

2. 在使用 [RegisterCACertificate](#) API 操作或 [register-ca-certificate](#) CLI 命令來註冊 CA 憑證時。您將指定啟動佈建範本的目錄，以及您在上一個步驟中儲存的角色 ARN 資訊：

以下範例展示如何使用 AWS CLI 在 DEFAULT 模式下註冊 CA 憑證：

```

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

```

以下範例展示如何使用 AWS CLI 在 SNI\_ONLY 模式下註冊 CA 憑證：

```

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

```

如需詳細資訊，請參閱 [註冊 CA 憑證](#)。

3. (選用) 使用 [UpdateCACertificate](#) API 操作或 [update-ca-certificate](#) CLI 命令來更新 CA 構憑證的設定。

以下範例展示如何使用 AWS CLI 更新 CA 憑證：

```

aws iot update-ca-certificate --certificate-id caCertificateId
--new-auto-registration-status ENABLE --registration-config
file://your-template

```

## 使用佈建範本名稱來註冊 CA

若要使用佈建範本名稱註冊 CA，請依照下列步驟執行：

1. 將佈建範本內文儲存為 JSON 檔案。您可以在[範例範本內文](#)中找到範例範本內文。
2. 若要建立佈建範本，請使用[CreateProvisioning範本](#) API 或 [create-provisioning-template](#) CLI 指令：

```
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

對於 just-in-time 佈建 (JITP)，您必須在建立佈建範本 JITP 時指定要使用的範本類型。如需有關範本類型的詳細資訊，請參閱 AWS API 參考中的[CreateProvisioning範本](#)。

3. 若要使用範本名稱註冊 CA，請使用 [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
```

## 大量註冊

您可以使用 [start-thing-registration-task](#) 命令來大量註冊物件。此命令需要佈建範本、S3 儲存貯體名稱、金鑰名稱，以及可存取 S3 儲存貯體檔案的角色 ARN。S3 儲存貯體的檔案，內含用於替換範本參數的值。該檔案必須為換行分隔的 JSON 檔案。每行均包含用於註冊單一裝置的所有參數值。例如：

```
{"ThingName": "foo", "SerialNumber": "123", "CSR": "csr1"}  
{"ThingName": "bar", "SerialNumber": "456", "CSR": "csr2"}
```

下列與大量註冊相關的 API 操作也許有用：

- [ListThingRegistrationTasks](#)：列出目前的大量物件佈建工作。

- [DescribeThingRegistrationTask](#)：提供有關特定大量物件註冊工作的資訊。
- [StopThingRegistrationTask](#)：停止大量物件註冊工作。
- [ListThingRegistrationTask報告](#)：用於檢查大量物件註冊任務的結果與失敗。

#### Note

- 一次僅可執行一個大量註冊操作任務 (每個帳戶)。
- 大量註冊作業會呼叫其他 AWS IoT 控制平面 API 作業。這些呼叫可能會超過您帳戶中的 [AWS IoT 調節配額](#)，並導致調節錯誤。如有必要，請聯絡 [AWS 客戶 Support](#) 以提高 AWS IoT 節流配額。

## 佈建範本

佈建範本是 JSON 文件，它使用參數描述您的裝置必須使用與之互動的資源 AWS IoT。佈建範本內含兩個部分：Parameters 和 Resources。中有兩種類型的佈建範本 AWS IoT。一個用於 just-in-time 佈建 (JITP) 和大量註冊，第二個則用於叢集佈建。

### 主題

- [參數部分](#)
- [資源部分](#)
- [大量註冊的範本範例](#)
- [just-in-time佈建的範本範例 \(JITP\)](#)
- [機群佈建](#)

## 參數部分

Parameters 部分宣告 Resources 部分所使用的參數。每個參數都會宣告名稱、類型和可選的預設值。與範本一同傳入的字典若未包含該參數的值，將使用預設值。範本文件的 Parameters 部分如下：

```
{
  "Parameters" : {
    "ThingName" : {
```

```
        "Type" : "String"
    },
    "SerialNumber" : {
        "Type" : "String"
    },
    "Location" : {
        "Type" : "String",
        "Default" : "WA"
    },
    "CSR" : {
        "Type" : "String"
    }
}
}
```

此範本內文片段宣告四項參數：ThingName、SerialNumber、Location 和 CSR。這些參數均為類型 String。Location 參數宣告的預設值為 "WA"。

## 資源部分

範本主體的Resources區段會宣告裝置與之通訊所需的資源 AWS IoT：物件、憑證和一或多個 IoT 原則。每一資源都要指定一個邏輯名稱、一種類型和一組屬性。

邏輯名稱可讓您引用範本其他位置的資源。

類型則指定您欲宣告的資源種類。有效類型為：

- AWS::IoT::Thing
- AWS::IoT::Certificate
- AWS::IoT::Policy

您指定的屬性取決於您欲宣告的資源類型。

## 物件資源

物件資源宣告使用下列屬性：

- ThingName：字串。
- AttributePayload：選用。名稱/值對清單。
- ThingTypeName：選用。與該物件類型相關的字串。

- ThingGroups : 選用。該物件所屬的群組清單。
- BillingGroup : 選用。關聯帳單群組名稱的字串。
- PackageVersions : 選用。相關套件和版本名稱的字串。

## 憑證資源

您可以使用下列其中一種方式來指定憑證：

- 憑證簽署要求 (CSR)。
- 現有裝置憑證的憑證 ID。(只有憑證 ID 可與機群佈建範本搭配使用。)
- 搭配憑證授權機構憑證向 AWS IoT 註冊所建立的裝置憑證。如果您的多個憑證授權機構憑證註冊使用相同的主體欄位，您也必須輸入用於簽署該裝置憑證的憑證授權機構憑證。

### Note

若您使用範本宣告憑證，請透過這些方法其中之一。例如，若您使用 CSR，則無法同時指定憑證 ID 或裝置憑證。如需詳細資訊，請參閱 [X.509 用戶端憑證](#)。

如需詳細資訊，請參閱 [X.509 憑證概觀](#)。

憑證資源宣告使用下列屬性：

- CertificateSigningRequest : 字串。
- CertificateId : 字串。
- CertificatePem : 字串。
- CACertificatePem : 字串。
- Status : 選用。可以是 ACTIVE 或 INACTIVE 的字串。預設為 ACTIVE。

範例：

- CSR 指定的憑證：

```
{  
  "certificate" : {
```



```

    "Type" : "AWS::IoT::Certificate",
    "Properties" : {
      "CertificateSigningRequest": {"Ref" : "CSR"},
      "Status" : "ACTIVE"
    }
  }
}

```

- 以現有憑證 ID 指定的憑證：

```

{
  "certificate" : {
    "Type" : "AWS::IoT::Certificate",
    "Properties" : {
      "CertificateId": {"Ref" : "CertificateId"}
    }
  }
}

```

- 以現有憑證 .pem 和憑證授權機構憑證 .pem 指定的憑證：

```

{
  "certificate" : {
    "Type" : "AWS::IoT::Certificate",
    "Properties" : {
      "CACertificatePem": {"Ref" : "CACertificatePem"},
      "CertificatePem": {"Ref" : "CertificatePem"}
    }
  }
}

```

## 政策資源

政策資源宣告使用下列屬性之一：

- **PolicyName**：選用。字串。預設為政策文件雜湊。此 PolicyName 只能參考 AWS IoT 政策，而不是 IAM 政策。如果您正在使用現有 AWS IoT 策略，請為該 PolicyName 內容輸入策略的名稱。請勿包含 PolicyDocument 屬性。
- **PolicyDocument**：選用。指定為逸出字串的 JSON 物件。若未提供 PolicyDocument，則該政策必已建立。

**Note**

若有 Policy 部分，必須指定 PolicyName 或 PolicyDocument 其一。

## 覆寫設定

若範本指定已存在的資源，OverrideSettings 部分可讓您指定欲採取的動作：

### DO\_NOTHING

保留原本資源。

### REPLACE

以範本指定的資源來取代原有資源。

### FAIL

因 ResourceConflictsException 造成請求失敗。

### MERGE

僅適用 ThingGroups 的 AttributePayload 及 thing 屬性。將物件現有屬性或群組成員資格，與範本指定的屬性合併。

當您宣告物件資源時，可為下列屬性指定 OverrideSettings：

- ATTRIBUTE\_PAYLOAD
- THING\_TYPE\_NAME
- THING\_GROUPS

當您宣告憑證資源時，可為 OverrideSettings 屬性指定 Status。

OverrideSettings 不適用於政策資源。

## 資源範例

下列範本片段宣告一個物件、一個憑證及一個政策：

```
{
```

```

"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\"] }]}"}
    }
  }
}
}

```

物件以下列項目進行宣告：

- 邏輯名稱 "thing"。
- 類型 AWS::IoT::Thing。
- 物件的一組屬性。

物件的屬性包含物件名稱、一組屬性、選用的物件類型名稱，以及該物件所屬的物件群組選用清單。

`{"Ref": "parameter-name"}` 會參考這些參數。在評估範本時，參數將取代為與範本一同傳入的字典的參數值。

憑證以下列項目進行宣告：

- 邏輯名稱 "certificate"。
- 類型 `AWS::IoT::Certificate`。
- 一組屬性。

這些屬性包括憑證的 CSR 及設定為 ACTIVE 的狀態。與範本一同傳入的字典內，CSR 文字以參數傳遞。

政策以下列項目進行宣告：

- 邏輯名稱 "policy"。
- 類型 `AWS::IoT::Policy`。
- 現有政策名稱或政策文件。

## 大量註冊的範本範例

下列 JSON 檔案為以 CSR 指定憑證的完整佈建範本的其中一個範例：

(PolicyDocument 欄位值必須是指定為逸出字串的 JSON 物件。)

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

```

## just-in-time 佈建的範本範例 (JITP)

下列 JSON 檔案為以憑證 ID 指定現有憑證的完整佈建範本的其中一個範例：

```

{
  "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

您必須在用於 JIT 佈建的範本中使用 CertificateId。

如需有關佈建範本類型的詳細資訊，請參閱 AWS API 參考資料 [CreateProvisioningTemplate](#) 中的。

如需如何使用此範本進行啟動設定的相關資訊，請參閱：[Just-in-Time just-in-time](#) 佈建。

## 機群佈建

叢集佈建範本可用 AWS IoT 來設定雲端和裝置組態。這些範本會使用與 JITP 和大量註冊範本相同的參數和資源。如需詳細資訊，請參閱 [佈建範本](#)。機群佈建範本可以包含一個 Mapping 區段和一個 DeviceConfiguration 區段。您可以使用機群佈建範本內的內建函數來產生裝置特定的組態。機群佈建範本是具名資源，由 ARN 識別 (例如，arn:aws:iot:us-west-2:1234568788:provisioningtemplate/*templateName*)。

### 映射項目

選用的 Mappings 區段會比對索引鍵與對應的一組命名值。例如，如果您要根據「AWS 區域」設定值，您可以建立一個使用 AWS 區域名稱作為索引鍵的對映，並包含您要為每個特定區域指定的值。您可以使用 Fn::FindInMap 內部函數來擷取映射中的值。

您不能在 Mappings 區段中包含參數、虛擬參數或呼叫內部函數。

### 裝置組態

裝置組態區段包含您要在佈建時傳送至裝置的任意資料。例如：

```
{
  "DeviceConfiguration": {
    "Foo": "Bar"
  }
}
```

```
}  
}
```

如果您使用 JavaScript 物件標記法 (JSON) 承載格式將訊息傳送至裝置，AWS IoT Core 請將此資料格式化為 JSON。如果您使用的是簡潔的二進位物件表示 (CBOR) 有效負載格式，請將此資料 AWS IoT Core 格式化為 CBOR。DeviceConfiguration 區段不支援巢狀 JSON 物件。

## 內部函數

內部函數用於佈建範本的任何區段，但 Mappings 區段除外。

### Fn::Join

將一組值附加至單一值，並以指定的分隔符號隔開。如果分隔符號是空白字串，系統即會串連這些值，而不使用分隔符號。

#### Important

[the section called “政策資源”](#) 不支援 Fn::Join。

### Fn::Select

依索引從物件清單傳回單一物件。

#### Important

Fn::Select 不會檢查 null 值或索引是否超出陣列邊界。這兩種情況都會導致佈建錯誤，因此請確保您選擇了有效的索引值，並且清單中包含非空值。

### Fn::FindInMap

傳回對應 Mappings 區段所宣告之兩個層級映射之索引鍵的值。

### Fn::Split

將字串分割成字串值清單，以便您可以從字串清單中選取元素。您可以指定分隔符號來決定分割字串的位置 (例如，逗號)。分割字串後，請使用 Fn::Select 來選取元素。

例如，如果以逗號分隔的子網路 ID 字串匯入至您的堆疊範本，您可以在每個逗號處分割字串。從子網路 ID 清單中，使用 Fn::Select 來指定資源的子網路 ID。



## Fn::Sub

用您指定的值替代輸入字串中的變數。您可以使用此函數來建構命令或輸出，其中包含建立或更新堆疊後才可供使用的數值。

### 機群佈建的範本範例

```
{
  "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

您可以更新現有的佈建範本，以新增 [pre-provisioning hook](#)。

## 預先佈建掛接

AWS 建議在建立佈建範本時使用預先佈建勾點功能，以進一步控制您帳戶上載的裝置和裝置數量。預先佈建掛接是 Lambda 函數，會先驗證從裝置傳遞的參數，然後才能佈建裝置。此 Lambda 函數必

須存在於您的帳戶中，才能佈建裝置，因為每次裝置透過 [the section called “RegisterThing”](#) 傳送要求時，都會呼叫該函數。

### ⚠ Important

請務必在連接到 Lambda 動作的政策的全域條件內容金鑰中包含 `source-arn` 或 `source-account`，以防止許可操作。如需此項目的詳細資訊，請參閱[預防跨服務混淆代理人](#)。

對於要佈建的裝置，您的 Lambda 函數必須接受輸入物件，並傳回本節所述的輸出物件。只有當 Lambda 函數透過 `"allowProvisioning": True` 傳回物件時，才會繼續佈建。

## 預先佈建掛接輸入

AWS IoT 當裝置向註冊時，會將此物件傳送至 Lambda 函數 AWS IoT。

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

傳遞給 Lambda 函數的 `parameters` 物件包含在 [the section called “RegisterThing”](#) 請求承載中傳遞之 `parameters` 引數中的屬性。

## 預先佈建掛接傳回值

Lambda 函數必須傳回回應，指出它是否已授權佈建請求，以及要覆寫的任何屬性值。

以下是預先佈建函數成功回應的範例。

```
{
  "allowProvisioning": true,
  "parameterOverrides" : {
```

```
        "Key": "newCustomValue",
        ...
    }
}
```

"parameterOverrides" 值將新增至 [the section called "RegisterThing"](#) 請求承載的 "parameters" 參數。

#### Note

- 如果 Lambda 函數失敗，佈建請求會失敗，ACCESS\_DENIED 並將錯誤記錄到 CloudWatch 日誌中。
- 如果 Lambda 函數沒有在回覆中傳回 "allowProvisioning": "true"，則佈建要求會失敗，顯示 ACCESS\_DENIED。
- 該 Lambda 函數必須在 5 秒內完成執行並返回，否則佈建請求將失敗。

## 預先佈建掛接 Lambda 範例

### Python

Python 中預先佈建掛接 Lambda 的範例。

```
import json

def pre_provisioning_hook(event, context):
    print(event)

    return {
        'allowProvisioning': True,
        'parameterOverrides': {
            'DeviceLocation': 'Seattle'
        }
    }
```

### Java

Java 中預先佈建掛接 Lambda 的範例。

處理常式類別：

```
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;
    }
}
```

請求類別：

```
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;
}
```

回應類別：

```
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;
}
```

## JavaScript

預先佈建掛接 Lambda 的範例。JavaScript

```
exports.handler = function(event, context, callback) {
    console.log(JSON.stringify(event, null, 2));
    var reply = {
        allowProvisioning: true,
        parameterOverrides: {
            DeviceLocation: 'Seattle'
        }
    };
    callback(null, reply);
}
```

# 使用憑證提供者自行管理 AWS IoT Core 憑證簽署

您可以建立 AWS IoT Core 憑證提供者，以在 AWS IoT 叢集佈建中簽署憑證簽署要求 (CSR)。憑證提供者會參考 Lambda 函數和 [CreateCertificateFromCsrMQTT API](#) 來進行叢集佈建。Lambda 函數接受 CSR 並傳回已簽署的用戶端憑證。

如果您沒有憑證提供者 AWS 帳戶，則會在叢集佈建中呼叫 [CreateCertificateFromCsr MQTT API](#)，以從 CSR 產生憑證。建立憑證提供者之後，[CreateCertificateFromCsr MQTT API](#) 的行為將會變更，且對此 [MQTT API](#) 的所有呼叫都會叫用憑證提供者來發行憑證。

透過 AWS IoT Core 憑證提供者，您可以實作使用私有憑證授權單位 (CA) 的解決方案 [AWS Private CA](#)，例如其他公開信任的 CA 或您自己的公開金鑰基礎結構 (PKI) 來簽署 CSR。此外，您可以使用憑證提供者來自訂用戶端憑證的欄位，例如有效期間、簽署演算法、發行者和延伸。

## Important

每個憑證提供者只能建立一個憑證提供者 AWS 帳戶。簽署行為變更會套用至呼叫 [CreateCertificateFromCsr MQTT API](#) 的整個叢集，直到您從您的憑證提供者刪除為止。AWS 帳戶

在本主題中：

- [自我管理憑證簽署在叢集佈建中的運作方式](#)
- [憑證提供程式 Lambda 函數](#)
- [憑證提供者 Lambda 函數傳回值](#)
- [Lambda 函數範例](#)
- [用於叢集佈建的自我管理憑證簽署](#)
- [AWS CLI 憑證提供者的指令](#)

## 自我管理憑證簽署在叢集佈建中的運作方式

### 重要概念

下列概念提供詳細資料，可協助您瞭解自我管理憑證簽署在 AWS IoT 叢集佈建中的運作方式。如需詳細資訊，請參閱[使用叢集佈建佈建來佈建沒有裝置憑證的裝置](#)。

## AWS IoT 叢集佈建

透過 AWS IoT 叢集佈建 (叢集佈建的簡稱)，可在裝置首次連線至裝置時 AWS IoT Core 產生並安全地傳遞裝置憑證給 AWS IoT Core 裝置。您可以使用叢集佈建將沒有裝置憑證的裝置連線到 AWS IoT Core。

### 憑證簽署要求

在叢集佈建過程中，裝置會 AWS IoT Core 透過[叢集佈建 MQTT API](#) 向其發出要求。此要求包含憑證簽署要求 (CSR)，此要求會經過簽署以建立用戶端憑證。

### AWS 叢集佈建中的受管憑證簽署

AWS managed 是叢集佈建中憑證簽署的預設設定。透過 AWS 受管理的憑證簽署，AWS IoT Core 將會使用自己的 CA 來簽署 CSR。

### 叢集佈建中的自我管理憑證簽署

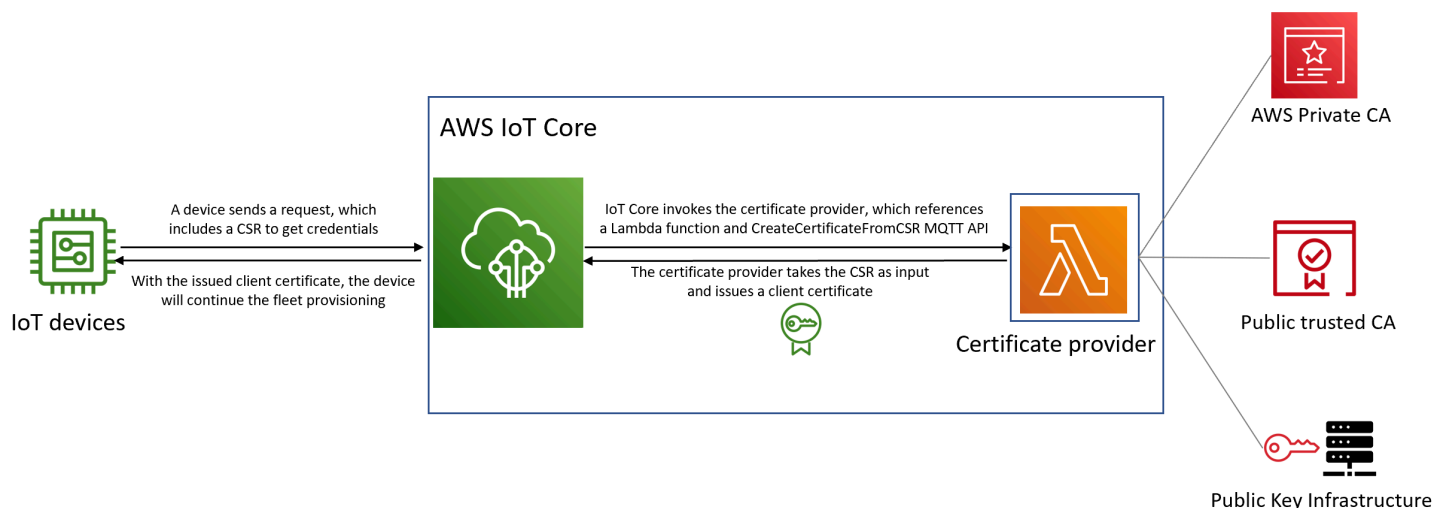
自我管理是叢集佈建中憑證簽署的另一個選項。透過自我管理的憑證簽署，您可以建立 AWS IoT Core 憑證提供者來簽署 CSR。您可以使用自我管理憑證簽署，使用 AWS 私有 CA 產生的 CA、其他公開信任的 CA 或您自己的公開金鑰基礎結構 (PKI) 來簽署 CSR。

### AWS IoT Core 憑證提供者

AWS IoT Core 憑證提供者 (憑證提供者的簡稱) 是客戶管理的資源，用於叢集佈建中的自我管理憑證簽署。

## 圖表

下圖是自我憑證簽署在 AWS IoT 叢集佈建中如何運作的簡化說明。





- 當新的 IoT 裝置製造或引入叢集時，它需要用戶端憑證來驗證本身 AWS IoT Core。
- 作為叢集佈建程序的一部分，裝置會透過[叢集佈建 MQTT](#) API 向 AWS IoT Core 用戶端憑證提出要求。此要求包含憑證簽署要求 (CSR)。
- AWS IoT Core 呼叫憑證提供者，並將 CSR 作為輸入傳遞給提供者。
- 憑證提供者會將 CSR 作為輸入，並發行用戶端憑證。

對於 AWS 受管憑證簽署，AWS IoT Core 請使用自己的 CA 簽署 CSR 並發行用戶端憑證。

- 使用已發行的用戶端憑證，裝置將繼續進行叢集佈建，並與之建立安全連線 AWS IoT Core。

## 憑證提供程式 Lambda 函數

AWS IoT Core 當裝置向 Lambda 函數註冊時，會將下列物件傳送至 Lambda 函數。的值 `certificateSigningRequest` 是要求中所提供之[隱私權增強郵件 \(PEM\) 格式](#)的 CSR。 `CreateCertificateFromCsrprincipalId` 這是提出要求 AWS IoT Core 時用來連線的主體識別 `CreateCertificateFromCsr` 碼。 `clientId` 是為 MQTT 連線設定的用戶端識別碼。

```
{
  "certificateSigningRequest": "string",
  "principalId": "string",
  "clientId": "string"
}
```

## 憑證提供者 Lambda 函數傳回值

Lambda 函數必須傳回包含該 `certificatePem` 值的回應。以下是成功回應的範例。AWS IoT Core 將使用傳回值 (`certificatePem`) 來建立憑證。

```
{
  "certificatePem": "string"
}
```

如果註冊成功，`CreateCertificateFromCsr` 將在 `CreateCertificateFromCsr` 響應 `certificatePem` 中返回相同的。如需詳細資訊，請參閱的回應承載範例 [CreateCertificateFromCsr](#)。

## Lambda 函數範例

在建立憑證提供者之前，您必須先建立 Lambda 函數來簽署 CSR。以下是 Python 中的一個示例 Lambda 函數。此函數呼叫 AWS Private CA 使用私有 CA 和 SHA256WITHRSA 簽署演算法來簽署輸入 CSR。返回的客戶端證書有效期為一年。如需有關以 AWS Private CA 及如何建立私有 CA 的詳細資訊，請參閱 [什麼是 AWS 私有 CA？](#) 並 [創建一個私有 CA](#)。

```
import os
import time
import uuid
import boto3

def lambda_handler(event, context):
    ca_arn = os.environ['CA_ARN']
    csr = (event['certificateSigningRequest']).encode('utf-8')

    acmpca = boto3.client('acm-pca')
    cert_arn = acmpca.issue_certificate(
        CertificateAuthorityArn=ca_arn,
        Csr=csr,
        Validity={"Type": "DAYS", "Value": 365},
        SigningAlgorithm='SHA256WITHRSA',
        IdempotencyToken=str(uuid.uuid4())
    )['CertificateArn']

    # Wait for certificate to be issued
    time.sleep(1)
    cert_pem = acmpca.get_certificate(
        CertificateAuthorityArn=ca_arn,
        CertificateArn=cert_arn
    )['Certificate']

    return {
        'certificatePem': cert_pem
    }
```

### Important

- Lambda 函數傳回的憑證必須具有與憑證簽署要求 (CSR) 相同的主體名稱和公開金鑰。
- Lambda 函數必須在 5 秒內完成執行。
- Lambda 函數必須 AWS 帳戶 與憑證提供者資源位於相同的區域中。

- AWS IoT 服務主體必須被授與 Lambda 函數的叫用權限。為了避免[混淆的副問題](#)，我們建議您設置sourceArn和調sourceAccount用權限。如需詳細資訊，請參閱[預防跨服務混淆代理人](#)。

以下 `InvokeLambda` 以資源為基礎的政策範例授與 AWS IoT 呼叫 Lambda 函數的權限：

```
{
  "Version": "2012-10-17",
  "Id": "InvokePermission",
  "Statement": [
    {
      "Sid": "LambdaAllowIotProvider",
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
      "Condition": {
        "StringEquals": {
          "AWS:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "AWS:SourceArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
        }
      }
    }
  ]
}
```

## 用於叢集佈建的自我管理憑證簽署

您可以選擇使用 AWS CLI 或 AWS Management Console 進行叢集佈建的自我管理憑證簽署。

### AWS CLI

若要選擇自我管理憑證簽署，您必須建立 AWS IoT Core 憑證提供者，以便在叢集佈建中簽署 CSR。AWS IoT Core 叫用憑證提供者，此提供者會接受 CSR 作為輸入並傳回用戶端憑證。若要建立憑證提供者，請使用 `CreateCertificateProvider` API 作業或 `create-certificate-provider` CLI 命令。

**Note**

建立憑證提供者之後，[叢集佈建的 CreateCertificateFromCsr API](#) 行為將會變更，以便所有呼叫都 CreateCertificateFromCsr 會叫用憑證提供者來建立憑證。建立憑證提供者之後，此行為可能需要幾分鐘的時間才會變更。

```
aws iot create-certificate-provider \  
    --certificateProviderName my-certificate-provider \  
    --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-  
function-1 \  
    --accountDefaultForOperations CreateCertificateFromCsr
```

以下顯示此命令的範例輸出：

```
{  
  "certificateProviderName": "my-certificate-provider",  
  "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-  
certificate-provider"  
}
```

如需詳細資訊，請參閱 [CreateCertificateProvider](#) AWS IoT API 參考。

## AWS Management Console

若要使用選擇自我管理憑證簽署 AWS Management Console，請依照下列步驟執行：

1. 前往 [AWS IoT 主控台](#)。
2. 在左側導覽列的 [安全性] 下，選擇 [憑證簽署]。
3. 在 [憑證簽署] 頁面的 [憑證簽署詳細資料] 下，選擇 [編輯憑證簽署方法]。
4. 在 [編輯憑證簽署方法] 頁面的 [憑證簽署方法] 下，選擇 [自我管理]。
5. 在「自我管理設定」區段中，輸入憑證提供者的名稱，然後建立或選擇 Lambda 函數。
6. 選擇 [更新憑證簽署]。

## AWS CLI 憑證提供者的指令

### 建立憑證提供者

若要建立憑證提供者，請使用 `CreateCertificateProvider` API 作業或 `create-certificate-provider` CLI 命令。

#### Note

建立憑證提供者之後，[叢集佈建的 `CreateCertificateFromCsr` API](#) 行為將會變更，以便所有呼叫都 `CreateCertificateFromCsr` 會叫用憑證提供者來建立憑證。建立憑證提供者之後，此行為可能需要幾分鐘的時間才會變更。

```
aws iot create-certificate-provider \  
    --certificateProviderName my-certificate-provider \  
    --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-  
function-1 \  
    --accountDefaultForOperations CreateCertificateFromCsr
```

以下顯示此命令的範例輸出：

```
{  
  "certificateProviderName": "my-certificate-provider",  
  "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-  
certificate-provider"  
}
```

如需詳細資訊，請參閱 [CreateCertificateProvider](#) AWS IoT API 參考。

### 更新憑證提供者

若要更新憑證提供者，請使用 `UpdateCertificateProvider` API 作業或 `update-certificate-provider` CLI 命令。

```
aws iot update-certificate-provider \  
    --certificateProviderName my-certificate-provider \  
    --lambdaFunctionArn arn:aws:lambda:us-east-1:123456789012:function:my-  
function-2 \  
    --accountDefaultForOperations CreateCertificateFromCsr
```

以下顯示此命令的範例輸出：

```
{
  "certificateProviderName": "my-certificate-provider",
  "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
}
```

如需詳細資訊，請參閱 [UpdateCertificateProvider](#) AWS IoTAPI 參考。

## 說明憑證提供者

若要描述憑證提供者，請使用 DescribeCertificateProvider API 作業或 describe-certificate-provider CLI 命令。

```
aws iot describe-certificate-provider --certificateProviderName my-certificate-provider
```

以下顯示此命令的範例輸出：

```
{
  "certificateProviderName": "my-certificate-provider",
  "lambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:my-function",
  "accountDefaultForOperations": [
    "CreateCertificateFromCsr"
  ],
  "creationDate": "2022-11-03T00:15",
  "lastModifiedDate": "2022-11-18T00:15"
}
```

如需詳細資訊，請參閱 [DescribeCertificateProvider](#) AWS IoTAPI 參考。

## 刪除憑證提供者

若要刪除憑證提供者，請使用 DeleteCertificateProvider API 作業或 delete-certificate-provider CLI 命令。如果您刪除憑證提供者資源，CreateCertificateFromCsr 將會繼續執行的行為，並 AWS IoT 建立 AWS IoT 由 CSR 簽署的憑證。

```
aws iot delete-certificate-provider --certificateProviderName my-certificate-provider
```

此命令不會產生任何輸出。

如需詳細資訊，請參閱 [DeleteCertificateProvider](#) AWS IoTAPI 參考。

## 列出憑證提供者

若要列出您中的憑證提供者 AWS 帳戶，請使用 ListCertificateProviders API 作業或 list-certificate-providers CLI 命令。

```
aws iot list-certificate-providers
```

以下顯示此命令的範例輸出：

```
{
  "certificateProviders": [
    {
      "certificateProviderName": "my-certificate-provider",
      "certificateProviderArn": "arn:aws:iot:us-east-1:123456789012:certificateprovider:my-certificate-provider"
    }
  ]
}
```

如需詳細資訊，請參閱 [ListCertificateProviderAWS IoTAPI](#) 參考。

## 為安裝裝置的使用者建立 IAM 政策和角色

### Note

這些程序僅供 AWS IoT 主控台指示時使用。  
若要從主控台前往此頁面，請開啟[建立新的佈建範本](#)。

### 為什麼不能在 AWS IoT 控制台中完成此操作？

為了獲得最安全的體驗，IAM 動作會在 IAM 主控台中執行。本區段中的程序會逐步引導您建立使用佈建範本所需的 IAM 角色和政策。

## 為安裝裝置的使用者建立 IAM 政策

此程序說明如何建立 IAM 政策，以授權使用者使用佈建範本安裝裝置。

執行此程序時，您將在 IAM 主控台和主控台之間切換。AWS IoT 我們建議您在執行此程序時同時開啟兩個主控台。

為安裝裝置的使用者建立 IAM 政策

1. 請開啟 [IAM 主控台的策略中樞](#)。
2. 選擇建立政策。
3. 在建立政策頁面上，選擇 JSON 標籤。
4. 切換至您選擇 [設定使用者原則和角色] 的 AWS IoT 主控台頁面。
5. 在 Sample provisioning policy (範例佈建政策) 中選擇 Copy (複製)。
6. 切換回 IAM 主控台。
7. 在 JSON 編輯器中，貼上您從 AWS IoT 主控台複製的政策。這項政策特定於您在 AWS IoT 主控台中建立的範本。
8. 若要繼續，請選擇 Next: Tags (下一步：標籤)。
9. 針對您要新增到此政策的每個標籤在 Add tags (Optional) (新增標籤 (選擇性)) 頁面上選擇 Add tag (新增標籤)。如果您沒有要新增任何標籤，則可省略此步驟。
10. 若要繼續，請選擇 Next: Review (下一步：檢閱)。
11. 在 Review Policy (檢閱政策) 頁面上，執行下列動作：
  - a. 針對 Name\* (姓名\*)，請輸入能協助您記住政策用途的政策名稱。  
請記住您提供給此政策的名稱，因為後續程序仍會使用該名稱。
  - b. 您可以為您建立的政策選擇性地輸入描述。
  - c. 檢閱本政策的其餘部分及其標籤。
12. 請選擇 Create Policy (建立政策) 來完成您的政策建立程序。

建立新政策之後，請繼續 [the section called “為安裝裝置的使用者建立 IAM 角色”](#) 以建立您要連接到此政策的使用者角色項目。

## 為安裝裝置的使用者建立 IAM 角色

這些步驟說明如何建立 IAM 角色，以驗證使用佈建範本安裝裝置的使用者。

為安裝裝置的使用者建立 IAM 政策

1. 開啟 [IAM 主控台中 Roles \(角色\) 頁面](#)。



2. 選擇建立角色。
3. 在 Select trusted entity (選擇信任實體) 選擇您要授予存取權的信任實體類型，這些實體類型能存取您建立範本。
4. 選擇或輸入您要授予存取許可的信任實體識別，然後選擇 Next (下一頁)。
5. 在 Add permissions (新增許可) 頁面的 Permission policies (許可政策) 上，於搜尋方塊中輸入您在[上一個程序中](#)建立的政策名稱。
6. 針對政策清單，請選擇您在第一個程序所建立的政策，然後選擇 Next (下一頁)。
7. 在 Name, review, and create (命名、檢閱和建立) 區段上，執行以下作業：
  - a. 針對 Role name (角色名稱)，請輸入可協助您記住此角色用途的角色名稱。
  - b. 針對 Description (描述)，您可以選擇輸入角色的選擇性描述。不需要此操作即可繼續。
  - c. 檢閱 Step 1 (步驟 1) 和 Step 2 (步驟 2) 的值。
  - d. 針對 Add tags (Optional) (新增標籤 (選擇性))，您可以選擇新增標籤至此角色。不需要此操作即可繼續。
  - e. 確認此頁面上的資訊是否完整無誤，然後選擇 Create role (建立角色)。

建立新角色之後，請返回 AWS IoT 主控台以繼續建立範本。

## 更新現有政策以授權新範本

下列步驟說明如何將新範本新增至 IAM 政策，以授權使用者使用佈建範本安裝裝置。

將新的範本新增至現有 IAM 政策

1. 請開啟 [IAM 主控台的政策中樞](#)。
2. 在搜尋方塊中，輸入政策名稱。
3. 在搜尋方塊下的清單中找到您要更新的政策，然後選擇政策名稱。
4. 針對 Policy summary (政策摘要)，如果尚未顯示該面板，請選擇 JSON 標籤。
5. 如果想要編輯政策，請選擇 Edit policy (編輯政策)。
6. 如果尚未顯示該面板，請在編輯器中選擇 JSON 標籤。
7. 在政策文件中，尋找包含 `iot:CreateProvisioningClaim` 動作的政策陳述式。

如果政策文件不包含有 `iot:CreateProvisioningClaim` 動作的政策陳述式，請複製下列陳述式程式碼片段，並將其做為其他項目貼到政策文件中的 Statement 陣列。

**Note**

這個程式碼片段必須放在 Statement 陣列中的結尾 ] 字元之前。您可能需要在此程式碼片段之前或之後加上逗號，以更正任何語法錯誤。

```
{
  "Effect": "Allow",
  "Action": [
    "iot:CreateProvisioningClaim"
  ],
  "Resource": [
    "--PUT YOUR NEW TEMPLATE ARN HERE--"
  ]
}
```

8. 切換至您選擇 [修改使用者角色權限] 的 AWS IoT 主控台頁面。
9. 尋找 Resource ARN (資源 ARN) 範本，並選擇 Copy (複製)。
10. 切換回 IAM 主控台。
11. 將複製的 Amazon Resource Name (ARN) 貼到範本 ARN 的頂端 Statement 陣列成為第一個項目。

如果這是陣列中唯一的 ARN，請移除貼上值的結尾逗號。

12. 檢閱更新後的政策陳述式，並更正編輯器指出的任何錯誤。
13. 若要儲存更新的政策文件，請選擇 Review policy (檢閱政策)。
14. 檢閱政策然後選擇 Save changes (儲存變更)。
15. 返回主 AWS IoT 控台。

## 裝置佈建 MQTT API

車隊佈建服務支援下列 MQTT API 作業：

- [the section called "CreateCertificateFromCsr"](#)
- [the section called "CreateKeysAndCertificate"](#)
- [the section called "RegisterThing"](#)

此 API 支援簡明二進位物件表示法 (CBOR) 格式和 JavaScript 物件符號 (JSON) 的回應緩衝區，視主題的有#####而定。為了清楚起見，本節中的響應和請求示例以 JSON 格式顯示。

#####	回應格式資料類型
cbor	Concise Binary Object Representation (CBOR)
json	JavaScript 对象标记法

### Important

發佈請求訊息主題之前，請先訂閱回應主題以接收回應。此 API 使用的訊息使用 MQTT 的發佈/訂閱通訊協定，以提供請求和回應互動。

如果您在發佈請求之前未訂閱回應主題，您可能不會收到該要求的結果。

## CreateCertificateFromCsr

從憑證簽署要求 (CSR) 建立憑證。AWS IoT 提供由 Amazon 根憑證授權單位 (CA) 簽署的用戶端憑證。新憑證的狀態為 PENDING\_ACTIVATION。當您呼叫 RegisterThing 以使用此憑證佈建實物時，憑證狀態會變更為 INACTIVE 或 ACTIVE，如範本中所述。

如需有關使用憑證授權單位憑證和憑證簽署請求建立用戶端憑證的詳細資訊，請參閱 [使用您的憑證授權機構憑證建立用戶端憑證](#)。

### Note

為了安全起見，由 [CreateCertificateFromCsr](#) 傳回的 certificateOwnershipToken 會在一小時後過期。必須在 certificateOwnershipToken 過期之前呼叫 [RegisterThing](#)。如果由建立的憑證在權杖到期時 [CreateCertificateFromCsr](#) 尚未啟動並附加至原則或物件，則會刪除憑證。如果字符過期，裝置可以呼叫 [CreateCertificateFromCsr](#) 來產生新憑證。

## CreateCertificateFromCsr 請求

發佈包含 \$aws/certificates/create-from-csr/*payload-format* 主題的訊息。

## payload-format

訊息承載格式為 cbor 或 json。

### CreateCertificateFromCsr 要求有效載

```
{
  "certificateSigningRequest": "string"
}
```

### certificateSigningRequest

CSR，採用 PEM 格式。

### CreateCertificateFromCsr 回應

訂閱 `$aws/certificates/create-from-csr/payload-format/accepted`。

## payload-format

訊息承載格式為 cbor 或 json。

### CreateCertificateFromCsr 回應有效載

```
{
  "certificateOwnershipToken": "string",
  "certificateId": "string",
  "certificatePem": "string"
}
```

### certificateOwnershipToken

在佈建期間證明憑證擁有權的字符。

### certificateId

憑證的 ID。僅需 `certificateId` 即可進行憑證管理操作。

### certificatePem

憑證資料 (PEM 格式)。

## CreateCertificateFromCsr 錯誤

若要接收錯誤回應，請訂閱 `$aws/certificates/create-from-csr/payload-format/rejected`。

payload-format

訊息承載格式為 cbor 或 json。

## CreateCertificateFromCsr 錯誤負載

```
{
  "statusCode": int,
  "errorCode": "string",
  "errorMessage": "string"
}
```

statusCode

狀態碼。

errorCode

錯誤代碼。

errorMessage

錯誤訊息。

## CreateKeysAndCertificate

建立新金鑰和憑證。AWS IoT 提供由 Amazon 根憑證授權單位 (CA) 簽署的用戶端憑證。新憑證的狀態為 PENDING\_ACTIVATION。當您呼叫 RegisterThing 以使用此憑證佈建實物時，憑證狀態會變更為 INACTIVE 或 ACTIVE，如範本中所述。

### Note

為了安全起見，由 [CreateKeysAndCertificate](#) 傳回的 certificateOwnershipToken 會在一小時後過期。必須在 certificateOwnershipToken 過期之前呼叫 [RegisterThing](#)。如果由建立的憑證在權杖到期時 [CreateKeysAndCertificate](#) 尚

未啟動並附加至原則或物件，則會刪除憑證。如果字符過期，裝置可以呼叫 [CreateKeysAndCertificate](#) 來產生新憑證。

## CreateKeysAndCertificate請求

在 `$aws/certificates/create/payload-format` 上發佈訊息，其中包含空的訊息承載。

`payload-format`

訊息承載格式為 `cbor` 或 `json`。

## CreateKeysAndCertificate回應

訂閱 `$aws/certificates/create/payload-format/accepted`。

`payload-format`

訊息承載格式為 `cbor` 或 `json`。

## CreateKeysAndCertificate回應

```
{
  "certificateId": "string",
  "certificatePem": "string",
  "privateKey": "string",
  "certificateOwnershipToken": "string"
}
```

`certificateId`

憑證 ID。

`certificatePem`

憑證資料 (PEM 格式)。

`privateKey`

私有金鑰。

## certificateOwnershipToken

在佈建期間證明憑證擁有權的字符。

## CreateKeysAndCertificate 錯誤

若要接收錯誤回應，請訂閱 `$aws/certificates/create/payload-format/rejected`。

### payload-format

訊息承載格式為 cbor 或 json。

## CreateKeysAndCertificate錯誤負載

```
{
  "statusCode": int,
  "errorCode": "string",
  "errorMessage": "string"
}
```

### statusCode

狀態碼。

### errorCode

錯誤代碼。

### errorMessage

錯誤訊息。

## RegisterThing

使用預先定義的範本來佈建實物。

## RegisterThing 請求

在 `$aws/provisioning-templates/templateName/provision/payload-format` 上發佈訊息。

## payload-format

訊息承載格式為 cbor 或 json。

## templateName

佈建範本名稱。

## RegisterThing 要求有效載

```
{
  "certificateOwnershipToken": "string",
  "parameters": {
    "string": "string",
    ...
  }
}
```

## certificateOwnershipToken

證明憑證擁有權的權杖。AWS IoT 當您透過 MQTT 建立憑證時，會產生權杖。

## parameters

選用。來自[預先佈建掛接](#)所使用之裝置的鍵值組，以評估註冊請求。

## RegisterThing 回應

訂閱 `$aws/provisioning-templates/templateName/provision/payload-format/accepted`。

## payload-format

訊息承載格式為 cbor 或 json。

## templateName

佈建範本名稱。

## RegisterThing 回應有效載

```
{
```



```
"deviceConfiguration": {
  "string": "string",
  ...
},
"thingName": "string"
}
```

deviceConfiguration

範本中定義的裝置組態。

thingName

佈建期間建立的 IoT 物件名稱。

## RegisterThing 錯誤回應

若要接收錯誤回應，請訂閱 `$aws/provisioning-templates/templateName/provision/payload-format/rejected`。

payload-format

訊息承載格式為 cbor 或 json。

templateName

佈建範本名稱。

## RegisterThing 錯誤回應有效載

```
{
  "statusCode": int,
  "errorCode": "string",
  "errorMessage": "string"
}
```

statusCode

狀態碼。

errorCode

錯誤代碼。

## errorMessage

錯誤訊息。

## 機群索引

您可以使用機群索引，從以下來源索引、搜尋和彙總裝置的資料：[AWS IoT 登錄檔](#)、[AWS IoT Device Shadow](#)、[AWS IoT 連線](#)、[AWS IoT 裝置管理軟體套件目錄](#)和 [AWS IoT Device Defender](#) 違規事件。您可以查詢一組裝置，並根據裝置屬性的不同組合 (包括狀態、連線和裝置違規) 彙總裝置記錄上的統計資料。您可以利用機群索引來組織和調查裝置機群，以及解決相關問題。

機群索引提供下列功能。

## 管理索引更新

您可以設定機群索引，為物件群組、物件登錄檔、裝置影子、裝置連線和裝置違規的更新編制索引。啟動機群索引時，AWS IoT 會為物件或物件群組建立索引。AWS\_Things 會為所有物件建立索引。AWS\_ThingGroups 是包含所有物件群組的索引。啟用機群索引後，您可以在索引中執行查詢。例如，您可以找到所有手持設備且電池壽命超過 70% 的設備。AWS IoT 使用您的最新資料持續更新索引。如需詳細資訊，請參閱[管理機群索引](#)。

## 跨資料來源搜尋

您可以建立一個基於[查詢語言](#)的查詢字串，然後將其用來跨資料來源搜尋。您還需要在叢集索引設定中配置資料來源，以便索引組態包含您要從中搜尋的資料來源。查詢字串描述了您想要尋找的物件。您可以使用 AWS 受管欄位、自訂欄位以及索引資料來源中的任何屬性來建立查詢。如需支援機群索引的資料來源詳細資訊，請參閱[管理物件索引](#)。

## 查詢彙總資料

您可以在裝置上搜尋彙總資料並傳回統計資料、百分位數、基數，或物件清單，其中包含有關特定欄位的搜尋查詢。您可以對 AWS 受管欄位或在叢集索引設定中設定為自訂欄位的任何屬性執行彙總。如需彙總查詢的詳細資訊，請參閱[查詢彙總資料](#)。

## 使用機群指標監控彙總資料和建立警示

您可以使用叢集指標 CloudWatch 自動傳送彙總資料、分析趨勢，以及建立警示，以根據預先定義的閾值監控叢集的彙總狀態。如需機群指標的詳細資訊，請參閱[機群指標](#)。

# 管理機群索引

機群索引可以管理兩種類型的索引：物件索引和物件群組索引。

## 物件索引

為所有物件建立的索引為 `AWS_Things`。物件索引支援以下資料來源：[AWS IoT 登錄檔](#) 資料、[AWS IoT Device Shadow](#) 資料、[AWS IoT 連線](#) 資料以及 [AWS IoT Device Defender](#) 違規資料。透過將這些資料來源新增至機群索引組態，您可以搜尋物件、查詢彙總資料，以及根據搜尋查詢建立動態物件群組和機群指標。

註冊表-AWS IoT 提供一個註冊表，幫助您管理事情。您可以將登錄資料新增至機群索引組態，根據物件名稱、描述和其他登錄屬性來搜尋裝置。如需登錄檔的詳細資訊，請參閱[如何使用登錄檔管理物件](#)。

影子：[AWS IoT Device Shadow 服務](#) 提供影子，協助您儲存裝置狀態資料。物件索引同時支持傳統的未命名影子和命名影子。若要索引已命名影子，請啟用已命名影子的設定，然後在物件索引組態中指定影子名稱。依預設，每個最多可以新增 10 個陰影名稱 AWS 帳戶。若要了解如何增加影子名稱的數量限制，請參閱《AWS 一般參考》中的 [AWS IoT Device Management 配額](#)。

若要新增要索引的已命名影子：

- 如果使用的是 [AWS IoT 主控台](#)，請開啟 Thing indexing (物件索引)，選擇 Add named shadows (新增已命名影子)，再透過 Named shadow selection (已命名影子選擇) 新增影子名稱。
- 如果使用 AWS Command Line Interface (AWS CLI)，請設定 `namedShadowIndexingMode` 為 `ON`，並在中指定陰影名稱 [IndexingFilter](#)。若要查看 CLI 命令範例，請參閱[管理物件索引](#)。

### Important

2022 年 7 月 20 日是 AWS IoT 裝置管理叢集索引的正式發行 (GA) 版本，與 AWS IoT Core 命名陰影整合並 AWS IoT Device Defender 偵測違規。使用此 GA 版本，您可以透過指定影子名稱來索引特定的已命名影子。如果您在 2021 年 11 月 30 日至 2022 年 7 月 19 日期間，即在此功能的公開預覽期間新增了要編製索引的已命名影子，我們建議您重新設定機群索引設定，並選擇特定的影子名稱，以降低索引成本並最佳化效能。

如需影子的詳細資訊，請參閱 [AWS IoT Device Shadow 服務](#)。

連線 - 裝置連線資料可協助您識別裝置的連線狀態。此連線資料是由[生命週期事件](#)驅動。當用戶端連線或中斷連線時，會將含有訊息的生命週期事件 AWS IoT 發佈至 MQTT 主題。連線或中斷連線訊息可以是提供連線狀態詳細資訊的 JSON 元素清單。如需裝置連線的詳細資訊，請參閱[生命週期事件](#)。

Device Defender 違規-AWS IoT Device Defender 違規資料有助於針對您在安全性設定檔中定義的正常行為，識別異常裝置行為。安全性設定檔包含一組預期裝置行為。每個行為都會使用指定為裝置正常行為的指標。如需有關裝置防禦者違規的詳細資訊，請參閱[AWS IoT Device Defender 偵測](#)。

如需詳細資訊，請參閱[管理物件索引](#)。

## 物件群組索引

AWS\_ThingGroups 是包含所有物件群組和帳單群組的索引。您可以使用此索引，根據群組名稱、描述、屬性和所有父系群組名稱搜尋群組。

如需詳細資訊，請參閱[管理物件群組索引](#)。

## 受管欄位

受管理欄位包含與物件、物件群組、裝置陰影、裝置連線和 Device Defender 違規相關聯的資料。AWS IoT 定義託管字段中的數據類型。您可以在建立 AWS IoT 物件時指定每個受管理欄位的值。例如，物件名稱、物件群組和物件描述都是受管欄位。機群索引會根據您指定的索引模式編製受管欄位的索引。無法在 customFields 中變更或顯示受管欄位。如需詳細資訊，請參閱[自訂欄位](#)。

下文列出物件索引的受管欄位：

- 登錄檔的受管欄位

```
"managedFields" : [
  {name:thingId, type:String},
  {name:thingName, type:String},
  {name:registry.version, type:Number},
  {name:registry.thingTypeName, type:String},
  {name:registry.thingGroupNames, type:String},
]
```

- 傳統未命名影子的受管欄位

```
"managedFields" : [
  {name:shadow.version, type:Number},
]
```

```
{name:shadow.hasDelta, type:Boolean}
]
```

- 命名影子的受管欄位

```
"managedFields" : [
  {name:shadow.name.shadowName.version, type:Number},
  {name:shadow.name.shadowName.hasDelta, type:Boolean}
]
```

- 物件連線的受管欄位

```
"managedFields" : [
  {name:connectivity.timestamp, type:Number},
  {name:connectivity.version, type:Number},
  {name:connectivity.connected, type:Boolean},
  {name:connectivity.disconnectReason, type:String}
]
```

- Device Defender 的受管欄位

```
"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}
]
```

- 物件群組的受管欄位

```
"managedFields" : [
  {name:description, type:String},
  {name:parentGroupNames, type:String},
  {name:thingGroupId, type:String},
  {name:thingGroupName, type:String},
  {name:version, type:Number},
]
```

下表列出無法搜尋的受管欄位。

資料來源	無法搜尋的受管欄位
登錄檔	registry.version
未命名的影子	shadow.version
已命名的影子	shadow.name.*.version
Device Defender	deviceDefender.version
物件群組	version

## 自訂欄位

您可以建立自訂欄位來編製索引，可以彙總物件屬性、Device Shadow 資料和 Device Defender 違規資料。customFields 屬性是欄位名稱和資料類型配對的清單。您可以執行基於資料類型的彙總查詢。您選擇的索引模式會影響可以在 customFields 中指定的欄位。例如，如果您指定 REGISTRY 索引模式，則無法從物件影子中指定自訂欄位。您可以使用 [update-indexing-configuration](#) CLI 命令來建立或更新自訂欄位 (請參閱[更新索引組態範例](#)的範例命令)。

- 自訂欄位名稱

物件和物件群組屬性的自訂欄位名稱以 attributes. 開頭，後接屬性名稱。如果開啟未命名的影子索引，則物件可使用以 shadow.desired 或者 shadow.reported 開頭的自訂欄位名稱，後接未命名的影子資料值名稱。如果開啟已命名的影子索引，則物件可使用以 shadow.name.\*.desired. 或者 shadow.name.\*.reported. 開頭的自訂欄位名稱，後接已命名的影子資料值。如果開啟 Device Defender 違規索引，則物件可使用以 deviceDefender. 開頭的自訂欄位名稱，後接 Device Defender 的違規資料值。

字首後面接的屬性或資料值名稱只能使用英數字元、- (連字號) 和 \_ (底線) 字元。不可有任何空格。

如果組態中的自訂欄位與正要編製索引的值之間出現類型不一致，機群索引會忽略彙總查詢的不一致值。CloudWatch 疑難排解彙總查詢問題時，記錄檔很有用。如需詳細資訊，請參閱 [機群索引服務的疑難排解彙總查詢](#)。

- 自訂欄位類型

自訂欄位類型擁有下列支援的值：Number、String，以及 Boolean。

## 管理物件索引

為所有物件建立的索引為 `AWS_Things`。您可以控制要從下列資料來源編製索引的內容：[AWS IoT 登錄檔資料](#)、[AWS IoT Device Shadow 資料](#)，[AWS IoT 連線資料](#)，以及 [AWS IoT Device Defender 違規資料](#)。

在本主題中：

- [啟用物件索引](#)
- [描述物件索引](#)
- [查詢物件索引](#)
- [法規與限制](#)
- [授權](#)

### 啟用物件索引

您可以使用[更新索引設定 CLI 命令](#)或[UpdateIndexing 設定 API 作業](#)來建立索引並控制其組態。您可以透過使用 `--thing-indexing-configuration` (`thingIndexingConfiguration`) 參數，來控制要編製索引的資料類型 (例如，登錄檔資料、影子整理、裝置連線資料以及 Device Defender 違規資料)。

`--thing-indexing-configuration` 參數採用具有以下結構的字符：

```
{
  "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" ],
  "geoLocations": [
    {
      "name": "String",
      "order": "LonLat|LatLon"
    }
  ]
}
}
}

```

## 物件索引模式

您可以在索引組態中指定不同的物件索引模式，具體取決於您要從中索引和搜尋裝置的資料來源：

- `thingIndexingMode`：控制登錄或陰影是否已編製索引。如果設定 `thingIndexingMode` 為 OFF，則會停用物件索引。
- `thingConnectivityIndexingMode`：指定物件連接資料是否已編製索引。
- `deviceDefenderIndexingMode`：指定是否要索引裝置防禦者違規資料。
- `namedShadowIndexingMode`：指定具名的陰影資料是否要編製索引。若要選取要新增至機群索引組態的已命名影子，請將 `namedShadowIndexingMode` 設為 ON，再在 [filter](#) 中指定已命名影子名稱。

下表顯示每個索引模式的有效值，以及針對每個值編製索引的資料來源。

屬性	有效值	登錄檔	影子	連線能力	DD 違規行為	已命名影子
thingIndexingMode	OFF					
	REGISTRY	✓				

屬性	有效值	登錄檔	影子	連線能力	DD 違規行為	已命名影子
	REGISTRY_ AND_SHADOW	✓	✓			
thingConnectivityIndexingMode	未指定。					
	OFF					
	STATUS			✓		
deviceDefenderIndexingMode	未指定。					
	OFF					
	VIOLATIONS				✓	
namedShadowIndexingMode	未指定。					
	OFF					
	ON					✓

## 受管欄位和自訂欄位

### 受管欄位

受管理欄位包含與物件、物件群組、裝置陰影、裝置連線和 Device Defender 違規相關聯的資料。AWS IoT 定義託管字段中的數據類型。請在建立 AWS IoT 物件時指定每個受管欄位的值。例如，物件名稱、物件群組和物件描述都是受管欄位。機群索引會根據您指定的索引模式編製受管欄位的索引。無法在 `customFields` 中變更或顯示受管欄位。

### 自訂欄位

您可以建立自訂欄位來彙總屬性、Device Shadow 資料和 Device Defender 違規資料，以便為這些資料編製索引。`customFields` 屬性是欄位名稱和資料類型配對的清單。您可以執行基於資料類型的彙總查詢。您選擇的索引模式會影響可以在 `customFields` 中指定的欄位。例如，如果您指定 REGISTRY 索引模式，則無法從物件影子中指定自訂欄位。您可以使用 [update-indexing-configuration](#)

CLI 命令來建立或更新自訂欄位 (請參閱[更新索引組態範例](#)的範例命令)。如需詳細資訊，請參閱[自訂欄位](#)。

## 索引篩選

索引篩選器可為已命名陰影和地理位置資料提供其他選取項。

### namedShadowNames

若要將命名陰影新增至叢集索引組態，請將其設定 `namedShadowIndexingMode` 為 `ON` 並在 `namedShadowNames` 篩選器中指定已命名的陰影名稱。

#### 範例

```
"filter": {
  "namedShadowNames": [ "namedShadow1", "namedShadow2" ]
}
```

### geoLocations

若要將地理位置資料新增至叢集索引設定：

- 如果您的地理位置數據存儲在傳統 ( 未命名的 ) 陰影中，請將其設定 `thingIndexingMode` 置為 `REGISTRY_AND_SHADOW`，並在過濾器中指定您的地理位置數據。 `geoLocations`

下面的示例過濾器指定了經典 ( 未命名 ) 陰影中的 `GeoLocation` 對象：

```
"filter": {
  "geoLocations": [
    {
      "name": "shadow.reported.location",
      "order": "LonLat"
    }
  ]
}
```

- 如果您的地理位置資料儲存在命名陰影中，請設定 `namedShadowIndexingMode` 為「開啟」，在篩選器中新增陰影名稱，並在 `namedShadowNames` 篩選器中指定地理位置資料。 `geoLocations`

下面的示例過濾器指定在一個命名的 shadow ( `nameShadow1` ) 的地理位置對象：

```

"filter": {
  "namedShadowNames": [ "namedShadow1" ],
  "geoLocations": [
    {
      "name": "shadow.name.namedShadow1.reported.location",
      "order": "LonLat"
    }
  ]
}

```

有關更多信息，請參閱 [IndexingFilter](#) 從 AWS IoTAPI 參考。

### 更新索引組態範例

若要更新索引組態，請使用 AWS IoT update-indexing-configuration CLI 指令。下列範例示範如何使用 update-indexing-configuration。

簡短語法：

```

aws iot update-indexing-configuration --thing-indexing-configuration \
'thingIndexingMode=REGISTRY_AND_SHADOW, deviceDefenderIndexingMode=VIOLATIONS,
namedShadowIndexingMode=ON,filter={namedShadowNames=[thing1shadow]},
thingConnectivityIndexingMode=STATUS,
customFields=[{name=attributes.version,type=Number},
{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 語法：

```

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} ] ] }'
```

此命令不會產生任何輸出。

若要檢查物件索引狀態，請執行 `describe-index` CLI 命令：

```
aws iot describe-index --index-name "AWS_Things"
```

`describe-index` 命令的輸出如下所示：

```
{  
  "indexName": "AWS_Things",  
  "indexStatus": "ACTIVE",  
  "schema": "MULTI_INDEXING_MODE"  
}
```

#### Note

機群索引可能需要一些時間才能更新機群索引。建議等到 `indexStatus` 顯示 `ACTIVE` (啟用) 之後才使用。結構描述欄位可以擁有的值，視已設定的資料來源而定。如需詳細資訊，請參閱[描述物件索引](#)。

若要獲取索引組態的詳細資訊，請執行 `get-indexing-configuration` CLI 命令：

```
aws iot get-indexing-configuration
```

`get-indexing-configuration` 命令的輸出如下所示：

```
{  
  "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": "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"
  }
}

```

若要更新自訂欄位，您可以執行 `update-indexing-configuration` 命令。範例如下：

```
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}]'
```

此命令已將 `shadow.desired.intensity` 新增至索引組態。

#### Note

更新索引組態中的自訂欄位會覆寫所有現有的自訂欄位。請務必在呼叫 `update-indexing-configuration` 時指定所有自訂欄位。

在重建索引之後，您可以在剛新增的欄位上使用彙總查詢，然後搜尋登錄檔資料、影子資料和物件連線狀態資料。

變更索引模式時，請使用新的索引模式確定所有自訂欄位都有效。例如，如果以自訂欄位名為 `shadow.desired.temperature` 的 `REGISTRY_AND_SHADOW` 模式開始，則必須先刪除 `shadow.desired.temperature` 自訂欄位，再將索引模式變更為 `REGISTRY`。如果索引組態包含未被索引模式編製索引的自訂欄位，更新就會失敗。

## 描述物件索引

以下命令說明如何使用 `describe-index` CLI 命令，擷取物件索引的目前狀態。

```
aws iot describe-index --index-name "AWS_Things"
```

命令的回應可能如下所示：

```
{  
  "indexName": "AWS_Things",  
  "indexStatus": "BUILDING",  
  "schema": "REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS"  
}
```

您第一次建立叢集索引時，會 AWS IoT 建立索引。 `indexStatus` 為 `BUILDING` 狀態時，便法在索引內進行查詢。物件索引的 `schema` 會指出系統會為何種類型的資料 (`REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS`) 建立索引。



若您變更索引的組態，將導致索引重新建立。在此程序期間的 `indexStatus` 為 `REBUILDING`。您可以在重建物件索引時，對物件索引內的資料執行查詢。例如，若您將索引組態從 `REGISTRY` 變更為 `REGISTRY_AND_SHADOW`，重建索引時，您可以查詢登錄檔資料 (包括最新的更新)。不過，您無法查詢影子資料，除非重建完成。建立或重建索引所需時間，取決於資料量。

結構描述欄位可以顯示不同的值，視已設定的資料來源而定。下表顯示了不同的結構描述值和相應描述：

結構描述	描述
OFF	未設定任何資料來源或為其編制索引。
REGISTRY	登錄檔資料會編製索引。
REGISTRY_AND_SHADOW	登錄檔資料和未命名 (傳統) 的影子資料都會編製索引。
REGISTRY_AND_CONNECTIVITY	登錄檔資料和連線資料都會編製索引。
REGISTRY_AND_SHADOW_AND_CONNECTIVITY_STATUS	登錄檔資料、未命名 (傳統) 影子資料和連線資料都會編製索引。
MULTI_INDEXING_MODE	除了登錄檔、未命名 (傳統) 影子或連線資料之外，還會編製已命名影子或 Device Defender 違規資料的索引。

## 查詢物件索引

使用 `search-index` CLI 命令來查詢索引中的資料。

```
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":"AQFCuvk7zZ3D9p0YMbFCeHbdZ+h=G"
}
```

在 JSON 回應中，"connectivity" (透過 thingConnectivityIndexingMode=STATUS 設定啟用) 會提供布林值、時間戳記以及 disconnectReason，以指出裝置是否已連接到 AWS IoT Core。因為 CONNECTION\_LOST，裝置 "mything1" 在 POSIX 時間 1556649874716 中斷連線 (false)。如需中斷連線原因的詳細資訊，請參閱[生命週期事件](#)。

```
"connectivity": {
    "connected":false,
    "timestamp":1556649874716,
    "disconnectReason": "CONNECTION_LOST"
}
```

裝置 "mything2" 在 POSIX 時間 1556649855046 連線 (true)：

```
"connectivity": {
    "connected":true,
    "timestamp":1556649855046
}
```

時間戳記是以從 epoch 算起的毫秒提供，因此 1556649855046 代表 2019 年 4 月 30 日星期二下午 6:44:15.046 (UTC)。

### ⚠ Important

如果裝置已中斷連線大約一小時，則可能會缺少連線狀態的 "timestamp" 值和 "disconnectReason" 值。

## 法規與限制

這些是 AWS\_Things 的限制。

### 類型複雜的影子欄位

只有當欄位的值為簡單類型，像是不包含陣列的 JSON 物件，或完全由簡單類型組成的陣列時，才會建立該影子欄位的索引。簡單類型是指字串、數值或下列其中一個常值：true 或 false。以下列影子狀態為例，欄位 "palette" 的值不會編製索引，因為它是包含複雜類型項目的陣列。而因為該陣列中的每個值皆為字串，所以 "colors" 欄位的值會建立索引。

```
{
  "state": {
    "reported": {
      "switched": "ON",
      "colors": [ "RED", "GREEN", "BLUE" ],
      "palette": [
        {
          "name": "RED",
          "intensity": 124
        },
        {
          "name": "GREEN",
          "intensity": 68
        },
        {
          "name": "BLUE",
          "intensity": 201
        }
      ]
    }
  }
}
```

```
}
```

## 巢狀影子欄位名稱

巢狀影子欄位的名稱會儲存為以句號(.) 分隔的字串。例如，假設影子文件為：

```
{
  "state": {
    "desired": {
      "one": {
        "two": {
          "three": "v2"
        }
      }
    }
  }
}
```

three 欄位的名稱會儲存為 desired.one.two.three。如果您也有影子文件，會如此儲存：

```
{
  "state": {
    "desired": {
      "one.two.three": "v2"
    }
  }
}
```

同時符合 shadow.desired.one.two.three:v2 的查詢。最佳實務是不要在影子欄位名稱中使用句點。

## 影子中繼資料

只有當建立影子的 "state" 區段中之對應欄位的索引時，才會建立影子中繼資料區段中之欄位的索引。(在前面的範例中，也不會建立影子中繼資料區段中 "palette" 欄位的索引。)

## 未註冊的裝置

機群索引會為裝置的連線狀態建立索引，其連線 clientId 與[登錄](#)中已註冊物件的 thingName 相同。

## 未登錄檔的影子

如果您使用 S [UpdateThingshadow](#) 使用尚未在 AWS IoT 帳戶中註冊的物件名稱建立陰影，則此陰影中的欄位不會建立索引。這適用於傳統的未命名影子和已命名影子。

## 數值

如果服務將任何登錄檔或影子資料辨識為數值，則會因其為數值而建立索引。您可以對數值運用包含範圍與比較運算子的查詢 (例如 "attribute.foo<5" 或 "shadow.reported.foo:[75 TO 80]")。若要識別為數字，資料的值必須是有效的文字類型 JSON 數字。數值可為介於範圍  $-2^{53} \dots 2^{53}-1$  之間的整數、使用選擇性指數表示法的雙精度浮點，或必須屬於僅包含這些值的陣列。

## Null 值

Null 值不編製索引。

## 最大值

彙總查詢的自訂欄位數量上限是 5。

彙總查詢所要求的百分位數上限是 100。

## 授權

您可以在 AWS IoT 政策動作中將物件索引指定為 Amazon 資源名稱 (ARN)，如下所示。

動作	資源
iot:SearchIndex	索引 ARN (例如 , arn:aws:iot: <i>your-aws-region</i> <i>your-aws-account</i> :index/AWS_Things )。
iot:DescribeIndex	索引 ARN (例如 , arn:aws:iot: <i>your-aws-region</i> :index/AWS_Things )。

### Note

如果您有查詢機群索引的許可，您就可以存取整個機群的物件資料。

## 管理物件群組索引

AWS\_ThingGroups 是包含所有物件群組和帳單群組的索引。您可以使用此索引，根據群組名稱、描述、屬性和所有父系群組名稱搜尋群組。

### 啟用物件群組索引

您可以使用 `thing-group-indexing-configuration` 設定 API 中的 [UpdateIndexing](#) 設定來建立索引 AWS\_ThingGroups 並控制其設定。您可以使用 [GetIndexing](#) 設定 API 擷取目前的索引設定。

若要更新物件群組索引的組態，請執行 `update-indexing-configuration` CLI 命令：

```
aws iot update-indexing-configuration --thing-group-indexing-configuration
thingGroupIndexingMode=ON
```

您也能以如下命令更新物件和物件群組索引的組態：

```
aws iot update-indexing-configuration --thing-indexing-configuration
thingIndexingMode=REGISTRY --thing-group-indexing-configuration
thingGroupIndexingMode=ON
```

以下是 `thingGroupIndexingMode` 的有效值。

#### OFF

不建立索引/刪除索引。

#### ON

建立或設定 AWS\_ThingGroups 索引。

若要擷取目前物件和物件群組索引組態，請執行 `get-indexing-configuration` CLI 命令：

```
aws iot get-indexing-configuration
```

命令的回應如下所示：

```
{
  "thingGroupIndexingConfiguration": {
    "thingGroupIndexingMode": "ON"
  }
}
```

```
}  
}
```

## 描述群組索引

若要擷取 describe-index 索引的目前狀態，請使用 AWS\_ThingGroups CLI 命令：

```
aws iot describe-index --index-name "AWS_ThingGroups"
```

命令的回應如下所示：

```
{  
  "indexStatus": "ACTIVE",  
  "indexName": "AWS_ThingGroups",  
  "schema": "THING_GROUPS"  
}
```

AWS IoT 第一次建立索引時建立索引。若 indexStatus 為 BUILDING，您無法在該索引內進行查詢。

## 查詢物件群組索引

若要查詢索引中的資料，請使用 search-index CLI 命令：

```
aws iot search-index --index-name "AWS_ThingGroups" --query-string  
"thingGroupName:mythinggroup*"
```

## 授權

您可以在 AWS IoT 原則動作中將物群組索引指定為資源 ARN，如下所示。

動作	資源
iot:SearchIndex	索引 ARN (例如，arn:aws:iot: <i>your-aws-region</i> :index/AWS_ThingGroups )。
iot:DescribeIndex	索引 ARN (例如，arn:aws:iot: <i>your-aws-region</i> :index/AWS_ThingGroups )。



## 查詢彙總資料

AWS IoT 提供四個 API (GetStatistics、GetCardinalityGetPercentiles、和GetBucketsAggregation) , 可讓您搜尋裝置叢集以取得彙總資料。

### Note

如需彙總 API 遺失或未預期值的問題，請閱讀[機群索引故障診斷指南](#)。

## GetStatistics

[GetStatistics](#) API 和 get-statistics CLI 命令會傳回指定彙總欄位的計數、平均值、總和、最小值、最大值、平方和、方差和標準差。

get-statistics CLI 命令會使用下列參數：

index-name

要搜尋的索引名稱。預設值為 AWS\_Things。

query-string

用於搜尋索引的查詢。您可 "\*" 以指定獲取 AWS 帳戶。

aggregationField

(選用) 要彙總的欄位。此欄位必須是您在呼叫 update-indexing-configuration 時定義的受管或自訂欄位。如果您未指定彙總欄位，則 registry.version 會用作彙總欄位。

query-version

要使用的查詢版本。預設值為 2017-09-30。

彙總欄位的類型可以影響傳回的統計資料。

## GetStatistics 與字符串值

如果您在字串欄位上進行彙總，則呼叫 GetStatistics 會傳回多少裝置具有符合查詢的屬性。例如：

```
aws iot get-statistics --aggregation-field 'attributes.stringAttribute'
```

```
--query-string '*'
```

此命令會傳回多少裝置包含稱為 `stringAttribute` 的屬性：

```
{
  "statistics": {
    "count": 3
  }
}
```

## GetStatistics 與布爾值

當您 `GetStatistics` 使用布爾聚合字段調用時：

- AVERAGE 是符合查詢的裝置百分比。
- 根據以下規則，MINIMUM 為 0 或 1：
  - 如果彙總欄位的所有值是 `false`，MINIMUM 為 0。
  - 如果彙總欄位的所有值是 `true`，MINIMUM 為 1。
  - 如果彙總欄位的值是 `false` 和 `true` 的混合，MINIMUM 為 0。
- 根據以下規則，MAXIMUM 為 0 或 1：
  - 如果彙總欄位的所有值是 `false`，MAXIMUM 為 0。
  - 如果彙總欄位的所有值是 `true`，MAXIMUM 為 1。
  - 如果彙總欄位的值是 `false` 和 `true` 的混合，則 MAXIMUM 為 1。
- SUM 是整數對等於布爾值的總和。
- COUNT 是符合查詢字串準則並包含有效彙總欄位值的物件計數。

## GetStatistics 具有數值

當您呼叫 `GetStatistics` 並指定類型為 `Number` 的彙總欄位時，`GetStatistics` 會傳回下列值：

`count`

符合查詢字串條件且包含有效彙總欄位值的物件計數。

`average`

符合查詢之數值的平均值。

## sum

符合查詢之數值的總和。

## minimum

符合查詢的最小數值。

## maximum

符合查詢的最大數值。

## 總和 OfSquares

符合查詢之數值的平方和。

## variance

符合查詢之數值的方差。一組值的方差是每個值與集合平均值之平方差的平均值。

## stdDeviation

符合查詢之數值的標準差。一組值的標準差是值如何分佈的度量。

以下範例展示如何使用數字自訂欄位呼叫 `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": 13472.222222222222,  
    "stdDeviation": 116.06990230986766  
  }  
}
```

對於數值彙總欄位，如果欄位值超過最大雙精度值，則統計資料為空白

## GetCardinality

[GetCardinality](#) API 和 `get-cardinality` CLI 命令會傳回符合查詢之唯一值的近似計數。例如，您可能想要尋找電池電量低於 50% 的裝置數量：

```
aws iot get-cardinality --index-name AWS_Things --query-string "batterylevel > 50" --aggregation-field "shadow.reported.batterylevel"
```

這個命令會傳回電池電量超過 50% 的物件數量：

```
{
  "cardinality": 100
}
```

`cardinality` 一律由 `get-cardinality` 傳回，即使沒有相符的欄位。例如：

```
aws iot get-cardinality --query-string "thingName:Non-existent*"
--aggregation-field "attributes.customField_STR"
```

```
{
  "cardinality": 0
}
```

`get-cardinality` CLI 命令會使用下列參數：

`index-name`

要搜尋的索引名稱。預設值為 `AWS_Things`。

`query-string`

用於搜尋索引的查詢。您可 "\*" 以指定獲取 AWS 帳戶。

`aggregationField`

要彙總的欄位。

`query-version`

要使用的查詢版本。預設值為 `2017-09-30`。

## GetPercentiles

[GetPercentiles](#) API 和 `get-percentiles` CLI 命令會將符合查詢的彙總值分組為百分位數群組。雖然預設的百分位數群組是：1、5、25、50、75、95、99，但是您可以在呼叫 `GetPercentiles` 時指定自己的百分位數群組。此函數會傳回所指定之每個百分位數群組 (預設百分位數群組) 的值。百分位數群組 "1" 包含的彙總欄位值，大約發生在符合查詢的百分之一值。百分位數群組 "5" 包含的彙總欄位值，大約發生在符合查詢的百分之五值，依此類推。結果是一個近似值，符合查詢的值越多，百分位數值就越精確。

以下範例顯示如何呼叫 `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
  }
]
}
```

以下命令顯示在沒有相符文件時從 `get-percentiles` 傳回的輸出。

```
aws iot get-percentiles --query-string "thingName:Non-existent*"
                        --aggregation-field "attributes.customField_NUM"
```

```
{
  "percentiles": []
}
```

`get-percentile` CLI 命令會使用下列參數：

`index-name`

要搜尋的索引名稱。預設值為 `AWS_Things`。

`query-string`

用於搜尋索引的查詢。您可 "\*" 以指定獲取 AWS 帳戶。

`aggregationField`

要彙總的欄位，其類型必須是 `Number`。

`query-version`

要使用的查詢版本。預設值為 `2017-09-30`。

## percents

(選用) 您可以使用此參數來指定自訂百分位數群組。

## GetBuckets聚合

[GetBuckets](#)彙總 API 和 `get-buckets-aggregation` CLI 命令會傳回值區清單，以及符合查詢字串條件的項目總數。

下列範例顯示如何呼叫 `get-buckets-aggregation` CLI 命令。

```
aws iot get-buckets-aggregation --query-string '*' --index-name AWS_Things --
aggregation-field 'shadow.reported.batterylevelpercent' --buckets-aggregation-type
'termsAggregation={maxBuckets=5}'
```

此命令會傳回下列：

```
{
  "totalCount": 20,
  "buckets": [
    {
      "keyValue": "100",
      "count": 12
    },
    {
      "keyValue": "90",
      "count": 5
    },
    {
      "keyValue": "75",
      "count": 3
    }
  ]
}
```

`get-buckets-aggregation` CLI 命令採用下列參數：

### index-name

要搜尋的索引名稱。預設值為 `AWS_Things`。

## query-string

用於搜尋索引的查詢。您可 "\*" 以指定獲取 AWS 帳戶。

## aggregation-field

要彙總的欄位。

## buckets-aggregation-type

回應形狀的基本控制和要執行的儲存貯體彙總類型。

## 授權

您可以在 AWS IoT 原則動作中將物群組索引指定為資源 ARN，如下所示。

動作	資源
iot:GetStatistics	索引 ARN (例如，arn:aws:iot: <i>your-aws-region</i> :index/AWS_Things 或 arn:aws:iot: <i>your-aws-region</i> :index/AWS_ThingGroups )。

## 查詢語法

可以在機群索引中使用查詢語法來指定查詢。

## 支援的功能

查詢語法支援以下功能：

- 詞彙和字詞
- 搜尋欄位
- 字首搜尋
- 範圍搜尋
- 布林運算子 AND、OR、NOT 及 -。連字號是用於從搜尋結果排除某些項目 (例如，thingName: (tv\* AND -plasma))。



- 分組
- 欄位分組
- 逸出特殊字元 (如同以 \)

## 不支援的功能

查詢語法不支援以下功能：

- 字首萬用字元搜尋 (例如 `"*xyz"`)，但若搜尋 `"**"`，所有物件都會相符
- 常規表達式
- 提升
- 排名
- 模糊搜尋
- 鄰近搜尋
- 排序
- 聚合
- 特殊字元：`、@、#、%、\、/、'、;、和、,。請注意，, 僅在地理查詢中支持。

## 備註

查詢語言的幾件注意事項：

- 預設運算子為 AND。對 `"thingName:abc thingType:xyz"` 的查詢等同於 `"thingName:abc AND thingType:xyz"`。
- 如果未指定欄位，請在所有登錄、Device Shadow 和裝置防禦者欄位中 AWS IoT 搜尋字詞。
- 所有欄位名稱皆會區分大小寫。
- 搜尋不區分大小寫。文字以 `Java Character.isWhitespace(int)` 定義的空白字元分隔。
- Device Shadow 資料 (未命名的影子和已命名的影子) 的索引包含已呈報、所需、差量和中繼資料部分。
- 無法搜尋裝置影子和登錄檔版本，但回應中會顯示這些資訊。
- 一條查詢的詞彙數量上限為 12。
- 僅在地理查詢中支援特殊字元，。

## 範例物件查詢

使用查詢語法指定查詢字串中的查詢。查詢會傳遞至 [SearchIndex](#) API。下表列出部分查詢字串範例。

查詢字串	結果
abc	在任何註冊表，影子 (經典未命名的影子和命名的影子) 或 Device Defender 違規欄位中的「abc」查詢。
thingName:myThingName	查詢名稱為「myThingName」的東西。
thingName:my*	可查詢名稱起始為 "my" 的物件。
thingName:ab?	可查詢具有「ab」加上一個額外字元的物件名稱 (例如，「aba」、「abb」、「abc」等)。
thingTypeName:aa	查詢與類型 "aa" 關聯的物件。
thingGroupNames:a	查詢父項物件群組名稱為「a」的物件。
thingGroupNames:a*	查詢父項物件群組名稱符合模式「a*」的物件。
attributes.myAttribute:75	可查詢具有 "myAttribute" 屬性名稱的物件，且其值為 75。
attributes.myAttribute:[75 TO 80]	可查詢具有 "myAttribute" 屬性名稱的物件，且其值介於數字範圍 (75–80，含) 內。
attributes.myAttribute:{75 TO 80}	可查詢具有 "myAttribute" 屬性名稱的物件，且其值介於數字範圍 (>75 且 <=80) 內。
attributes.serialNumber:["abcd" TO "abcf"]	可查詢具有 "serialNumber" 屬性名稱的物件，且其值介於一組字母和數字的字串範圍內。以此查詢而言，傳回的物件必須具有 "serialNumber" 屬性且其值為 "abcd"、"abce" 或 "abcf"。
attributes.myAttribute:i*t	可查詢具有 "myAttribute" 屬性名稱的物件，且其值以 "i" 為首、後方接任意數量的字元，最後則是 "t"。

查詢字串	結果
<code>attributes.attr1:abc AND attributes.attr2&lt;5 NOT attributes.attr3&gt;10</code>	使用布林值運算式，查詢結合詞彙的物件。以此查詢而言，傳回的物件必須具有「attr1」屬性名稱且其值為「abc」具有「attr2」屬性名稱且其值小於 5 以及具有「attr3」屬性名稱且其值不超過 10。
<code>shadow.hasDelta:true</code>	查詢具有增量元素的未命名影子物件。
<code>NOT attributes.model:legacy</code>	可查詢屬性名稱為 "model" 而不是 "legacy" 的物件。
<code>shadow.reported.stats.battery:{70 TO 100} (v2 OR v3) NOT attributes.model:legacy</code>	可查詢符合下列標準的物件： <ul style="list-style-type: none"> <li>物件影子 stats.battery 屬性值在 70 至 100 之間。</li> <li>物件名稱、類型名稱或屬性值出現文字 "v2" 或 "v3"。</li> <li>物件 model 屬性未設定為 "legacy"。</li> </ul>
<code>shadow.reported.myvalues:2</code>	查詢影子回報區段中之 myvalues 陣列包含值為 2 的物件。
<code>shadow.reported.location:* NOT shadow.desired.stats.battery:*</code>	可查詢符合下列標準的物件： <ul style="list-style-type: none"> <li>location 屬性存在於影子的 reported 區段中。</li> <li>stats.battery 屬性不存在於影子的 desired 區段中。</li> </ul>
<code>shadow.name.&lt;shadowName&gt;.hasDelta:true</code>	查詢具有給定名稱的影子和增量元素的物件。
<code>shadow.name.&lt;shadowName&gt;.desired.filament:*</code>	查詢具有給定名稱的影子和所需絲狀結構屬性的物件。
<code>shadow.name.&lt;shadowName&gt;.reported.location:*</code>	查詢具有給定名稱影子的事物以及已命名影子報告區段中有 location 屬性存在的物件。

查詢字串	結果
<code>connectivity.connected:true</code>	查詢所有已連線的裝置。
<code>connectivity.connected:false</code>	查詢所有中斷連線的裝置。
<code>connectivity.connected:true AND connectivity.timestamp : [1557651600000 TO 1557867600000]</code>	查詢連線時間戳記 $\geq 1557651600000$ 且 $\leq 1557867600000$ 的所有連網裝置。時間戳記是以從 Epoch 算起的毫秒提供。
<code>connectivity.connected:false AND connectivity.timestamp : [1557651600000 TO 1557867600000]</code>	查詢中斷連線時間戳記 $\geq 1557651600000$ 且 $\leq 1557867600000$ 的所有中斷連網裝置。時間戳記是以從 Epoch 算起的毫秒提供。
<code>connectivity.connected:true AND connectivity.timestamp &gt; 1557651600000</code>	查詢連線時間戳記 $> 1557651600000$ 的所有連網裝置。時間戳記是以從 Epoch 算起的毫秒提供。
<code>connectivity.connected:*</code>	查詢具連線資訊的所有裝置。
<code>connectivity.disconnectReason:*</code>	查詢具有連線 <code>disconnectReason</code> 的所有裝置。
<code>connectivity.disconnectReason:CLIENT_INITIATED_DISCONNECT</code>	查詢因 <code>CLIENT_INITIATED_DISCONNECT</code> 而中斷連線的所有裝置。
<code>deviceDefender.violationCount:[0 TO 100]</code>	查詢 Device Defender 違規計數值落在數字範圍 (0-100 , 含) 內的物件。
<code>deviceDefender.&lt;device-SecurityProfile&gt;.disconnectBehavior.inViolation:true</code>	查詢違反如安全性設定檔 <code>device-SecurityProfile</code> 所定義行為 <code>disconnectBehavior</code> 的物件。請注意 , <code>inViolation:false</code> 不是有效的查詢。

查詢字串	結果
<code>deviceDefender.&lt;device-SecurityProfile&gt;.disconnectBehavior.lastViolationValue.number&gt;2</code>	查詢違反行為的事件 (disconnectBehavior 如安全性設定檔裝置中定義SecurityProfile 的最後違規事件值大於 2)。
<code>deviceDefender.&lt;device-SecurityProfile&gt;.disconnectBehavior.lastViolationTime&gt;1634227200000</code>	查詢違反安全性設定檔裝置中disconnectBehavior 所定義之行為的事件，且在指定SecurityProfile 的紀元時間之後發生最後一個違規事件。
<code>shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km</code>	查詢距離 47.6204，-122.3491 坐標 15.5 公里的徑向距離內的東西。當您的位置資料儲存在具名陰影中時，此查詢字串適用於。
<code>shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km</code>	查詢距離 47.6204，-122.3491 坐標 15.5 公里的徑向距離內的東西。當您的位置資料儲存在傳統陰影中時，此查詢字串適用於。

## 範例物件群組查詢

查詢是以使用查詢語法的字串指定，並傳送至 [SearchIndex](#) API。下表列出部分查詢字串範例。

查詢字串	結果
<code>abc</code>	任何欄位的 "abc" 查詢。
<code>thingGroupName:myGroupThingName</code>	查詢名稱為「我的名GroupThing稱」的物件群組。
<code>thingGroupName:my*</code>	可查詢名稱以 "my" 開頭的物件群組。

查詢字串	結果
<code>thingGroupName:ab?</code>	可查詢具有「ab」加上一個額外字元的物件群組名稱 (例如：「aba」、「abb」、「abc」等)。
<code>attributes.myAttribute:75</code>	可查詢具有 "myAttribute" 屬性名稱的物件群組，且其值為 75。
<code>attributes.myAttribute:[75 TO 80]</code>	可查詢具有 "myAttribute" 屬性名稱的物件群組，且其值介於一組數字範圍 (75–80，含) 內。
<code>attributes.myAttribute:[75 TO 80]</code>	可查詢具有 "myAttribute" 屬性名稱的物件群組，且其值介於一組數字範圍 (>75 且 <=80) 內。
<code>attributes.myAttribute:["abcd" TO "abcf"]</code>	可查詢具有 "myAttribute" 屬性名稱的物件群組，且其值介於一組字母和數字的字串範圍內。以此查詢而言，傳回的物件群組必須具有 "serialNumber" 屬性且其值為 "abcd"、"abce" 或 "abcf"。
<code>attributes.myAttribute:i*t</code>	可查詢具有 "myAttribute" 屬性名稱的物件群組，且其值以 "i" 為首、後方接任意數量的字元，最後則是 "t"。
<code>attributes.attr1:abc AND attributes.attr2&lt;5 NOT attributes.attr3&gt;10</code>	使用布林值表達式，查詢結合詞彙的物件群組。以此查詢而言，傳回的物件群組必須具有「attr1」屬性名稱且其值為「abc」、具有「attr2」屬性名稱且其值小於 5 以及具有「attr3」屬性名稱且其值不超過 10。
<code>NOT attributes.myAttribute:cde</code>	可查詢具有 "myAttribute" 屬性名稱的物件群組，且不是 "cde"。
<code>parentGroupNames:(myParentThingGroupName)</code>	查詢父群組名稱符合「my ParentThingGroupName」的物件群組。
<code>parentGroupNames:(myParentThingGroupName OR myRootThingGroupName)</code>	查詢父群組名稱符合「my」或「my ParentThingGroupName RootThingGroupName」的物件群組。
<code>parentGroupNames:(myParentThingGroupNa*)</code>	查詢父群組名稱以「my ParentThingGroupNa」開頭的物件群組。

# 索引位置資料

您可以使用[AWS IoT 叢集索引來索引](#)裝置上次傳送的位置資料，並使用地理查詢搜尋裝置。此功能可解決裝置監控和管理使用案例，例如位置追蹤和鄰近搜尋。位置索引的運作方式與其他叢集索引功能類似，並具有可在[物件索引](#)中指定的其他組態。

常見使用案例包括：搜尋和彙總位於所需地理界限內的裝置、使用與裝置中繼資料相關的查詢字詞以及索引資料來源中的狀態取得位置特定見解、提供精細檢視，例如將結果篩選至特定地理區域，以減少叢集監控地圖內的轉譯延遲，並追蹤上次報告的裝置位置，以及識別超出所需邊界限制的裝置，並使用[叢集指標](#)產生警示。若要開始使用位置索引和地理查詢，請參閱[???](#)。

## 支援的資料格式

AWS IoT 叢集索引支援下列位置資料格式：

### 1. 座標參考系統的已知文字表示

位於[地理資訊之後的字串-座標參考系統格式的已知文字表示法](#)。一個例子可以是"POINT(long lat)"。

### 2. 表示坐標的字符串

格式為"latitude, longitude"或的字串"longitude, latitude"。如果使用"longitude, latitude"，您還必須order在中指定geoLocations。一個例子可以是"41.12, -71.34"。

### 3. 緯度 ( 緯度 ) ， lon ( 經度 ) 鍵的對象

此格式適用於經典陰影和命名陰影。支援的按鍵：latlatitude、lon、long、longitude。一個例子可以是{"lat": 41.12, "lon": -71.34}。

### 4. 表示坐標的數組

格式為[lat,lon]或的陣列[lon,lat]。如果您使用的格式[lon,lat]與[GeoJSON](#)中的座標相同 (適用於傳統陰影和命名陰影)，您還必須order在中geoLocations指定。

一個例子可以是：

```
{
  "location": {
    "coordinates": [
```

```
    **Longitude**,
    **Latitude**
  ],
  "type": "Point",
  "properties": {
    "country": "United States",
    "city": "New York",
    "postalCode": "*****",
    "horizontalAccuracy": 20,
    "horizontalConfidenceLevel": 0.67,
    "state": "New York",
    "timestamp": "2023-01-04T20:59:13.024Z"
  }
}
```

## 如何為位置資料建立索引

以下步驟說明如何更新位置資料的索引設定，以及如何使用地理查詢來搜尋裝置。

### 1. 瞭解您的位置資料的儲存位置

叢集索引目前支援索引儲存在傳統陰影或已命名陰影中的位置資料。

### 2. 使用支援的位置資料格式

請確定您的位置資料格式遵循其中一種[支援的資料格式](#)。

### 3. 更新索引配置

至少需要啟用物件 (登錄) 索引設定。您還必須在包含位置數據的傳統陰影或命名陰影上啟用索引。更新物件索引時，您應該在索引組態中包含您的位置資料。

### 4. 建立和執行地理查詢

根據您的使用案例，建立地理查詢並執行它們以搜尋裝置。[您撰寫的地理位置必須遵循查詢語法](#)。您可以在[???](#)中找到一些範例。

## 更新物件索引組態

若要為位置資料建立索引，您必須更新索引設定並包含您的位置資料。根據您的位置資料的儲存位置，請依照下列步驟更新索引設定：



## 儲存在傳統陰影中的位置資料

如果您的位置資料儲存在傳統陰影中，您必須將位置資料設定thingIndexingMode為，REGISTRY\_AND\_SHADOW並在中的geoLocations欄位(name和order) 中指定位置資料[filter](#)。

在下列物件索引組態範例中，您可以將位置資料路徑指定shadow.reported.coordinatesLonLat為name和order。

```
{
  "thingIndexingMode": "REGISTRY_AND_SHADOW",
  "filter": {
    "geoLocations": [
      {
        "name": "shadow.reported.coordinates",
        "order": "LonLat"
      }
    ]
  }
}
```

- thingIndexingMode

索引模式可控制登錄或陰影是否已建立索引。如果設定thingIndexingMode為OFF，則會停用物件索引。

若要為傳統陰影中儲存的位置資料建立索引，您必須thingIndexingMode將設定為REGISTRY\_AND\_SHADOW。如需詳細資訊，請參閱 [???](#)。

- filter

索引篩選器可為已命名陰影和地理位置資料提供其他選取項。如需詳細資訊，請參閱 [???](#)。

- geoLocations

您選取要編製索引的地理位置目標清單。用於索引的地理位置目標的預設最大數目為。1若要提高限制，請參閱[AWS IoT Device Management 配額](#)。

- name

地理位置目標欄位的名稱。的範例值name可以是陰影的位置資料路徑：shadow.reported.coordinates。

- order

地理位置目標欄位的順序。有效值：LatLon和LonLat。LatLon意味著經緯度。LonLat表示經度和緯度。此欄位為選用欄位。預設值為 LatLon。

### 儲存在具名陰影中的位置資料

如果您的位置資料儲存在具名陰影中，請將設定namedShadowIndexingMode為ON，將您的命名陰影名稱新增至中的namedShadowNames欄位 [filter](#)，並在中的geoLocations欄位中指定您的位置資料路徑[filter](#)。

在下列物件索引組態範例中，您可以將位置資料路徑指定shadow.name.namedShadow1.reported.coordinatesLonLat為name和order。

```
{
  "thingIndexingMode": "REGISTRY",
  "namedShadowIndexingMode": "ON",
  "filter": {
    "namedShadowNames": [
      "namedShadow1"
    ],
    "geoLocations": [
      {
        "name": "shadow.name.namedShadow1.reported.coordinates",
        "order": "LonLat"
      }
    ]
  }
}
```

- thingIndexingMode

索引模式可控制登錄或陰影是否已建立索引。如果設定thingIndexingMode為OFF，則會停用物件索引。

若要為儲存在具名陰影中的位置資料建立索引，您必thingIndexingMode須將設定為 REGISTRY (或REGISTRY\_AND\_SHADOW)。如需詳細資訊，請參閱 [???](#)。

- filter

索引篩選器可為已命名陰影和地理位置資料提供其他選取項。如需詳細資訊，請參閱 [???](#)。

- geoLocations

您選取要編製索引的地理位置目標清單。用於索引的地理位置目標的預設最大數目為 1。若要提高限制，請參閱 [AWS IoT Device Management 配額](#)。

- name

地理位置目標欄位的名稱。的範例值 name 可以是陰影的位置資料路徑：`shadow.name.namedShadow1.reported.coordinates`。

- order

地理位置目標欄位的順序。有效值：`LatLon` 和 `LonLat`。`LatLon` 意味著經緯度。`LonLat` 表示經度和緯度。此欄位為選用欄位。預設值為 `LatLon`。

## 地理查詢示例

完成位置資料的索引設定後，請執行地理查詢以搜尋裝置。您也可以將您的地理查詢與其他查詢字串結合在一起。如需詳細資訊，請參閱 [???](#) 及 [???](#)。

### 查詢範例 1

此範例假設位置資料儲存在具名陰影中 `gps-tracker`。此指令的輸出為距離中心點 15.5 公里之徑向距離內的裝置清單 (47.6204, -122.3491)。

```
aws iot search-index --query-string \  
"shadow.name.gps-tracker.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

### 查詢範例 2

此範例假設位置資料儲存在傳統陰影中。此指令的輸出為距離中心點 15.5 公里之徑向距離內的裝置清單 (47.6204, -122.3491)。

```
aws iot search-index --query-string \  
"shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

### 查詢範例 3

此範例假設位置資料儲存在傳統陰影中。該命令的輸出是未連接的設備列表，並且距離中心點 15.5 公里的徑向距離以外，具有坐標 ( 47.6204 , -122.3491 )。

```
aws iot search-index --query-string \  
"
```

```
"connectivity.connected:false AND (NOT  
shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km)"
```

## 入門教學課程

本教學課程示範如何使用[叢集索引編製位置資料](#)的索引。為了簡化起見，您可以建立表示裝置的物件，並將位置資料儲存在具名陰影中，更新物件索引組態以進行位置索引，並執行範例地理查詢以搜尋徑向邊界內的裝置。

此教學課程約需 15 分鐘方能完成。

在本主題中：

- [必要條件](#)
- [創建事物和陰影](#)
- [更新物件索引組態](#)
- [執行地理查詢](#)

### 必要條件

- 安裝最新版本的 [AWS CLI](#)。
- [熟悉位置索引和地理查詢、管理物件索引和查詢語法。](#)

### 創建事物和陰影

您可以創建一個代表您的設備的東西，並創建一個命名的陰影來存儲其位置數據（坐標 47.61564，-122.33584）。

1. 運行以下命令來創建代表您名為 Bike-1 的自行車的東西。如需如何使用建立物件的詳細資訊 AWS CLI，請參閱從參考[建立物件](#)。AWS CLI

```
aws iot create-thing --thing-name "Bike-1" \  
--attribute-payload '{"attributes": {"model":"OEM-2302-12", "battery":"35",  
"acqDate":"06/09/23"}}'
```

此命令的輸出如下所示：

```
{  
  "thingName": "Bike-1",
```

```
"thingArn": "arn:aws:iot:us-east-1:123456789012:thing/Bike-1",
"thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df"
}
```

2. 執行下列指令以建立具名陰影來儲存自行車 1 號的位置資料 (座標 47.61564, -122.33584)。如需有關如何使用建立命名陰影的詳細資訊 AWS CLI，請參閱參照中的[更新物件陰影](#)。AWS CLI

```
aws iot-data update-thing-shadow \
--thing-name Bike-1 \
--shadow-name Bike1-shadow \
--cli-binary-format raw-in-base64-out \
--payload '{"state":{"reported":{"coordinates":{"lat": 47.6153, "lon": -122.3333}}}}' \
"output.txt" \
```

此命令不會產生任何輸出。若要檢視您建立的已命名陰影，您可以執行[清單命名的陰影換物 CLI 命令](#)。

```
aws iot-data list-named-shadows-for-thing --thing-name Bike-1
```

此命令的輸出如下所示：

```
{
  "results": [
    "Bike1-shadow"
  ],
  "timestamp": 1699574309
}
```

## 更新物件索引組態

若要為位置資料建立索引，您必須更新物件索引組態以包含位置資料。由於您的位置資料儲存在本自學課程中的具名陰影中，因此請將其設定 `thingIndexingMode` 為 `REGISTRY` (在最低需求下)、設定 `namedShadowIndexingMode` 為 `ON`，然後將位置資料加入至規劃。在此範例中，您必須將已命名陰影的名稱和陰影的位置資料路徑新增至 `filter`。

1. 執行指令以更新位置索引的索引組態。

```
aws iot update-indexing-configuration --cli-input-json '{
```

```
"thingIndexingConfiguration": { "thingIndexingMode": "REGISTRY",
"thingConnectivityIndexingMode": "OFF",
"deviceDefenderIndexingMode": "OFF",
"namedShadowIndexingMode": "ON",
"filter": {
  "namedShadowNames": ["Bike1-shadow"],
  "geoLocations": [{
    "name": "shadow.name.Bike1-shadow.reported.coordinates"
  }]
},
"customFields": [
{ "name": "attributes.battery",
" type": "Number"}] } }'
```

命令不會產生任何輸出。您可能需要等待片刻，直到更新完成。若要檢查狀態，請執行[描述索引 CLI](#) 命令。如果您看到indexStatus顯示：ACTIVE，則表示您的物件索引更新已完成。

## 2. 執行命令以驗證您的索引組態。此為選擇性步驟。

```
aws iot get-indexing-configuration
```

輸出可能如下所示：

```
{
  "thingIndexingConfiguration": {
    "thingIndexingMode": "REGISTRY",
    "thingConnectivityIndexingMode": "OFF",
    "deviceDefenderIndexingMode": "OFF",
    "namedShadowIndexingMode": "ON",
    "managedFields": [
      {
        "name": "shadow.name.*.hasDelta",
        "type": "Boolean"
      },
      {
        "name": "registry.version",
        "type": "Number"
      },
      {
        "name": "registry.thingTypeName",
        "type": "String"
      }
    ]
  }
}
```

```
        "name": "registry.thingGroupNames",
        "type": "String"
    },
    {
        "name": "shadow.name.*.version",
        "type": "Number"
    },
    {
        "name": "thingName",
        "type": "String"
    },
    {
        "name": "thingId",
        "type": "String"
    }
],
"customFields": [
    {
        "name": "attributes.battery",
        "type": "Number"
    }
],
"filter": {
    "namedShadowNames": [
        "Bike1-shadow"
    ],
    "geoLocations": [
        {
            "name": "shadow.name.Bike1-shadow.reported.coordinates",
            "order": "LatLon"
        }
    ]
}
},
"thingGroupIndexingConfiguration": {
    "thingGroupIndexingMode": "OFF"
}
}
```

## 執行地理查詢

現在，您已更新物件索引組態以包含位置資料。嘗試創建一些地理查詢並運行它們以查看是否可以獲得所需的搜索結果。地理查詢必須遵循[查詢語法](#)。您可以在[???](#)中找到一些有用的地理查詢示例。

在以下範例指令中，您可以使用地理查詢`shadow.name.Bike1-shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km`來搜尋距中心點 15.5 公里之徑向距離內的裝置 (47.6204, -122.3491)。

```
aws iot search-index --query-string "shadow.name.Bike1-  
shadow.reported.coordinates:geo_distance,47.6204,-122.3491,15.5km"
```

因為您的設備位於坐標「緯度」：47.6153，「LON」：-122.3333，該設備落在距中心點 15.5 公里的距離內，因此您應該能夠在輸出中看到此設備（自行車 -1）。輸出可能如下所示：

```
{  
  "things": [  
    {  
      "thingName": "Bike-1",  
      "thingId": "df9cf01d-b0c8-48fe-a2e2-e16cff6b23df",  
      "attributes": {  
        "acqDate": "06/09/23",  
        "battery": "35",  
        "model": "OEM-2302-12"  
      },  
      "shadow": "{\"reported\":{\"coordinates\":{\"lat\":47.6153,\"lon  
\":-122.3333}},\"metadata\":{\"reported\":{\"coordinates\":{\"lat\":{\"timestamp  
\":1699572906},\"lon\":{\"timestamp\":1699572906}}}},\"hasDelta\":false,\"version\":1}"  
    }  
  ]  
}
```

如需更多詳細資訊，請參閱 [???](#)。

## 機群指標

叢集指標是[叢集索引](#)的一項功能，這是一項託管服務，可讓您在 AWS IoT 其中索引、搜尋和彙總裝置資料。您可以使用叢集指標來監控叢集裝置在[CloudWatch](#)一段時間內的彙總狀態，包括檢視叢集裝置的斷線速率或指定期間的平均電池電量變更。



使用叢集指標，您可以建立[彙總查詢](#)，其結果會持續發放到[CloudWatch](#)作為分析趨勢和建立警示的指標。您可以為監控任務指定以下彙總類型的彙總查詢：Statistics (統計數字)、Cardinality (基數) 以及 Percentile (百分位數)。您可以儲存所有彙總查詢來建立機群指標，以供日後重複使用。

## 入門教學課程

在此教學課程中，您要建立能夠監控感應器溫度的[機群指標](#)，以偵測潛在的異常情況。建立機群指標時，您可以定義[彙總查詢](#)，其可偵測溫度超過華氏 80 度的感應器數目。您可以指定每 60 秒執行一次查詢，並會發出查詢結果 CloudWatch，您可以在其中檢視具有潛在高溫風險的感測器數目，並設定警示。為完成此教學課程，您需要使用 [AWS CLI](#)。

於本教學課程中，您會了解如何：

- [設定](#)
- [建立機群指標](#)
- [檢視度量 CloudWatch](#)
- [清除資源](#)

此教學課程約需 15 分鐘方能完成。

### 必要條件

- 安裝 [AWS CLI](#) 的最新版本
- 熟悉[如何查詢彙總資料](#)
- 熟悉[使用 Amazon 指標 CloudWatch](#)

### 設定

若要使用機群指標，請啟用機群索引。若要針對具有指定資料來源和相關組態的物件或物件群組啟用機群索引，請遵循[管理物件索引](#)和[管理物件群組索引](#)中的指示。

### 設定

1. 執行下列命令來啟用機群索引，並指定要從中搜尋的資料來源。

```
aws iot update-indexing-configuration \  
--thing-indexing-configuration  
"thingIndexingMode=REGISTRY_AND_SHADOW,customFields=[{name=attributes.temperature,type=Num
```

```
{name=attributes.rackId,type=String},
{name=attributes.stateNormal,type=Boolean}],thingConnectivityIndexingMode=STATUS" \
```

上述的 CLI 命令範例可啟用機群索引，支援使用 `AWS_Things` 索引來搜尋登錄檔資料、影子資料和物件連線狀態。

組態變更可能需要幾分鐘時間才能完成。在建立機群指標之前，請驗證是否已啟用機群索引。

若要檢查機群索引是否已啟用，請執行下列 CLI 命令：

```
aws --region us-east-1 iot describe-index --index-name "AWS_Things"
```

如需詳細資訊，請參閱[啟用物件索引](#)。

## 2. 執行以下 Bash 指令碼來建立十個物件並提供相應描述。

```
# 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 true 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
```

這個指令碼建立了十個物件來表示十個感應器。如下表所述，每個物件都有 `temperature`、`rackId` 以及 `stateNormal` 屬性：

屬性	資料類型	描述
<code>temperature</code>	Number	以華氏為單位的溫度值
<code>rackId</code>	字串	包含感應器的伺服器機架 ID
<code>stateNormal</code>	Boolean	感應器溫度值是否正常

此指令碼的輸出包含十個 JSON 檔案。其中一個 JSON 檔案如下所示：

```
{
  "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"
}
```

如需詳細資訊，請參閱[建立物件](#)。

## 建立機群指標

若要建立機群指標

1. 執行下列命令建立名為 *high\_temp\_FM* 的機群指標。您可以建立車隊指標，以監控溫度超過華氏 80 度的感測器數量。CloudWatch

```
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
```

`--metric-name`

資料類型：字串。此 `--metric-name` 參數會指定機群指標名稱。在此範例中，您建立的是名為 `high_temp_FM` 的機群指標。

`--query-string`

資料類型：字串。此 `--query-string` 參數會指定查詢字串。在此範例中，查詢字串意味著查詢名稱以 `TempSensor` 及溫度高於華氏 80 度的所有項目。如需詳細資訊，請參閱[查詢語法](#)。

`--period`

資料類型：整數。此 `--period` 參數會指定彙總資料的擷取時間 (以秒為單位)。在此範例中，您指定要建立的機群指標每 60 秒擷取一次彙總資料。

`--aggregation-field`

資料類型：字串。此 `--aggregation-field` 參數會指定要評估的屬性。在此範例中，要評估的是溫度屬性。

`--aggregation-type`

此 `--aggregation-type` 參數指定要在機群指標中顯示的統計摘要。您可以為監控任務自訂以下彙總類型的彙總查詢屬性：Statistics (統計數字)、Cardinality (基數) 以及 Percentile (百分位數)。在此範例中，您可以指定彙總類型的計數和統計資料，以傳回具有符合查詢之屬性的裝置計數，換句話說，傳回名稱開TempSensor頭為且溫度高於華氏 80 度的裝置計數。如需詳細資訊，請參閱[查詢彙總資料](#)。

此命令的輸出結果如下所示：

```
{
  "metricArn": "arn:aws:iot:region:111122223333:fleetmetric/high_temp_FM",
  "metricName": "high_temp_FM"
}
```

#### Note

資料點可能需要一點時間才能在其中顯示 CloudWatch。

若要進一步了解如何建立機群指標，請參閱[管理機群指標](#)。

如果無法建立機群指標，請參閱[機群指標故障診斷](#)。

2. (選用) 執行下列命令來描述名為 `high_temp_FM` 的機群指標：

```
aws iot describe-fleet-metric --metric-name "high_temp_FM"
```

此命令的輸出結果如下所示：

```
{
```

```
"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": [
    "count"
  ],
  "name": "Statistics"
},
"indexName": "AWS_Things",
"creationDate": 1625249775.834,
"metricName": "high_temp_FM"
}
```

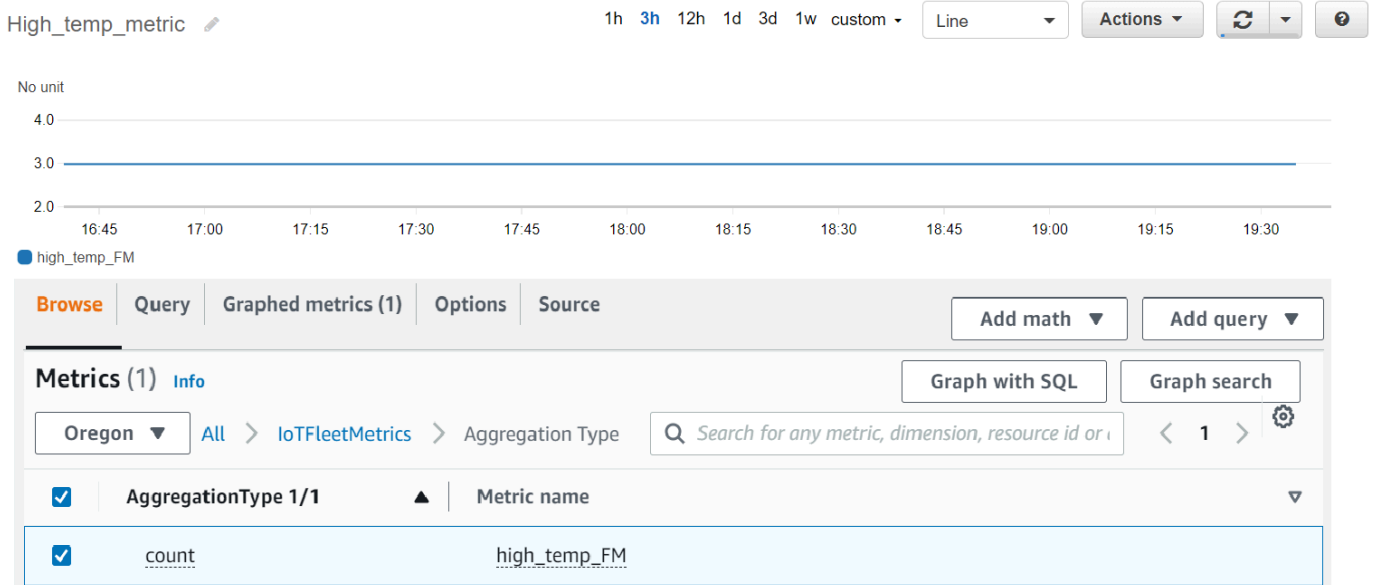
## 檢視叢集指標 CloudWatch

建立叢集測量結果後，您可以在中檢視測量結果資料 CloudWatch。在本教程中，您將看到顯示名稱開頭且溫度高於華氏 80 度的 TempSensor 傳感器數量的度量標準。

### 若要檢視資料點 CloudWatch

1. [請在以下位置開啟 CloudWatch 主控台。](https://console.aws.amazon.com/cloudwatch/) <https://console.aws.amazon.com/cloudwatch/>
2. 在左側面板的 CloudWatch 功能表上，選擇「度量」以展開子選單，然後選擇「所有量度」。此操作會開啟頁面，其中上半部分會顯示圖形，下半部分則包含四個標籤式區段。
3. 第一個索引標籤區段「所有指標」會列出您可以在群組中檢視的所有指標，請選擇 IoT FleetMetrics。這包含您的所有機群指標。
4. 在 All metrics (所有指標) 索引標籤的 Aggregation type (彙總類型) 區段，選擇 Aggregation type (彙總類型) 檢視建立的所有機群指標。
5. 選擇機群指標，在 Aggregation type (彙總類型) 區段的左側顯示圖表。您會看到指標名稱左側的值 *count*，這是您在本教學課程的[建立機群指標](#)區段中指定的彙總類型值。
6. 選擇位於 All metrics (所有指標) 索引標籤右側的名為 Graphed metrics (圖表化指標) 的第二個索引標籤，檢視從上一步中選擇的機群指標。

您應該可以看到如下所示的圖表，其中顯示溫度高於華氏 80 度的感應器數量：



### Note

中的「期間」屬性 CloudWatch 預設為 5 分鐘。這是顯示在中的數據點之間的時間間隔 CloudWatch。您可以根據需求變更 Period (期間) 設定。

## 7. (選用) 您可以設定指標警示。

1. 在左側面板的 CloudWatch 功能表上，選擇 [警報] 以展開子功能表，然後選擇 [所有鬧鐘]。
2. 在 Alarms (警示) 頁面中，選擇右上角的 Create alarm (建立警示)。請遵循主控台中的 Create alarm (建立警示) 指示，視需要建立警示。如需詳細資訊，請參閱[使用 Amazon CloudWatch 警示](#)。

若要進一步了解，請參閱[使用 Amazon CloudWatch 指標](#)。

如果在中看不到資料點 CloudWatch，請參閱[疑難排解叢集指標](#)。

## 清除

### 刪除機群指標

可以使用 delete-fleet-metric CLI 命令來刪除機群指標。

若要刪除名為 high\_temp\_FM 的機群指標，請執行下列命令。

```
aws iot delete-fleet-metric --metric-name "high_temp_FM"
```

## 清理物件

可以使用 delete-thing CLI 命令來刪除物件。

若要刪除您已建立的十個物件，請執行下列指令碼：

```
# 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
```

## 若要清理量度 CloudWatch

CloudWatch 不支援指標刪除。指標的過期時間因保留排程而定。若要進一步了解，請[使用 Amazon CloudWatch 指標](#)。

## 管理機群指標

本主題說明如何使用主 AWS IoT 控制台 AWS CLI 以及管理叢集指標。

### 主題

- [管理機群指標 \(主控台\)](#)
- [管理機群指標 \(CLI\)](#)
- [授權對 IoT 資源進行標記](#)

## 管理機群指標 (主控台)

以下各節說明如何使用主 AWS IoT 控制台來管理叢集指標。在建立機群指標之前，請確保已啟用機群索引與相關資料來源和組態。

### 啟用機群索引

如果已啟用機群索引，請略過此區段。

如果尚未啟用機群索引，請依照下列指示進行。

1. 請在以下位置開啟 AWS IoT 主機：<https://console.aws.amazon.com/iot/>。

2. 在 AWS IoT 功能表上，選擇 [設定]。
3. 若要檢視詳細設定，請在 Settings (設定) 頁面向下捲動到 Fleet indexing (機群索引) 區段。
4. 若要更新機群索引設定，請在 Fleet indexing (機群索引) 區段的右側，選取 Manage indexing (管理索引)。
5. 在 Manage fleet indexing (管理機群索引) 頁面上，根據需求更新機群索引設定。

- 組態

若要開啟物件索引，請將 Thing indexing (物件索引) 開啟，然後選取要從中建立索引的資料來源。

若要開啟物件群組索引，請將 Thing group indexing (物件群組索引) 開啟。

- Custom fields for aggregation - optional (彙總的自訂欄位 – 選用)

自訂欄位是欄位名稱和欄位類型對的清單。

若要新增自訂欄位配對，請選擇 Add new field (新增欄位)。請輸入自訂欄位名稱 (例如 `attributes.temperature`)，然後從 Field type (欄位類型) 選單中選取欄位類型。請注意，自訂欄位名稱開頭為 `attributes.` 並會儲存為屬性，以執行 [物件彙總查詢](#)。

若要更新並儲存設定，請選擇 Update (更新)。

## 建立機群指標

1. 請在以下位置開啟 AWS IoT 主機：<https://console.aws.amazon.com/iot/>。
2. 在 AWS IoT 功能表上選擇 [管理]，然後選擇 [叢集量度]。
3. 在 Fleet metrics (機群指標) 頁面中，選擇 Create fleet metric (建立機群指標) 並完成建立步驟。
4. 在步驟 1 中設定機群指標
  - 在 Query (查詢) 區段中輸入查詢字串，指定要執行彙總搜尋的物件或事物群組。查詢字串由屬性和值組成。對於 Properties (屬性)，選擇所需屬性；如果所需屬性未顯示在清單中，請在欄位中輸入屬性。輸入：後面的值。範例查詢字串可以是 `thingName:TempSensor*`。每輸入一個查詢字串，就按一次鍵盤上的 Enter 鍵。如果輸入多個查詢字串，請選取 and (和)、or (或)、and not (而非) 或 or not (或非) 來指定其關係。
  - 在 Report properties (報告屬性) 中，從其各自的清單中選擇 Index name (索引名稱)、Aggregation type (彙總類型) 以及 Aggregation field (彙總欄位)。接下來，在 Select data (選取資料) 中選取要彙總的資料，可以在其中選取多個資料值。
  - 選擇下一步。



## 5. 在步驟 2 中指定機群指標屬性

- 在 Fleet metric name (機群指標名稱) 欄位中，為要建立的機群指標輸入名稱。
- 在 Description - optional (描述 – 選用) 欄位中，為要建立的機群指標輸入描述。此欄位為選用欄位。
- 在小時和分鐘欄位中，輸入要叢集測量結果發送資料的時間 (多久)。CloudWatch
- 選擇下一步。

## 6. 在步驟 3 中檢閱和建立

- 檢閱步驟 1 和步驟 2 的設定。若要編輯設定，請選擇 Edit (編輯)。
- 選擇 Create fleet metric (建立機群指標)。

成功建立之後，機群指標會列示在 Fleet metric (機群指標) 頁面上。

### 更新機群指標

1. 在機群指標頁面上，選擇要更新的機群指標。
2. 在機群指標 Details (詳細資訊) 頁面上，選擇 Edit (編輯)。此操作會開啟建立步驟，讓您可以在這三個步驟中的任一步驟更新機群指標。
3. 完成機群指標的更新後，選擇 Update fleet metric (更新機群指標)。

### 刪除機群指標

1. 在機群指標頁面上，選擇要刪除的機群指標。
2. 在顯示機群指標詳細資訊的下一個頁面上，選擇 Delete (刪除)。
3. 在對話方塊中，輸入機群指標的名稱以確認刪除。
4. 選擇刪除。此步驟會永久刪除機群指標。

## 管理機群指標 (CLI)

以下各節說明如何使用 AWS CLI 來管理叢集指標。在建立機群指標之前，請確保已啟用機群索引與相關資料來源和組態。若要為物件或物件群組啟用機群索引，請遵循[管理物件索引](#)或者[管理物件群組索引](#)中的指示。

### 建立機群指標

您可以使用 create-fleet-metric CLI 命令建立叢集指標。

```
aws iot create-fleet-metric --metric-name "YourFleetMetricName" --query-string "*" --period 60 --aggregation-field "registry.version" --aggregation-type name=Statistics,values=sum
```

此命令的輸出包含機群指標的名稱和 Amazon Resource Name (ARN)。輸出看起來如下：

```
{
  "metricArn": "arn:aws:iot:us-east-1:111122223333:fleetmetric/YourFleetMetricName",
  "metricName": "YourFleetMetricName"
}
```

### 清單機群指標

您可以使用 `list-fleet-metric` CLI 命令列出帳戶中的所有叢集指標。

```
aws iot list-fleet-metrics
```

此命令的輸出包含所有機群指標。輸出看起來如下：

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

### 描述機群指標

您可以使用 `describe-fleet-metric` CLI 命令來顯示有關叢集測量結果的更多詳細資訊。

```
aws iot describe-fleet-metric --metric-name "YourFleetMetricName"
```

此命令的輸出包含與指定機群指標相關的詳細資訊。輸出看起來如下：

```
{
  "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"
}
```

## 更新機群指標

您可以使用 `update-fleet-metric` CLI 命令更新叢集指標。

```
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
```

該 `update-fleet-metric` 命令不會產生任何輸出。您可以使用 `describe-fleet-metric` CLI 命令查看結果。

```
{
  "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"
}
```

## 刪除機群指標

使用 `delete-fleet-metric` CLI 命令刪除叢集測量結果。

```
aws iot delete-fleet-metric --metric-name "YourFleetMetricName"
```

如果刪除成功或指定機群指標不存在，則此命令不會產生任何輸出。

如需詳細資訊，請參閱[針對機群指標進行疑難排解](#)。

## 授權對 IoT 資源進行標記

为了更好地控制可以建立、修改或使用的叢集指標，您可以將標籤附加至叢集指標。

若要標記您使用 AWS Management Console 或建立的叢集指標 AWS CLI，您必須在 IAM 政策中包含 `iot:TagResource` 動作，才能授與使用者權限。如果您的 IAM 政策不包含 `iot:TagResource`，任何使用標籤建立叢集指標的動作都會傳回錯誤 `AccessDeniedException` 誤。

有關標記資源的一般資訊，請參閱[標記資 AWS IoT 源](#)。

## IAM 政策範例

建立叢集指標時，請參閱下列 IAM 政策範例授與標記權限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iot:TagResource"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iot:*:*:fleetmetric/*"
      ]
    }
  ]
}
```

```
},
{
  "Action": [
    "iot:CreateFleetMetric"
  ],
  "Effect": "Allow",
  "Resource": [
    "arn:aws:iot:*:*:index/*",
    "arn:aws:iot:*:*:fleetmetric/*"
  ]
}
]
```

如需詳細資訊，請參閱[適用於 AWS IoT 的動作、資源和條件鍵](#)。

# MQTT 型檔案交付

您可以用來管理檔案並將檔案傳輸到叢集中的 AWS IoT 裝置的其中一個選項是以 MQTT 為基礎的檔案傳遞。使用 AWS Cloud 中的此功能，您可以創建包含多個文件的流，可以更新流數據（文件列表和描述），獲取流數據等。AWS IoT 基於 MQTT 的文件傳遞可以使用 MQTT 協議將數據以小塊的形式傳輸到您的 IoT 設備，並支持 JSON 或 CBOR 中的請求和響應消息。

如需使用在 IoT 裝置之間傳輸資料的方式的詳細資訊 AWS IoT，請參閱[將裝置連接至 AWS IoT](#)。

## 主題

- [什麼是串流？](#)
- [管理 AWS 雲端中的串流](#)
- [在裝置中使用以 AWS IoT MQTT 為基礎的檔案傳遞](#)
- [FreeRTOS OTA 中的範例使用案例](#)

## 什麼是串流？

在中 AWS IoT，串流是可公開尋址的資源，是可傳輸至 IoT 裝置之檔案清單的抽象概念。典型的串流包含下列資訊：

- 一個在特定時間唯一識別串流的 Amazon 資源名稱 (ARN)。此 ARN 的模式為 `arn:partition:iot:region:account-ID:stream/stream ID`。
- 可識別您的串流並在 () 或 SDK 命令中使用 AWS Command Line Interface (通常需要 AWS CLI) 的串流 ID。
- 串流說明，提供串流資源的說明。
- 串流版本，識別串流的特定版本。由於串流資料可在裝置開始資料傳輸之前立即修改，因此裝置可使用串流版本來強制執行一致性檢查。
- 檔案清單，可傳輸至裝置。對於清單中的每個檔案，串流會記錄檔案 ID、檔案大小及檔案的地址資訊，這些資訊包含例如 Amazon S3 儲存貯體名稱、物件金鑰和物件版本。
- 一種 AWS Identity and Access Management (IAM) 角色，授與 AWS IoT MQTT 型檔案傳遞權限，以讀取儲存在資料儲存體中的串流檔案。

AWS IoT 以 MQTT 為基礎的檔案傳送提供下列功能，讓裝置可以從雲端傳輸資料：

- 使用 MQTT 通訊協定傳輸資料。

- 支援 JSON 或 CBOR 格式。
- 說明串流 ([DescribeStream](#) API) 的能力，以取得串流檔案清單、串流版本及相關資訊。
- 在小區塊形式傳送資料 ([GetStream](#) API) 的能力，使具有硬體限制的裝置可收到區塊。
- 支援每個請求的動態區塊大小，以支援具有不同記憶體容量的裝置。
- 當多個裝置從同一個串流檔案請求資料區塊時，最佳化並行串流請求。
- Amazon S3 作為串流檔案的資料儲存。
- Support 從 AWS IoT MQTT 檔案傳送至 CloudWatch

如需 MQTT 型檔案交付配額，請參閱 AWS 一般參考 中的 [AWS IoT Core Service Quotas](#)。

## 管理 AWS 雲端中的串流

AWS IoT 提供 AWS SDK 和 AWS CLI 命令，您可以用來管理 AWS 雲端中的串流。您可使用這些命令來執行下列：

- 建立串流 [CLI](#) / [SDK](#)
- 說明串流以取得其資訊。 [CLI](#) / [SDK](#)
- 列出您的 AWS 帳戶。 [CLI](#) / [SDK](#)
- 更新串流中的檔案清單或串流說明。 [CLI](#) / [SDK](#)
- 刪除串流。 [CLI](#) / [SDK](#)

### Note

目前，串流不會顯示於 AWS Management Console 之中。您必須使用 AWS CLI 或 AWS SDK 來管理中的串流 AWS IoT。此外，[嵌入式 C SDK](#) 是唯一支援 MQTT 型檔案傳輸的 SDK。

從您的裝置使用以 AWS IoT MQTT 為基礎的檔案傳送之前，您必須確定您的裝置符合下列條件，如下節所示：

- 反映透過 MQTT 傳輸資料所需的正確權限的政策。
- 您的裝置可以連線至 AWS IoT 裝置閘道。
- 一份政策聲明，說明您可以標記資源。如果 `CreateStream` 使用標籤調用，`iot:TagResource` 則需要。

從裝置使用以 AWS IoT MQTT 為基礎的檔案傳遞之前，您必須遵循下一節中的步驟，以確保您的裝置已獲得正確授權並可連線至裝置閘道。AWS IoT

## 授與您的裝置許可

您可遵循[建立 AWS IoT 政策](#)中的步驟，來建立裝置政策或使用現有的裝置政策。將政策連接至與您裝置相關聯的憑證，並將下列許可新增至裝置政策。

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

## 將您的設備 Connect 到 AWS IoT

使用 AWS IoT MQTT 型檔案傳送的裝置必須與之連線。AWS IoT 以 MQTT 為基礎的檔案傳送與 AWS IoT 雲端整合，因此您的裝置應該直接連線至資料層的[AWS IoT 端點](#)。



**Note**

AWS IoT 資料平面的端點特定於 AWS 帳戶 和區域。您必須將端點用於裝置註冊所在的區域  
AWS 帳戶 和區域 AWS IoT。

如需詳細資訊，請參閱[連接到 AWS IoT Core](#)。

## TagResource 用法

CreateStreamAPI 動作會建立串流，以透過 MQTT 傳送一或多個區塊的大型檔案。

成功的 CreateStream API 呼叫需要下列權限：

- `iot:CreateStream`
- `iot:TagResource` ( 如果CreateStream是帶標籤 )

支援這兩個權限的原則如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Action": [ "iot:CreateStream", "iot:TagResource" ],
    "Effect": "Allow",
    "Resource": "arn:partition:iot:region:accountID:stream/streamId",
  }
}
```

必須執行`iot:TagResource`原則陳述式動作，以確保使用者無法在沒有適當權限的情況下建立或更新資源上的標籤。如果沒有的特定政策聲明操作`iot:TagResource`，`AccessDeniedException`如果請求帶有標籤，則 `CreateStream` API 調用將返回。

如需詳細資訊，請參閱下列連結：

- [CreateStream](#)
- [TagResource](#)
- [標籤](#)

## 在裝置中使用以 AWS IoT MQTT 為基礎的檔案傳遞

如要啟動資料傳輸程序，裝置必須接收初始資料集，其至少包含串流 ID。在任務文件中含括初始資料集，您可使用 [任務](#) 來排程裝置的資料傳輸任務。當設備接收到初始數據集時，它應該開始與 AWS IoT 基於 MQTT 的文件傳遞進行交互。若要以 AWS IoT MQTT 為基礎的檔案傳送交換資料，裝置應該：

- 使用 MQTT 通訊協定來訂閱 [MQTT 型檔案交付主題](#)。
- 使用 MQTT 訊息傳送請求，然後等待接收回應。

您可選擇性地在初始資料集中包含串流檔案 ID 和串流版本。將串流檔案 ID 傳送至裝置可簡化裝置韌體/軟體的程式設計，因為其不需要提出 DescribeStream 請求來取得此 ID。裝置可指定 GetStream 請求中的串流版本，強制執行一致性檢查，以防串流意外更新。

### 用 DescribeStream 於獲取流數據

AWS IoT 基於 MQTT 的文件傳遞提供了將流數據發送到設備的 DescribeStream API。此 API 傳回的串流資料包括串流 ID、串流版本、串流說明和串流檔案清單，每個檔案都有一個檔案 ID 和檔案大小 (以位元組為單位)。裝置可利用此資訊來選取任意檔案以啟動資料傳輸程序。

#### Note

若您的裝置在初始資料集中收到所有必要的串流檔案 ID，您無須使用 DescribeStream API。

請依照下列步驟提出 DescribeStream 請求。

1. 訂閱「已接受」主題篩選條件 `$aws/things/ThingName/streams/StreamId/description/json`。
2. 訂閱「已拒絕」主題篩選條件 `$aws/things/ThingName/streams/StreamId/rejected/json`。
3. 將訊息傳送至 `$aws/things/ThingName/streams/StreamId/describe/json` 來發佈 DescribeStream 請求。
4. 若接受該請求，您的裝置將會在「已接受」主題篩選條件上收到 DescribeStream 回應。
5. 若請求遭到拒絕，您的裝置將會在「已拒絕」主題篩選條件上收到錯誤回應。

**Note**

若您在顯示的主題和主題篩選條件中將 `json` 替換為 `cbor`，您的裝置將會收到 CBOR 格式的訊息，這比 JSON 更精簡。

## DescribeStream 請求

JSON 中的典型 DescribeStream 請求看起來會如下列範例所示。

```
{
  "c": "ec944cfb-1e3c-49ac-97de-9dc4aaad0039"
}
```

- (選用) "c" 為用戶端字欄位。

用戶端字欄不可超過 64 個位元組。長於 64 個位元組的用戶端字欄會造成錯誤回應及 `InvalidRequest` 錯誤訊息。

## DescribeStream 回應

JSON 中的 DescribeStream 回應看起來會如下列範例所示。

```
{
  "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" 為用戶端字符欄位。若其提供於 DescribeStream 請求中，則會傳回。使用用戶端字符，使回應與其請求產生關聯。
- "s" 是整數形式的串流版本。您可使用此版本，對您的 GetStream 請求執行一致性檢查。
- "r" 包含串流中的一份檔案清單。
  - "f" 是整數形式的串流檔案 ID。
  - "z" 是以位元組數為單位的串流檔案大小。
- "d" 包含串流的說明。

## 從串流檔案取得資料區塊

您可使用 GetStream API，裝置便可接收小型資料區塊形式的串流檔案，因此其可供處理大型區塊大小有限制的裝置使用。如要接收整個資料檔案，裝置可能需要傳送或接收多個請求和回應，直至所有資料區塊已接收並加以處理為止。

### GetStream 請求

請依照下列步驟提出 GetStream 請求。

1. 訂閱「已接受」主題篩選條件 `$aws/things/ThingName/streams/StreamId/data/json`。
2. 訂閱「已拒絕」主題篩選條件 `$aws/things/ThingName/streams/StreamId/rejected/json`。
3. 將 GetStream 請求發佈至主題 `$aws/things/ThingName/streams/StreamId/get/json`。
4. 若接受該請求，您的裝置將會在「已接受」主題篩選條件上收到一個或多個 GetStream 回應。每個回應訊息包含一個單一區塊的基本資訊和資料承載。
5. 重複步驟 3 和 4，以接收所有資料區塊。若請求的資料量大於 128 KB，您必須重複這些步驟。您必須設定裝置的程式，來使用多個 GetStream 請求，以接收所有請求的資料。
6. 若請求遭到拒絕，您的裝置將會在「已拒絕」主題篩選條件上收到錯誤回應。

#### Note

- 若您在顯示的主題和主題篩選條件中將 "json" 替換為 "cbor"，您的裝置將會收到 CBOR 格式的訊息，這比 JSON 更精簡。

- AWS IoT 以 MQTT 為基礎的檔案傳遞會將區塊的大小限制為 128 KB。若您對超過 128 KB 的區塊提出請求，請求將會失敗。
- 您可對總大小大於 128 KB 的多個區塊提出請求 (例如，若您對總量為 160 KB 的資料提出 5 個區塊 (每個區塊 32 KB) 的請求)。在這種情況下，請求不會失敗，但是您的設備必須提出多個請求才能接收請求的所有數據。當您的裝置提出其他請求時，服務會傳送其他區塊。我們建議您僅於正確接收並處理先前的回應之後，才可繼續進行新的請求。
- 無論所請求的資料大小總量為何，您都應該將裝置設定為在未收到區塊或未正確收到區塊時啟動重試。

JSON 中的典型 GetStream 請求看起來會如下列範例所示。

```
{
  "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
  "s": 1,
  "f": 0,
  "l": 4096,
  "o": 2,
  "n": 100,
  "b": "..."}
}
```

- [選用] "c" 為用戶端字符欄位。

用戶端字符不能超過 64 位元組。長於 64 個位元組的用戶端字符會造成錯誤回應及 `InvalidRequest` 錯誤訊息。

- [選用] "s" 為串流版本欄位 (一個整數)。

MQTT 型檔案交付會依據此請求的版本和雲端中的最新串流版本套用一致性檢查。若從 GetStream 請求中的裝置傳送的串流版本與雲端中最新串流版本不相符，則服務會傳送錯誤回應及 `VersionMismatch` 錯誤訊息。一般而言，裝置會在初始資料集或 `DescribeStream` 的回應中接收預期的 (最新的) 串流版本。

- "f" 是串流檔案 ID (範圍為 0 到 255 的整數)。

使用或 SDK 建立或更新串流時，需要串流檔案 ID。AWS CLI 若裝置請求 ID 不存在的串流檔案，則服務會傳送錯誤回應和 `ResourceNotFound` 錯誤訊息。

- "l" 為以位元組為單位的資料區塊大小 (範圍為 256 到 131,072 的整數)。

請參閱 [建立 GetStream 要求的點陣圖](#)，以取得有關如何使用點陣圖欄位來指定 GetStream 回應中將會傳回之串流檔案部分的說明。若裝置指定一個超出範圍的區塊大小，則服務會傳送錯誤回應和 `BlockSizeOutOfBounds` 錯誤訊息。

- [選用] "o" 為串流檔案中區塊的偏移量 (範圍為 0 到 98,304 的整數)。

請參閱 [建立 GetStream 要求的點陣圖](#)，以取得有關如何使用點陣圖欄位來指定 GetStream 回應中將會傳回之串流檔案部分的說明。98,304 的最大值是以 24 MB 串流檔案大小限制為基礎，最小區塊大小為 256 個位元組。若未指定，則預設值為 0。

- [選用] "n" 為請求的區塊數量 (範圍為 0 到 98,304 的整數)。

"n" 欄位指定 (1) 請求的區塊數量，或 (2) 使用點陣圖欄位 ("b") 時，點陣圖請求將會傳回區塊數量的限制。此第二次使用是可選的。如果未定義，則預設值為 `131072/DataBlockSize`。

- [選用] "b" 是個代表所請求區塊的點陣圖。

使用點陣圖，您的裝置可以請求不連續的區塊，這使得在錯誤後處理重試更為方便。請參閱 [建立 GetStream 要求的點陣圖](#)，以取得有關如何使用點陣圖欄位來指定 GetStream 回應中將會傳回之串流檔案部分的說明。對此欄位，請將點陣圖轉換為以十六進記法表示點陣圖值的字串。點陣圖必須小於 12,288 個位元組。

#### Important

應指定 "n" 或 "b"。如果未指定，當檔案大小小於 131072 位元組 (128 KB) 時，GetStream 請求可能無效。

## GetStream 回應

對於請求的每個資料區塊，JSON 中的 GetStream 回應看似此範例。

```
{
  "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380",
  "f": 0,
  "l": 4096,
  "i": 2,
  "p": "...
}
```

- "c" 為用戶端字符欄位。若其提供於 GetStream 請求中，則會傳回。使用用戶端字符，使回應與其請求產生關聯。
- "f" 為目前資料區塊承載所屬串流檔案的 ID。
- "l" 為資料區塊承載的大小 (以位元組為單位)。
- "i" 為包含於承載中的資料區塊 ID。資料區塊的編號從 0 開始。
- "p" 包含資料區塊承載。此欄位是一個字串，表示以 [Base64](#) 編碼的資料區塊值。

## 建立 GetStream 要求的點陣圖

您可使用 GetStream 請求中的點陣圖欄位 (b)，從串流檔案獲取非連續區塊。此有助於 RAM 容量有限的裝置處理網路交付問題。裝置僅能請求那些未接收或未正確接收的區塊。點陣圖決定將會傳回的串流檔案區塊。對於點陣圖中設定為 1 的每個位元，會傳回串流檔案相對應的區塊。

下列範例說明如何在 GetStream 請求中指定點陣圖及其支援欄位。例如，您想要以 256 個位元組 (區塊大小) 的區塊接收串流檔案。將每個 256 個位元組的區塊視為具有一個數字，指定其在檔案中的位置，從 0 開始。所以區塊 0 是檔案中第一個 256 個位元組的區塊，區塊 1 是第二個區塊，依此類推。您想要從該檔案請求區塊 20、21、24 和 43。

### 區塊偏移

因為第一個區塊為數字 20，請指定偏移 (欄位 o) 為 20 以節省點陣圖中的空間。

### 區塊的數量

為確保您的裝置不會收到超過其在有限記憶體資源下可處理的區塊數，您可指定 MQTT 型檔案交付傳送的每則訊息中應傳回的區塊數量上限。請注意，若點陣圖本身指定的區塊數量小於此數，或者其讓 MQTT 型檔案交付傳送的回應訊息總大小大於每個 GetStream 請求 128 KB 的服務限制，則會忽略此值。

### 區塊點陣圖

點陣圖本身是一個以十六進記法表示的無符號位元組陣列，包含於 GetStream 請求中作為數字的字串表示。但若要建構此字串，讓我們先將點陣圖視為一長串的位元 (二進制數)。若在此序列中的某個位元設定為 1，則串流檔案相對應的區塊將會傳送回裝置。對於我們的範例，我們想要接收區塊 20、21、24 和 43，因此我們必須在點陣圖中設定位元 20、21、24 和 43。我們可以使用區塊偏移來節省空間，因此在我們從每個區塊減去偏移量之後，我們要設定位元 0、1、4 和 23，如下列範例所示。

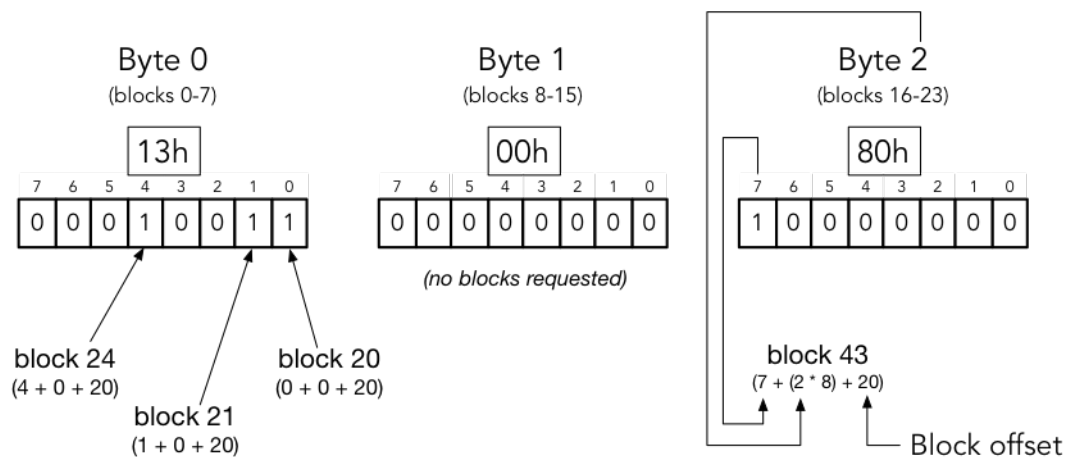
```
1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

一次取一個位元組 (8 位元)，這通常會寫成："0b00010011"、"0b00000000" 和 "0b10000000"。位元 0 出現在我們的二進製表示中的第一個位元組末尾，位元 23 出現在最後一個位元組的開頭。除非您知道慣例，否則這可能會令人困擾。第一個位元組包含位元 7-0 (依該順序)，第二個位元組包含位元 15-8，第三個位元組包含位元 23-16，依此類推。於十六進記法中，此會轉換為 "0x130080"。

### **i** Tip

您可將標準二進製轉換為十六進記法。一次取四個二進製數字，並將其轉換為相對應的十六進制。例如，"0001" 變成 "1"，"0011" 變成 "3" 等等。

### Block bitmap breakdown



$$\text{block number} = (\text{bit position} + (\text{byte offset} * 8) + \text{base offset})$$

綜上所述，我們 GetStream 請求的 JSON 如下所示。

```
{
  "c" : "1",
  "s" : 1,
  "l" : 256,
  "f" : 1,
  "o" : 20,
  "n" : 32,
  "b" : "130080"
}
```

- "c" 為用戶端字符欄位。



- 「s」是預期串流版本。
- "l" 為資料區塊承載的大小 (以位元組為單位)。
- 「f」是來源檔案索引的 ID。
- 「o」是區塊位移。
- 「n」是區塊數量。
- 「b」是從位移開始遺失的區塊點陣圖。此值必須是 based64 編碼。

## 處理以 AWS IoT MQTT 為基礎的檔案傳送的錯誤

對 DescribeStream 和 GetStream 兩個 API 傳送至裝置的錯誤回應包含用戶端字符、錯誤碼及錯誤訊息。典型的錯誤回應看似下列範例所示。

```
{
  "o": "BlockSizeOutOfBounds",
  "m": "The block size is out of bounds",
  "c": "1bb8aaa1-5c18-4d21-80c2-0b44fee10380"
}
```

- "o" 為指出錯誤發生原因的錯誤代碼。如需詳細資訊，請參閱本節稍後的錯誤碼。
- "m" 為包含錯誤詳細資料的錯誤訊息。
- "c" 為用戶端字符欄位。若其提供於 DescribeStream 請求中，則可能會傳回。您可使用用戶端字符，使回應與其請求產生關聯。

用戶端字符欄位不一定會包含在錯誤回應中。當提供於請求中的用戶端字符無效或格式錯誤時，其不會傳回錯誤回應。

### Note

若為回溯相容性，錯誤回應中的欄位可能為非縮寫形式。例如，錯誤代碼可能由「代碼」或 "o" 欄位進行指定，用戶端字符欄位可由 "clientToken" 或 "c" 欄位進行指定。我們建議您使用上述縮寫形式。

## InvalidTopic

串流訊息的 MQTT 主題是無效的。

## InvalidJson

串流請求不是一個有效的 JSON 文件。

## InvalidCbor

串流請求不是有效的 CBOR 文件。

## InvalidRequest

該請求通常被標識為格式錯誤。如需詳細資訊，請參閱錯誤訊息。

## 未經授權

該請求無權存取儲存媒體 (例如 Amazon S3) 中的串流資料檔案。如需詳細資訊，請參閱錯誤訊息。

## BlockSizeOutOfBounds

區塊大小超出範圍。請參閱 [AWS IoT Core Service Quotas](#) 中的「MQTT 型檔案交付」一節。

## OffsetOutOfBounds

偏移超出範圍。請參閱 [AWS IoT Core Service Quotas](#) 中的「MQTT 型檔案交付」一節。

## BlockCountLimitExceeded

請求區塊的數量超出範圍。請參閱 [AWS IoT Core Service Quotas](#) 中的「MQTT 型檔案交付」一節。

## BlockBitmapLimitExceeded

請求點陣圖的大小超出範圍。請參閱 [AWS IoT Core Service Quotas](#) 中的「MQTT 型檔案交付」一節。

## ResourceNotFound

找不到請求的串流、檔案、檔案版本或區塊。如需詳細資訊，請參閱錯誤訊息。

## VersionMismatch

請求中的串流版本與 MQTT 型檔案交付功能中的串流版本不相符。這表示自裝置最初接收串流版本以來，串流資料已遭修改。

## E TagMismatch

串流中的 S3 ETag 與最新 S3 物件版本的 ETag 不相符。

## InternalError

於 MQTT 型檔案交付時發生內部錯誤。

## FreeRTOS OTA 中的範例使用案例

FreeRTOS OTA (over-the-air) 代理程式會使用以 AWS IoT MQTT 為基礎的檔案傳送，將 FreeRTOS 韌體映像檔傳輸到 FreeRTOS 裝置。要將初始數據集發送到設備，它使用 AWS IoT Job 服務將 OTA 更新任務安排到 FreeRTOS 設備。

如需 MQTT 型檔案交互用戶端的參考實作，請參閱 FreeRTOS 文件中的 [FreeRTOS OTA 代理程式程式碼](#)。

# Device Advisor

[Device Advisor](#) 是一種以雲端為基礎的全受管測試功能，可在裝置軟體開發期間驗證 IoT 裝置。Device Advisor 提供預先建置的測試，您可以在將裝置部署到生產環境之前 AWS IoT Core，先驗證 IoT 裝置是否具有可靠且安全的連線能力。Device Advisor 的預先建置測試可協助您針對最佳實務驗證您的裝置軟體，是否可以使用 [TLS](#)、[MQTT](#)、[Device Shadow](#)，以及 [IoT 任務](#)。您也可以下載已簽署的資格報告，提交給 AWS 合作夥伴網路，讓您的裝置符合 [AWS Partner Device Catalog](#) 的資格，無需將裝置送入其中並等待裝置進行測試。

## Note

us-east-1、us-west-2、ap-northeast-1 和 eu-west-1 區域中支援 Device Advisor。  
裝置建議程式支援使用 MQTT 和 MQTT 透過 WebSocket 安全 (WSS) 通訊協定發佈和訂閱訊息的裝置和用戶端。所有通訊協定皆支援 IPv4 和 IPv6。  
Device Advisor 支援 RSA 伺服器憑證。

任何已建立連線的裝置都 AWS IoT Core 可以利用「裝置建議程式」。您可以從[AWS IoT 主控台](#)或使用 AWS CLI 或 SDK 存取裝置建議程式。當您準備好測試裝置時，請使用裝置建議程式端點註冊裝置軟體，AWS IoT Core 並設定裝置軟體。然後選擇預先建置的測試、設定它們、在您裝置上執行這些測試，並取得測試結果以及詳細的日誌或資格報告。

裝置建議程式是 AWS 雲端中的測試端點。您可以測試裝置，方法為將它們設定為連接至 Device Advisor 提供的測試端點。將設備配置為連接到測試端點後，您可以訪問 Device Advisor 的控制台或使用 AWS SDK 選擇要在設備上運行的測試。Device Advisor 接著會管理測試的完整生命週期，包括佈建資源、排程測試程序、管理狀態機器、記錄裝置行為、記錄結果，以及以測試報告的形式提供最終結果。

## TLS 通訊協定

Transport Layer Security (TLS) 通訊協定用來加密通過不安全網路 (例如網際網路) 的機密資料。TLS 通訊協定的前身是 Secure Sockets Layer (SSL) 通訊協定。

Device Advisor 支援下列 TLS 通訊協定：

- TLS 1.3 (建議)
- TLS 1.2

## 通訊協定、連接埠映射和身分驗證

裝置通訊協定是由裝置或用戶端用來使用裝置端點連線至訊息代理程式。下表列出 Device Advisor 端點支援的通訊協定，以及它們使用的身分驗證方法和連接埠。

### 通訊協定、身分驗證和連接埠對應

通訊協定	支援的操作	身分驗證	連線埠	ALPN 通訊協定名稱
MQTT 超過 WebSocket	發佈、訂閱	Signature 第 4 版	443	N/A
MQTT	發佈、訂閱	X.509 用戶端憑證	8883	x-amzn-mqtt-ca
MQTT	發佈、訂閱	X.509 用戶端憑證	443	N/A

本章包含下列部分：

- [設定](#)
- [在主控台中開始使用 Device Advisor](#)
- [Device Advisor 工作流程](#)
- [Device Advisor 詳細主控台工作流程](#)
- [長時間測試主控台工作流程](#)
- [Device Advisor VPC 端點 \(AWS PrivateLink\)](#)
- [Device Advisor 測試案例](#)

## 設定

第一次使用 Device Advisor 之前，請先完成以下任務：

### 建立 IoT 物件

首先，建立 IoT 物件並將憑證連接至該物件。如需如何建立物件的教學課程，請參閱[建立物件](#)。

## 建立要用作裝置角色的 IAM 角色

### Note


您可以使用 Device Advisor 主控台快速建立裝置角色。若要了解如何使用 Device Advisor 主控台設定裝置角色，請參閱[在主控台中開始使用 Device Advisor](#)。

1. 移至[AWS Identity and Access Management 主控台](#)，然後登入 AWS 帳戶 您用於「裝置建議程式」測試的。
2. 在左側導覽窗格上，選擇 Policies (政策)。
3. 選擇 Create policy (建立政策)。
4. 在 Create policy (建立政策) 下，執行下列操作：
  - a. 針對 Service (服務)，選擇 IoT。
  - b. 在動作下，執行下列其中一個動作：
    - (建議) 根據連接至您在上一節中建立之 IoT 物件或憑證的政策選取動作。
    - 在篩選動作方塊中搜尋下列動作，並選取它們：
      - Connect
      - Publish
      - Subscribe
      - Receive
      - RetainPublish
  - c. 在資源下，限制用戶端、主題和主題資源。限制這些資源是安全最佳實務。若要限制資源，請執行下列動作：
    - i. 選擇 Specify client resource ARN for the Connect action (指定 Connect 動作的用戶端資源 ARN)。
    - ii. 選擇新增 ARN，然後執行下列任一動作：

### Note


此 clientId 是您裝置用來與 Device Advisor 互動的 MQTT 用戶端 ID。

- 在視覺化 ARN 編輯器中指定區域、accountID 和 clientID。
  - 手動輸入您想要透過其執行測試案例之 IoT 主題的 Amazon Resource Name (ARN)。
- iii. 選擇新增。
  - iv. 選擇指定 Receive 動作和另外 1 個動作的主題資源 ARN。
  - v. 選擇新增 ARN，然後執行下列任一動作：

 Note

主題名稱是您裝置將訊息發佈至其中的 MQTT 主題。

- 在視覺化 ARN 編輯器中指定區域、accountID 和主題名稱。
  - 手動輸入您想要透過其執行測試案例之 IoT 主題的 ARN。
- vi. 選擇 Add (新增)。
  - vii. 選擇指定 Subscribe 動作的 topicFilter 資源 ARN。
  - viii. 選擇新增 ARN，然後執行下列任一動作：

 Note

主題名稱是您裝置訂閱的 MQTT 主題。

- 在視覺化 ARN 編輯器中指定區域、accountID 和主題名稱。
  - 手動輸入您想要透過其執行測試案例之 IoT 主題的 ARN。
- ix. 選擇新增。
5. 選擇下一步：標籤。
  6. 選擇下一步：檢閱。
  7. 在檢閱政策下，針對您的政策輸入一個名稱。
  8. 選擇 Create policy (建立政策)。
  9. 在左側導覽窗格上，選擇 Roles (角色)。
  10. 選擇建立角色。

11. 在選取信任的實體下，選擇自訂信任政策。
12. 在自訂信任政策方塊中輸入下列信任政策。建議您新增全域條件內容金鑰 [aws:SourceArn](#) 和 [aws:SourceAccount](#) 至政策，保護自己免受混淆代理人問題的困擾。

**⚠ Important**

您的 `aws:SourceArn` 必須與 format: `arn:aws:iotdeviceadvisor:region:account-id:*`. 相符。請確認 `region` 與您的 AWS IoT 區域符合，且 `account-id` 與您的客戶帳戶 ID 相符。如需詳細資訊，請參閱 [預防跨服務混淆代理人](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAwsIoTCoreDeviceAdvisor",
      "Effect": "Allow",
      "Principal": {
        "Service": "iotdeviceadvisor.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn":
            "arn:aws:iotdeviceadvisor:*:111122223333:suitedefinition/*"
        }
      }
    }
  ]
}
```

13. 選擇下一步。
14. 選擇您在步驟 4 所建立的政策。
15. (選用) 在設定許可界限下，選擇使用許可界限來控制角色許可上限，然後選取您已建立的政策。
16. 選擇下一步。



17. 輸入 Role name (角色名稱) 和 Role description (角色描述)。
18. 選擇 Create Role (建立角色)。

## 為要使用 Device Advisor 的 IAM 使用者建立自訂受管政策

1. 導覽至 IAM 主控台，網址為 <https://console.aws.amazon.com/iam/>。若出現提示，請輸入 AWS 憑證以進行登入。
2. 在左側導覽窗格中選擇 Policies (政策)。
3. 選擇 Create Policy (建立政策)，然後選擇 JSON 標籤。
4. 新增必要的許可以使用 Device Advisor。政策文件可在[安全最佳實務](#)主題中找到。
5. 選擇檢閱政策。
6. 輸入 Name (名稱) 和 Description (描述)。
7. 選擇 Create Policy (建立政策)。

## 建立要使用 Device Advisor 的 IAM 使用者

### Note

我們建議您建立要在執行 Device Advisor 測試時使用的 IAM 使用者。從管理員使用者執行 Device Advisor 測試可能會造成安全風險，因此不建議此操作。

1. 導覽至位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台；若出現提示，請輸入 AWS 憑證以進行登入。
2. 在左側導覽窗格中，選擇 Users (使用者)。
3. 選擇 Add User (新增使用者)。
4. 輸入 User name (使用者名稱)。
5. 如果使用者想要與 AWS 之外的 AWS Management Console 授與程式設計存取 AWS 取權的方式取決於正在存取的使用者類型。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分  (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。	<p>請依照您要使用的介面所提供的指示操作。</p> <ul style="list-style-type: none"> <li>如需詳細資訊 AWS CLI，請參閱 <a href="#">《使 AWS CLI 用 AWS Command Line Interface 者指南》</a> AWS IAM Identity Center 中的〈配置使用〉。</li> <li>如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 <a href="#">IAM 身分中心身分驗證</a>。</li> </ul>
IAM	使用臨時登入資料來簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。	遵循 <a href="#">《IAM 使用者指南》</a> 中的〈 <a href="#">將臨時登入資料搭配 AWS 資源使用</a> 〉中的指示
IAM	(不建議使用) 使用長期認證簽署對 AWS CLI、AWS SDK 或 AWS API 的程式設計要求。	<p>請依照您要使用的介面所提供的指示操作。</p> <ul style="list-style-type: none"> <li>如需相關資訊 AWS CLI，請參閱使用指南中的 <a href="#">使用 IAM 使用者登入資料進行驗證</a>。AWS Command Line Interface</li> <li>對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的 <a href="#">使用長期憑據進行身份驗證</a>。</li> <li>如需 AWS API，請參閱 <a href="#">IAM 使用者指南中的管理 IAM 使用者的存取金鑰</a>。</li> </ul>

6. 選擇下一步：許可。
7. 若要提供存取權，請新增權限至您的使用者、群組或角色：
  - 使用者和群組位於 AWS IAM Identity Center：  
建立權限合集。請遵循 AWS IAM Identity Center 使用者指南的 [建立許可集合](#) 中的指示。
  - 透過身分提供者在 IAM 中管理的使用者：  
建立聯合身分的角色。請遵循 IAM 使用者指南的 [為第三方身分提供者 \(聯合\) 建立角色](#) 中的指示。
  - IAM 使用者：
    - 建立您的使用者可擔任的角色。請遵循 IAM 使用者指南的 [為 IAM 使用者建立角色](#) 中的指示。
    - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的 [新增許可到使用者 \(主控台\)](#) 中的指示。
8. 在搜尋方塊中，輸入您已建立之客戶管理政策的名稱。然後，選取政策名稱的核取方塊。
9. 選擇下一步：標籤。
10. 選擇 Next:Review (下一步：檢閱)。
11. 選擇 Create user (建立使用者)。
12. 選擇關閉。

裝置顧問需要代表您存取您的 AWS 資源 (物件、憑證和端點)。您的 IAM 使用者必須擁有必要的許可。CloudWatch 如果您將必要的許可政策附加到 IAM 使用者，裝置顧問也會將日誌發佈到 Amazon。

## 設定您的裝置

Device Advisor 會使用伺服器名稱指示 (SNI) TLS 延伸來套用 TLS 組態。在裝置連線並傳遞與 Device Advisor 測試端點相同的伺服器名稱時，它們必須使用此延伸。

當測試處於 Running 狀態時，Device Advisor 允許 TLS 連線。它會在每次測試執行的前後拒絕 TLS 連線。基於這個原因，我們建議您使用裝置連線重試機制，以取得 Device Advisor 的全自動化測試體驗。您可以執行測試套件，其中包含多個測試案例，例如 TLS 連線、MQTT 連線和 MQTT 發佈。如果您執行多個測試案例，我們建議您的裝置每五秒嘗試連線到我們的測試端點。然後，您可以使多個測試案例依序自動執行。

**Note**

若要準備好您的裝置軟體進行測試，我們建議您使用可以連線至 AWS IoT Core 的 SDK。然後，您應該使用為 AWS 帳戶提供的 Device Advisor 測試端點來更新 SDK。

Device Advisor 支援兩種類型的端點：帳戶層級和裝置層級端點。選擇最適合您使用案例的端點。若要針對不同裝置同時執行多個測試套件，請使用裝置層級端點。

請執行下列命令來取得裝置層級端點：

若為使用 X.509 用戶端憑證的 MQTT 客戶：

```
aws iotdeviceadvisor get-endpoint --thing-arn your-thing-arn
```

或

```
aws iotdeviceadvisor get-endpoint --certificate-arn your-certificate-arn
```

對於使用「簽名版本 4」的 WebSocket 客戶的 MQTT：

```
aws iotdeviceadvisor get-endpoint --device-role-arn your-device-role-arn --  
authentication-method SignatureVersion4
```

若要一次執行一個測試套件，請選擇 Account-level endpoint (帳戶層級端點)。請執行下列命令來取得帳戶層級端點：

```
aws iotdeviceadvisor get-endpoint
```

## 在主控台中開始使用 Device Advisor

本教學課程可協助您在主控台 AWS IoT Core Device Advisor 上開始使用。Device Advisor 提供必要測試和已簽署的資格報告等功能。您可以使用這些測試和報告，來限定並列出 [AWS Partner Device Catalog](#) 中的裝置，如 [AWS IoT Core 資格計畫](#) 所詳述。

如需使用 Device Advisor 的詳細資訊，請參閱 [Device Advisor 工作流程](#) 和 [Device Advisor 詳細主控台工作流程](#)。

若要完成此教學課程，請遵循 [設定](#) 中概述的步驟。

### Note

下列項目支援「裝置建議程式」AWS 區域：

- 美國東部 (維吉尼亞北部)
- 美國西部 (奧勒岡)
- 亞太區域 (東京)
- 歐洲 (愛爾蘭)

## 開始使用

1. 在 [AWS IoT 主控台](#) 導覽窗格的測試下，選擇 Device Advisor。然後，選擇主控台上的開始演練按鈕。

The screenshot shows the AWS IoT Core console interface. On the left, the navigation pane is visible with the 'Test' section expanded and 'Device Advisor' highlighted. The main content area displays the Device Advisor landing page, which includes the title 'Device Advisor Fully managed test capability for IoT devices', a 'Getting started' section with a 'Start walkthrough' button, and a 'How it works' diagram. The diagram illustrates the process where a user connects and tests IoT devices configured with Device Advisor's test end point, resulting in logs and results.

2. 開始使用 Device Advisor 頁面概述建立測試套件和針對您裝置執行測試所需的步驟。您也可以在這裡針對您的帳戶尋找 Device Advisor 測試端點。您必須在用於測試的裝置上設定韌體或軟體，才能連線到此測試端點。

若要完成此教學課程，請先 [建立物件和憑證](#)。在您檢閱運作方式下的資訊之後，請選擇下一步。

**Getting started**

**How it works**

Device Advisor is a fully managed test capability for IoT Devices intending to connect to AWS IoT Core.

**Step 1: Select a protocol**  
Select a communication protocol used by your device. Tests for that particular protocol will be presented when you create your test suite.

**Step 2: Create a test suite**  
Create a test suite with at least one test group and one test. You can make your own test suite from tests that verify your devices can reliably and securely connect to AWS IoT. You will specify the test settings that allow Device advisor to work with your particular device.  
[Learn more about test suites](#)

**Step 3: Configure device settings**  
Configure device settings to test. Device Advisor will verify that the device can securely and reliably connect to, interact with and receive updates from AWS IoT. You can get detailed logs to troubleshoot device issues.

Cancel **Next**

3. 在步驟 1：選取通訊協定中，從列出的選項中選取通訊協定。然後選擇下一步。

**Select a protocol**

**Protocol info**

Choose the communication protocol used by your device. Tests for that particular protocol will be presented when you create your test suite.

MQTT 3.1.1  MQTT 5

MQTT 3.1.1 over WebSocket  MQTT 5 over WebSocket

Cancel **Next**

4. 在 Step 2 (步驟 2) 中，您可以建立和設定自訂測試套件。一個自訂測試套件必須至少具有一個測試群組，而且每個測試群組必須至少具有一個測試案例。我們已新增 MQTT Connect 測試案例，供您開始使用。

選擇 Test suite properties (測試套件屬性)。

The screenshot displays the 'Create test suite' interface in the AWS IoT Core console. The breadcrumb trail is 'AWS IoT > Test > Device Advisor > Getting started'. The main heading is 'Create test suite'. Below the heading, there is a progress bar with four steps: Step 1 (Select a protocol), Step 2 (Create test suite), Step 3 (Configure device settings), and Step 4 (Review). The 'Create test suite' step is currently active. The interface shows a 'Test suite March 22, 2023, 10:29:27 (UTC-0700)' with a 'Test suite name' field. A red box highlights the 'Test suite properties' button in the top right corner. The 'Test cases' section shows a list of MQTT test cases, including 'MQTT Connect', 'MQTT Connect Jitter Retries', and 'MQTT Connect Exponential Backoff Retries'. The 'Start' section shows a diagram of the test suite execution flow, starting from the 'Start' point and moving to 'Test group 1'. The 'Test group 1' section has an 'Edit' button.

在創建測試套件時提供測試套件屬性。您可以設定以下套件層級屬性：

- Test suite name (測試套件名稱)：輸入測試套件名稱。
- 逾時 (選用)：目前測試套件中每個測試案例的逾時 (以秒為單位)。如果您未指定逾時值，則系統會使用預設值。
- Tags (標籤) (選用)：將標籤新增至測試套件。

完成後，請選擇 Update properties (更新屬性)。

**Test suite properties** ✕

**Test suite name**  
Specify a name for this test suite that you can search.

Device Advisor Demo Suite

**Timeout - optional**  
Optional, in seconds. Maximum time Device Advisor waits for a device to respond before tests fail.

300

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

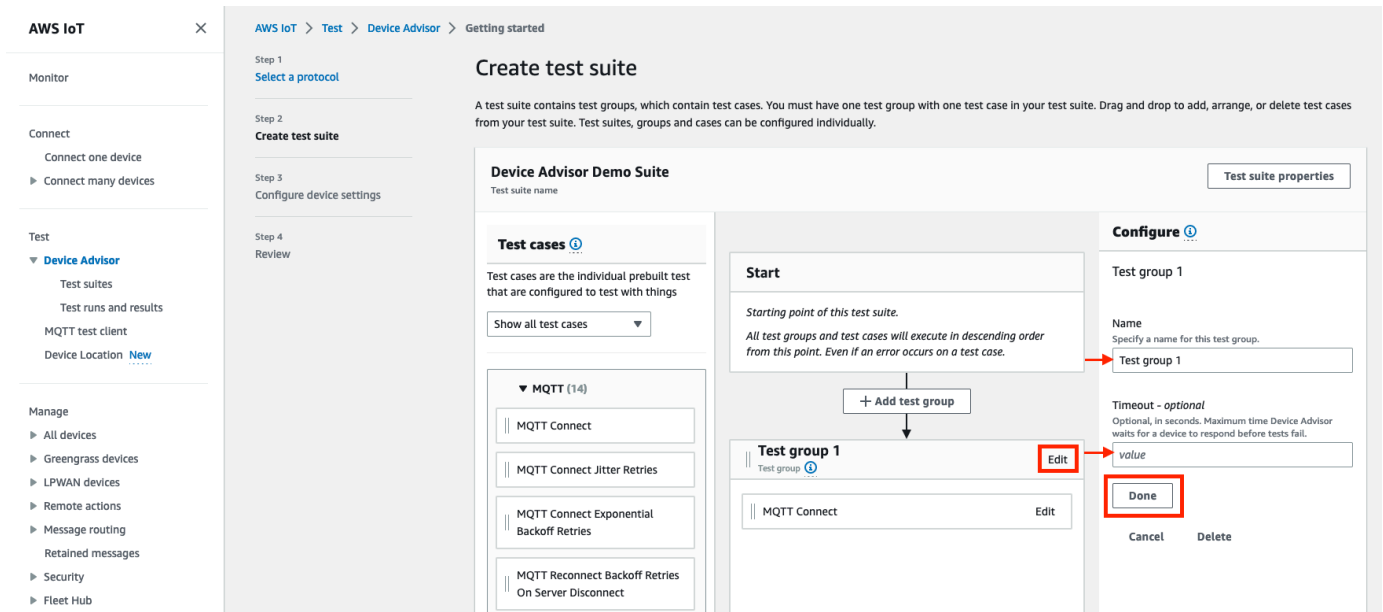
Cancel **Update properties**

5. (選用) 若要更新測試套件群組組態，請選擇測試群組名稱旁邊的編輯按鈕。

- Name (名稱)：輸入測試套件群組的自訂名稱。
- 逾時 (選用)：目前測試套件中每個測試案例的逾時 (以秒為單位)。如果您未指定逾時值，則系統會使用預設值。

完成時，請選擇完成以繼續。

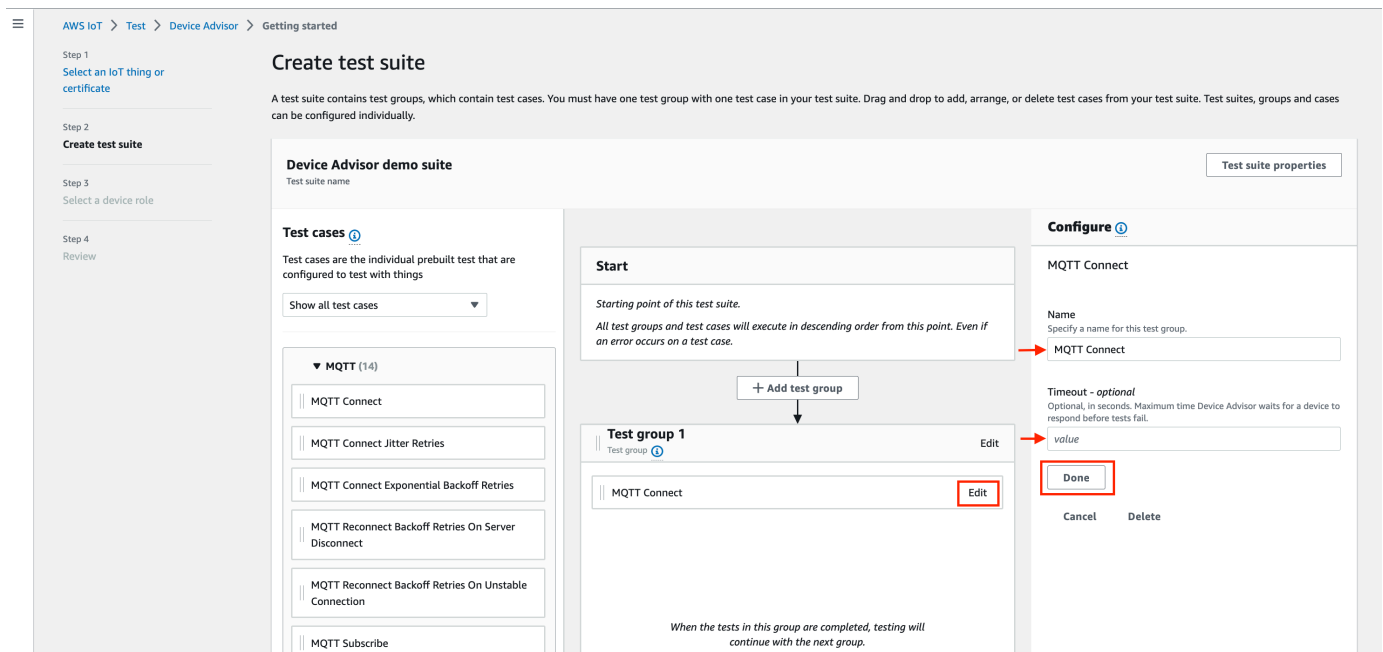




6. (選用) 若要更新測試案例的測試案例組態，請選擇測試案例名稱旁邊的編輯按鈕。

- Name (名稱)：輸入測試套件群組的自訂名稱。
- 逾時 (選用)：所選測試案例的逾時 (以秒為單位)。如果您未指定逾時值，則系統會使用預設值。

完成時，請選擇完成以繼續。



7. (選用) 若要將更多測試群組新增至測試套件，請選擇新增測試群組，然後遵循步驟 5 中的指示。

8. (選用) 若要新增更多測試案例，請將測試案例區段中的測試案例拖曳至到您的任何測試群組。

The screenshot displays the 'Create test suite' wizard in the AWS IoT Core console. It is currently on Step 3, 'Configure device settings'. The 'Test cases' section lists 14 MQTT test cases, with 'MQTT Subscribe' highlighted in a red box. The 'Configure' section shows a flowchart for 'Test group 1' containing 'MQTT Connect' and 'MQTT Subscribe'.

9. 您可以變更測試群組和測試案例的順序。若要進行變更，請在清單中將列出的測試案例向上或向下拖曳。Device Advisor 會依照您列出測試的順序執行這些測試。

在設定了您的測試套件之後，請選擇 Next (下一步)。

10. 在步驟 3 中，選取要使用「裝置建議程式」測試的 AWS IoT 物件或憑證。如果沒有任何現有物件或憑證，請參閱[設定](#)。

The screenshot displays the 'Configure device settings' wizard in the AWS IoT Core console. It is currently on Step 3, 'Configure device settings'. The 'Select a thing or a certificate' section shows a list of things, with 'MyThing' selected.

11. 您可以設定「裝置建議程式」用來代表測試裝置執行 AWS IoT MQTT 動作的裝置角色。僅針對 MQTT Connect 測試案例，才會自動選取 Connect 動作。這是因為裝置角色需要此許可才能執行測試套件。對於其他測試案例，系統會選取對應的動作。

提供每個選取動作的資源值。例如，對於 Connect 動作，請提供裝置用來連線至 Device Advisor 端點的用戶端 ID。您可以使用逗號分隔值提供多個值，並以萬用字元 (\*) 作為這些值的字首。例如，若要對以 MyTopic 為開頭的任何主題提供發佈許可，請輸入 **MyTopic\*** 作為資源值。

**AWS IoT** ×

Monitor

Connect

- Connect one device
- ▶ Connect many devices

Test

- ▼ **Device Advisor**
  - Test suites
  - Test runs and results
  - MQTT test client
  - Device Location [New](#)

Manage

- ▼ All devices
  - Things
  - Thing groups
  - Thing types
  - Fleet metrics
- ▶ Greengrass devices
- ▶ LPWAN devices

**Select a device role**

Device role [Info](#)  
AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.

Create new role  
Create and use a new device role

Select an existing role  
Use an existing device role

Role name  
DeviceAdvisorServiceRole

Permissions [Info](#)  
Choose which actions and the associated resources for AWS IoT Core Device Advisor to access using this role. You can enter a specific resource or resource prefix. To enter multiple values for a resource, use commas to separate the values. [Learn more](#)

Action	Resource type	Resource
<input checked="" type="checkbox"/> Connect	Clientid	MyClient <small>We support \$ and other special characters. * asterisk can only be added at the end</small>
<input type="checkbox"/> Publish	Topic	Specify topics to publish to, e.g. MyTopic, MyTopic* <small>We support \$ and other special characters. * asterisk can only be added at the end</small>
<input type="checkbox"/> Subscribe	TopicFilter	Specify topic filters to subscribe to, e.g. MyTopic, MyTopic* <small>We support \$ and other special characters. * asterisk can only be added at the end</small>
<input type="checkbox"/> Receive	Topic	Specify topics to receive from e.g. MyTopic, MyTopic* <small>We support \$ and other special characters. * asterisk can only be added at the end</small>
<input type="checkbox"/> RetainPublish	Topic	Specify topics to publish a retained message to, e.g. MyTopic, MyTopic* <small>We support \$ and other special characters. * asterisk can only be added at the end</small>

若要從 [設定](#) 使用先前建立的裝置角色，請選擇選取現有角色。然後在選取角色下選擇您的裝置角色。

Device Location [New](#)

Manage

- ▼ All devices
  - Things
  - Thing groups
  - Thing types
  - Fleet metrics
- ▶ Greengrass devices
- ▶ LPWAN devices

**Select a device role**

Device role [Info](#)  
AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.

Create new role  
Create and use a new device role

Select an existing role  
Use an existing device role

Select role  
DeviceAdvisorServiceRole

使用提供的兩個選項之一來設定裝置角色，然後選擇下一步。

- 在測試端點區段中，選取最適合使用案例的端點。要使用相同的測試套件同時運行多個測試套件 AWS 帳戶，請選擇設備級端點。若要一次執行一個測試套件，請選取帳戶層級端點。

LPWAN DEVICES

- ▶ Remote actions
- ▶ Message routing
  - Retained messages
- ▶ Security
- ▶ Fleet Hub

Device Software

- Billing groups
- Settings
- Feature spotlight
- Documentation [↗](#)

**Test endpoint**

Choose the endpoint that best fits your situation. If you want to simultaneously run multiple test suites then use 'Device-level endpoint', if you want to run only one test suite at a time then choose the 'Account-level endpoint'.

Account-level endpoint  
Using this endpoint, you can only run one test suite at a time.

Device-level endpoint  
Using this endpoint, you can run multiple test suites simultaneously.

Copy and paste this endpoint to your test device.  
[Redacted] amazonaws.com

Cancel Previous **Next**

13. 步驟 4 顯示所選測試裝置、測試端點、測試套件，以及所設定之測試裝置角色的概觀。若要對區段進行變更，請為要編輯的區段選擇編輯按鈕。一旦確認了您的測試組態，請選擇執行來建立測試套件並執行您的測試。

### Note

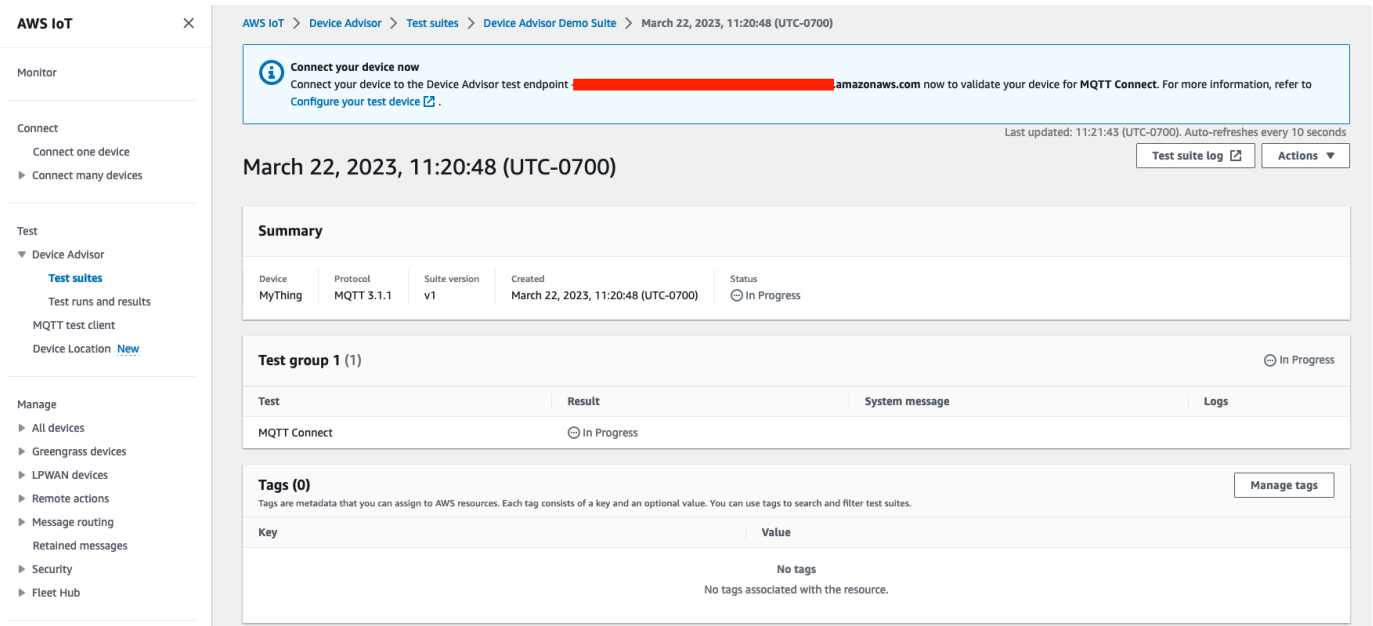
如需最佳結果，您可以在開始執行測試套件之前，將選取的測試裝置連線至 Device Advisor 測試端點。我們建議您為裝置建置一個機制，每隔五秒嘗試連接到我們的測試端點，最多持續一到兩分鐘。

The screenshot displays the AWS IoT Device Advisor console during the 'Review' phase of test suite creation. The left sidebar shows navigation options under 'Test' and 'Manage'. The main content area is titled 'Review' and is divided into three steps:

- Step 1: Select a protocol**: Shows 'Test suite type' as 'Custom test suite' and 'Protocol' as 'MQTT 3.1.1'.
- Step 2: Create test suite**: Shows 'Test suite details' including 'Test suite name' (Device Advisor Demo Suite), 'Suite version' (v1), and 'Test type' (Custom test suite). It also displays a flowchart with 'Start', 'Test group 1' (containing 'MQTT Connect'), and 'End' steps.
- Step 3: Configure device settings**: Shows 'Device role details' with 'Device' (MyThing), 'Thing ID' (redacted), 'Device role type' (Create new role), 'Thing name' (MyThing), 'Thing ARN' (redacted), and 'Device role name' (DeviceAdvisorServiceRole). It also shows 'Test endpoint' (redacted) and 'amazonaws.com'.

At the bottom right, there are 'Cancel', 'Previous', and 'Run' buttons, with 'Run' highlighted in red.

14. 在導覽窗格的測試下，選擇 Device Advisor，然後選擇測試執行和結果。選取測試套件執行，以檢視其執行詳細資訊和記錄。



The screenshot shows the AWS IoT Device Advisor console interface. On the left is a navigation sidebar with sections for Monitor, Connect, Test, and Manage. The main content area displays the breadcrumb path: AWS IoT > Device Advisor > Test suites > Device Advisor Demo Suite > March 22, 2023, 11:20:48 (UTC-0700). A notification banner at the top prompts to connect the device. Below this, the test suite details are shown: Device (MyThing), Protocol (MQTT 3.1.1), Suite version (v1), Created (March 22, 2023, 11:20:48 (UTC-0700)), and Status (In Progress). A table lists the test results for 'Test group 1 (1)', showing a single test 'MQTT Connect' with a status of 'In Progress'. At the bottom, there is a 'Tags (0)' section indicating no tags are associated with the resource.

15. 要訪問套件的 Amazon CloudWatch 日誌，請運行：

- 選擇測試套件 CloudWatch 日誌以查看測試套件運行的日誌。
- 為任何測試用例選擇測試用例日誌以查看特定於測試用例的 CloudWatch 日誌。

16. 根據您的測試結果，[疑難排解](#)您的裝置，直到所有測試都通過。

## Device Advisor 工作流程

本教學課程說明如何建立自訂測試套件，以及針對您要在主控台中測試的裝置執行測試。在測試完成之後，您可以檢視測試結果和詳細記錄。

### 必要條件

開始本教學課程之前，請先完成[設定](#)中概述的步驟。

### 建立測試套件定義

首先，[安裝一個 AWS SDK](#)。

## rootGroup 語法

根群組是 JSON 字串，指定哪些測試案例包含在您的測試套件中。它也會針對這些測試案例指定任何必要的組態。使用根群組根據您的需求建構和排序您的測試套件。測試套件的階層如下：

```
test suite # test group(s) # test case(s)
```

一個測試套件必須至少具有一個測試群組，而且每個測試群組必須至少具有一個測試案例。Device Advisor 會依您定義測試群組和測試案例的順序執行測試。

每個根群組都遵循這個基本結構：

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

在根群組中，您可以使用 `name`、`configuration`，以及群組包含的 `tests` 來定義測試套件。`tests` 群組包含個別測試的定義。您可以使用 `name`、`configuration`，以及針對該測試定義測試案例的 `test` 區塊來定義每個測試。最後，每個測試案例都是使用 `id` 和 `version` 定義的。

如需如何為每個測試案例 (`test` 區塊) 使用 `"id"` 和 `"version"` 欄位的相關資訊，請參閱 [Device Advisor 測試案例](#)。該節還包含可用 `configuration` 設定的相關資訊。

下列區塊是根群組組態的範例。此組態指定 MQTT Connect Happy Case 和 MQTT Connect Exponential Backoff Retries 測試案例，以及組態欄位的描述。

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

您必須在建立測試套件定義時提供根群組組態。儲存回應物件中傳回的 `suiteDefinitionId`。您可以使用此 ID 擷取測試套件定義資訊，以及執行測試套件。

以下是一個 Java 開發套件範例：

```
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")
                .deviceRoleArn("your-device-role-arn") //if using SigV4 for
MQTT over WebSocket
            )
            .build()
        )
        .rootGroup("your-root-group-configuration")
        .devicePermissionRoleArn("your-device-permission-role-arn")
        .protocol("MqttV3_1_1 || MqttV5 || MqttV3_1_1_OverWebSocket ||
MqttV5_OverWebSocket")
        .build()
    )
    .build()
)

```

## 取得測試套件定義

在建立您的測試套件定義之後，您會在 CreateSuiteDefinition API 操作的回應物件中收到 suiteDefinitionId。

當此操作傳回 suiteDefinitionId 時，您可能會在每個群組內看到新的 id 欄位，以及在根群組內看到測試案例定義。您可以使用這些 ID 來執行測試套件定義的子集。

Java 開發套件範例：

```

response = iotDeviceAdvisorClient.GetSuiteDefinition(
    GetSuiteDefinitionRequest.builder()
        .suiteDefinitionId("your-suite-definition-id")
        .build()
)

```

## 取得測試端點

使用 GetEndpoint API 操作取得裝置所使用的測試端點。選取最適合您測試的端點。若要同時執行多個測試套件，請透過提供 thing ARN、certificate ARN 或 device role ARN 來使用裝置層級端點。若要執行單一測試套件，請不要為 GetEndpoint 作業提供任何引數以選擇帳戶層級端點。



開發套件範例：

```
response = iotDeviceAdvisorClient.getEndpoint(GetEndpointRequest.builder()
    .certificateArn("your-test-device-certificate-arn")
    .thingArn("your-test-device-thing-arn")
    .deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

    .build())
```

## 啟動測試套件執行

在建立了測試套件定義，並將測試裝置設定為連線至 Device Advisor 測試端點之後，請使用 StartSuiteRun API 執行您的測試套件。

對於 MQTT 客戶，請使用 certificateArn 或 thingArn 執行測試套件。當設定這兩者時，如果憑證屬於物件，則會使用此憑證。

對於 MQTT WebSocket 客戶，請使用 deviceRoleArn 來運行測試套件。如果指定的角色與測試套件定義中指定的角色不同，則指定的角色會覆寫定義的角色。

對於 .parallelRun()，如果使用裝置層級端點，來使用一個 AWS 帳戶平行執行多個測試套件，請使用 true。

開發套件範例：

```
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")
            .deviceRoleArn("your-device-role-arn") //if using SigV4 for MQTT over WebSocket

            .build())
        .parallelRun(true | false)
        .build())
    .build())
```

從回應儲存 suiteRunId。您將使用此資訊來擷取此測試套件執行的結果。

## 取得測試套件執行

在您開始測試套件執行之後，您可以使用 GetSuiteRun API 來檢查其進度及結果。

開發套件範例：

```
// Using the SDK, call the GetSuiteRun API.  
  
response = iotDeviceAdvisorClient.GetSuiteRun(  
  GetSuiteRunRequest.builder()  
    .suiteDefinitionId("your-suite-definition-id")  
    .suiteRunId("your-suite-run-id")  
  .build())
```

## 停止測試套件執行

若要停止仍在進行的測試套件執行，您可以呼叫 StopSuiteRun API 操作。在您呼叫 StopSuiteRun API 操作之後，服務將會啟動清除程序。當服務執行清除程序時，測試套件執行狀態會更新為 Stopping。清除程序需要幾分鐘的時間來完成。一旦此程序完成，測試套件執行狀態就會更新為 Stopped。在測試執行完全停止之後，您可以啟動另一個測試套件執行。您可以使用 GetSuiteRun API 操作定期檢查套件執行狀態，如上節所示。

開發套件範例：

```
// Using the SDK, call the StopSuiteRun API.  
  
response = iotDeviceAdvisorClient.StopSuiteRun(  
  StopSuiteRun.builder()  
    .suiteDefinitionId("your-suite-definition-id")  
    .suiteRunId("your-suite-run-id")  
  .build())
```

## 取得成功資格測試套件執行的資格報告

如果執行成功完成的資格測試套件，則可以使用 GetSuiteRunReport API 操作擷取資格報告。您可以使用此資格報告，搭配 AWS IoT Core 資格計畫限定您的裝置。若要判斷您的測試套件是否為資格測試套件，請檢查 intendedForQualification 參數是否設定為 true。在呼叫 GetSuiteRunReport API 操作之後，您可以從傳回的 URL 下載報告，最長可達 90 秒。如果上次呼叫 GetSuiteRunReport API 操作的經歷時間超過 90 秒，請再次呼叫該操作來擷取新的有效 URL。

開發套件範例：

```
// 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 詳細主控台工作流程

在此教學課程中，您已建立自訂測試套件，並針對您要在主控台中測試的裝置執行測試。在測試完成之後，您可以檢視測試結果和詳細記錄。

### 教學課程

- [必要條件](#)
- [建立測試套件定義](#)
- [啟動測試套件執行](#)
- [停止測試套件執行 \(選用\)](#)
- [檢視測試套件執行詳細資訊和記錄](#)
- [下載 AWS IoT 資格報告](#)

## 必要條件

為完成此教學課程，您需要[建立物件和憑證](#)。

## 建立測試套件定義

1. 在 [AWS IoT 主控台](#) 的導覽窗格中，展開 Test (測試)、Device Advisor，然後選擇 Test suites (測試套件)。

The screenshot shows the AWS IoT Core console interface. On the left, the navigation menu is expanded to 'Device Advisor', with 'Test suites' highlighted. The main content area is titled 'Test suites' and includes a 'How it works' section with three cards: 'AWS IoT Core qualification test suite', 'Long duration test suite', and 'Custom test suite'. Below this, there is a table showing 'Test suites (0)' and a 'Create test suite' button.

選擇 Create Test Suite (建立測試套件)。

- 擇一選取 Use the AWS Qualification test suite 或者 Create a new test suite。

對於通訊協定，請選擇 MQTT 3.1.1 或 MQTT 5。

The screenshot shows the 'Create test suite' wizard in the AWS IoT Core console. It is divided into four steps: Step 1 'Create test suite', Step 2 'Configure test suite', Step 3 'Select a device role', and Step 4 'Review'. In Step 1, there are three radio button options for test suite types: 'AWS IoT Core qualification test suite', 'Long duration test suite', and 'Custom test suite'. In Step 2, there are two radio button options for protocols: 'MQTT 3.1.1' (which is selected) and 'MQTT 5'. 'Cancel' and 'Next' buttons are visible at the bottom right.

選取 Use the AWS Qualification test suite 以符合資格，並將您的裝置列入 AWS 合作夥伴裝置目錄。透過選擇此選項，根據 AWS IoT Core 資格計劃限定您裝置所需的測試案例是預先選取的。無法新增或移除測試群組和測試案例。不過，您仍然需要設定測試套件屬性。

選取 Create a new test suite 來建立和設定自訂測試套件。我們建議您從這個選項開始進行初始測試和疑難排解。一個自訂測試套件必須至少具有一個測試群組，而且每個測試群組必須至少具有一個測試案例。基於本教學課程的目的，我們將選取此選項並選擇 Next (下一步)。

3. 選擇 Test suite properties (測試套件屬性)。必須在建立測試套件時建立測試套件屬性。

**Configure test suite**

A test suite contains test groups, which contain test cases. You must have one test group with one test case in your test suite. Drag and drop to add, arrange, or delete test cases from your test suite. Test suites, groups and cases can be configured individually.

**Test suite December 22, 2022, 11:24:37 (UTC-0800)**  
Test suite name

**Test cases**

Test cases are the individual prebuilt test that are configured to test with things

Show all test cases

MQTT (14)

- MQTT Connect
- MQTT Connect Jitter Retries
- MQTT Connect Exponential Backoff Retries
- MQTT Reconnect Backoff Retries On Server Disconnect
- MQTT Reconnect Backoff Retries On Unstable Connection

**Start**

Starting point of this test suite.

All test groups and test cases will execute in descending order from this point. Even if an error occurs on a test case.

+ Add test group

**Test group 1**  
Test group

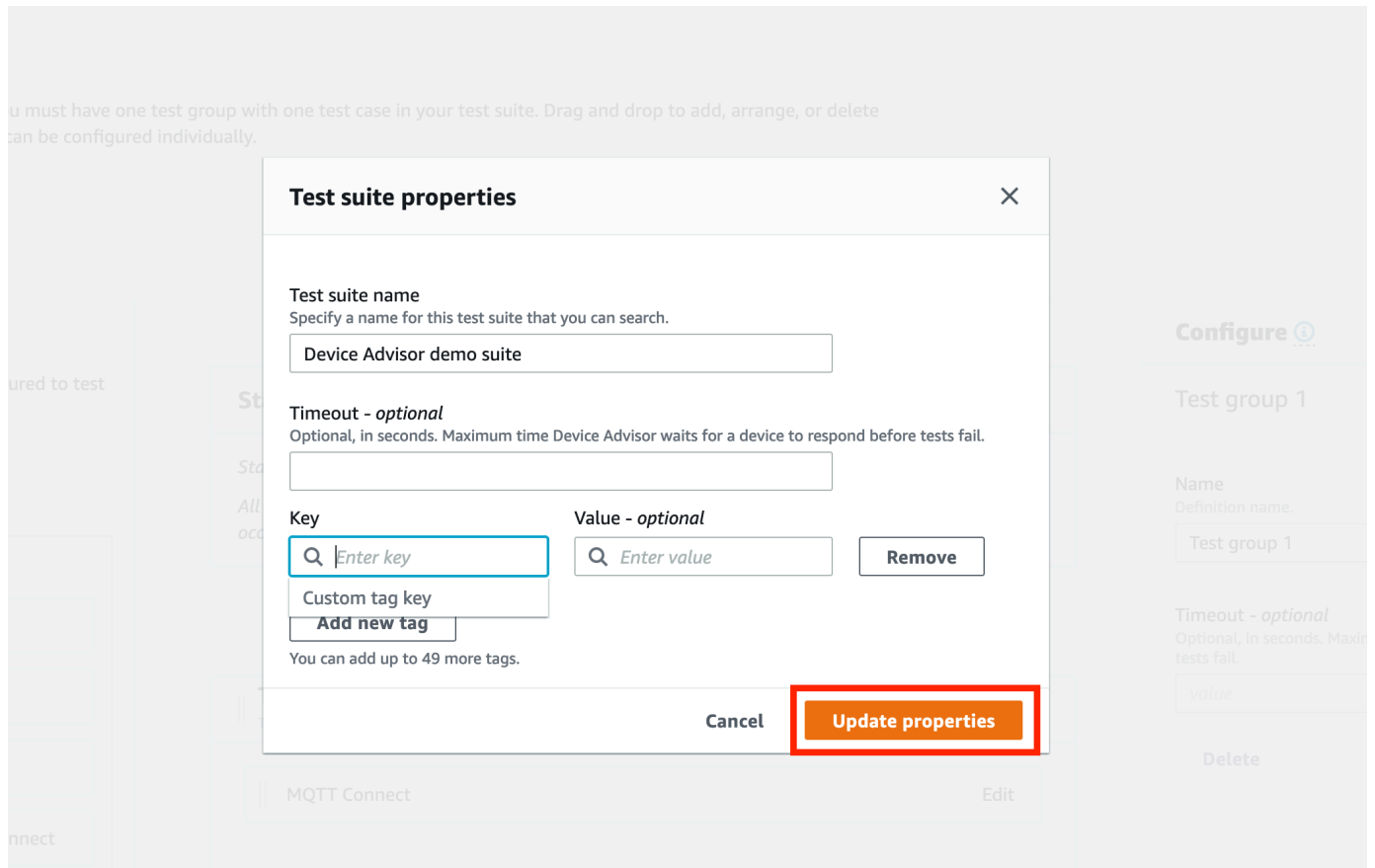
No test cases have been added to this test group.

When the tests in this group are completed, testing will continue with the next group.

**Test suite properties**

在 Test suite properties (測試套件屬性) 下，請填寫下列資訊：

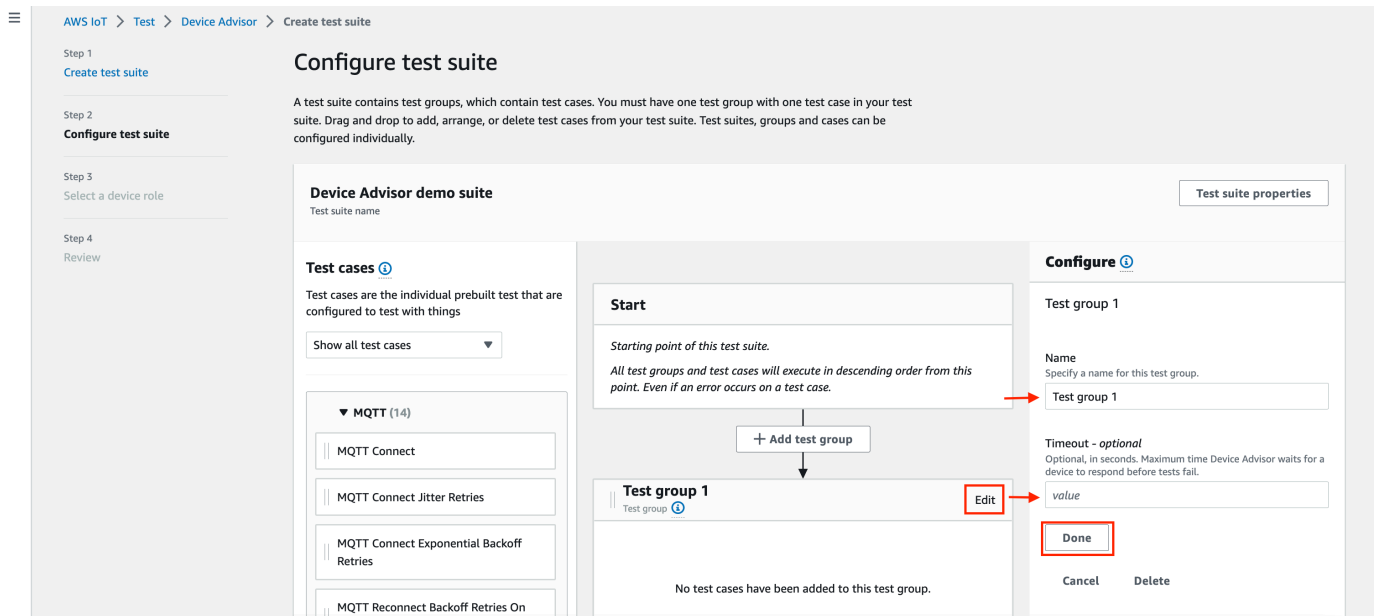
- Test suite name (測試套件名稱)：您可以使用自訂名稱建立套件。
- Timeout (逾時) (選用)：目前測試套件中每個測試案例的逾時 (以秒為單位)。如果您未指定逾時值，則系統會使用預設值。
- Tags (標籤) (選用)：將標籤新增至測試套件。



完成後，請選擇 Update properties (更新屬性)。

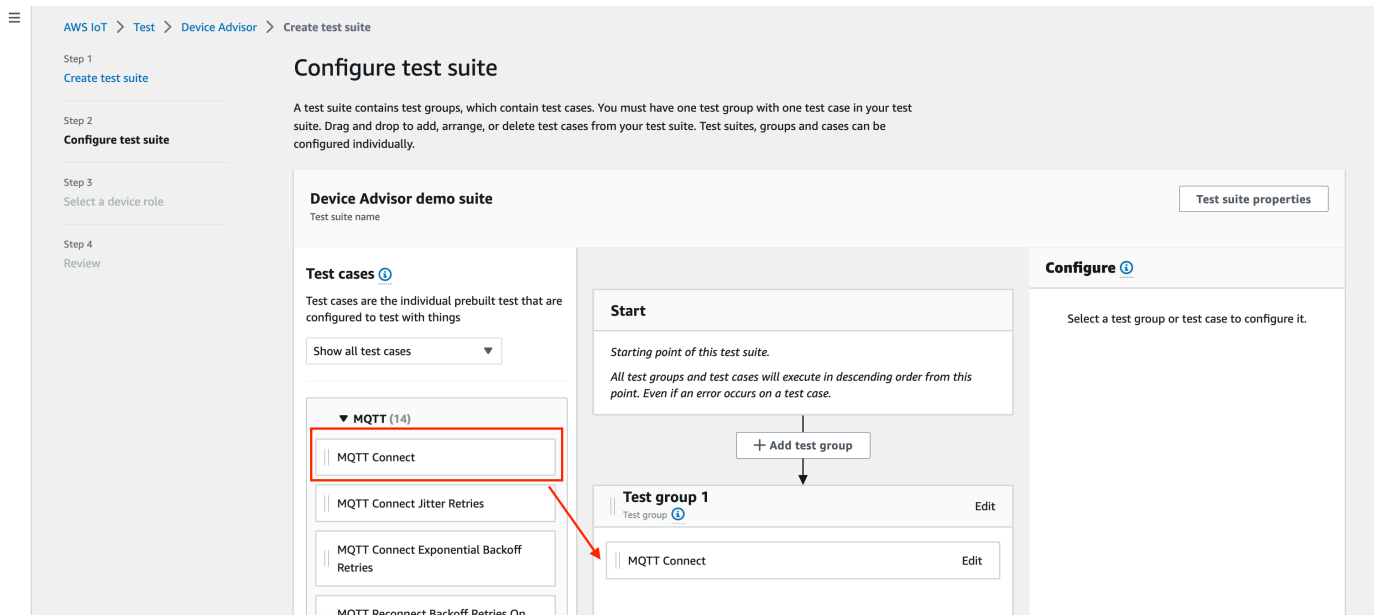
- 若要修改群組層級組態，請在 Test group 1 下，選擇 Edit (編輯)。接著，輸入 Name (名稱)，為群組提供自訂名稱。

您也可以選擇性地在選取的測試群組下輸入 Timeout (逾時) 值 (以秒為單位)。如果您未指定逾時值，則系統會使用預設值。



選擇 Done (完成)。

- 將其中一個可用的測試案例從 Test cases (測試案例) 拖曳至測試群組。



- 若要為新增至測試群組的測試案例修改測試案例層級組態，請選擇 Edit (編輯)。接著，輸入 Name (名稱)，為群組提供自訂名稱。

您也可以選擇性地在選取的測試群組下輸入 Timeout (逾時) 值 (以秒為單位)。如果您未指定逾時值，則系統會使用預設值。



The screenshot shows the 'Configure test suite' interface in AWS IoT Core. It includes a sidebar with steps: Step 1 (Create test suite), Step 2 (Configure test suite), Step 3 (Select a device role), and Step 4 (Review). The main content area is titled 'Configure test suite' and includes a 'Device Advisor demo suite' section with a 'Test cases' list (MQTT Connect, MQTT Connect Jitter Retries, etc.) and a 'Test group 1' section with an 'MQTT Connect' test case. A 'Configure' dialog box is open for the 'MQTT Connect' test case, showing fields for 'Name' (MQTT Connect) and 'Timeout - optional' (value). Red arrows point to the 'MQTT Connect' name field and the 'Edit' button for 'Test group 1'.

選擇 Done (完成)。

### Note

若要將更多測試群組新增至測試套件，請選擇 Add test group (新增測試群組)。依照上述步驟建立和設定更多測試群組，或將更多測試案例新增至一或多個測試群組。選擇測試案例並將其拖曳至所需位置，即可將測試群組和測試案例重新排序。Device Advisor 會依您定義測試群組和測試案例的順序執行測試。

7. 選擇下一步。
8. 在步驟 3 中，設定裝置角色，裝置建議程式將用來代表您的測試裝置執行 AWS IoT MQTT 動作。

如果僅在 Step 2 (步驟 2) 中選取 MQTT Connect 測試案例，系統會自動檢查 Connect (連接) 動作，因為裝置角色需要該許可才能執行此測試套件。如果選取了其他測試案例，系統會檢查相應的必要動作。確保已為每個動作提供資源值。例如，對於 Connect (連接) 動作，請提供裝置在連接至 Device Advisor 端點所用的用戶端 ID。可以使用逗號分隔值的方式提供多個值，也可以使用萬用字元 (\*) 字元來提供字首值。若要提供以 MyTopic 為開頭的主題發佈許可，則可提供 MyTopic\* 作為資源值。

**Select a device role**

**Device role** Info  
AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.

**Create new role**  
Create and use a new device role
  **Select an existing role**  
Use an existing device role

Role name  
MyDevicedvisorDeviceRole

**Permissions** Info  
Choose which actions and the associated resources for AWS IoT Core Device Advisor to access using this role. You can enter a specific resource or resource prefix. To enter multiple values for a resource, use commas to separate the values. [Learn more](#)

Action	Resource type	Resource
<input checked="" type="checkbox"/> Connect	Clientid	MyClient
<input type="checkbox"/> Publish	Topic	Specify topics to publish to, e.g. MyTopic, MyTopic*
<input type="checkbox"/> Subscribe	TopicFilter	Specify topic filters to subscribe to, e.g. MyTopic, MyTopic*
<input type="checkbox"/> Receive	Topic	Specify topics to receive from e.g. MyTopic, MyTopic*
<input type="checkbox"/> RetainPublish	Topic	Specify topics to publish a retained message to, e.g. MyTopic, MyTopic*

Cancel Previous **Next**

如果先前已建立裝置角色，而且希望使用該角色，請選擇 **Select an existing role** (選取現有角色)，然後在 **Select role** (選取角色) 下選擇裝置角色。

**Select a device role**

**Device role** Info  
AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.

**Create new role**  
Create and use a new device role
  **Select an existing role**  
Use an existing device role

Select role  
Select a device role

Cancel Previous **Next**

使用提供的兩個選項之一來設定裝置角色，然後選擇 **Next** (下一步)。

- 在 Step 4 (步驟 4)，請確保每個步驟中提供的組態正確無誤。若要編輯針對特定步驟提供的組態，請選擇 **Edit** (編輯) 來取得相應的步驟。

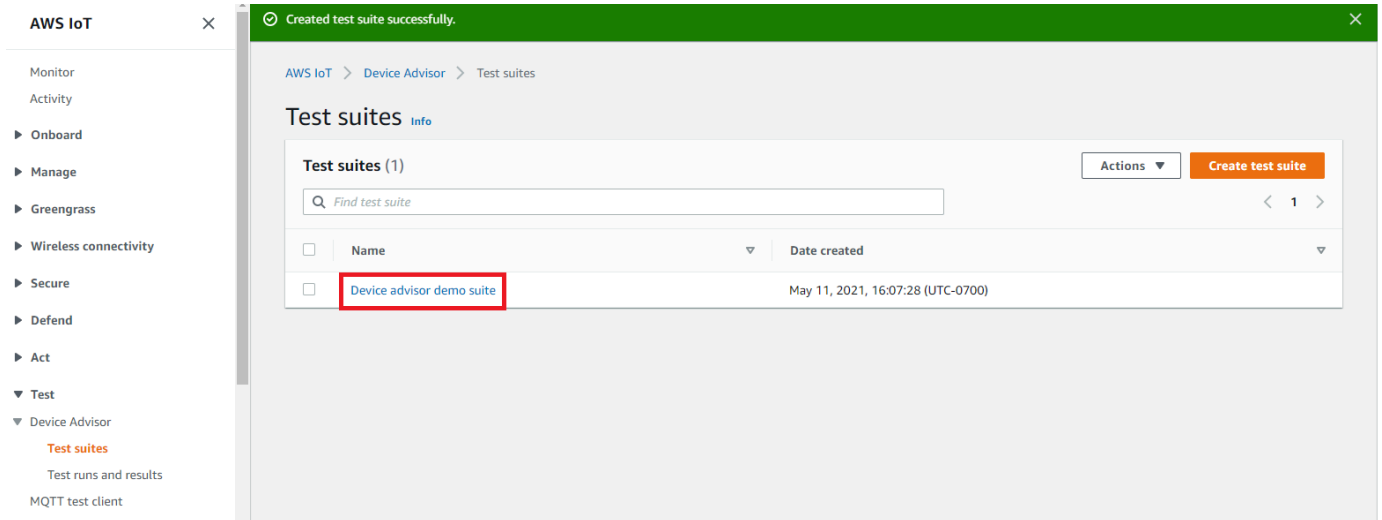
確認組態之後，請選擇 **Create test suite** (建立測試套件)。

應該成功建立測試套件，並且您將被重新導向到 **Test suites** (測試套件) 頁面，在這裡您可以檢視所有已建立的測試套件。

如果建立測試套件失敗，請確保已根據先前的指示，設定測試套件、測試群組、測試案例和裝置角色。

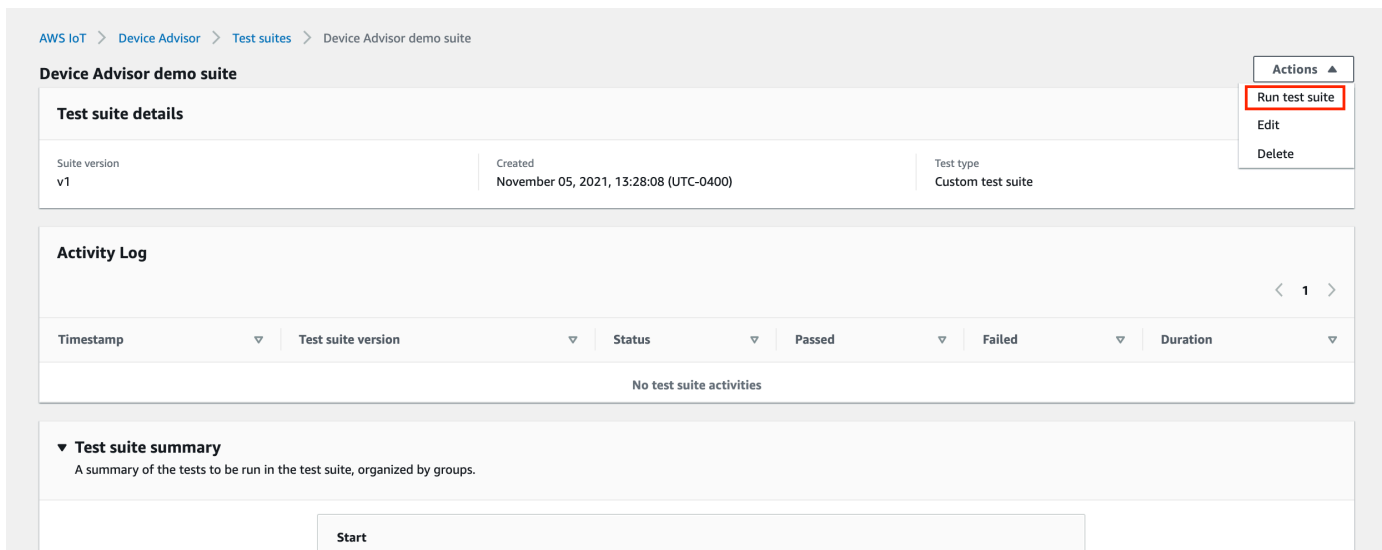
## 啟動測試套件執行

1. 在 [AWS IoT 主控台](#) 的導覽窗格中，展開 Test (測試)、Device Advisor，然後選擇 Test suites (測試套件)。
2. 選擇您要檢視哪個測試套件的詳細資訊。



測試套件詳細資訊頁面會顯示與測試套件相關的所有資訊。

3. 選擇 Actions (動作)，然後 Run test suite (執行測試套件)。



4. 在 [執行組態] 下，您需要選取要使用 [裝置建議程式] 測試的 AWS IoT 物件或憑證。如果您沒有任何現有的東西或證書，請先 [創建 AWS IoT Core 資源](#)。

在 Test endpoint (測試端點) 區段中，選取最適合案例的端點。如果您計劃 future 使用同一 AWS 帳戶同時執行多個測試套件，請選取裝置層級端點。如果打算一次只執行一個測試套件，請選取 Account-level endpoint (帳戶層級端點)。

使用選取的 Device Advisor 測試端點來設定測試裝置。

選取物件或憑證並選擇 Device Advisor 端點後，請選擇 Run test (執行測試)。

**Run configuration**

**Select test devices**

Select the IoT thing/certificate to test using the test suite. If not listed below, you must first create a thing/certificate registered with IoT Core before you can run the test suite.

Things  
Choose a thing for this test suite. To create a new thing, go to [IoT Things](#).

Certificates  
Choose a certificate for this test suite. To create a new certificate, go to [IoT Certificates](#).

**Things (1)**

filter things

Name	Type
MyThing	

**Test endpoint**

Choose the endpoint that best fits your situation. If you want to simultaneously run multiple test suites then use 'Device-level endpoint'. If you want to run only one test suite at a time then choose the 'Account-level endpoint'.

Account-level endpoint  
Using this endpoint, you can only run one test suite at a time.

Device-level endpoint  
Using this endpoint, you can run multiple test suites simultaneously.

Copy and paste this endpoint to your test device.  
t86dc41394y919y9tzu6.gamma.us-west-2.advisor.iot.aws.dev

**Tags - optional**

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Run test

5. 選擇頂端橫幅上的 Go to results (前往結果)，以檢視測試執行詳細資訊。

**'Device Advisor demo suite' is in progress with 'MyThing'.** Go to results

AWS IoT > Device Advisor > Test suites > Device Advisor demo suite

**Device Advisor demo suite** Actions

**Test suite details** v1

Suite version	Created	Test type
v1	November 05, 2021, 13:40:33 (UTC-0400)	Custom test suite

**Activity Log**

Timestamp	Test suite version	Status	Passed	Failed	Duration
November 05, 2021, 13:53:23 (UTC-0400)	v1	Pending	-	-	-

## 停止測試套件執行 (選用)

1. 在 [AWS IoT 主控台](#) 的導覽窗格中，展開 Test (測試)、Device Advisor，然後選擇 Test runs and results (測試執行和結果)。
2. 選擇您要停止的進行中測試套件。

The screenshot shows the AWS IoT console interface. On the left is a navigation sidebar with categories like Monitor, Activity, Onboard, Manage, Greengrass, Secure, Defend, Act, Test, and Device Advisor. The main content area is titled 'Test runs and results' and includes a 'Summary' section with three metrics: 'Number of IoT things available' (1), 'Number of IoT certificates available' (6), and 'Number of test suites running' (1). Below this is a table titled 'Results of test runs (in progress and completed)'. The table has columns for Name, Timestamp, Test suite version, Status, Passed, Failed, and Duration. One row is highlighted with a red box: 'Device Advisor demo suite' with a timestamp of 'December 07, 2020, 11:16:46 (UTC-0800)' and a status of 'In Progress'.

3. 選擇 Actions (動作)，然後 Stop test suite (停止測試套件)。

The screenshot shows the 'Activity log details' page in the AWS IoT console. At the top, there is a notification banner about connecting the device. Below that, the page title is 'May 11, 2021, 16:15:43 (UTC-0700)'. There are buttons for 'Test suite log' and 'Actions'. The 'Activity log details' section shows a table with columns for Device, Suite version, Created, and Status. The 'Device' is 'MyThing', 'Suite version' is 'v1', and 'Created' is 'May 11, 2021, 16:15:43 (UTC-0700)'. Below this is a 'Test group 1 (1)' section with a table showing test results. The 'Test' is 'MQTT Connect' and the 'Result' is 'In Progress'. At the bottom, there is a 'Tags - optional' section with 'Cancel' and 'Save changes' buttons. The 'Stop test suite' button in the 'Actions' menu is highlighted with a red box.

4. 此清除程序需要幾分鐘的時間來完成。當清除程序執行時，測試執行狀態將為 STOPPING。等待清除程序完成，且測試套件狀態變更為 STOPPED 狀態，再開始新的套件執行。

The screenshot displays the AWS IoT Device Advisor console interface. On the left is a navigation sidebar with categories like Monitor, Activity, Onboard, Manage, Greengrass, Wireless connectivity, Secure, Defend, Act, and Test. The main content area shows the details for a test suite named "Device advisor demo suite" which is currently "stopping". The breadcrumb path is "AWS IoT > Device Advisor > Test suites > Device advisor demo suite > May 11, 2021, 16:15:43 (UTC-0700)". The page title is "May 11, 2021, 16:15:43 (UTC-0700)".

**Activity log details**

Device	Suite version	Created	Status
MyThing	v1	May 11, 2021, 16:15:43 (UTC-0700)	Stopped

▼ Test group 1 (1) Stopped

Test	Result	System message	Logs
MQTT Connect	Stopped	No issues found	<a href="#">Test case log</a>

**Tags - optional**

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Cancel Save changes

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tags.

## 檢視測試套件執行詳細資訊和記錄

1. 在 [AWS IoT 主控台](#) 的導覽窗格中，展開 Test (測試)、Device Advisor，然後選擇 Test runs and results (測試執行和結果)。

此頁面顯示：

- IoT 物件的數目
- IoT 憑證的數目
- 目前執行的測試套件數目
- 所有已建立的測試套件執行

2. 選擇您要檢視哪個測試套件的執行詳細資訊和記錄。

The screenshot shows the 'Test runs and results' page in the AWS IoT Device Advisor console. The left sidebar contains navigation options like Monitor, Activity, Onboard, Manage, Greengrass, Secure, Defend, Act, Test, and Device Advisor. The main content area displays a summary of test runs and a table of results. The table has columns for Name, Timestamp, Test suite version, Status, Passed, Failed, and Duration. One row is highlighted with a red box, showing 'Device Advisor demo suite' with a status of 'In Progress'.

執行摘要頁面會顯示目前測試套件執行的狀態。此頁面每隔 10 秒自動重新整理一次。我們建議您為裝置建置一個機制，每隔五秒嘗試連接到我們的測試端點，持續一到兩分鐘。然後，您可以採取自動方式依序執行多個測試用例。

The screenshot shows the 'Activity log details' page in the AWS IoT Device Advisor console. The page displays details for a specific test run, including the device name, suite version, creation time, and status. A table shows the test results for 'MQTT Connect', which passed.

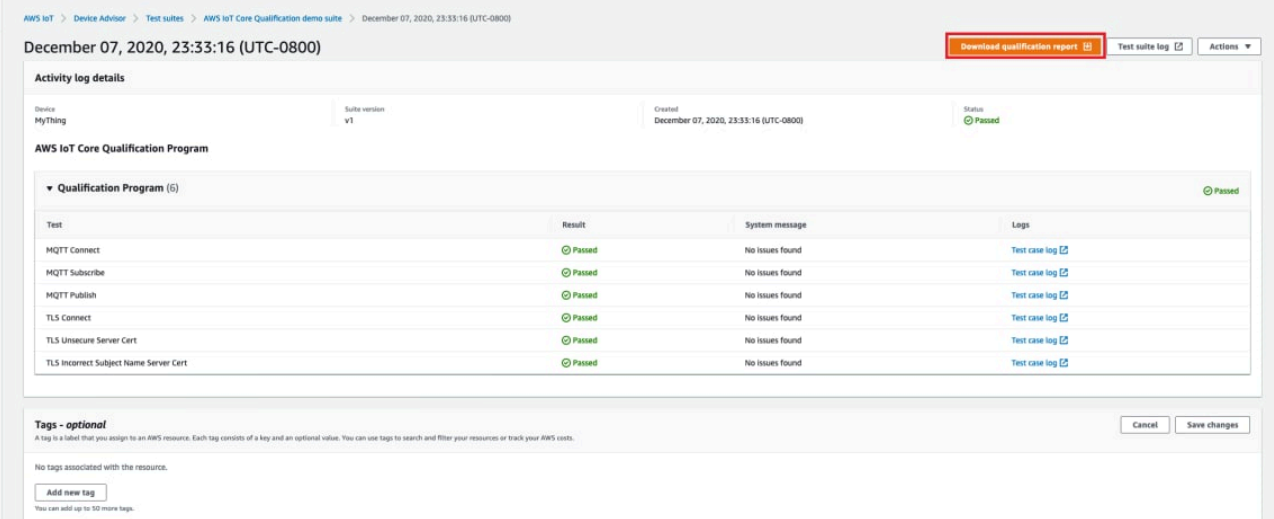
3. 要訪問測試套件運行的 CloudWatch 日誌，請選擇測試套件日誌。

要訪問任何測試用例的 CloudWatch 日誌，請選擇測試用例日誌。

4. 根據您的測試結果，[疑難排解](#)您的裝置，直到所有測試都通過。

## 下載 AWS IoT 資格報告

如果您在建立測試套件時選擇了使用 AWS IoT 資格測試套件選項，並且能夠執行資格測試套件，則可以在測試回合摘要頁面中選擇 [下載資格報告] 來下載資格報告。



December 07, 2020, 23:33:16 (UTC-0800)

Download qualification report Test suite tag Actions

Activity log details

Device: MyThing Suite version: v1 Created: December 07, 2020, 23:33:16 (UTC-0800) Status: Passed

AWS IoT Core Qualification Program

Qualification Program (6) Passed

Test	Result	System message	Logs
MQTT Connect	Passed	No issues found	<a href="#">Test case log</a>
MQTT Subscribe	Passed	No issues found	<a href="#">Test case log</a>
MQTT Publish	Passed	No issues found	<a href="#">Test case log</a>
TLS Connect	Passed	No issues found	<a href="#">Test case log</a>
TLS Unsecure Server Cert	Passed	No issues found	<a href="#">Test case log</a>
TLS Incorrect Subject Name Server Cert	Passed	No issues found	<a href="#">Test case log</a>

Tags - optional

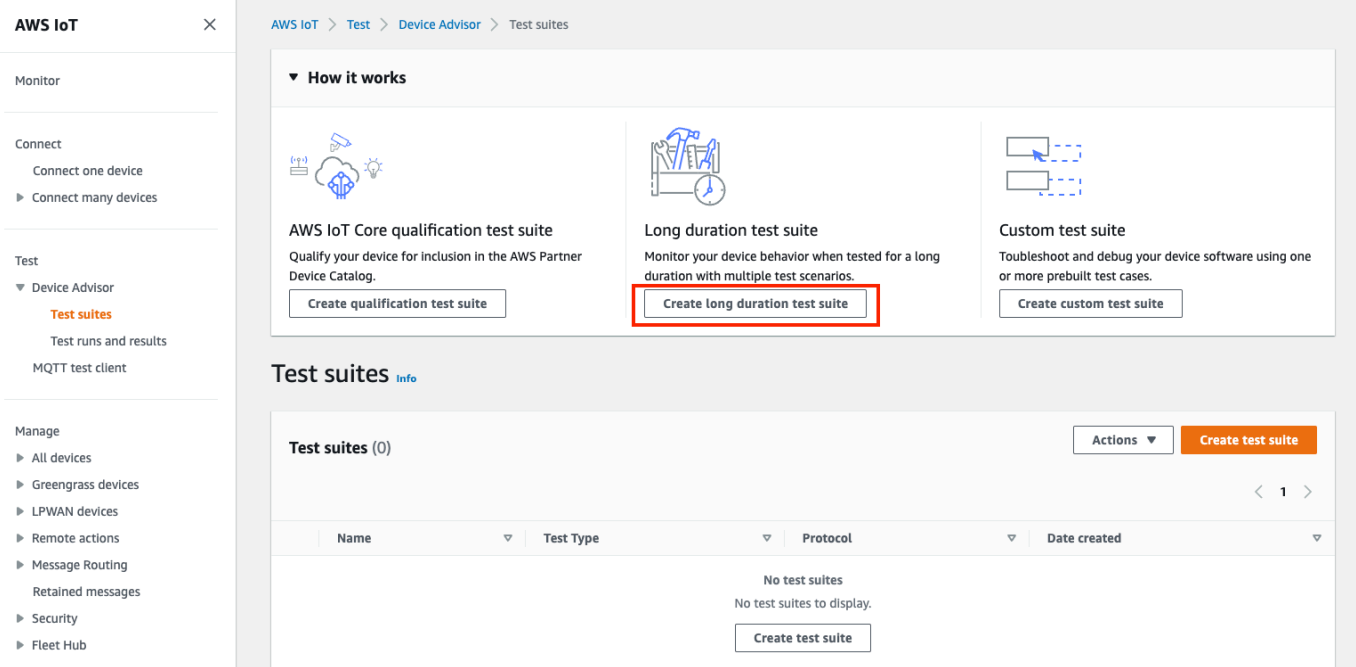
No tags associated with the resource.

Add new tag

## 長時間測試主控台工作流程

本教學課程可協助您使用主控台對 Device Advisor 進行長時間測試。若要完成教學課程，請遵循 [設定](#) 中的步驟。

1. 在 [AWS IoT 主控台](#) 的導覽窗格中，展開 Test (測試)、Device Advisor，然後選擇 Test suites (測試套件)。在頁面上，選取 Create long duration test suite (建立長時間測試套件)。



AWS IoT

Monitor

Connect

Connect one device

Connect many devices

Test

Device Advisor

Test suites

Test runs and results

MQTT test client

Manage

All devices

Greengrass devices

LPWAN devices

Remote actions

Message Routing

Retained messages

Security

Fleet Hub

AWS IoT > Test > Device Advisor > Test suites

How it works

AWS IoT Core qualification test suite

Qualify your device for inclusion in the AWS Partner Device Catalog.

Create qualification test suite

Long duration test suite

Monitor your device behavior when tested for a long duration with multiple test scenarios.

Create long duration test suite

Custom test suite

Troubleshoot and debug your device software using one or more prebuilt test cases.

Create custom test suite

Test suites

Test suites (0)

Actions Create test suite

Name	Test Type	Protocol	Date created
No test suites			
No test suites to display.			

Create test suite

2. 在 Create test suite (建立測試套件) 頁面，選取 Long duration test suite (長時間測試套件)，然後選擇 Next (下一步)。



對於通訊協定，請選擇 MQTT 3.1.1 或 MQTT 5。

The screenshot shows the AWS IoT Core console interface for creating a test suite. The left sidebar contains navigation options under 'Test', with 'Device Advisor' expanded to show 'Test suites'. The main panel is titled 'Create test suite' and shows a progress indicator for four steps. Step 1, 'Create test suite', is active. It features a 'Choose test suite type' section with three radio button options: 'AWS IoT Core qualification test suite', 'Long duration test suite' (selected), and 'Custom test suite'. Below this is a 'Protocol' section with two radio button options: 'MQTT 3.1.1' (selected) and 'MQTT 5'. At the bottom right, there are 'Cancel' and 'Next' buttons.

3. 在 Configure test suite (設定測試套件) 頁面上，執行以下操作：
  - a. 更新 Test suite name (測試套件名稱) 欄位。
  - b. 更新 Test group name (測試群組名稱) 欄位。
  - c. 選擇裝置可以執行的 Device operations (裝置操作)。此步驟將會選擇要執行的測試。
  - d. 選取 Settings (設定) 選項。

4. (選擇性) 輸入 Device Advisor 等待基本測試完成所需的時間上限。選取 Save (儲存)。

5. 在 Advanced tests (進階測試) 和 Additional settings (其他設定) 區段中執行下列動作。
- 選取或取消選取您要在此測試中執行的 Advanced tests (進階測試)。
  - Edit (編輯) 測試組態 (如果適用)。
  - 在 Additional settings (其他設定) 區段下設定 Additional execution time (其他執行時間)。
  - 選擇 Next (下一步)，並進行下一個步驟。

**Basic tests**  
All basic tests relevant to the device operations selected above will be executed.

- Connect**  
Device can connect to IoT Core
- Reconnect**  
Device can reconnect to IoT Core
- Publish**  
Device can publish to topics
- Subscribe**  
Device can subscribe to topics

**Advanced tests**  
In addition, you can select and configure any advanced tests that you would like to execute

<input checked="" type="checkbox"/>	Test case	Description	Configure
<input checked="" type="checkbox"/>	Return PUBACK on Qos1 subscription	Device can return a PUBACK message for a message published to a subscribed Qos1 topic.	-
<input checked="" type="checkbox"/>	Receive large payload	Device can receive the large payload message	Edit
<input checked="" type="checkbox"/>	Persistent session	Device can reconnect, receive stored messages and maintain a persistent session	-
<input checked="" type="checkbox"/>	Keep Alive	Device can disconnect and reconnect to keep alive	-
<input checked="" type="checkbox"/>	Intermittent connectivity	Device reconnects when disconnected at random intervals	-
<input checked="" type="checkbox"/>	Reconnect backoff	Device has a backoff mechanism when disconnected	Edit
<input checked="" type="checkbox"/>	Long server disconnect	Device reconnects when disconnected for long period	Edit

**Additional settings**  
Additional execution time - Optional  
Maximum time Device Advisor waits after completing all our test cases, before ending the test session. Enter value 0 - 120 minutes.

Cancel Previous **Next**

6. 在此步驟中選擇 **Create a new role (建立新角色)** 或 **Select an existing role (選取現有角色)**。如需詳細資訊，請參閱 [建立要用作裝置角色的 IAM 角色](#)。

**Select a device role**

**Device role info**  
AWS IoT Core Device Advisor requires permission to perform AWS IoT MQTT actions on behalf of your test device.

**Create new role**  
Create and use a new device role

**Select an existing role**  
Use an existing device role

Role name  
DeviceAdvisorServiceRole-lhqPgxBS

**Permissions**  
Choose which actions and the associated resources for AWS IoT Core Device Advisor to access using this role. You can enter a specific resource or resource prefix. To enter multiple values for a resource, use commas to separate the values. [Learn more](#)

Action	Resource type	Resource
<input checked="" type="checkbox"/> Connect	Clientid	myClientid
<input checked="" type="checkbox"/> Publish	Topic	MyTopic
<input checked="" type="checkbox"/> Subscribe	TopicFilter	MyTopic
<input type="checkbox"/> Receive	Topic	Specify topics to receive from e.g. MyTopic, MyTopic*

Cancel Previous **Next**

7. 檢閱所有在此步驟之前建立的組態，然後選取 **Create test suite (建立測試套件)**。

**AWS IoT Core**

Monitor

Connect

Connect one device

Connect many devices

Test

**Device Advisor**

Test suites

Test runs and results

MQTT test client

Manage

All devices

Greengrass devices

LPWAN devices

Remote actions

Message Routing

Retained messages

Security

Fleet Hub

Device Software

Billing groups

Settings

Learn

Feature spotlight

Documentation

AWS IoT > Test > Device Advisor > Create test suite

Step 1  
Create test suite

Step 2  
Configure test suite

Step 3  
Select a device role

Step 4  
**Review**

## Review

**Step 1: Test suite type** Edit

**Test suite type details**

Test suite type	Protocol
Long duration	MQTT 3.1.1

**Step 2: Test suite** Edit

**Test suite details**

Test suite name	Test group name
Long Duration Demo	MQTT Test Group

Device operations  
CONNECT, PUBLISH, SUBSCRIBE

**Basic tests**

All basic tests relevant to the device operations selected above will be executed.

- Connect**  
Device can connect to IoT Core
- Reconnect**  
Device can reconnect to IoT Core
- Publish**  
Device can publish to topics
- Subscribe**  
Device can subscribe to topics

**Advanced tests**

In addition, you can select and configure any advanced tests that you would like to execute

- Return PUBACK on Qos1 subscription** - Device can return a PUBACK message for a message published to a subscribed Qos1 topic.
- Receive large payload** - Device can receive the large payload message
- Persistent session** - Device can reconnect, receive stored messages and maintain a persistent session
- Keep Alive** - Device can disconnect and reconnect to keep alive
- Intermittent connectivity** - Device reconnects when disconnected at random intervals
- Reconnect backoff** - Device has a backoff mechanism when disconnected
- Long server disconnect** - Device reconnects when disconnected for long period

**Step 3: Device role** Edit

**Device role detail**

Device role type	Device role name
Select an existing role	DeviceAdvisorDUTRole

Cancel Previous **Create test suite**

8. 建立的測試套件位於 Test suites (測試套件) 區段下。選取套件以檢視詳細資訊。

Created test suite successfully.

AWS IoT > Test > Device Advisor > Test suites

**How it works**

- AWS IoT Core qualification test suite**  
Qualify your device for inclusion in the AWS Partner Device Catalog.  
[Create qualification test suite](#)
- Long duration test suite**  
Monitor your device behavior when tested for a long duration with multiple test scenarios.  
[Create long duration test suite](#)
- Custom test suite**  
Troubleshoot and debug your device software using one or more prebuilt test cases.  
[Create custom test suite](#)

**Test suites** Info

Test suites (1) Actions Create test suite

Find test suite

Name	Test Type	Protocol	Date created
Long Duration Demo	Long duration	MQTT 3.1.1	October 12, 2022, 11:10:53 (UTC-0700)

9. 要執行建立的測試套件，請選取 Actions (動作)，然後選擇 Run test suite (執行測試套件)。

AWS IoT > Test > Device Advisor > Test suites > Long Duration Demo

**Long Duration Demo**

**Test suite details**

Suite definition ARN  
arn:aws:iotdeviceadvisor:ap-northeast-1:507237901444:suitedefinition/jl7u6vutzki

Suite version  
v1

Created  
October 12, 2022, 11:10:53 (UTC-0700)

Test type  
Long duration

**Activity Log**

Timestamp	Test suite version	Status	Passed	Failed	Duration
No test suite activities					

**Test suite summary**  
A summary of the tests to be run in the test suite, organized by groups.

**Test suite details**

Test suite name  
Long Duration Demo

Test group name  
MQTT Test Group

Device operations  
CONNECT, PUBLISH, SUBSCRIBE

10. 在 Run configuration (執行組態) 頁面選擇組態選項。

- 選取要執行測試的 Things (物件) 或 Certificate (憑證)。
- 選取 Account-level endpoint (帳戶層級端點) 或 Device-level endpoint (裝置層級端點)。
- 選擇 Test (測試) 以執行測試。

**Run configuration**

**Select test devices**

Select the IoT thing/certificate to test using the test suite. If not listed below, you must first create a thing/certificate registered with IoT Core before you can run the test suite.

Things  
Choose a thing for this test suite. To create a new thing, go to [IoT Things](#).

Certificates  
Choose a certificate for this test suite. To create a new certificate, go to [IoT Certificates](#).

**Things (3)**

Filter things

Name	Type
DeviceAdvisorVirtualDevice	

**Test endpoint**

Choose the endpoint that best fits your situation. If you want to simultaneously run multiple test suites then use 'Device-level endpoint', if you want to run only one test suite at a time then choose the 'Account-level endpoint'.

Account-level endpoint  
Using this endpoint, you can only run one test suite at a time.

Device-level endpoint  
Using this endpoint, you can run multiple test suites simultaneously.

Copy and paste this endpoint to your test device.  
t3q0wka5209bwx.deviceadvisor.iot.ap-northeast-1.amazonaws.com

**Tags - optional**

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

You can add up to 50 more tags.

Cancel

11. 要查看測試套件執行的結果，請選擇左側導航窗格中的 Test runs and results (測試執行和結果)。選擇已執行的測試套件，以檢視結果的詳細資訊。

**Test runs and results**

**Summary**

Number of IoT things available	Number of IoT certificates available	Number of test suites running
3	3	1

**Results of test runs (in progress and completed)**

Name	Timestamp	Test suite version	Status	Passed	Failed	Duration
Long Duration Demo	October 12, 2022, 11:16:13 (UTC-0700)	v1	In Progress	-	-	-

12. 上一個步驟會顯示測試摘要頁面。所有關於測試執行的詳細資訊都會顯示在此頁面中。當主控台提示您開始進行裝置連線時，請將您的裝置連線至提供的端點。測試的進度會顯示在此頁面。

**Test log summary**

Timestamp	Message
October 12, 2022, 11:16:17 (UTC-0700)	Starting CONNECT scenario.
October 12, 2022, 11:16:17 (UTC-0700)	Starting PUBLISH scenario.
October 12, 2022, 11:16:17 (UTC-0700)	Starting SUBSCRIBE scenario.
No more events.	

13. 長時間測試會在側邊面板列出額外的 Test log summary (測試日誌摘要)，以近乎即時的速度顯示所有發生於裝置與代理程式之間的重要事件。要查看更深入的詳細日誌，請按一下 Test case log (測試案例日誌)。

**Test log summary**

Timestamp	Message
October 12, 2022, 11:16:17 (UTC-0700)	Starting CONNECT scenario.
October 12, 2022, 11:16:17 (UTC-0700)	Starting PUBLISH scenario.
October 12, 2022, 11:16:17 (UTC-0700)	Starting SUBSCRIBE scenario.
No more events.	

## Device Advisor VPC 端點 (AWS PrivateLink)

您可以透過建立介面 VPC 端點，在 VPC 和 AWS IoT Core Device Advisor 測試端點 (資料平面) 之間建立私人連線。在將裝置部署到生產環境 AWS IoT Core 之前，您可以使用此端點驗證 AWS IoT 裝置是否具有可靠且安全的連線能力。Device Advisor 的預先建置測試可協助依據最佳實務驗證您的裝置軟體對於 [TLS](#)、[MQTT](#)、[Device Shadow](#) 和 [AWS IoT 工作](#) 的使用。

[AWS PrivateLink](#) 為您的 IoT 裝置使用的介面端點提供動力。此服務可協助您在不使用網際網路閘道、NAT 裝置、VPC 連線或 AWS Direct Connect 連接的情況下，以私有方式存取 AWS IoT Core Device Advisor 測試端點。VPC 中傳送 TCP 和 MQTT 封包的執行個體不需要公用 IP 位址即可與 AWS IoT Core Device Advisor 測試端點通訊。您的 VPC 和 AWS IoT Core Device Advisor 不會離開 AWS 雲端之間的流量。IoT 裝置與 Device Advisor 測試案例之間的任何 TLS 和 MQTT 通訊都會保留在您 AWS 帳戶的資源中。

每個介面端點都由子網路中的一個或多個[彈性網路介面](#)表示。

如需 VPC 端點的詳細資訊，請參閱《Amazon VPC 使用者指南》中的[介面 VPC 端點 \(AWS PrivateLink\)](#)。

### AWS IoT Core Device Advisor VPC 端點的考量

設定介面 VPC 端點之前，請務必檢閱《Amazon VPC 使用者指南》中的[介面端點屬性和限制](#)。繼續之前，請考慮以下項目：

- AWS IoT Core Device Advisor 目前支援從 VPC 呼叫「裝置建議程式」測試端點 (資料平面)。訊息代理程式使用資料平面通訊來傳送和接收資料。它藉由 TLS 和 MQTT 封包的協助做到這一點。將您的裝置 AWS IoT Core Device Advisor 連線至「AWS IoT 裝置建議程式」測試端點的 VPC 端點。此 VPC 端點不會使用[控制平面 API 動作](#)。若要建立或執行測試套件或其他控制平面 API，請透過公用網際網路使用主控台、AWS SDK 或 AWS 命令列介面。
- 下列項目 AWS 區域 支援 VPC 端點：AWS IoT Core Device Advisor
  - 美國東部 (維吉尼亞北部)
  - 美國西部 (奧勒岡)
  - 亞太區域 (東京)
  - 歐洲 (愛爾蘭)
- Device Advisor 支援具有 X.509 用戶端憑證和 RSA 伺服器的 MQTT。
- 目前不支援 [VPC 端點政策](#)。



- 如需如何[建立資源](#) (連接 VPC 端點) 的指示，請檢查 VPC 端點的[必要條件](#)。您必須建立 VPC 和私有子網路才能使用 AWS IoT Core Device Advisor VPC 端點。
- 您的 AWS PrivateLink 資源有配額。如需詳細資訊，請參閱 [AWS PrivateLink 配額](#)。
- VPC 端點僅支援 IPv4 流量。

## 為 AWS IoT Core Device Advisor 建立界面 VPC 端點

若要開始使用 VPC 端點，請[建立界面 VPC 端點](#)。接下來，選擇 AWS IoT Core Device Advisor 作為 AWS 服務。如果您正在使用 AWS CLI，請撥[describe-vpc-endpoint-services](#) 打電話以確認 AWS IoT Core Device Advisor 認存在於您的 AWS 區域。確認連接到端點的安全群組允許 MQTT 和 TLS 流量進行 [TCP 協定通訊](#)。例如，在美國東部 (維吉尼亞北部) 區域，使用下列命令：

```
aws ec2 describe-vpc-endpoint-services --service-name com.amazonaws.us-east-1.deviceadvisor.iot
```

您可以使用下列服務名稱建立 AWS IoT Core 立 VPC 端點：

- com.amazonaws.region.deviceadvisor.iot

根據預設，端點的私有 DNS 為開啟。這可確保使用預設測試端點會保留在您的私有子網路中。若要取得您的帳戶或裝置層級端點，請使用主控台 AWS CLI 或 AWS SDK。例如，如果您在公有子網路或公用網際網路上執行 [get-point](#)，您可以取得端點並使用它來連線至 Device Advisor。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[透過介面端點存取服務](#)。

若要將 MQTT 用戶端連線到 VPC 端點介面，AWS PrivateLink 服務會在連接到 VPC 的私有託管區域中建立 DNS 記錄。這些 DNS 記錄會將 AWS IoT 裝置的要求導向至 VPC 端點。

## 控制對 VPC 端點 AWS IoT Core Device Advisor 的存取

您可以使用 VPC [條件內容](#) 金鑰來限制裝置存取，AWS IoT Core Device Advisor 並僅允許透過 VPC 端點進行存取。AWS IoT Core 支援下列 VPC 相關內容金鑰：

- [SourceVpc](#)
- [SourceVpce](#)
- [VPCSourceIp](#)

**Note**

AWS IoT Core Device Advisor 目前不支援 [VPC 端點原則](#)。

下列原則授與使用符合物名稱的 AWS IoT Core Device Advisor 用戶端 ID 連線的權限。它也會發佈至以物件名稱為前置詞的任何主題。該政策取決於連接到具有特定 VPC 端點 ID 的 VPC 端點的裝置。此政策會拒絕與公有 AWS IoT Core Device Advisor 測試端點的連線嘗試。

```
{
  "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 測試案例

Device Advisor 提供六種類別的預先建置測試。

- [TLS](#)
- [MQTT](#)
- [影子](#)
- [任務執行](#)
- [許可和政策](#)
- [長時間測試](#)

裝置顧問測試案例，以符合「AWS 裝置資格認證計劃」資格。

根據 [AWS 裝置資格計畫](#)，您的裝置必須通過下列測試才能符合資格。

### Note

這是資格測試修訂清單。

- [TLS Connect](#) (TLS 連接) (「TLS 連接」)
- [TLS Incorrect Subject Name Server Cert](#) (TLS 不正確的主體名稱伺服器憑證) (「不正確的主體通用名稱 (CN) / 主體別名 (SAN)」)
- [TLS Unsecure Server Cert](#) (TLS 不安全的伺服器憑證) (「未由認可的 CA 簽署」)
- 加密套件的 [TLS 裝置 Support](#) (「[AWS IoT 建議 AWS IoT 加密套件](#)的 TLS 裝置 Support」)
- [TLS 接收最大大小的片段](#) (「TLS 接收最大大小的片段」)
- [TLS 過期伺服器憑證](#) (「過期伺服器憑證」)
- [TLS 大型伺服器憑證](#) (「TLS 大型伺服器憑證」)
- [MQTT Connect](#) (「設備發送連接到 AWS IoT Core (快樂案例)」)
- [MQTT Subscribe](#) (MQTT 訂閱) (「可訂閱 (Happy 案例)」)
- [MQTT Publish](#) (MQTT 發佈) (「QOS0 (Happy 案例)」)
- [MQTT 連線抖動重試](#) (「裝置在抖動退避的情況下重試連接：沒有 CONNACK 回應」)

## TLS

使用這些測試來判斷裝置之間的傳輸層安全性通訊協定 (TLS) AWS IoT 是否安全。

### Note

Device Advisor 現在支援 TLS 1.3。

## Happy Path

### TLS Connect

驗證被測設備是否可以完成 TLS 握手。AWS IoT 此測試不會驗證用戶端裝置的 MQTT 實作。

Example API 測試案例定義：

### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。為了獲得最佳結果，我們建議使用 2 分鐘的逾時值。

```
"tests":[
  {
    "name":"my_tls_connect_test",
    "configuration": {
      // optional:
      "EXECUTION_TIMEOUT":"300", //in seconds
    },
    "test":{
      "id":"TLS_Connect",
      "version":"0.0.0"
    }
  }
]
```

Example 測試案例輸出：

- 通過 — 被測裝置已完成 TLS 交涉 AWS IoT。

- 通過但有警告 — 被測裝置已完成 TLS 交涉 AWS IoT，但裝置或有 TLS 警告訊息 AWS IoT。
- 失敗 — 被測裝置因交握錯誤 AWS IoT 而無法完成 TLS 交握。

## TLS 接收最大大小的片段

此測試案例驗證您的裝置是否可以接收和處理 TLS 最大大小的片段。您的測試裝置必須訂閱 QoS 1 的預先設定主題，才能接收大型承載。您可以使用組態 `${payload}` 自訂承載。

Example API 測試案例定義：

### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。為了獲得最佳結果，我們建議使用 2 分鐘的逾時值。

```
"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"
    }
  }
]
```

## 密碼套件

### AWS IoT 建議加密套件的 TLS 裝置 Support

驗證來自待測裝置的 TLS 用戶端 Hello 訊息中的密碼套件是否包含建議的 [AWS IoT 密碼套件](#)。它提供了有關設備支持的加密套件的更多見解。

## Example API 測試案例定義：

### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"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 測試案例輸出：

- 通過 — 正在測試的設備密碼套件包含至少一個推薦的 AWS IoT 密碼套件，並且不包含任何不支持的加密套件。
- 通過但有警告 - 裝置密碼套件至少包含其中一個 AWS IoT 加密套件，但是：
  1. 它不包含任何建議的密碼套件
  2. 它包含不支援的密碼套件。AWS IoT

我們建議您驗證任何不支援的密碼套件是否安全。

- 失敗 — 正在測試的設備密碼套件不包含任何 AWS IoT 支持的加密套件。

## 較大型的伺服器憑證

### TLS 大型伺服器憑證

接收和處理較大型伺服器憑證時，驗證您的裝置是否可以與 AWS IoT 完成 TLS 交握。此測試使用的伺服器憑證大小 (以位元組為單位) 大於 TLS Connect 測試案例和 IoT Core 目前使用的大小 20

在此測試案例期間，AWS IoT 測試裝置的 TLS 緩衝區空間如果緩衝區空間足夠大，則 TLS 握手不會發生錯誤。此測試不會驗證裝置的 MQTT 實作。TLS 交握程序完成後，測試案例即會停止。

Example API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。為了獲得最佳結果，我們建議使用 2 分鐘的逾時值。如果此測試案例失敗，但 TLS Connect 測試案例通過，我們建議您增加裝置的 TLS 緩衝區空間限制。增加緩衝區空間限制，確保您的裝置可以在大小增加的情況下處理更大的伺服器憑證。

```
"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 測試案例輸出：


- 通過：待測裝置已完成與 AWS IoT 的 TLS 交握。
- 通過但有警告 — 被測裝置已完成 TLS 交握 AWS IoT，但是裝置或有 TLS 警告訊息 AWS IoT。
- 失敗 — 被測裝置無法完成 TLS 交握，AWS IoT 因為握手程序期間發生錯誤。

## TLS 不安全伺服器憑證

### 未由認可的 CA 簽署

驗證若提供的伺服器憑證沒有來自 ATS CA 的有效簽章，待測裝置是否會關閉連線。裝置應該只會連接到提供有效憑證的端點。

## Example API 測試案例定義：

 Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"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 測試案例輸出：

- 通過：待測裝置已關閉連線。
- 失敗 — 被測裝置已完成 TLS 交握 AWS IoT。

## TLS 不正確的主體名稱伺服器憑證 / 不正確的主體通用名稱 (CN) / 主體別名 (SAN)

驗證若為網域名稱提供的伺服器憑證不同於所請求的伺服器憑證，測試下的裝置是否會關閉連線。

## Example API 測試案例定義：

 Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"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 測試案例輸出：

- 通過：待測裝置已關閉連線。
- 失敗 — 被測裝置已完成 TLS 交握 AWS IoT。

## TLS 過期伺服器憑證

### 過期的伺服器憑證

驗證如果所提供伺服器憑證已過期，測試下的裝置是否會關閉連線。

Example API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"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 測試案例輸出：

- 通過 — 被測裝置拒絕完成 TLS 握手 AWS IoT。裝置會在關閉連線前傳送 TLS 警示訊息。
- 通過但有警告 - 待測裝置拒絕完成與 AWS IoT 的 TLS 交握。不過，它不會在關閉連線之前傳送 TLS 警示訊息。
- 失敗 — 被測裝置完成 TLS 交涉 AWS IoT。

## MQTT

### CONNECT、DISCONNECT 和 RECONNECT

「設備發送連接到 AWS IoT Core (快樂案例)」

驗證測試下的裝置是否傳送 CONNECT 請求。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"tests":[
  {
    "name":"my_mqtt_connect_test",
    "configuration": {
      // optional:
      "EXECUTION_TIMEOUT":"300", // in seconds
    },
    "test":{
      "id":"MQTT_Connect",
      "version":"0.0.0"
    }
  }
]
```

「裝置可將 PUBACK 傳回到 QoS1 的任意主題」

該測試用例將檢查設備 (客戶端) 是否可以返回 PUBACK 消息，如果在訂閱具有 QoS1 的主題後收到代理發布消息。

此測試案例的承載內容和承載大小是可設定的。如果已設定承載大小，Device Advisor 會覆寫承載內容的值，並將預先定義的承載傳送至具有所需大小的裝置。承載大小是介於 0 到 128 之間的值，且不能超過 128 KB。AWS IoT Core 會拒絕發佈和連接大於 128 KB 的請求，如 [AWS IoT Core 訊息代理程式和通訊協定限制和配額](#) 頁面所示。

API 測試案例定義：

**Note**

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。PAYLOAD\_SIZE 可以設定為介於 0 到 128 KB 之間的值。定義承載大小會覆寫承載內容，因為 Device Advisor 會將具有指定大小的預先定義承載傳送回裝置。

```
"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"
    }
  }
]
```


「裝置在抖動退避的情況下重試連接：沒有 CONNACK 回應」

驗證測試下的裝置在與代理程式重新連線至少五次時，是否使用適當的抖動退避。代理程式記錄測試下裝置的 CONNECT 請求的時間戳記、執行封包驗證、暫停而不傳送 CONNACK 到測試下的裝置，以及等待測試下的裝置重新傳送請求。允許第六次連線嘗試通過，並允許 CONNACK 回流到測試下的裝置。

再次執行上述程序。此測試案例需要裝置總共連接至少 12 次。收集的時間戳記用來驗證測試下的裝置是否使用抖動退避。如果待測裝置具有嚴格的指數退避延遲，則此測試案例會通過但有警告。

建議對待測裝置實作 [指數退避和抖動](#) 機制，以通過此測試案例。

## API 測試案例定義：

 Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 4 分鐘。


```
"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"
    }
  }
]
```

## 「裝置在指數退避的情況下重試連接：沒有 CONNACK 回應」

驗證測試下的裝置在與代理程式重新連線至少五次時，是否使用適當的指數退避。代理程式記錄測試下裝置的 CONNECT 請求的時間戳記、執行封包驗證、暫停而不傳送 CONNACK 到用戶端裝置，以及等待測試下的裝置重新傳送請求。收集的時間戳記用來驗證待測裝置是否使用指數退避。

建議對待測裝置實作[指數退避和抖動](#)機制，以通過此測試案例。

## API 測試案例定義：

 Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 4 分鐘。

```
"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"
  }
}
]

```

「裝置在抖動退避的情況下重新連線：在伺服器中斷連線後」

驗證待測裝置在與伺服器中斷連線後重新連接時，是否使用必要的抖動和退避。Device Advisor 將裝置與伺服器的連線中斷至少五次，並觀察裝置在 MQTT 重新連線時的行為。Device Advisor 記錄待測裝置的 CONNECT 請求的時間戳記、執行封包驗證、暫停而不將 CONNACK 傳送到用戶端裝置，以及等待待測裝置重新傳送請求。收集的時間戳記用來驗證在重新連接時，待測裝置是否使用抖動和退避。如果待測裝置具有嚴格的指數退避，或未實作適當的抖動退避機制，則此測試案例會通過但有警告。如果待測裝置已實作線性退避或常數退避機制，則測試會失敗。

若要通過這個測試案例，建議對待測裝置實作[指數退避和抖動](#)機制。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 4 分鐘。

可以透過指定 RECONNECTION\_ATTEMPTS 來變更驗證退避的重新連線嘗試次數。號碼必須為介於 5 到 10 之間的數字。預設值為 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"
    }
  }
]

```

```

    }
  }
]

```

### 「裝置在抖動退避的情況下重新連線 - 連線不穩定」

驗證待測裝置在連線不穩定時，是否使用必要的抖動和退避。Device Advisor 會在 5 次成功連線後中斷裝置與伺服器的連線，並觀察裝置在 MQTT 重新連線時的行為。Device Advisor 會記錄待測裝置的 CONNECT 請求的時間戳記、執行封包驗證、傳送回 CONNACK、中斷連線、記錄中斷連線的時間戳記，以及等待待測裝置重新傳送請求。收集的時間戳記用來驗證在成功但不穩定的連線後重新連線時，待測裝置是否使用抖動和退避。如果待測裝置具有嚴格的指數退避，或未實作適當的抖動退避機制，則此測試案例會通過但有警告。如果待測裝置已實作線性退避或常數退避機制，則測試會失敗。

若要通過這個測試案例，建議對待測裝置實作[指數退避和抖動](#)機制。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 4 分鐘。

可以透過指定 RECONNECTION\_ATTEMPTS 來變更驗證退避的重新連線嘗試次數。號碼必須為介於 5 到 10 之間的數字。預設值為 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"
    }
  }
]

```

## 發佈

### 「QoS0 (Happy 案例)」

驗證被測設備是否使用 QoS0 或 QoS1 發布消息。您也可以在測試設定中設定主題值和承載，來驗證訊息和承載的主題。

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"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 發佈重試：沒有 PUBACK」

驗證若代理程式未傳送 PUBACK，測試下的裝置是否重新發佈與 QoS1 一起傳送的訊息。您也可以在測試設定中指定此主題，來驗證訊息的主題。用戶端裝置在重新發佈訊息之前不得中斷連線。此測試也會驗證重新發佈的訊息是否具有與原始訊息相同的封包識別符。測試執行期間，如果裝置失去連線並重新連接，則測試案例將重置而不會故障，並且裝置必須再次執行測試案例步驟。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。建議至少 4 分鐘。

```
"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"
    }
  }
]
```

### 「發佈保留的訊息」

驗證受測裝置是否會發佈 retainFlag 設定為 true 的訊息。您可以在測試設定中設定主題值和承載，來驗證訊息和承載的主題。如果在 PUBLISH 封包中傳送的 retainFlag 未設定為 true，測試案例將會失敗。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。若要執行此測試案例，請在 [device role](#) (裝置角色) 中新增 iot:RetainPublish 動作。

```
"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"
    }
  }
]
```

### 「以使用者屬性進行發佈」

驗證受測裝置是否發佈含有正確使用者屬性的訊息。您可以在測試設定中設定名稱/值對來驗證使用者屬性。如果未提供使用者屬性或不相符，測試案例將會失敗。

API 測試案例定義：

#### Note

這是僅限 MQTT5 的測試案例。

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"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"
    }
  }
]
```

## 訂閱

### 「可以訂閱 (Happy 案例)」

驗證測試下的裝置是否訂閱 MQTT 主題。您也可以在此測試設定中指定此主題，來驗證測試下裝置所訂閱的主題。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"tests":[
  {
    "name":"my_mqtt_subscribe_test",
    "configuration":{
      // optional:
      "EXECUTION_TIMEOUT":"300", // in seconds
      "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
      ["my_TOPIC_FOR_PUBLISH_VALIDATION_a", "my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
    },
    "test":{
      "id":"MQTT_Subscribe",
      "version":"0.0.0"
    }
  }
]
```

### 「訂閱重試：沒有 SUBACK」

驗證測試下的裝置是否重試失敗的 MQTT 主題訂閱。伺服器接著會等待，而且不會傳送 SUBACK。如果用戶端裝置沒有重試訂閱，測試會失敗。用戶端裝置必須使用相同的封包 ID 重試失敗的訂閱。您也可以在此測試設定中指定此主題，來驗證測試下裝置所訂閱的主題。測試執行期間，如果裝置失去連線並重新連接，則測試案例將重置而不會故障，並且裝置必須再次執行測試案例步驟。

API 測試案例定義：

**Note**

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 4 分鐘。

```
"tests":[
  {
    "name":"my_mqtt_subscribe_retry_test",
    "configuration":{
      "EXECUTION_TIMEOUT":"300", // in seconds
      // optional:
      "TOPIC_LIST_FOR_SUBSCRIPTION_VALIDATION":
      ["my_TOPIC_FOR_PUBLISH_VALIDATION_a", "my_TOPIC_FOR_PUBLISH_VALIDATION_b"]
    },
    "test":{
      "id":"MQTT_Subscribe_Retry_No_Suback",
      "version":"0.0.0"
    }
  }
]
```

## Keep-Alive

### 「Mqtt 沒有確認」 PingResp

這個測試案例會驗證，如果待測裝置在未收到 Ping 回應時是否會中斷連線。在此測試案例中，「裝置建議程式」會封鎖 AWS IoT Core 針對發佈、訂閱和 Ping 要求傳送的回應。其也會驗證待測裝置是否中斷與 MQTT 的連線。

API 測試案例定義：

**Note**

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。建議使用的逾時值大於 keepAliveTime 值的 1.5 倍。  
此測試的最keepAliveTime大值不得超過 230 秒。

```
"tests":[
```

```
{
  "name": "Mqtt No Ack PingResp",
  "configuration":
    //optional:
    "EXECUTION_TIMEOUT": "306", // in seconds
  },
  "test": {
    "id": "MQTT_No_Ack_PingResp",
    "version": "0.0.0"
  }
}
```

## 持久性工作階段

### 「持久性階段作業 (Happy 案例)」

此測試案例會驗證裝置與持久性階段作業中斷連結時的行為。此測試案例會檢查裝置是否可以重新連接、恢復其觸發程序主題訂閱而不需明確重新訂閱、接收儲存於主題中的訊息，以及在持久性階段作業期間如預期運作。當此測試用例通過時，表明客戶端設備能夠以預期的方式與 AWS IoT Core 代理維護持久性會話。如需 AWS IoT 持續工作階段的詳細資訊，請參閱[使用 MQTT 持續性工作階段](#)。

在此測試案例中，用戶端裝置應在乾淨的階段作業旗標設為 false 下與 AWS IoT Core 連結，並且訂閱主題篩選條件。成功訂閱後，裝置建議程式將中斷 AWS IoT Core 裝置的連線。當裝置處於中斷連結的狀態時，QoS 1 訊息承載將儲存在該主題中。Device Advisor 將允許用戶端裝置與測試端點重新連結。此時由於有持久性工作階段，用戶端裝置應恢復其主題訂閱，而不需傳送任何額外的訂閱封包，並從代理程式接收 QoS 1 訊息。重新連結後，如果用戶端裝置傳送額外的 SUBSCRIBE 封包，再次重新訂閱其觸發程序主題，和/或如果用戶端並未收到來自觸發程序主題的儲存訊息，測試案例將會失敗。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為至少 4 分鐘。第一次連線時，用戶端裝置需要明確訂閱之前沒有訂閱的 TRIGGER\_TOPIC。若要通過測試案例，用戶端裝置必須使用 QoS 1 成功訂閱 TRIGGER\_TOPIC。重新連結後，用戶端裝置應了解有

作用中的持久性工作階段；因此應接受觸發程序主題傳送的已儲存訊息，並針對該特定訊息傳回 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"
    }
  }
]
```

#### 「持久性工作階段到期 - 工作階段到期」

此測試案例有助於在已中斷連線的裝置重新連線到過期的持久性工作階段時，驗證裝置行為。工作階段過期後，我們希望裝置透過明確傳送新的 SUBSCRIBE 封包來重新訂閱之前訂閱的主題。

在第一次連線期間，我們預期測試裝置與 AWS IoT 代理程式連線，因為其 CleanSession 旗標設定為 false 以啟動持續性工作階段。然後，裝置應訂閱觸發程序主題。接著，裝 AWS IoT Core 置建議程式會在成功訂閱並啟動持續工作階段後，裝置建議程式中斷連線。中斷連線之後，「AWS IoT Core 裝置建議程式」會允許測試裝置與測試端點重新連線。此時，當測試裝置傳送另一個 CONNECT 封包時，「AWS IoT Core 裝置建議程式」會傳回 CONNACK 封包，指出持續性工作階段已過期。測試裝置需要正確解譯此封包，並且在持久性工作階段終止時，應再次重新訂閱相同的觸發程序主題。如果測試裝置未重新訂閱其主題觸發程序，則測試案例失敗。為了通過測試，裝置需要了解持久性工作階段已經結束，並在第二次連線對同一個觸發程序主題回傳一個新的 SUBSCRIBE 封包。

如果此測試案例對於某測試裝置通過，表示該裝置能在持久性工作階段過期時以預期的方式處理重新連線。

API 測試案例定義：

**Note**

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為至少 4 分鐘。測試裝置需要明確訂閱之前沒有訂閱的 TRIGGER\_TOPIC。要通過測試案例，測試裝置必須在 CleanSession 旗標設為 false 下傳送 CONNECT 封包，並成功訂閱具有 QoS 1 的觸發程序主題。成功連線之後，「AWS IoT Core 裝置建議程式」會中斷裝置的連線。中斷連線之後，「AWS IoT Core 裝置建議程式」會允許裝置重新連線，而且裝置預期會重新訂閱相同的訂閱，TRIGGER\_TOPIC 因為「AWS IoT Core 裝置建議程式」會終止持續性工作階段。

```
"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

使用這些測試來驗證正在測試的設備正確使用 Dev AWS IoT Core Shadow 服務。如需詳細資訊，請參閱 [AWS IoT Device Shadow 服務](#)。如果這些測試案例是在您的測試套件中設定，則在啟動套件執行時需要提供一個物件。

目前不支援 MQTT 超過 WebSocket。

## 發佈

「裝置在連接後發佈狀態 (Happy 案例)」

驗證設備在連接到後是否可以發布其狀態 AWS IoT Core

API 測試案例定義：

### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"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"
    }
  }
]
```

可以提供 REPORTED\_STATE，以在您的裝置連接之後，對其確切的影子狀態進行額外驗證。根據預設，此測試案例會驗證您的裝置發佈狀態。

如果未提供 *SHADOW\_NAME*，則測試案例會依預設尋找發佈至未具名 (傳統) 影子類型之主題字首的訊息。如果您的裝置使用具名影子類型，請提供影子名稱。如需詳細資訊，請參閱[在裝置中使用影子](#)。

## 更新

### 「裝置將回報狀態更新為所需狀態 (Happy 案例)」

驗證您的裝置是否讀取所有收到的更新訊息，並同步處理裝置的狀態以符合所需的狀態屬性。您的裝置應該在同步處理之後發佈其最新回報狀態。如果您的裝置在執行測試之前已有現有影子，請確定針對測試案例設定的所需狀態，以及現有回報狀態並未符合。您可以查看陰影文件中的ClientToken欄位，以識別裝置建議程式傳送的陰影更新訊息DeviceAdvisorShadowTestCaseSetup。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 2 分鐘。

```
"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 應該至少有一個屬性和相關聯的值。

如果未提供 SHADOW\_NAME，則測試案例會依預設尋找發佈至未具名 (傳統) 影子類型之主題字首的訊息。如果您的裝置使用具名影子類型，請提供影子名稱。如需詳細資訊，請參閱[在裝置中使用影子](#)。



## 任務執行

### 「裝置可以完成任務執行」

此測試用例可幫助您驗證設備是否能夠使用 AWS IoT 作業接收更新，並發布成功更新的狀態。如需 AWS IoT 工作的詳細資訊，請參閱[工作](#)。

要成功運行此測試用例，需要授予[設備角色](#)兩個保留 AWS 主題。若要訂閱作業活動相關的訊息，請使用 `notify` 和 `notify-next` 主題。您的裝置角色必須將 PUBLISH 動作授予下列主題：

- `$aws/things/thingName/jobs/jobId/get`
- `$aws/things/thingName/jobs/jobId/update`

建議您將 SUBSCRIBE 和 RECEIVE 動作授予下列主題：

- `$aws/things/thingName/jobs/get/accepted`
- `$aws/things/thingName/jobs/jobId/get/rejected`
- `$aws/things/thingName/jobs/jobId/update/accepted`
- `$aws/things/thingName/jobs/jobId/update/rejected`

建議您將 SUBSCRIBE 動作授予下列主題：

- `$aws/things/thingName/jobs/notify-next`

如需有關這些保留主題的詳細資訊，請參閱[AWS IoT Job](#) 的保留主題。

目前不支援 MQTT 超過 WebSocket。

API 測試案例定義：

#### Note

EXECUTION\_TIMEOUT 的預設值為 5 分鐘。我們建議的逾時值為 3 分鐘。根據提供的 AWS IoT Job 文件或來源，調整逾時值 (例如，如果工作需要很長時間才能執行，請為測試案例定義較長的逾時值)。要運行測試，需要有效的 AWS IoT Job 文檔或已經存在的作業 ID。AWS IoT Job 文件可以以 JSON 文件或 S3 連結的形式提供。如果提供 Job 文件，則可選擇性提供作業 ID。如果提供 Job ID，則「裝置建議程式」會在代表您建立 AWS IoT 工作時使用該 ID。如果未提供 Job 文件，您可以提供與執行測試案例位於相同區域的現有 ID。在此情況下，「裝置建議程式」會在執行測試案例時使用該 AWS IoT Job。

```

"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"
    }
  }
]

```

如需建立和使用任務文件的詳細資訊，請參閱[任務文件](#)。

## 許可和政策

您可以使用這些測試，來判斷與裝置憑證相連的政策是否遵循標準最佳實務。

目前不支援 MQTT 超過 WebSocket。

「裝置憑證連接政策不包含萬用字元」

驗證與裝置相關聯的許可政策是否遵循最佳實務，且未授與裝置超過所需的許可。

API 測試案例定義：

### Note

EXECUTION\_TIMEOUT 的預設值為 1 分鐘。建議設定至少 30 秒的逾時時間。

```
"tests":[
  {
    "name": "my_security_device_policies",
    "configuration": {
      // optional:
      "EXECUTION_TIMEOUT": "60" // in seconds
    },
    "test": {
      "id": "Security_Device_Policies",
      "version": "0.0.0"
    }
  }
]
```

## 長時間測試

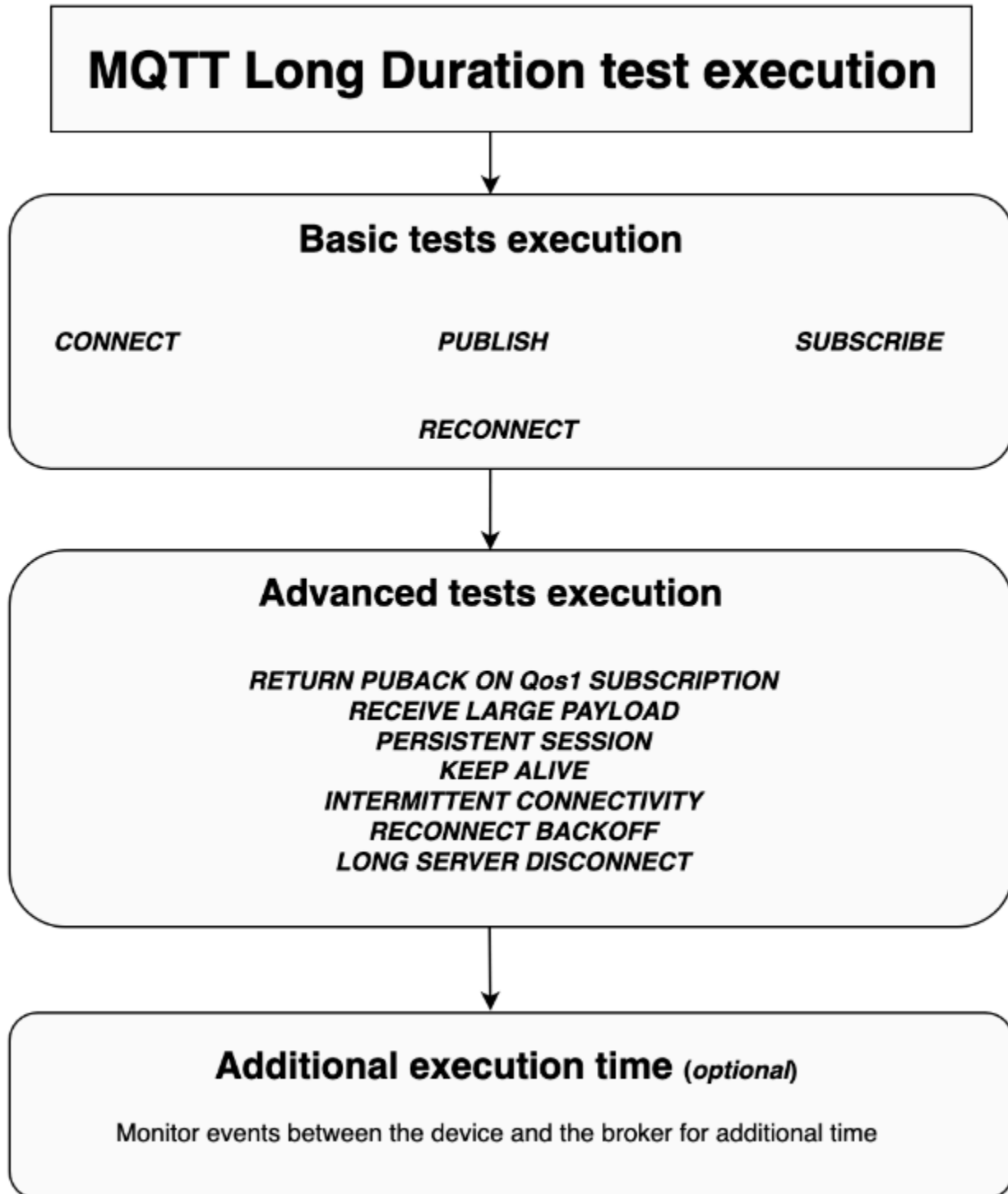
長時間測試是一款新型測試套件，可監測裝置在長時間運作期間的行為。相較於執行著重於裝置特定行為的個別測試，長時間測試會檢查裝置在其使用壽命內各種實際案例中的行為。Device Advisor 會以最有效率的順序來協調測試。該測試會產生結果和日誌，其中包括一份摘要日誌，詳載有關裝置效能的實用指標。

## MQTT 長時間測試案例

在 MQTT 長時間測試案例中，起初會以 MQTT Connect、訂閱、發佈和重新連線等簡易案例的情境來觀察裝置行為。然後，在多種複雜的故障情境下觀察裝置，例如 MQTT 重新連線輪詢、長時間伺服器中斷連線以及間歇性連線。

### MQTT 長時間測試案例執行流程

MQTT 長時間測試案例的執行分為三個階段：



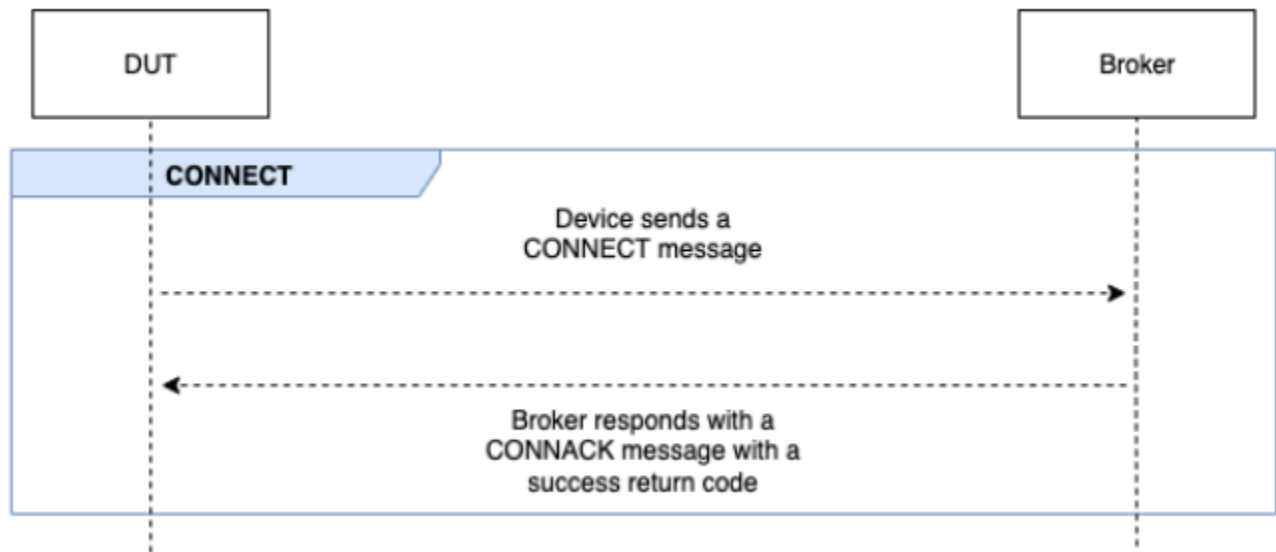
## 基本測試執行

此階段的測試案例會以並行方式執行簡易測試。測試將驗證裝置是否依據在組態中所做的選擇進行操作。

根據所選操作，一組基本測試可以包括以下內容：

### CONNECT

此案例會驗證裝置是否能夠與代理程式成功建立連線。

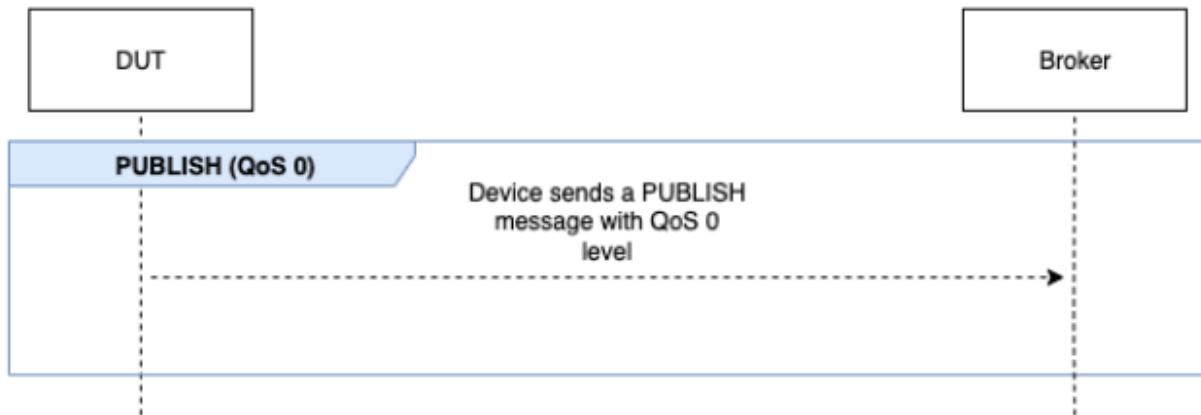


### 發佈

此案例會驗證裝置是否成功針對代理程式進行發佈。

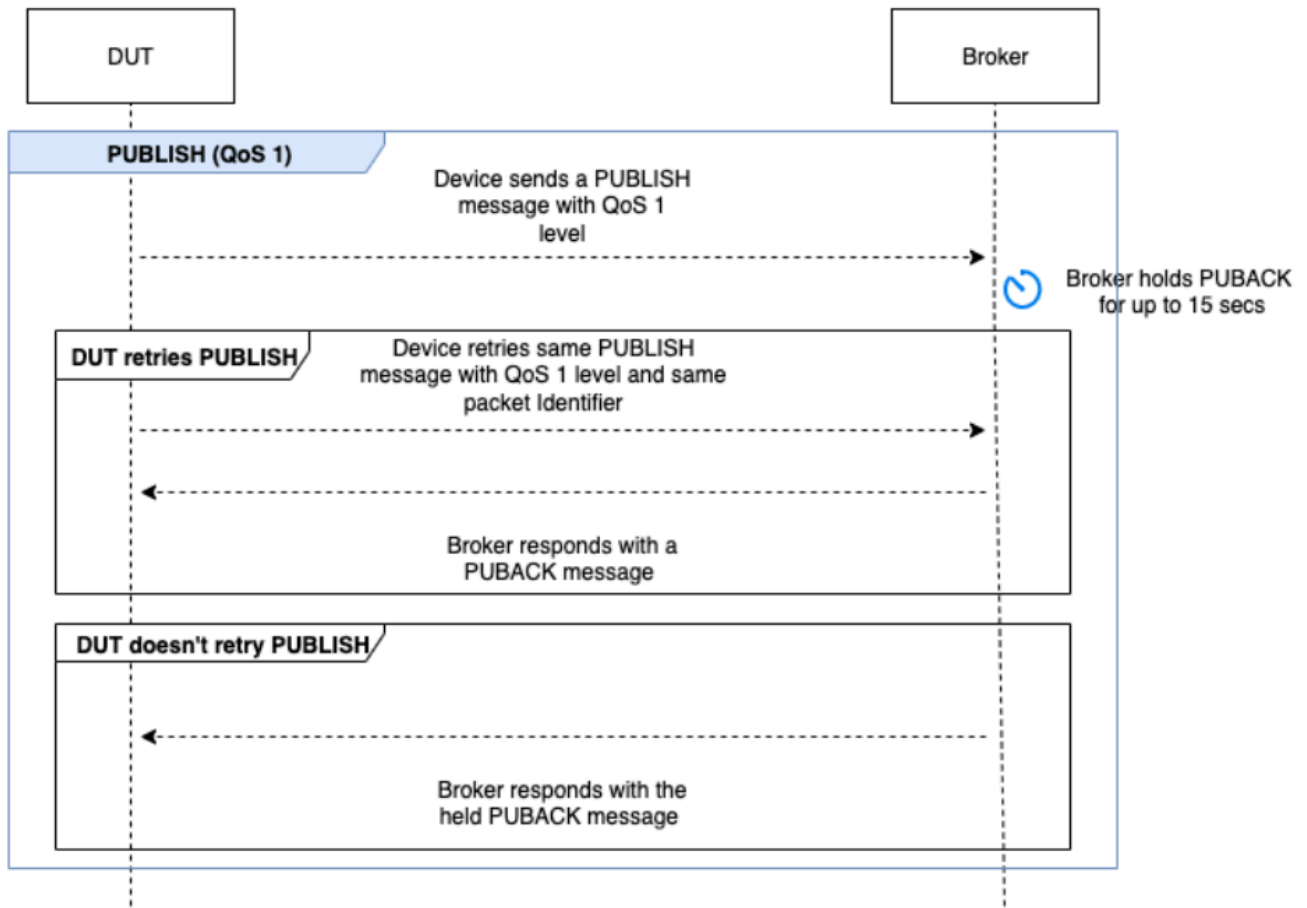
### QoS 0

此測試案例會驗證裝置在以 QoS 0 發佈期間是否成功將 PUBLISH 訊息傳送至代理程式。測試不會等待裝置接收關於 PUBACK 的訊息。



## QoS 1

在此測試案例中，裝置預計將向具有 QoS 1 的代理程式傳送兩條 PUBLISH 訊息。在第一條 PUBLISH 訊息之後，代理程式會等待最多 15 秒，然後再回應。裝置必須在 15 秒的時段中重新嘗試具有相同封包識別碼的原始 PUBLISH 訊息。在此情況下，代理程式會以 PUBACK 訊息回應，且測試會進行驗證。如果裝置沒有重試 PUBLISH，則原始 PUBACK 會傳送到裝置，並將測試標記為附帶警告的通過，同時提供系統訊息。測試執行期間，如果裝置失去連線並重新連接，則測試案例將重置而不會故障，並且裝置必須再次執行測試案例步驟。



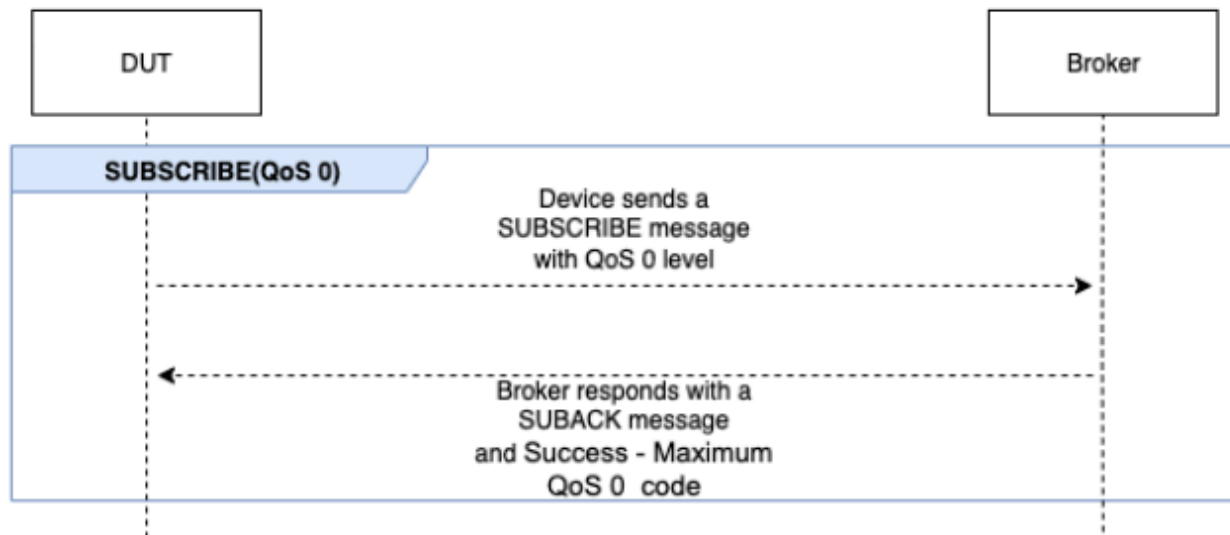
### 訂閱

此案例會驗證裝置是否成功針對代理程式進行訂閱。

### QoS 0

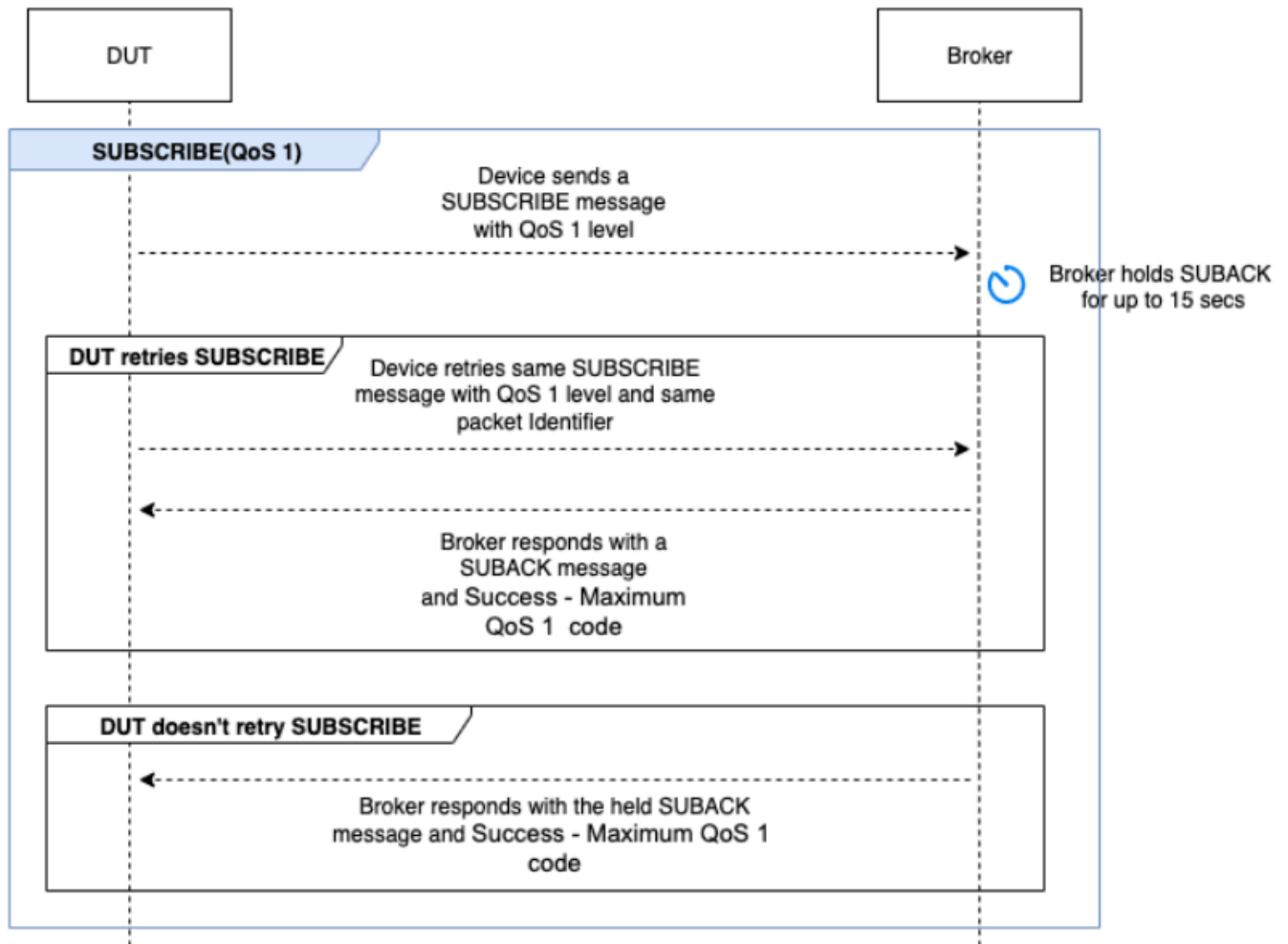
此測試案例會驗證裝置在以 QoS 0 訂閱期間是否成功將 SUBSCRIBE 訊息傳送至代理程式。測試不會等待裝置收到 SUBACK 訊息。





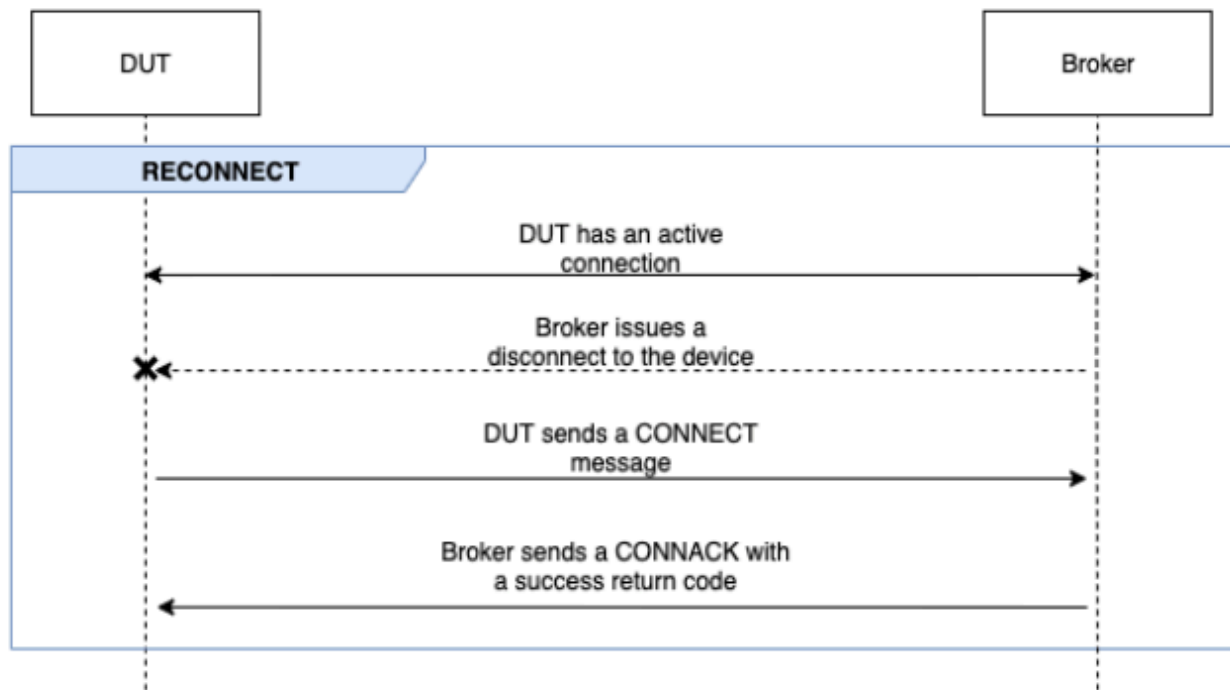
## QoS 1

在此測試案例中，裝置預計將向具有 QoS 1 的代理程式傳送兩條 SUBSCRIBE 訊息。在第一條 SUBSCRIBE 訊息之後，代理程式會等待最多 15 秒，然後再回應。裝置必須在 15 秒的時段中重新嘗試具有相同封包識別碼的原始 SUBSCRIBE 訊息。在此情況下，代理程式會以 SUBACK 訊息回應，且測試會進行驗證。如果裝置沒有重試 SUBSCRIBE，則原始 SUBACK 會傳送到裝置，並將測試標記為附帶警告的通過，同時提供系統訊息。測試執行期間，如果裝置失去連線並重新連接，則測試案例將重置而不會故障，並且裝置必須再次執行測試案例步驟。



## 重新連線

此案例會驗證裝置在從成功的連線中斷之後，是否成功與代理程式重新連線。如果先前已在測試套件期間連線多次，則 Device Advisor 不會中斷裝置連線。相反，它會將測試標記為通過。



### 進階測試執行

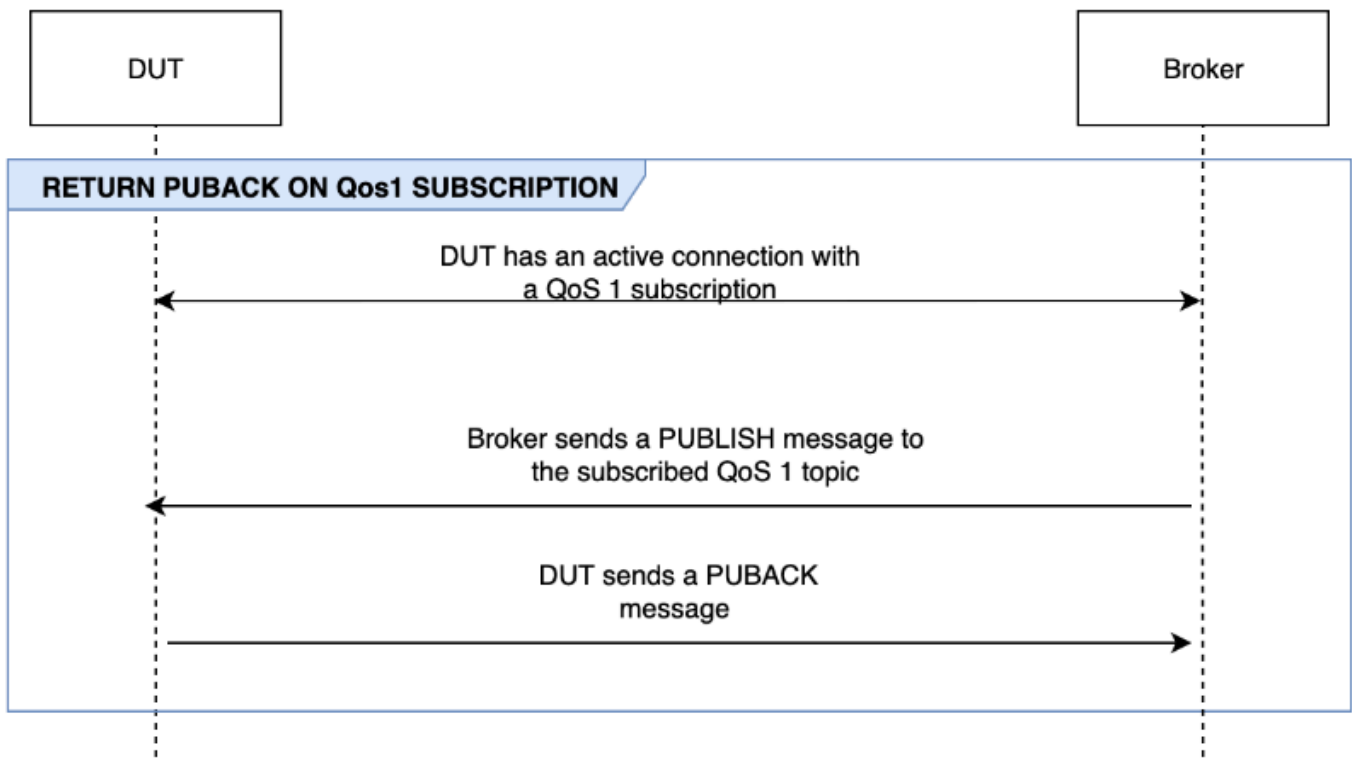
此階段的測試案例會以序列方式執行較複雜的測試，以驗證裝置是否遵循最佳實務。這些進階測試可供選擇，若無需要可以選擇不執行。根據案例需求，每項進階測試都各有專屬的逾期值。

### RETURN PUBACK ON QoS 1 SUBSCRIPTION

#### **i** Note

只有當您的裝置能夠執行 QoS 1 訂閱時，才可選取此案例。

此案例會驗證在裝置訂閱主題並收到來自代理人的 PUBLISH 訊息之後，是否會傳回 PUBACK 訊息。

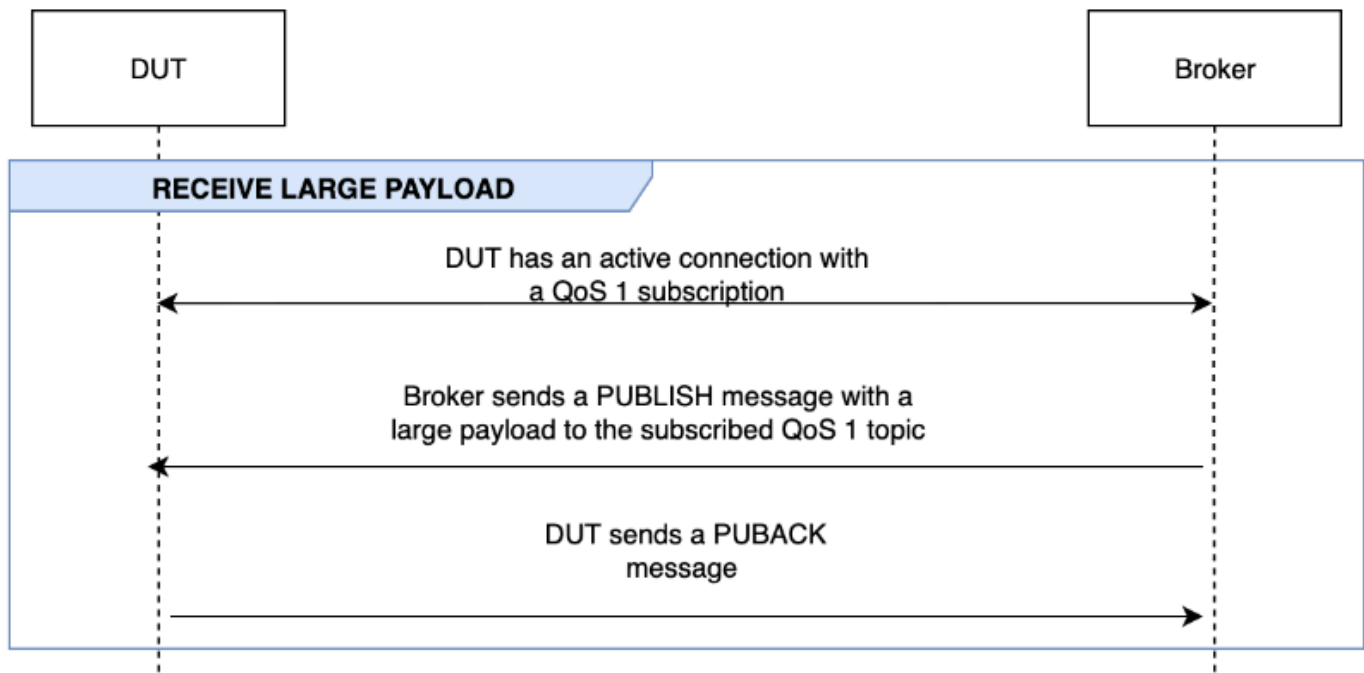


## RECEIVE LARGE PAYLOAD

### Note

只有當您的裝置能夠執行 QoS 1 訂閱時，才可選取此案例。

此案例會驗證裝置在收到具有大型承載的 QoS 1 主題的代理程式發出的 PUBLISH 訊息後，是否會以 PUBACK 訊息進行回應。可以使用 `LONG_PAYLOAD_FORMAT` 選項設定預期承載的格式。



## 持久性工作階段

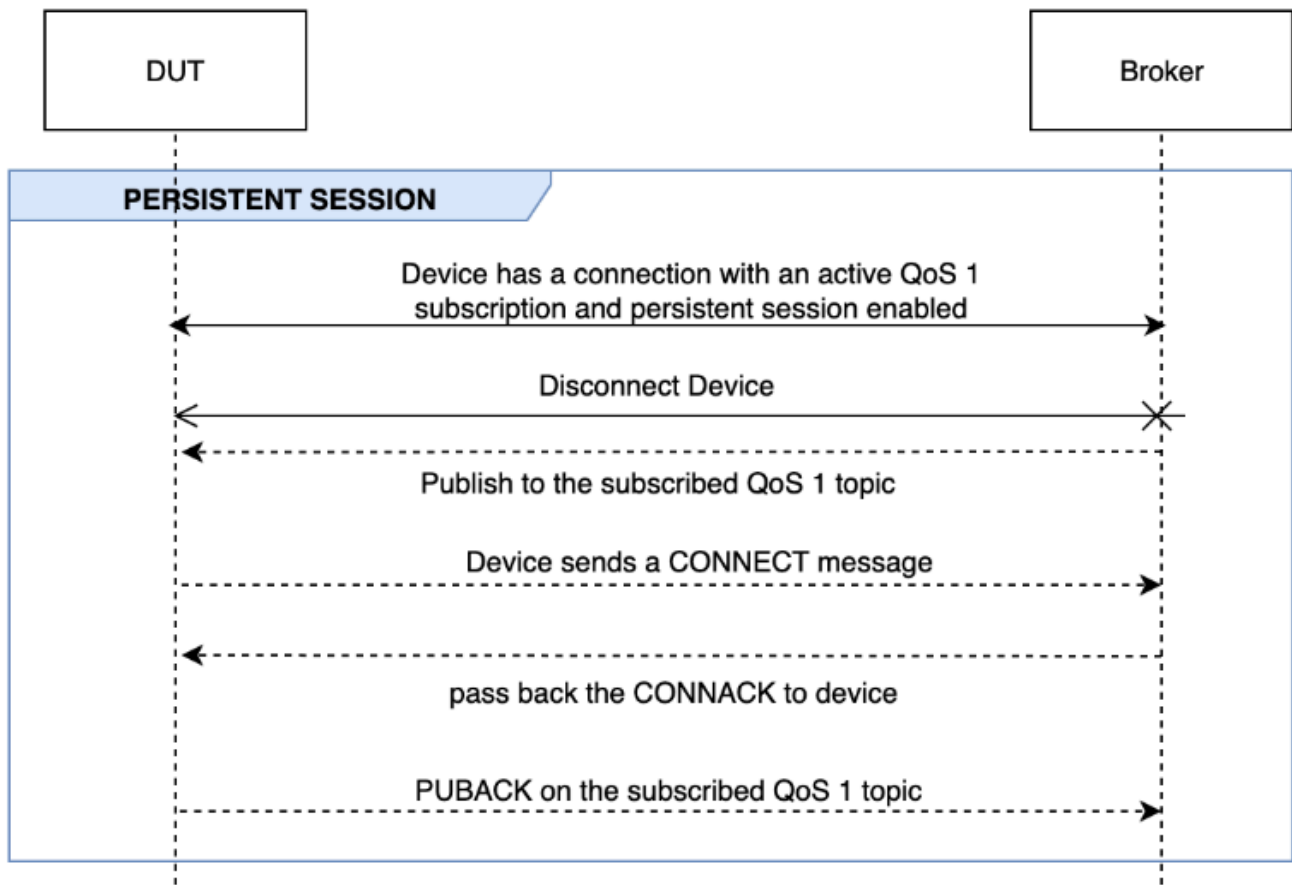
### Note

只有當您的裝置能夠執行 QoS 1 訂閱且可以維持持久性工作階段時，才可選取此案例。

此案例會驗證維持持久性工作階段時的裝置行為。測試會驗證是否滿足下列條件：

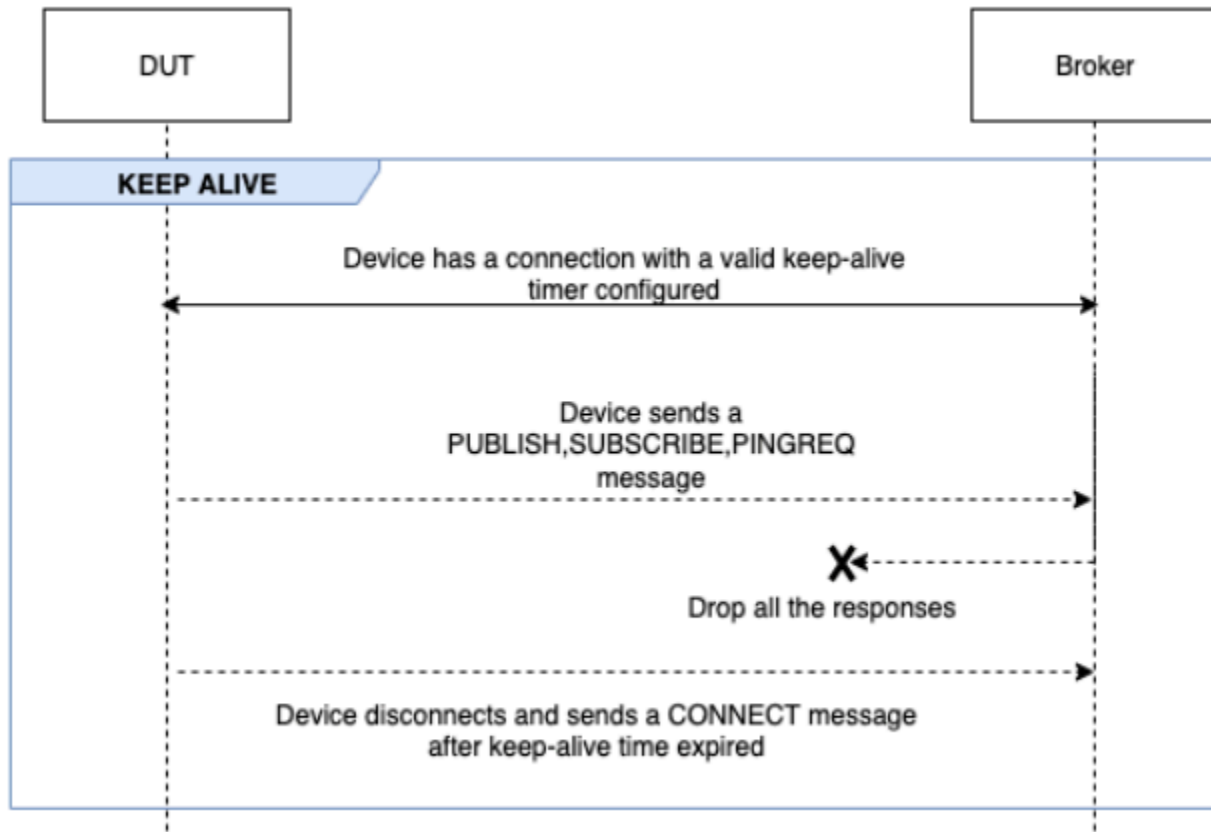
- 裝置連線至具有作用中 QoS 1 訂閱且已啟用持久性工作階段的代理程式。
- 裝置在工作階段期間成功中斷與代理程式的連線。
- 裝置會重新連線至代理程式，並繼續訂閱其觸發主題，而不會明確重新訂閱這些主題。
- 裝置成功接收代理程式為其訂閱主題所儲存的訊息，並如預期執行。

如需 AWS IoT 持續工作階段的詳細資訊，請參閱[使用 MQTT 持續性工作階段](#)。



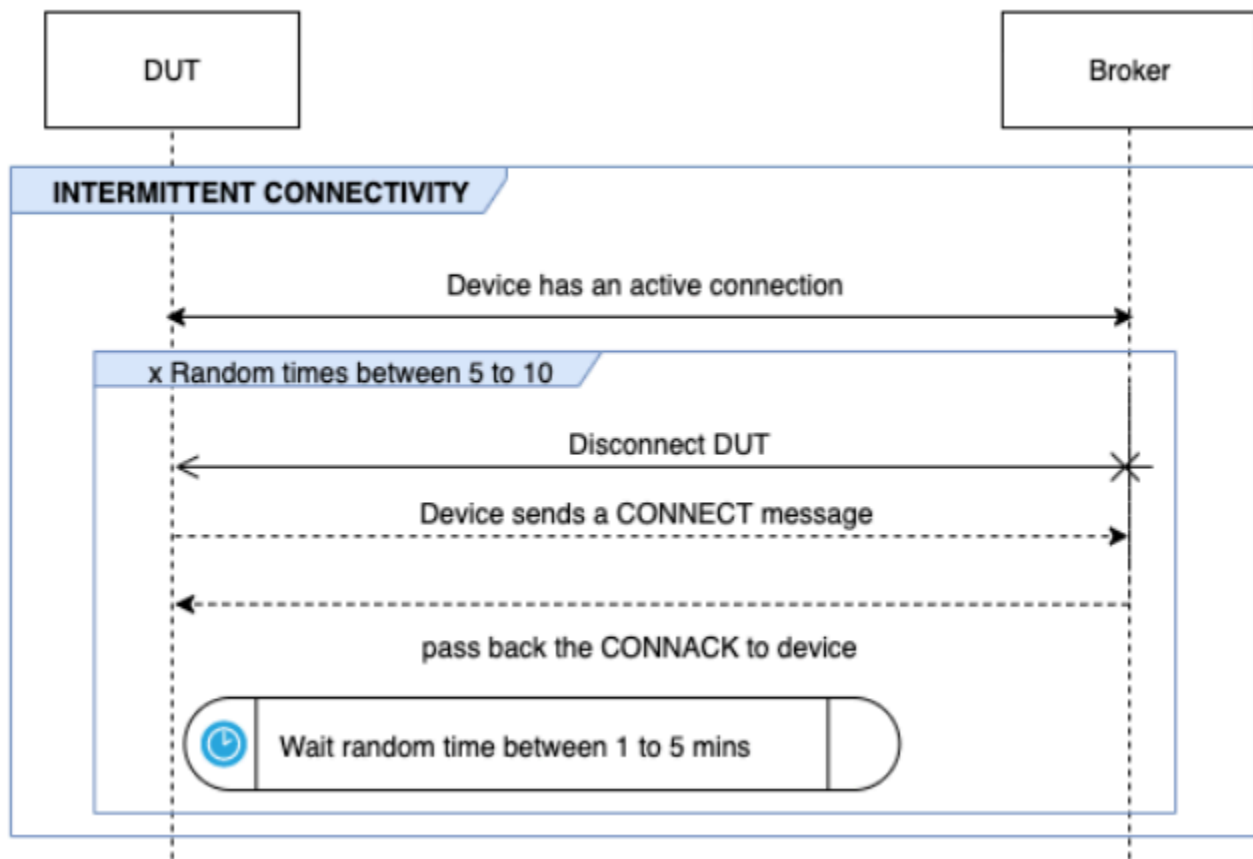
## Keep-Alive

此案例會驗證裝置在未收到代理程式的 Ping 回應後是否能成功中斷連線。必須為連線設定有效的保持連線計時器。在此測試中，代理程式會封鎖所有針對 PUBLISH、SUBSCRIBE 和 PINGREQ 訊息而傳送的回應。其也會驗證待測裝置是否中斷與 MQTT 的連線。



### INTERMITTENT CONNECTIVITY

此案例會驗證代理程式在隨機間隔內與裝置中斷連線之後，裝置是否可以恢復與代理程式的連線。

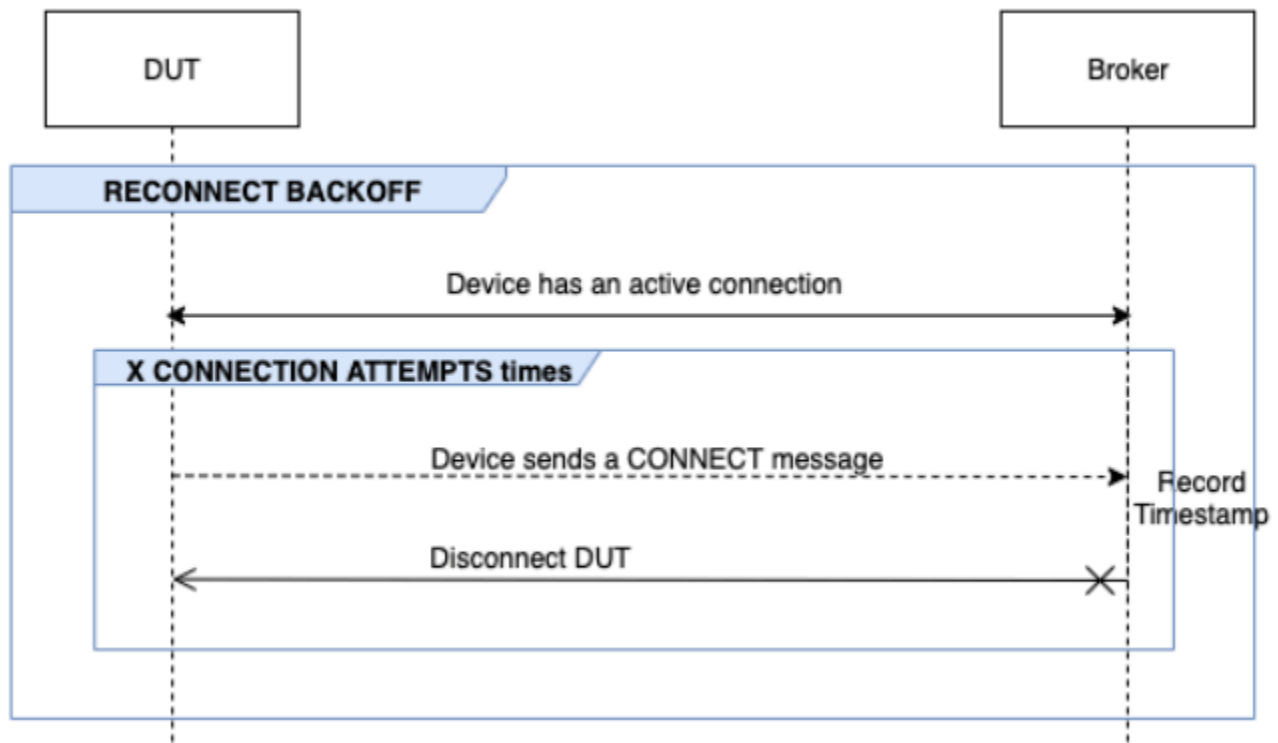


## RECONNECT BACKOFF

此案例會驗證代理程式多次中斷連線後，裝置是否會實作退避機制。Device Advisor 會將退避類型報告為指數、抖動、線性或常數。您可以使用 `BACKOFF_CONNECTION_ATTEMPTS` 選項來設定退避嘗試次數。預設值為 5。可以設定介於 5 到 10 之間的值。

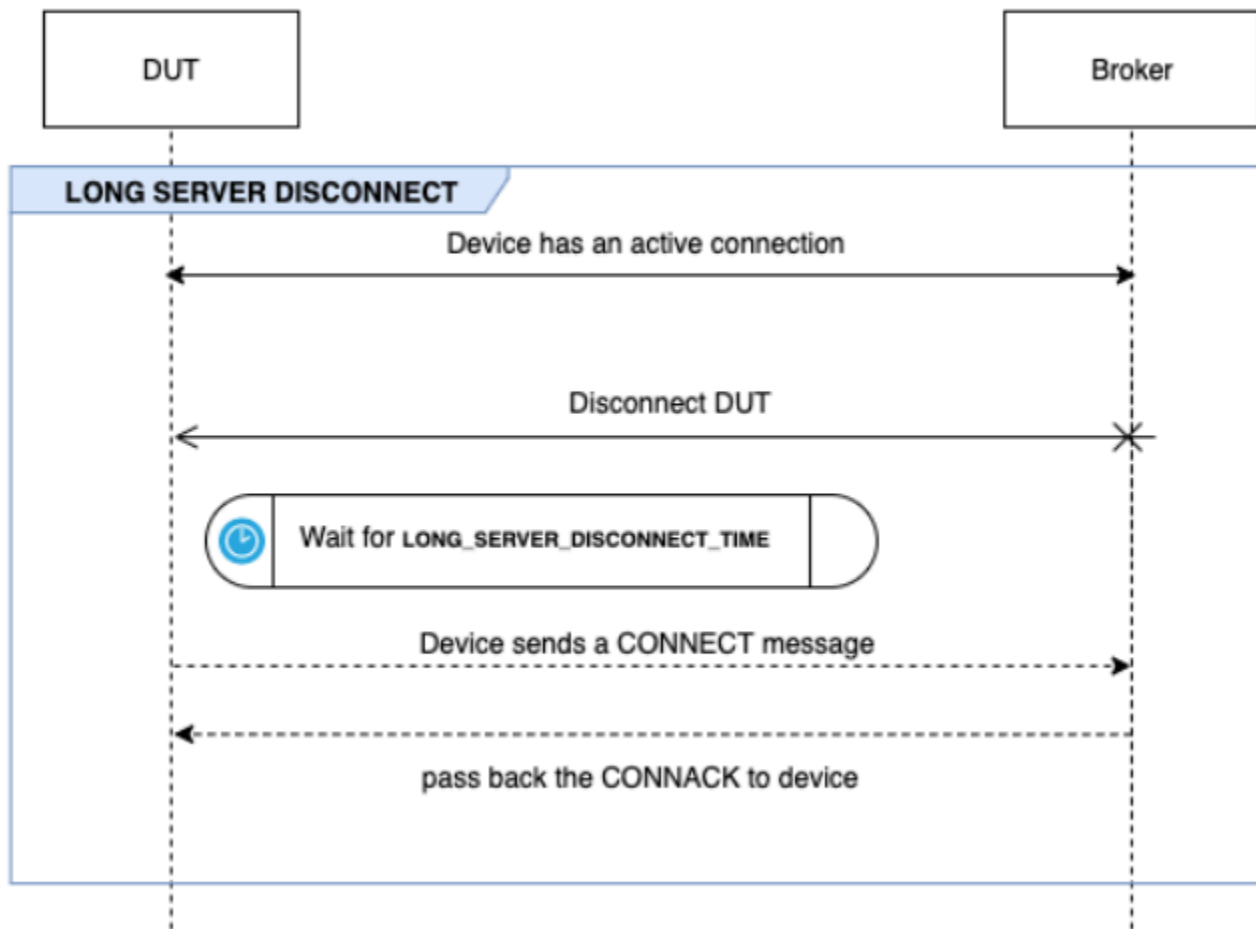
若要通過此測試，建議對受測裝置實作[指數退避和抖動](#)機制。





### 長時間伺服器中斷連線

此案例會驗證在代理程式長時間 (最多 120 分鐘) 中斷與裝置的連線之後，裝置是否可以成功重新連線。可以使用 `LONG_SERVER_DISCONNECT_TIME` 選項來設定伺服器中斷連線的時間。預設值為 120 分鐘。此值可以設定的範圍介於 30 至 120 分鐘。



## 額外執行時間

額外執行時間是指測試從完成上述所有測試之後到結束測試案例之前所等待的時間。客戶可使用此額外時段來監控裝置並記錄裝置與代理程式的所有通訊。可以使用 `ADDITIONAL_EXECUTION_TIME` 選項來設定額外執行時間。此選項依預設為 0 分鐘，可設定的範圍介於 0 到 120 分鐘。

## MQTT 長時間測試組態選項

為 MQTT 長時間測試提供的所有組態選項均非強制性。以下是可用的選項：

### 操作

裝置執行的操作清單，例如 `CONNECT`、`PUBLISH` 和 `SUBSCRIBE`。測試案例會依據指定的操作來執行情境案例。系統會將未指定的操作假設為有效。

```
{
  "OPERATIONS": ["PUBLISH", "SUBSCRIBE"]
}
```

```
//by default the test assumes device can CONNECT  
}
```

## 案例

根據所選操作，測試案例會執行情境案例來驗證裝置行為。案例有兩種類型：

- 基本案例屬於簡易測試，用於驗證裝置是否可以執行在組態中選擇的操作。這些條件會根據組態中指定的操作預先選取。在配置中不再需要輸入。
- 進階案例是對裝置執行較複雜的情境案例，以驗證裝置在真實世界條件下是否遵循最佳實務。這些選項均非強制性質，可以作為案例陣列傳遞給測試套件的組態輸入項。

```
{  
  "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:

測試案例等待所有基本測試完成的時間上限。預設值為 60 分鐘。此值可以設定的範圍介於 30 至 120 分鐘。

### LONG\_SERVER\_DISCONNECT\_TIME:

在「長時間伺服器中斷連線」測試期間，測試案例中斷並恢復與裝置連線所花費的時間。預設值為 60 分鐘。此值可以設定的範圍介於 30 至 120 分鐘。

### ADDITIONAL\_EXECUTION\_TIME:

若設定此選項，系統會在完成所有測試後提供一個時段，用以監控裝置與代理程式之間的事件。預設值為 0 分鐘。此值可以設定的範圍介於 0 至 120 分鐘。

### BACKOFF\_CONNECTION\_ATTEMPTS:

此選項可設定測試案例中斷與裝置連線的次數。重新連線退避測試會使用此功能。預設值為 5 次嘗試。此值可以設定的範圍介於 5 至 10 分鐘。

## LONG\_PAYLOAD\_FORMAT:

當測試案例發佈到裝置訂閱的 QoS 1 主題時，裝置所期望的訊息承載格式。

API 測試案例定義：

```
{
  "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 長時間測試案例摘要日誌

MQTT 長時間測試案例的執行時間較常規測試案例更長。會提供個別摘要日誌，其中列出執行期間的裝置連線、發佈和訂閱等重要事件。詳細資訊包括已測試的項目、未測試的項目以及失敗的項目。測試功能會在日誌結尾列出測試案例執行期間所發生全部事件的摘要。其中包含：

- 在裝置上設定的保持連線計時器。

- 裝置上設定的持久性工作階段旗標。
- 裝置在測試執行期間的連線次數。
- 裝置重新連線退避類型 (若已通過重新連線退避測試的驗證)。
- 在測試案例執行期間作為裝置發佈目標的主題。
- 裝置在測試案例執行期間訂閱的主題。

# AWS IoT Device Management 軟體 Package 件目錄

使用 AWS IoT Device Management 軟體 Package 件目錄，您可以維護軟體套件及其版本的清查。您可以將封裝版本關聯至個別物件和 AWS IoT 動態物群組，並透過內部流程或[AWS IoT 工作](#)來部署它們。

軟體套件包含一或多個套件版本，它是可以部署為單一單元的檔案集合。套件版本可以包含韌體、作業系統更新、裝置應用程式、組態和安全性修補程式。軟體會隨著時間發展，您可以建立新套件版本並將其部署到您的機群。

AWS IoT 軟體套件集線器位於內部 AWS IoT Core。您可以使用該中樞集中註冊和維護軟體套件庫存和中繼資料，以建立軟體套件及其版本的目錄。您可以選擇根據裝置上部署的軟體套件和套件版本，對裝置進行分組。此功能可讓您將裝置端套件庫存保留為已命名影子、根據版本為裝置建立關聯和分組，並使用機群指標來視覺化跨機群的套件版本分佈。

若您已建立內部軟體部署系統，就能繼續使用該程序來部署套件版本。若您尚未建立部署程序，或若您偏好的話，建議您使用 [AWS IoT 任務](#)，以使用軟體套件目錄中的功能。如需詳細資訊，請參閱[準備 AWS IoT 工作](#)。

本章包含下列部分：

- [準備使用軟體套件目錄](#)
- [準備安全性](#)
- [準備機群索引](#)
- [準備 AWS IoT 工作](#)
- [開始使用軟體套件目錄](#)

## 準備使用軟體套件目錄

下節提供 Package 件版本生命週期的概觀，以及使用「AWS IoT Device Management 軟體套件目錄」的資訊。

### 套件版本生命週期

套件版本可以經歷下列生命週期狀態：draft、published、和 deprecated。也可以為 deleted。



- 草稿

當您建立套件版本時，它draft處於狀態。此狀態表示軟體套件正在準備中或不完整。

當套件版本處於此狀態時，您無法部署它。您可以編輯套件版本的描述、屬性和標籤。

您可以使用主控台或發出「版本」published 或「版本」API 作業，將draftdeleted狀態為狀態的套件[UpdatePackage](#)版本轉換為或轉換為狀態的套件[DeletePackage](#)版本。

- 已發佈

當您的套件版本已準備好部署時，請將套件版本轉換為published狀態。處於此狀態時，您可以透過在主控台中編輯軟體套件或透過 [UpdatePackage](#)API 作業，選擇將套件版本識別為預設版本。在此狀態下，您只能編輯描述和標籤。

您可以使用主控台或發出「版本」 deprecated 或「版本」API 作業，將publisheddeleted狀態為狀態的套件UpdatePackage版本轉換為或DeletePackage將其轉換為狀態。

- 已棄用

若有新的套件版本可用，您可以將較早的套件版本轉換為 deprecated。您仍然可以使用已取代的套件版本部署工作。您也可以將已取代的套件版本命名為預設版本，並僅編輯描述和標籤。

請考慮將套件版本轉換為版本已過期的deprecated時間，但由於執行階段的相依性，您仍然擁有使用舊版本的裝置，或者需要維護套件版本。

您可以使用控制台或發出「版本」`published` 或「版本」API 操作，將 `deprecated deleted` 狀態為狀態的軟件包 [UpdatePackage 版本](#) 轉換為或處於該狀態的軟件包 [DeletePackage 版本](#)。

- Deleted (已刪除)

當您不想再使用套件版本時，您可以使用主控台或發出 Ver [DeletePackageVersion](#) API 作業將其刪除。

#### Note

若您在有待處理任務參照套件版本時刪除該套件版本，當任務成功完成並嘗試更新預留已命名影子時，您會收到錯誤訊息。

若您要刪除的軟體套件版本已指定為預設套件版本，您必須先更新該套件，將其他版本指定為預設版本，或將該欄位保留為未指定狀態。您可以使用控制台或 [UpdatePackage 版本](#) API 操作來執行此操作。（要刪除任何命名的軟件包版本作為默認值，請在發出 [UpdatePackage](#) API 操作時將 `unset DefaultVersion` 參數設置為 `true`）。

若您透過主控台刪除軟體套件，則會刪除與該套件相關聯的所有套件版本，除非某個套件版本已指定為預設版本。

## 套件版本命名慣例

命名套件版本時，請務必規劃並套用邏輯命名策略，以便您和其他人員皆可輕易識別最新的套件版本和版本進度。建立套件版本時，您必須提供版本名稱，但策略和格式主要取決於您的商業案例。

最佳作法是建議使用語意版本控制 [SemVer](#) 格式。例如，`1.2.3`，其中 1 是功能上不相容變更的主要版本，2 是功能上相容變更的主要版本，3 是修補程式版本 (用於錯誤修正)。如需詳細資訊，請參閱 [語意版本控制 2.0.0](#)。如需有關套件版本名稱需求的詳細資訊，請參閱 AWS IoT API 參考指南中的 [VersionName](#)。

## 預設版本

您不一定要將版本設定為預設版本。您可以新增或移除預設套件版本。您也可以部署未指定為預設版本的套件版本。

套件版本在建立時會處於 `draft` 狀態，在您將套件版本轉換為「已發佈」之前，無法將其指定為預設版本。軟體套件目錄不會自動選取某個版本作為預設版本，也不會自動將較新的套件版本更新為預設版本。您必須故意命名您通過控制台或發出 Version API 操作選擇的軟件包 [UpdatePackage 版本](#)。



## 版本屬性

版本屬性及其值皆包含套件版本的重要資訊。建議您定義套件或套件版本的一般用途屬性。例如，您可以為平台、架構、作業系統、發行日期、作者或 Amazon S3 URL 建立名稱/值配對。

當您使用 AWS IoT 工作文件建立工作時，您也可以選擇使用參照屬性值的替代變數 (`$parameter`)。如需詳細資訊，請參閱[準備 AWS IoT 工作](#)。

封裝版本中使用的版本屬性不會自動新增至保留的名為 `shadow`，也無法透過叢集索引直接建立索引或查詢。若要透過叢集索引建立索引或查詢封裝版本屬性，您可以在保留的已命名 `shadow` 中填入 `version` 屬性。

我們建議在保留的版本屬性參數中命名為陰影捕獲設備報告的屬性，例如操作系統和安裝時間。您也可以透過「叢集索引」建立索引和查詢它們。

版本屬性不需要遵循特定的命名慣例。您可以建立名稱/值配對，以滿足您的商業需求。套件版本上所有屬性的整體大小限制為 3 KB。如需詳細資訊，請參閱[軟體 Package 件目錄軟體套件和套件版本限制](#)。

## 啟用 AWS IoT 叢集索引

您必須啟動軟體套件目錄的機群索引，才能建立或更新軟體套件和套件版本。叢集索引提供支援，AWS IoT 可透過依版本篩選的動態物件群組來分組物件。例如，機群索引可以識別已安裝或未安裝特定套件版本、未安裝任何套件版本或符合特定名稱/值配對的物件。最後，機群索引也提供標準和自訂指標，您可以使用這些指標來深入瞭解機群狀態。如需詳細資訊，請參閱[準備機群索引](#)。

### Note

啟用軟體套件目錄的機群索引會產生標準服務費用。如需詳細資訊，請參閱[AWS IoT Device Management定價](#)。

## 預留已命名影子

預留已命名影子 `$package` 會反映安裝於裝置的軟體套件和套件版本狀態。機群索引使用預留已命名影子做為資料來源，以建立標準和自訂指標，讓您可以查詢機群狀態。如需詳細資訊，請參閱[準備機群索引](#)。

預留已命名影子類似[已命名影子](#)，但其名稱為預先定義，無法變更。此外，預留已命名影子不會隨中繼資料更新，且只會使用 `version` 和 `attributes` 關鍵字。

包含其他關鍵字 (例如description) 的更新要求會在rejected主題下收到錯誤回應。如需詳細資訊，請參閱[裝置影子錯誤訊息](#)。

當您透過主控台建立 AWS IoT 物件、AWS IoT 工作成功完成並更新陰影，以及發出 `UpdateThingShadow` API 作業時，都可以建立它。如需詳細資訊，請參閱 AWS IoT Core 開發人員指南中的 [UpdateThingShadow](#)。

### Note

索引預留已命名影子不會計入機群索引可以索引的已命名影子數量。如需詳細資訊，請參閱 [AWS IoT Device Management 機群索引限制和配額](#)。此外，如果您選擇讓工作在 AWS IoT 工作順利完成時更新保留的具名陰影，則 API 呼叫會計入您的 Device Shadow 和登錄作業，並可能產生費用。如需詳細資訊，請參閱 [AWS IoT Device Management 作業限制和配額](#) 和 [IndexingFilter](#) API 資料類型。

## \$package 影子的結構

預留已命名影子包含以下內容：

```
{
  "state": {
    "reported": {
      "<packageName>": {
        "version": "",
        "attributes": {
        }
      }
    }
  },
  "version" : 1
  "timestamp" : 1672531201
}
```

影子屬性會更新為下列資訊：

- `<packageName>`：已安裝軟體套件的名稱，此套件會以 [PackageName](#) 參數更新。
- `version`：已安裝套件版本的名稱，此版本會以 [versionName](#) 參數更新。
- `attributes`：由裝置儲存並依機群索引編製索引的選用中繼資料。這可讓客戶根據儲存的資料查詢其索引。

- `version`：影子的版本號碼。每次影子更新並開始於 1 時，皆會自動遞增。
- `timestamp`：顯示上次影子更新時間並以 [Unix 時間](#) 記錄。

如需詳細了解已命名影子的格式和行為，請參閱 [AWS IoT Device Shadow 服務 訊息順序](#)。

## 刪除軟體套件及其套件版本

刪除軟體套件之前，請執行下列動作：

- 確認套件及其版本並未主動部署。
- 先刪除所有相關版本。若其中一個版本為預設版本，您必須從套件中移除具名預設版本。因為您不一定要指定預設版本，移除預設版本並不會造成衝突。若要從軟體套件中移除預設版本，請透過主控台編輯套件或使用 [UpdatePackageVersionAPI](#) 作業。

只要沒有具名預設套件版本，您便可以使用主控台刪除軟體套件，且其所有套件版本也將一併刪除。若您使用 API 呼叫來刪除軟體套件，您必須先刪除套件版本再刪除軟體套件。

## 準備安全性

本節討論「AWS IoT Device Management 軟體 Package 件目錄」的主要安全性需求。

### 以資源為基礎的身分驗證

軟體套件目錄在更新機群上的軟體時，會使用以資源為基礎的授權來提升安全性。這表示您必須建立 AWS Identity and Access Management (IAM) 政策，以授與對軟體套件和套件版本執行 `createreadupdatedelete`、`list` 動作的權限，並參考您要在本 `Resources` 節中部署的特定軟體套件和套件版本。您也需要這些權限，以便更新 [預留已命名影子](#)。您可以在每個實體中納入 Amazon Resource Name (ARN) 來參考軟體套件和套件版本。

#### Note

如果您想要授與套件版本 API 呼叫的權限 (例如 [CreatePackage版本](#)、[版本](#)、[UpdatePackageDeletePackage版本](#)) 的權限，則您必須在原則中同時包含軟體套件和套件版本 ARN。如果您想要授與軟體套件 API 呼叫的權限 (例如 [CreatePackageUpdatePackage](#)、和 [DeletePackage](#)) 的原則，您必須在原則中僅包含軟體套件 ARN。

請遵循下方說明，建構軟體套件和套件版本 ARN：

- 軟體套件：arn:aws:iot:<region>:<accountID>:package/<packageName>/package
- 套件版本：arn:aws:iot:<region>:<accountID>:package/<packageName>/version/<versionName>

**Note**

您可能會在此政策中納入其他相關權利。例如，您可以會納入 job、thinggroup 和 jobtemplate 的 ARN。如需詳細資訊和原則選項的完整清單，請參閱使用 [AWS IoT 工作保護使用者和裝置](#)。

例如，若您有如下名稱的軟體套件和套件版本：

- AWS IoT 事情：myThing
- 套件名稱：samplePackage
- 版本 1.0.0

該政策看起來類似以下範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:createPackage",
        "iot:createPackageVersion",
        "iot:updatePackage",
        "iot:updatePackageVersion"
      ],
      "Resource": [
        "arn:aws:iot:us-east-1:111122223333:package/samplePackage",
        "arn:aws:iot:us-east-1:111122223333:package/samplePackage/version/1.0.0"
      ]
    },
    {
      "Effect": "Allow",
```

```

    "Action": [
      "iot:GetThingShadow",
      "iot:UpdateThingShadow"
    ],
    "Resource": "arn:aws:iot:us-east-1:111122223333:thing/myThing/$package"
  }
]
}

```

## AWS IoT 部署套件版本的 Job 權限

基於安全考量，您必須授予部署套件和套件版本的權限，並為允許部署的特定套件和套件版本命名。若要執行這項動作，您可以建立 IAM 角色和政策，以授予透過套件版本部署任務的權限。政策必須將目的地套件版本指定為資源。

### IAM 政策

IAM 政策授予建立任務的權限，其中包含在 Resource 區段中命名的套件和版本。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:CreateJob",
        "iot:CreateJobTemplate"
      ],
      "Resource": [
        "arn:aws:iot:*:111122223333:job/<jobId>",
        "arn:aws:iot:*:111122223333:thing/<thingName>/$package",
        "arn:aws:iot:*:111122223333:thinggroup/<thingGroupName>",
        "arn:aws:iot:*:111122223333:jobtemplate/<jobTemplateName>",
        "arn:aws:iot:*:111122223333:package/<packageName>/
        version/<versionName>"
      ]
    }
  ]
}

```

**Note**

如果您要部署解除安裝軟體套件和套件版本的工作，您必須授權 ARN，其中包含套件版本 \$null，如下所示：

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

## AWS IoT 更新保留的命名陰影的 Job 權限

若要允許任務在完成時更新物件的預留已命名影子，您必須建立 IAM 角色和政策。您可以在 AWS IoT 控制台透過兩種方式執行此動作。第一，在控制台建立軟體套件時。若您看到啟用套件管理的相依性對話方塊，您可以選擇使用現有角色或建立新角色。或者，在 AWS IoT 主控台選擇設定，接著選擇管理索引以及管理裝置套件和版本的索引。

**Note**

如果您選擇讓 AWS IoT Job 服務在工作成功完成時更新保留的具名陰影，則 API 呼叫會計入您的 Device Shadow 和登錄作業，並可能產生費用。如需詳細資訊，請參閱 [AWS IoT Core 定價](#)。

使用建立角色選項時產生的角色名稱開頭為 `aws-iot-role-update-shadows` 且包含下列政策：

設定角色

許可

權限政策授予查詢和更新物件影子的權限。資源 ARN 中的 `$package` 參數則針對預留已命名影子。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:DescribeEndpoint",
      "Resource": ""
    }
  ],
}
```

```

    {
      "Effect": "Allow",
      "Action": [
        "iot:GetThingShadow",
        "iot:UpdateThingShadow"
      ],
      "Resource": [
        "arn:aws:iot:<regionCode>:111122223333:thing/<thingName>/$package"
      ]
    }
  ]
}

```

## 信任關係

除了權限政策之外，角色也需要與 AWS IoT Core 的信任關係，以便實體擔任該角色並更新預留已命名影子。

```

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

```

## 設定使用者政策

### IAM : PassRole 許可

最後，您必須具有在呼叫 [UpdatePackageConfiguration](#) API 作業 AWS IoT Core 時將角色傳遞給的權限。

```

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

```

```
    "Effect": "Allow",
    "Action": [
        "iam:PassRole",
        "iot:UpdatePackageConfiguration"
    ],
    "Resource": "arn:aws:iam::111122223333:role/<roleName>"
}
]
```

## AWS IoT 可從 Amazon S3 下載的任務許可

任務文件會儲存於 Amazon S3。透過 AWS IoT 任務分派時，請參考此檔案。您必須向 AWS IoT 工作提供下載檔案的權限 (s3:GetObject)。您也必須在 Amazon S3 和 AWS IoT 任務之間建立信任關係。如需建立這些政策的指示，請參閱[管理任務](#)中的[預先簽章的 URL](#)。

## 準備機群索引

透過 AWS IoT 叢集索引，您可以使用保留的命名 shadow (\$package) 來搜尋和彙總資料。您也可以透過查詢[預留已命名影子](#)和[動態 AWS IoT 物件群組來群組物件](#)。例如，您可以找到有關哪 AWS IoT 些項目使用特定軟件包版本，未安裝特定軟件包版本或沒有安裝任何軟件包版本的信息。您可以組合屬性以獲得進一步分析。例如，識別具有特定版本且屬於特定物件類型的物件 (例如版本 1.0.0 和 pump\_sensor 的物件類型)。如需詳細資訊，請參閱[機群索引](#)。

## 將 \$package 影子設定為資料來源

若要搭配使用機群索引與軟體套件目錄，您必須啟用機群索引、將已命名影子設定為資料來源，並定義 \$package 為已命名影子篩選條件。若您尚未啟用機群索引，您可以在此程序中啟用。在主控台的[AWS IoT Core](#) 開啟設定，選擇管理索引，接著選擇新增已命名影子、新增裝置軟體套件和版本，以及更新。如需詳細資訊，請參閱[管理物件索引](#)。

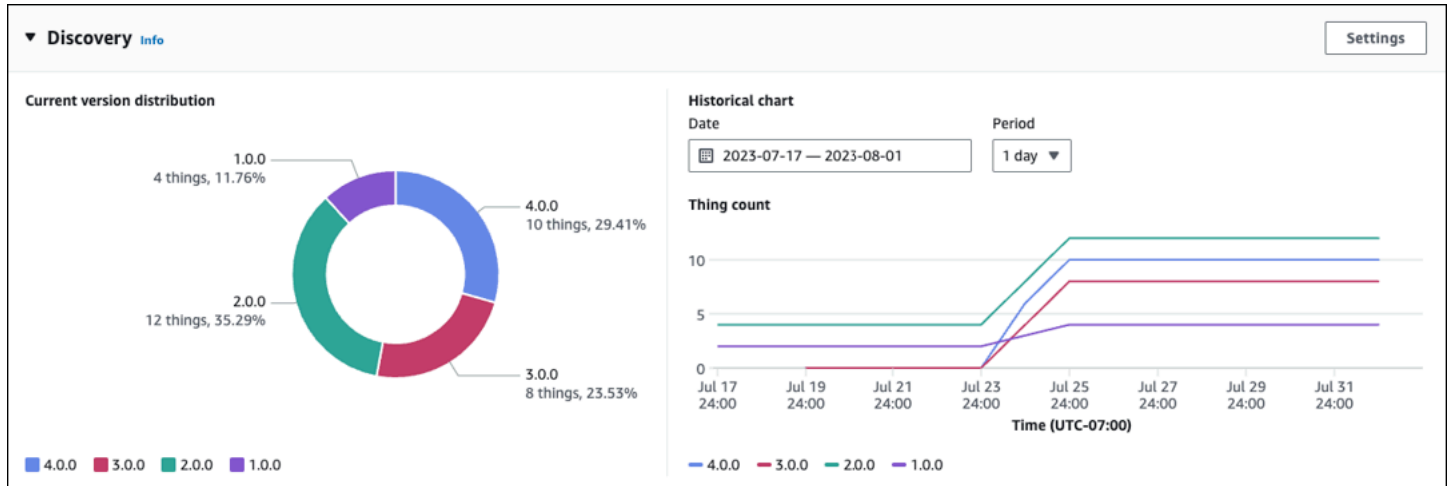
或者，您也可以在建​​立第一個套件時啟用機群索引。當啟用套件管理的相依性對話方塊出現時，請選擇將裝置軟體套件和版本新增為資料來源至機群索引。透過此選項，您也可以啟用機群索引。

### Note

啟用軟體套件目錄的機群索引會產生標準服務費用。如需詳細資訊，請參閱[AWS IoT Device Management定價](#)。



## 控制台顯示的指標



在 AWS IoT 主控台軟體套件詳細資料頁面上，「探索」面板會顯示透過 `$package` 陰影擷取的標準度量。

- 「目前版本」散佈圖會顯示與此軟體套件相關聯之所有裝置之物件的 10 個 AWS IoT 最新套件版本的裝置數目和百分比。注意：若軟體套件的套件版本多於圖表中標示的版本，您在其他中會發現它們已分組。
- 歷史圖表會顯示指定期間內，與所選套件版本相關聯的裝置數量。圖表一開始會是空白的，您可以選取最多 5 個套件版本並定義日期範圍和時間間隔。若要選取圖表的參數，請選擇設定。歷史圖表顯示的資料可能與目前版本分佈圖表不同，這是因為兩者顯示的套件版本數量不同，也因為歷史圖表可以選擇要分析哪些套件版本。注意：選取要進行視覺化的套件版本時，會計入機群指標限制的上限數量。如需詳細資訊，請參閱[機群索引限制和配額](#)。

如需了解收集套件版本分佈的其他方法，請參閱[透過 `getBucketsAggregation` 收集套件版本分佈](#)。

## 查詢模式

搭配軟體套件目錄進行機群索引時，會使用大多數受支援的功能 (例如詞彙和字詞以及搜尋欄位)，這些功能是標準的機群索引功能。例外狀況是，`comparison` 和 `range` 查詢不適用於預留已命名影子 (`$package`) `version` 索引鍵。但是，這些查詢可用於 `attributes` 索引鍵。如需詳細資訊，請參閱[查詢語法](#)。

## 範例資料

注意：如需詳細了解預留已命名影子及其結構，請參閱[預留已命名影子](#)。

在此範例中，第一部裝置名為 Anything 且已安裝下列套件：

- 軟體套件：SamplePackage

套件版本：1.0.0

套件 ID：1111

影子如下所示：

```
{
  "state": {
    "reported": {
      "SamplePackage": {
        "version": "1.0.0",
        "attributes": {
          "s3UrlForSamplePackage": "https://EXAMPLEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
          "packageID": "1111"
        }
      }
    }
  }
}
```

第二部裝置名為 AnotherThing 且已安裝下列套件：

- 軟體套件：SamplePackage

套件版本：1.0.0

套件 ID：1111

- 軟體套件：OtherPackage

套件版本：1.2.5

套件 ID：2222

影子如下所示：

```
{
```

```

"state": {
  "reported": {
    "SamplePackage": {
      "version": "1.0.0",
      "attributes": {
        "s3UrlForSamplePackage": "https://EXAMPLEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile1",
        "packageID": "1111"
      }
    },
    "OtherPackage": {
      "version": "1.2.5",
      "attributes": {
        "s3UrlForOtherPackage": "https://EXAMPLEBUCKET.s3.us-
west-2.amazonaws.com/exampleCodeFile2",
        "packageID": "2222"
      }
    }
  },
}
}
}

```

## 範例查詢

下表列出根據 Anything 和 AnotherThing 範例裝置影子的查詢範例。如需詳細資訊，請參閱[範例物件查詢](#)。

FreeRTOS 的 AWS IoT Device Tester 最新版本

已請求的資訊	查詢	結果
已安裝特定套件版本的物件	shadow.name.\$package.reported.SamplePackage.version:1.0.0	Anything, OtherThing
未安裝特定套件版本的物件	NOT shadow.name.\$package.reported.OtherPackage.version:1.2.5	Anything

已請求的資訊	查詢	結果
所使用套件版本的套件 ID 大於 1500 的任何裝置	<code>shadow.name.\$package.reported.*.attributes.packageID&gt;1500"</code>	OtherThing
已安裝特定套件且安裝多個套件的物件	<code>shadow.name.\$package.reported.SamplePackage.version:1.0.0 AND shadow.name.\$package.reported.totalCount:2</code>	OtherThing

## 透過 `getBucketsAggregation` 收集套件版本分佈

除了 AWS IoT 主控台中的「探索」面板之外，您也可以使用 [GetBucketsAggregation](#) API 作業取得套件版本發佈資訊。若要取得套件版本分佈資訊，您必須執行下列動作：

- 在機群索引中為每個軟體套件定義自訂欄位。注意：建立自訂欄位會計入 [AWS IoT 機群索引服務配額](#)。
- 自訂欄位的格式如下所示：

```
shadow.name.$package.reported.<packageName>.version
```

如需詳細資訊，請參閱 AWS IoT 叢集索引中的 [\[自訂欄位\]](#) 一節。

## 準備 AWS IoT 工作

AWS IoT Device Management 「軟體 Package 件目錄」透過替代參數擴充 AWS IoT 工作，並與 AWS IoT 叢集索引、動態物件群組以及 AWS IoT 物件的保留命名 shadow 整合。

**Note**

若要使用「軟體 Package 件目錄」提供的所有功能，您必須建立下列 AWS Identity and Access Management (IAM) 角色和政策：[部署套件版本的AWS IoT 工具AWS IoT作權限](#)，以及[更新保留命名為 shadow 的工作權限](#)。如需詳細資訊，請參閱[準備安全性](#)。

## AWS IoT 工作的替代參數

您可以使用替代參數作為 AWS IoT 工作文件中的預留位置。當任務服務遇到替換參數時，會將任務指向參數值的具名軟體版本屬性。您可以使用此程序建立單一任務文件，並透過一般用途屬性將中繼資料傳遞至任務。例如，您可以透過套件版本屬性將 Amazon Simple Storage Service (Amazon S3) URL、軟體套件 Amazon Resource Name (ARN) 或簽章傳遞至任務文件。

替換參數應於任務文件如下格式化：

```
${aws:iot:package:<packageName>:version:<versionName>:attributes:<anyAttributeName>}
```

在此範例中，有一個名為 samplePackage 的軟體套件，其名為 2.1.5 的套件版本具有下列屬性：

- 名稱：s3URL，值：https://EXAMPLEBUCKET.s3.us-west-2.amazonaws.com/exampleCodeFile
  - 此屬性會識別儲存於 Amazon S3 的程式碼檔案位置。
- 名稱：signature，值：aaaaabbbbccccddddddeeeeffffffggggghhhhhiiiiijjjj
  - 此屬性會提供裝置做為安全措施所需的程式碼簽章值。如需詳細資訊，請參閱[適用於任務的程式碼簽署](#)。注意：此屬性僅為範例，不是軟體套件目錄或任務的必要部分。

針對 downloads，任務文件參數的寫入方式如下：

```
{
  "samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
}
```

針對 signature，任務文件參數的寫入方式如下：

```
{
  "samplePackage": "${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
}
```

```
}
```

完整的任務文件寫入方式如下：

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage":
"${aws:iot:package:samplePackage1:version:2.1.5:attributes:s3URL}"
      },
    ],
    "signature": [
      "samplePackage" :
"${aws:iot:package:samplePackage1:version:2.1.5:attributes:signature}"
    ]
  }
}
```

進行替換後，下列任務文件會部署至裝置：

```
{
  ...
  "Steps": {
    "uninstall": ["samplePackage"],
    "download": [
      {
        "samplePackage": "https://EXAMPIEBUCKET.s3.us-west-2.amazonaws.com/
exampleCodeFile"
      },
    ],
    "signature": [
      "samplePackage" : "aaaaabbbbccccccddddeeeeffffffggggghhhhhiiiiijjjj"
    ]
  }
}
```

如需有關 AWS IoT 工作、建立工作文件和部署工作的詳細資訊，請參閱[工作](#)。

## 準備任務文件和套件版本以進行部署

建立套件版本時，它會處於draft狀態，表示它正在準備進行部署。若要準備要部署的套件版本，您必須建立工作文件，將文件儲存在任務可存取的位置 (例如 Amazon S3)，並確認套件版本具有您希望工作文件使用的屬性值。(注意：您只能在套件版本處於draft狀態時更新它的屬性。)

如果您對套件版本感到滿意，請透過 AWS IoT 主控台中的軟體套件詳細資料頁面或發出 Version API 作業來發佈套件 [UpdatePackage](#) 版本。然後，您可以在透過 AWS IoT 主控台或發出 [CreateJob](#) API 作業來建立工作時參照套件版本。

## 部署時命名套件和版本

部署 AWS IoT 工作時，必須命名與工作部署 (destinationPackageVersions) 中工作文件中命名的相同軟體套件和套件版本。若不執行此動作，您會收到錯誤訊息，顯示遺失的套件版本。

您可以納入任務文件中未包含的其他軟體套件和套件版本。如此一來，任務不會向裝置提供說明，解釋如何處理這些檔案，設備應能自行處理檔案。例如，若其他檔案包含裝置可能參考的資料，您可能會將其他檔案傳送至裝置。

## 透過 AWS IoT 動態物件群組鎖定工作

軟體套件目錄可與 [機群索引](#)、[AWS IoT 任務](#) 和 [AWS IoT 動態物件群組](#) 搭配使用，以篩選並鎖定機群中的裝置，選取要部署至裝置的套件版本。您可以根據裝置目前的套件資訊執行叢集索引查詢，並針對 AWS IoT 工作指定這些項目。您也可以發行軟體更新，但僅限符合資格的目標裝置。例如，您可以指定只要將組態部署到目前執行 iot-device-client 1.5.09 的裝置。如需詳細資訊，請參閱 [建立動態物件群組](#)。

## 預留已命名影子和套件版本

如果已設定，則 AWS IoT 作業可在工作順利完成時，更新物件的保留名為 shadow (\$package)。如此一來，您便不需要手動將套件版本與物件的預留已命名影子建立關聯。

在下列情況下，您可以選擇手動更新套件版本，或將其與物件的預留已命名影子建立關聯：

- 您可以在 AWS IoT Core 不關聯已安裝套件版本的情況下註冊物件。
- AWS IoT 工作未設定為更新物件的保留名為 shadow。
- 您可以使用內部程序將套件版本傳送至您的叢集，而該程序在完成 AWS IoT Core 時不會更新。

**Note**

我們建議您使用 AWS IoT 工作來更新保留名為 shadow (\$package) 中的套件版本。當 AWS IoT 作業也設定為更新 \$package 陰影時，透過其他處理程序 (例如手動或程式化 API 呼叫) 更新陰影中的版本參數，可能會導致裝置上的實際版本與回報給保留命名 shadow 的版本之間的不一致。

您可以透過主控台或 [UpdateThingShadow](#) API 操作，將套件版本新增或更新至物件的預留已命名影子 (\$package)。如需詳細資訊，請參閱[將封裝版本與 AWS IoT 物件相關聯](#)。

**Note**

將套件版本與 AWS IoT 物件相關聯並不會直接更新裝置軟體。您必須將套件版本部署至裝置，才能更新裝置軟體。

## 解除安裝軟體套件及其套件版本

\$null 是預留的預留位置，會提示 AWS IoT Jobs 服務從裝置保留的命名 shadow 中移除現有的軟體套件和套件版本 \$package。如需詳細資訊，請參閱[預留已命名影子](#)。

若要使用此功能，請將 [目標 PackageVersion](#) Amazon 資源名稱 (ARN) 結尾的版本名稱取代為。\$null 之後，您必須指示服務從設備中刪除軟件。

授權的 ARN 使用以下格式：

```
arn:aws:iot:<regionCode>:111122223333:package/<packageName>/version/$null
```

例如

```
$ aws iot create-job \  
  ... \  
  --destinationPackageVersions ["arn:aws:iot:us-east-1:111122223333:package/  
samplePackage/version/$null"]
```



# 開始使用軟體套件目錄

您可以透過 AWS Management Console、AWS IoT Core API 作業和 AWS Command Line Interface (CLI) 來建置和維護 AWS IoT Device Management 軟體 Package 件目AWS CLI錄。

## 使用主控台

若要使用 AWS Management Console，請登入您的 AWS 帳戶並瀏覽至[AWS IoT Core](#)。在導覽窗格中，選擇軟體套件。然後，您可以從本章節建立管理套件及其版本。

## 使用 API 或 CLI 操作

您可以使用 AWS IoT Core API 作業來建立和管理「軟體 Package 件目錄」功能。如需詳細資訊，請參閱 [AWS IoT API 參考](#)和 [AWS SDK 與工具組](#)。這些指 AWS CLI 令也會管理您的目錄。如需詳細資訊，請參閱 [AWS IoT CLI 指令參考](#)。

本章包含下列部分：

- [建立軟體套件和套件版本](#)
- [透過 AWS IoT 工作部署套件版本](#)
- [將封裝版本與物件相關聯 AWS IoT](#)

## 建立軟體套件和套件版本

您可以使用下列步驟，透過 AWS Management Console建立套件和初始版本物件。

若要建立軟體套件

1. 登入您的 AWS 帳戶，然後瀏覽至主[AWS IoT 控制台](#)。
2. 在導覽窗格中，選擇軟體套件。
3. 在 AWS IoT 軟體套件頁面上，選擇建立套件。啟用套件管理的相依性對話方塊隨即出現。
4. 在機群索引下，選取新增裝置軟體套件和版本。這是軟體套件目錄的必要條件，並會提供機群的機群索引和指標。
5. [選擇性] 如果您希望 AWS IoT 工作在工作順利完成時更新保留的命名陰影，請選取「自動更新工作的陰影」。如果您不希望 AWS IoT 工作進行此更新，請不要選取此核取方塊。
6. [選擇性] 若要授與 AWS IoT 工作更新保留的具名陰影的權限，請在「選取角色」下選擇「建立角色」。如果您不希望 AWS IoT 工作進行此更新，則不需要此角色。
7. 建立或選取角色。

- a. 若您沒有此用途的角色：顯示建立角色對話方塊時，請輸入角色名稱，然後選擇建立。
  - b. 若您具有此用途的角色：針對選取角色，請選擇您的角色，然後確定已選取將政策附加至 IAM 角色核取方塊。
8. 選擇確認。建立新套件頁面隨即出現。
  9. 在套件詳細資訊下，輸入套件名稱。
  10. 在套件描述下，輸入相關資訊協助您識別管理此套件。
  11. [選用] 您可以使用標籤來協助您分類管理此套件。若要新增標籤，請展開標籤，選擇新增標籤，然後輸入金鑰-值配對。您最多可輸入 50 個標籤。如需詳細資訊，請參閱[標記 AWS IoT 資源](#)。

若要在建立新套件時新增套裝版本

1. 在第一個版本下，輸入版本名稱。

我們建議您使用[SemVer 格式](#) (例如1.0.0.0) 來唯一識別您的套件版本。您也可以使用更適合您使用案例的其他格式方式。如需詳細資訊，請參閱[套件版本生命週期](#)。

2. 在版本描述下，輸入相關資訊協助您識別管理此套件版本。

#### Note

預設版本核取方塊會停用，因為套件版本是在 draft 狀態下建立的。您可以在建立封裝版本之後以及將狀態變更為之後命名預設版本published。如需詳細資訊，請參閱[套件版本生命週期](#)。

3. [選用] 若要協助您管理此版本或將資訊傳送給您的裝置，請為版本屬性輸入一或多個名稱/值配對。為您輸入的每個名稱/值配對選擇新增屬性。如需詳細資訊，請參閱[版本屬性](#)。
4. [選用] 您可以使用標籤來協助您分類管理此套件。若要新增標籤，請展開標籤，選擇新增標籤，然後輸入金鑰-值配對。您最多可輸入 50 個標籤。如需詳細資訊，請參閱[標記 AWS IoT 資源](#)。
5. 選擇 Create package (建立套件)。AWS IoT 軟體套件頁面隨即出現，且您的套件會列於套件表中。
6. [選用] 若要檢視您所建軟體套件和套件版本的相關資訊，請選擇套件名稱。套件詳細資訊頁面隨即顯示。

## 透過 AWS IoT 工作部署套件版本

您可以使用下列步驟，透過 AWS Management Console部署套件版本。

先決條件：

開始之前，請執行以下動作：

- 註冊的 AWS IoT 東西 AWS IoT Core。如需將裝置新增至的指示 AWS IoT Core，請參閱[建立物件物件](#)。
- [選用] 建立 AWS IoT 物群組或動態物件群組，以鎖定您要部署套件版本的裝置。如需相關指示建立物件群組，請參閱[建立靜態物件群組](#)。如需相關指示建立動態物件群組，請參閱[建立動態物件群組](#)。
- 建立軟體套件和套件版本。如需詳細資訊，請參閱 [建立軟體套件和套件版本](#)。
- 建立任務文件。如需詳細資訊，請參閱[準備任務文件和套件版本以進行部署](#)。

若要部署 AWS IoT 工作

1. 在 [AWS IoT 主控台](#) 上，選擇軟體套件。
2. 選擇您要部署的軟體套件。軟體套件詳細資訊頁面隨即顯示。
3. 在版本之下選擇您要部署的套件版本，然後選擇部署任務版本。
4. 若這是您第一次透過此入口網站部署任務，您會看到附有規範說明的對話方塊。請檢閱資訊，然後選擇確認。
5. 輸入部署操作的名稱，或在名稱欄位保留自動產生的名稱。
6. [選用] 在描述欄位中，輸入識別部署目的或內容的描述，或保留自動產生的資訊。

注意：建議您不要在任務名稱和描述欄位中使用個人識別資訊。

7. [選用] 新增任何要與此任務建立關聯的標籤。
8. 選擇下一步。
9. 在任務目標之下，選擇應負責接收任務的物件或物件群組。
10. 在任務檔案欄位中，指定任務文件 JSON 檔案。
11. 開啟與套件目錄服務整合的任務。
12. 選取任務文件中所指定的套件和版本。

#### Note

您必須選擇在任務文件中所指定的相同套件和套件版本。您可以納入更多項目，但該任務只會針對任務文件中包含的套件和版本發出指示。如需詳細資訊，請參閱[部署時命名套件和版本](#)。

13. 選擇下一步。
14. 在任務組態頁面上，選取任務組態對話方塊中下列任一任務類型：
  - 快照任務：快照任務在目標裝置和群組上完成執行後即完成。
  - 連續任務：連續任務適用於物件群組，並在稍後新增至所指定目標群組的任何裝置上執行。
15. 在其他組態 - 選用對話方塊中，檢閱下列選用任務組態，並視需要進行選擇。如需詳細資訊，請參閱[任務推展、排程和中止組態](#)和[任務執行逾時和重試組態](#)。
  - 推展組態
  - Scheduling configuration (排程組態)
  - 任務執行逾時組態
  - 任務執行重試組態
  - 中止組態
16. 檢視任務選擇，然後選擇送出。

建立任務後，主控台會產生 JSON 簽章，並將其放置在您的任務文件中。您可以使用 AWS IoT 主控台來檢視工作狀態，或取消或刪除工作。若要管理任務，請前往[主控台的任務中心](#)。

## 將封裝版本與物件相關聯 AWS IoT

在裝置上安裝軟體之後，您可以將套件版本與保留名為 shadow 的 AWS IoT 物件建立關聯。如果 AWS IoT 工作已設定為在工作部署後更新已保留的名為 shadow 的物件並順利完成，則不需要完成此程序。如需詳細資訊，請參閱[預留已命名影子](#)。

先決條件：

開始之前，請執行以下動作：

- 創建一個 AWS IoT 東西，或事物，並通過建立遙測 AWS IoT Core。如需詳細資訊，請參閱[開始使用 AWS IoT Core](#)。
- 建立軟體套件和套件版本。如需詳細資訊，請參閱[建立軟體套件和套件版本](#)。
- 在裝置上安裝套件版本軟體。

**Note**

將套件版本與物件相 AWS IoT 關聯並不會在實體裝置上更新或安裝軟體。套件版本必須部署至裝置。

### 欲將封裝版本與 AWS IoT 物件相關聯

1. 在 [AWS IoT 主控台](#) 瀏覽窗格中，展開所有裝置選單，然後選擇物件。
2. 從清單中識別您 AWS IoT 要更新的物件，然後選擇要顯示其詳細資訊頁面的物件名稱。
3. 在詳細資訊區段中，選擇套件和版本。
4. 選擇新增至套件和版本。
5. 針對選擇裝置套件，選擇您想要的軟體套件。
6. 針對選擇版本，選擇您想要的軟體版本。
7. 選擇新增裝置套件。

套件和版本會顯示在所選套件和版本清單中。

8. 針對您要與此物件建立關聯的每個套件和版本重複這些步驟。
9. 完成後，請選擇新增套件和版本詳細資訊。物件詳細資訊頁面隨即開啟，您可以在清單中看到新的套件與版本。

# AWS IoT Core 裝置位置

使用「AWS IoT Core 裝置位置」功能之前，請先檢閱此功能的「條款與條件」。請注意，AWS 可能會將您的地理位置搜尋要求參數 (例如用於執行搜尋的位置資料和其他資訊) 傳送給您選擇的第三方資料提供者，這些參數可能不在您目前使用的資料提供者之外。AWS 區域 如需詳細資訊，請參閱 [AWS 服務條款](#)。

使用 AWS IoT Core 裝置位置來使用協力廠商求解器測試 IoT 裝置的位置。求解器是由第三方提供的演算法，可解析測量資料並估算裝置的位置。藉由識別裝置的位置，您就可以在現場對其進行追蹤和偵錯，進而排解任何疑難問題。

針對收集自各種來源的測量資料進行解析，並將地理位置資訊報告為 [GeoJSON](#) 承載。GeoJSON 格式可用於對地理資料結構進行編碼。承載含有裝置位置的緯度和經度坐標 (以 [世界大地坐標系統 \(WGS84\)](#) 為準)。

## 主題

- [測量類型和求解器](#)
- [AWS IoT Core 裝置位置的運作方式](#)
- [如何使用 AWS IoT Core 裝置位置](#)
- [解析 IoT 裝置的位置](#)
- [使用 AWS IoT Core 裝置位置 MQTT 主題解析裝置位置](#)
- [位置求解器和裝置承載](#)

## 測量類型和求解器

AWS IoT Core 裝置位置與第三方廠商合作，以解析測量資料並提供預估的裝置位置。下表顯示測量類型和第三方位位置求解器，以及支援裝置的相關資訊。如需 LoRa WAN 裝置及其設定裝置位置的相關資訊，請參閱 [設定 LoRa WAN 資源的位置](#)。

### 測量類型和求解器

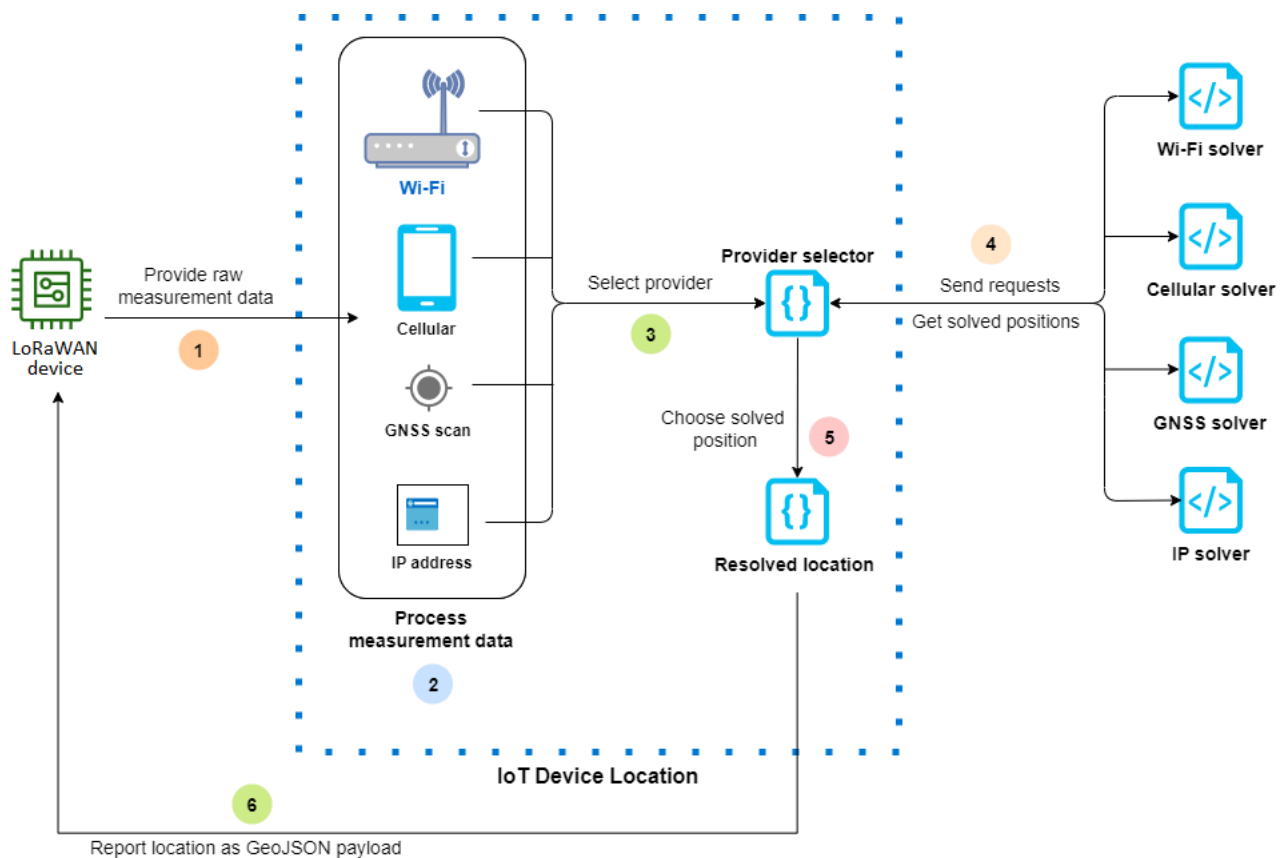
測量類型	第三方求解器	支援的裝置
Wi-Fi 存取點	Wi-Fi 型的求解器	一般 IoT 裝置和 LoRa WAN 裝置

測量類型	第三方求解器	支援的裝置
行動網路無線電塔： GSM、LTE、CDMA、SCDMA 、WCMDA 和 TD-SCDMA 資料	行動網路型求解器	一般 IoT 裝置和 LoRa WAN 裝置
IP 地址	IP 反向查詢求解器	一般 IoT 裝置
GNSS 掃描資料 (NAV 訊息)	GNSS 求解器	一般 IoT 裝置和 LoRa WAN 裝置

如需有關位置求解器的詳細資訊及顯示各種測量類型的裝置承載範例，請參閱 [位置求解器和裝置承載](#)。

## AWS IoT Core 裝置位置的運作方式

下圖顯示「AWS IoT Core 裝置位置」如何收集測量資料並解析裝置的位置資訊。



下列步驟顯示 AWS IoT Core 裝置位置的運作方式。

## 1. 接收測量資料

先從裝置傳送與裝置位置相關的原始測量資料。將測量資料指定為 JSON 承載。

## 2. 處理測量資料

系統會處理量測資料，「AWS IoT Core 裝置位置」會選擇要使用的測量資料，可以是 Wi-Fi、行動數據、GNSS 掃描或 IP 位址資訊。

## 3. 選擇求解器

根據測量資料選擇第三方求解器。例如，如果測量資料包含 Wi-Fi 和 IP 地址資訊，則會選擇 Wi-Fi 求解器和 IP 反向查詢求解器。

## 4. 取得解析的位置

會將 API 要求傳送至求解器提供者，要求解析位置。AWS IoT Core 然後，「裝置位置」會從求解器取得估計的地理位置資訊。

## 5. 選擇解析的位置

系統會比較已解析的位置資訊及其準確性，然後「AWS IoT Core 裝置位置」選擇具有最高精確度的地理位置結果。

## 6. 輸出位置資訊

地理位置資訊會以 GeoJSON 承載的形式傳送給您。承載包含 WGS84 地理座標、準確度資訊、可信度，以及取得解析位置的時間戳記。

# 如何使用 AWS IoT Core 裝置位置

下列步驟顯示如何使用 AWS IoT Core 裝置位置。

## 1. 提供測量資料

將與裝置位置相關的原始測量資料指定為 JSON 承載。要檢索有效負載測量數據，請轉到設備日誌或使用 CloudWatch 日誌，然後複製有效負載數據信息。JSON 承載必須包含一或多種資料測量類型。如需顯示各種求解器承載格式的範例，請參閱 [位置求解器和裝置承載](#)。

## 2. 解析位置資訊

使用主 AWS IoT 控制台或 [GetPositionEstimate](#) API 作業中的「[裝置位置](#)」頁面，傳遞承載量測資料並解析裝置位置。AWS IoT Core 然後，設備位置選擇具有最高精度的求解器並報告設備位置。如需詳細資訊，請參閱 [解析 IoT 裝置的位置](#)。



### 3. 複製位置資訊

驗證由「AWS IoT Core 裝置位置」解析並報告為 GeoJSON 承載的地理位置資訊。您可以複製承載，以便與應用程式和其他 AWS 服務應用程式搭配使用。例如，您可以使用規則動作將地理位置資料傳送至 Amazon [位置](#) AWS IoT 定位服務。

下列主題顯示如何使用 AWS IoT Core 裝置位置以及裝置位置承載的範例。

- [解析 IoT 裝置的位置](#)
- [位置求解器和裝置承載](#)

## 解析 IoT 裝置的位置

使用 AWS IoT Core 裝置位置解碼裝置中的量測資料，並使用協力廠商求解器解析裝置位置。系統會產生解析後的位置作為 GeoJSON 承載，其中包含地理坐標和準確性資訊。您可以從 AWS IoT 主控台、AWS IoT Wireless API 或解析裝置的位置 AWS CLI。

### 主題

- [解析裝置位置 \(主控台\)](#)
- [解析裝置位置 \(API\)](#)
- [對解析位置時發生的錯誤進行疑難排解](#)

## 解析裝置位置 (主控台)

### 若要解析裝置位置 (主控台)

1. 前往主控台內的「[裝置位置](#)」頁 AWS IoT 面。
2. 從您的裝置記錄或記錄中 CloudWatch 取得有效負載測量資料，然後在「透過裝載解決位置」區段中輸入該資料。

以下程式碼顯示 JSON 承載的範例。承載包含行動網路和 Wi-Fi 測量資料。如果您的承載包含其他類型的測量資料，則會使用具有最佳準確度的求解器。如需詳細資訊和承載範例，請參閱 [the section called “位置求解器和裝置承載”](#)。

**Note**

JSON 承載必須包含一或多種資料測量類型。

```
{
  "Timestamp": 1664313161,
  "Ip": {
    "IpAddress": "54.240.198.35"
  },
  "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": [ {
```

```

        "Uarfcnd1": 10786,
        "UtranCid": 14674663,
        "Psc": 149
      },
      {
        "Uarfcnd1": 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. 若要解析位置資訊，請選擇 Resolve (解析)。

位置資訊的類型為 blob，並以採用 GeoJSON 格式的承載形式傳回，此格式可用於對地理資料結構進行編碼。該承載包含：

- WGS84 地理座標，其中包括緯度和經度資訊。其亦可能包括海拔高度資訊。

- 報告的位置資訊類型，例如 Point (點)。點位置類型將位置表示為 WGS84 緯度和經度，並編碼為 [GeoJSON 點](#) 的型態。
- 水平和垂直準確度資訊，指出求解器預估的位置資訊與實際裝置位置之間的差異 (單位公尺)。
- 可信度，此資料表示位置回應估計值的不確定性。預設值為 0.68，表示實際裝置位置處於預估位置不確定半徑內的機率为 68%。
- 裝置所在的城市、州、國家/地區和郵遞區號。只有在使用 IP 反向查詢求解器時，才會報告此資訊。
- 時間戳記資訊對應於解析位置的日期和時間。其使用的是 Unix 時間戳記格式。

以下程式碼顯示藉由解析位置而傳回的 GeoJSON 裝載範例。

#### Note

如果「AWS IoT Core 裝置位置」在嘗試解決位置時報告錯誤，您可以疑難排解錯誤並解決該位置。如需詳細資訊，請參閱 [對解析位置時發生的錯誤進行疑難排解](#)。

```
{
  "coordinates": [
    13.376076698303223,
    52.51823043823242
  ],
  "type": "Point",
  "properties": {
    "verticalAccuracy": 45,
    "verticalConfidenceLevel": 0.68,
    "horizontalAccuracy": 303,
    "horizontalConfidenceLevel": 0.68,
    "country": "USA",
    "state": "CA",
    "city": "Sunnyvale",
    "postalCode": "91234",
    "timestamp": "2022-11-18T12:23:58.189Z"
  }
}
```

- 轉到資源位置部分，並驗證設備位置報告的地理位 AWS IoT Core 置信息。您可以複製承載，以便與其他應用程式搭配使用。AWS 服務例如，您可以使用 [位置](#) 將地理位置資料傳送至 Amazon Location Service。

## 解析裝置位置 (API)

若要使用 AWS IoT Wireless API 解析裝置位置，請使用 [GetPositionEstimate](#) API 作業或 [get-position-estimate](#) CLI 指令。將承載測量資料指定為輸入項，然後執行 API 操作以解析裝置位置。

### Note

GetPositionEstimate API 操作不會儲存任何裝置或狀態資訊，也無法用於擷取歷史位置資料。此功能會執行一次性操作，以解析測量資料並產生位置估計值。若要擷取位置資訊，在每次執行此 API 操作時，都必須指定承載資訊。

下列命令範例顯示如何使用此 API 操作解析位置。

### Note

執行 `get-position-estimate` CLI 命令時，您必須將輸出 JSON 檔案指定為第一個輸入。這個 JSON 檔案會以 GeoJSON 格式儲存從 CLI 回應取得的估計位置資訊。例如，下列命令會將位置資訊儲存在 `locationout.json` 檔案中。

```
aws iotwireless get-position-estimate locationout.json \  
  --ip IpAddress="54.240.198.35" \  
  --wi-fi-access-points \  
    MacAddress="A0:EC:F9:1E:32:C1",Rss=-75 \  
    MacAddress="A0:EC:F9:15:72:5E",Rss=-67
```

此範例包括 Wi-Fi 存取點和 IP 位址做為測量類型。AWS IoT Core 設備位置在 Wi-Fi 求解器和 IP 反向查找求解器之間進行選擇，然後選擇準確度更高的求解器。

系統會將解析的位置傳回為採用 GeoJSON 格式的承載，此格式可用於對地理資料結構進行編碼。然後它儲存在 `locationout.json` 檔案中。承載包含 WGS84 緯度和經度座標、準確度和可信度資訊、位置資料類型，以及解析位置的時間戳記。

```
{
```

```
"coordinates": [
  13.37704086303711,
  52.51865005493164
],
"type": "Point",
"properties": {
  "verticalAccuracy": 707,
  "verticalConfidenceLevel": 0.68,
  "horizontalAccuracy": 389,
  "horizontalConfidenceLevel": 0.68,
  "country": "USA",
  "state": "CA",
  "city": "Sunnyvalue",
  "postalCode": "91234",
  "timestamp": "2022-11-18T14:03:57.391Z"
}
}
```

## 對解析位置時發生的錯誤進行疑難排解

當您嘗試解析位置時，您可能會看到下列任何錯誤碼。AWS IoT Core 使用 `GetPositionEstimate` API 操作時，設備位置可能會產生錯誤，否則請參考 AWS IoT 控制台中與錯誤對應的行號。

### • 400 - 錯誤

此錯誤表示設備有效負載 JSON 的格式無法通過 AWS IoT Core 設備位置進行驗證。發生錯誤的原因可能是：

- JSON 測量資料的格式不正確。
- 裝載只包含時間戳記資訊。
- 測量資料參數 (例如 IP 地址) 無效。

若要解決此錯誤，請檢查您的 JSON 格式是否正確，並包含來自一或多種測量類型的資料作為輸入項。如果 IP 地址無效，請參閱 [IP 反向查詢求解器](#) 以深入了解如何提供有效 IP 地址以解決錯誤。

### • 403 - 錯誤

此錯誤表示您沒有執行 API 操作或使用 AWS IoT 控制台檢索設備位置的權限。若要解決此錯誤，請確認您具有執行此動作所需的許可。如果您的會 AWS Management Console 話或會話令牌已過期，則可能 AWS CLI 會發生此錯誤。若要解決此錯誤，請重新整理工作階段 Token 以使用 AWS CLI，或登出，AWS Management Console 然後使用您的認證登入。

### • 404 - 錯誤

此錯誤表示裝置位置找不到或解決任何位 AWS IoT Core 置資訊。由於輸入的測量資料中可能有資料不足等情況，因此可能會發生錯誤。例如：

- MAC 地址或行動網路基地台的資訊不夠充分。
- 無法使用 IP 地址查詢和擷取位置。
- GNSS 承載不足。

若要解決這種情況下的錯誤，請檢查您的測量資料是否包含解析裝置位置所需的足夠資訊。

- 500 - 錯誤

此錯誤表示 AWS IoT Core 裝置位置功能嘗試解析位置時，內部伺服器發生異常狀況。若要嘗試修正此錯誤，請重新整理工作階段，然後重新嘗試傳送要解析的測量資料。

## 使用 AWS IoT Core 裝置位置 MQTT 主題解析裝置位置

您可以使用保留的 MQTT 主題，透過裝置位置功能取得裝置的最新位 AWS IoT Core 置資訊。

### 裝置位置 MQTT 主題的格式

「AWS IoT Core 裝置位置」的保留主題使用下列前置詞：

```
$aws/device_location/{customer_device_id}/
```

若要建立完整的主題，請先將 *customer\_device\_id* 取代為您用來識別裝置的唯一 ID。我們建議您指定 `WirelessDeviceId`，例如針對 LoRa WAN 和人行道裝置，以及 *thingName* 如果您的裝置已註冊為 AWS IoT 物件。然後，您可以將主題附加到主題 Stub，例如 `get_position_estimate` 或 `get_position_estimate/accepted`，如下節所示。

#### Note

*{customer\_device\_id}* 只能包含英文字母、數字和破折號。訂閱裝置位置主題時，您只能使用加號 (+) 作為萬用字元。例如，您可以針對 *{customer\_device\_id}* 使用 + 萬用字元，以取得您裝置的位置資訊。當您訂閱主題 `$aws/device_location/+/  
get_position_estimate/accepted` 時，如果已成功解析，則會發佈一則訊息，其中包含符合任何裝置 ID 之裝置的位置資訊。

以下是用來與 AWS IoT Core 裝置位置互動的保留主題。

## 裝置位置 MQTT 主題

主題	允許操作	描述
\$aws/device_location/ <i>customer_device_id</i> /get_position_estimate	發佈	裝置會發佈至此主題，以取得由裝置位置解析的掃描原始量測資料。
\$aws/device_location/ <i>customer_device_id</i> /get_position_estimate/accepted	訂閱	AWS IoT Core 裝置位置成功解析裝置位置時，會將位置資訊發佈至此主題。
\$aws/device_location/ <i>customer_device_id</i> /get_position_estimate/rejected	訂閱	AWS IoT Core 裝置位置無法解析裝置位置時，會將錯誤資訊發佈至此主題。

## 裝置位置 MQTT 主題的政策

若要接收來自裝置位置主題的訊息，您的裝置必須使用允許裝置閘道連線至 AWS IoT 裝置閘道並訂閱 MQTT 主題的原則。

以下為接收各種主題訊息所需的政策範例。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate"
      ]
    },
    {
```



```
    "Effect": "Allow",
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate/accepted",
      "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate/rejected"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iot:Subscribe"
    ],
    "Resource": [
      "arn:aws:iot:region:account:topicfilter/$aws/device_location/customer_device_id/get_position_estimate/accepted",
      "arn:aws:iot:region:account:topicfilter/$aws/device_location/customer_device_id/get_position_estimate/rejected"
    ]
  }
]
```

## 裝置位置主題和承載

以下顯示 AWS IoT Core 裝置位置主題、其訊息承載的格式，以及每個主題的範例原則。

### 主題

- [/get\\_position\\_estimate](#)
- [/get\\_position\\_estimate/accepted](#)
- [/get\\_position\\_estimate/rejected](#)

### /get\_position\_estimate

將訊息發佈至此主題，以從裝置取得原始測量資料，以透過「AWS IoT Core 裝置位置」解析。

```
$aws/device_location/customer_device_id/get_position_estimate
```

AWS IoT Core 裝置位置會透過發佈至 [/get\\_position\\_estimate/accepted](#) 或來回應 [/get\\_position\\_estimate/rejected](#)。

### Note

發佈至此主題的訊息必須是有效的 JSON 承載。如果輸入訊息不是有效的 JSON 格式，您將不會收到任何回應。如需詳細資訊，請參閱 [訊息承載](#)。

## 訊息承載

訊息承載格式遵循與 AWS IoT Wireless API 操作請求內文類似的結構 ([GetPositionEstimate](#))。它包含以下內容：

- 選用的 Timestamp 字串，其對應至位置的解析日期和時間。Timestamp 字串的長度最短可以是 1，最長可以是 10。
- 選用的 MessageId 字串，其可以用來將請求對應至回應。如果您指定此字串，發佈至 [get\\_position\\_estimate/accepted](#) 或 [get\\_position\\_estimate/rejected](#) 主題的訊息將會包含此 MessageId。MessageID 字串的長度最短可以是 1，最長可以是 256。
- 來自裝置的測量資料，其中包含下列一或多種測量類型：
  - [WiFiAccessPoint](#)
  - [CellTowers](#)
  - [IpAddress](#)
  - [Gnss](#)

以下顯示範例訊息承載。

```
{
  "Timestamp": "1664313161",
  "MessageId": "ABCD1",
  "WiFiAccessPoints": [
    {
      "MacAddress": "A0:EC:F9:1E:32:C1",
      "Rss": -66
    }
  ],
  "Ip": {
    "IpAddress": "54.192.168.0"
```

```

    },
    "Gnss":{
      "Payload":"8295A614A2029517F4F77C0A7823B161A6FC57E25183D96535E3689783F6CA48",
      "CaptureTime":1354393948
    }
  }
}

```

## 範例 政策

以下為所需政策的範例：

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/get_position_estimate"
      ]
    }
  ]
}

```

## /get\_position\_estimate/accepted

AWS IoT Core 裝置位置會在傳回您裝置的已解析位置資訊時，發佈對此主題的回應。位置資訊會以 [GeoJSON 格式](#) 傳回。

```
$aws/device_location/customer_device_id/get_position_estimate/accepted
```

以下顯示訊息承載和範例政策。

## 訊息承載

以下是 GeoJSON 格式的訊息承載範例。如果您 MessageId 在原始測量資料中指定了 a，且「AWS IoT Core 裝置位置」已成功解析位置資訊，則訊息承載會傳回相同的 MessageId 資訊。

```
{
```

```
"coordinates": [
  13.37704086303711,
  52.51865005493164
],
"type": "Point",
"properties": {
  "verticalAccuracy": 707,
  "verticalConfidenceLevel": 0.68,
  "horizontalAccuracy": 389,
  "horizontalConfidenceLevel": 0.68,
  "country": "USA",
  "state": "CA",
  "city": "Sunnyvale",
  "postalCode": "91234",
  "timestamp": "2022-11-18T14:03:57.391Z",
  "messageId": "ABCD1"
}
}
```

## 範例 政策

以下為所需政策的範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/
device_location/customer_device_id/get_position_estimate/accepted"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/
get_position_estimate/accepted"
      ]
    }
  ]
}
```

```
    ]
  }
]
}
```

## /get\_position\_estimate/rejected

AWS IoT Core 裝置位置無法解析裝置位置時，會針對此主題發佈錯誤回應。

```
$aws/device_location/customer_device_id/get_position_estimate/rejected
```

以下顯示訊息承載和範例政策。如需錯誤的相關資訊，請參閱 [對解析位置時發生的錯誤進行疑難排解](#)。

### 訊息承載

以下是提供錯誤碼和訊息的訊息承載範例，指出 AWS IoT Core 裝置位置無法解析位置資訊的原因。如果您在提供原始測量資料MessageId時指定了 a，且 AWS IoT Core 裝置位置無法解析位置資訊，則MessageId訊息承載中會傳回相同的資訊。

```
{
  "errorCode": 500,
  "errorMessage": "Internal server error",
  "messageId": "ABCD1"
}
```

### 範例 政策

以下為所需政策的範例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:region:account:topicfilter/$aws/
device_location/customer_device_id/get_position_estimate/rejected"
      ]
    }
  ]
}
```

```
    ]
  },
  {
    "Action": [
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:region:account:topic/$aws/device_location/customer_device_id/
      get_position_estimate/rejected"
    ]
  }
]
```

## 位置求解器和裝置承載

位置求解器是可用來解析 IoT 裝置位置的演算法。AWS IoT Core 「裝置位置」支援下列位置求解器。您會看到這些測量類型的 JSON 承載資料格式範例、求解器支援的裝置，以及解析位置的方式。

若要解析裝置位置，請指定一或多個測量資料類型。系統會綜合所有測量資料傳回單一的解析位置。

### 主題

- [基於 Wi-Fi 的求解器](#)
- [行動網路求解器](#)
- [IP 反向查詢求解器](#)
- [求解器](#)

## 基於 Wi-Fi 的求解器

使用 Wi-Fi 解析器依據來自 Wi-Fi 存取點的掃描資訊來解析位置。該求解器支持 WLAN 技術，可用於計算一般 IoT 設備和 LoRa WAN 無線設備的設備位置。

LoRaWAN 裝置必須具有 LoRa Edge 晶片組，才能解碼傳入的 Wi-Fi 掃描資訊。LoRa Edge 是一個超低功耗平台，集成了遠程 LoRa 收發器，多星座 GNSS 掃描儀和被動 Wi-Fi MAC 掃描儀，針對地理定位應用。從裝置收到上行訊息時，Wi-Fi 掃描資料會傳送至 AWS IoT Core 裝置位置，並根據 Wi-Fi 掃描結果估算位置。解碼後的資訊會傳遞到 Wi-Fi 求解器以擷取位置資訊。

## Wi-Fi 求解器承載範例

下列程式碼範例顯示了含有測量資料的裝置所提供的 JSON 承載。當 AWS IoT Core 設備位置接收此數據作為輸入時，它會向求解器提供者發送 HTTP 請求以解析位置信息。若要擷取資訊，請指定 MAC 地址和 RSS (接收訊號強度) 的值。若要這麼做，請使用此格式提供 JSON 承載，或使用 [GetPositionEstimate](#) API 作業的 [WiFiAccessPoints](#) 物件參數。

```
{
  "Timestamp": 1664313161,    // optional
  "WiFiAccessPoints": [
    {
      "MacAddress": "A0:EC:F9:1E:32:C1", // required
      "Rss": -75                    // required
    }
  ]
}
```

## 行動網路求解器

您可以使用行動網路求解器使用從行動網路無線電塔取得的測量資料來解析位置。求解器支援下列技術。即使您納入任何或所有上述技術的測量資料，也只會得出單筆解析位置資訊。

- GSM
- CDMA
- WCDMA
- TD-SCDMA
- LTE

## 行動網路求解器承載範例

下列程式碼範例顯示了含有行動網路測量資料的裝置所提供的 JSON 承載。當 AWS IoT Core 設備位置接收此數據作為輸入時，它會向求解器提供者發送 HTTP 請求以解析位置信息。若要擷取資訊，您可以在主控台中使用此格式提供 JSON 承載，或指定 [GetPositionEstimate](#) API 作業 [CellTowers](#) 參數的值。您可以使用上述任何或所有行動網路技術來指定參數值，藉以提供測量資料。

## LTE (長期演進技術)

使用此測量資料時，您必須指定特定資訊，例如網路或行動網路的國家/地區代碼及其他選擇性參數，包括有關本機 ID 的資訊。下列程式碼顯示承載格式的範例。如需這些參數的詳細資訊，請參閱 [LTE 物件](#)。

```
{
  "Timestamp": 1664313161,          // optional
  "CellTowers": {
    "Lte": [
      {
        "Mcc": int,                  // required
        "Mnc": int,                  // required
        "EutranCid": int,            // required. Make sure that you use int for
EutranCid.
        "Tac": int,                  // optional
        "LteLocalId": {              // optional
          "Pci": int,                // required
          "Earfcn": int,             // required
        },
        "LteTimingAdvance": int,     // optional
        "Rsrp": int,                 // optional
        "Rsrq": float,               // optional
        "NrCapable": boolean,        // optional
        "LteNmr": [                  // optional
          {
            "Pci": int,              // required
            "Earfcn": int,           // required
            "EutranCid": int,        // required
            "Rsrp": int,             // optional
            "Rsrq": float            // optional
          }
        ]
      }
    ]
  }
}
```

## GSM (全球行動通訊系統)

使用此測量資料時，您必須指定特定資訊，例如網路或行動網路的國家/地區代碼、基地台資訊及其他選擇性參數。下列程式碼顯示承載格式的範例。如需這些參數的詳細資訊，請參閱 [GSM 物件](#)。



```

{
  "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 (分碼多重進接)

使用此測量資料時，您必須指定特定資訊，例如訊號功率和識別資訊、基地台資訊及其他選擇性參數。下列程式碼顯示承載格式的範例。如需這些參數的詳細資訊，請參閱 [CDMA 物件](#)。

```

{
  "Timestamp": 1664313161,          // optional
  "CellTowers": {
    "Cdma": [
      {
        "SystemId": int,            // required
        "NetworkId": int,          // required
        "BaseStationId": int,      // required
      }
    ]
  }
}

```



```

        {
            "Uarfcndl": int,        // required
            "Psc": int,            // required
            "UtranCid": int,       // required
            "Rscp": int,           // optional
            "Pathloss": int,       // optional
        }
    ]
}
]
}
}
}

```

### TD-SCDMA (分時-同步分碼多重進接)

使用此測量資料時，您必須指定特定資訊，例如網路與國家/地區代碼、訊號功率和識別資訊、基地台資訊及其他選擇性參數。下列程式碼顯示承載格式的範例。如需這些參數的詳細資訊，請參閱 [CDMA 物件](#)。

```

{
    "Timestamp": 1664313161,      // optional
    "CellTowers": {
        "Tdsdma": [
            {
                "Mcc": int,        // required
                "Mnc": int,        // required
                "UtranCid": int,   // required
                "Lac": int,        // optional
                "TdsdmaLocalId": { // optional
                    "Uarfcn": int, // required
                    "CellParams": int, // required
                },
                "TdsdmaTimingAdvance": int, // optional
                "Rscp": int,         // optional
                "Pathloss": int,     // optional
                "TdsdmaNmr": [      // optional
                    {
                        "Uarfcn": int, // required
                        "CellParams": int, // required
                        "UtranCid": int, // optional
                        "Rscp": int, // optional
                        "Pathloss": int, // optional
                    }
                ]
            }
        ]
    }
}

```

```
    ]
  }
]
}
```

## IP 反向查詢求解器

您可以使用 IP 反向查詢求解器，以 IP 地址作為輸入項來解析位置。求解器可以從已佈建的裝置取得位置資訊。AWS IoT 使用 IPv4 或 IPv6 標準模式格式，或 IPv6 十六進位壓縮模式來指定 IP 地址資訊。然後，您將取得解析位置估計值，包括裝置所在城市和國家/地區等額外資訊。

### Note

使用 IP 反向查詢，即表示您同意不將其用於識別或定位特定家庭或街道地址。

### IP 反向查詢求解器承載範例

下列程式碼範例顯示了含有測量資料的裝置所提供的 JSON 承載。當 AWS IoT Core 裝置位置接收到測量資料中的 IP 位址資訊時，它會在求解器提供者的資料庫中查詢此資訊，然後用來解析位置資訊。若要擷取資訊，請使用此格式提供 JSON 承載，或指定 [GetPositionEstimate](#) API 作業的 `Ip` 參數值。

### Note

使用此求解器時，除了座標之外，還會報告裝置所在的城市、州、國家/地區和郵遞區號。如需範例，請參閱[解析裝置位置 \(主控台\)](#)。

```
{
  "Timestamp": 1664313161,
  "Ip":{
    "IpAddress": "54.240.198.35"
  }
}
```

## 求解器

使用 GNSS (全球導航衛星系統) 求解器，依據 GNSS 掃描結果訊息或 NAV 訊息包含的資訊來擷取裝置位置。您可以選擇性地提供額外的 GNSS 輔助資訊，以減少求解器搜尋訊號時必須使用的變數數

目。藉由提供此輔助資訊 (包括位置、高度以及擷取時間和準確度資訊)，求解器可以輕鬆識別視圖中的衛星並計算裝置位置。

此求解器可用於 LoRa WAN 裝置，以及已佈建的其他裝置。AWS IoT 對於一般 IoT 裝置，如果裝置支援使用 GNSS 進行位置估算，則收發器會在從裝置接收到 GNSS 掃描資訊後解析位置資訊。對於 LoRa WAN 裝置，這些裝置必須具有 LoRa Edge 晶片組。從設備接收到上行鏈路消息時，GNSS 掃描數據將發送到 AWS IoT Core for LoRaWAN，並根據收發器的掃描結果估計該位置。

### GNSS 求解器承載範例

下列程式碼範例顯示了含有測量資料的裝置所提供的 JSON 承載。當 AWS IoT Core 設備位置接收到包含測量數據中有效載荷的 GNSS 掃描信息時，它會使用收發器和包含的任何其他輔助信息來搜索信號並解析位置信息。若要擷取資訊，請使用此格式提供 JSON 承載，或為 [GetPositionEstimate](#) API 作業的 [Gnss](#) 參數指定值。

#### Note

您必須先 AWS IoT Core 從承載中移除目標位置，裝置位置才能解析裝置位置。

```
{
  "Timestamp": 1664313161,           // optional
  "Gnss": {
    "AssistAltitude": number,       // optional
    "AssistPosition": [ number ],   // optional
    "CaptureTime": number,          // optional
    "CaptureTimeAccuracy": number,  // optional
    "Payload": "string",            // required
    "Use2DSolver": boolean          // optional
  }
}
```

# 事件訊息

本節包含更新或變更物件或工作 AWS IoT 時所發佈之郵件的相關資訊。如需有關可讓您建立偵測器以監視裝置故障或作業變更的 AWS IoT Events 服務，以及在動作發生時觸發動作的相關資訊，請參閱[AWS IoT Events](#)。

## 如何產生事件訊息

AWS IoT 發生特定事件時，會發佈事件訊息。例如，在新增、更新或刪除事物時，由登錄檔產生的事件。每個事件都會觸發傳送一則事件訊息。事件訊息會透過具 JSON 承載的 MQTT 發佈。承載內容取決於事件的類型。

### Note

事件訊息保證會發佈一次。事件訊息也有可能發佈超過一次。事件訊息的順序無法保證。

## 接收事件訊息的政策

若要接收事件訊息，您的裝置必須使用適當的政策，以允許裝置連線至 AWS IoT 裝置閘道並訂閱 MQTT 事件主題。您也必須訂閱合適的主題篩選條件。

以下為接收生命週期事件所需的政策範例：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "iot:Subscribe",
      "iot:Receive"
    ],
    "Resource": [
      "arn:aws:iot:region:account:/aws/events/*"
    ]
  }]
}
```

## 啟用事件 AWS IoT

在保留主題的訂閱者可以接收訊息之前，您必須從或使用 API AWS Management Console 或 CLI 啟用事件訊息。如需不同選項管理之事件訊息的相關資訊，請參閱[AWS IoT 事件組態設定表格](#)。

- 若要啟用事件訊息，請移至 AWS IoT 主控台的 [\[設定\]](#) 索引標籤，然後在 [事件型訊息] 區段中選擇 [管理事件]。您可以指定要管理的事件。
- 若要控制使用 API 或 CLI 發佈哪些事件類型，請呼叫 [UpdateEvent設定](#) API 或使用 `update-event-configurations` CLI 命令。例如：

```
aws iot update-event-configurations --event-configurations "{\"THING\":{\"Enabled\":true}}"
```

### Note

所有引號 (") 都會與反斜線 (\) 一起逸出。

您可以呼叫 [設定 API](#) 或使用 `describe-event-configurations` CLI 命令來取得目前的事件組態。[DescribeEvent](#) 例如：

```
aws iot describe-event-configurations
```

### AWS IoT 事件組態設定資料表

事件類別 ( AWS IoT 控制台：設置：基於事件的消息 )	<b>eventConfigurations</b> 金鑰值 (AWS CLI/API)	事件訊息主題
(只能使用 AWS CLI/API 進行設定)	CA_CERTIFICATE	<code>\$aws/events/certificates/registered/<i>caCertificateId</i></code>

事件類別 ( AWS IoT 控制台：設置：基於事件的消息 )	<b>eventConfigurations</b> 金鑰值 (AWS CLI/API)	事件訊息主題
(只能使用 AWS CLI/API 進行設定)	CERTIFICATE	\$aws/events/ presence/connected/ <i>clientId</i>
(只能使用 AWS CLI/API 進行設定)	CERTIFICATE	\$aws/events/ presence/disconnected/ <i>clientId</i>
(只能使用 AWS CLI/API 進行設定)	CERTIFICATE	\$aws/events/subscriptions/subscribed/ <i>clientId</i>
(只能使用 AWS CLI/API 進行設定)	CERTIFICATE	\$aws/events/subscriptions/unsubscribed/ <i>clientId</i>
任務完成，取消	JOB	\$aws/events/ job/ <i>jobID</i> /canceled
任務完成，取消	JOB	\$aws/events/ job/ <i>jobID</i> /cancellation_in_progress
任務完成，取消	JOB	\$aws/events/ job/ <i>jobID</i> /completed
任務完成，取消	JOB	\$aws/events/ job/ <i>jobID</i> /deleted
任務完成，取消	JOB	\$aws/events/ job/ <i>jobID</i> /deletion_in_progress



事件類別 ( AWS IoT 控制台 : 設置 : 基於事件的消息 )	eventConfigurations 金鑰值 (AWS CLI/API)	事件訊息主題
任務執行 : 成功、失敗、拒絕、取消、移除	JOB_EXECUTION	\$aws/events/jobExecution/ <i>jobID</i> /canceled
任務執行 : 成功、失敗、拒絕、取消、移除	JOB_EXECUTION	\$aws/events/jobExecution/ <i>jobID</i> /deleted
任務執行 : 成功、失敗、拒絕、取消、移除	JOB_EXECUTION	\$aws/events/jobExecution/ <i>jobID</i> /failed
任務執行 : 成功、失敗、拒絕、取消、移除	JOB_EXECUTION	\$aws/events/jobExecution/ <i>jobID</i> /rejected
任務執行 : 成功、失敗、拒絕、取消、移除	JOB_EXECUTION	\$aws/events/jobExecution/ <i>jobID</i> /removed
任務執行 : 成功、失敗、拒絕、取消、移除	JOB_EXECUTION	\$aws/events/jobExecution/ <i>jobID</i> /succeeded
任務執行 : 成功、失敗、拒絕、取消、移除	JOB_EXECUTION	\$aws/events/jobExecution/ <i>jobID</i> /timed_out
物件 : 建立 , 更新 , 刪除	THING	\$aws/events/thing/ <i>thingName</i> /created
物件 : 建立 , 更新 , 刪除	THING	\$aws/events/thing/ <i>thingName</i> /updated
物件 : 建立 , 更新 , 刪除	THING	\$aws/events/thing/ <i>thingName</i> /deleted

事件類別 ( AWS IoT 控制台 : 設置 : 基於事件的消息 )	eventConfigurations 金鑰值 (AWS CLI/API)	事件訊息主題
物件群組 : 新增、移除	THING_GROUP	\$aws/events/thingGroup/ <i>thingGroupName</i> /created
物件群組 : 新增、移除	THING_GROUP	\$aws/events/thingGroup/ <i>thingGroupName</i> /updated
物件群組 : 新增、移除	THING_GROUP	\$aws/events/thingGroup/ <i>thingGroupName</i> /deleted
物件群組階層 : 新增、移除	THING_GROUP_HIERARCHY	\$aws/events/thingGroupHierarchy/thingGroup/ <i>parentThingGroupName</i> /childThingGroup/ <i>childThingGroupName</i> /added
物件群組階層 : 新增、移除	THING_GROUP_HIERARCHY	\$aws/events/thingGroupHierarchy/thingGroup/ <i>parentThingGroupName</i> /childThingGroup/ <i>childThingGroupName</i> /removed
物件群組成員資格 : 新增、移除	THING_GROUP_MEMBERSHIP	\$aws/events/thingGroupMembership/thingGroup/ <i>thingGroupName</i> /thing/ <i>thingName</i> /added

事件類別 ( AWS IoT 控制台：設置：基於事件的消息 )	eventConfigurations 金鑰值 (AWS CLI/API)	事件訊息主題
物件群組成員資格：新增、移除	THING_GROUP_MEMBERSHIP	\$aws/events/thingGroupMembership/thingGroup/ <i>thingGroupName</i> /thing/ <i>thingName</i> /removed
物件類型：建立、更新、刪除	THING_TYPE	\$aws/events/thingType/ <i>thingTypeName</i> /created
物件類型：建立、更新、刪除	THING_TYPE	\$aws/events/thingType/ <i>thingTypeName</i> /updated
物件類型：建立、更新、刪除	THING_TYPE	\$aws/events/thingType/ <i>thingTypeName</i> /deleted
物件類型關聯：新增、移除	THING_TYPE_ASSOCIATION	\$aws/events/thingTypeAssociation/thing/ <i>thingName</i> /thingType/ <i>thingTypeName</i> /added  \$aws/events/thingTypeAssociation/thing/ <i>thingName</i> /thingType/ <i>thingTypeName</i> /removed

## 登錄檔事件

該登錄檔可能會在建立、更新或刪除物件、物件類型和物件群組時發佈事件訊息。不過，這些事件預設為無法使用。如需如何開啟這些事件的詳細資訊，請參閱 [啟用事件 AWS IoT](#)。

此登錄檔可以提供下列事件類型：

- [物件事件](#)
- [物件類型事件](#)
- [物件群組事件](#)

### 物件事件

建立/更新/刪除的物件

該登錄檔會在建立、更新或刪除事物時發佈以下事件訊息：

- `$aws/events/thing/thingName/created`
- `$aws/events/thing/thingName/updated`
- `$aws/events/thing/thingName/deleted`

訊息包含的承載範例如下：

```
{
  "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"
  }
}
```

承載包含以下屬性：

eventType

設定為「THING\_EVENT」。

eventId

獨特的事件 ID (字串)。

timestamp

事件發生時的 UNIX 時間戳記。

operation

觸發事件的操作。有效的 值如下：

- CREATED (已建立)
- UPDATED (已更新)
- DELETED

accountId

您的 AWS 帳戶 身份證。

thingId

建立、更新或刪除的物件之 ID。

thingName

建立、更新或刪除的物件之名稱。

versionNumber

建立、更新或刪除的物件之版本。物件建立時，此值設定為 1。物件更新時，此值會增加 1。

事情 TypeName

與物件關聯的物件類型 (若存在)。否則為 null。

屬性

與物件關聯的名稱/值對之集合。

## 物件類型事件

物件類型相關事件：

- [建立/棄用/取消棄用/刪除物件類型](#)
- [關聯事物類型或取消關聯事物](#)

## 建立/棄用/取消棄用/刪除物件類型

該登錄檔會在建立、棄用、取消棄用或刪除事物類型時發佈以下事件訊息：

- `$aws/events/thingType/thingTypeName/created`
- `$aws/events/thingType/thingTypeName/updated`
- `$aws/events/thingType/thingTypeName/deleted`

訊息包含的承載範例如下：

```
{
  "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"
}
```

承載包含以下屬性：

`eventType`

設定為「THING\_TYPE\_EVENT」。

`eventId`

獨特的事件 ID (字串)。

`timestamp`

事件發生時的 UNIX 時間戳記。

## operation

觸發事件的操作。有效的 值如下：

- CREATED (已建立)
- UPDATED (已更新)
- DELETED

## accountId

您的 AWS 帳戶 身份證。

## thingTypeId

建立、棄用或刪除的物件類型之 ID。

## thing TypeName

建立、棄用或刪除的物件類型之名稱。

## isDeprecated

如果指定物件類型已棄用，則為 true。否則為 false。

## deprecationDate

棄用物件類型的 UNIX 時間戳記。

## searchableAttributes

與可用於搜尋的物件類型關聯之名稱/值對集合。

## description

物件類型描述。

## 關聯事物類型或取消關聯事物

當事物類型與事物關聯或取消關聯時，登錄檔會發佈以下事件訊息。

- \$aws/events/thingTypeAssociation/thing/*thingName*/thingType/*typeName*/added
- \$aws/events/thingTypeAssociation/thing/*thingName*/thingType/*typeName*/removed

以下是 added 承載的範例。removed 訊息的承載類似。

```
{
  "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,
}
```

承載包含以下屬性：

eventId

獨特的事件 ID (字串)。

eventType

設定為 "THING\_TYPE\_ASSOCIATION\_EVENT"。

operation

觸發事件的操作。有效的 值如下：

- ADDED
- REMOVED

thingId

類型關聯已變更的物件 ID。

thingName

類型關聯已變更的物件名稱。

事情 TypeName

與物件相關聯或不再相關聯的物件類型。

timestamp

事件發生時的 UNIX 時間戳記。

## 物件群組事件

物件群組相關事件：



- [建立/更新/刪除物件群組](#)
- [從事物群組新增或移除的事物](#)
- [從事物群組新增或移除的事物群組](#)

## 建立/更新/刪除物件群組

登錄檔會在建立、更新或刪除事物群組時發佈以下事件訊息。

- `$aws/events/thingGroup/groupName/created`
- `$aws/events/thingGroup/groupName/updated`
- `$aws/events/thingGroup/groupName/deleted`

以下是 updated 承載的範例。created 和 deleted 訊息的承載類似。

```
{
  "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_level11",
      "groupId": "bc1643e1-5a85-4eac-b45a-92509cbe2a77"
    },
    {
      "groupArn": "arn:aws:iot:us-west-2:571EXAMPLE833:thinggroup/Tg_level12",
      "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
}
```

承載包含以下屬性：

eventType

設定為「THING\_GROUP\_EVENT」。

eventId

獨特的事件 ID (字串)。

timestamp

事件發生時的 UNIX 時間戳記。

operation

觸發事件的操作。有效的 值如下：

- CREATED (已建立)
- UPDATED (已更新)
- DELETED

accountId

您的 AWS 帳戶 身份證。

事情 GroupId

建立、更新或刪除的物件群組之 ID。

## 事情 GroupName

建立、更新或刪除的物件群組之名稱。

## versionNumber

物件群組版本。物件群組建立時，此值設定為 1。物件群組更新時，此值會增加 1。

## 父母 GroupName

父事物群組名稱 (若有的話)。

## 父母 GroupId

父事物群組 ID (若有的話)。

## description

物件群組描述。

## 根 ToParent ThingGroups

關於父物件群組的一系列資訊。每個父物件群組都有一個元素，從根物件群組開始，一直到物件群組的父項。每個項目都包含物件群組的 `groupArn` 和 `groupId`。

## 屬性

與物件群組關聯的名稱/值對之集合。

## 從事物群組新增或移除的事物

該登錄檔會在事物新增至事物群組，或從事物群組移除時發佈以下事件訊息。

- `$aws/events/thingGroupMembership/thingGroup/thingGroupName/thing/thingName/added`
- `$aws/events/thingGroupMembership/thingGroup/thingGroupName/thing/thingName/removed`

訊息包含的承載範例如下：

```
{
  "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"
}
```

承載包含以下屬性：

eventType

設定為「THING\_GROUP\_MEMBERSHIP\_EVENT」。

eventId

事件 ID。

timestamp

事件發生時的 UNIX 時間戳記。

operation

當物件新增到物件群組時，ADDED。當物件從物件群組移除時，REMOVED。

accountId

您的 AWS 帳戶 身份證。

groupArn

物件群組的 ARN。

groupId

群組的 ID。

thingArn

物件群組新增或移除之物件的 ARN。

thingId

物件群組新增或移除之物件的 ID。

## membershipId

代表物件和物件群組之間關係的 ID。當您在物件群組新增物件，就會產生此值。

## 從事物群組新增或移除的事物群組

當一個事物群組新增至另一個事物群組，或從另一個事物群組中移除事物群組時，登錄檔都會發佈以下事件訊息。

- `$aws/events/thingGroupHierarchy/thingGroup/parentThingGroupName/childThingGroupName/added`
- `$aws/events/thingGroupHierarchy/thingGroup/parentThingGroupName/childThingGroupName/removed`

訊息包含的承載範例如下：

```
{
  "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"
}
```

承載包含以下屬性：

### eventType

設定為「THING\_GROUP\_HIERARCHY\_EVENT」。

### eventId

事件 ID。

### timestamp

事件發生時的 UNIX 時間戳記。

## operation

當物件新增到物件群組時，ADDED。當物件從物件群組移除時，REMOVED。

## accountId

您的 AWS 帳戶 身份證。

## 事情 GroupId

父物件群組的 ID。

## 事情 GroupName

父物件群組的名稱。

## 兒童 GroupId

子物件群組的 ID。

## 兒童 GroupName

子物件群組的名稱。

## 任務事件

當 AWS IoT 工作擱置、完成或取消，以及裝置在執行作業時報告成功或失敗時，Jobs 服務會發佈至 MQTT 通訊協定上的保留主題。藉由訂閱這些主題，裝置或管理及監控應用程式就能夠追蹤任務的狀態。

### 如何啟用任務事件

來自 AWS IoT Jobs 服務的回應訊息不會透過訊息代理程式傳遞，而且無法由其他用戶端或規則訂閱這些訊息。若要訂閱任務活動相關的訊息，請使用 `notify` 和 `notify-next` 主題。如需任務主題的詳細資訊，請參閱 [任務主題](#)。

若要收到工作更新的通知，請使用 AWS Management Console、或使用 API 或 CLI 來啟用這些工作事件。如需詳細資訊，請參閱 [啟用事件 AWS IoT](#)。

### 任務事件的運作方式

由於取消和刪除任務可能需要一些時間，系統會傳送兩個訊息來表示請求開始和結束。例如，當取消請求開始時，系統會將訊息傳送到 `$aws/events/job/jobID/cancellation_in_progress` 主題。當取消請求完成時，系統會將訊息傳送到 `$aws/events/job/jobID/canceled` 主題。

任務刪除請求的程序亦同。管理及監控應用程式可以藉由訂閱這些主題，以追蹤任務的狀態。如需發佈與訂閱 MQTT 主題的詳細資訊，請參閱 [the section called “裝置通訊協定”](#)。

## 任務事件類型

下文顯示不同類型的任務事件：

### 完成/取消/刪除的任務

當 AWS IoT 工作完成、取消、刪除或取消或刪除正在進行時，Jobs 服務會在 MQTT 主題上發佈訊息：

- `$aws/events/job/jobID/completed`
- `$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 訊息包含的承載範例如下：

```
{
  "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 訊息包含的承載範例如下。

```
{  
  "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 訊息包含的承載範例如下。

```
{  
  "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` 訊息包含的承載範例如下：

```
{
  "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` 訊息包含的承載範例如下：

```
{
  "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"  
}
```

## 任務執行結束狀態

當設備將作業執行更新為終端狀態時，J AWS IoT obs 服務會發布一條消息：

- \$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

訊息包含的承載範例如下：

```
{  
  "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"  
  }  
}
```

## 生命週期事件

AWS IoT 可以發佈 MQTT 主題的生命週期事件。預設情況下，這些事件可用且無法停用。

### Note

生命週期的訊息可能不會按照順序傳送。您可能會收到重複的訊息。

在本主題中：

- [連線/中斷連線事件](#)
- [訂閱/取消訂閱事件](#)

## 連線/中斷連線事件

### Note

透過 AWS IoT 裝置管理叢集索引，您可以搜尋物件、執行彙總查詢，以及根據物件連線/中斷連線事件建立動態群組。如需詳細資訊，請參閱[機群索引](#)。

AWS IoT 當用戶端連線或中斷連線時，將訊息發佈至下列 MQTT 主題：

- `$aws/events/presence/connected/clientId`：用戶端已連接到訊息代理程式。
- `$aws/events/presence/disconnected/clientId`：用戶端已與訊息代理程式中斷連線。

以下為發佈至 `$aws/events/presence/connected/clientId` 主題之連線/中斷連線訊息所包含的 JSON 元素。

`clientId`

連線或中斷連線之用戶端的用戶端 ID。

### Note

包含 # 或 + 的用戶端 ID 不會收到生命週期事件。

客戶端 `InitiatedDisconnect`

如果用戶端已啟動中斷連線，則為 `True`。否則為 `false`。僅在中斷連線訊息中發現。

`disconnectReason`

用戶端中斷連線的原因。僅在中斷連線訊息中發現。下表包含有效值，以及代理程式是否會在連線中斷時傳送 [Last Will and Testament \(LWT\) 訊息](#)。

中斷連線原因	描述	代理程式將傳送 LWT 訊息
AUTH_ERROR	用戶端驗證失敗或授權失敗。	是。如果裝置收到此錯誤之前有作用中連線。
CLIENT_INITIATED_DISCONNECT	用戶端表示將中斷連線。用戶端可以傳送 MQTT DISCONNECT 控制封包，或者如果用戶端正在使用連線，Close frame則可以執行此操作 WebSocket。	否。
CLIENT_ERROR	用戶端執行錯誤操作導致其中斷連線。例如，若在相同連線上傳送超過 1 個 MQTT CONNECT 封包，或者用戶端嘗試以超過承載限制的承載發佈，用戶端將會中斷連線。	是。
CONNECTION_LOST	用戶端伺服器連線已中斷。這可能會在高網路延遲期間或網際網路連線中斷時發生。	是。
DUPLICATE_CLIENTID	用戶端使用已在使用中的用戶端 ID。在這種情況下，已經連線的用戶端會因為此中斷連線的原因而中斷連線。	是。
FORBIDDEN_ACCESS	用戶端不允許連線。例如，具有連線失敗的 IP 地址用戶端將無法連線。	是。如果裝置收到此錯誤之前有作用中連線。
MQTT_KEEP_ALIVE_TIMEOUT	如果用戶端持續作用時間的 1.5 倍沒有用戶端伺服器通訊，則用戶端會中斷連線。	是。
SERVER_ERROR	因未預期的伺服器問題而中斷連線。	是。
SERVER_INITIATED_DISCONNECT	因為操作原因導致伺服器故意中斷用戶端的連線。	是。

中斷連線原因	描述	代理程式將傳送 LWT 訊息
THROTTLED	因為超過調節限制導致用戶端中斷連線。	是。
WEBSOCKET_TTL_EXPIRATION	用戶端 WebSocket 已中斷連線，因為 a 的連線時間超過其 time-to-live 值。	是。
CUSTOMAUTH_TTL_EXPIRATION	用戶端已中斷連線，因為其連線時間超過其自訂授權者的 time-to-live 值。	是。

### eventType

事件的類型。有效值為 `connected` 或 `disconnected`。

### ipAddress

連線用戶端的 IP 地址。這可以是 IPv4 或 IPv6 格式。僅在連線訊息中發現。

### principalIdentifier

用於身分驗證的憑證。對於 TLS 交互身分驗證憑證，此為憑證 ID。對於其他連線，此為 IAM 憑證。

### sessionIdIdentifier

在工作階段生命週期內存在的全域唯一識別碼。AWS IoT

### timestamp

事件發生的約略時間。

### versionNumber

生命週期事件的版本號碼。這是每個用戶端 ID 連線的依序遞增長整數值。訂閱者戶可以使用版本號碼來推斷生命週期事件的順序。

#### Note

用戶端連線的連線和中斷連線訊息具有相同的版本號碼。  
版本號碼可能會略過值，並且每個事件並不保證會以 1 為值持續增加。

如果用戶端約一小時未連線，則版本號碼將重設為 0。對於持續工作階段，當用戶端中斷連線的時間超過持續性工作階段所設定的 time-to-live (TTL) 時，版本號碼便會重設為 0。

連線訊息具有以下結構。

```
{
  "clientId": "186b5",
  "timestamp": 1573002230757,
  "eventType": "connected",
  "sessionId": "a4666d2a7d844ae4ac5d7b38c9cb7967",
  "principalIdentifier": "12345678901234567890123456789012",
  "ipAddress": "192.0.2.0",
  "versionNumber": 0
}
```

中斷連線訊息具有以下結構。

```
{
  "clientId": "186b5",
  "timestamp": 1573002340451,
  "eventType": "disconnected",
  "sessionId": "a4666d2a7d844ae4ac5d7b38c9cb7967",
  "principalIdentifier": "12345678901234567890123456789012",
  "clientInitiatedDisconnect": true,
  "disconnectReason": "CLIENT_INITIATED_DISCONNECT",
  "versionNumber": 0
}
```

## 處理用戶端中斷連線

最佳實務是一律為生命週期事件實作等待狀態，包括 [Last Will and Testament \(LWT\) 訊息](#)。收到中斷連線的訊息時，您的程式碼會等待一段時間並驗證裝置是否仍然離線，才採取動作。使用 [SQS 延遲佇列](#) 是執行此操作的其中一種方法。當用戶端收到 LWT 或生命週期事件時，您可以將訊息加入佇列 (例如，5 秒鐘)。當該訊息變為可用並由 Lambda 或其他服務進行處理時，您可以在進行下一個動作前，先檢查該裝置是否仍處於離線狀態。

## 訂閱/取消訂閱事件

AWS IoT 當客戶訂閱或取消訂閱 MQTT 主題時，會將訊息發佈至下列 MQTT 主題：

```
$aws/events/subscriptions/subscribed/clientId
```

或

```
$aws/events/subscriptions/unsubscribed/clientId
```

其中 `clientId` 是連線至 AWS IoT 訊息代理程式的 MQTT 用戶端 ID。


發佈至此主題的訊息具有以下結構：

```
{
  "clientId": "186b5",
  "timestamp": 1460065214626,
  "eventType": "subscribed" | "unsubscribed",
  "sessionId": "00000000-0000-0000-0000-000000000000",
  "principalIdentifier": "000000000000/ABCDEFGHIJKLMNQRSTU:some-user/
ABCDEFGHIJKLMNQRSTU:some-user",
  "topics" : ["foo/bar","device/data","dog/cat"]
}
```

以下為發佈至 `$aws/events/subscriptions/subscribed/clientId` 和 `$aws/events/subscriptions/unsubscribed/clientId` 主題之訂閱與取消訂閱訊息所包含的 JSON 元素。

`clientId`

訂閱或取消訂閱之用戶端的用戶端 ID。

 Note

包含 # 或 + 的用戶端 ID 不會收到生命週期事件。

`eventType`

事件的類型。有效值為 `subscribed` 或 `unsubscribed`。

`principalIdentifier`

用於身分驗證的憑證。對於 TLS 交互身分驗證憑證，此為憑證 ID。對於其他連線，此為 IAM 憑證。

## sessionId

在工作階段生命週期內存在的全域唯一識別碼。AWS IoT

## timestamp

事件發生的約略時間。

## topics


用戶端已訂閱的一系列 MQTT 主題。

### Note

生命週期的訊息可能不會按照順序傳送。您可能會收到重複的訊息。



# 疑難排 AWS IoT

 協助我們改善此主題


[讓我們知道如何能使其變得更好](#)

以下資訊可能有助於診斷 AWS IoT 內的常見問題。

## 任務

- [AWS IoT Core 疑難排解指](#)
- [AWS IoT 裝置顧問疑難排解指](#)
- [AWS IoT Device Management 疑難排解指](#)
- [AWS IoT 錯誤](#)

## AWS IoT Core 疑難排解指

 協助我們改善此主題

[讓我們知道如何能使其變得更好](#)

這是的疑難排解一節 AWS IoT Core。

## 主題

- [診斷連線問題](#)
- [診斷規則問題](#)
- [診斷影子的問題](#)
- [診斷 Salesforce IoT 輸入串流動作問題](#)
- [診斷串流限制](#)
- [對裝置機群中斷連線進行疑難排解](#)

## 診斷連線問題

 協助我們改善此主題

[讓我們知道如何能使其變得更好](#)

成功的連線 AWS IoT 需要：

- 有效的連線
- 有效且作用中的憑證
- 允許所需連線和作業的政策

### 連線

如何找到正確的端點？

- `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`

如何找到正確的伺服器名稱指示 (SNI) 值？

正確的 SNI 值是 [describe-endpoint](#) 傳回的 `endpointAddress`，或是 [describe-domain-configuration](#) 命令傳回的 `domainName`。其地址與上一個步驟中的端點相同。將裝置連線到時 AWS IoT Core，用戶端可以傳送 [伺服器名稱指示 \(SNI\) 延伸模組](#)，這不是必要的，但強烈建議使用。您必須使用 SNI 擴充功能，才能使用如 [多帳戶註冊](#)、[自訂網域](#) 及 [VPC 端點](#) 等功能。如需詳細資訊，請參閱 [中的傳輸安全性 AWS IoT](#)。

如何解決持續存在的連線問題？

您可以使用 AWS 裝置建議程式協助疑難排解。Device Advisor 的預先建置測試可協助您針對最佳實務，驗證您的裝置軟體是否使用了 [TLS](#)、[MQTT](#)、[AWS IoT Device Shadow](#)，以及 [AWS IoT 工作](#)。

這是現有 [Device Advisor](#) 內容的連結。

## 身分驗證

裝置必須經過驗證才能連線到 AWS IoT 端點。對於用於驗證的裝置，憑證必須在中註冊 AWS IoT 並處 [X.509 用戶端憑證](#) 於作用中狀態。

我的裝置如何驗證 AWS IoT 端點？

將 AWS IoT CA 憑證新增至用戶端的信任存放區。請參閱 [AWS IoT Core 中的伺服器驗證](#) 文件，然後遵循連結下載適當的憑證授權機構憑證。

設備連接到時檢查什麼 AWS IoT？

當裝置嘗試連接至 AWS IoT 時：

1. AWS IoT 檢查是否有效的憑證和伺服器名稱指示 (SNI) 值。
2. AWS IoT 檢查使用的憑證是否已在「AWS IoT 帳戶」中註冊，以及是否已啟用。
3. 當裝置嘗試在中 AWS IoT 執行任何動作 (例如訂閱或發佈訊息) 時，系統會檢查附加至其連線憑證的原則，以確認裝置已獲授權執行該動作。

如何驗證憑證是否正確設定？

請使用 OpenSSL `s_client` 命令來測試與 AWS IoT 端點的連線：

```
openssl s_client -connect custom_endpoint.iot.aws-region.amazonaws.com:8443 -  
CAfile CA.pem -cert cert.pem -key privateKey.pem
```

如需使用 `openssl s_client` 的詳細資訊，請參閱 [OpenSSL s\\_client 文件](#)。

如何檢查憑證的狀態？

- 列出憑證

若您不知道憑證 ID，您可使用 `aws iot list-certificates` 命令，查看您所有憑證的狀態。

- 顯示憑證的詳細資訊

若您知道憑證的 ID，此命令會顯示更多有關憑證的詳細資訊。

```
aws iot describe-certificate --certificate-id "certificateId"
```

- 在 AWS IoT 主控台中檢閱憑證

在 [AWS IoT 主控台](#) 的左側選單中，依序選擇 Secure (安全) 和 Certificates (憑證)。

從清單中選擇您用來連線的憑證，開啟其詳細資訊頁面。

於憑證的詳細資訊頁面中，您可查看其目前的狀態。

可使用詳細資訊頁面右上角的 Actions (動作) 選單來變更憑證的狀態。

## 授權

AWS IoT 資源用 [AWS IoT Core 政策](#) 於授權這些資源以執行 [動作](#)。若要獲得授權的動作，指定的 AWS IoT 資源必須附加一份原則文件，以授與執行該動作的權限。

我從代理程式收到 PUBNACK 或 SUBNACK 的回應？我要怎麼做？

請確定您用來呼叫的憑證上有附加原則 AWS IoT。根據預設，所有發佈/訂閱的操作都會遭拒。

請確定連接的政策會授權您正嘗試執行的 [動作](#)。

請確定連接的政策會授權您正嘗試執行授權動作的 [資源](#)。

我的日誌中有一則 AUTHORIZATION\_FAILURE 項目。

請確定您用來呼叫的憑證上有附加原則 AWS IoT。根據預設，所有發佈/訂閱的操作都會遭拒。

請確定連接的政策會授權您正嘗試執行的 [動作](#)。

請確定連接的政策會授權您正嘗試執行授權動作的 [資源](#)。

如何查看政策授權的內容？

在 [AWS IoT 主控台](#) 的左側選單中，依序選擇 Secure (安全) 和 Certificates (憑證)。

從清單中選擇您用來連線的憑證，開啟其詳細資訊頁面。

於憑證的詳細資訊頁面中，您可查看其目前的狀態。

於憑證詳細資訊頁面的左側選單中，選擇 Policies (政策)，查看連接至憑證的政策。

選擇想要的政策以查看其詳細資訊頁面。

於政策的詳細資訊頁面中，檢閱政策的政策文件以查看其授權內容。


選擇 Edit policy document (編輯政策文件)，對政策文件進行變更。

## 安全性與身分

當您為 AWS IoT 自訂網域組態提供伺服器憑證時，憑證最多有四個網域名稱。

如需詳細資訊，請參閱 [AWS IoT Core 端點和配額](#)。

## 診斷規則問題

 協助我們改善此主題

[讓我們知道如何能使其變得更好](#)

本節說明當您遇到規則問題時，要檢查的一些事項。

### 設定疑難排解的 CloudWatch 記錄

調試規則遇到的問題的最佳方法是使用 CloudWatch 日誌。當您啟用 CloudWatch 記錄檔時 AWS IoT，您可以查看哪些規則已觸發，以及其成功或失敗。也會獲得 WHERE 子句條件是否符合的資訊。如需詳細資訊，請參閱 [AWS IoT 使用 CloudWatch 記錄監控](#)。

最常見的問題在於授權。記錄會顯示您的角色是否未獲授權在資源 AssumeRole 上執行。以下為 [精細記錄](#) 所產生的日誌範例：

```
{
  "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": "ABCDEFGH1234567ABCD890: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)"
}
```

以下為[全域記錄](#)所產生的類似日誌範例：

```
2017-12-09 22:49:17.954 TRACEID:ff562535-6964-506a-e141-78d40375fc4e
PRINCIPALID:ABCDEFGH1234567ABCD890: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.
```

如需詳細資訊，請參閱 [the section called “在 CloudWatch 主控台中檢視 AWS IoT 記錄”](#)。

## 診斷外部服務

外部服務受最終使用者的控制。在執行規則之前，請確認您連結至規則的外部服務已經設定，並為您的應用程式提供足夠的輸送量和容量單位。

## 診斷 SQL 問題

若您的 SQL 查詢並未傳回您想要的資料：

- 請檢閱錯誤訊息的日誌。
- 確認您的 SQL 語法符合訊息中的 JSON 文件。

檢閱用於查詢中的物件和屬性名稱，及主題訊息承載的 JSON 文件中所使用名稱。如需 SQL 查詢中 JSON 格式的詳細資訊，請參閱 [JSON Extensions](#)。

- 檢查 JSON 物件或屬性名稱是否包含保留或數字字元。

如需 SQL 查詢中 JSON 物件參考中保留字元的詳細資訊，請參閱 [JSON Extensions](#)。

## 診斷影子的問題

 協助我們改善此主題


[讓我們知道如何能使其變得更好](#)

### 診斷影子

問題	準則疑難排解
裝置的影子文件遭受 Invalid JSON document 拒絕。	如果您不熟悉 JSON，請修改此指南中提供的範本供自行使用。如需詳細資訊，請參閱 <a href="#">影子文件範例</a> 。
我已提交正確的 JSON，但該檔案完全未存放於裝置的影子文件中，或僅存放了部分。	請確認您是否有按照 JSON 格式編排準則進行。僅有 desired 和 reported 內的 JSON 欄位會儲存。在這些部分之外的 JSON 內容 (即使格式正確) 會被忽略。
我收到裝置影子超出允許大小的錯誤。	裝置影子僅支援 8 KB 的資料。請嘗試縮短您 JSON 文件內的欄位名稱，或直接建立更多物件，以建立更多的影子。可與裝置相關聯的物件/影子數目，並不受限制。唯一的要求是，每個物件名稱在您的帳戶中皆不得重複。
當我收到裝置的影子時，其大於 8 KB。怎麼會出現這種情況？	收到後，AWS IoT 服務會將元數據添加到設備的陰影中。該服務會在回應中包含此資料，而不會計入 8 KB 的限制之中。只有傳送至裝置影子的狀態文件內之 desired 與 reported 狀態的資料，才會計入限制。
我的請求因為版本錯誤而遭拒。我該怎麼辦？	請執行 GET 操作，以同步到文件的最新版本。當使用 MQTT 時，訂閱 <code>./update/accepted</code> 主

問題	準則疑難排解
	題，以接收狀態變更的通知和 JSON 文件的最新版本。
時間戳記出現幾秒鐘的差距。	當 AWS IoT 服務接收文件或將狀態文件發行到上時，個別欄位的時間戳記和整個 JSON 文件會更新。/更新/接受和。/更新/增量消息。網路傳遞的訊息可能會延遲幾秒，而導致時間戳記有幾秒鐘的差距。
我的裝置可發佈於對應的影子主題且已訂閱該主題，但當我嘗試透過 HTTP REST API 更新影子文件時，卻出現 HTTP 403。	請確認已在 IAM 中建立政策，以存取這類主題及您所使用憑證相應的動作 (UPDATE/GET/DELETE)。IAM 政策與憑證政策彼此獨立。
其他問題。	Device Shadow 服務會將錯誤記 CloudWatch 錄到記錄檔。若要識別裝置和設定問題，請啟動 CloudWatch 記錄檔並檢視記錄檔以取得偵錯資訊。

## 診斷 Salesforce IoT 輸入串流動作問題

 協助我們改善此主題

[讓我們知道如何能使其變得更好](#)

### 執行追蹤

如何查看 Salesforce 動作的執行追蹤？

請參閱 [AWS IoT 使用 CloudWatch 記錄監控](#) 一節。啟動日誌後，即可看見 Salesforce 動作的執行追蹤。



## 作業成功和失敗

該如何確認訊息已成功傳送至 Salesforce IoT input 串流？

檢視「記錄檔」中執行 Salesforce 動作所產生的 CloudWatch 記錄檔。如果您看到 Action executed successfully，則表示 AWS IoT 規則引擎收到來自 Salesforce IoT 的確認訊息已成功推送至目標輸入串流。

如果您遭遇到 Salesforce IoT 平台的任何問題，請聯絡 Salesforce IoT 支援。

若訊息並未成功傳送至 Salesforce IoT input 串流，我該怎麼辦？

檢視「記錄檔」中執行 Salesforce 動作所產生的 CloudWatch 記錄檔。您可以根據日誌項嘗試以下動作：

Failed to locate the host

請確認該動作的 url 參數是否正確，以及您的 Salesforce IoT 輸入串流是否存在。

Received Internal Server Error from Salesforce

請重試。如果問題仍存在，請聯絡 Salesforce IoT 支援部門。

Received Bad Request Exception from Salesforce

請檢查您傳送的承載是否有錯誤。

Received Unsupported Media Type Exception from Salesforce

Salesforce IoT 目前不支援二位元承載。請確認您傳送得是否為 JSON 承載。

Received Unauthorized Exception from Salesforce

請確認該動作的 token 參數是否正確，以及您的字符是否仍有效。

Received Not Found Exception from Salesforce

請確認該動作的 url 參數是否正確，以及您的 Salesforce IoT 輸入串流是否存在。

如果您收到此處未列出的錯誤訊息，請聯絡 Sup AWS IoT port 部門。

## 診斷串流限制

疑難排解「超出 AWS 帳戶的串流限制」

如果您看到 "Error: You have exceeded the limit for the number of streams in your AWS account."，您可以清除帳戶中未使用的串流，而不要求增加限制。

若要清除您使用 AWS CLI 或 SDK 建立的未使用串流，請執行下列動作：

```
aws iot delete-stream --stream-id value
```

如需詳細資訊，請參閱 [delete-stream](#)。

#### Note

您可使用 `list-streams` 命令來找出串流 ID。

## 對裝置機群中斷連線進行疑難排解

#### 協助我們改善此主題

[讓我們知道如何能使其變得更好](#)

AWS IoT 裝置叢集中斷可能發生多種原因。本文將說明如何診斷中斷連線的原因，以及如何處理因定期維護 AWS IoT 服務或節流限制而造成的中斷連線。

### 如要診斷中斷連線的原因

您可以在中檢查 [AWSIoTLogsV2](#) 個記錄群組，[CloudWatch](#) 以識別記錄項目 `disconnectReason` 欄位中斷連線的原因。

您也可以使用 AWS IoT 的 [生命週期事件](#) 功能來識別中斷連線的原因。如果您已訂閱 [生命週期的中斷連線事件](#) (`$aws/events/presence/disconnected/clientId`)，則會在中斷連線發生 AWS IoT 時收到通知。您可在通知的 `disconnectReason` 欄位中識別中斷連線的原因。

如需詳細資訊，請參閱 [CloudWatch AWS IoT 記錄項目](#) 和 [生命週期事件](#)。

### 疑難排解因 AWS IoT 服務維護而中斷連線的問題


由服務維護造成 AWS IoT 的中斷連線會記錄為 `SERVER_INITIATED_DISCONNECT` in AWS IoT 的生命週期事件和 CloudWatch。若要處理這些中斷連線，請調整用戶端設定，以確保您的裝置可以自動重新連線至平台。

## 如要對因調節限制而造成的中斷連線進行疑難排解

由節流限制造成的中斷連線會記錄為 THROTTLED in AWS IoT 的生命週期事件和。CloudWatch 如要處理這些中斷連線，您可依裝置計數增加，而請求 [訊息代理程式限制增加](#)。

如需詳細資訊，請參閱 [AWS IoT 核心訊息代理程式](#)。

## AWS IoT 裝置顧問疑難排解指

 協助我們改善此主題

[讓我們知道如何能使其變得更好](#)

### 一般

問：我可以平行執行多個測試套件嗎？

答案：是。Device Advisor 現在支援使用裝置層級端點，在不同裝置上執行多個測試套件。如果您使用帳戶級別的端點，則可以一次執行一個套件，因為每個帳戶只會有一個 Device Advisor 端點。如需詳細資訊，請參閱 [如何設定裝置](#)。

問：我從我的裝置看到 TLS 連線遭到 Device Advisor 的拒絕。這是預期的行為嗎？

答：是。Device Advisor 會在每次測試執行的前後拒絕 TLS 連線。我們建議使用者實作裝置重試機制，以獲取 Device Advisor 的全自動化測試體驗。如果您執行具有多個測試案例的測試套件 (例如 TLS 連線、MQTT 連線和 MQTT 發佈)，則建議您為裝置建置一個機制。該機制可以嘗試每 5 秒連線到我們的測試端點，持續一到兩分鐘。然後，您可以採取自動方式依序執行多個測試用例。

問：我能否取得從我的帳戶呼叫 Device Advisor API 的歷史記錄，以進行安全分析和作業的疑難排解？

答：可以。若要接收對您帳戶進行的 Device Advisor API 呼叫歷史記錄，只要 CloudTrail 在 AWS IoT 管理主控台中開啟，然後篩選事件來源即可 `iotdeviceadvisor.amazonaws.com`。

問：如何檢視「裝置建議程式」登入 CloudWatch？

答：如果將所需策略 ( 例 CloudWatch 如 ) 添加到服務角色 ( 請 [設定](#) 參閱 CloudWatchFullAccess )，則在測試套件運行期間生成的日誌會上傳到。如果測試套件中至少有一個測試用例，則會使用兩個日誌流創建日誌組「aws/iot/ 設備顧問/testSuiteId\$」。一個流是「\$testRunId」，包括在測試套件中執行測試用例之前和之後採取的操作日誌，例如設置和清理步

驟。另一個日誌流是「\$ suiteRunId\_\$」testRunId，這是特定於測試套件運行。從裝置傳送的事件，並 AWS IoT Core 將記錄至此記錄資料流。

問：裝置許可角色的用途為何？

答：設備顧問站在您的測試設備之間，並 AWS IoT Core 模擬測試場景。其會接受來自您測試裝置的連接和訊息，透過承擔您裝置許可角色並代表您啟動連線，將其轉送至 AWS IoT Core。確保設備角色權限與用於運行測試的證書上的權限相同非常重要。AWS IoT 當「裝置建議程式」使用裝置權限角色代表您起始連線時，不會強制執行憑證原則。AWS IoT Core 不過，會強制執行您設定之裝置許可角色的許可。

問：Device Advisor 支援哪些區域？

答：在 us-east-1、us-west-2、ap-northeast-1 和 eu-west-1 區域中支援 Device Advisor。

問：為什麼我會看到不一致的結果？

答：結果不一致的主要原因之一是將測試的 EXECUTION\_TIMEOUT 值設定過低。如需建議和預設 EXECUTION\_TIMEOUT 值的詳細資訊，請參閱 [Device Advisor 測試案例](#)。

問：Device Advisor 支援的 MQTT 通訊協定為何？

答：Device Advisor 支援具有 X509 用戶端憑證的 MQTT 版本 3.1.1。

問：如果嘗試將裝置連結到測試端點後，我的測試案例依舊失敗，並且顯示執行逾時的訊息，那麼該怎麼辦？

答：驗證 [建立要用作裝置角色的 IAM 角色](#) 之下的所有步驟。如果測試仍然失敗，則可能是因為裝置未傳送正確的伺服器名稱指示 (SNI) 擴充功能，Device Advisor 必須有這項功能才能運作。正確的 SNI 值是遵循「[設定您的裝置](#)」[區段後傳回的端點位址](#)。AWS IoT 也需要裝置將伺服器名稱指示 (SNI) 延伸功能傳送至傳輸層安全性 (TLS) 通訊協定。如需詳細資訊，請參閱 AWS IoT。

問：我的 MQTT 連線失敗，並顯示「libaws-c-mqtt: AWS\_ERROR\_MQTT\_UNT\_HANGUP」錯誤 (或) 我的裝置的 MQTT 連線正在自動從裝置建議程式端點中斷連線。如何解決此錯誤？


答：導致這種特定錯誤代碼和非預期中斷連線狀況的原因有很多，但最有可能與連接至裝置的 [裝置角色](#) 有關。以下檢查點 (依優先順序排列) 將解決此問題。

- 連接至裝置的裝置角色必須具有執行測試所需的最低 IAM 許可。裝置建議程式將使用連接的裝置角色代表測試裝置執行 AWS IoT MQTT 動作。若缺少所需的許可，當裝置嘗試連線至 Device Advisor 端點時，則會顯示 AWS\_ERROR\_MQTT\_UNEXPECTED\_HANGUP 錯誤或發生非預期的中斷連線狀況。例如，如果您選取執行 MQTT Publish 測試案例，則「Connect」和「發佈」動作都必須包含在具有對應 ClientId 和「主題」的角色中 (您可以使用逗號分隔值來提供多個值，也可

以使用萬用字元 (\*) 字元來提供前置字元值。例如：若要提供開頭為 TestTopic 的任一主題發佈許可，您可提供 "TestTopic\*" 作為資源值。以下是一些[政策範例](#)。

- 在裝置角色中為資源類型定義的值，與程式碼中使用的實際值不相符。例如：角色中 ClientId 定義的不匹配和設備代碼中實際 ClientId 使用的不匹配。設備角色和代碼中的值（例 ClientId 如，主題和 ) TopicFilter 必須相同。
- 裝置所連接的裝置憑證必須處於作用中，且使用[資源](#)所需的[動作許可](#)，將[政策](#)連接至這些裝置。請注意，裝置憑證原則會授與或拒絕 AWS IoT 資源和資 AWS IoT Core 料平面作業的存取權。Device Advisor 會要求您將作用中的裝置憑證連接至您的裝置，以便授予測試案例期間所使用的動作許可。

## AWS IoT Device Management 疑難排解指

 協助我們改善此主題

[讓我們知道如何能使其變得更好](#)

這是的疑難排解一節 AWS IoT Device Management。

主題

- [AWS IoT 工作疑難解](#)
- [機群索引疑難排解指南](#)

### AWS IoT 工作疑難解

這是「工作」的疑難排解 AWS IoT 區段。

如何找到 AWS IoT 工作端點？

如何找到 AWS IoT Jobs 控制平面端點？

AWS IoT 工作支援使用 HTTPS 通訊協定的控制平面 API 作業。確認您已使用 HTTPS 通訊協定連線到正確的控制平面端點。

如需 AWS 區域特定端點的清單，請參閱[AWS IoT 核心-控制平面端點](#)。

如需 FIPS 相容 AWS IoT 工作控制平面端點的清單，請參閱 [FIPS 端點](#) (依服務分類)

**Note**

AWS IoT 工作和 AWS IoT Core 共用相同的 AWS 區域特定端點。

## 如何找到 AWS IoT Jobs 資料平面端點？

AWS IoT 工作支援使用 HTTPS 和 MQTT 通訊協定的資料平面 API 作業。確認您已使用 HTTPS 或 MQTT 通訊協定連線到正確的資料平面端點。

- 通訊協定
  - 使用下面顯示的 [describe-endpoint](#) CLI 命令或 [DescribeEndpoint](#) 其餘 API。對於端點類型，請使用 `iot:Jobs`。

```
aws iot describe-endpoint --endpoint-type iot:Jobs
```

- MQTT 通訊協定
  - 使用下面顯示的 [describe-endpoint](#) CLI 命令或 [DescribeEndpoint](#) 其餘 API。對於端點類型，請使用 `iot>Data-ATS` (建議) 或 `iot>Data`。

```
aws iot describe-endpoint --endpoint-type iot>Data-ATS (recommended)
```

```
aws iot describe-endpoint --endpoint-type iot>Data
```

如需 FIPS 相容 AWS IoT 工作資料平面端點的清單，請參閱 [FIPS 端點](#) (依服務分類)

## 如何監視 AWS IoT 工作活動並提供指標？

使用 Amazon 監控任 AWS IoT 務活動可 CloudWatch 讓您即時掌握進行中的任務 AWS IoT 作業，並透過 AWS IoT 規則透過 CloudWatch 警示協助控制成本。您必須先設定記錄，才能監視 AWS IoT 工作活動和設定 CloudWatch 警示。如需設定記錄的詳細資訊，請參閱 [設定 AWS IoT 記錄](#)。

如需 Amazon 的詳細資訊，以 CloudWatch 及如何透過 IAM 使用者角色設定許可以使用 CloudWatch 資源，請參閱 [Amazon 的身分和存取管理 CloudWatch](#)。

## 如何使用 Amazon 設置任 AWS IoT 務指標和監控 CloudWatch ？

若要設定 AWS IoT 記錄，請遵循設定 [AWS IoT 記錄](#) 中所述的步驟。AWS IoT 您可以在 AWS Management Console、AWS CLI 或 API 中完成記錄設定。AWS IoT 針對特定物件群組設定的記錄必須僅在 AWS CLI 或 API 中完成。

[AWS IoT 工作測量結果](#) 段落包含用於監督「AWS IoT 工作」活動的「AWS IoT 工作」測量結果。它說明如何檢視和中的量 AWS Management Console 度 AWS CLI。

此外，您可以設定 CloudWatch 警示，提醒您要密切監控的特定指標。如需有關警示設定的指引，請參閱 [使用 Amazon CloudWatch 警示](#)。

## 裝置叢集與單一裝置疑難排解

### 工作執行會維持 QUEUED 無限期的狀態

當狀態為的工作執行 QUEUED 未進入下一個邏輯狀態 (例如 IN\_PROGRESS、或 TIMED\_OUT) 時 FAILED，下列其中一個案例可能是原因：

- 在 [CloudWatch 主控台](#) 的 CloudWatch 記錄中查看您的裝置活動。如需詳細資訊，請參閱 [AWS IoT 使用 CloudWatch 記錄監視](#)。
- 與該工作相關聯的 IAM 角色以及後續任務執行可能沒有列在附加到該 IAM 角色的 IAM 政策的其中一個政策聲明中的正確許可。使用 [describe-job](#) API 識別與該工作連結的 IAM 角色以及後續工作執行，並檢閱 IAM 政策以取得正確的許可。更新原則權限陳述式之後，您應該可以對資源執行 [AssumeRole](#) API 命令。

### 未為我的物件或物件群組建立工作執行

工作將狀態更新為 IN\_PROGRESS，工作文件會開始向目標群組中的所有裝置推出工作文件。此狀態更新將為每個目標裝置建立作業執行。如果未針對其中一個目標裝置建立工作執行，請參閱下列指南：

- 工作是否 thing 直接鎖定目標，工作的狀態更新為 IN\_PROGRESS，且工作為並行？如果滿足所有三個條件，則工作仍會傳送工作執行到目標群組中的所有裝置，而該裝置尚 thing 未收到其工作執行。
  - 在 AWS 管理主控台中檢閱目標群組中的裝置，以瞭解工作和工作狀態更新，或使用 [describe-job](#) API 命令。
  - 使用 [describe-job](#) API 命令檢閱工作的 IsConcurrent 屬性是否設定為 true 或 false。如需詳細資訊，請參閱 [Job 限制](#)。

- 不 *thing* 是工作的直接目標。
- 如果 Thing 已新增至工作，ThingGroup 並將工作鎖定為目標 ThingGroup，請確認 Thing 是的一部分 ThingGroup。
- 如果工作是狀態為 IN\_PROGRESS 且且為並行的快照工作，則該工作仍會將工作執行傳送至目標群組中的所有裝置，而該裝置尚 Thing 未收到其工作執行。
- 如果工作是狀態為 IN\_PROGRESS 且為並行的連續工作，則該工作仍會將工作執行傳送至目標群組中的所有裝置，而該裝置尚 Thing 未收到其工作執行。僅針對連續工作，您也可以 Thing 從中移除，ThingGroup 然後再將 Thing 其新增至 ThingGroup。
- 如果工作是狀態為且不是 IN\_PROGRESS 並行的快照工作，則可能是 AWS IoT 工作未確認 Thing 或 ThingGroup 成員資格關係。建議您在 AddThingToThingGroup 通話後增加幾秒鐘的等待時間，然後再創建您的 Job。或者，您可以將目標選擇切換至 Continuous，以便讓服務回填延遲 Thing 和 ThingGroup 成員資格附件事件。

## 新工作因 `LimitedExceededException` 錯誤而失敗

如果您的工作建立失敗且錯誤回應為 `LimitedExceededException`，請呼叫 `list-jobs` API 並檢閱所有工作，`isConcurrent=true` 以判斷您是否達到工作並行限制。如需並行 [Job 的其他資訊](#)，請參閱 [工作限制](#)。若要檢視工作並行限制並要求提高限制，請參閱 [AWS IoT Device Management 作業限制和配額](#)。

## Job 文件大小限制

工作文件大小受 MQTT 承載大小的限制。如果您需要的任務文件大於 32 kB (KB)、32,000 B (位元組)，請在 Amazon S3 中建立並存放任務文件，並在 `CreateJob` API 的 `documentSource` 欄位中新增 Amazon S3 物件 URL 或使用 AWS CLI 對於 AWS Management Console，在建立任務時，在 Amazon S3 URL 文字方塊中新增 Amazon S3 物件 URL。

- AWS Management Console 建立工作文件：[建立和管理工作 AWS Management Console](#)
- AWS CLI 建立工作文件：[建立和管理工作 AWS CLI](#)
- `CreateJob` API 文件：[CreateJob](#)

## 裝置端 MQTT 訊息要求節流限制

如果您收到錯誤碼 `400ThrottlingException`，則設備端 MQTT 消息因達到同時設備端請求的限制而失敗。請參閱 [AWS IoT Device Management 工作限制和配額](#)，以取得有關節流限制及其是否可調整的詳細資訊。



## 連線逾時錯誤

錯誤碼 400 RequestExpired 表示連線失敗，因為高延遲或低用戶端逾時值。

- [有關測試客戶端和服務器端之間連接的信息](#)，請參閱測試與設備數據端點的連接。

## 無效的 API 指令

確認輸入正確的 API 命令，以避免出現指出 API 命令無效的錯誤訊息。有關所有 [AWS IoT API 命令的完整列表](#)，請參閱 AWS IoT API 參考。

## 服務端連接錯誤

錯誤代碼 503 ServiceUnavailable 表示錯誤源自服務器端。

- 如需[所有 AWS 服務的目前狀態](#)，請參閱[AWS Health Dashboard \(所有 AWS 服務\)](#)。
- 有關您[個人的當前狀態](#)，請參閱[AWS Health Dashboard \(個人 AWS 帳戶\)](#) AWS 帳戶。

## 機群索引疑難排解指南

### 機群索引服務的疑難排解彙總查詢

如果您遇到類型不匹配錯誤，可以使用 CloudWatch Logs 來疑難排解問題。CloudWatch 在「叢集索引」服務寫入記錄檔之前，必須先啟用記錄檔。如需詳細資訊，請參閱 [AWS IoT 使用 CloudWatch 記錄監控](#)。

若要對非受管欄位進行彙總查詢，您必須指定您在 customFields 引數中定義的欄位傳遞給 UpdateIndexingConfiguration 或 update-indexing-configuration。如果欄位值與設定的欄位資料類型不一致，則在您執行彙總查詢時會忽略此值。

如果欄位因為類型不相符而無法編製索引，叢集索引服務會將錯誤記錄傳送至 CloudWatch 記錄檔。錯誤日誌包含欄位名稱、無法轉換的值，以及裝置的物件名稱。以下是範例錯誤日誌：

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

如果裝置已中斷連線大約一小時，則可能會缺少連線狀態的 timestamp 值。對於持續工作階段，當用戶端中斷連線的時間長於持續性工作階段所設定的 time-to-live (TTL) 時，該值可能會遺失。只有針對用戶端 ID 有相符物件名稱的連線，連線狀態資料才會建立索引。用戶端 ID 是用來連接裝置的值 AWS IoT Core。)

## 疑難排解機群索引組態

### 無法降級機群索引組態

移除與機群指標或動態群組相關聯的資料來源時，不支援機群索引組態降級。

例如，若您的索引組態具有登錄資料、影子資料和連線資料，且查詢 `thingName:TempSensor* AND shadow.desired.temperature>80` 有機群指標，則更新索引組態以僅包含登錄資料將導致錯誤。

不支援修改現有機群指標使用的自訂欄位。

由於不相容的機群指標或動態群組，無法更新您的索引組態

若由於不相容的機群指標或動態群組而無法更新索引組態，請先刪除不相容的機群指標或動態群組，然後再更新索引組態。

## 疑難排解位置索引和地理查詢

若要疑難排解位置索引和地理查詢中的不相符類型錯誤，您可以啟用 CloudWatch 記錄。如需有關如何監視 AWS IoT 使用的[詳細資訊 CloudWatch](#)，請遵循 [step-by-step 指南](#)。

使用地理查詢為UpdateIndexingConfiguration位置資料建立索引時，您在中指定的位置欄位geoLocations必須與您傳遞的位置欄位相符。如果發生不相符，叢集索引會傳送不相符的類型錯誤給。CloudWatch錯誤日誌包含欄位名稱、無法轉換的值，以及裝置的物件名稱。

以下是範例錯誤日誌：

```
{
  "timestamp": "2023-11-09 01:39:43.466",
  "logLevel": "ERROR",
  "traceId": "79738924-1025-3a00-a669-7bec69f7f07a",
  "accountId": "123456789012",
  "status": "Failure",
  "eventType": "IndexingGeoLocationFieldFailed",
  "thingName": "thing0",
  "failedGeolocationFields": [
    {
      "Name": "attributeName1",
      "Value": "apple",
      "ExpectedType": "Geopoint"
    }
  ],
  "reason": "failed to index the field because it could not be converted to one of the expected geoLocation formats."
}
```

如需詳細資訊，請參閱 [???](#)。

## 機群指標疑難排解

在中看不到資料點 CloudWatch

如果您可以建立叢集量度，但在中看不到資料點 CloudWatch，很可能是您沒有符合查詢字串條件的項目。

請參閱下列範例指令，了解如何建立機群指標：

```
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
```

如果沒有符合查詢字串條件 --query-string "thingName:TempSensor\* AND attributes.temperature>80" 的物件：

- 使用values=count，您將能夠創建一個艦隊指標，並且將有數據點顯示在中 CloudWatch。count 數值的資料點一律為 0。
- 除此之外values外count，您將能夠建立叢集指標，但不會在中看到叢集指標 CloudWatch，也不會顯示任何資料點 CloudWatch。

## AWS IoT 錯誤

 協助我們改善此主題

[讓我們知道如何能使其變得更好](#)

本節列出由傳送的錯誤碼 AWS IoT。

### 訊息代理程式錯誤代碼

錯誤代碼	錯誤說明
400	錯誤的請求。
401	未經授權。
403	禁止。
503	服務無法使用。

### 身分與安全性錯誤代碼

錯誤代碼	錯誤說明
401	未經授權。

### Device Shadow 錯誤代碼

錯誤代碼	錯誤說明
400	錯誤的請求。

錯誤代碼	錯誤說明
401	未經授權。
403	禁止。
404	未找到。
409	衝突。
413	請求過大。
422	無法處理請求。
429	請求過多。
500	內部錯誤。
503	服務無法使用。

# AWS IoT 裝置 SDK、行動 SDK 和 AWS IoT 裝置用戶端

本頁總結列出 AWS IoT Device SDK、開放原始碼程式庫、開發人員指南、範例應用程式和移植指南，協助您使用 AWS IoT 及選擇的硬體平台建置創新的 IoT 解決方案。

這些 SDK 適用於您的 IoT 裝置。如果您是開發 IoT 應用程式以在行動裝置上使用，請參閱 [AWS 行動開發套件](#)。如果您是開發 IoT 應用程式或伺服器端程式，請參閱 [AWS 開發套件](#)。

## AWS IoT 裝置開發套件

AWS IoT Device SDK 包含開放原始碼程式庫、含範例的開發人員指南，以及移植指南，讓您可以在自選的硬體平台上建置創新的 IoT 產品或解決方案。

### Note

AWS IoT 裝置開發套件已發行 MQTT 5 用戶端。AWS IoT 裝置開發套件不支援在 macOS 上使用 TLS 1.3。

這些 SDK 可協助您使用 MQTT 和 WSS 通訊協定將 IoT 裝置連接至 AWS IoT。

### C++

#### AWS IoT C++ 設備開發套件

AWS IoT C++ 設備 SDK 允許開發人員使用 AWS 和 AWS IoT API 構建連接的應用程式。此 SDK 特別是為並未受限於資源的裝置所設計，需要訊佇列、多重執行緒支援、最新語言功能等進階功能。如需詳細資訊，請參閱下列內容：

- [AWS IoT 開啟裝置開發套件 C++ V2 GitHub](#)
- [AWS IoT 設備 SDK C++ V2 自述文件](#)
- [AWS IoT 裝置開發套件 C++ V2 範例](#)
- [AWS IoT 設備開發套件 C++ v2 API 文檔](#)

### Python

#### AWS IoT Python 的設備開發套件

適用於 Python 的 AWS IoT 設備 SDK 使開發人員可以編寫 Python 腳本以使用他們的設備通過通信協議通過 MQTT 或 MQTT 訪問 AWS IoT 平台。WebSocket 透過將其裝置連接到 AWS IoT，使用者可以安全地使用 Kinesis 和 Amazon S3 等 AWS Lambda 其他 AWS 服務提供的訊息代理程式、規則和陰影。AWS IoT

- [AWS IoT 適用於 Python 的設備開發套件 GitHub](#)
- [AWS IoT 適用於 Python v2 自述文件的設備 SDK](#)
- [AWS IoT 適用於 Python 2 範例的裝置 SDK](#)
- [AWS IoT 適用於 Python 的裝置開發套件 API 文件](#)

## JavaScript

### AWS IoT 適用於的裝置 SDK JavaScript

aws-iot-device-sdk.js 套件可讓開發人員撰寫透過通訊協定 AWS IoT 使用 MQTT 或 MQTT 存取的 JavaScript 應用程式。WebSocket 可用於 Node.js 環境和瀏覽器應用程式中。如需詳細資訊，請參閱下列內容：

- [AWS IoT 適用於 JavaScript v2 的裝置 SDK GitHub](#)
- [AWS IoT JavaScript v2 自述文件的設備 SDK](#)
- [AWS IoT 適用於 JavaScript v2 範例的裝置 SDK](#)
- [AWS IoT 適用於 JavaScript v2 API 文件的裝置 SDK](#)

## Java

### AWS IoT Java 的裝置開發套件

Java 的 AWS IoT 裝置 SDK 可讓 Java 開發人員透過通訊協定透過 MQTT 或 MQTT 存取 AWS IoT 平台。WebSocket 此 SDK 內建有影子支援。您可以使用 HTTP 方法來存取影子，包括 GET、UPDATE 與 DELETE。此 SDK 亦支援簡化的影子存取模式，開發人員只需使用 getter 和 setter 方法，即可與影子交換資料，而無需將任何 JSON 文件序列化或還原序列化。

#### Note

適用於 Java v2 的 AWS IoT 裝置開發套件現在支援安卓系統開發。如需詳細資訊，請參閱[適用於 Android 的 AWS IoT 裝置 SDK](#)。

如需詳細資訊，請參閱下列內容：

- [AWS IoT 適用於 Java v2 的裝置開發套件 GitHub](#)
- [AWS IoT 適用於 Java v2 自述文件的設備 SDK](#)
- [AWS IoT 適用於 Java V2 範例的裝置 SDK](#)
- [AWS IoT 適用於 Java v2 的裝置開發套件 API 文件](#)

## AWS IoT 適用於嵌入式 C 的裝置 SDK

### Note

此 SDK 適合經驗豐富的嵌入式軟體開發人員使用。

適用於 Embedded C 的 AWS IoT Device SDK (C-SDK) 是 MIT 開放原始碼授權下的 C 原始碼檔案集合，可用於嵌入式應用程式，以安全地將 IoT 裝置連接到 AWS IoT Core。它包括 MQTT 用戶端、JSON 剖析器和 AWS IoT Device Shadow、AWS IoT 作業、AWS IoT 叢集佈建和 AWS IoT Device Defender 程式庫。此 SDK 以來源形式分配，並且可與應用程式碼、其他程式庫及您選擇的作業系統 (OS) 作業系統一起內建於客戶韌體中。

適用於 Embedded C 的 AWS IoT Device SDK 常針對需要優化 C 語言運行時的資源受限設備。您可以在任何作業系統上使用軟體開發套件，並將其裝載在任何處理器類型 (例如 MCU 和 MPU) 上。

如需詳細資訊，請參閱下列內容：

- [AWS IoT 適用於嵌入式 C 的裝置 SDK GitHub](#)
- [AWS IoT 嵌入式 C 自述文件的設備 SDK](#)
- [AWS IoT 適用於嵌入式 C 範例的裝置 SDK](#)

## 舊版 AWS IoT 裝置 SDK

這些是早期版本的 AWS IoT 裝置 SDK，已由上述較新版本所取代。這些 SDK 只接收維護和安全性更新。它們不會更新以包含新功能，而且不應該用於新專案。

- [AWS IoT C ++ 設備開發套件 GitHub](#)
- [AWS IoT C ++ 設備 SDK 自述文件](#)



- [AWS IoT 適用於 Python V1 的裝置開發套件 GitHub](#)
- [AWS IoT 用於 Python V1 自述文件的設備 SDK](#)
- [AWS IoT 適用於 Java 的裝置開發套件 GitHub](#)
- [AWS IoT Java 自述文件的設備 SDK](#)
- [AWS IoT 適用於開 JavaScript 啟的裝置 SDK GitHub](#)
- [AWS IoT JavaScript 自述文件的設備 SDK](#)
- [阿爾杜伊諾尤恩 SDK 上 GitHub](#)
- [Arduino Yún SDK 讀我檔案](#)

## AWS 行動開發套件

行 AWS 動開發套件為服務的 API、使用 MQTT 的 IoT 裝置通訊，以及其他 AWS IoT Core 服務的 API，提供行動應用程式開發人員平台特定的支援。AWS

### Android

#### AWS Mobile SDK for Android

包 AWS Mobile SDK for Android 含程式庫、範例和文件，供開發人員使用 AWS. 此 SDK 還包括對 MQTT 設備通信和調用服務的 API 的 AWS IoT Core 支持。如需詳細資訊，請參閱下列內容：

- [AWS Mobile SDK for Android 上 GitHub](#)
- [AWS Mobile SDK for Android 讀我檔案](#)
- [AWS Mobile SDK for Android 範例](#)
- [AWS Mobile SDK for Android API 參考](#)
- [AWSIoTClient 類參考文檔](#)

### iOS

#### AWS Mobile SDK for iOS

AWS Mobile SDK for iOS 這是一個開源軟體開發工具包，在 Apache 開源許可證下分發。提 AWS Mobile SDK for iOS 供程式庫、程式碼範例和文件，以協助開發人員使用 AWS. 此 SDK 也包括支援 MQTT 裝置通訊和呼叫 AWS IoT Core 服務的 API。如需詳細資訊，請參閱下列內容：

- [AWS Mobile SDK for iOS 上 GitHub](#)

- [AWS Mobile SDK for iOS 讀我檔案](#)
- [AWS Mobile SDK for iOS 範例](#)
- [AWS IoT 類別參考文件 AWS Mobile SDK for iOS](#)

## AWS IoT 裝置用戶端

AWS IoT Device Client 提供的程式碼可協助您的裝置連線 AWS IoT、執行叢集佈建工作、支援裝置安全性原則、使用安全通道進行連線，以及處理裝置上的工作。您可以在裝置上安裝此軟體，來處理這些例行裝置任務，以便可以專注於您的特定解決方案。

### Note

AWS IoT 裝置用戶端可搭配使用具有 x86\_64 或 ARM 處理器和一般 Linux 作業系統的微處理器 IoT 裝置。

## C++

### AWS IoT 裝置用戶端

如需 C++ 中 AWS IoT 裝置用戶端的詳細資訊，請參閱下列內容：

- [AWS IoT C++ 源代碼中的設備客戶端 GitHub](#)
- [AWS IoT C++ 自述文件中的設備客戶](#)

# AWS IoT 使用 AWS SDK 的程式碼範例

下列程式碼範例顯示如何搭 AWS IoT 配 AWS 軟體開發套件 (SDK) 使用。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境和跨服務範例中查看內容中的動作。

Scenarios (案例) 是向您展示如何呼叫相同服務中的多個函數來完成特定任務的程式碼範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含入門相關資訊和舊版 SDK 的詳細資訊。

開始使用

## 你好 AWS IoT

下列程式碼範例示範如何開始使用 AWS IoT。

C++

適用於 C++ 的 SDK

C MakeLists.txt 的 CMake 文件的代碼。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
  libraries for the AWS SDK.
```

```
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
    "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
    for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
    may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```

hello\_iot.cpp 來源檔案的程式碼。

```
#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 * A "Hello IoT" starter application which initializes an AWS IoT client and
 * lists the AWS IoT topics in the current account.
 *
 * main function
 *
 * Usage: 'hello_iot'
 *
 */
```

```
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optional: change the log level for debugging.
    // options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
iotClient.ListThings(
                listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                    << listThingsOutcome.GetError().GetMessage() <<
std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << allThings.size() << " thing(s) found." << std::endl;
        for (auto const &thing: allThings) {
            std::cout << thing.GetThingName() << std::endl;
        }
    }
}
```

```
}

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- 有關 API 詳細信息，請參閱 AWS SDK for C++ API 參考中的[列出事物](#)。

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.ListThingsRequest;
import software.amazon.awssdk.services.iot.model.ListThingsResponse;
import software.amazon.awssdk.services.iot.model.ThingAttribute;
import java.util.List;

public class HelloIoT {
    public static void main(String[] args) {
        System.out.println("Hello AWS IoT. Here is a listing of your AWS IoT
Things:");
        IotClient iotClient = IotClient.builder()
            .region(Region.US_EAST_1)
            .build();

        listAllThings(iotClient);
    }
}
```

```
    }

    public static void listAllThings( IotClient iotClient) {
        ListThingsRequest thingsRequest = ListThingsRequest.builder()
            .maxResults(10)
            .build();

        ListThingsResponse response = iotClient.listThings(thingsRequest) ;
        List<ThingAttribute> thingList = response.things();
        for (ThingAttribute attribute : thingList) {
            System.out.println("Thing name: "+attribute.thingName());
            System.out.println("Thing ARN: "+attribute.thingArn());
        }
    }
}
```

- 有關 API 詳細信息，請參閱 AWS SDK for Java 2.x API 參考中的[列出事物](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.ListThingsRequest

suspend fun main() {
    println("A listing of your AWS IoT Things:")
    listAllThings()
}

suspend fun listAllThings() {
    val thingsRequest =
        ListThingsRequest {
            maxResults = 10
        }
}
```

```
IotClient { region = "us-east-1" }.use { iotClient ->
    val response = iotClient.listThings(thingsRequest)
    val thingList = response.things
    if (thingList != null) {
        for (attribute in thingList) {
            println("Thing name ${attribute.thingName}")
            println("Thing ARN: ${attribute.thingArn}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱[在AWS SDK 中列出了 Kotlin API 參考](#)。

## 程式碼範例

- [AWS IoT 使用 AWS SDK 的動作](#)
  - [搭AttachThingPrincipal配 AWS 開發套件或 CLI 使用](#)
  - [搭CreateKeysAndCertificate配 AWS 開發套件或 CLI 使用](#)
  - [搭CreateThing配 AWS 開發套件或 CLI 使用](#)
  - [搭CreateTopicRule配 AWS 開發套件或 CLI 使用](#)
  - [搭DeleteCertificate配 AWS 開發套件或 CLI 使用](#)
  - [搭DeleteThing配 AWS 開發套件或 CLI 使用](#)
  - [搭DeleteTopicRule配 AWS 開發套件或 CLI 使用](#)
  - [搭DescribeEndpoint配 AWS 開發套件或 CLI 使用](#)
  - [搭DescribeThing配 AWS 開發套件或 CLI 使用](#)
  - [搭DetachThingPrincipal配 AWS 開發套件或 CLI 使用](#)
  - [搭ListCertificates配 AWS 開發套件或 CLI 使用](#)
  - [搭ListThings配 AWS 開發套件或 CLI 使用](#)
  - [搭SearchIndex配 AWS 開發套件或 CLI 使用](#)
  - [搭UpdateIndexingConfiguration配 AWS 開發套件或 CLI 使用](#)
  - [搭UpdateThing配 AWS 開發套件或 CLI 使用](#)
- [AWS IoT 使用 AWS SDK 的案例](#)
  - [使用 AWS IoT SDK 處理 AWS IoT 裝置、物件和陰影](#)



## AWS IoT 使用 AWS SDK 的動作

下列程式碼範例示範如何使用 AWS SDK 執 AWS IoT 行個別動作。這些摘錄會呼叫 AWS IoT API，是來自必須在內容中執行的大型程式碼摘錄。每個範例都包含一個連結 GitHub，您可以在其中找到設定和執程式碼的指示。

下列範例僅包含最常使用的動作。如需完整清單，請參閱 [《AWS IoT API 參考》](#)。

### 範例


- [搭AttachThingPrincipal配 AWS 開發套件或 CLI 使用](#)
- [搭CreateKeysAndCertificate配 AWS 開發套件或 CLI 使用](#)
- [搭CreateThing配 AWS 開發套件或 CLI 使用](#)
- [搭CreateTopicRule配 AWS 開發套件或 CLI 使用](#)
- [搭DeleteCertificate配 AWS 開發套件或 CLI 使用](#)
- [搭DeleteThing配 AWS 開發套件或 CLI 使用](#)
- [搭DeleteTopicRule配 AWS 開發套件或 CLI 使用](#)
- [搭DescribeEndpoint配 AWS 開發套件或 CLI 使用](#)
- [搭DescribeThing配 AWS 開發套件或 CLI 使用](#)
- [搭DetachThingPrincipal配 AWS 開發套件或 CLI 使用](#)
- [搭ListCertificates配 AWS 開發套件或 CLI 使用](#)
- [搭ListThings配 AWS 開發套件或 CLI 使用](#)
- [搭SearchIndex配 AWS 開發套件或 CLI 使用](#)
- [搭UpdateIndexingConfiguration配 AWS 開發套件或 CLI 使用](#)
- [搭UpdateThing配 AWS 開發套件或 CLI 使用](#)

### 搭AttachThingPrincipal配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AttachThingPrincipal。

## C++

## 適用於 C++ 的 SDK

 Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
#!/ Attach a principal to an AWS IoT thing.
/*!
 \param principal: A principal to attach.
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
client.AttachThingPrincipal(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的[AttachThing主體](#)。

## CLI

### AWS CLI

若要將憑證附加至您的物件

下列 `attach-thing-principal` 範例會將憑證附加至 `MyTemperatureSensor` 物件。憑證由 ARN 識別。您可以在 AWS IoT 主控台中找到憑證的 ARN。

```
aws iot attach-thing-principal \  
  --thing-name MyTemperatureSensor \  
  --principal arn:aws:iot:us-  
west-2:123456789012:cert/2e1eb273792174ec2b9bf4e9b37e6c6c692345499506002a35159767055278e8
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的 [如何使用登錄來管理物件](#)。

- 如需 API 詳細資訊，請參閱 AWS CLI 命令參考中的 [AttachThing 主體](#)。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void attachCertificateToThing(IotClient iotClient, String  
thingName, String certificateArn) {  
    // Attach the certificate to the thing.  
    AttachThingPrincipalRequest principalRequest =  
AttachThingPrincipalRequest.builder()  
        .thingName(thingName)  
        .principal(certificateArn)  
        .build();  
  
    AttachThingPrincipalResponse attachResponse =  
iotClient.attachThingPrincipal(principalRequest);
```

```
// Verify the attachment was successful.
if (attachResponse.sdkHttpResponse().isSuccessful()) {
    System.out.println("Certificate attached to Thing successfully.");

    // Print additional information about the Thing.
    describeThing(iotClient, thingName);
} else {
    System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
        attachResponse.sdkHttpResponse().statusCode());
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [AttachThing 主體](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的 [AttachThing 主體](#) 以獲取 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 CreateKeysAndCertificate 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 CreateKeysAndCertificate。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
#!/ Create keys and certificate for an Aws IoT device.
#!/ This routine will save certificates and keys to an output folder, if
provided.
/*!
 \param outputFolder: Location for storing output in files, ignored when string
is empty.
 \param certificateARNResult: A string to receive the ARN of the created
certificate.
 \param certificateID: A string to receive the ID of the created certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                          Aws::String &certificateARNResult,
                                          Aws::String &certificateID,
                                          const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
createKeysAndCertificateRequest;
```

```
Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
    client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
if (outcome.IsSuccess()) {
    std::cout << "Successfully created a certificate and keys" << std::endl;
    certificateARNResult = outcome.GetResult().GetCertificateArn();
    certificateID = outcome.GetResult().GetCertificateId();
    std::cout << "Certificate ARN: " << certificateARNResult << ",
certificate ID: "
        << certificateID << std::endl;

    if (!outputFolder.empty()) {
        std::cout << "Writing certificate and keys to the folder '" <<
outputFolder
            << "'." << std::endl;
        std::cout << "Be sure these files are stored securely." << std::endl;

        Aws::String certificateFilePath = outputFolder + "/"
certificate.pem.crt";
        std::ofstream certificateFile(certificateFilePath);
        if (!certificateFile.is_open()) {
            std::cerr << "Error opening certificate file, '" <<
certificateFilePath
                << "'."
                << std::endl;
            return false;
        }
        certificateFile << outcome.GetResult().GetCertificatePem();
        certificateFile.close();

        const Aws::IoT::Model::KeyPair &keyPair =
outcome.GetResult().GetKeyPair();

        Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
        std::ofstream privateKeyFile(privateKeyFilePath);
        if (!privateKeyFile.is_open()) {
            std::cerr << "Error opening private key file, '" <<
privateKeyFilePath
                << "'."
                << std::endl;
            return false;
        }
        privateKeyFile << keyPair.GetPrivateKey();
        privateKeyFile.close();
```

```
        Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
        std::ofstream publicKeyFile(publicKeyFilePath);
        if (!publicKeyFile.is_open()) {
            std::cerr << "Error opening public key file, '" <<
publicKeyFilePath
                    << "'."
                    << std::endl;
            return false;
        }
        publicKeyFile << keyPair.GetPublicKey();
    }
}
else {
    std::cerr << "Error creating keys and certificate: "
              << outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[CreateKeysAndCertificate](#)中的。

## CLI

### AWS CLI

若要建立 RSA key pair 並發行 X.509 憑證

以下內容 `create-keys-and-certificate` 會建立 2048 位元 RSA key pair，並使用已核發的公開金鑰發行 X.509 憑證。因為這是 AWS IoT 唯一提供此憑證的私密金鑰時，請務必將其保存在安全的位置。

```
aws iot create-keys-and-certificate \
  --certificate-pem-outfile "myTest.cert.pem" \
  --public-key-outfile "myTest.public.key" \
  --private-key-outfile "myTest.private.key"
```

輸出：

```
{
```

```

"certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2
"certificateId":
"9894ba17925e663f1d29c23af4582b8e3b7619c31f3fbd93adcb51ae54b83dc2",
"certificatePem": "
-----BEGIN CERTIFICATE-----
MIICiTCCEXAMPLE6m7oRw0uX0jANBgkqhkiG9w0BAQUFADCBiDELMakGA1UEBhMC
VVMxZCZAJBgNVBAGEXAMPLEAwDgYDVQHQEwdTZWF0dGxLMQ8wDQYDVQQKEwZBbWF6
b24xFDA5BgNVBA5TC01BTSEXAMPLE2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAd
BgkqhkiG9w0BCQEWEG5vb251QGFtYEXAMPLEb20wHhcNMTEwNDI1MjA0NTIxWhcN
MTIwNDI0MjA0NTIxWjCBiDELMakGA1UEBhMCEXAMPLEJBgNVBAGTAldBMRAwDgYD
VQHQEwdTZWF0dGxLMQ8wDQYDVQQKEwZBbWF6b24xFDAEXAMPLEsTC01BTSBDb25z
b2x1MRIwEAYDVQQDEw1UZXR0Q21sYWxhZAdBgkqhkiG9w0BCQEXAMPLE251QGFt
YXpvi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+aEXAMPLE
EXAMPLEfEvySwTc2XADZ4nB+BLyGVIk60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9T
rDHudUZEXAMPLELg5M43q7Wgc/MbQITx0USQv7c7ugFFDzQGBzZswY6786m86gpE
Ibb30hjZnzcVQEXAMPLEWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
nUhVVxYUntneD9+h8Mg9qEXAMPLEEyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
FFBjvSfpJI1J00zbhNYS5f6GuoEDEXAMPLEBHjJnyp3780D8uTs7fLvJx79LjSTb
NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
-----END CERTIFICATE-----\n",
"keyPair": {
  "PublicKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQ0CAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\nnMMEXAMPLEuuN/
dMAS3fyce8DW/4+EXAMPLEYjmoF/YVF/gHr99VEEXAMPLE5VF13\nn59VK7cEXAMPLE67GK+y
+jikqX0gHh/xJTtwo
+sGpWEXAMPLEDz18x0d2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkr7qkEXAMPLE1DR6sx2Hocli00Ltu6Fkw91swQ
\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQ0UXP6p1fgxwKRX2fEXAMPLEDa
\nhJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFR188eGdsAEXAMPLE1nI9EesG\nnFQIDAQAB
\n-----END PUBLIC KEY-----\n",
  "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\nkey omitted for security
reasons\n-----END RSA PRIVATE KEY-----\n"
}
}

```

如需詳細資訊，請參閱 IoT 開發人員指南中的建立和註冊 AWS AWS IoT [裝置憑證](#)。

- 如需 API 詳細資訊，請參閱 AWS CLI 命令參考 [CreateKeysAndCertificate](#) 中的。



## Java

### 適用於 Java 2.x 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[CreateKeysAndCertificate](#)中的。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK [CreateKeysAndCertificate](#) 中的 Kotlin API 參考。


如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 **CreateThing** 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 `CreateThing`。

## C++

## 適用於 C++ 的 SDK

 Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
#!/ Create an AWS IoT thing.
/*!
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
        createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[CreateThing](#)中的。

## CLI

## AWS CLI

範例 1：若要在登錄中建立物件記錄

下列create-thing範例會在 AWS IoT 物件登錄中為裝置建立項目。

```
aws iot create-thing \  
  --thing-name SampleIoTThing
```

輸出：

```
{  
  "thingName": "SampleIoTThing",  
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",  
  "thingId": " EXAMPLE1-90ab-cdef-fedc-ba987EXAMPLE "  
}
```

範例 2：若要定義與物件類型相關聯的物件

下列create-thing範例會建立具有指定物件類型及其屬性的物件。

```
aws iot create-thing \  
  --thing-name "MyLightBulb" \  
  --thing-type-name "LightBulb" \  
  --attribute-payload '{"attributes": {"wattage": "75", "model": "123"}}'
```

輸出：

```
{  
  "thingName": "MyLightBulb",  
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",  
  "thingId": "40da2e73-c6af-406e-b415-15acae538797"  
}
```

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的[如何使用登錄和物件類型管理](#)物件。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[CreateThing](#)中的。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
            iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN
            value is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[CreateThing](#)中的。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK [CreateThing](#) 中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 `CreateTopicRule` 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 `CreateTopicRule`。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
//! Create an AWS IoT rule with an SNS topic as the target.
/*!
    \param ruleName: The name for the rule.
    \param snsTopic: The SNS topic ARN for the action.
    \param sql: The SQL statement used to query the topic.
    \param roleARN: The IAM role ARN for the action.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
```

```
*/
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String
                             &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
                             &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的 [CreateTopic規則](#)。

## CLI

## AWS CLI

若要建立傳送 Amazon SNS 警示的規則

下列create-topic-rule範例會建立一個規則，當土壤濕度讀數 (如裝置陰影中所找到) 低時，傳送 Amazon SNS 訊息。

```
aws iot create-topic-rule \  
  --rule-name "LowMoistureRule" \  
  --topic-rule-payload file://plant-rule.json
```

此範例需要將下列 JSON 程式碼儲存到名為的檔案中plant-rule.json：

```
{  
  "sql": "SELECT * FROM '$aws/things/MyRPi/shadow/update/accepted' WHERE  
state.reported.moisture = 'low'\n",  
  "description": "Sends an alert whenever soil moisture level readings are too  
low.",  
  "ruleDisabled": false,  
  "awsIotSqlVersion": "2016-03-23",  
  "actions": [{  
    "sns": {  
      "targetArn": "arn:aws:sns:us-  
west-2:123456789012:MyRPiLowMoistureTopic",  
      "roleArn": "arn:aws:iam::123456789012:role/service-role/  
MyRPiLowMoistureTopicRole",  
      "messageFormat": "RAW"  
    }  
  }  
}]  
}
```

此命令不會產生輸出。


如需詳細資訊，請參閱 [AWS IoT 開發人員指南中的建立AWS IoT 規則](#)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的 [CreateTopic規則](#)。



## Java

## 適用於 Java 2.x 的 SDK

 Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
        CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");

    } catch (IotException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的[CreateTopic規則](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }

    val topicRuleRequest =
```

```

        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

        IotClient { region = "us-east-1" }.use { iotClient ->
            iotClient.createTopicRule(topicRuleRequest)
            println("IoT rule created successfully.")
        }
    }
}

```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的[CreateTopic規則](#)以獲取 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭DeleteCertificate配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteCertificate。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```

//! Delete a certificate.
/*!
 \param certificateID: The ID of a certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

```

```
Aws::IoT::Model::DeleteCertificateRequest request;
request.SetCertificateId(certificateID);

Aws::IoT::Model::DeleteCertificateOutcome outcome =
iotClient.DeleteCertificate(
    request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted certificate " << certificateID <<
std::endl;
}
else {
    std::cerr << "Error deleting certificate " << certificateID << ": " <<
outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[DeleteCertificate](#)中的。

## CLI

### AWS CLI

#### 刪除裝置憑證

下列delete-certificate範例會刪除具有指定 ID 的裝置憑證。

```
aws iot delete-certificate \  
    --certificate-id  
    c0c57bbc8baaf4631a9a0345c957657f5e710473e3ddbbee1428d216d54d53ac9
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IoT API 參考資料[DeleteCertificate](#)中的。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[DeleteCertificate](#)中的。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[DeleteCertificate](#)中的。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
    IotClient { region = "us-east-1" }.use { iotClient ->
```

```
        iotClient.deleteCertificate(certificateProviderRequest)
        println("$certificateArn was successfully deleted.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK [DeleteCertificate](#) 中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 DeleteThing 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 DeleteThing。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
//! Delete an AWS IoT thing.
/*!
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
}
```

```
else {
    std::cerr << "Error deleting thing " << thingName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[DeleteThing](#)中的。

## CLI

### AWS CLI

若要顯示物件的詳細資訊

下列delete-thing範例會從您 AWS 帳戶的 AWS IoT 登錄中刪除物件。

```
aws iot 刪除的東西-事物名稱 "「 FourthBulb
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的[如何使用登錄來管理物件](#)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[DeleteThing](#)中的。

## Java

適用於 Java 2.x 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();
```

```
        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考 [DeleteThing](#) 中的。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK [DeleteThing](#) 中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。



## 搭DeleteTopicRule配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteTopicRule。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
#!/ Delete an AWS IoT rule.
/*!
 \param ruleName: The name for the rule.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                  const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
            ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的[DeleteTopic規則](#)。

## CLI

### AWS CLI

#### 刪除規則

下列delete-topic-rule範例會刪除指定的規則。

```
aws iot delete-topic-rule \  
    --rule-name "LowMoistureRule"
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的[刪除規則](#)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的[DeleteTopic規則](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭DescribeEndpoint配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DescribeEndpoint。

### C++

#### 適用於 C++ 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
//! Describe the endpoint specific to the AWS account making the call.  
/*!  
    \param endpointResult: String to receive the endpoint result.  
    \param clientConfiguration: AWS client configuration.  
    \return bool: Function succeeded.  
*/
```

```
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
        "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
    iotClient.DescribeEndpoint(
        describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[DescribeEndpoint](#)中的。

## CLI

### AWS CLI

範例 1：取得目前的 AWS 端點

下列describe-endpoint範例會擷取套用所有命令的預設 AWS 端點。

```
aws iot describe-endpoint
```

輸出：

```
{
```

```
"endpointAddress": "abc123defghijk.iot.us-west-2.amazonaws.com"
}
```

如需詳細資訊，請參閱 AWS IoT 開發人員指南 [DescribeEndpoint](#) 中的。

## 範例 2：取得您的 ATS 端點

下列 describe-endpoint 範例會擷取 Amazon 信任服務 (ATS) 端點。

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

輸出：

```
{
  "endpointAddress": "abc123defghijk-ats.iot.us-west-2.amazonaws.com"
}
```

如需詳細資訊，請參閱 [AWS IoT 開發人員指南中的 X.509 憑證和 AWS IoT](#)。

- 如需 API 詳細資訊，請參閱 AWS CLI 命令參考 [DescribeEndpoint](#) 中的。

## Java

適用於 Java 2.x 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
            iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
    }
}
```

```
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考 [DescribeEndpoint](#) 中的。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK [DescribeEndpoint](#) 中的 Kotlin API 參考。

## Rust

### 適用於 Rust 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
async fn show_address(client: &Client, endpoint_type: &str) -> Result<(), Error>
{
    let resp = client
        .describe_endpoint()
        .endpoint_type(endpoint_type)
        .send()
        .await?;

    println!("Endpoint address: {}", resp.endpoint_address.unwrap());

    println!();

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK [DescribeEndpoint](#) 中的 Rust API 參考資料。


如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 DescribeThing 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 DescribeThing。

## C++

## 適用於 C++ 的 SDK

 Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
//! Describe an AWS IoT thing.
/*!
  \param thingName: The name for the thing.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
    iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing " << result.GetThingName() << " " <<
std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
<< std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << "  attribute: " << attribute.first << "=" <<
attribute.second
<< std::endl;
        }
    }
    else {
        std::cerr << "Error describing thing " << thingName << ": " <<
```

```
        outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[DescribeThing](#)中的。

## CLI

### AWS CLI

若要顯示物件的詳細資訊

下列describe-thing範例會顯示在 AWS IoT 登錄中為您的 AWS 帳戶定義之物件 (裝置) 的相關資訊。

aws iot 描述的東西-事物名稱「燈泡」MyLight

輸出：

```
{
  "defaultClientId": "MyLightBulb",
  "thingName": "MyLightBulb",
  "thingId": "40da2e73-c6af-406e-b415-15acae538797",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
  "thingTypeName": "LightBulb",
  "attributes": {
    "model": "123",
    "wattage": "75"
  },
  "version": 1
}
```

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的[如何使用登錄來管理物件](#)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[DescribeThing](#)中的。



## Java

### 適用於 Java 2.x 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build() ;

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[DescribeThing](#)中的。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
        DescribeThingRequest {
            thingName = thingNameVal
        }

    // Print Thing details.
    IotClient { region = "us-east-1" }.use { iotClient ->
        val describeResponse = iotClient.describeThing(thingRequest)
        println("Thing details:")
        println("Thing name: ${describeResponse.thingName}")
        println("Thing ARN:  ${describeResponse.thingArn}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK [DescribeThing](#) 中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 `DetachThingPrincipal` 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 `DetachThingPrincipal`。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 [GitHub](#)。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```
//! Detach a principal from an AWS IoT thing.
/*!
    \param principal: A principal to detach.
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
```

```
*/
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
    iotClient.DetachThingPrincipal(
        detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
thing "
                << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
                << thingName << ": "
                << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的[DetachThing主體](#)。

## CLI

### AWS CLI

從物件中分離認證/主體的步驟

下列detach-thing-principal範例會從指定的物件移除代表主參與者的憑證。

```
aws iot detach-thing-principal \  
  --thing-name "MyLightBulb" \  
  --principal-name "MyPrincipal" \  
  --thing-id "MyThingId" \  
  --region "us-east-1" \  
  --profile "MyProfile" \  
  --output "text" \  
  --verbose
```

```
--principal "arn:aws:iot:us-west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36"
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的[如何使用登錄來管理物件](#)。

- 如需 API 詳細資訊，請參閱 AWS CLI 命令參考中的[DetachThing 主體](#)。

## Java

適用於 Java 2.x 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void detachThingPrincipal(IotClient iotClient, String
thingName, String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的[DetachThing 主體](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun detachThingPrincipal(  
    thingNameVal: String,  
    certificateArn: String,  
) {  
    val thingPrincipalRequest =  
        DetachThingPrincipalRequest {  
            principal = certificateArn  
            thingName = thingNameVal  
        }  
  
    IotClient { region = "us-east-1" }.use { iotClient ->  
        iotClient.detachThingPrincipal(thingPrincipalRequest)  
        println("$certificateArn was successfully removed from $thingNameVal")  
    }  
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的[DetachThing主體](#)以獲取 Kotlin API 參考。


如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭ListCertificates配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListCertificates。

## C++

## 適用於 C++ 的 SDK

 Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
//! List certificates registered in the AWS account making the call.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::IoT::listCertificates(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
        iotClient.ListCertificates(
            request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
            outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                  result.GetCertificates().begin(),
                                  result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
            std::endl;
        }
    }
}
```

```
        return false;
    }
} while (!marker.empty());

std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

for (auto &certificate: allCertificates) {
    std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
std::endl;
    std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
        << std::endl;
    std::cout << std::endl;
}

return true;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[ListCertificates](#)中的。

## CLI

### AWS CLI

#### 範例 1：列出您 AWS 帳戶中註冊的憑證

下列 `list-certificates` 範例會列出您帳戶中註冊的所有憑證。如果您的預設分頁限制超過 25，您可以使用此命令的 `nextMarker` 回應值，並將其提供給下一個命令，以取得下一批結果。重複直到沒有值的 `nextMarker` 返回。

```
aws iot list-certificates
```

輸出：

```
{
  "certificates": [
    {
      "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
      "certificateId":
"604c48437a57b7d5fc5d137c5be75011c6ee67c9a6943683a1acb4b1626bac36",
      "status": "ACTIVE",
```

```
        "creationDate": 1556810537.617
    },
    {
        "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
        "certificateId":
"262a1ac8a7d8aa72f6e96e365480f7313aa9db74b8339ec65d34dc3074e1c31e",
        "status": "ACTIVE",
        "creationDate": 1546447050.885
    },
    {
        "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
        "certificateId":
"b193ab7162c0fadca83246d24fa090300a1236fe58137e121b011804d8ac1d6b",
        "status": "ACTIVE",
        "creationDate": 1546292258.322
    },
    {
        "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
        "certificateId":
"7aebeea3845d14a44ec80b06b8b78a89f3f8a706974b8b34d18f5adf0741db42",
        "status": "ACTIVE",
        "creationDate": 1541457693.453
    },
    {
        "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
        "certificateId":
"54458aa39ebb3eb39c91ffbbdcc3a6ca1c7c094d1644b889f735a6fc2cd9a7e3",
        "status": "ACTIVE",
        "creationDate": 1541113568.611
    },
    {
        "certificateArn": "arn:aws:iot:us-
west-2:123456789012:cert/4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
        "certificateId":
"4f0ba725787aa94d67d2fca420eca022242532e8b3c58e7465c7778b443fd65e",
        "status": "ACTIVE",
        "creationDate": 1541022751.983
    }
}
]
```



```
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[ListCertificates](#)中的。

## Java

適用於 Java 2.x 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[ListCertificates](#)中的。

## Kotlin

適用於 Kotlin 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
    }
}
```

```
    val certList = response.certificates
    certList?.forEach { cert ->
        println("Cert id: ${cert.certificateId}")
        println("Cert Arn: ${cert.certificateArn}")
    }
}
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK [ListCertificates](#) 中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 ListThings 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 ListThings。

### CLI

#### AWS CLI

示例 1：列出註冊表中的所有內容

下列 list-things 範例會列出您 AWS 帳戶在 AWS IoT 登錄中定義的物件 (裝置)。

```
aws iot list-things
```

輸出：

```
{
  "things": [
    {
      "thingName": "ThirdBulb",
      "thingTypeName": "LightBulb",
      "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/ThirdBulb",
      "attributes": {
        "model": "123",
        "wattage": "75"
      },
      "version": 2
    },
  ],
}
```

```
{
  "thingName": "MyOtherLightBulb",
  "thingTypeName": "LightBulb",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyOtherLightBulb",
  "attributes": {
    "model": "123",
    "wattage": "75"
  },
  "version": 3
},
{
  "thingName": "MyLightBulb",
  "thingTypeName": "LightBulb",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
  "attributes": {
    "model": "123",
    "wattage": "75"
  },
  "version": 1
},
{
  "thingName": "SampleIoTThing",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/SampleIoTThing",
  "attributes": {},
  "version": 1
}
]
```

## 範例 2：列出具有特定屬性的已定義物件

下列 `list-things` 範例會顯示具有名為屬性的物件清單 `wattage`。

```
aws iot list-things \
  --attribute-name wattage
```

輸出：

```
{
  "things": [
    {
      "thingName": "MyLightBulb",
```

```
    "thingTypeName": "LightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyLightBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "version": 1
  },
  {
    "thingName": "MyOtherLightBulb",
    "thingTypeName": "LightBulb",
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/
MyOtherLightBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "version": 3
  }
]
}
```

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的[如何使用登錄來管理物件](#)。

- 如需 API 詳細資訊，請參閱 AWS CLI 命令參考[ListThings](#)中的。

## Rust

### 適用於 Rust 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
async fn show_things(client: &Client) -> Result<(), Error> {
    let resp = client.list_things().send().await?;

    println!("Things:");

    for thing in resp.things.unwrap() {
```

```
println!(
    " Name: {}",
    thing.thing_name.as_deref().unwrap_or_default()
);
println!(
    " Type: {}",
    thing.thing_type_name.as_deref().unwrap_or_default()
);
println!(
    " ARN: {}",
    thing.thing_arn.as_deref().unwrap_or_default()
);
println!();
}

println!();

Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK [ListThings](#) 中的 Rust API 參考資料。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 SearchIndex 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 SearchIndex。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
//! Query the AWS IoT fleet index.
```

```
//! For query information, see https://docs.aws.amazon.com/iot/latest/developerguide/query-syntax.html
/*!
  \param query: The query string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                     result.GetThings().cbegin(),
                                     result.GetThings().cend());
            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "Error in SearchIndex: " <<
outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
std::endl;
```

```
for (const auto thingDocument: allThingDocuments) {
    std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
              << std::endl;
}
return true;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[SearchIndex](#)中的。

## CLI

### AWS CLI

若要查詢物件索引

下列search-index範例會查詢具有類型之物件的AWS\_Things索引LightBulb。

```
aws iot search-index \
  --index-name "AWS_Things" \
  --query-string "thingTypeName:LightBulb"
```

輸出：

```
{
  "things": [
    {
      "thingName": "MyLightBulb",
      "thingId": "40da2e73-c6af-406e-b415-15acae538797",
      "thingTypeName": "LightBulb",
      "thingGroupNames": [
        "LightBulbs",
        "DeadBulbs"
      ],
      "attributes": {
        "model": "123",
        "wattage": "75"
      },
      "connectivity": {
        "connected": false
      }
    },
    {
```

```
    "thingName": "ThirdBulb",
    "thingId": "615c8455-33d5-40e8-95fd-3ee8b24490af",
    "thingTypeName": "LightBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "connectivity": {
      "connected": false
    }
  },
  {
    "thingName": "MyOtherLightBulb",
    "thingId": "6dae0d3f-40c1-476a-80c4-1ed24ba6aa11",
    "thingTypeName": "LightBulb",
    "attributes": {
      "model": "123",
      "wattage": "75"
    },
    "connectivity": {
      "connected": false
    }
  }
]
```

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的[管理物件索引](#)。

- 如需 API 詳細資訊，請參閱 AWS CLI 命令參考[SearchIndex](#)中的。

## Java

適用於 Java 2.x 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
```



```
        .queryString(queryString)
        .build();

    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
    iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
    System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考[SearchIndex](#)中的。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 [GitHub](#)。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
    }
}
```

```

        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
            searchIndexResponse.things
                ?.forEach { thing -> println("Thing id found using search is
                ${thing.thingId}") }
        }
    }
}

```

- 有關 API 的詳細信息，請參閱 AWS SDK [SearchIndex](#) 中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭 UpdateIndexingConfiguration 配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用 UpdateIndexingConfiguration。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執行程式碼範例儲存庫](#)。

```

//! Update the indexing configuration.
/*!
 \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
 ignored if not set.
 \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration
 object which is ignored if not set.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateIndexingConfiguration(

```

```
    const Aws::IoT::Model::ThingIndexingConfiguration
&thingIndexingConfiguration,
    const Aws::IoT::Model::ThingGroupIndexingConfiguration
&thingGroupIndexingConfiguration,
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {
request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
iotClient.UpdateIndexingConfiguration(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有關 API 詳細信息，請參閱 AWS SDK for C++ API 參考中的 [UpdateIndexing配置](#)。

## CLI

### AWS CLI

若要啟用物件索引

下列update-indexing-configuration範例會啟用物件索引，以支援使用 AWS\_Things 索引來搜尋登錄資料、陰影資料及物件連線狀態。

```
aws iot update-indexing-configuration
  --thing-indexing-configuration
  thingIndexingMode=REGISTRY_AND_SHADOW,thingConnectivityIndexingMode=STATUS
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IoT 開發人員指南中的[管理物件索引](#)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的[UpdateIndexing組態](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 搭UpdateThing配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateThing。

C++

適用於 C++ 的 SDK

### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
//! Update an AWS IoT thing with attributes.
/*!
 \param thingName: The name for the thing.
 \param attributeMap: A map of key/value attributes/
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                             const std::map<Aws::String, Aws::String>
                             &attributeMap,
```

```
const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[UpdateThing](#)中的。

## CLI

### AWS CLI

將物件與物件類型產生關聯

下列update-thing範例會將 AWS IoT 登錄中的物件與物件類型相關聯。建立關聯時，您會為物件類型定義的屬性提供值。

```
aws iot update-thing \  
  --thing-name "MyOtherLightBulb" \  
  --thing-type-name "LightBulb" \  
  --attribute-payload '{"attributes": {"wattage": "75", "model": "123"}}'
```

此命令不會產生輸出。使用describe-thing指令查看結果。

如需詳細資訊，請參閱 [AWS IoT 開發人員指南中的物類型](#)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考[UpdateThing](#)中的。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考 [UpdateThing](#) 中的。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在 [AWS 設定和執程式碼範例儲存庫](#)。

```
suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK [UpdateThing](#) 中的 Kotlin API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## AWS IoT 使用 AWS SDK 的案例

下列程式碼範例說明如何在 AWS SDK 中 AWS IoT 實作常見案例。這些案例會示範如何透過在其中呼叫多個函式來完成特定工作 AWS IoT。每個案例都包含一個連結 GitHub，您可以在其中找到如何設定和執行程式碼的指示。

### 範例

- [使用 AWS IoT SDK 處理 AWS IoT 裝置、物件和陰影](#)

## 使用 AWS IoT SDK 處理 AWS IoT 裝置、物件和陰影

下列程式碼範例說明如何使用 AWS IoT SDK 處理 AWS IoT 裝置管理使用案例

### C++

適用於 C++ 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

創建一個 AWS IoT 東西。

```
Aws::String thingName = askQuestion("Enter a thing name: ");

if (!createThing(thingName, clientConfiguration)) {
    std::cerr << "Exiting because createThing failed." << std::endl;
    cleanup("", "", "", "", "", false, clientConfiguration);
    return false;
}
```

```
//! Create an AWS IoT thing.
/*!
```



```

\param thingName: The name for the thing.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
        createThingRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

產生並附加裝置憑證。

```

    Aws::String certificateARN;
    Aws::String certificateID;
    if (askYesNoQuestion("Would you like to create a certificate for your thing?
(y/n) ")) {
        Aws::String outputFolder;
        if (askYesNoQuestion(
            "Would you like to save the certificate and keys to file? (y/n)
")) {
            outputFolder = std::filesystem::current_path();
            outputFolder += "/device_keys_and_certificates";

            std::filesystem::create_directories(outputFolder);

            std::cout << "The certificate and keys will be saved to the folder: "
                << outputFolder << std::endl;
        }
    }

```

```

        if (!createKeysAndCertificate(outputFolder, certificateARN,
certificateID,
                                clientConfiguration)) {
            std::cerr << "Exiting because createKeysAndCertificate failed."
                << std::endl;
            cleanup(thingName, "", "", "", "", false, clientConfiguration);
            return false;
        }

        std::cout << "\nNext, the certificate will be attached to the thing.\n"
            << std::endl;
        if (!attachThingPrincipal(certificateARN, thingName,
clientConfiguration)) {
            std::cerr << "Exiting because attachThingPrincipal failed." <<
std::endl;
            cleanup(thingName, certificateARN, certificateID, "", "",
                false,
                clientConfiguration);
            return false;
        }
    }
}

```

```

/*! Create keys and certificate for an Aws IoT device.
    /*! This routine will save certificates and keys to an output folder, if
        provided.
    /*!
        \param outputFolder: Location for storing output in files, ignored when string
            is empty.
        \param certificateARNResult: A string to receive the ARN of the created
            certificate.
        \param certificateID: A string to receive the ID of the created certificate.
        \param clientConfiguration: AWS client configuration.
        \return bool: Function succeeded.
    */
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);

```

```
Aws::IoT::Model::CreateKeysAndCertificateRequest
createKeysAndCertificateRequest;

Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
    client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
if (outcome.IsSuccess()) {
    std::cout << "Successfully created a certificate and keys" << std::endl;
    certificateARNResult = outcome.GetResult().GetCertificateArn();
    certificateID = outcome.GetResult().GetCertificateId();
    std::cout << "Certificate ARN: " << certificateARNResult << ",
certificate ID: "
        << certificateID << std::endl;

    if (!outputFolder.empty()) {
        std::cout << "Writing certificate and keys to the folder '" <<
outputFolder
            << "'." << std::endl;
        std::cout << "Be sure these files are stored securely." << std::endl;

        Aws::String certificateFilePath = outputFolder + "/"
certificate.pem.crt";
        std::ofstream certificateFile(certificateFilePath);
        if (!certificateFile.is_open()) {
            std::cerr << "Error opening certificate file, '" <<
certificateFilePath
                << "'."
                << std::endl;
            return false;
        }
        certificateFile << outcome.GetResult().GetCertificatePem();
        certificateFile.close();

        const Aws::IoT::Model::KeyPair &keyPair =
outcome.GetResult().GetKeyPair();

        Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
        std::ofstream privateKeyFile(privateKeyFilePath);
        if (!privateKeyFile.is_open()) {
            std::cerr << "Error opening private key file, '" <<
privateKeyFilePath
                << "'."
                << std::endl;
            return false;
        }
    }
}
```

```

        privateKeyFile << keyPair.GetPrivateKey();
        privateKeyFile.close();

        Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
        std::ofstream publicKeyFile(publicKeyFilePath);
        if (!publicKeyFile.is_open()) {
            std::cerr << "Error opening public key file, '" <<
publicKeyFilePath
                << "'."
                << std::endl;
            return false;
        }
        publicKeyFile << keyPair.GetPublicKey();
    }
}
else {
    std::cerr << "Error creating keys and certificate: "
        << outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
    \param principal: A principal to attach.
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
client.AttachThingPrincipal(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
}

```

```
else {
    std::cerr << "Failed to attach principal to thing." <<
        outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}
```

對 AWS IoT 物件執行各種操作。

```
if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
"v2.0"} }, clientConfiguration)) {
    std::cerr << "Exiting because updateThing failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
        clientConfiguration);
    return false;
}

printAsterisksLine();

std::cout << "Now an endpoint will be retrieved for your account.\n" <<
std::endl;
std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
Locator that serves as the entry point\n"
    << "for communication between IoT devices and the AWS IoT service." <<
std::endl;

askQuestion("Press Enter to continue:", alwaysTrueTest);

Aws::String endpoint;
if (!describeEndpoint(endpoint, clientConfiguration)) {
    std::cerr << "Exiting because getEndpoint failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
        clientConfiguration);
    return false;
}
std::cout <<"Your endpoint is " << endpoint << "." << std::endl;
printAsterisksLine();

std::cout << "Now the certificates in your account will be listed." <<
std::endl;
askQuestion("Press Enter to continue:", alwaysTrueTest);
```

```
if (!listCertificates(clientConfiguration)) {
    std::cerr << "Exiting because listCertificates failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}

printAsterisksLine();

std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
std::cout << "A thing shadow refers to a feature that enables you to create a
virtual representation, or \"shadow,\"\n"
<< "of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between\n"
<< "the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow." << std::endl;
askQuestion("Press Enter to continue:", alwaysTrueTest);

if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
    std::cerr << "Exiting because updateThingShadow failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}

printAsterisksLine();

std::cout << "Now, the state information for the shadow will be retrieved.\n"
<< std::endl;
askQuestion("Press Enter to continue:", alwaysTrueTest);

Aws::String shadowState;
if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
    std::cerr << "Exiting because getThingShadow failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}
std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

printAsterisksLine();
```

```

std::cout << "A rule with now be added to to the thing.\n" << std::endl;
std::cout << "Any user who has permission to create rules will be able to
access data processed by the rule." << std::endl;
std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
std::cout << "These resources will be created using a CloudFormation
template." << std::endl;
std::cout << "Stack creation may take a few minutes." << std::endl;

askQuestion("Press Enter to continue: ", alwaysTrueTest);
Aws::Map<Aws::String, Aws::String> outputs
=createCloudFormationStack(STACK_NAME,clientConfiguration);
if (outputs.empty()) {
    std::cerr << "Exiting because createCloudFormationStack failed." <<
std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}

// Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
    std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
    "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack."
<< std::endl;
    cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
            false,
            clientConfiguration);
    return false;
}

Aws::String topicArn = topicArnIter->second;
Aws::String roleArn = roleArnIter->second;
Aws::String sqlStatement = "SELECT * FROM ";
sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
sqlStatement += "";

printAsterisksLine();

std::cout << "Now a rule will be created.\n" << std::endl;
std::cout << "Rules are an administrator-level action. Any user who has
permission\n"

```

```
        << "to create rules will be able to access data processed by the
rule." << std::endl;
    std::cout << "In this case, the rule will use an SNS topic" << std::endl;
    std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
std::endl;
    std::cout << "For more information on IoT SQL, see https://
docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" <<
std::endl;
    Aws::String ruleName = askQuestion("Enter a rule name: ");
    if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
clientConfiguration)) {
        std::cerr << "Exiting because createRule failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
            false,
            clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now your rules will be listed.\n" << std::endl;
    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    if (!listTopicRules(clientConfiguration)) {
        std::cerr << "Exiting because listRules failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
            false,
            clientConfiguration);
        return false;
    }

    printAsterisksLine();
    Aws::String queryString = "thingName:" + thingName;
    std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
    << queryString << "'.\n" << std::endl;
    std::cout << "For query information, see https://docs.aws.amazon.com/iot/
latest/developerguide/query-syntax.html" << std::endl;

    std::cout << "For this query to work, thing indexing must be enabled in your
account.\n"
    << "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
    << "or it can be done programmatically." << std::endl;
    std::cout << "For more information, see https://docs.aws.amazon.com/iot/
latest/developerguide/managing-index.html" << std::endl;
```



```

    if (askYesNoQuestion("Do you want to enable thing indexing in your account?
(y/n) "))
    {
        Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGI

thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnect
    // The ThingGroupIndexingConfiguration object is ignored if not set.
    Aws::IoT::Model::ThingGroupIndexingConfiguration
thingGroupIndexingConfiguration;
        if (!updateIndexingConfiguration(thingIndexingConfiguration,
thingGroupIndexingConfiguration, clientConfiguration)) {
            std::cerr << "Exiting because updateIndexingConfiguration failed." <<
std::endl;
            cleanup(thingName, certificateARN, certificateID, STACK_NAME,
                ruleName, false,
                clientConfiguration);
            return false;
        }
    }

    if (!searchIndex(queryString, clientConfiguration)) {

        std::cerr << "Exiting because searchIndex failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
            false,
            clientConfiguration);
        return false;
    }
}

```

```

//! Update an AWS IoT thing with attributes.
/*!
 \param thingName: The name for the thing.
 \param attributeMap: A map of key/value attributes/
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                             const std::map<Aws::String, Aws::String>
&attributeMap,

```

```

        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
    \param endpointResult: String to receive the endpoint result.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
        "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome =
    iotClient.DescribeEndpoint(
        describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
    }
}

```

```
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" <<
outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::listCertificates(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
iotClient.ListCertificates(
            request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                result.GetCertificates().begin(),
                                result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() <<
std::endl;
            return false;
        }
    }
}
```

```

    }
} while (!marker.empty());

std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

for (auto &certificate: allCertificates) {
    std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
std::endl;
    std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
        << std::endl;
    std::cout << std::endl;
}

return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
 \param thingName: The name for the thing.
 \param document: The state information, in JSON format.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                     const Aws::String &document,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient
iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
std::make_shared<std::stringstream>(
        document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
iotDataPlaneClient.UpdateThingShadow(
        updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
            << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
 \param thingName: The name for the thing.
 \param documentResult: String to receive the state information, in JSON format.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                Aws::String &documentResult,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.
/*!
 \param ruleName: The name for the rule.
 \param snsTopic: The SNS topic ARN for the action.
 \param sql: The SQL statement used to query the topic.
 \param roleARN: The IAM role ARN for the action.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
```

```

        const Aws::String &snsTopicARN, const Aws::String
&sql,
        const Aws::String &roleARN,
        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

//! Lists the AWS IoT topic rules.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::listTopicRules(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

```

```
Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
Aws::String nextToken; // Used for pagination.
do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }

    Aws::IoT::Model::ListTopicRulesOutcome outcome =
iotClient.ListTopicRules(
        request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::ListTopicRulesResult &result =
outcome.GetResult();
        allRules.insert(allRules.end(),
            result.GetRules().cbegin(),
            result.GetRules().cend());

        nextToken = result.GetNextToken();
    }
    else {
        std::cerr << "ListTopicRules error: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} while (!nextToken.empty());

std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
    << std::endl;
for (auto &rule: allRules) {
    std::cout << " Rule name: " << rule.GetRuleName() << ", rule ARN: "
        << rule.GetRuleArn() << "." << std::endl;
}

return true;
}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/developerguide/query-syntax.html
/*!
    \param query: The query string.
    \param clientConfiguration: AWS client configuration.
```

```
\return bool: Function succeeded.
*/
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result =
outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                    result.GetThings().cbegin(),
                                    result.GetThings().cend());
            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "Error in SearchIndex: " <<
outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << " Thing name: " << thingDocument.GetThingName() << "."
                << std::endl;
    }
    return true;
}
```



```
}

```

清理資源。

```
bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
&certificateARN,
                    const Aws::String &certificateID, const Aws::String
&stackName,
                    const Aws::String &ruleName, bool askForConfirmation,
                    const Aws::Client::ClientConfiguration &clientConfiguration)
{
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
        askYesNoQuestion("Delete the rule '" + ruleName +
            "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
    }

    Aws::CloudFormation::CloudFormationClient
cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
        askYesNoQuestion(
            "Delete the CloudFormation stack '" +
stackName +
            "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
        askYesNoQuestion("Delete the certificate '" +
            certificateARN + "'? (y/n)
""))) {
        result &= detachThingPrincipal(certificateARN, thingName,
clientConfiguration);
        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||

```

```

        askYesNoQuestion("Delete the thing '" + thingName
+
        "'? (y/n) ")) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}

```

```

//! Detach a principal from an AWS IoT thing.
/*!
    \param principal: A principal to detach.
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
    iotClient.DetachThingPrincipal(
        detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from
thing "
        << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
        << thingName << ": "
        << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

```
}

//! Delete a certificate.
/*!
 \param certificateID: The ID of a certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome =
    iotClient.DeleteCertificate(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
        std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
        outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
 \param ruleName: The name for the rule.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);
```

```
Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted rule " << ruleName << std::endl;
}
else {
    std::cerr << "Failed to delete rule " << ruleName <<
        ": " << outcome.GetError().GetMessage() << std::endl;
}


return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
}
```

## Java

## 適用於 Java 2.x 的 SDK

 Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iot.IotClient;
import software.amazon.awssdk.services.iot.model.Action;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.AttachThingPrincipalResponse;
import software.amazon.awssdk.services.iot.model.AttributePayload;
import software.amazon.awssdk.services.iot.model.Certificate;
import
    software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointRequest;
import software.amazon.awssdk.services.iot.model.DescribeEndpointResponse;
import software.amazon.awssdk.services.iot.model.DescribeThingRequest;
import software.amazon.awssdk.services.iot.model.DescribeThingResponse;
import software.amazon.awssdk.services.iot.model.DetachThingPrincipalRequest;
import software.amazon.awssdk.services.iot.model.IotException;
import software.amazon.awssdk.services.iot.model.ListCertificatesResponse;
import software.amazon.awssdk.services.iot.model.ListTopicRulesRequest;
import software.amazon.awssdk.services.iot.model.ListTopicRulesResponse;
import software.amazon.awssdk.services.iot.model.SearchIndexRequest;
import software.amazon.awssdk.services.iot.model.SearchIndexResponse;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iot.model.UpdateThingRequest;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient;
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowRequest;
```

```
import software.amazon.awssdk.services.iotdataplane.model.GetThingShadowResponse;
import
    software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowRequest;
import java.net.URI;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates an AWS IoT Thing.
 * 2. Generate and attach a device certificate.
 * 3. Update an AWS IoT Thing with Attributes.
 * 4. Get an AWS IoT Endpoint.
 * 5. List your certificates.
 * 6. Updates the shadow for the specified thing..
 * 7. Write out the state information, in JSON format
 * 8. Creates a rule
 * 9. List rules
 * 10. Search things
 * 11. Detach and delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    private static final String TOPIC = "your-iot-topic";
    public static void main(String[] args) {
        final String usage =
            ""
            Usage:

```

```
        <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
            snsAction - An ARN of an SNS topic.
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String thingName;
    String ruleName;
    String roleARN = args[0];
    String snsAction = args[1];
    Scanner scanner = new Scanner(System.in);
    IotClient iotClient = IotClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS IoT example workflow.");
    System.out.println("""
        This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series
of steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
        It utilizes the AWS SDK for Java V2 and incorporates functionality
for creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS
IoT capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

        """);
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an AWS IoT Thing.");
    System.out.println("""
```

An AWS IoT Thing represents a virtual entity in the AWS IoT service that can be associated with a physical device.

```
        """);
// Prompt the user for input.
System.out.print("Enter Thing name: ");
thingName = scanner.nextLine();
createIoTThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Generate a device certificate.");
System.out.println("""
    A device certificate performs a role in securing the communication
    between devices (Things) and the AWS IoT platform.
    """);

System.out.print("Do you want to create a certificate for " +thingName
+"? (y/n)");
String certAns = scanner.nextLine();
String certificateArn="" ;
if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
    certificateArn = createCertificate(iotClient);
    System.out.println("Attach the certificate to the AWS IoT Thing.");
    attachCertificateToThing(iotClient, thingName, certificateArn);
} else {
    System.out.println("A device certificate was not created.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Update an AWS IoT Thing with Attributes.");
System.out.println("""
    IoT Thing attributes, represented as key-value pairs, offer a
    pivotal advantage in facilitating efficient data
    management and retrieval within the AWS IoT ecosystem.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
updateThing(iotClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Return a unique endpoint specific to the Amazon
Web Services account.");
```



```
System.out.println("""
    An IoT Endpoint refers to a specific URL or Uniform Resource Locator
    that serves as the entry point for communication between IoT devices and the AWS
    IoT service.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
String endpointUrl = describeEndpoint(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List your AWS IoT certificates");
System.out.print("Press Enter to continue...");
scanner.nextLine();
if (certificateArn.length() > 0) {
    listCertificates(iotClient);
} else {
    System.out.println("You did not create a certificates. Skipping this
step.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
System.out.println("""
    A Thing Shadow refers to a feature that enables you to create a
    virtual representation, or "shadow,"
    of a physical device or thing. The Thing Shadow allows you to
    synchronize and control the state of a device between
    the cloud and the device itself. and the AWS IoT service. For
    example, you can write and retrieve JSON data from a Thing Shadow.
    """);
System.out.print("Press Enter to continue...");
scanner.nextLine();
IotDataPlaneClient iotPlaneClient = IotDataPlaneClient.builder()
    .region(Region.US_EAST_1)
    .endpointOverride(URI.create(endpointUrl))
    .build();

updateShadowThing(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
System.out.println("7. Write out the state information, in JSON
format.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
getPayload(iotPlaneClient, thingName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Creates a rule");
System.out.println("""
Creates a rule that is an administrator-level action.
Any user who has permission to create rules will be able to access data
processed by the rule.
""");
System.out.print("Enter Rule name: ");
ruleName = scanner.nextLine();
createIoTRule(iotClient, roleARN, ruleName, snsAction);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. List your rules.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
listIoTRules(iotClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Search things using the Thing name.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
String queryString = "thingName:"+thingName ;
searchThings(iotClient, queryString);
System.out.println(DASHES);

System.out.println(DASHES);
if (certificateArn.length() > 0) {
    System.out.print("Do you want to detach and delete the certificate
for " +thingName +"? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        System.out.println("11. You selected to detach amd delete the
certificate.");
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
    }
}
```

```
        detachThingPrincipal(iotClient, thingName, certificateArn);
        deleteCertificate(iotClient, certificateArn);
    } else {
        System.out.println("11. You selected not to delete the
certificate.");
    }
} else {
    System.out.println("11. You did not create a certificate so there is
nothing to delete.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Delete the AWS IoT Thing.");
System.out.print("Do you want to delete the IoT Thing? (y/n)");
String delAns = scanner.nextLine();
if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
    deleteIoTThing(iotClient, thingName);
} else {
    System.out.println("The IoT Thing was not deleted.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS IoT workflow has successfully completed.");
System.out.println(DASHES);
}

public static void listCertificates(IotClient iotClient) {
    ListCertificatesResponse response = iotClient.listCertificates();
    List<Certificate> certList = response.certificates();
    for (Certificate cert : certList) {
        System.out.println("Cert id: " + cert.certificateId());
        System.out.println("Cert Arn: " + cert.certificateArn());
    }
}

public static void listIoTRules(IotClient iotClient) {
    try {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        ListTopicRulesResponse listTopicRulesResponse =
iotClient.listTopicRules(listTopicRulesRequest);
        System.out.println("List of IoT Rules:");
    }
}
```

```
List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
for (TopicRuleListItem rule : ruleList) {
    System.out.println("Rule Name: " + rule.ruleName());
    System.out.println("Rule ARN: " + rule.ruleArn());
    System.out.println("-----");
}

} catch (IotException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}

public static void createIoTRule(IotClient iotClient, String roleARN, String
ruleName, String action) {
    try {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest =
CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        // Create the rule.
        iotClient.createTopicRule(topicRuleRequest);
        System.out.println("IoT Rule created successfully.");
    }
}
```

```
        } catch (IotException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPayload(IotDataPlaneClient iotPlaneClient, String
thingName) {
        try {
            GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
                .thingName(thingName)
                .build();

            GetThingShadowResponse getThingShadowResponse =
iotPlaneClient.getThingShadow(getThingShadowRequest);

            // Extracting payload from response.
            SdkBytes payload = getThingShadowResponse.payload();
            String payloadString = payload.asUtf8String();
            System.out.println("Received Shadow Data: " + payloadString);

        } catch (IotException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateShadowThing(IotDataPlaneClient iotPlaneClient,
String thingName) {
        try {
            // Create Thing Shadow State Document.
            String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}\"";
            SdkBytes data= SdkBytes.fromString(stateDocument,
StandardCharsets.UTF_8 );
            UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
                .thingName(thingName)
                .payload(data)
                .build();

            // Update Thing Shadow.
            iotPlaneClient.updateThingShadow(updateThingShadowRequest);
```

```
        System.out.println("Thing Shadow updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void updateThing(IotClient iotClient, String thingName) {
    // Specify the new attribute values.
    String newLocation = "Office";
    String newFirmwareVersion = "v2.0";

    Map<String, String> attMap = new HashMap<>();
    attMap.put("location", newLocation);
    attMap.put("firmwareVersion", newFirmwareVersion);

    AttributePayload attributePayload = AttributePayload.builder()
        .attributes(attMap)
        .build();

    UpdateThingRequest updateThingRequest = UpdateThingRequest.builder()
        .thingName(thingName)
        .attributePayload(attributePayload)
        .build();

    try {
        // Update the IoT Thing attributes.
        iotClient.updateThing(updateThingRequest);
        System.out.println("Thing attributes updated successfully.");

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String describeEndpoint(IotClient iotClient) {
    try {
        DescribeEndpointResponse endpointResponse =
iotClient.describeEndpoint(DescribeEndpointRequest.builder().build());

        // Get the endpoint URL.
        String endpointUrl = endpointResponse.endpointAddress();
    }
}
```

```
        String exString = getValue(endpointUrl);
        String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

        System.out.println("Full Endpoint URL: " + fullEndpoint);
        return fullEndpoint;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "" ;
}

public static void detachThingPrincipal(IotClient iotClient, String
thingName, String certificateArn){
    try {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        iotClient.detachThingPrincipal(thingPrincipalRequest);
        System.out.println(certificateArn + " was successfully removed from "
+thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteCertificate(IotClient iotClient, String
certificateArn ) {
    DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
        .certificateId(extractCertificateId(certificateArn))
        .build();

    iotClient.deleteCertificate(certificateProviderRequest);
    System.out.println(certificateArn + " was successfully deleted.");
}
```

```
// Get the cert Id from the Cert ARN value.
private static String extractCertificateId(String certificateArn) {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    String[] arnParts = certificateArn.split(":");
    String certificateIdPart = arnParts[arnParts.length - 1];
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") +
1);
}

public static String createCertificate(IotClient iotClient) {
    try {
        CreateKeysAndCertificateResponse response =
iotClient.createKeysAndCertificate();
        String certificatePem = response.certificatePem();
        String certificateArn = response.certificateArn();

        // Print the details.
        System.out.println("\nCertificate:");
        System.out.println(certificatePem);
        System.out.println("\nCertificate ARN:");
        System.out.println(certificateArn);
        return certificateArn;

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}

public static void attachCertificateToThing(IotClient iotClient, String
thingName, String certificateArn) {
    // Attach the certificate to the thing.
    AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
        .thingName(thingName)
        .principal(certificateArn)
        .build();

    AttachThingPrincipalResponse attachResponse =
iotClient.attachThingPrincipal(principalRequest);

    // Verify the attachment was successful.
```



```
        if (attachResponse.sdkHttpResponse().isSuccessful()) {
            System.out.println("Certificate attached to Thing successfully.");

            // Print additional information about the Thing.
            describeThing(iotClient, thingName);
        } else {
            System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                attachResponse.sdkHttpResponse().statusCode());
        }
    }
}

private static void describeThing(IotClient iotClient, String thingName) {
    try {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        // Print Thing details.
        DescribeThingResponse describeResponse =
iotClient.describeThing(thingRequest);
        System.out.println("Thing Details:");
        System.out.println("Thing Name: " + describeResponse.thingName());
        System.out.println("Thing ARN: " + describeResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteIoTThing(IotClient iotClient, String thingName) {
    try {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        iotClient.deleteThing(deleteThingRequest);
        System.out.println("Deleted Thing " + thingName);

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}

public static void createIoTThing(IotClient iotClient, String thingName) {
    try {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CreateThingResponse createThingResponse =
iotClient.createThing(createThingRequest);
        System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());

    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

private static String getValue(String input) {
    // Define a regular expression pattern for extracting the subdomain.
    Pattern pattern = Pattern.compile("^(.*)\\.iot\\.us-east-1\\.amazonaws\\
\\.com");

    // Match the pattern against the input string.
    Matcher matcher = pattern.matcher(input);

    // Check if a match is found.
    if (matcher.find()) {
        // Extract the subdomain from the first capturing group.
        String subdomain = matcher.group(1);
        System.out.println("Extracted subdomain: " + subdomain);
        return subdomain ;
    } else {
        System.out.println("No match found");
    }
    return "" ;
}

public static void searchThings(IotClient iotClient, String queryString){
    SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
        .queryString(queryString)
        .build();
```

```
    try {
        // Perform the search and get the result.
        SearchIndexResponse searchIndexResponse =
iotClient.searchIndex(searchIndexRequest);

        // Process the result.
        if (searchIndexResponse.things().isEmpty()) {
            System.out.println("No things found.");
        } else {
            searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
        }
    } catch (IotException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
import aws.sdk.kotlin.services.iot.IotClient
import aws.sdk.kotlin.services.iot.model.Action
import aws.sdk.kotlin.services.iot.model.AttachThingPrincipalRequest
import aws.sdk.kotlin.services.iot.model.AttributePayload
import aws.sdk.kotlin.services.iot.model.CreateThingRequest
import aws.sdk.kotlin.services.iot.model.CreateTopicRuleRequest
import aws.sdk.kotlin.services.iot.model.DeleteCertificateRequest
import aws.sdk.kotlin.services.iot.model.DeleteThingRequest
import aws.sdk.kotlin.services.iot.model.DescribeEndpointRequest
import aws.sdk.kotlin.services.iot.model.DescribeThingRequest
import aws.sdk.kotlin.services.iot.model.DetachThingPrincipalRequest
```

```
import aws.sdk.kotlin.services.iot.model.ListTopicRulesRequest
import aws.sdk.kotlin.services.iot.model.SearchIndexRequest
import aws.sdk.kotlin.services.iot.model.SnsAction
import aws.sdk.kotlin.services.iot.model.TopicRulePayload
import aws.sdk.kotlin.services.iot.model.UpdateThingRequest
import aws.sdk.kotlin.services.iotdataplane.IotDataPlaneClient
import aws.sdk.kotlin.services.iotdataplane.model.GetThingShadowRequest
import aws.sdk.kotlin.services.iotdataplane.model.UpdateThingShadowRequest
import aws.smithy.kotlin.runtime.content.ByteString
import aws.smithy.kotlin.runtime.content.toByteArray
import java.util.Scanner
import java.util.regex.Pattern
import kotlin.system.exitProcess

/**
 * Before running this Kotlin code example, ensure that your development
 * environment
 * is set up, including configuring your credentials.
 *
 * For detailed instructions, refer to the following documentation topic:
 * [Setting Up Your Development Environment](https://docs.aws.amazon.com/sdk-for-
kotlin/latest/developer-guide/setup.html)
 *
 * This code example requires an SNS topic and an IAM Role.
 * Follow the steps in the documentation to set up these resources:
 *
 * - [Creating an SNS Topic](https://docs.aws.amazon.com/sns/latest/dg/sns-
getting-started.html#step-create-topic)
 * - [Creating an IAM Role](https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html)
 */

val DASHES = String(CharArray(80)).replace("\u0000", "-")
val TOPIC = "your-iot-topic"

suspend fun main(args: Array<String>) {
    val usage =
        """
        Usage:
            <roleARN> <snsAction>

        Where:
            roleARN - The ARN of an IAM role that has permission to work with AWS
        IOT.
```

```
        snsAction - An ARN of an SNS topic.

        """.trimIndent()

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    var thingName: String
    val roleARN = args[0]
    val snsAction = args[1]
    val scanner = Scanner(System.`in`)

    println(DASHES)
    println("Welcome to the AWS IoT example scenario.")
    println(
        """
        This example program demonstrates various interactions with the AWS
        Internet of Things (IoT) Core service.
        The program guides you through a series of steps, including creating an
        IoT thing, generating a device certificate,
        updating the thing with attributes, and so on.

        It utilizes the AWS SDK for Kotlin and incorporates functionality for
        creating and managing IoT things, certificates, rules,
        shadows, and performing searches. The program aims to showcase AWS IoT
        capabilities and provides a comprehensive example for
        developers working with AWS IoT in a Kotlin environment.
        """.trimIndent(),
    )

    print("Press Enter to continue...")
    scanner.nextLine()
    println(DASHES)

    println(DASHES)
    println("1. Create an AWS IoT thing.")
    println(
        """
        An AWS IoT thing represents a virtual entity in the AWS IoT service that
        can be associated with a physical device.
        """.trimIndent(),
    )
}
```

```
// Prompt the user for input.
print("Enter thing name: ")
thingName = scanner.nextLine()
createIoTThing(thingName)
describeThing(thingName)
println(DASHES)

println(DASHES)
println("2. Generate a device certificate.")
println(
    """"
        A device certificate performs a role in securing the communication
between devices (things) and the AWS IoT platform.
        """"trimIndent(),
)

print("Do you want to create a certificate for $thingName? (y/n)")
val certAns = scanner.nextLine()
var certificateArn: String? = ""
if (certAns != null && certAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
    certificateArn = createCertificate()
    println("Attach the certificate to the AWS IoT thing.")
    attachCertificateToThing(thingName, certificateArn)
} else {
    println("A device certificate was not created.")
}
println(DASHES)

println(DASHES)
println("3. Update an AWS IoT thing with Attributes.")
println(
    """"
        IoT thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
        management and retrieval within the AWS IoT ecosystem.
        """"trimIndent(),
)
print("Press Enter to continue...")
scanner.nextLine()
updateThing(thingName)
println(DASHES)

println(DASHES)
```

```
println("4. Return a unique endpoint specific to the Amazon Web Services
account.")
println(
    """
        An IoT Endpoint refers to a specific URL or Uniform Resource Locator that
serves as the entry point for communication between IoT devices and the AWS IoT
service.
        """.trimIndent(),
    )
print("Press Enter to continue...")
scanner.nextLine()
val endpointUrl = describeEndpoint()
println(DASHES)

println(DASHES)
println("5. List your AWS IoT certificates")
print("Press Enter to continue...")
scanner.nextLine()
if (certificateArn!!.isNotEmpty()) {
    listCertificates()
} else {
    println("You did not create a certificates. Skipping this step.")
}
println(DASHES)

println(DASHES)
println("6. Create an IoT shadow that refers to a digital representation or
virtual twin of a physical IoT device")
println(
    """
        A thing shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
        of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between
        the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a thing shadow.

        """.trimIndent(),
    )
print("Press Enter to continue...")
scanner.nextLine()
updateShawdowThing(thingName)
println(DASHES)
```

```
println(DASHES)
println("7. Write out the state information, in JSON format.")
print("Press Enter to continue...")
scanner.nextLine()
getPayload(thingName)
println(DASHES)

println(DASHES)
println("8. Creates a rule")
println(
    """
    Creates a rule that is an administrator-level action.
    Any user who has permission to create rules will be able to access data
    processed by the rule.
    """).trimIndent(),
)
print("Enter Rule name: ")
val ruleName = scanner.nextLine()
createIoTRule(roleARN, ruleName, snsAction)
println(DASHES)

println(DASHES)
println("9. List your rules.")
print("Press Enter to continue...")
scanner.nextLine()
listIoTRules()
println(DASHES)

println(DASHES)
println("10. Search things using the name.")
print("Press Enter to continue...")
scanner.nextLine()
val queryString = "thingName:$thingName"
searchThings(queryString)
println(DASHES)

println(DASHES)
if (certificateArn.length > 0) {
    print("Do you want to detach and delete the certificate for $thingName?
    (y/n)")
    val delAns = scanner.nextLine()
    if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
    true)) {
        println("11. You selected to detach amd delete the certificate.")
    }
}
```



```
        print("Press Enter to continue...")
        scanner.nextLine()
        detachThingPrincipal(thingName, certificateArn)
        deleteCertificate(certificateArn)
    } else {
        println("11. You selected not to delete the certificate.")
    }
} else {
    println("11. You did not create a certificate so there is nothing to
delete.")
}
println(DASHES)

println(DASHES)
println("12. Delete the AWS IoT thing.")
print("Do you want to delete the IoT thing? (y/n)")
val delAns = scanner.nextLine()
if (delAns != null && delAns.trim { it <= ' ' }.equals("y", ignoreCase =
true)) {
    deleteIoTThing(thingName)
} else {
    println("The IoT thing was not deleted.")
}
println(DASHES)

println(DASHES)
println("The AWS IoT workflow has successfully completed.")
println(DASHES)
}

suspend fun deleteIoTThing(thingNameVal: String) {
    val deleteThingRequest =
        DeleteThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.deleteThing(deleteThingRequest)
        println("Deleted $thingNameVal")
    }
}

suspend fun deleteCertificate(certificateArn: String) {
    val certificateProviderRequest =
```

```
        DeleteCertificateRequest {
            certificateId = extractCertificateId(certificateArn)
        }
        IotClient { region = "us-east-1" }.use { iotClient ->
            iotClient.deleteCertificate(certificateProviderRequest)
            println("$certificateArn was successfully deleted.")
        }
    }

private fun extractCertificateId(certificateArn: String): String? {
    // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
    val arnParts = certificateArn.split(":").toRegex().dropLastWhile
    { it.isEmpty() }.toTypedArray()
    val certificateIdPart = arnParts[arnParts.size - 1]
    return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1)
}

suspend fun detachThingPrincipal(
    thingNameVal: String,
    certificateArn: String,
) {
    val thingPrincipalRequest =
        DetachThingPrincipalRequest {
            principal = certificateArn
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.detachThingPrincipal(thingPrincipalRequest)
        println("$certificateArn was successfully removed from $thingNameVal")
    }
}

suspend fun searchThings(queryStringVal: String?) {
    val searchIndexRequest =
        SearchIndexRequest {
            queryString = queryStringVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        val searchIndexResponse = iotClient.searchIndex(searchIndexRequest)
        if (searchIndexResponse.things?.isEmpty() == true) {
            println("No things found.")
        } else {
```

```

        searchIndexResponse.things
            ?.forEach { thing -> println("Thing id found using search is
${thing.thingId}") }
        }
    }
}

suspend fun listIoTRules() {
    val listTopicRulesRequest = ListTopicRulesRequest {}

    IotClient { region = "us-east-1" }.use { iotClient ->
        val listTopicRulesResponse =
            iotClient.listTopicRules(listTopicRulesRequest)
        println("List of IoT rules:")
        val ruleList = listTopicRulesResponse.rules
        ruleList?.forEach { rule ->
            println("Rule name: ${rule.ruleName}")
            println("Rule ARN: ${rule.ruleArn}")
            println("-----")
        }
    }
}

suspend fun createIoTRule(
    roleARNVal: String?,
    ruleNameVal: String?,
    action: String?,
) {
    val sqlVal = "SELECT * FROM '$TOPIC '"
    val action1 =
        SnsAction {
            targetArn = action
            roleArn = roleARNVal
        }

    val myAction =
        Action {
            sns = action1
        }

    val topicRulePayloadVal =
        TopicRulePayload {
            sql = sqlVal
            actions = listOf(myAction)
        }
}

```

```
    }

    val topicRuleRequest =
        CreateTopicRuleRequest {
            ruleName = ruleNameVal
            topicRulePayload = topicRulePayloadVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createTopicRule(topicRuleRequest)
        println("IoT rule created successfully.")
    }
}

suspend fun getPayload(thingNameVal: String?) {
    val getThingShadowRequest =
        GetThingShadowRequest {
            thingName = thingNameVal
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        val getThingShadowResponse =
            iotPlaneClient.getThingShadow(getThingShadowRequest)
        val payload = getThingShadowResponse.payload
        val payloadString = payload?.let { java.lang.String(it, Charsets.UTF_8) }
        println("Received shadow data: $payloadString")
    }
}

suspend fun listCertificates() {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.listCertificates()
        val certList = response.certificates
        certList?.forEach { cert ->
            println("Cert id: ${cert.certificateId}")
            println("Cert Arn: ${cert.certificateArn}")
        }
    }
}

suspend fun describeEndpoint(): String? {
    val request = DescribeEndpointRequest {}
    IotClient { region = "us-east-1" }.use { iotClient ->
        val endpointResponse = iotClient.describeEndpoint(request)
    }
}
```

```
        val endpointUrl: String? = endpointResponse.endpointAddress
        val exString: String = getValue(endpointUrl)
        val fullEndpoint = "https://$exString-ats.iot.us-east-1.amazonaws.com"
        println("Full endpoint URL: $fullEndpoint")
        return fullEndpoint
    }
}

private fun getValue(input: String?): String {
    // Define a regular expression pattern for extracting the subdomain.
    val pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\\.com")

    // Match the pattern against the input string.
    val matcher = pattern.matcher(input)

    // Check if a match is found.
    if (matcher.find()) {
        val subdomain = matcher.group(1)
        println("Extracted subdomain: $subdomain")
        return subdomain
    } else {
        println("No match found")
    }
    return ""
}

suspend fun updateThing(thingNameVal: String?) {
    val newLocation = "Office"
    val newFirmwareVersion = "v2.0"
    val attMap: MutableMap<String, String> = HashMap()
    attMap["location"] = newLocation
    attMap["firmwareVersion"] = newFirmwareVersion

    val attributePayloadVal =
        AttributePayload {
            attributes = attMap
        }

    val updateThingRequest =
        UpdateThingRequest {
            thingName = thingNameVal
            attributePayload = attributePayloadVal
        }
}
```

```
    IotClient { region = "us-east-1" }.use { iotClient ->
        // Update the IoT thing attributes.
        iotClient.updateThing(updateThingRequest)
        println("$thingNameVal attributes updated successfully.")
    }
}

suspend fun updateShadowThing(thingNameVal: String?) {
    // Create the thing shadow state document.
    val stateDocument = "{\"state\":{\"reported\":{\"temperature\":25, \"humidity\":50}}}"
    val byteStream: ByteStream = ByteStream.fromString(stateDocument)
    val byteArray: ByteArray = byteStream.toByteArray()

    val updateThingShadowRequest =
        UpdateThingShadowRequest {
            thingName = thingNameVal
            payload = byteArray
        }

    IotDataPlaneClient { region = "us-east-1" }.use { iotPlaneClient ->
        iotPlaneClient.updateThingShadow(updateThingShadowRequest)
        println("The thing shadow was updated successfully.")
    }
}

suspend fun attachCertificateToThing(
    thingNameVal: String?,
    certificateArn: String?,
) {
    val principalRequest =
        AttachThingPrincipalRequest {
            thingName = thingNameVal
            principal = certificateArn
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.attachThingPrincipal(principalRequest)
        println("Certificate attached to $thingNameVal successfully.")
    }
}

suspend fun describeThing(thingNameVal: String) {
    val thingRequest =
```

```
        DescribeThingRequest {
            thingName = thingNameVal
        }

// Print Thing details.
IotClient { region = "us-east-1" }.use { iotClient ->
    val describeResponse = iotClient.describeThing(thingRequest)
    println("Thing details:")
    println("Thing name: ${describeResponse.thingName}")
    println("Thing ARN:  ${describeResponse.thingArn}")
}
}

suspend fun createCertificate(): String? {
    IotClient { region = "us-east-1" }.use { iotClient ->
        val response = iotClient.createKeysAndCertificate()
        val certificatePem = response.certificatePem
        val certificateArn = response.certificateArn

        // Print the details.
        println("\nCertificate:")
        println(certificatePem)
        println("\nCertificate ARN:")
        println(certificateArn)
        return certificateArn
    }
}

suspend fun createIoTThing(thingNameVal: String) {
    val createThingRequest =
        CreateThingRequest {
            thingName = thingNameVal
        }

    IotClient { region = "us-east-1" }.use { iotClient ->
        iotClient.createThing(createThingRequest)
        println("Created $thingNameVal")
    }
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱[搭 AWS IoT 配 AWS SDK 使用](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。



# AWS IoT 配額

您可以於 AWS 一般參考中找到 AWS IoT 配額的資訊。

- 如需 AWS IoT Core 配額的詳細資訊，請參閱 [AWS IoT Core 端點和配額](#)。
- 如需 AWS IoT Device Management 配額的詳細資訊，請參閱 [AWS IoT Device Management 端點和配額](#)。
- 如需 AWS IoT Device Defender 配額的詳細資訊，請參閱 [AWS IoT Device Defender 端點和配額](#)。

# AWS IoT Core 定價

您可以在 AWS 行銷頁面和 [AWS 定價計算器](#) 中找到 AWS IoT Core 定價的相關資訊。

- 若要查看 AWS IoT Core 定價資訊，請參閱 [AWS IoT Core 定價](#)。
- 若要預估架構師解決方案的成本，請參閱 [AWS 定價計算器](#)。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。