



使用者指南

AWS IoT SiteWise



AWS IoT SiteWise: 使用者指南

Copyright © 2025 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 SiteWise ?	1
如何 AWS IoT SiteWise 運作	1
擷取工業資料	2
建立資產模型以將收集的資料脈絡化	3
使用查詢、警示和預測進行分析	3
視覺化操作	4
存放資料	4
與其他 服務整合	4
的使用案例 AWS IoT SiteWise	4
生產製造	4
食品和飲料	5
能源與公用事業	5
使用 AWS SDKs	5
概念	6
開始使用	11
要求	11
設定 AWS 帳戶	12
註冊 AWS 帳戶	12
建立具有管理存取權的使用者	12
使用快速入門示範	14
建立 AWS IoT SiteWise 示範	14
刪除 AWS IoT SiteWise 示範	16
教學課程	17
計算 OEE	17
先決條件	17
如何計算 OEE	18
擷取資料	19
先決條件	20
步驟 1：建立 AWS IoT 政策	21
步驟 2：建立 AWS IoT 物件	23
步驟 3：建立裝置資產模型	25
步驟 4：建立裝置機群資產模型	26
步驟 5：建立和設定裝置資產	27
步驟 6：建立和設定裝置機群資產	28

步驟 7：在 AWS IoT Core 中建立規則，以將資料傳送至裝置資產	29
步驟 8：執行裝置用戶端指令碼	32
步驟 9：在教學課程後清除資源	39
在 SiteWise Monitor 中視覺化和共用資料	40
先決條件	41
步驟 1：在 SiteWise Monitor 中建立入口網站	41
步驟 2：登入入口網站	45
步驟 3：建立風力發電廠專案	47
步驟 4：建立儀表板以視覺化風力發電廠資料	50
步驟 5：探索入口網站	56
步驟 6：在教學課程後清除資源	57
發佈至 Amazon DynamoDB	59
先決條件	60
步驟 1：設定 AWS IoT SiteWise 以發佈屬性值更新	60
步驟 2：在 AWS IoT 核心中建立規則	62
步驟 3：建立 DynamoDB 資料表	65
步驟 4：設定 DynamoDB 規則動作	67
步驟 5：探索 DynamoDB 中的資料	68
步驟 6：在教學課程後清除資源	69
將資料擷取至 AWS IoT SiteWise	73
管理資料串流	73
設定許可和設定	74
將資料串流與資產屬性建立關聯	76
取消資料串流與資產屬性的關聯	77
刪除資料串流	77
更新資產屬性別名	78
常用案例	79
使用 AWS IoT SiteWise APIs 擷取資料	81
BatchPutAssetPropertyValue API	81
CreateBulkImportJob API	84
使用 AWS IoT Core 規則	91
授予必要的存取權	92
設定規則動作	93
使用基本擷取降低成本	101
使用 AWS IoT Events 動作	101
使用 AWS IoT Greengrass 串流管理員	102

使用 SiteWise Edge 閘道	103
閘道金鑰概念	103
實作 SiteWise Edge 的優勢	104
自我託管閘道	104
要求	105
建立閘道	110
安裝閘道軟體	112
啟用 MQTT 的 V3 閘道	115
傳統串流、V2 閘道	166
新增資料來源	178
SiteWise Edge 的元件	216
篩選資產	217
Proxy 支援和信任存放區	218
使用 API	224
在 上託管閘道 Siemens Industrial Edge	240
安全	241
Siemens Secure Storage 和 AWS IoT SiteWise Edge 應用程式	242
Siemens Industrial Edge 裝置目的地	242
從預覽應用程式遷移	243
故障診斷	244
AWS IoT SiteWise Edge 應用程式變更日誌	244
要求	245
建立閘道	246
建立 Siemens Databus user	247
存取應用程式	248
安裝應用程式	249
更新已安裝的應用程式組態	251
目的地和路徑篩選條件	251
了解目的地	252
了解路徑篩選條件	254
新增即時目的地	257
使用 Amazon S3 新增緩衝目的地	265
新增路徑篩選條件	272
管理目的地	275
管理閘道	280
使用 AWS IoT SiteWise 主控台管理您的 SiteWise Edge 閘道	281

使用 AWS OpsHub for 管理 SiteWise Edge 閘道 AWS IoT SiteWise	281
使用本機作業系統登入資料存取 SiteWise Edge 閘道	283
管理 SiteWise Edge 閘道憑證	285
變更 SiteWise Edge 閘道元件套件的版本	286
列出 SiteWise Edge 閘道	286
描述 SiteWise Edge 閘道	287
建立 SiteWise Edge 閘道	287
更新 SiteWise Edge 閘道	289
更新閘道功能組態	289
標記閘道資源	291
列出閘道的標籤	292
從閘道移除標籤	293
更新 AWS IoT SiteWise 元件的版本	293
刪除 SiteWise Edge 閘道	294
備份和還原閘道	295
指標資料的每日備份	296
還原 SiteWise Edge 閘道	296
還原 AWS IoT SiteWise 資料	297
驗證成功的備份和還原	298
舊版閘道 (AWS IoT Greengrass Version 1)	299
模型工業資產	300
資產和模型狀態	301
檢查資產的狀態	302
檢查資產或元件模型的狀態	304
資產模型版本	305
擷取資產模型或元件模型的作用中版本 (主控台)	306
擷取資產模型或元件模型的作用中版本 (AWS CLI)	307
自訂複合模型 (元件)	308
內嵌自訂複合模型	308
Component-model-based 自訂複合模型	310
使用路徑來參考自訂複合模型屬性	311
設定物件 IDs	313
使用物件 UUIDs	313
使用外部 IDs	313
建立模型	315
在 中建立資產模型 AWS IoT SiteWise	316

建立元件模型	330
定義資料屬性	333
建立自訂複合模型（元件）	404
建立資產	408
建立資產（主控台）	408
建立資產 (AWS CLI)	409
設定新的資產	410
搜尋資產	411
先決條件	411
上的進階搜尋 AWS IoT SiteWise 主控台	411
更新屬性值	414
關聯和取消關聯資產	416
關聯和取消關聯資產（主控台）	417
關聯和取消關聯資產 (AWS CLI)	418
更新資產和模型	420
在 中更新資產 AWS IoT SiteWise	420
更新資產模型和元件模型	421
更新自訂複合模型（元件）	426
資產模型寫入的樂觀鎖定	429
在 中刪除資產和模型 AWS IoT SiteWise	433
刪除資產	433
刪除資產模型	435
使用資產和模型進行大量操作	437
重要概念和術語	438
受支援的功能	438
大量操作先決條件	439
執行大量匯入任務	441
執行大量匯出任務	443
任務進度追蹤和錯誤處理	446
匯入中繼資料範例	451
匯出中繼資料範例	466
中繼資料傳輸任務結構描述	469
使用警示監控資料	488
警示類型	488
警示狀態	489
警示狀態屬性	490

在資產模型上定義警示	493
警示通知的需求	496
定義 AWS IoT Events 警示	496
定義外部警示	528
在資產上設定警示	530
設定閾值（主控台）	531
設定閾值 (AWS CLI)	531
設定通知設定	533
回應警示	535
回應警示（主控台）	536
回應警示 (API)	539
擷取外部警示狀態	539
映射外部警示狀態串流	540
擷取警示狀態資料	541
AWS IoT SiteWise 助理	543
設定 AWS IoT SiteWise 助理	543
建立資料集	544
編輯資料集	548
刪除資料集	550
AWS IoT SiteWise 助理問題	551
使用 監控資料 AWS IoT SiteWise Monitor	552
SiteWise Monitor 角色	553
SAML 聯合	554
SiteWise Monitor 概念	555
開始使用 AWS IoT SiteWise Monitor（傳統）	556
建立入口網站	557
設定您的入口網站	558
邀請管理員	562
新增入口網站使用者	564
建立儀表板 (CLI)	568
開啟入口網站的警示	573
在邊緣啟用您的入口網站	576
管理您的入口網站	576
開始使用 AWS IoT SiteWise Monitor (AI 感知)	585
建立入口網站	586
設定您的入口網站	587

管理您的入口網站	589
刪除入口網站	593
使用 建立儀表板 AWS CLI	594
入口網站登入	598
建立專案	599
更新專案	599
刪除專案	600
建立儀表板	600
更新儀表板	602
刪除儀表板	603
設定儀表板	603
從 查詢資料 AWS IoT SiteWise	623
查詢目前的資產值	623
查詢資產屬性的目前值 (主控台)	624
查詢資產屬性的目前值 (AWS CLI)	624
查詢歷史資產屬性值	625
查詢資產屬性彙總	627
彙總資產屬性 (API)	627
彙總資產屬性 (AWS CLI)	628
AWS IoT SiteWise 查詢語言	629
先決條件	630
查詢語言參考	631
與其他 服務互動	639
了解 MQTT 主題中的資產屬性	639
使用通知	640
開啟資產屬性通知 (主控台)	640
開啟資產屬性通知 (AWS CLI)	641
查詢通知	643
將資料匯出至 Amazon S3	645
整合 Grafana	646
與 整合 AWS IoT TwinMaker	647
啟用整合	648
整合 AWS IoT SiteWise 和 AWS IoT TwinMaker	648
偵測設備異常	649
新增預測定義 (主控台)	650
訓練預測 (主控台)	653

在預測上開始或停止推論 (主控台)	654
新增預測定義 (CLI)	655
訓練預測並開始推論 (CLI)	658
訓練預測 (CLI)	659
在預測上啟動或停止推論 (CLI)	661
管理資料儲存	664
設定儲存體設定	665
資料保留影響	665
設定暖層 (主控台)	665
設定暖層 (AWS CLI)(	667
設定冷層 (主控台)	670
針對冷方案設定 (AWS CLI)	672
對儲存設定進行故障診斷	678
錯誤：儲存貯體不存在	678
錯誤：拒絕存取 Amazon S3 路徑	678
錯誤：無法擔任角色 ARN	679
錯誤：無法存取跨區域 Amazon S3 儲存貯體	679
儲存在冷層中的資料檔案路徑和結構描述	679
設備資料 (度量)	679
指標、轉換和彙總	683
資產中繼資料	687
資產階層中繼資料	690
儲存資料索引檔案	692
程式碼範例	694
基本概念	698
您好 AWS IoT SiteWise	699
了解基本概念	702
動作	766
安全	841
資料保護	841
網際網路流量隱私權	842
AWS IoT SiteWise 商業服務改進助理	843
資料加密	843
靜態加密	843
傳輸中加密	846
金鑰管理	847

身分與存取管理	849
目標對象	849
使用身分進行驗證	850
AWS IoT SiteWise 如何使用 IAM	852
受管政策	868
服務連結角色	871
設定警示的許可	887
中的跨服務混淆代理人預防 AWS IoT SiteWise	893
對 身分與存取進行疑難排解	894
法規遵循驗證	896
恢復能力	896
基礎架構安全	897
組態與漏洞分析	898
VPC 端點	898
支援的 API 操作	898
建立介面 VPC 端點	901
AWS IoT SiteWise 透過介面 VPC 端點存取	901
建立 VPC 端點政策	903
安全最佳實務	904
在 OPC UA 伺服器上使用身分驗證憑證	904
為您的 OPC UA 伺服器使用加密的通訊模式	904
讓您的元件保持在最新狀態	904
加密 SiteWise Edge 閘道的檔案系統	904
安全存取您的邊緣組態	905
保護 上的資料 Siemens Industrial Edge Management	905
授予 SiteWise Monitor 使用者最低可能許可	905
請勿公開敏感資訊	905
遵循 AWS IoT Greengrass 安全最佳實務	906
另請參閱	906
記錄並監控	907
監控服務日誌	907
管理登入 AWS IoT SiteWise	909
範例：AWS IoT SiteWise 日誌檔案項目	910
監控 SiteWise Edge 閘道日誌	911
使用 Amazon CloudWatch Logs	911
使用服務日誌	913

使用 Amazon CloudWatch 指標進行監控	914
AWS IoT Greengrass Version 2 閘道指標	915
使用 記錄 API 呼叫 AWS CloudTrail	923
AWS IoT SiteWise CloudTrail 中的資訊	923
AWS IoT SiteWise CloudTrail 中的資料事件	924
AWS IoT SiteWise CloudTrail 中的 管理事件	926
範例：AWS IoT SiteWise 日誌檔案項目	926
標記您的 資源	928
在 中使用標籤 AWS IoT SiteWise	928
使用 標記 AWS Management Console	928
使用 AWS IoT SiteWise API 標記	928
搭配 IAM 政策使用標籤	930
疑難排解	931
閘道疑難排解	931
設定和存取 SiteWise Edge 閘道日誌	931
故障診斷 SiteWise Edge 閘道問題	932
對 上的 AWS IoT SiteWise Edge 應用程式進行故障診斷 Siemens Industrial Edge	937
對 Edge 的開放原始碼整合進行故障診斷	938
故障診斷 AWS IoT Greengrass 問題	940
對入口網站進行故障診斷	940
使用者和管理員無法存取 AWS IoT SiteWise 入口網站	941
對 AWS IoT SiteWise 規則動作進行故障診斷	942
設定 AWS IoT Core 日誌	942
設定重新發佈錯誤動作	943
對規則問題進行故障診斷	945
對規則進行故障診斷 (AWS IoT SiteWise)	947
對規則進行故障診斷 (DynamoDB)	948
對大量匯入和匯出進行故障診斷	951
端點和配額	953
端點	953
配額	953
AWS IoT SiteWise 助理 API 限流限制	965
異常偵測的配額	965
文件歷史紀錄	966
.....	cmlxxx

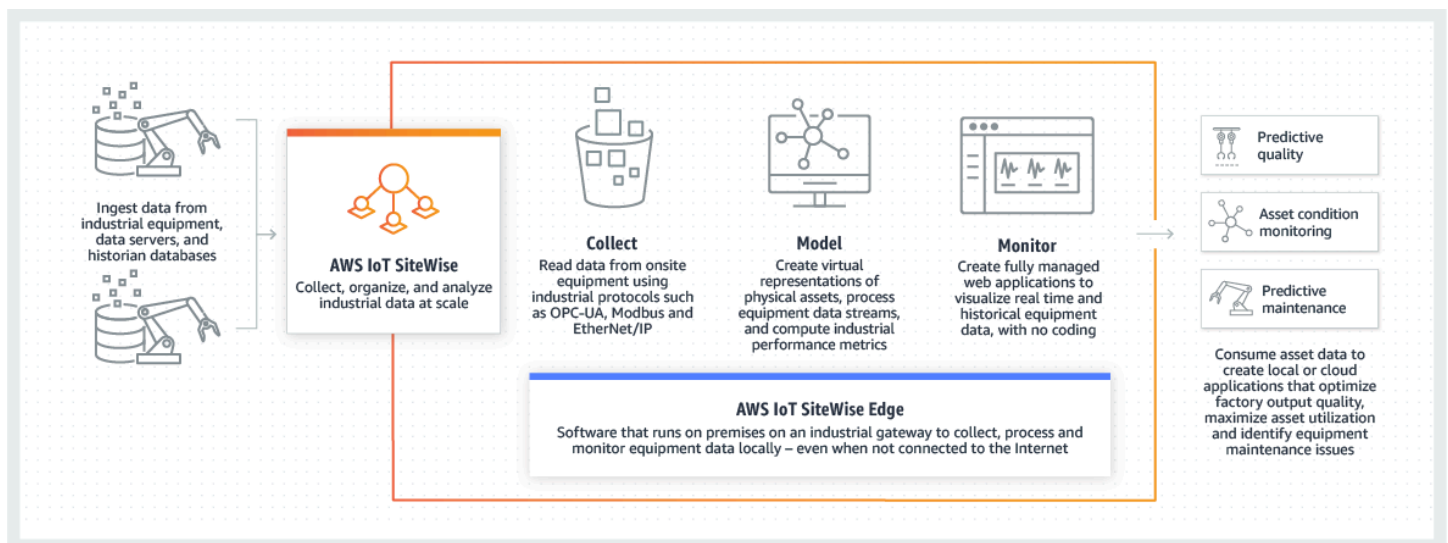
什麼是 AWS IoT SiteWise ？

AWS IoT SiteWise 是一項受管服務，可讓您大規模收集、存放、整理和監控工業設備的資料，協助您做出更好的資料驅動型決策。您可以使用 AWS IoT SiteWise 監控設施間的操作、快速運算常見的工業效能指標，以及建立可分析工業設備資料的應用程式，以防止代價高昂的設備問題並減少生產落差。

透過 AWS IoT SiteWise Monitor，您的營運使用者可以建立 Web 應用程式，以即時檢視和分析工業資料。您可以透過設定和監控指標 (例如，故障之間的平均時間和整體設備效率 (OEE))，來深入了解您的工業操作。

AWS IoT SiteWise Edge 是 的元件 AWS IoT SiteWise，允許收集、儲存和處理本機裝置上的資料。如果您的網際網路存取有限，或需要保持資料的私密性，這會很有用。

下圖顯示 的基本架構 AWS IoT SiteWise：



主題

- [如何 AWS IoT SiteWise 運作](#)
- [的使用案例 AWS IoT SiteWise](#)
- [搭配 AWS SDK 使用此服務](#)
- [AWS IoT SiteWise 概念](#)

如何 AWS IoT SiteWise 運作

AWS IoT SiteWise 提供資源建模架構，可讓您用來建立工業裝置、程序和設施的表示法。設備和程序的呈現方式稱為資產模型 AWS IoT SiteWise。透過資產模型，您可以定義要使用的原始資料，以及如

何將其處理為有用的指標。在 [AWS IoT SiteWise 主控台](#) 中為您的工業操作建置和視覺化資產和模型。您也可以設定資產模型，在邊緣或 AWS 雲端中收集和處理資料。

主題

- [擷取工業資料](#)
- [建立資產模型以將收集的資料脈絡化](#)
- [使用查詢、警示和預測進行分析](#)
- [視覺化操作](#)
- [存放資料](#)
- [與其他 服務整合](#)

擷取工業資料

AWS IoT SiteWise 開始透過擷取工業資料來使用。擷取資料的方式有下列其中一種：

- 從現場伺服器直接擷取：利用 OPC UA 等通訊協定直接從現場裝置讀取資料。將 SiteWise Edge 閘道軟體部署在各種平台上 AWS IoT Greengrass V2，例如常見的工業閘道或虛擬伺服器。您最多可以將 100 個 OPC UA 伺服器連接到單一 AWS IoT SiteWise 閘道。如需詳細資訊，請參閱[AWS IoT SiteWise Edge 自我託管閘道需求](#)。
- 請注意，Modbus TCP 和 Ethernet/IP (EIP) 等通訊協定可透過與 Domatica 的合作內容來支援 AWS IoT Greengrass V2。
- 使用套件進行邊緣資料處理：新增套件以啟用全面的邊緣功能，增強 SiteWise Edge 閘道。使用上的 SiteWise Edge AWS IoT Greengrass V2，資料處理會直接在現場執行，然後使用 AWS IoT Greengrass 串流安全地傳輸至 AWS 雲端。如需詳細資訊，請參閱[在 SiteWise Edge 中設定 OPC UA 來源](#)。
- 透過具有大量操作的 Amazon S3 進行自適應擷取：使用大量資產或資產模型時，請使用大量操作從 Amazon S3 儲存貯體大量匯入和匯出資源。如需詳細資訊，請參閱[使用資產和模型進行大量操作](#)。
- 具有 AWS IoT 核心規則的 MQTT 訊息：對於連線至 AWS IoT Core 傳送 MQTT 訊息的裝置，請使用 AWS IoT Core 規則引擎將這些訊息導向 AWS IoT SiteWise。如果您的裝置連線至 AWS IoT Core 傳送 [MQTT](#) 訊息，請使用 AWS IoT Core 規則引擎將這些訊息路由至 AWS IoT SiteWise。如需詳細資訊，請參閱[AWS IoT SiteWise 使用 AWS IoT Core 規則將資料擷取至](#)。
- 事件觸發的資料擷取：使用 AWS IoT Events 動作在 中設定 IoT SiteWise 動作 AWS IoT Events，以在事件發生 AWS IoT SiteWise 時將資料傳送至。如需詳細資訊，請參閱[AWS IoT SiteWise 從擷取資料至 AWS IoT Events](#)。

- AWS IoT SiteWise API：您的 Edge 或雲端應用程式可以直接傳送資料至 AWS IoT SiteWise。如需詳細資訊，請參閱[使用 AWS IoT SiteWise APIs 擷取資料](#)。

建立資產模型以將收集的資料脈絡化

擷取資料之後，您可以使用資料來建立實體操作的模型，以虛擬方式呈現資產、程序和設施。代表裝置或程序的資產會將資料串流傳輸到 AWS 雲端。資產也可以表示邏輯裝置群組。階層的形成方式是將資產關聯以鏡像複雜的操作。這些階層允許資產從相關聯的子資產存取資料。資產是從資產模型建立的。資產模型是標準化資產格式的宣告式結構。重複使用資產的元件來組織和維護模型。如需詳細資訊，請參閱[模型工業資產](#)。

使用 AWS IoT SiteWise，您可以設定資產，將傳入的資料轉換為內容指標和轉換。

- 接收設備資料時轉換工作。
- 指標會依您定義的間隔計算。

指標和轉換同時適用於個別資產或多個資產。AWS IoT SiteWise 會在與您的設備資料、指標和轉換相關的各種時間範圍內，自動運算常用的統計彙總，例如平均值、總和和計數。

您可以使用 同步資產 AWS IoT TwinMaker。如需詳細資訊，請參閱[整合 AWS IoT SiteWise 和 AWS IoT TwinMaker](#)。

使用查詢、警示和預測進行分析

透過執行查詢和設定警示 AWS IoT SiteWise 來分析與 收集的日期。您也可以使用 Amazon Lookout 自動偵測指標中的異常，並識別其根本原因。

- 設定特定警示，在設備或程序偏離最佳效能時提醒您的團隊，確保快速的問題識別和解決。如需詳細資訊，請參閱[在 中使用警示監控資料 AWS IoT SiteWise](#)。
- 使用 AWS IoT SiteWise API 操作查詢資產屬性的目前值、歷史值和在特定時間間隔的彙總。如需詳細資訊，請參閱[從 查詢資料 AWS IoT SiteWise](#)。
- 搭配 Amazon Lookout for Equipment 使用異常偵測，以識別和視覺化設備或操作條件的變更。透過異常偵測，您可以判斷操作的預防性維護措施。此整合可讓客戶在 AWS IoT SiteWise 和 Amazon Lookout for Equipment 之間同步資料。如需詳細資訊，請參閱[使用 Lookout for Equipment 偵測異常](#)。

視覺化操作

設定 SiteWise Monitor 為您的營運員工建立 Web 應用程式。Web 應用程式可協助員工將操作視覺化。使用 IAM Identity Center 或 IAM 為您的員工處理不同層級的存取。為每位員工設定唯一登入和許可，以檢視整個工業操作的特定子集。為這些員工 AWS IoT SiteWise 提供[應用程式指南](#)，以了解如何使用 SiteWise Monitor。

如需視覺化操作的詳細資訊，請參閱 [使用 監控資料 AWS IoT SiteWise Monitor](#)。

存放資料

您可以將時間序列儲存與您的工業資料湖整合。AWS IoT SiteWise 具有三個工業資料的儲存層：

- 熱儲存層，已針對即時應用程式進行最佳化。
- 針對分析工作負載最佳化的暖儲存層。
- 使用 Amazon S3 的客戶受管冷儲存層，適用於具有高延遲容錯能力的操作資料應用程式。

AWS IoT SiteWise 透過將最近的資料保留在熱儲存層中，協助您管理儲存成本。然後，您可以定義資料保留政策，將歷史資料移至暖層或冷層儲存。如需詳細資訊，請參閱[在 中管理資料儲存 AWS IoT SiteWise](#)。

您也可以匯入和匯出資產中繼資料。如需更多資訊，請參閱[資產中繼資料](#)。

與其他 服務整合

AWS IoT SiteWise 與數個 AWS 服務整合，以在 AWS 雲端中開發完整的 AWS IoT 解決方案。如需詳細資訊，請參閱[與其他 AWS 服務互動](#)。

的使用案例 AWS IoT SiteWise

AWS IoT SiteWise 用於許多產業資料收集和分析應用程式。

從您的所有來源持續收集資料，以協助快速解決問題。AWS IoT SiteWise 提供遠端監控，以直接在現場收集資料，或從多個設施的多個來源收集資料。為工業 IoT 資料解決方案 AWS IoT SiteWise 提供必要的彈性。

生產製造

AWS IoT SiteWise 可簡化從設備收集和利用資料的程序，以精確找出效率低下並盡量減少效率，進而增強工業營運。AWS IoT SiteWise 可協助您從生產線和設備收集資料。使用 AWS IoT SiteWise，您

可以將資料傳輸至 AWS 雲端，並針對特定設備和程序建置效能指標。您可以使用產生的指標來了解營運的整體有效性，並識別創新和改進的機會。您也可以檢視您的製造程序，並識別設備和程序缺陷、生產差距或產品缺陷。

食品和飲料

食品和飲料產業的設施會進行各式各樣的食品加工，包括將穀物磨成麵粉、屠宰和包裝肉類，以及組合、烹調和冷凍可微波的餐點。食品處理工廠通常跨越多個位置，並在集中位置使用工廠和設備操作員來監控流程和設備。例如，冷凍單位會評估配料處理和過期。它們會監控設施之間的浪費建立，以確保營運效率。使用 AWS IoT SiteWise，您可以依生產線和設施將來自多個位置的感應器資料串流分組，以便您的程序工程師能夠更好地了解和改進設施。

能源與公用事業

透過 AWS IoT SiteWise，您可以更輕鬆、更有效率地解決設備問題。您可以遠端和即時監控資產效能。從任何地方存取歷史設備資料，以精確找出潛在問題、分派準確的資源，以及更快地防止和修正問題。

搭配 AWS SDK 使用此服務

AWS 軟體開發套件 (SDKs) 適用於許多熱門的程式設計語言。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
適用於 C++ 的 AWS SDK	適用於 C++ 的 AWS SDK 程式碼範例
AWS CLI	AWS CLI 程式碼範例
適用於 Go 的 AWS SDK	適用於 Go 的 AWS SDK 程式碼範例
適用於 Java 的 AWS SDK	適用於 Java 的 AWS SDK 程式碼範例
適用於 JavaScript 的 AWS SDK	適用於 JavaScript 的 AWS SDK 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
適用於 .NET 的 AWS SDK	適用於 .NET 的 AWS SDK 程式碼範例
適用於 PHP 的 AWS SDK	適用於 PHP 的 AWS SDK 程式碼範例

SDK 文件	代碼範例
AWS Tools for PowerShell	AWS Tools for PowerShell 程式碼範例
適用於 Python (Boto3) 的 AWS SDK	適用於 Python (Boto3) 的 AWS SDK 程式碼範例
適用於 Ruby 的 AWS SDK	適用於 Ruby 的 AWS SDK 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

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。

AWS IoT SiteWise 概念

以下是 的核心概念 AWS IoT SiteWise：

Aggregate

彙總是 AWS IoT SiteWise 自動計算所有時間序列資料的基本指標或衡量值。如需詳細資訊，請參閱 [在中查詢資產屬性彙總 AWS IoT SiteWise](#)。

資產

當您 AWS IoT SiteWise 從工業設備輸入或擷取資料至時，您的裝置、設備和程序都會顯示為資產。每個資產都有相關聯的資料。例如，設備可能具有序號、位置、品牌和模型，以及安裝日期。它也可能具有可用性、效能、品質、溫度、壓力等的時間序列值。將資產分組為階層，允許資產存取存放在其子資產中的資料。如需詳細資訊，請參閱 [模型工業資產](#)。

資產階層

設定資產階層以建立工業操作的邏輯表示法。若要這樣做，請在資產模型中定義階層，並將從該模型建立的資產與指定的階層建立關聯。父資產中的指標可以結合子資產屬性中的資料，讓您計算指標，以深入了解整體操作或其特定部分。如需詳細資訊，請參閱 [定義資產模型階層](#)。

資產模型

每個資產都是使用資產模型來製作。資產模型是定義和標準化資產格式的結構。它們可確保相同類型的多個資產之間的一致資訊，可讓您處理代表裝置群組的資產中的資料。在每個資產模型中，您可以定義**屬性**、時間序列輸入 ([測量](#))、時間序列轉換 ([轉換](#))、時間序列彙整 ([指標](#))，以及**資產階層**。如需詳細資訊，請參閱[模型工業資產](#)。

透過設定邊緣的資產模型，決定處理資產模型屬性的位置。利用此功能來處理和監控本機裝置上的資產資料。

資產屬性

資產屬性是每個資產中存放工業資料的結構。每個屬性都有資料類型，也可以有單位。屬性 (property) 可以是**屬性 (attribute)**、**測量**、**轉換**或**指標**。如需詳細資訊，請參閱[定義資料屬性](#)。

設定資產屬性以在邊緣運算。如需在邊緣處理資料的詳細資訊，請參閱 [在 SiteWise Edge 中設定 OPC UA 來源](#)。

屬性

屬性是資產的屬性，通常保持不變，例如裝置製造商或裝置位置。屬性可以有預設值。從資產模型建立的每個資產都包含該模型中定義之屬性的預設值。如需詳細資訊，請參閱[定義靜態資料 \(屬性\)](#)。

儀表板

每個專案都包含一組儀表板。儀表板可提供一組資產集合值的視覺效果。專案擁有者會建立儀表板和包含的視覺效果。當專案擁有者準備好共用該組儀表板時，擁有者可以將檢視者邀請到專案中，讓他們能夠存取專案中的所有儀表板。如果您需要另一組適用於不同儀表板的檢視者，您必須分割專案之間的儀表板。當瀏覽者查看儀表板時，他們可以自訂時間範圍來查看特定資料。

資料集

資料集是資料集合，代表時間序列資料、non-time-series資料和非設備資料，例如輪班排程、維護記錄和員工資料庫。它們支援外部資料並使用 AWS IoT SiteWise 分析功能。它包含資料集來源、資料集結構描述和資料集參數。AWS IoT SiteWise 助理使用取用 Amazon Kendra 索引的資料集。

資料串流

AWS IoT SiteWise 甚至在建立資產模型和資產之前，將工業資料輸入或擷取至。AWS IoT SiteWise 會自動產生資料串流，以從您的設備收集原始資料串流。

資料串流別名

資料串流別名可協助您輕鬆識別資料串流。例如，別名server1-windfarm/3/turbine/7/temperature表示來自風力發電廠 #3 中渦輪 #7 的溫度值。一詞server1是資料來源名稱，可協助識別 OPC UA 伺服器，server1-並且是連接至從此 OPC UA 伺服器報告之所有資料串流的字首。

資料串流關聯

建立資產模型和資產之後，請將資料串流與資產中定義的資產屬性建立關聯，以建構您的資料。然後，AWS IoT SiteWise 可以使用資產模型和資產來處理來自資料串流的傳入資料。您也可以取消資料串流與資產屬性的關聯。如需詳細資訊，請參閱[管理的資料串流 AWS IoT SiteWise](#)。

目的地

SiteWise Edge 中的目的地代表您要傳送遙測或已處理資料的端點。SiteWise Edge 支援 AWS IoT SiteWise 熱層、緩衝擷取或 Amazon S3 儲存貯體做為目的地。您可以使用路徑篩選條件設定目的地以訂閱特定 MQTT 主題。如需詳細資訊，請參閱[了解 AWS IoT SiteWise Edge 目的地](#)。

公式

每個轉換和指標屬性都隨附一個公式，概述屬性如何轉換或彙總資料。這些公式包括提供的屬性輸入、運算子和函數 AWS IoT SiteWise。如需詳細資訊，請參閱[使用公式表達式](#)。

測量

測量是資產的屬性，描述來自裝置或設備的原始感應器時間序列資料串流。如需詳細資訊，請參閱[從設備定義資料串流（度量）](#)。

指標

指標是代表彙總時間序列資料的資產屬性。每個指標都隨附數學表達式（公式），概述如何彙總資料點和計算該彙總的時間間隔。指標會為每個指定的時間間隔產生單一資料點。如需詳細資訊，請參閱[從屬性和其他資產彙總資料（指標）](#)。

MQTT

MQTT（訊息佇列遙測傳輸）是感應器和裝置的輕量型傳訊通訊協定。

Packs (套件)

SiteWise Edge 閘道使用套件來判斷如何收集、處理和路由資料。如需 SiteWise Edge 閘道可用套件的詳細資訊，請參閱 [the section called “使用套件”](#)。

資料收集套件

使用資料收集套件，讓您的 SiteWise Edge 閘道可以收集工業資料，並將其路由至您選擇的 AWS 目的地。

資料處理套件

使用資料處理套件在邊緣處理、儲存和擷取您的資料，最長可達 30 天。透過 SiteWise Edge APIs，在本機應用程式之間交換邊緣處理的資料。

OPC UA

OPC UA（開放式平台通訊統一架構）是工業自動化的通訊協定。

路徑篩選條件

使用閘道內的路徑篩選條件來訂閱 MQTT 主題，並發佈至 AWS IoT SiteWise 支援的目的地。以 MQTT 為基礎的來源、資料處理管道，以及使用 MQTT 主題在自我託管的啟用 MQTT 的 V3 閘道上將所有交換資料設為目的地。您可以定義主題篩選條件，以指定要擷取或路由至不同目的地的資料。

Portal

AWS IoT SiteWise Monitor 入口網站是一種 Web 應用程式，可用來視覺化和共用 AWS IoT SiteWise 您的資料。入口網站有一或多個管理員，且包含零或多個專案。

入口網站管理員

每個 SiteWise Monitor 入口網站都有一或多個入口網站管理員。入口網站管理員會使用入口網站來建立包含資產和儀表板集合的專案。接著，入口網站管理員會為每個專案指派資產和擁有者。入口網站管理員會透過控制對專案的存取權，指定專案擁有者和檢視者可以看到的資產。

專案

每個 SiteWise Monitor 入口網站都包含一組專案。每個專案都有一個與其相關聯的 AWS IoT SiteWise 資產子集。專案擁有者會建立一或多個儀表板，以便透過一致的方式檢視與這些資產相關聯的資料。專案擁有者可以將檢視者邀請到專案中，讓檢視者能夠檢視專案中的資產和儀表板。專案是 SiteWise Monitor 內共用的基本單位。專案擁有者可以邀請管理員授予入口網站存取權的使用者 AWS。必須先確保使用者具備入口網站存取權，才能與該使用者共用該入口網站中的專案。

專案擁有者

每個 SiteWise Monitor 專案都有擁有者。專案擁有者會以儀表板形式建立視覺效果，以一致的方式呈現營運資料。儀表板準備好共用時，專案擁有者可以將檢視者邀請到專案中。專案擁有者也可為專案指派其他擁有者。專案擁有者可以設定警示的閾值和通知設定。

專案檢視者

每個 SiteWise Monitor 專案都有檢視器。專案檢視者可以連線到入口網站，檢視專案擁有者建立的儀表板。在每個儀表板中，專案檢視器可以調整時間範圍，以更了解操作資料。專案檢視者只能檢視他們可存取之專案中的儀表板。專案檢視者可以確認和暫停警示。

屬性別名

您可以選擇在資產屬性上建立別名，例如 OPC UA 伺服器資料串流路徑（例如 /company/windfarm/3/turbine/7/temperature），簡化資產屬性在擷取或擷取資產資料期間識別。當您使用 [SiteWise Edge 閘道](#) 從伺服器擷取資料時，您的屬性別名必須符合原始資料串流的路徑。如需詳細資訊，請參閱 [管理 的資料串流 AWS IoT SiteWise](#)。

屬性通知

當您啟用資產屬性的屬性通知時，AWS IoT Core 會在每次該屬性收到新值時 AWS IoT SiteWise，發佈 MQTT 訊息至。訊息承載包含該屬性值的更新詳細資訊。使用屬性值通知來建立解決方案，將您的 中的工業資料 AWS IoT SiteWise 與其他 AWS 服務連線。如需詳細資訊，請參閱 [與其他 AWS 服務互動](#)。

SiteWise Edge 閘道

SiteWise Edge 閘道安裝在客戶的現場部署，以收集、處理和直接資料。SiteWise Edge 閘道會透過各種通訊協定連線至工業資料來源，以收集和處理資料，並將其傳送至 AWS 雲端。SiteWise Edge 閘道也可以連線到合作夥伴資料來源。如需詳細資訊，請參閱 [使用 AWS IoT SiteWise Edge 閘道](#)。

轉換

轉換是代表轉換時間序列資料的資產屬性。每個轉換都隨附數學表達式 ([公式](#))，指定如何將資料點從一種形式轉換為另一種形式。轉換的資料點與輸入資料點保持一對一的關係。如需詳細資訊，請參閱 [轉換資料 \(轉換\)](#)。

視覺效果

在每個儀表板中，專案擁有者決定如何顯示與專案相關聯的資產屬性和警示。可用性可能以折線圖表示，而其他值可能以長條圖或關鍵績效指標 (KPIs) 顯示。警示最好顯示為狀態網格和狀態時間表。專案擁有者會自訂每個視覺效果，讓使用者充分理解該資產的資料。

開始使用 AWS IoT SiteWise

您可以使用 AWS IoT SiteWise 收集、組織、分析和視覺化您的資料。

AWS IoT SiteWise 提供示範，您可以用來探索服務，而無需設定真正的資料來源。如需詳細資訊，請參閱 [使用 AWS IoT SiteWise 示範](#)。

您可以完成下列教學課程，以探索 的特定功能 AWS IoT SiteWise：

- [AWS IoT SiteWise 從 AWS IoT 實物擷取資料至](#)
- [在 SiteWise Monitor 中視覺化和共用風力發電廠資料](#)
- [將屬性值更新發佈至 Amazon DynamoDB](#)

如需進一步了解，請參閱下列主題 AWS IoT SiteWise：

- [將資料擷取至 AWS IoT SiteWise](#)
- [模型工業資產](#)
- [在 Edge 上設定 AWS IoT SiteWise 邊緣功能](#)
- [使用 監控資料 AWS IoT SiteWise Monitor](#)
- [從 查詢資料 AWS IoT SiteWise](#)
- [與其他 AWS 服務互動](#)

主題

- [要求](#)
- [設定 AWS 帳戶](#)

要求

您必須擁有 AWS 帳戶才能開始使用 AWS IoT SiteWise。如果您沒有帳戶，請參閱下一節以取得如何設定 帳戶的指示。

使用 AWS IoT SiteWise 可使用的區域。如需詳細資訊，請參閱 [AWS IoT SiteWise 端點和配額](#)。您可以使用 中的區域選擇器 AWS Management Console 來切換到其中一個區域。

設定 AWS 帳戶

主題

- [註冊 AWS 帳戶](#)
- [建立具有管理存取權的使用者](#)

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

註冊 AWS 帳戶

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

註冊程序的一部分包括接聽電話或文字訊息，並在電話鍵盤上輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行[需要根使用者存取權的任務](#)。

AWS 會在註冊程序完成後傳送確認電子郵件給您。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。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶您的電子郵件地址，以帳戶擁有者[AWS Management Console](#)身分登入。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)。

如需說明，請參閱《IAM 使用者指南》中的[為您的 AWS 帳戶 根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

以具有管理存取權的使用者身分登入

- 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 Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》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 Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[建立許可集](#)。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

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

使用 AWS IoT SiteWise 示範

您可以使用 AWS IoT SiteWise AWS IoT SiteWise 示範輕鬆探索。AWS IoT SiteWise 提供示範做為範本，您可以部署該 AWS CloudFormation 範本來建立資產模型、資產和 SiteWise Monitor 入口網站，並產生長達一週的範例資料。

Important

建立示範後，您將開始支付此示範建立和使用的資源的費用。

主題

- [建立 AWS IoT SiteWise 示範](#)
- [刪除 AWS IoT SiteWise 示範](#)

建立 AWS IoT SiteWise 示範

您可以從 AWS IoT SiteWise 主控台建立 AWS IoT SiteWise 示範。

Note

示範會建立 Lambda 函數、一個 CloudWatch Events 規則，以及示範所需的 AWS Identity and Access Management (IAM) 角色。您可能會在 AWS 帳戶中看到這些資源。我們建議您在完成示範之前保留這些資源。如果您刪除資源，示範可能會停止正常運作。

在 AWS IoT SiteWise 主控台中建立示範

1. 導覽至 [AWS IoT SiteWise 主控台](#)，並在頁面右上角尋找 SiteWise 示範。
2. (選用) 在 SiteWise demo (SiteWise 示範) 下，變更 Days to keep demo assets (示範資產保留天數) 欄位以指定示範在刪除之前要保留多少天。
3. (選用) 若要建立 SiteWise Monitor 入口網站來監控範例資料，請執行下列動作。

 Note

您將需要為此示範建立和使用的 SiteWise Monitor 資源付費。如需詳細資訊，請參閱 AWS IoT SiteWise 定價中的 [SiteWise Monitor](#)。

- a. 選擇監控資源。
- b. 選擇許可。
- c. 選擇現有的 IAM 角色，授予聯合 IAM 使用者存取入口網站的權限。

 Important

您的 IAM 角色必須具有下列許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:Describe*",
        "iotsitewise:List*",
        "iotsitewise:Get*",
        "cloudformation:DescribeStacks",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "sso:DescribeRegisteredRegions",
        "organizations:DescribeOrganization"
      ],
      "Resource": "*"
    }
  ]
}
```

如需如何使用 SiteWise Monitor 的詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的 [什麼是 AWS IoT SiteWise Monitor ?](#)。

4. 選擇 Create demo (建立示範)。

示範大約需要 3 分鐘的時間來建立。如果示範無法建立，則表示您的帳戶可能沒有足夠的權限。切換至有管理許可的帳戶，或是使用以下步驟刪除示範並再試一次：

a. 選擇 Delete demo (刪除示範)。

示範大約需要 15 分鐘的時間來刪除。

b. 如果示範無法刪除，請開啟 [AWS CloudFormation 主控台](#)，選擇名為 IoTSiteWiseDemoAssets 的堆疊，然後選擇右上角的 Delete (刪除)。

c. 如果示範再次無法刪除，請依照 AWS CloudFormation 主控台中的步驟略過無法刪除的資源，然後再試一次。

5. 成功建立示範後，您可以在 [AWS IoT SiteWise 主控台](#) 中探索示範資產和資料。

刪除 AWS IoT SiteWise 示範

AWS IoT SiteWise 示範會在一週後自行刪除，或者如果您從 AWS CloudFormation 主控台建立示範堆疊，則會刪除您選擇的天數。如果您已經使用完示範資源，就可以將示範刪除。如果示範無法建立，您也可以將其刪除。使用以下步驟手動刪除示範。

刪除 AWS IoT SiteWise 示範

1. 導覽至 [AWS CloudFormation 主控台](#)。
2. IoTSiteWiseDemoAssets 從 Stacks 清單中選擇。
3. 選擇 刪除。

當您刪除堆疊時，為示範建立的所有資源均會受到刪除。

4. 在確認對話方塊中，選擇 Delete stack (刪除堆疊)。

堆疊大約需要 15 分鐘的時間來刪除。如果示範無法刪除，請再次選擇右上角的 Delete (刪除)。如果示範再次無法刪除，請依照 AWS CloudFormation 主控台中的步驟略過無法刪除的資源，然後再試一次。

AWS IoT SiteWise 教學課程

歡迎使用 AWS IoT SiteWise 教學課程頁面。這個不斷成長的教學課程集合，可讓您掌握導覽複雜度所需的知識和技能 AWS IoT SiteWise。這些教學課程提供各種基本主題，以滿足您的需求。當您深入探索教學課程時，請探索 各方面的寶貴洞見 AWS IoT SiteWise。

每個教學課程都使用特定的設備範例。這些教學課程適用於測試環境，並使用虛構的公司名稱、模型、資產、屬性等。他們的目的是提供一般指導。本教學課程不適用於生產環境中的直接使用，而無需仔細檢閱和調整，以滿足組織的獨特需求。

主題

- [在中計算 OEE AWS IoT SiteWise](#)
- [AWS IoT SiteWise 從 AWS IoT 實物擷取資料至](#)
- [在 SiteWise Monitor 中視覺化和共用風力發電廠資料](#)
- [將屬性值更新發佈至 Amazon DynamoDB](#)

在中計算 OEE AWS IoT SiteWise

本教學提供如何為製造程序計算整體設備效率 (OEE) 的範例。因此，您的 OEE 計算或公式可能與此處顯示的計算或公式不同。一般來說，OEE 被定義為 $\text{Availability} * \text{Quality} * \text{Performance}$ 。若要進一步了解如何計算 OEE，請參閱 Wikipedia [的整體設備效率](#)。

先決條件

若要完成此教學，您必須為具有以下三個資料串流的裝置設定資料擷取：

- Equipment_State – 代表機器狀態的數值代碼，例如閒置、故障、計劃停止或正常操作。
- Good_Count – 資料串流，其中每個資料點包含自上次資料點以來成功操作的數量。
- Bad_Count – 資料串流，其中每個資料點包含自上次資料點以來失敗的操作次數。

若要設定資料擷取，請參閱 [將資料擷取至 AWS IoT SiteWise](#)。如果您沒有可用的工業操作，您可以透過 AWS IoT SiteWise API 撰寫會產生和上傳範例資料指令碼。

如何計算 OEE

在此教學中，您會透過三個資料輸入串流建立會計算 OEE 的資產模型：Equipment_State、Good_Count 和 Bad_Count。在這個範例中，請考慮通用包裝機器，例如，用於包裝糖果、洋芋片或顏料等東西的包裝機器。在 [AWS IoT SiteWise 主控台](#) 中，使用下列測量、轉換和指標建立 AWS IoT SiteWise 資產模型。然後，您可以建立資產來代表封裝機器，並觀察如何 AWS IoT SiteWise 計算 OEE。

定義下列[衡量值](#)，以表示來自包裝機器的原始資料串流。

衡量值

- Equipment_State – 以數字代碼提供封裝機器目前狀態的資料串流（或測量）：
 - 1024 – 機器處於閒置狀態。
 - 1020 – 錯誤，例如錯誤或延遲。
 - 1000 – 計劃的停止。
 - 1111 – 正常操作。
- Good_Count – 資料串流，其中每個資料點包含自上次資料點以來成功操作的數量。
- Bad_Count – 資料串流，其中每個資料點包含自上次資料點以來失敗的操作次數。

使用 Equipment_State 衡量值資料串流及其包含的代碼，定義以下[轉換](#) (或衍生的衡量值)。轉換與原始衡量值有一對一的關係。

轉換

- Idle = eq(Equipment_State, 1024) – 包含機器閒置狀態的轉換資料串流。
- Fault = eq(Equipment_State, 1020) – 包含機器故障狀態的轉換資料串流。
- Stop = eq(Equipment_State, 1000) – 包含機器計劃停止狀態的轉換資料串流。
- Running = eq(Equipment_State, 1111) – 包含機器正常操作狀態的轉換資料串流。

使用原始衡量值和轉換衡量值，定義下列在指定時間間隔內彙總機器資料的[指標](#)。在此區段定義指標時，請為每個指標選擇相同的時間間隔。

指標

- Successes = sum(Good_Count) – 在指定時間間隔內成功填充的套件數量。

- $Failures = \text{sum}(\text{Bad\_Count})$ – 指定時間間隔內未成功填充的套件數量。
- $Idle\_Time = \text{statetime}(\text{Idle})$ – 機器每個指定時間間隔的總閒置時間（以秒為單位）。
- $Fault\_Time = \text{statetime}(\text{Fault})$ – 機器每個指定時間間隔的總故障時間（以秒為單位）。
- $Stop\_Time = \text{statetime}(\text{Stop})$ – 機器每個指定時間間隔的總計劃停止時間（以秒為單位）。
- $Run\_Time = \text{statetime}(\text{Running})$ – 機器每個指定時間間隔執行時沒有問題的總時間（以秒為單位）。
- $Down\_Time = Idle\_Time + Fault\_Time + Stop\_Time$ – 機器在指定時間間隔內的總停機時間（以秒為單位），計算方式為以外機器狀態的總和 Run_Time 。
- $Availability = Run\_Time / (Run\_Time + Down\_Time)$ – 機器的正常運作時間或機器可在指定時間間隔內運作的排程時間百分比。
- $Quality = Successes / (Successes + Failures)$ – 機器在指定時間間隔內成功填充套件的百分比。
- $Performance = ((Successes + Failures) / Run\_Time) / \text{Ideal\_Run\_Rate}$ – 機器在指定時間間隔內的效能，以您程序理想執行率（以秒為單位）的百分比表示。

例如，您的 Ideal_Run_Rate 每分鐘可能有 60 個包裝（每秒 1 個包裝）。如果您的 Ideal_Run_Rate 是每分鐘或每小時，則需要將其除以適當的單位轉換係數，因為 Run_Time 是以秒為單位。

- $OEE = Availability * Quality * Performance$ – 機器在指定時間間隔內的整體設備效率。此公式會將 OEE 計算為 1 的分數。

Note

如果將 OEE 定義為轉換，則會計算每個輸入值的輸出值。當轉換評估考慮公式中所有貢獻屬性的最新可用值時，有可能產生非預期的值。對於具有相同時間戳記的屬性更新，輸出值可能會被其他傳入屬性的更新覆寫。例如，計算可用性、品質和效能時，會使用其他兩個屬性的最後一個可用資料點來計算 OEE。這些貢獻值共用時間戳記，並導致 OEE 的輸出值不正確。不保證轉換運算的順序。

AWS IoT SiteWise 從 AWS IoT 實物擷取資料至

在本教學課程中，了解如何使用裝置陰影 AWS IoT SiteWise，從實 AWS IoT 物機群將資料擷取至。裝置陰影是存放 AWS IoT 裝置目前狀態資訊的 JSON 物件。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[裝置陰影服務](#)。

完成本教學課程後，您可以 AWS IoT SiteWise 根據 AWS IoT 實物在 中設定 操作。透過使用 AWS IoT 實物，您可以將操作與 的其他實用功能整合 AWS IoT。例如，您可以設定 AWS IoT 功能來執行下列任務：

- 設定其他規則將資料串流至 [AWS IoT Events](#)、[Amazon DynamoDB](#) 和其他 AWS 服務。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[規則](#)。
- 使用 AWS IoT 機群索引服務索引、搜尋和彙總您的裝置資料。如需詳細資訊，請參閱《[開發人員指南](#)》中的[機群索引服務](#)。AWS IoT
- 使用 稽核並保護您的裝置 AWS IoT Device Defender。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的 [AWS IoT Device Defender](#)。

在本教學課程中，您將了解如何將資料從 AWS IoT 物件的裝置影子擷取至資產 AWS IoT SiteWise。若要這樣做，您可以建立一或多個 AWS IoT 物件，並執行指令碼，使用 CPU 和記憶體用量資料更新每個物件的裝置影子。您會使用本教學課程中的 CPU 和記憶體用量資料來模擬實際感應器資料。然後，您建立具有 AWS IoT SiteWise 動作的規則，此動作 AWS IoT SiteWise 會在每次物件的裝置影子更新時將此資料傳送至資產。如需詳細資訊，請參閱[AWS IoT SiteWise 使用 AWS IoT Core 規則將資料擷取至](#)。

主題

- [先決條件](#)
- [步驟 1：建立 AWS IoT 政策](#)
- [步驟 2：建立和設定物件 AWS IoT](#)
- [步驟 3：建立裝置資產模型](#)
- [步驟 4：建立裝置機群資產模型](#)
- [步驟 5：建立和設定裝置資產](#)
- [步驟 6：建立和設定裝置機群資產](#)
- [步驟 7：在 AWS IoT Core 中建立規則，以將資料傳送至裝置資產](#)
- [步驟 8：執行裝置用戶端指令碼](#)
- [步驟 9：在教學課程後清除資源](#)

先決條件

為了完成本教學，您需要以下項目：

- AWS 帳戶。如果您沒有帳戶，請參閱 [設定 AWS 帳戶](#)。

- 執行 Windows、Linux、macOS 或 Unix 以存取 的開發電腦 AWS Management Console。如需詳細資訊，請參閱 [AWS Management Console 入門](#)。
- 具有管理員許可的 AWS Identity and Access Management (IAM) 使用者。
- Python 3 安裝在開發電腦上，或安裝在您要註冊為 AWS IoT 物件的裝置上。

步驟 1：建立 AWS IoT 政策

在此程序中，建立允許 AWS IoT 物件存取本教學課程中所用資源 AWS IoT 的政策。

建立 AWS IoT 政策

1. 登入 [AWS Management Console](#)。
2. 檢閱 AWS IoT SiteWise 支援的 [AWS 區域](#)。視需要切換到其中一個支援的區域。
3. 導覽至 [AWS IoT 主控台](#)。如果出現 Connect 裝置按鈕，請選擇它。
4. 在左側導覽窗格中，選擇安全性，然後選擇政策。
5. 選擇建立。
6. 輸入 AWS IoT 政策的名稱（例如 **SiteWiseTutorialDevicePolicy**）。
7. 在政策文件下，選擇 JSON 以 JSON 格式輸入下列政策。將 *region* 和 *account-id* 取代為您的區域和帳戶 ID，例如 **us-east-1** 和 **123456789012**。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:region:account-id:client/SiteWiseTutorialDevice*"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": [
        "arn:aws:iot:region:account-id:topic/$aws/things/
        ${iot:Connection.Thing.ThingName}/shadow/update",
        "arn:aws:iot:region:account-id:topic/$aws/things/
        ${iot:Connection.Thing.ThingName}/shadow/delete",
        "arn:aws:iot:region:account-id:topic/$aws/things/
        ${iot:Connection.Thing.ThingName}/shadow/get"
      ]
    }
  ]
}
```

```

    ]
  },
  {
    "Effect": "Allow",
    "Action": "iot:Receive",
    "Resource": [
      "arn:aws:iot:region:account-id:topic/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/update/accepted",
      "arn:aws:iot:region:account-id:topic/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/delete/accepted",
      "arn:aws:iot:region:account-id:topic/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/get/accepted",
      "arn:aws:iot:region:account-id:topic/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/update/rejected",
      "arn:aws:iot:region:account-id:topic/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/delete/rejected"
    ]
  },
  {
    "Effect": "Allow",
    "Action": "iot:Subscribe",
    "Resource": [
      "arn:aws:iot:region:account-id:topicfilter/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/update/accepted",
      "arn:aws:iot:region:account-id:topicfilter/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/delete/accepted",
      "arn:aws:iot:region:account-id:topicfilter/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/get/accepted",
      "arn:aws:iot:region:account-id:topicfilter/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/update/rejected",
      "arn:aws:iot:region:account-id:topicfilter/$aws/things/
      ${iot:Connection.Thing.ThingName}/shadow/delete/rejected"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iot:GetThingShadow",
      "iot:UpdateThingShadow",
      "iot:DeleteThingShadow"
    ],
    "Resource": "arn:aws:iot:region:account-id:thing/SiteWiseTutorialDevice*"
  }
]

```

```
}
```

此政策可讓 AWS IoT 您的裝置建立連線，並使用 MQTT 訊息與裝置陰影通訊。如需 MQTT 訊息的詳細資訊，請參閱[什麼是 MQTT？](#)。若要與裝置影子互動，您的 AWS IoT 物件會在開頭為的主題上發佈和接收 MQTT 訊息 `$aws/things/thing-name/shadow/`。此政策包含稱為的物件政策變數 `${iot:Connection.Thing.ThingName}`。此變數會取代每個主題中連線的物件名稱。iot:Connect 陳述式會設定哪些裝置可以建立連線的限制，確保物件政策變數只能取代以開頭的名稱 `SiteWiseTutorialDevice`。

如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[物件政策變數](#)。

Note

此政策適用於名稱開頭為 `SiteWiseTutorialDevice` 的實物。若要使用不同的實物名稱，您必須據此更新政策。

8. 選擇建立。

步驟 2：建立和設定物件 AWS IoT

在此程序中，您會建立和設定 AWS IoT 物件。您可以將開發電腦指定為 AWS IoT 物件。隨著您的進度，請記住，您在這裡學習的原則可以套用至實際專案。您可以靈活地在任何能夠執行 AWS IoT SDK 的裝置上建立和設定 AWS IoT 物件，包括 AWS IoT Greengrass 和 FreeRTOS。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[AWS IoT SDKs](#)。

建立和設定 AWS IoT 物件

1. 開啟命令列並執行下列命令，為本教學課程建立目錄。

```
mkdir iot-sitewise-rule-tutorial
cd iot-sitewise-rule-tutorial
```

2. 執行下列命令，為實物的憑證建立目錄。

```
mkdir device1
```

如果您正在建立其他實物，請據此在目錄名稱中使用遞增的編號，以追蹤哪些憑證屬於哪些實物。

3. 導覽至 [AWS IoT 主控台](#)。

4. 在左側導覽窗格中，選擇管理區段中的所有裝置。然後選擇 Things (實物)。
5. 如果出現 You don't have any things yet (您尚未有任何實物) 對話方塊，請選擇 Create a thing (建立實物)。否則，請選擇建立物件。
6. 在建立物件頁面上，選擇建立單一物件，然後選擇下一步。
7. 在指定物件屬性頁面上，輸入物件 AWS IoT 的名稱（例如，**SiteWiseTutorialDevice1**），然後選擇下一步。如果您正在建立其他實物，請據此在實物名稱中使用遞增的編號。

 Important

物件名稱必須符合您在步驟 1：建立政策中建立 AWS IoT 的政策中使用的名稱。否則，您的裝置無法連線到 AWS IoT。

8. 在設定裝置憑證 - 選用頁面上，選擇自動產生新憑證（建議），然後選擇下一步。憑證 AWS IoT 可讓安全地識別您的裝置。
9. 在將政策連接至憑證 - 選用頁面上，選取您在步驟 1：建立 AWS IoT 政策中建立的政策，然後選擇建立物件。
10. 在下載憑證和金鑰對話方塊中，執行下列動作：
 - a. 選擇 Download (下載) 連結，以下載實物的憑證、公有金鑰和私有金鑰。將這三個檔案全部儲存到您為實物憑證建立的目錄 (例如 `iot-sitewise-rule-tutorial/device1`)。

 Important

只有此時能夠下載實物的憑證和金鑰，裝置需要這些資訊才能成功連線到 AWS IoT。

- b. 選擇下載連結以下載根 CA 憑證。將根憑證授權機構憑證儲存至 `iot-sitewise-rule-tutorial`。建議您下載 Amazon 根憑證授權機構 1。
11. 選擇完成。

您現在已在電腦上註冊 AWS IoT 實物。採取下列其中一個後續步驟：

- 繼續步驟 3：建立裝置資產模型而不建立其他 AWS IoT 物件。您可以只使用一個實物完成本教學課程。
- 在另一部電腦或裝置上重複本節的步驟，以建立更多 AWS IoT 實物。針對本教學課程，建議您按照此選項進行，這樣就能從多個裝置擷取唯一的 CPU 和記憶體用量資料。

- 在同一部裝置 (您的電腦) 上重複本節的步驟，以建立更多 AWS IoT 實物。每個 AWS IoT 物件都會從您的電腦接收類似的 CPU 和記憶體用量資料，因此請使用此方法示範從多個裝置擷取非唯一的資料。

步驟 3：建立裝置資產模型

在此程序中，您會在 中建立資產模型 AWS IoT SiteWise，以代表串流 CPU 和記憶體用量資料的裝置。若要處理代表裝置群組的資產中的資料，資產模型會跨相同類型的多個資產強制執行一致的資訊。如需詳細資訊，請參閱[模型工業資產](#)。

建立代表裝置的資產模型

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中選擇 Models (模型)。
3. 選擇建立模型。
4. 在模型詳細資訊下，輸入模型的名稱。例如 **SiteWise Tutorial Device Model**。
5. 在 Measurement definitions (衡量值定義) 底下，執行下列操作：
 - a. 在名稱中，輸入 **CPU Usage**。
 - b. 在 Unit (單位) 中輸入 %。
 - c. 將 Data type (資料類型) 保留為 Double (雙倍)。

衡量值屬性代表裝置的原始資料串流。如需詳細資訊，請參閱[從設備定義資料串流 \(度量\)](#)。

6. 選擇新增測量以新增第二個測量屬性。
7. 在 Measurement definitions (衡量值定義) 底下的第二列中，執行下列操作：
 - a. 在名稱中，輸入 **Memory Usage**。
 - b. 在 Unit (單位) 中輸入 %。
 - c. 將 Data type (資料類型) 保留為 Double (雙倍)。
8. 在 Metric definitions (指標定義) 底下，執行下列操作：
 - a. 在名稱中，輸入 **Average CPU Usage**。
 - b. 在 Formula (公式) 中輸入 **avg(CPU Usage)**。自動完成清單中出現 CPU Usage 時，請選擇此選項。
 - c. 在 Time interval (時間間隔) 中輸入 **5 minutes**。

指標屬性會定義彙總計算，處理某個間隔的所有輸入資料點，並在每個間隔輸出單一資料點。此指標屬性會每隔 5 分鐘計算一次每部裝置的平均 CPU 用量。如需詳細資訊，請參閱[從屬性和其他資產彙總資料（指標）](#)。

9. 選擇新增指標以新增第二個指標屬性。
10. 在 Metric definitions (指標定義) 底下的第二列中，執行下列操作：
 - a. 在名稱中，輸入 **Average Memory Usage**。
 - b. 在 Formula (公式) 中輸入 **avg(Memory Usage)**。自動完成清單中出現 Memory Usage 時，請選擇此選項。
 - c. 在 Time interval (時間間隔) 中輸入 **5 minutes**。

此指標屬性會每隔 5 分鐘計算一次每部裝置的平均記憶體用量。

11. (選用) 新增您想對每部裝置計算的其他指標。有一些有趣的函數，包括 min 和 max。如需詳細資訊，請參閱[使用公式表達式](#)。在步驟 4：建立裝置機群資產模型中，您可以建立父系資產，以使用來自整個裝置機群的資料來計算指標。
12. 選擇建立模型。

步驟 4：建立裝置機群資產模型

在此程序中，您會在 中製作資產模型 AWS IoT SiteWise，以表示裝置集合的符號。在此資產模型中，您會建立結構，可讓您將許多裝置資產連結至一個整體機群資產。接下來，您會在機群資產模型中概述指標，以合併來自所有連線裝置資產的資料。此方法可讓您全面了解整個機群的集體效能。

建立代表裝置機群的資產模型

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中選擇 Models (模型)。
3. 選擇建立模型。
4. 在模型詳細資訊下，輸入模型的名稱。例如 **SiteWise Tutorial Device Fleet Model**。
5. 在 Hierarchy definitions (階層定義) 底下，執行下列操作：
 - a. 在 Hierarchy name (階層名稱) 中輸入 **Device**。
 - b. 在 Hierarchy model (階層模型) 中，選擇裝置資產模型 (**SiteWise Tutorial Device Model**)。

階層會定義父系 (機群) 資產模型與子系 (裝置) 資產模型之間的關係。父系資產可存取子系資產的屬性資料。當您稍後建立資產時，您必須根據父系資產模型中的階層定義，建立子系資產與父系資產的關聯。如需詳細資訊，請參閱[定義資產模型階層](#)。

6. 在 Metric definitions (指標定義) 底下，執行下列操作：
 - a. 在名稱中，輸入 **Average CPU Usage**。
 - b. 在 Formula (公式) 中輸入 **avg(Device | Average CPU Usage)**。自動完成清單出現時，請選擇 Device 以選擇階層，然後選擇 Average CPU Usage 以從您先前建立的裝置資產中選擇指標。
 - c. 在 Time interval (時間間隔) 中輸入 **5 minutes**。

此指標屬性會透過 **Device** 階層，計算與機群資產相關聯之所有裝置資產的平均 CPU 用量。

7. 選擇新增指標以新增第二個指標屬性。
8. 在 Metric definitions (指標定義) 底下的第二列中，執行下列操作：
 - a. 在名稱中，輸入 **Average Memory Usage**。
 - b. 在 Formula (公式) 中輸入 **avg(Device | Average Memory Usage)**。自動完成清單出現時，請選擇 Device 以選擇階層，然後選擇 Average Memory Usage 以從您先前建立的裝置資產中選擇指標。
 - c. 在 Time interval (時間間隔) 中輸入 **5 minutes**。

此指標屬性會透過 **Device** 階層，計算與機群資產相關聯之所有裝置資產的平均記憶體用量。

9. (選用) 新增您想在裝置機群中計算的其他指標。
10. 選擇建立模型。

步驟 5：建立和設定裝置資產

在此程序中，您會根據裝置資產模型產生裝置資產。接著，您會定義每個衡量值屬性的屬性別名。屬性別名是可識別資產屬性的唯一字串。稍後，您可以使用別名來識別資料上傳的屬性，而不是資產 ID 和屬性 ID。如需詳細資訊，請參閱[管理的資料串流 AWS IoT SiteWise](#)。

建立裝置資產和定義屬性別名

1. 導覽至 [AWS IoT SiteWise 主控台](#)。

2. 在左側導覽窗格中，選擇 Assets (資產)。
3. 選擇 Create asset (建立資產)。
4. 在模型資訊下，選擇您的裝置資產模型 **SiteWise Tutorial Device Model**。
5. 在資產資訊下，輸入資產的名稱。例如 **SiteWise Tutorial Device 1**。
6. 選擇 Create asset (建立資產)。
7. 針對新裝置資產，選擇 Edit (編輯)。
8. 在 CPU Usage 底下，輸入 **/tutorial/device/SiteWiseTutorialDevice1/cpu** 做為屬性別名。您可以在屬性別名中包含 AWS IoT 物件的名稱，以便使用單一 AWS IoT 規則從所有裝置擷取資料。
9. 在 Memory Usage 底下，輸入 **/tutorial/device/SiteWiseTutorialDevice1/memory** 做為屬性別名。
10. 選擇儲存。

如果您先前已建立多個 AWS IoT 物件，請為每個裝置重複步驟 3 到 10，並相應地增加資產名稱和屬性別名中的數字。例如，第二個裝置資產的名稱應為 **SiteWise Tutorial Device 2**，而其屬性別名應為 **/tutorial/device/SiteWiseTutorialDevice2/cpu** 及 **/tutorial/device/SiteWiseTutorialDevice2/memory**。

步驟 6：建立和設定裝置機群資產

在此程序中，您會形成衍生自裝置機群資產模型的裝置機群資產。然後，您將個別裝置資產連結至機群資產。此關聯可讓機群資產的指標屬性從多個裝置編譯和分析資料。此資料為您提供整個機群集體效能的合併檢視。

建立裝置機群資產並與裝置資產建立關聯

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中，選擇 Assets (資產)。
3. 選擇 Create asset (建立資產)。
4. 在模型資訊下，選擇您的裝置機群資產模型 **SiteWise Tutorial Device Fleet Model**。
5. 在資產資訊下，輸入資產的名稱。例如 **SiteWise Tutorial Device Fleet 1**。
6. 選擇 Create asset (建立資產)。
7. 針對新裝置機群資產，選擇 Edit (編輯)。
8. 在與此資產相關聯的資產下，選擇新增相關聯的資產並執行下列動作：

- a. 在 Hierarchy (階層) 底下，選擇 Device。此階層會識別裝置和裝置機群資產之間的階層關係。您先前已按照本教學課程在裝置機群資產模型中定義此階層。
 - b. 在 Asset (資產) 底下，選擇裝置資產 SiteWise Tutorial Device 1。
9. (選用) 如果您先前已建立多個裝置資產，請為您建立的每個裝置資產重複步驟 8 到 10。
 10. 選擇儲存。

您現在應該會看到裝置資產整理為階層的形式。

步驟 7：在 AWS IoT Core 中建立規則，以將資料傳送至裝置資產

在此程序中，您會在 中建立規則 AWS IoT Core。此規則旨在解譯來自裝置影子的通知訊息，並將資料傳輸至 中的裝置資產 AWS IoT SiteWise。每次您裝置的影子更新時，都會 AWS IoT 傳送 MQTT 訊息。您可建立規則，根據 MQTT 訊息在裝置陰影變更時執行動作。在此情況下，目標是處理更新訊息、擷取屬性值，並將其傳輸至您的裝置資產 AWS IoT SiteWise。

使用 AWS IoT SiteWise 動作建立規則

1. 導覽至 [AWS IoT 主控台](#)。
2. 在左側導覽窗格中，選擇訊息路由，然後選擇規則。
3. 選擇建立規則。
4. 輸入規則的名稱和描述，然後選擇下一步。
5. 輸入下列 SQL 陳述式，然後選擇下一步。

```
SELECT
  *
FROM
  '$aws/things/+/shadow/update/accepted'
WHERE
  startsWith(topic(3), "SiteWiseTutorialDevice")
```

此規則查詢陳述式能夠運作的原因在於，裝置陰影服務會將陰影更新發佈到 `$aws/things/thingName/shadow/update/accepted`。如需裝置影子的詳細資訊，請參閱《AWS IoT 開發人員指南》中的 [裝置影子服務](#)。

在 WHERE 子句中，此規則查詢陳述式使用 `topic(3)` 函數，從本主題的第三個區段中取得實物名稱。接著，陳述式會篩選掉那些具有不符合教學課程裝置之名稱的裝置。如需 AWS IoT SQL 的詳細資訊，請參閱《AWS IoT 開發人員指南》中的 [AWS IoT SQL 參考](#)。

6. 在規則動作下，選擇將訊息資料傳送至 中的資產屬性 AWS IoT SiteWise，然後執行下列動作：
 - a. 選擇 By property alias (依據屬性別名)。
 - b. 在 Property alias (屬性別名) 中輸入 `/tutorial/device/${topic(3)}/cpu`。

`${...}` 語法是替代範本。會 AWS IoT 評估括號中的內容。此替代範本會從主題中提取實物名稱，建立每個實物的唯一別名。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的 [替代範本](#)。

 Note

因為替代範本中的運算式與 SELECT 陳述式是分開計算的，所以您不能使用替代範本來參考使用 AS 子句所建立的別名。除了支援的函數與運算子以外，您可以僅參照原始承載中出現的資訊。

- c. 在項目 ID - 選用中，輸入 `${concat(topic(3), "-cpu-", floor(state.reported.timestamp))}`。

項目 ID 會唯一識別每個值項目的嘗試動作。如果項目傳回錯誤，您可以在錯誤輸出內容中尋找項目 ID，以針對問題進行故障診斷。此項目 ID 中的替代範本會結合實物名稱與裝置回報的時間戳記。例如，產生的項目 ID 看起來可能與 `SiteWiseTutorialDevice1-cpu-1579808494` 類似。

- d. 在 Time in seconds (以秒為單位的時間) 中輸入 `${floor(state.reported.timestamp)}`。

此替代範本會從裝置回報的時間戳記中計算時間 (以秒為單位)。在本教學課程中，裝置會回報 Unix Epoch 時間格式的時間戳記 (以秒為單位)，做為浮點數。

- e. 在以奈米為單位的位移 - 選用，輸入 `${floor((state.reported.timestamp % 1) * 1E9)}`。

此替代範本會轉換裝置回報的時間戳記的小數部分，藉此從時間中計算奈米秒位移 (以秒為單位)。

Note

AWS IoT SiteWise 需要您的資料具有以 Unix epoch 時間為單位的目前時間戳記。如果您的裝置未準確報告時間，您可以從具有 [timestamp\(\)](#) 的 AWS IoT 規則引擎取得目前的時間。此函數會回報以毫秒為單位的時間，因此您必須將規則動作的時間參數更新為以下的值：

- 在 Time in seconds (以秒為單位的時間) 中輸入 `${floor(timestamp() / 1E3)}`。
- 在 Offset in nanos (以奈米為單位的位移) 中輸入 `${(timestamp() % 1E3) * 1E6}`。

- f. 在 Data type (資料類型) 中選擇 Double (雙倍)。

此資料類型必須符合您在資產模型中定義之資產屬性的資料類型。

- g. 在 Value (值) 中輸入 `${state.reported.cpu}`。在替代範本中，您會使用 `.` 運算子，以從 JSON 結構內擷取值。
- h. 選擇 Add entry (新增項目) 以新增記憶體用量屬性的新項目，然後再次針對該屬性完成下列步驟：
- 選擇 By property alias (依據屬性別名)。
 - 在 Property alias (屬性別名) 中輸入 `/tutorial/device/${topic(3)}/memory`。
 - 在項目 ID - 選用中，輸入 `${concat(topic(3), "-memory-", floor(state.reported.timestamp))}`。
 - 在 Time in seconds (以秒為單位的時間) 中輸入 `${floor(state.reported.timestamp)}`。
 - 在以奈米為單位的位移 - 選用，輸入 `${floor((state.reported.timestamp % 1) * 1E9)}`。
 - 在 Data type (資料類型) 中選擇 Double (雙倍)。
 - 在 Value (值) 中輸入 `${state.reported.memory}`。
- i. 在 IAM 角色下，選擇建立新角色，為此規則動作建立 IAM 角色。此角色允許 AWS IoT 將資料推送至裝置機群資產及其資產階層中的屬性。
- j. 輸入角色名稱，然後選擇建立。

7. (選用) 設定可用來針對規則進行故障診斷的錯誤動作。如需詳細資訊，請參閱[對規則進行故障診斷 \(AWS IoT SiteWise\)](#)。
8. 選擇下一步。
9. 檢閱設定，然後選擇建立以建立規則。

步驟 8：執行裝置用戶端指令碼

在本教學課程中，您未使用實際裝置來報告資料。反之，您會執行指令碼以使用 CPU 和記憶體用量來更新 AWS IoT 物件的裝置影子，以模擬真實的感應器資料。若要執行指令碼，您必須先安裝必要的 Python 套件。在此程序中，您會安裝所需的 Python 套件，然後執行裝置用戶端指令碼。

設定和執行裝置用戶端指令碼

1. 導覽至 [AWS IoT 主控台](#)。
2. 在左側導覽窗格的底部，選擇 Settings (設定)。
3. 儲存自訂端點，以與裝置用戶端指令碼搭配使用。您會使用此端點與實物的陰影進行互動。此端點在您目前區域的帳戶中是唯一的。

您的自訂端點看起來應該會與以下範例類似。

```
identifier.iot.region.amazonaws.com
```

4. 開啟命令列並執行下列命令，以導覽到您先前建立的教學課程目錄。

```
cd iot-sitewise-rule-tutorial
```

5. 執行下列命令以安裝適用於 Python 的 AWS IoT Device SDK。

```
pip3 install AWSIoTPythonSDK
```

如需詳細資訊，請參閱《AWS IoT 開發人員指南[適用於 Python 的 AWS IoT Device SDK](#)》中的

6. 執行下列命令來安裝 psutil，這是跨平台程序和系統公用程式程式庫。

```
pip3 install psutil
```

如需詳細資訊，請參閱 Python 套件索引中的 [psutil](#)。

7. 在 `iot-sitewise-rule-tutorial` 目錄中建立名為 `thing_performance.py` 的檔案，然後將下列 Python 程式碼複製到檔案中。

```
import AWSIoTPythonSDK.MQTTLib as AWSIoTPyMQTT

import json
import psutil
import argparse
import logging
import time

# Configures the argument parser for this program.
def configureParser():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-e",
        "--endpoint",
        action="store",
        required=True,
        dest="host",
        help="Your AWS IoT custom endpoint",
    )
    parser.add_argument(
        "-r",
        "--rootCA",
        action="store",
        required=True,
        dest="rootCAPath",
        help="Root CA file path",
    )
    parser.add_argument(
        "-c",
        "--cert",
        action="store",
        required=True,
        dest="certificatePath",
        help="Certificate file path",
    )
    parser.add_argument(
        "-k",
        "--key",
        action="store",
```

```
        required=True,
        dest="privateKeyPath",
        help="Private key file path",
    )
    parser.add_argument(
        "-p",
        "--port",
        action="store",
        dest="port",
        type=int,
        default=8883,
        help="Port number override",
    )
    parser.add_argument(
        "-n",
        "--thingName",
        action="store",
        required=True,
        dest="thingName",
        help="Targeted thing name",
    )
    parser.add_argument(
        "-d",
        "--requestDelay",
        action="store",
        dest="requestDelay",
        type=float,
        default=1,
        help="Time between requests (in seconds)",
    )
    parser.add_argument(
        "-v",
        "--enableLogging",
        action="store_true",
        dest="enableLogging",
        help="Enable logging for the AWS IoT Device SDK for Python",
    )
    return parser

# An MQTT shadow client that uploads device performance data to AWS IoT at a
# regular interval.
class PerformanceShadowClient:
    def __init__(
```

```

        self,
        thingName,
        host,
        port,
        rootCAPath,
        privateKeyPath,
        certificatePath,
        requestDelay,
    ):
        self.thingName = thingName
        self.host = host
        self.port = port
        self.rootCAPath = rootCAPath
        self.privateKeyPath = privateKeyPath
        self.certificatePath = certificatePath
        self.requestDelay = requestDelay

    # Updates this thing's shadow with system performance data at a regular
    interval.
    def run(self):
        print("Connecting MQTT client for {}".format(self.thingName))
        mqttClient = self.configureMQTTClient()
        mqttClient.connect()
        print("MQTT client for {} connected".format(self.thingName))
        deviceShadowHandler = mqttClient.createShadowHandlerWithName(
            self.thingName, True
        )

        print("Running performance shadow client for {}".format(self.thingName))
        while True:
            performance = self.readPerformance()
            print("[{}]" .format(self.thingName))
            print("CPU:\t{}%".format(performance["cpu"]))
            print("Memory:\t{}%\n".format(performance["memory"]))
            payload = {"state": {"reported": performance}}
            deviceShadowHandler.shadowUpdate(
                json.dumps(payload), self.shadowUpdateCallback, 5
            )
            time.sleep(args.requestDelay)

    # Configures the MQTT shadow client for this thing.
    def configureMQTTClient(self):
        mqttClient = AWSIoTPyMQTT.AWSIoTMQTTShadowClient(self.thingName)

```

```

        mqttClient.configureEndpoint(self.host, self.port)
        mqttClient.configureCredentials(
            self.rootCAPath, self.privateKeyPath, self.certificatePath
        )
        mqttClient.configureAutoReconnectBackoffTime(1, 32, 20)
        mqttClient.configureConnectDisconnectTimeout(10)
        mqttClient.configureMQTTOperationTimeout(5)
        return mqttClient

# Returns the local device's CPU usage, memory usage, and timestamp.
def readPerformance(self):
    cpu = psutil.cpu_percent()
    memory = psutil.virtual_memory().percent
    timestamp = time.time()
    return {"cpu": cpu, "memory": memory, "timestamp": timestamp}

# Prints the result of a shadow update call.
def shadowUpdateCallback(self, payload, responseStatus, token):
    print("[{}].format(self.thingName))
    print("Update request {} {} \n".format(token, responseStatus))

# Configures debug logging for the AWS IoT Device SDK for Python.
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)

# Runs the performance shadow client with user arguments.
if __name__ == "__main__":
    parser = configureParser()
    args = parser.parse_args()
    if args.enableLogging:
        configureLogging()
    thingClient = PerformanceShadowClient(
        args.thingName,
        args.host,
        args.port,

```

```

    args.rootCAPath,
    args.privateKeyPath,
    args.certificatePath,
    args.requestDelay,
)
thingClient.run()

```

8. 從命令列搭配下列參數執行 `thing_performance.py`：

- `-n` , `--thingName` – 您的物件名稱，例如 **SiteWiseTutorialDevice1**。
- `-e` , `--endpoint` – 您在此程序中稍早儲存的自訂 AWS IoT 端點。
- `-r` , `--rootCA` – AWS IoT 根 CA 憑證的路徑。
- `-c` , `--cert` – 物件 AWS IoT 憑證的路徑。
- `-k` , `--key` – 物件 AWS IoT 憑證私有金鑰的路徑。
- `-d` , `--requestDelay` – (選用) 每次裝置陰影更新之間等待的時間，以秒為單位。預設為 1 秒。
- `-v` , `--enableLogging` – (選用) 如果此參數存在，指令碼會從 列印偵錯訊息 適用於 Python 的 AWS IoT Device SDK。

命令看起來應該會與以下範例類似。

```

python3 thing_performance.py \
  --thingName SiteWiseTutorialDevice1 \
  --endpoint identifier.iot.region.amazonaws.com \
  --rootCA AmazonRootCA1.pem \
  --cert device1/thing-id-certificate.pem.crt \
  --key device1/thing-id-private.pem.key

```

如果您正在執行指令碼以取得其他 AWS IoT 物件，請相應地更新物件名稱和憑證目錄。

9. 嘗試開啟和關閉裝置上的程式，看看 CPU 和記憶體用量有何變化。指令碼會列印每個 CPU 和記憶體用量的讀值。如果指令碼成功將資料上傳到裝置陰影服務，指令碼的輸出看起來應該會與以下範例類似。

```

[SiteWiseTutorialDevice1]
CPU:      24.6%
Memory: 85.2%

[SiteWiseTutorialDevice1]

```

Update request e6686e44-fca0-44db-aa48-3ca81726f3e3 accepted

10. 請按照下列步驟，確認指令碼是否正在更新裝置陰影：

- 導覽至 [AWS IoT 主控台](#)。
- 在左側導覽窗格中，選擇所有裝置，然後選擇實物。
- 選擇您的物件 SiteWiseTutorialDevice。
- 選擇 Device Shadows 索引標籤，選擇 Classic Shadow，然後確認 Shadow 狀態看起來如下範例所示。

```
{
  "reported": {
    "cpu": 24.6,
    "memory": 85.2,
    "timestamp": 1579567542.2835066
  }
}
```

如果您物件的影子狀態為空，或看起來與上一個範例不同，請檢查指令碼是否正在執行並成功連線 AWS IoT。如果指令碼在連線至 時繼續逾時 AWS IoT，請檢查您的[物件政策](#)是否已根據本教學課程設定。

11. 遵循下列步驟，確認規則動作是否正在將資料傳送到 AWS IoT SiteWise：

- 導覽至 [AWS IoT SiteWise 主控台](#)。
- 在左側導覽窗格中，選擇 Assets (資產)。
- 選擇裝置機群資產 (SiteWise Tutorial Device Fleet 1 1) 旁的箭號以展開其資產階層，然後選擇裝置資產 (SiteWise Tutorial Device 1)。
- 選擇 Measurements (衡量值)。
- 確認 Latest value (最新值) 儲存格有 CPU Usage 和 Memory Usage 屬性的值。

Measurements				
Name	Alias	Notification status	Notification topic	Latest value
CPU Usage	/tutorial/device/SiteWiseTutorialDevice1/cpu	⊖ Disabled	–	24.6
Memory Usage	/tutorial/device/SiteWiseTutorialDevice1/memory	⊖ Disabled	–	85.2

- 如果 CPU Usage 和 Memory Usage 屬性沒有最新的值，請重新整理頁面。如果值在幾分鐘後未出現，請參閱[對規則進行故障診斷 \(AWS IoT SiteWise\)](#)。

您已完成本教學課程。若要探索資料的即時視覺效果，您可以在 AWS IoT SiteWise Monitor 中設定入口網站。如需詳細資訊，請參閱[使用 監控資料 AWS IoT SiteWise Monitor](#)。否則，您可以在命令提示字元中按 CTRL+C，以停止裝置用戶端指令碼。Python 程式不太可能傳送足夠的訊息因而衍生費用，但最佳實務是在完成時停止程式。

步驟 9：在教學課程後清除資源

完成從 AWS IoT 實物擷取資料的教學課程後，請清理您的資源，以避免產生額外費用。

在 中刪除階層式資產 AWS IoT SiteWise

1. 導覽至 [AWS IoT SiteWise 主控台](#)
2. 在左側導覽窗格中，選擇 Assets (資產)。
3. 當您在 中刪除資產時 AWS IoT SiteWise，您必須先取消它們的關聯。

完成下列步驟，以取消裝置資產與裝置機群資產的關聯：

- a. 選擇您的裝置機群資產 ()SiteWise Tutorial Device Fleet 1。
- b. 選擇編輯。
- c. 在 Assets associated to this asset (與此資產相關聯的資產) 底下，針對與此裝置機群資產相關聯的每個裝置資產選擇 Disassociate (取消關聯)。
- d. 選擇儲存。

您現在應該會看到裝置資產不再整理為階層的形式。

4. 選擇裝置資產 (SiteWise Tutorial Device 1)。
5. 選擇 刪除。
6. 在確認對話方塊中輸入 **Delete**，然後選擇 Delete (刪除)。
7. 為每個裝置資產和裝置機群資產 () 重複步驟 4 到 6SiteWise Tutorial Device Fleet 1。

在 中刪除階層式資產模型 AWS IoT SiteWise

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 如果您尚未刪除裝置和裝置機群資產，請執行此操作。如需詳細資訊，請參閱[上一個程序](#)。如果您有已從該模型建立的資產，則無法刪除模型。
3. 在左側導覽窗格中選擇 Models (模型)。
4. 選擇裝置機群資產模型 (SiteWise Tutorial Device Fleet Model 1)。

刪除階層資產模型時，請先刪除父資產模型。

5. 選擇 刪除。
6. 在確認對話方塊中輸入 **Delete**，然後選擇 Delete (刪除)。
7. 針對裝置資產模型 (SiteWise Tutorial Device Model) 重複步驟 4 到 6。

在 中停用或刪除規則 AWS IoT Core

1. 導覽至 [AWS IoT 主控台](#)。
2. 在左側導覽窗格中，選擇訊息路由，然後選擇規則。
3. 選取您的規則，然後選擇刪除。
4. 在確認對話方塊中，輸入規則的名稱，然後選擇刪除。

在 SiteWise Monitor 中視覺化和共用風力發電廠資料

本教學課程說明如何使用 透過稱為入口網站的受管 Web 應用程式 AWS IoT SiteWise Monitor 視覺化和共用工業資料。每個入口網站都包含 專案，讓您靈活地選擇每個專案中可存取的資料。然後，指定組織中可以存取每個入口網站的人員。您的使用者使用 AWS IAM Identity Center 帳戶登入入口網站，因此您可以使用現有的身分存放區或 管理的存放區 AWS。

您和具有足夠許可的使用者都能在各專案中建立儀表板，來以有意義的方式視覺化您的工業資料。您的使用者隨後能檢視這些儀表板，快速獲得資料的洞見並監控您的操作。您可以為公司中的所有使用者設定各專案的管理或唯讀許可。如需詳細資訊，請參閱[使用 監控資料 AWS IoT SiteWise Monitor](#)。

在整個教學課程中，您會增強 AWS IoT SiteWise 示範，為風力發電廠提供範例資料集。您可以在 SiteWise Monitor 中設定入口網站、建立專案和儀表板，以視覺化風力發電廠資料。本教學課程也涵蓋建立其他使用者，以及指派許可來擁有或檢視專案及其相關聯的儀表板。

Note

當您使用 SiteWise Monitor 時，每個登入入口網站的使用者（每月）都需要付費。在本教學中，您會建立三個使用者，但您只需要以一個使用者登入。完成本教學後，您需要支付一位使用者的費用。如需詳細資訊，請參閱[AWS IoT SiteWise 定價](#)。

主題

- [先決條件](#)

- [步驟 1：在 SiteWise Monitor 中建立入口網站](#)
- [步驟 2：登入入口網站](#)
- [步驟 3：建立風力發電廠專案](#)
- [步驟 4：建立儀表板以視覺化風力發電廠資料](#)
- [步驟 5：探索入口網站](#)
- [步驟 6：在教學課程後清除資源](#)

先決條件

為了完成本教學，您需要以下項目：

- AWS 帳戶。如果您沒有帳戶，請參閱 [設定 AWS 帳戶](#)。
- 執行 Windows、Linux、或 macOSUnix 以存取 的開發電腦 AWS Management Console。如需詳細資訊，請參閱 [什麼是 AWS Management Console？](#)。
- 具有管理員許可的 AWS Identity and Access Management (IAM) 使用者。
- 執行中的 AWS IoT SiteWise 風力發電廠示範。當您設定示範時，它會在 中定義模型和資產 AWS IoT SiteWise，並將資料串流至它們，以代表風力發電廠。如需詳細資訊，請參閱 [使用 AWS IoT SiteWise 示範](#)。
- 如果您在帳戶中啟用 IAM Identity Center，請登入您的 AWS Organizations 管理帳戶。如需詳細資訊，請參閱 [AWS Organizations 術語與概念](#)。如果您尚未啟用 IAM Identity Center，您會在此教學課程中啟用它，並將您的帳戶設定為管理帳戶。

如果您無法登入您的 AWS Organizations 管理帳戶，只要您組織中有 IAM Identity Center 使用者，就可以完成部分教學課程。在此情況下，您可以建立入口網站和儀表板，但無法建立新的 IAM Identity Center 使用者以指派給專案。

步驟 1：在 SiteWise Monitor 中建立入口網站

在本程序中，您會在 AWS IoT SiteWise Monitor 中建立入口網站。每個入口網站都是受管 Web 應用程式，您和您的使用者可以使用 AWS IAM Identity Center 帳戶登入。透過 IAM Identity Center，您可以使用您公司的現有身分存放區，或建立一個由 管理的身分存放區 AWS。您公司的員工可以登入，而無需建立單獨的 AWS 帳戶。

建立入口網站

1. 登入 [AWS IoT SiteWise 主控台](#)。

2. 檢閱 AWS IoT SiteWise 支援的[AWS IoT SiteWise 端點和配額](#)，並視需要切換區域。您必須在相同的區域中執行 AWS IoT SiteWise 示範。
3. 在左側導覽窗格中選擇 Portals (入口網站)。
4. 選擇 Create portal (建立入口網站)。
5. 如果您已啟用 IAM Identity Center，請跳至步驟 6。否則，請完成下列步驟以啟用 IAM Identity Center：
 - a. 在啟用 AWS IAM Identity Center (SSO) 頁面上，輸入您的電子郵件地址、名字和姓氏，以建立 IAM Identity Center 使用者，讓您自己成為入口網站管理員。使用您可以存取的電子郵件地址，以便您可以接收電子郵件，為新的 IAM Identity Center 使用者設定密碼。

入口網站管理員會在入口網站中建立專案，以及將使用者指派到專案。您可以稍後再建立更多使用者。

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Enable SSO

Step 2
Portal configuration

Step 3
Invite administrators

Step 4
Assign users

Enable AWS Single Sign-On (SSO)

AWS IoT SiteWise Monitor requires SSO to create a portal and invite users. Create your first user below to enable AWS Single-Sign On. Later in this process, you'll have the opportunity to create other users by using the AWS SSO console. [Learn more](#)

Create a user

Email address
john.doe@example.com

First name
John

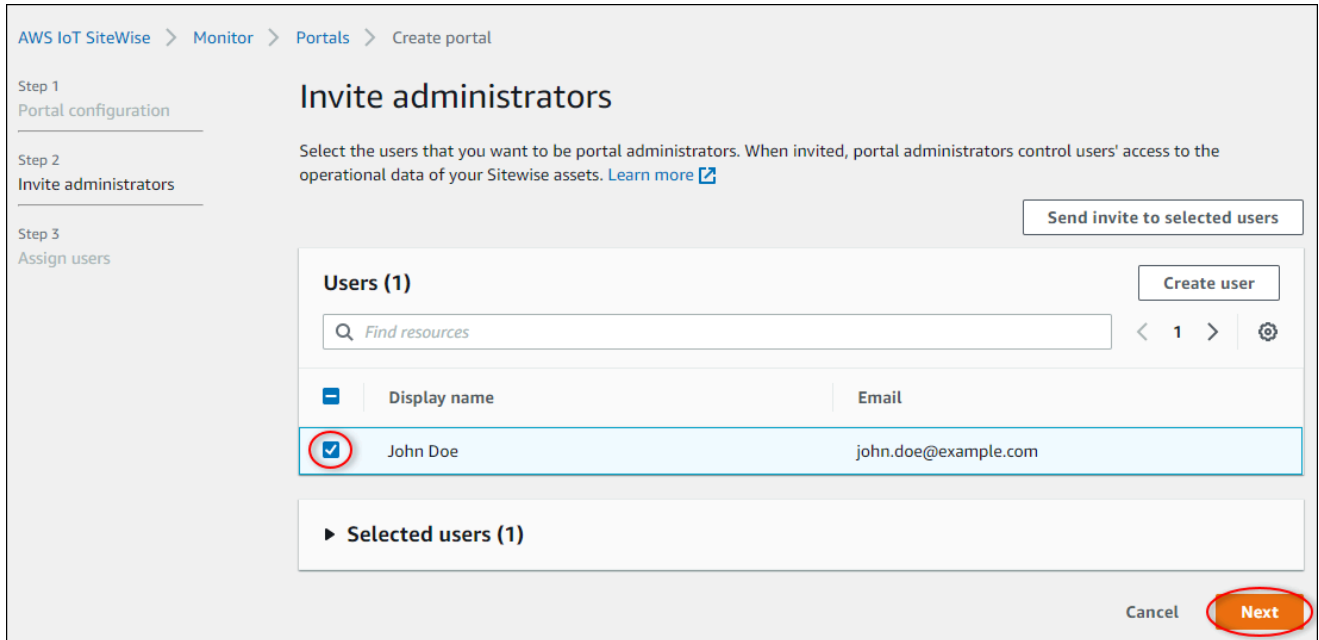
Last name
Doe

Upon creation this application will enable AWS Organizations and Single Sign-On. [Learn more](#)

Cancel **Create user**

- b. 選擇 Create user (建立使用者)。
6. 在 Portal configuration (入口網站組態) 頁面上，完成以下步驟：
 - a. 輸入入口網站的名稱，例如 **WindFarmPortal**。
 - b. (選擇性) 輸入入口網站的描述。如果您有多個入口網站，請使用有意義的描述來追蹤每個入口網站包含的內容。
 - c. (選用) 上傳要在入口網站中顯示的映像。

- d. 輸入入口網站使用者在入口網站發生問題時可以聯絡的電子郵件地址，並需要您公司的 AWS 管理員協助來解決該問題。
 - e. 選擇 Create portal (建立入口網站)。
7. 在邀請管理員頁面上，您可以將 IAM Identity Center 使用者指派為入口網站的管理員。入口網站管理員會在入口網站內管理許可和專案。在此頁面上，執行下列動作：
 - a. 選取要成為入口網站管理員的使用者。如果您在本教學課程稍早啟用了 IAM Identity Center，請選取您建立的使用者。



- b. (選擇性) 選擇 Send invite to selected users (將邀請傳送給選取的使用者)。您的電子郵件用戶端隨即開啟，並在訊息內文中顯示邀請。您可以先自訂電子郵件，然後再將其傳送給入口網站管理員。您也可以稍後將電子郵件傳送給入口網站管理員。如果您是第一次嘗試 SiteWise Monitor，而且將是入口網站管理員，則不需要自行傳送電子郵件。
 - c. 選擇 Next (下一步)。
8. 在指派使用者頁面上，您可以將 IAM Identity Center 使用者指派給入口網站。入口網站管理員稍後可以將這些使用者指派為專案擁有者或檢視器。專案擁有者可以在專案中建立儀表板。專案檢視者可以唯讀存取其指派的專案。在此頁面上，您可以建立要新增至入口網站的 IAM Identity Center 使用者。

Note

如果您未登入 AWS Organizations 管理帳戶，則無法建立 IAM Identity Center 使用者。選擇指派使用者以建立沒有入口網站使用者的入口網站，然後略過此步驟。

在此頁面上，執行下列動作：

- a. 完成下列步驟兩次，以建立兩個 IAM Identity Center 使用者：
 - i. 選擇建立使用者以開啟對話方塊，讓您輸入新使用者的詳細資訊。
 - ii. 輸入新使用者的電子郵件地址、名字和姓氏。IAM Identity Center 會傳送電子郵件給使用者，讓他們設定密碼。如果您想要以這些使用者身分登入入口網站，請選擇您可以存取的電子郵件地址。每個電子郵件地址必須是唯一的。您的使用者使用其電子郵件地址作為使用者名稱登入入口網站。

Create user [X]

Create a new AWS user. You can assign this user access to AWS applications and services

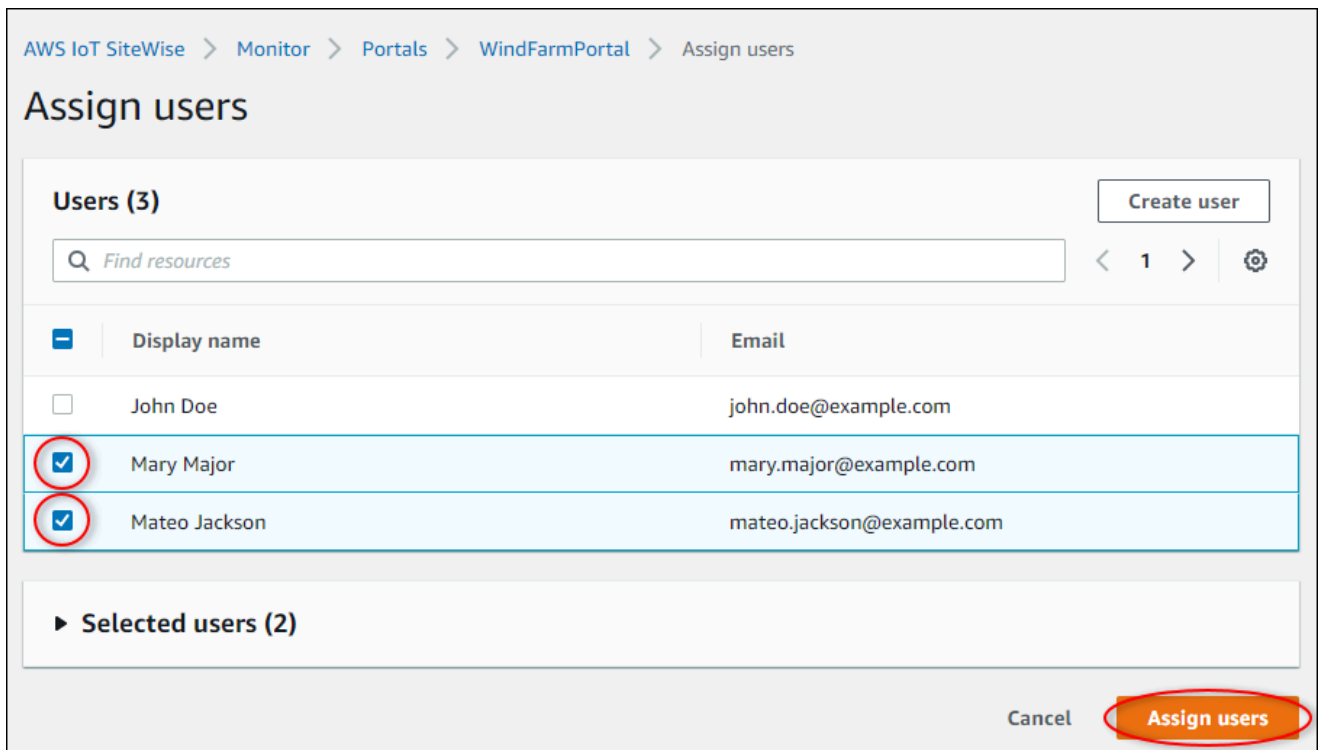
Email address
mary.major@example.com

First name
Mary

Last name
Major

Cancel Create user

- iii. 選擇 Create user (建立使用者)。
- b. 選取您在上一個步驟中建立的兩個 IAM Identity Center 使用者。



c. 選擇指派使用者，將這些使用者新增至入口網站。

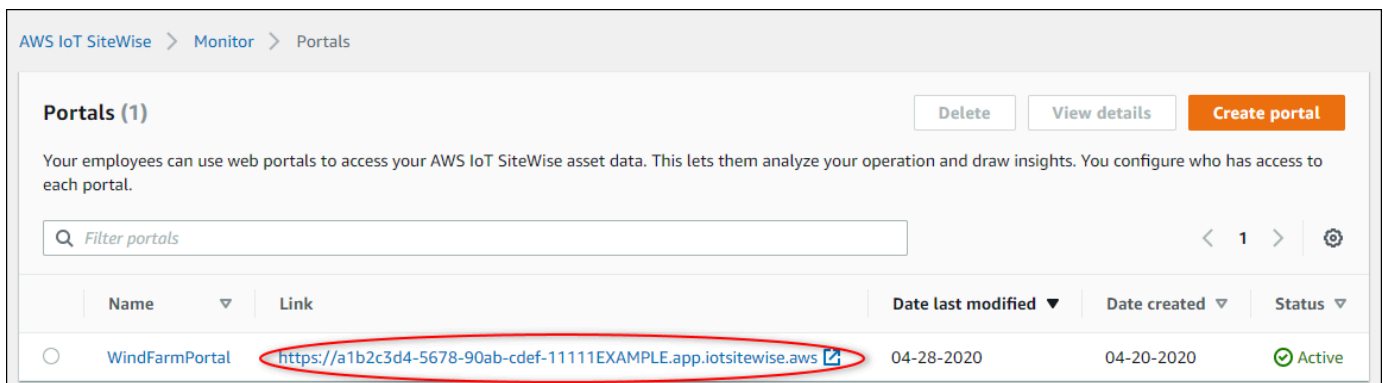
入口網站頁面隨即開啟，並列出您的新入口網站。

步驟 2：登入入口網站

在此程序中，您會使用新增至入口網站 AWS IAM Identity Center 的使用者登入新的入口網站。

登入入口網站

1. 在 Portals (入口網站) 頁面上，選擇新入口網站的 Link (連結) 來在新標籤中開啟入口網站。



2. 如果您先前已在教學課程中建立第一個 IAM Identity Center 使用者，請使用下列步驟為您的使用者建立密碼：
 - a. 查看電子郵件，找到主旨行 Invitation to join AWS IAM Identity Center。
 - b. 開啟邀請電子郵件，並選擇 Accept invitation。
 - c. 在新視窗中，為您的 IAM Identity Center 使用者設定密碼。

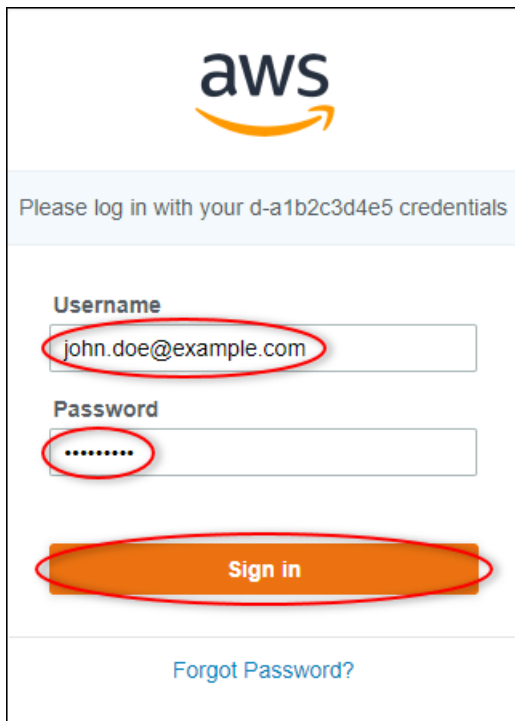
如果您想要稍後以您先前建立的第二個和第三個 IAM Identity Center 使用者身分登入入口網站，您也可以完成這些步驟來設定這些使用者的密碼。

Note

如果您沒有收到電子郵件，您可以在 IAM Identity Center 主控台中為您的使用者產生密碼。如需詳細資訊，請參閱 [《使用者指南》中的重設最終使用者的 IAM Identity Center 使用者密碼](#)。AWS IAM Identity Center

3. 輸入您的 IAM Identity Center Username 和 Password。如果您先前在本教學課程中建立了 IAM Identity Center 使用者，您的 Username 是您建立的入口網站管理員使用者的電子郵件地址。

所有入口網站使用者，包括入口網站管理員，都必須使用其 IAM Identity Center 使用者憑證登入。這些登入資料通常與您登入 AWS Management Console 所用的登入資料不同。



4. 選擇 Sign in。

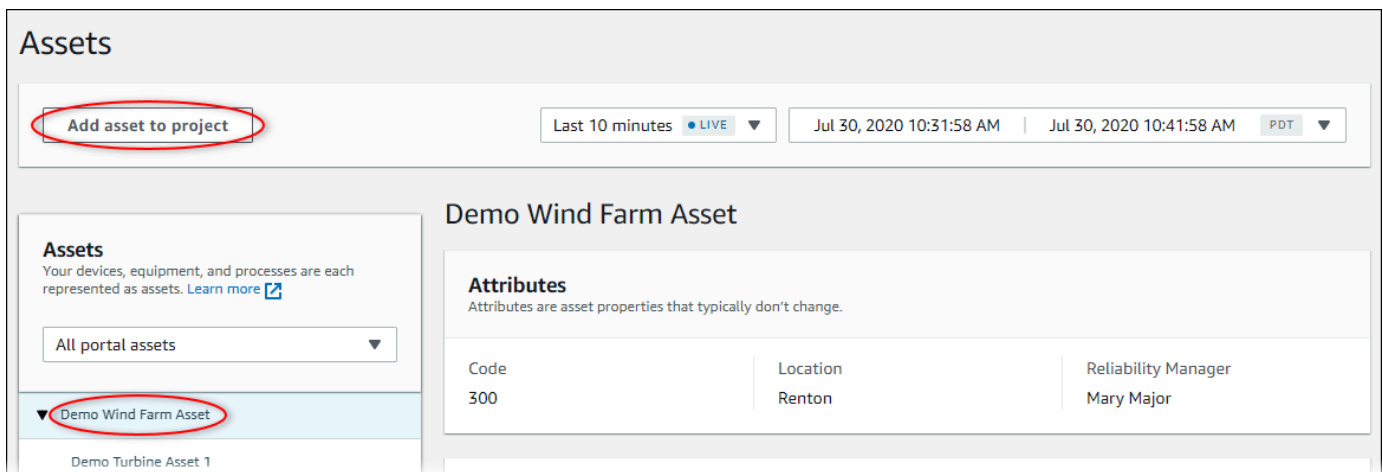
入口網站隨即開啟。

步驟 3：建立風力發電廠專案

在本程序中，您會在入口網站中建立專案。專案是定義一組許可、資產和儀表板的資源，您可以設定這些資源來視覺化該專案中的資產資料。您可以透過專案來定義誰能存取操作的哪個子集，以及這些子集的資料視覺化方式。您可以將入口網站使用者指派為每個專案的擁有者或檢視器。專案擁有者可以建立儀表板來視覺化資料，並與其他使用者共用專案。專案檢視者可以檢視儀表板，但無法編輯。如需 SiteWise Monitor 中角色的詳細資訊，請參閱 [SiteWise Monitor 角色](#)。

建立風力發電廠專案

1. 在入口網站的左側導覽窗格中，選擇資產索引標籤。在資產頁面上，您可以探索入口網站中可用的所有資產，並將資產新增至專案。
2. 在資產瀏覽器中，選擇 Demo Wind Farm Asset。選擇資產時，您可以探索該資產的即時和歷史資料。您也可以按下 Shift 來選取多個資產，來並排比較其資料。
3. 選擇左上角的新增資產至專案。專案會包含您入口網站使用者能檢視來探索資料的儀表板。每個專案都可以存取 中的資產子集 AWS IoT SiteWise。當您將資產新增到專案時，具有該專案存取權的所有使用者也都能存取該資產和其子系的資料。



4. 在將資產新增至專案對話方塊中，選擇建立新專案，然後選擇下一步。

Add asset to project [X]

Selected node and all of its descendant assets will be added to the project.

► Demo Wind Farm Asset

Select project or create new project

☒ Create new project

☐ Select existing project

Cancel Next

5. 在建立新專案對話方塊中，輸入專案的專案名稱和專案描述，然後選擇將資產新增至專案。

Create new project [X]

Project name

Wind Farm 1

The project name can have up to 256 characters.

Project description

A project that contains dashboards for wind farm #1.

The project description can have up to 2048 characters.

Cancel Previous Add asset to project

新專案的頁面隨即開啟。

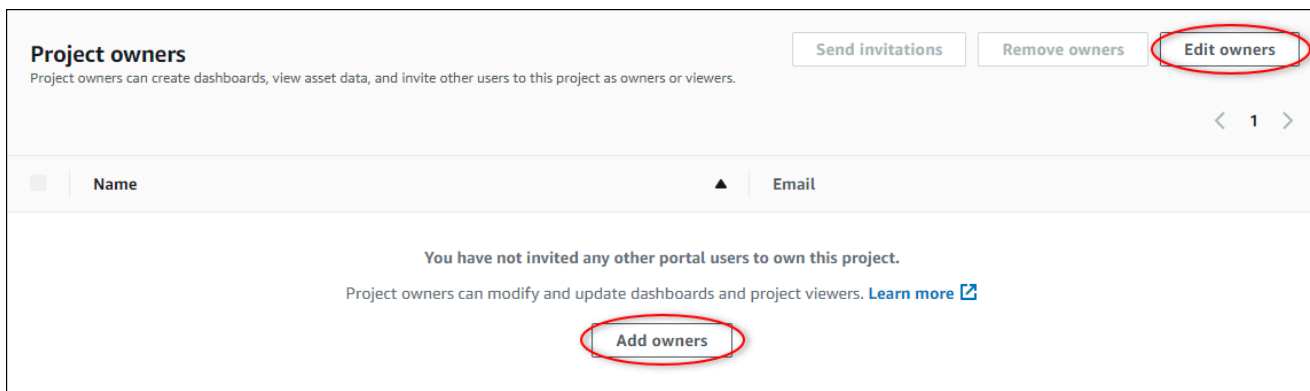
6. 在專案的頁面上，您可以將入口網站使用者新增為此專案的擁有者或檢視器。

Note

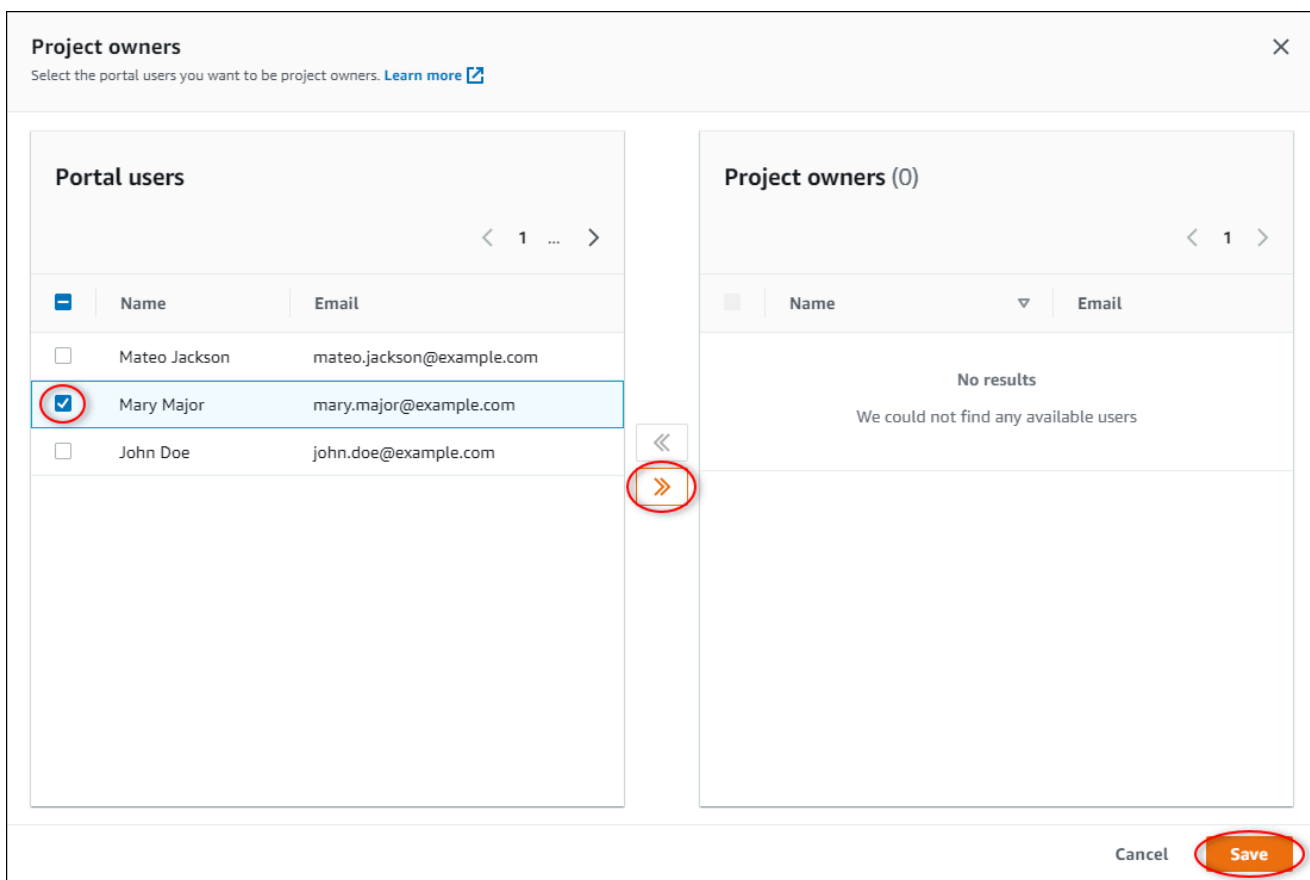
如果您未登入您的 AWS Organizations 管理帳戶，您可能沒有入口網站使用者可指派給此專案，因此您可以略過此步驟。

在此頁面上，執行下列動作：

- a. 在專案擁有者下，選擇新增擁有者或編輯使用者。



- b. 選擇要新增為專案擁有者的使用者 (例如 Mary Major)，然後選擇 >> 圖示。



- c. 選擇 Save (儲存)。

您的 IAM Identity Center 使用者可以 Mary Major 登入此入口網站，以編輯此專案中的儀表板，並與此入口網站中的其他使用者共用此專案。

- d. 在專案檢視器下，選擇新增檢視器或編輯使用者。
- e. 選擇要新增為專案檢視器的使用者（例如 Mateo Jackson），然後選擇 >> 圖示。
- f. 選擇 Save (儲存)。

您的 IAM Identity Center 使用者可以Mateo Jackson登入此入口網站，以檢視但無法編輯風力發電廠專案中的儀表板。

步驟 4：建立儀表板以視覺化風力發電廠資料

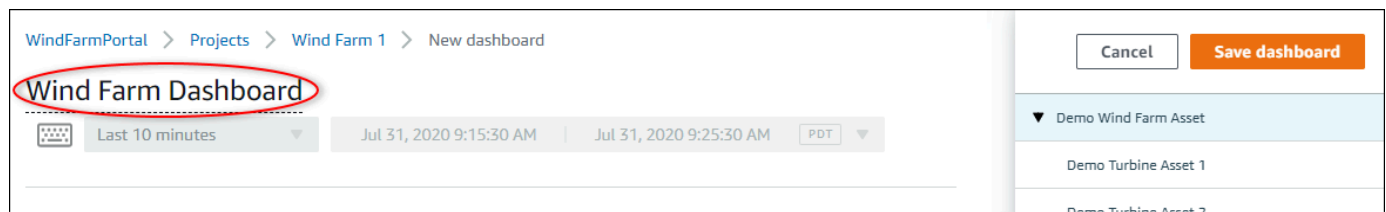
在此程序中，您會建立儀表板來將示範風力發電廠資料視覺化。儀表板包含專案資產資料的可自訂視覺效果。每個視覺效果可以有不同的類型，例如折線圖、長條圖或關鍵效能指標 (KPI) 顯示。您可以選擇最適合您資料的視覺效果類型。專案擁有者可以編輯儀表板，而專案檢視器只能檢視儀表板以取得洞見。

建立具有視覺效果的儀表板

1. 在新專案的頁面上，選擇建立儀表板以建立儀表板並開啟其編輯頁面。

在儀表板的編輯頁面中，您可以從資產階層拖曳資產屬性到儀表板，來建立視覺效果。您接著可以編輯每個視覺效果的標題、圖例標題、類型、大小和在儀表板中的位置。

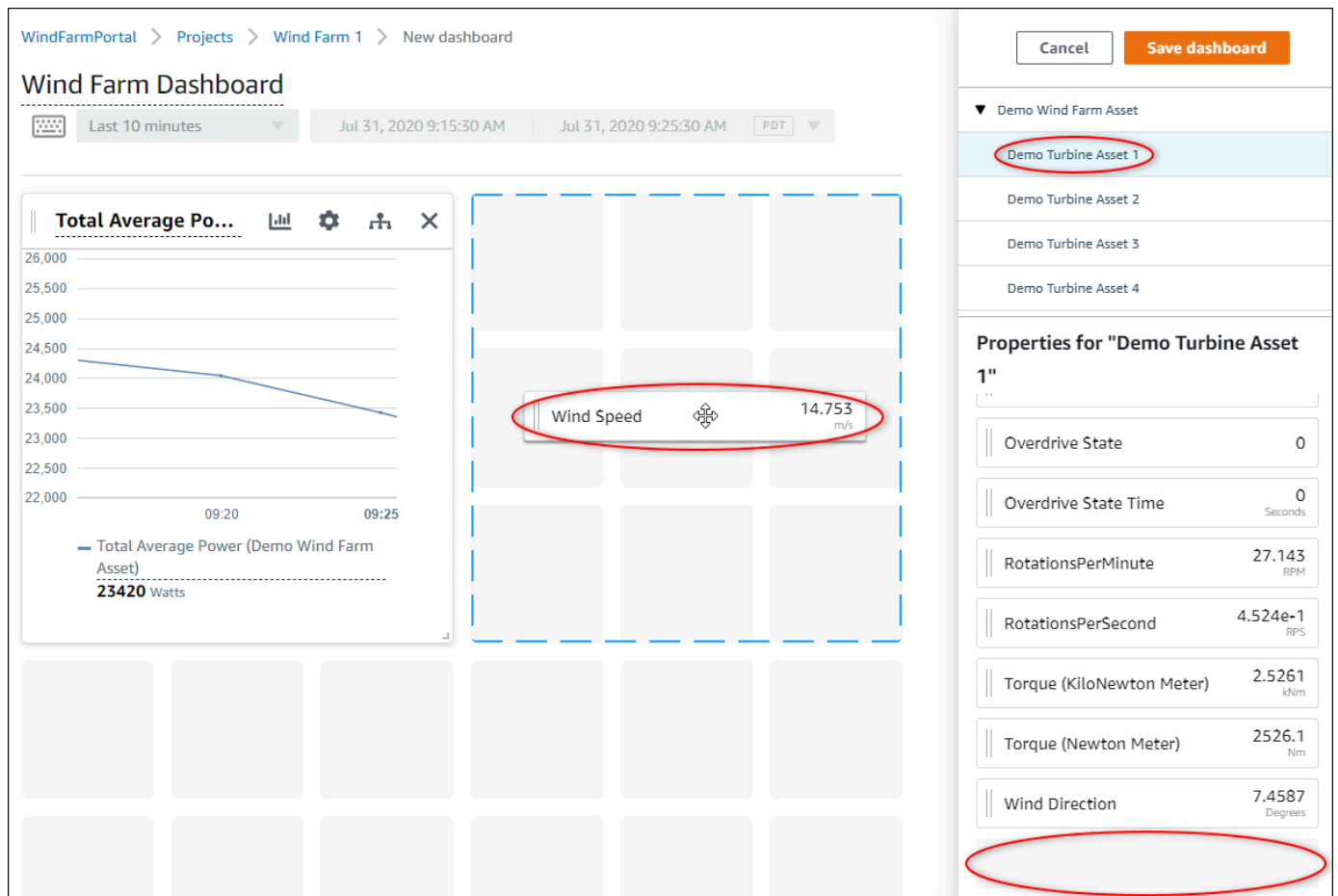
2. 輸入您的儀表板名稱。



3. Total Average Power 從 拖曳Demo Wind Farm Asset至儀表板以建立視覺化效果。

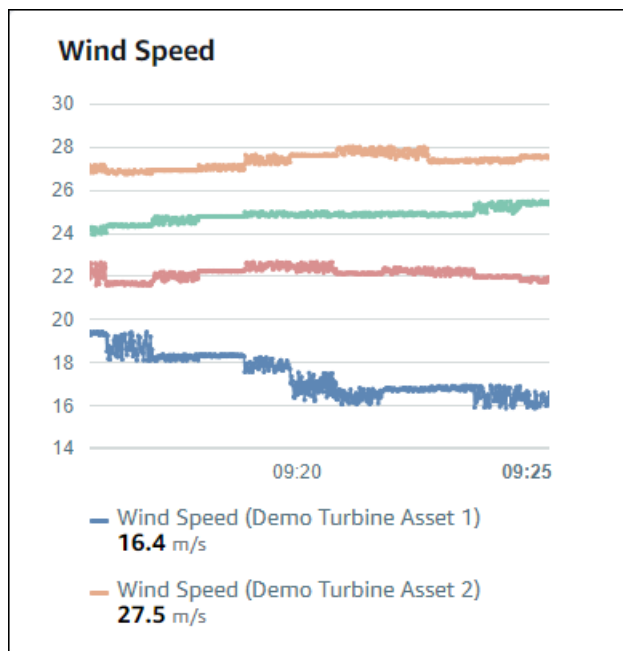
The screenshot shows the 'Wind Farm Dashboard' in the AWS IoT SiteWise console. The breadcrumb trail is 'WindFarmPortal > Projects > Wind Farm 1 > New dashboard'. The dashboard title is 'Wind Farm Dashboard'. Below the title, there are filters for 'Last 10 minutes', a date range from 'Jul 31, 2020 9:15:30 AM' to 'Jul 31, 2020 9:25:30 AM', and a time zone dropdown set to 'PDT'. The main area is a grid of 12 placeholder widgets. A red dashed box highlights a 3x2 area in the top-left, and a red oval highlights a widget titled 'Total Average Power' with a value of '24038 Watts'. On the right, a sidebar shows a list of assets under 'Demo Wind Farm Asset': 'Demo Turbine Asset 1', 'Demo Turbine Asset 2', 'Demo Turbine Asset 3', and 'Demo Turbine Asset 4'. Below this, the 'Properties for "Demo Wind Farm Asset"' are shown, including 'Code' (300) and 'Total Overdrive State Time' (0 seconds). A red oval highlights the empty space between the 'Code' and 'Total Overdrive State Time' properties.

4. 選擇 Demo Turbine Asset 1以顯示該資產的屬性，然後拖曳Wind Speed至儀表板以建立風速的視覺化效果。

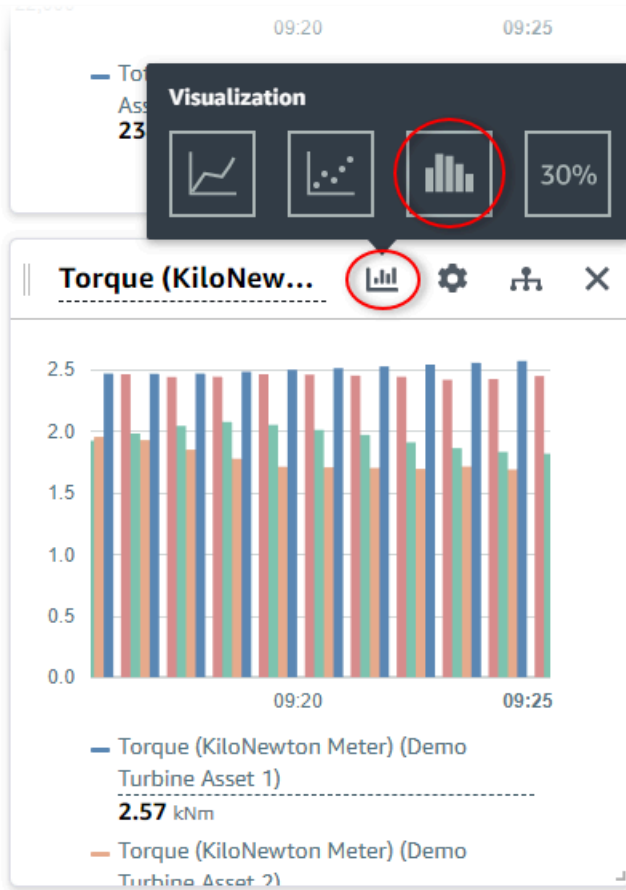


5. 將 Wind Speed 新增至每個 Demo Turbine Asset 2、3 和 的新風速視覺化 4 (依該順序)。

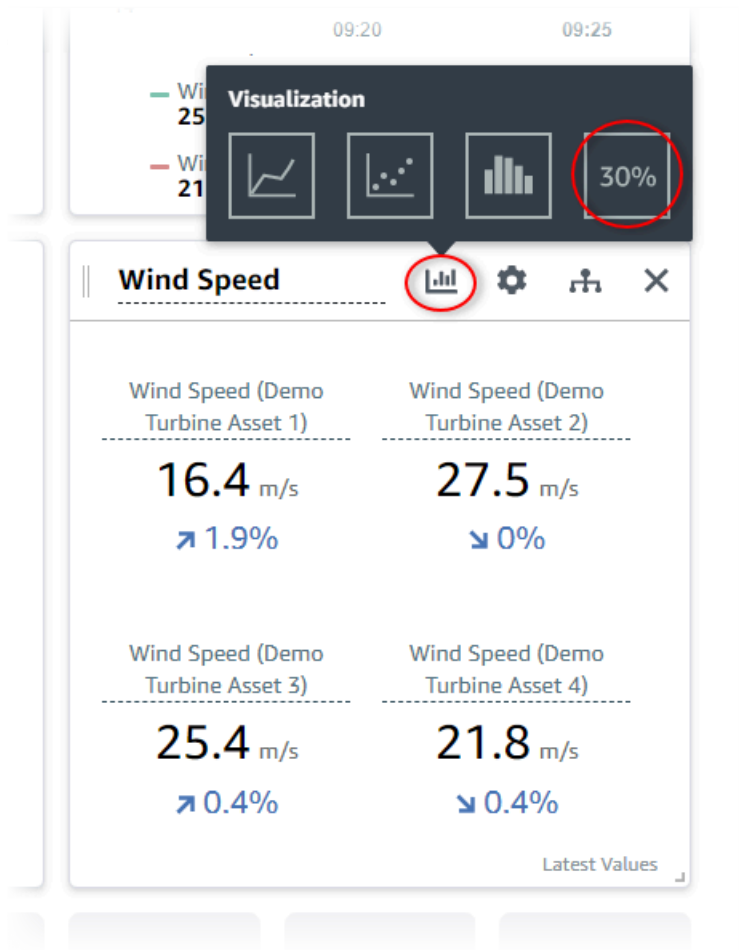
Wind Speed 視覺效果看起來應與下列螢幕擷取畫面類似。



6. 針對風力渦輪機的Torque (KiloNewton Meter)屬性重複步驟 4 和 5，以建立風力渦輪機扭力的視覺化效果。
7. 選擇 Torque (KiloNewton Meter) 視覺效果的視覺效果類型圖示，然後選擇長條圖圖示。

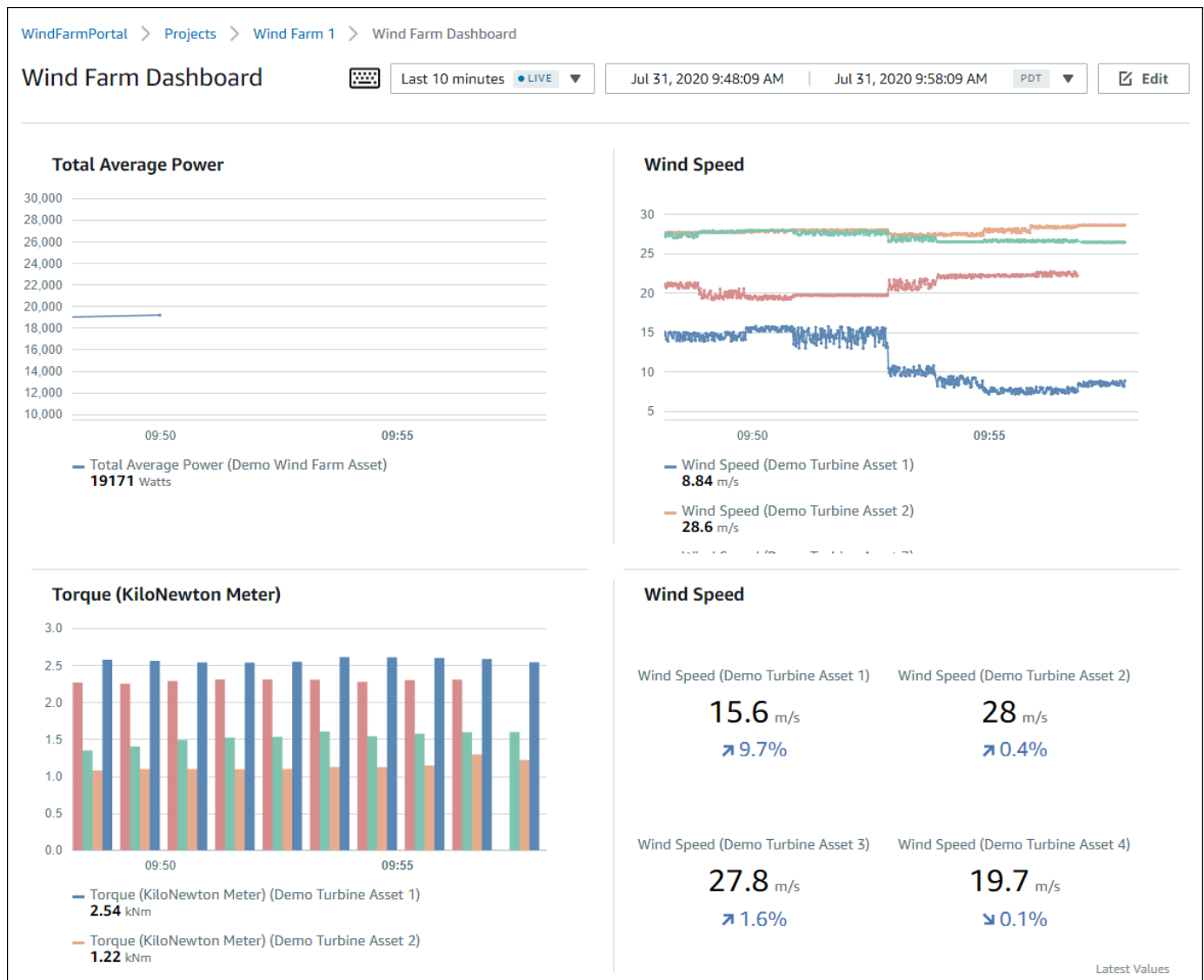


8. 針對風力發電機的Wind Direction屬性重複步驟 4 和 5，以建立風向的視覺化。
9. 選擇 Wind Direction 視覺效果的視覺效果類型圖示，然後選擇 KPI 圖表圖示 (30%)。



10. (選用) 您可以視需要對各個視覺效果的標題、圖例標題、類型、大小和位置進行其他變更。
11. 選擇右上角的儲存儀表板，以儲存儀表板。

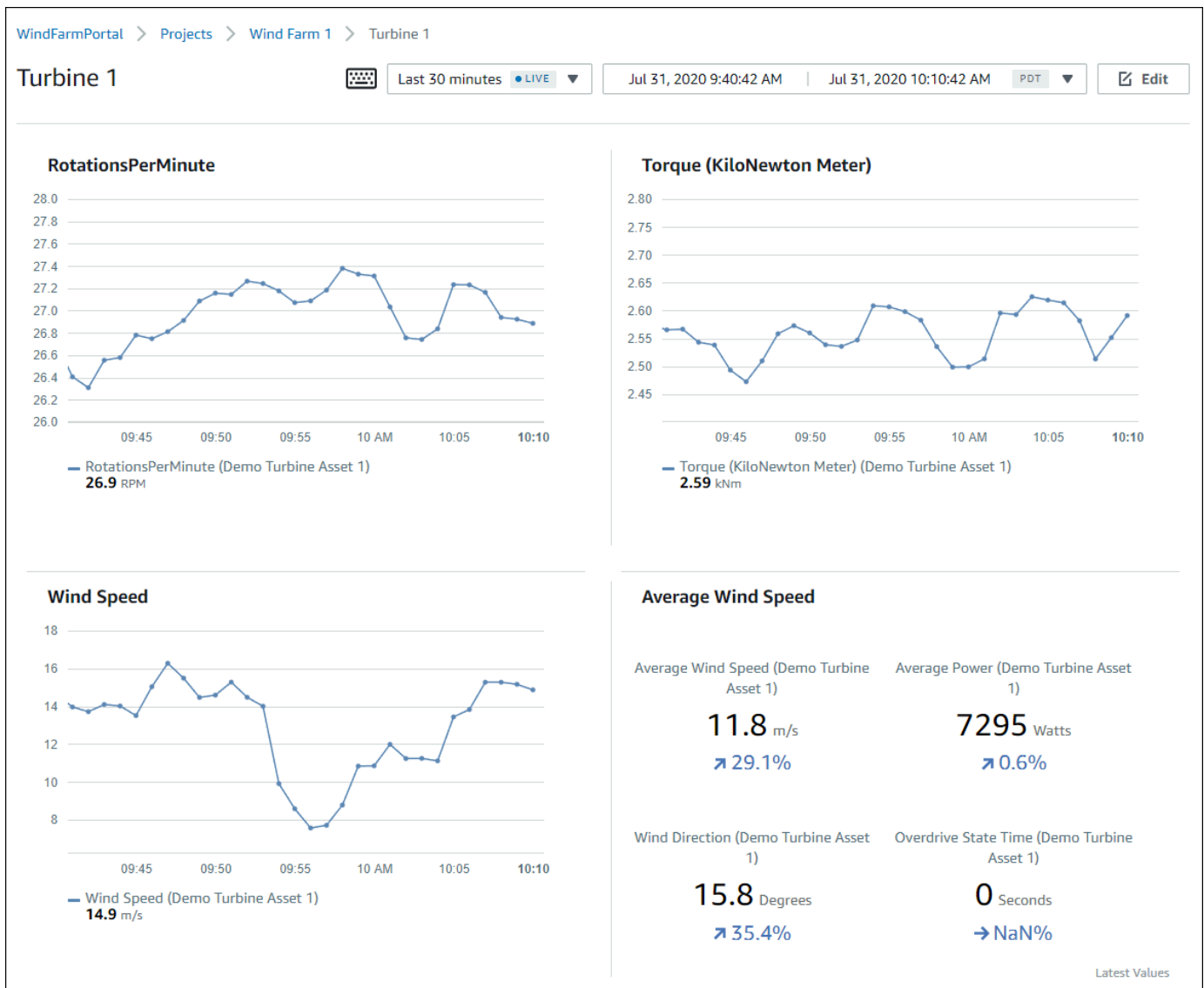
您儀表板看起來應與下列螢幕擷取畫面類似。



12. (選用) 為各個風力渦輪機資產建立其他儀表板。

建議的最佳實務是為每個資產建立儀表板，以便專案檢視者能調查每個個別資產的任何問題。您最多只能為每個視覺效果新增 5 個資產，因此，您必須在許多情況下為階層資產建立多個儀表板。

示範風力渦輪機的儀表板看起來應與下列螢幕擷取畫面類似。



13. (選用) 變更時間軸或選取視覺效果上的資料點，來探索儀表板中的資料。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[檢視儀表板](#)。

步驟 5：探索入口網站

在此程序中，您可以用比入口網站管理員更少的許可來探索 AWS IoT SiteWise 入口網站。

探索入口網站並完成教學課程

- (選用) 如果您以擁有者或瀏覽者身分將其他使用者新增至專案，則可以這些使用者身分登入入口網站。這可讓您以具有比入口網站管理員更少許可的使用者身分探索入口網站。

⚠ Important

您需要為登入入口網站的每個使用者支付費用。如需詳細資訊，請參閱 [AWS IoT SiteWise 定價](#)。

若要以其他使用者身分探索入口網站，請執行下列動作：

- a. 選擇入口網站左下角的登出，以結束 Web 應用程式。
- b. 選擇 IAM Identity Center 應用程式入口網站右上角的登出，以登出您的 IAM Identity Center 使用者。
- c. 以您指派為專案擁有者或專案檢視器的 IAM Identity Center 使用者身分登入入口網站。如需詳細資訊，請參閱 [步驟 2：登入入口網站](#)。

您已完成教學課程。當您在 SiteWise Monitor 中完成探索示範風力發電廠時，請遵循下一個程序來清理資源。

步驟 6：在教學課程後清除資源

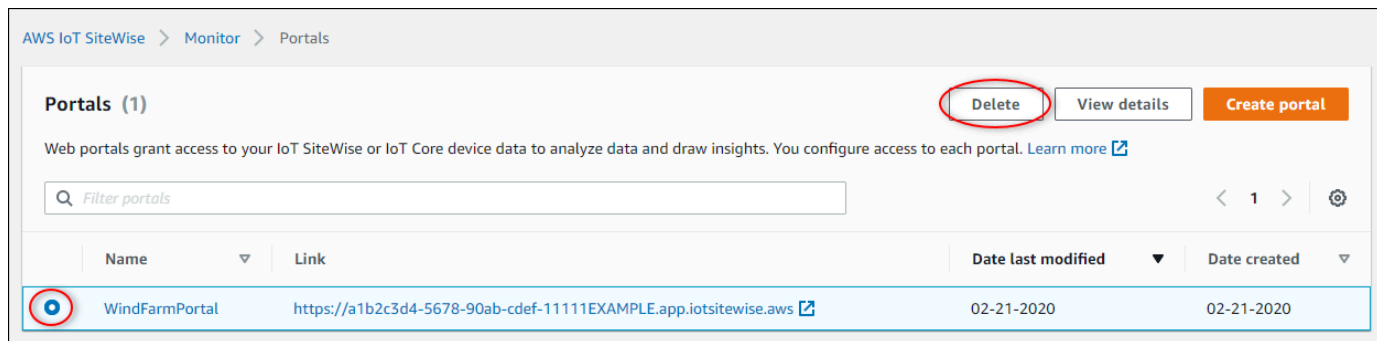
完成教學後，您便可清除資源。如果使用者未登入您的入口網站，就不需要支付 AWS IoT SiteWise 的費用，但您可以刪除入口網站和 AWS IAM Identity Center 目錄使用者。您的示範風力發電廠資產會在經過建立示範時選擇的持續時間後刪除，此外也能手動刪除。如需詳細資訊，請參閱 [刪除 AWS IoT SiteWise 示範](#)。

使用下列程序刪除您的入口網站和 IAM Identity Center 使用者。

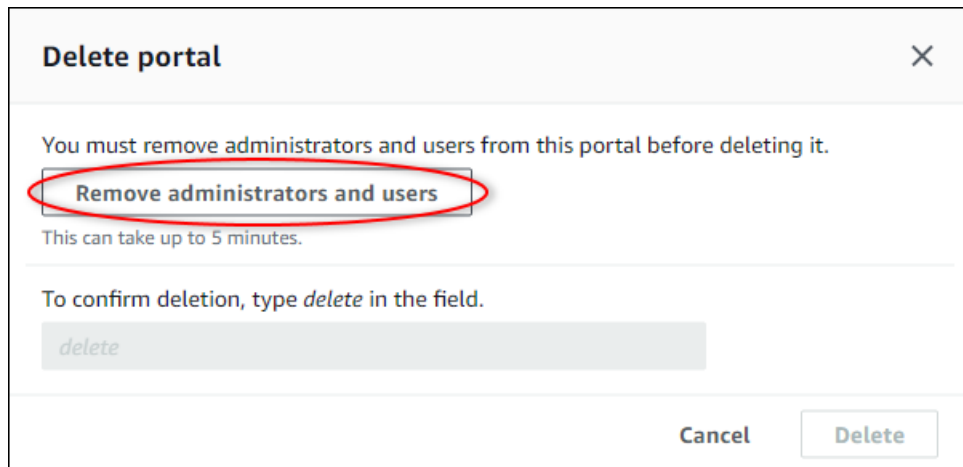
刪除入口網站

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中選擇 Portals (入口網站)。
3. 依序選擇您的入口網站 WindFarmPortal 和 Delete (刪除)。

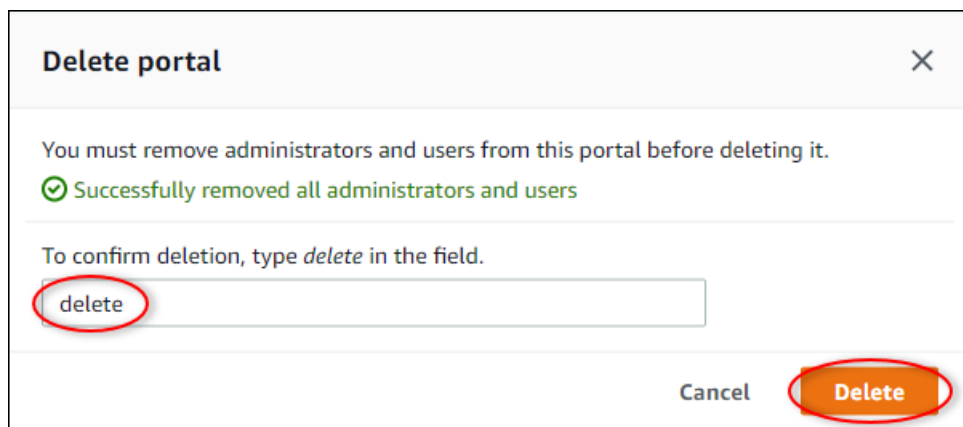
刪除入口網站或專案時，與受刪除專案相關聯的資產不會受到影響。



4. 在刪除入口網站對話方塊中，選擇移除管理員和使用者。

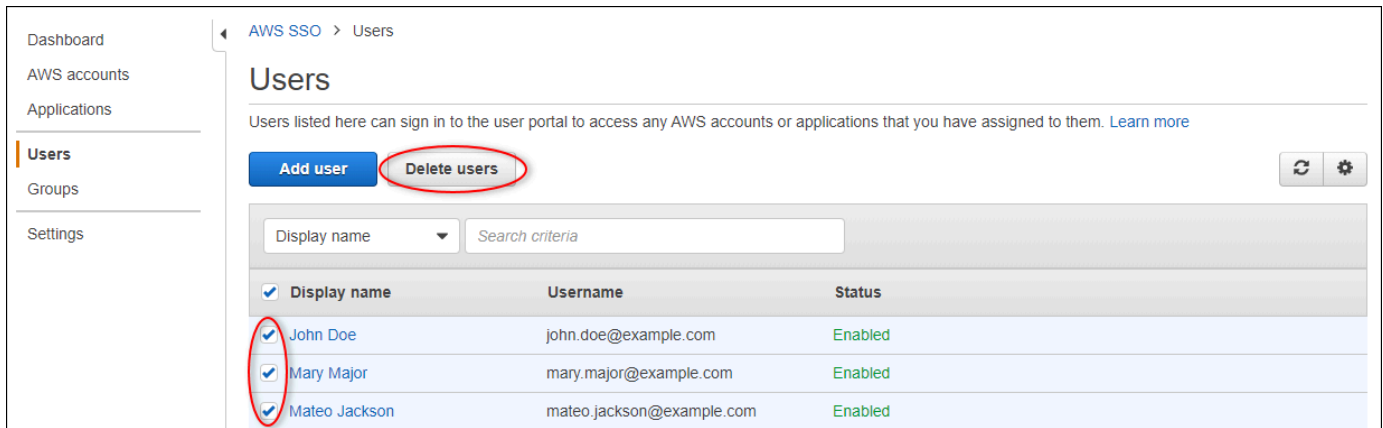


5. 輸入 **delete** 以確認刪除，然後選擇 Delete (刪除)。

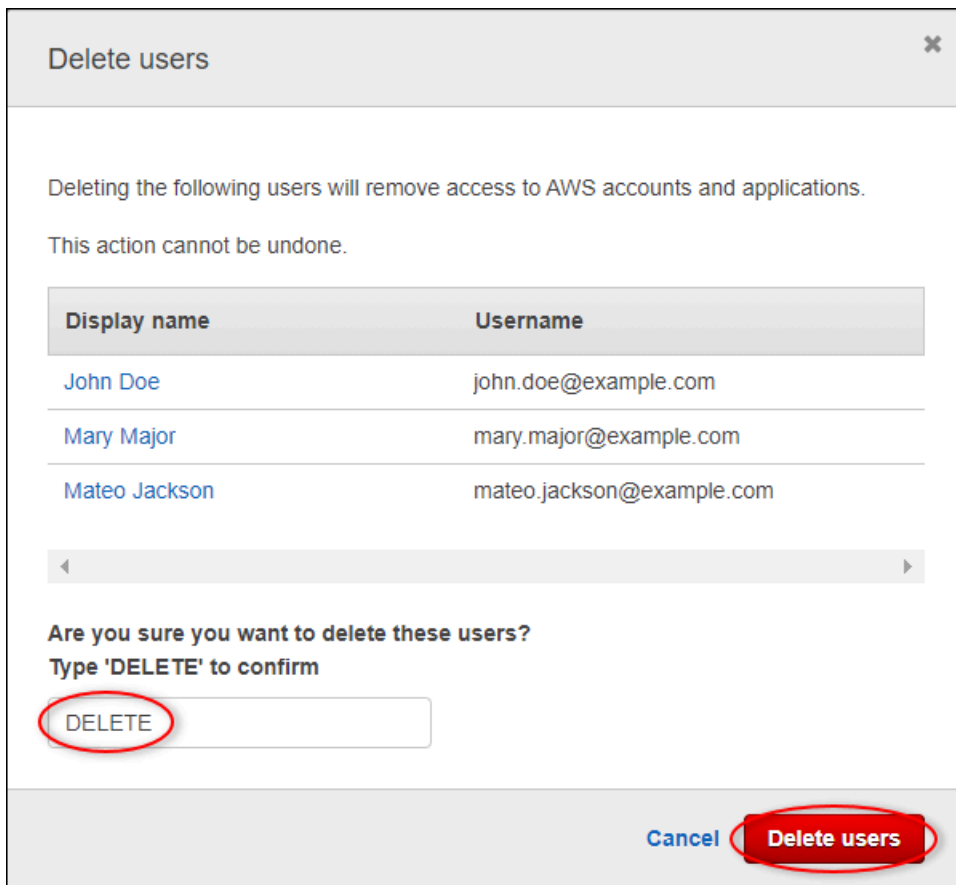


刪除 IAM Identity Center 使用者

1. 導覽至 [IAM Identity Center 主控台](#)。
2. 在左側導覽窗格中，選擇 Users (使用者)。
3. 選取每位欲刪除使用者的核取方塊，然後選擇 Delete users (刪除使用者)。



4. 在刪除使用者對話方塊中，輸入 **DELETE**，然後選擇刪除使用者。



將屬性值更新發佈至 Amazon DynamoDB

本教學課程介紹使用 [Amazon DynamoDB](#) 存放資料的便利方式，讓您更輕鬆地存取歷史資產資料，而無需重複查詢 AWS IoT SiteWise API。完成本教學課程後，您可以建立使用資產資料的自訂軟體，例如整個風力發電廠的風速和方向即時地圖。如果您想要在不實作自訂軟體解決方案的情況下監控和視覺化資料，請參閱 [使用 監控資料 AWS IoT SiteWise Monitor](#)。

在本教學課程中，您會建置在示範的基礎上，該 AWS IoT SiteWise 示範會提供風力發電廠的範例資料集。您可以從風力發電廠示範設定屬性值更新，透過 AWS IoT Core 規則將資料傳送至您建立的 DynamoDB 資料表。當您啟用屬性值更新時，AWS IoT SiteWise 會在 MQTT 訊息 AWS IoT Core 中傳送您的資料至。然後，根據這些訊息的內容，定義執行動作 AWS IoT 的核心規則，例如 DynamoDB 動作。如需詳細資訊，請參閱[與其他 AWS 服務互動](#)。

主題

- [先決條件](#)
- [步驟 1：設定 AWS IoT SiteWise 以發佈屬性值更新](#)
- [步驟 2：在 AWS IoT 核心中建立規則](#)
- [步驟 3：建立 DynamoDB 資料表](#)
- [步驟 4：設定 DynamoDB 規則動作](#)
- [步驟 5：探索 DynamoDB 中的資料](#)
- [步驟 6：在教學課程後清除資源](#)

先決條件

為了完成本教學，您需要以下項目：

- AWS 帳戶。如果您沒有帳戶，請參閱[設定 AWS 帳戶](#)。
- 執行 Windows、macOS、Linux 或 Unix 以存取 的開發電腦 AWS Management Console。如需詳細資訊，請參閱[什麼是 AWS Management Console？](#)
- 擁有管理員許可的 IAM 使用者。
- 執行中的 AWS IoT SiteWise 風力發電廠示範。當您設定示範時，它會在 中定義模型和資產 AWS IoT SiteWise，並將資料串流至它們，以代表風力發電廠。如需詳細資訊，請參閱[使用 AWS IoT SiteWise 示範](#)。

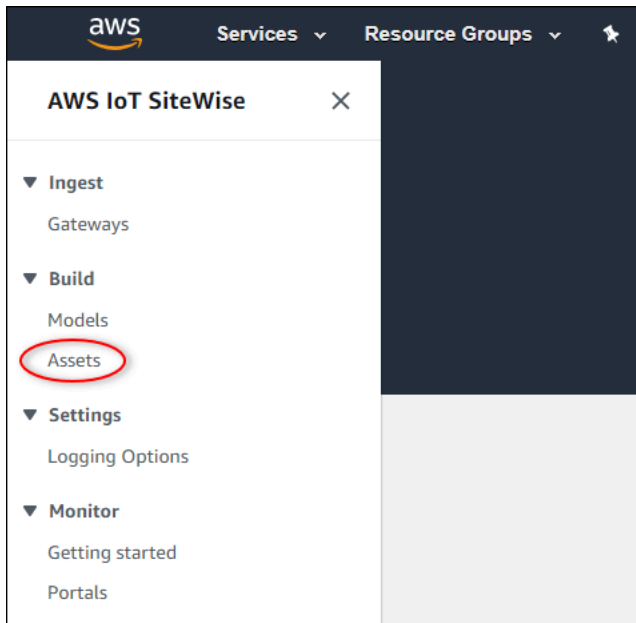
步驟 1：設定 AWS IoT SiteWise 以發佈屬性值更新

在此程序中，您會在示範渦輪機資產的 Wind Speed (風速) 屬性上啟用屬性值通知。啟用屬性值通知後，會將 MQTT 訊息中的每個值更新 AWS IoT SiteWise 發佈至 AWS IoT 核心。

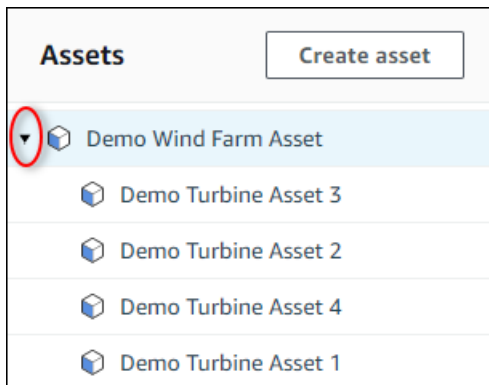
啟用資產屬性的屬性值更新通知

1. 登入 [AWS IoT SiteWise 主控台](#)。

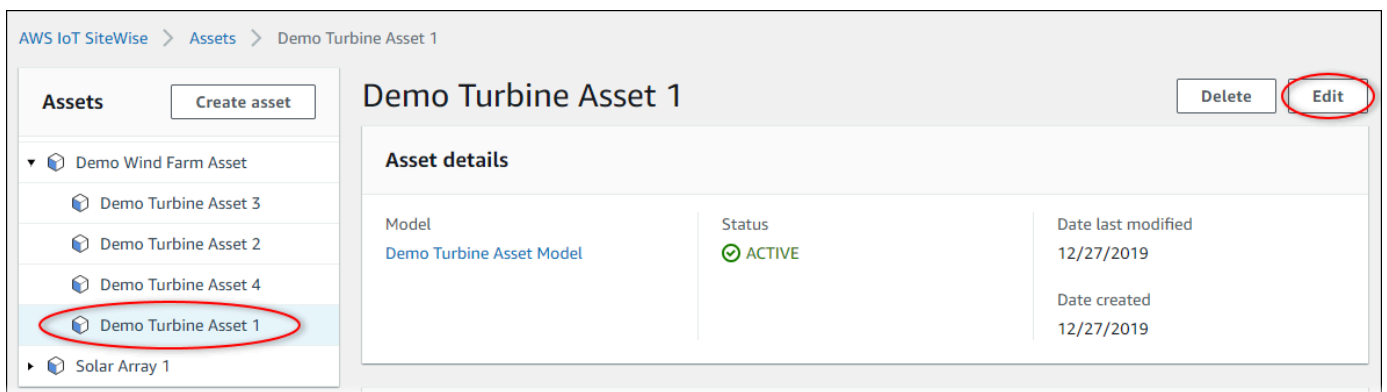
- 檢閱 AWS IoT SiteWise 支援的[AWS IoT SiteWise 端點和配額](#)，並視需要切換 AWS 區域。切換到您正在執行 AWS IoT SiteWise 示範的區域。
- 在左側導覽窗格中，選擇 Assets (資產)。



- 選擇 Demo Wind Farm Asset 旁的箭號，以展開發力發電廠資產的階層。



- 選擇一個示範渦輪機，然後選擇 Edit (編輯)。



6. 將 Wind Speed 屬性的 Notification status (通知狀態) 更新為 ENABLED (啟用)。

"Wind Speed"

Enter a property alias

Must be less than 2048 characters.

Notification status

ENABLED

Notification will be published to topic \$aws/sitewise/asset-models/d8f8f20a-4d3a-491c-a9c5-352736979bdb/assets/db36f80f-ed03-44d9-84ef-817eb30d5497/properties/ca5b9e21-f19c-4ea1-8472-0e9400fc12bf

- 選擇頁面底部的 Save asset (儲存資產)。
- 針對每個示範渦輪機資產重複步驟 5 到 7。
- 選擇示範渦輪機 (例如 Demo Turbine Asset 1 (示範渦輪機資產))。
- 選擇 Measurements (衡量值)。
- 選擇 Wind Speed 屬性旁的複製圖示，以將通知主題複製到剪貼簿。儲存通知主題，以在本教學的稍後使用。您只需要記錄一台渦輪機的通知主題。

Torque (KiloNewton Meter)	-	⊖ Disabled	-	2.128123
Wind Speed	-	✔ Enabled	\$aws/sitewise/asset-models/d8f8f...	26.49812

通知主題看起來應該會與以下範例相似。

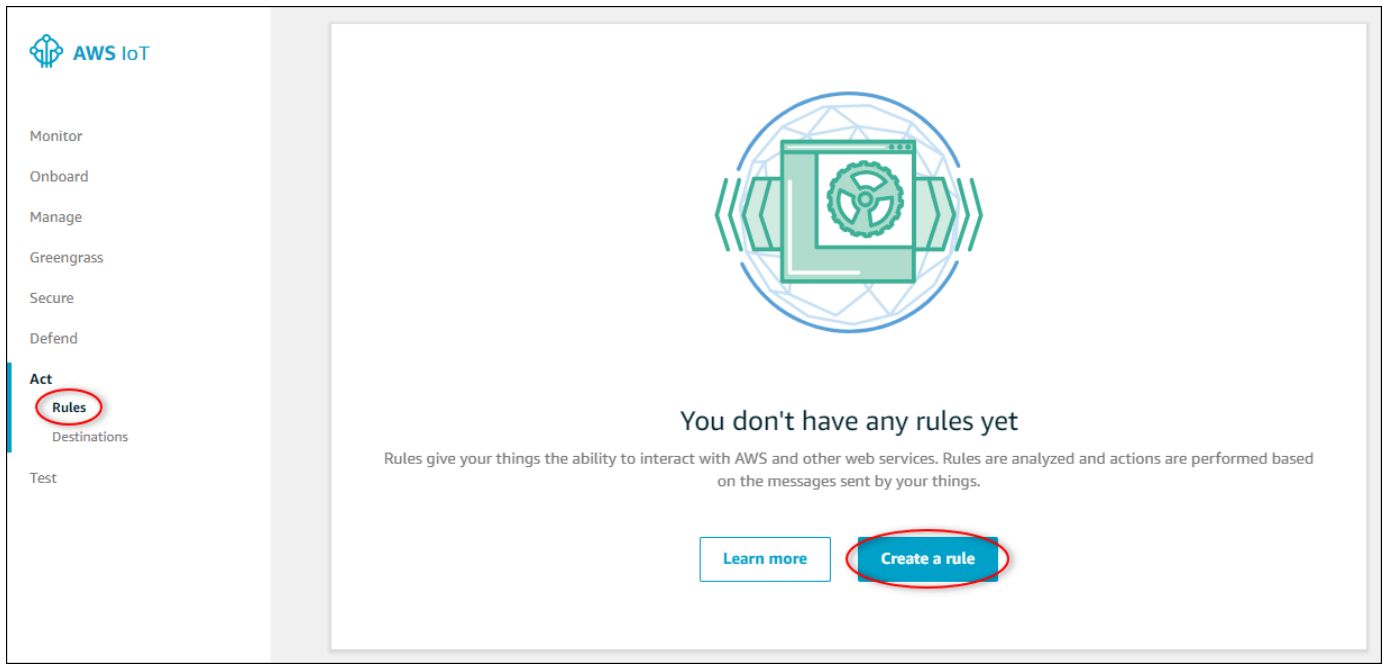
```
$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE
```

步驟 2：在 AWS IoT 核心中建立規則

在此程序中，您會在 AWS IoT Core 中建立規則，以剖析屬性值通知訊息，並將資料插入 Amazon DynamoDB 資料表。AWS IoT Core 規則會剖析 MQTT 訊息，並根據每則訊息的內容和主題執行動作。然後，您可以使用 DynamoDB 動作建立規則，將資料插入您在本教學課程中建立的 DynamoDB 資料表。

使用 DynamoDB 動作建立規則

- 導覽至 [AWS IoT 主控台](#)。如果出現 Get started (開始使用) 按鈕，請選擇此按鈕。
- 在左側導覽窗格中，選擇 Act (動作)，然後選擇 Rules (規則)。



3. 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 Create (建立)。
4. 輸入規則的名稱和描述。

The screenshot displays the 'Create a rule' form. It has a blue header with the title 'Create a rule'. Below the header, there is instructional text: 'Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function)'. The form contains two input fields. The first field, labeled 'Name', contains the text 'WindSpeedRule'. The second field, labeled 'Description', contains the text 'A DynamoDBv2 rule that records wind data from wind turbine assets in AWS IoT SiteWise.' Both input fields are circled in red.

5. 尋找您在本教學中稍早儲存的通知主題。

```
$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/  
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-  
cdef-33333EXAMPLE
```

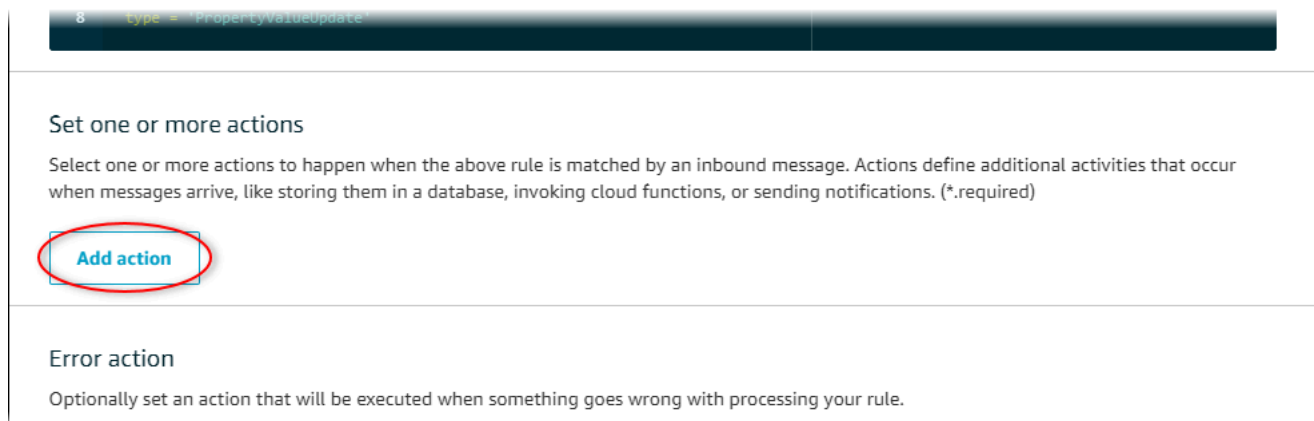
將主題中的資產 ID (之後的 IDassets/) 取代為 +。這會選取所有示範風力發電機資產的風速屬性。+ 主題篩選條件接受主題中單一層級的所有節點。您的主題應如下所示。

```
$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/+/  
properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

6. 輸入以下規則查詢陳述式。將 FROM 區段中的主題取代成您的通知主題。

```
SELECT  
  payload.assetId AS asset,  
  (SELECT VALUE (value.doubleValue) FROM payload.values) AS windspeed,  
  timestamp() AS timestamp  
FROM  
  '$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/+/  
properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'  
WHERE  
  type = 'PropertyValueUpdate'
```

7. 在 Set one or more actions (設定一或多個動作) 下，選擇 Add action (新增動作)。



8 type = 'PropertyValueUpdate'

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

Add action

Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

8. 在選取動作頁面上，選擇將訊息分割為 DynamoDB 資料表 (DynamoDBv2) 的多個資料欄。



9. 選擇頁面底部的 Configure action (設定動作)。
10. 在 Configure action (設定動作) 頁面，選擇 Create a new resource (建立新的資源)。

DynamoDB 主控台會在新索引標籤中開啟。在您完成下列程序期間，請將規則動作標籤維持在開啟狀態。

步驟 3：建立 DynamoDB 資料表

在此程序中，您會建立 Amazon DynamoDB 資料表，從規則動作接收風速資料。

建立 DynamoDB 資料表

1. 在 DynamoDB 主控台儀表板中，選擇建立資料表。
2. 輸入資料表的名稱。

Create DynamoDB table

Tutorial ?

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name*

WindSpeedData

i

Primary key*

Partition key

timestamp

Number

i

☒ Add sort key

asset

String

i

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☒ Use default settings

- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type.

i

You do not have the required role to enable Auto Scaling by default.
Please refer to [documentation](#).

+ Add tags

NEW!

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Cancel

Create

3. 針對 Primary key (主索引鍵)，執行以下作業：

- 輸入 **timestamp** 做為分割區索引鍵。
- 選擇 Number (數字) 類型。
- 選取 Add sort key (新增排序索引鍵) 核取方塊。
- 輸入 **asset** 做為排序索引鍵，並保留預設的 String (字串) 排序索引鍵類型。

4. 選擇 Create (建立)。

當 Table is being created (正在建立資料表) 通知消失時，您的資料表便已準備就緒。

5. 返回包含 Configure action (設定動作) 頁面的標籤。當您完成下列程序時，請保持 DynamoDB 索引標籤開啟。

步驟 4：設定 DynamoDB 規則動作

在此程序中，您會設定 Amazon DynamoDB 規則動作，將屬性值更新的資料插入新的 DynamoDB 資料表。

設定 DynamoDB 規則動作

1. 在設定動作頁面上，重新整理資料表名稱清單，然後選擇新的 DynamoDB 資料表。

2. 選擇建立角色以建立 IAM 角色，授予 AWS IoT 核心存取權來執行規則動作。
3. 輸入角色名稱，然後選擇 Create role (建立角色)。

4. 選擇新增動作。
5. 選擇頁面底部的 Create rule (建立規則) 來完成建立規則。

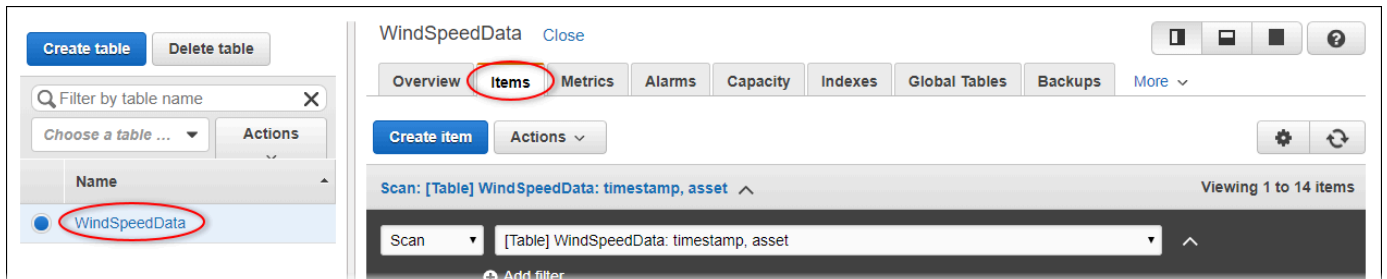
您的示範資產資料應該會開始出現在 DynamoDB 資料表中。

步驟 5：探索 DynamoDB 中的資料

在此程序中，您會在新的 Amazon DynamoDB 資料表中探索示範資產的風速資料。

在 DynamoDB 中探索資產資料

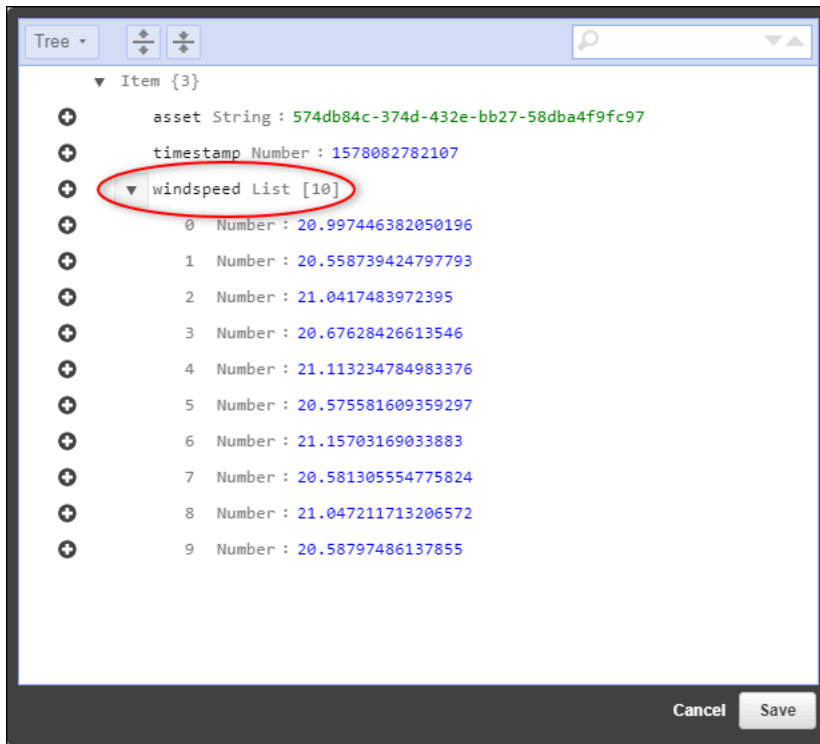
1. 返回標籤，並開啟 DynamoDB 資料表。
2. 在您稍早建立的資料表中，選擇 Items (項目) 標籤，以檢視資料表中的資料。如果您在資料表中沒有看見資料列，請重新整理頁面。如果在幾分鐘之後資料列沒有出現，請參閱[對規則進行故障診斷 \(DynamoDB\)](#)。



3. 在資料表的資料列中，選擇編輯圖示以展開資料。

Start search			
	timestamp ⓘ	asset	windspeed
☐	1578093637414	db36f80f-ed03-44d9-84ef-817eb30d5497	[{"N": "40.18707553698584"}, {"N": "40.20834808480326"}, {"N": "40.21081344172715"}, {"N": "40.218280888809424"}, {"N": "40.218912043562895"}, {"N": "40.22691091326525"}, {"N": "40.22876939941959"}, {"N": "40.21820505495924"}, {"N": "40.21820505495924"}, {"N": "40.21820505495924"}]
☐	1578093637422	db36f80f-ed03-44d9-84ef-817eb30d5497	[{"N": "40.21081344172715"}, {"N": "40.218280888809424"}, {"N": "40.218912043562895"}, {"N": "40.22691091326525"}, {"N": "40.22876939941959"}, {"N": "40.21820505495924"}, {"N": "40.21820505495924"}, {"N": "40.21820505495924"}]
☐	1578093637451	db36f80f-ed03-44d9-84ef-817eb30d5497	[{"N": "40.218912043562895"}, {"N": "40.22691091326525"}, {"N": "40.22876939941959"}, {"N": "40.21820505495924"}, {"N": "40.21820505495924"}, {"N": "40.21820505495924"}]
☐	1578093637453	db36f80f-ed03-44d9-84ef-817eb30d5497	[{"N": "40.22876939941959"}, {"N": "40.21820505495924"}, {"N": "40.21820505495924"}, {"N": "40.21820505495924"}]

4. 選擇 windspeed 結構旁邊的箭頭，展開風速資料點的清單。每個清單都會反映風力發電廠示範傳送至的 AWS IoT SiteWise 風速資料點批次。如果您為自身用途設定了規則動作，建議您使用不同的資料格式。如需詳細資訊，請參閱[在中查詢資產屬性通知 AWS IoT SiteWise](#)。



現在您已完成教學課程，請停用或刪除規則並刪除 DynamoDB 資料表，以避免產生額外費用。若要清除您的資源，請參閱 [步驟 6：在教學課程後清除資源](#)。

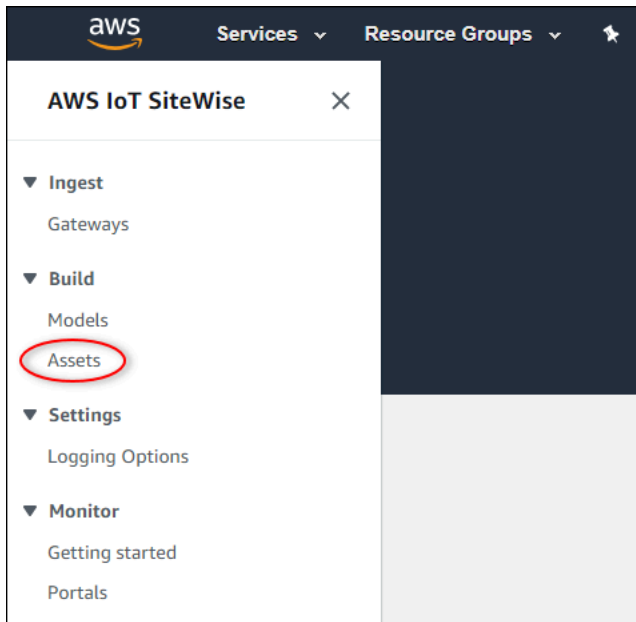
步驟 6：在教學課程後清除資源

完成本教學課程之後，請清除資源，以避免產生額外的費用。您的示範風力發電廠資產會在您建立示範時所選擇的持續時間結束時刪除。您也可以手動刪除示範。如需詳細資訊，請參閱[刪除 AWS IoT SiteWise 示範](#)。

使用下列程序來停用屬性值更新通知（如果您未刪除示範）、停用或刪除 AWS IoT 規則，以及刪除 DynamoDB 資料表。

停用資產屬性的屬性值更新通知

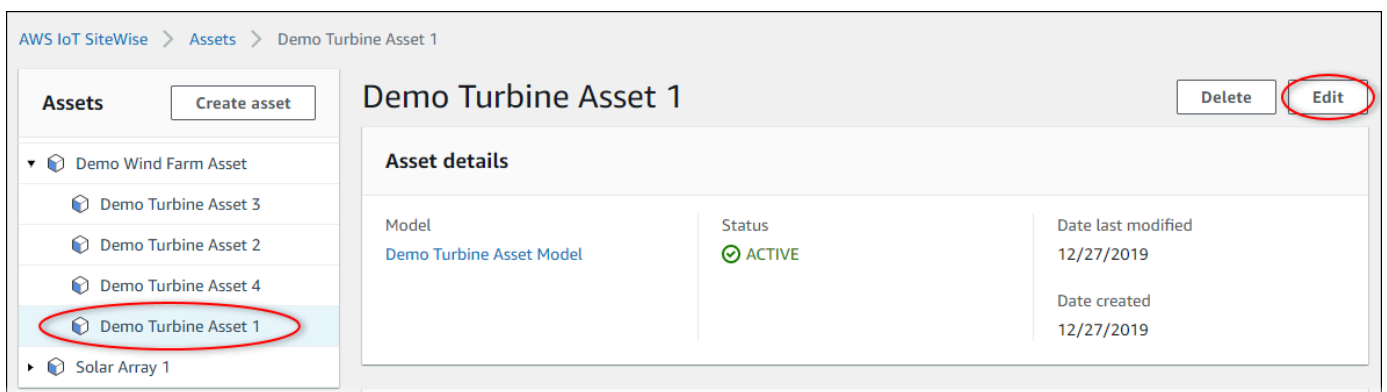
1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中，選擇 Assets (資產)。



3. 選擇 Demo Wind Farm Asset 旁的箭號，以展開發力發電廠資產的階層。



4. 選擇一個示範渦輪機，然後選擇 Edit (編輯)。



5. 將 Wind Speed 屬性的 Notification status (通知狀態) 更新為 DISABLED (停用)。

"Wind Speed"

Enter a property alias

Must be less than 2048 characters.

Notification status

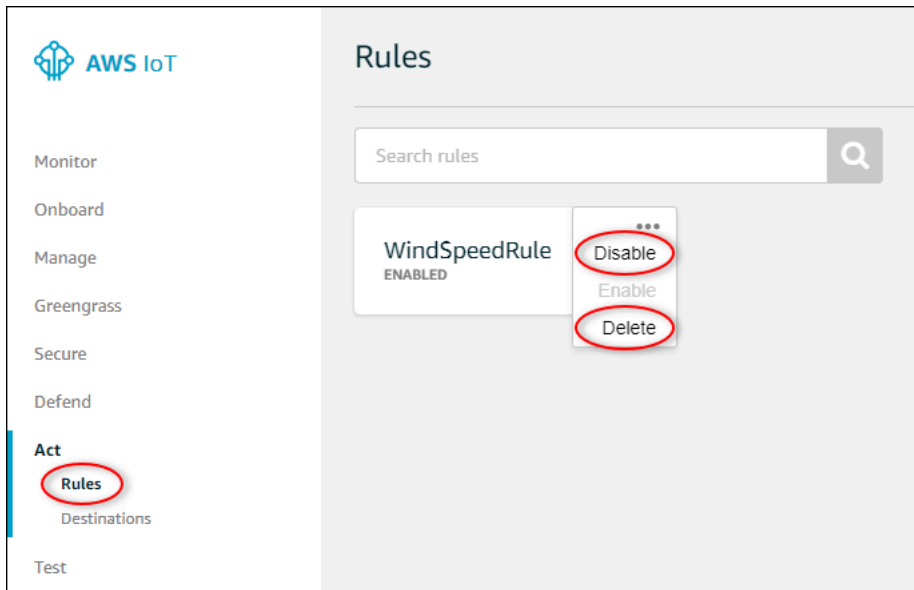
DISABLED

Notification will be published to topic \$aws/sitewise/asset-models/d8f8f20a-4d3a-491c-a9c5-352736979bdb/assets/db36f80f-ed03-44d9-84ef-817eb30d5497/properties/ca5b9e21-f19c-4ea1-8472-0e9400fc12bf

6. 選擇頁面底部的 Save asset (儲存資產)。
7. 針對每個示範渦輪機資產重複步驟 4 到 6。

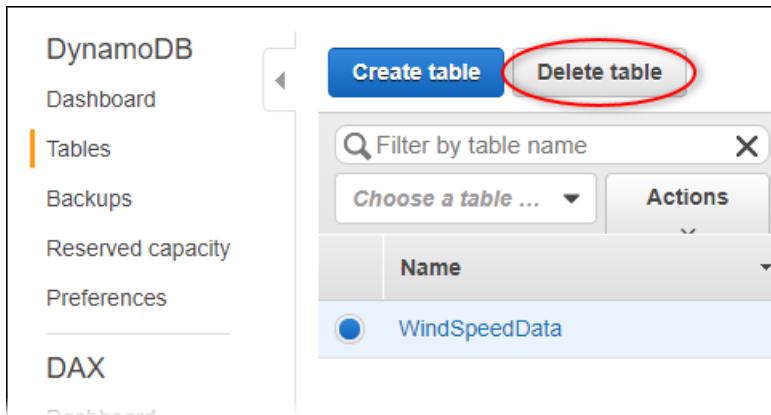
在中停用或刪除規則 AWS IoT Core

1. 導覽至 [AWS IoT 主控台](#)。
2. 在左側導覽窗格中，選擇 Act (動作)，然後選擇 Rules (規則)。
3. 選擇您規則上的選單，然後選擇 Disable (停用) 或 Delete (刪除)。

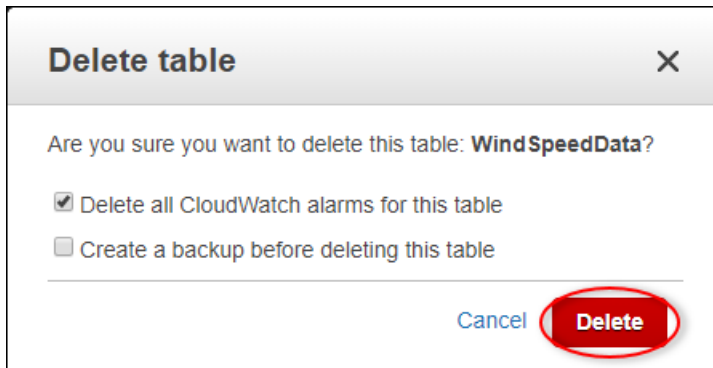


刪除 DynamoDB 資料表

1. 導覽至 [DynamoDB 主控台](#)。
2. 在左側導覽窗格中，選擇 Tables (資料表)。
3. 選擇您之前建立的資料表 WindSpeedData。
4. 選擇 刪除資料表。



5. 在 Delete table (刪除資料表) 對話方塊中，選擇 Delete (刪除)。



將資料擷取至 AWS IoT SiteWise

AWS IoT SiteWise 旨在有效率地收集工業資料，並將其與對應的資產相互關聯，代表工業操作的各個層面。本文件著重於將資料擷取到其中的實際層面 AWS IoT SiteWise，提供針對各種工業使用案例量身打造的多種方法。如需建立虛擬工業操作的指示，請參閱 [模型工業資產](#)。

您可以使用下列 AWS IoT SiteWise 任一選項，將工業資料傳送至：

- AWS IoT SiteWise Edge - 使用 [SiteWise Edge 閘道](#) 做為 AWS IoT SiteWise 與資料伺服器之間的媒介。AWS IoT SiteWise 提供元件，您可以在任何可執行的平台上部署這些 AWS IoT Greengrass 元件，AWS IoT Greengrass 以設定 SiteWise Edge 閘道。此選項支援與 [OPC UA](#) 伺服器通訊協定連結。
- AWS IoT SiteWise API – 使用 [AWS IoT SiteWise API](#) 從任何其他來源上傳資料。使用串流 [BatchPutAssetPropertyValue](#) API 在幾秒鐘內擷取，或批次導向的 [CreateBulkImportJob](#) API，以促進較大批次的成本效益擷取。
- AWS IoT 核心規則 – 使用 [AWS IoT 核心規則](#) 從 AWS IoT 物件或其他 AWS 服務發佈的 MQTT 訊息上傳資料。
- AWS IoT Events 動作 – 使用中特定事件觸發 [AWS IoT Events 的動作](#) AWS IoT Events。此方法適用於資料上傳與事件發生相關的案例。
- AWS IoT Greengrass 串流管理員 – 使用 [AWS IoT Greengrass 串流管理員](#)，使用邊緣裝置從本機資料來源上傳資料。此選項可因應資料來自內部部署或節點的情況。

這些方法提供各種解決方案，用於管理來自不同來源的資料。深入了解每個選項的詳細資訊，以全面了解資料擷取功能 AWS IoT SiteWise 所提供的功能。

管理 的資料串流 AWS IoT SiteWise

資料串流是包含歷史時間序列資料的資源。每個資料串流都由唯一的別名識別，讓您更輕鬆地追蹤每個資料片段的原始伺服器。收到第一次序列資料 AWS IoT SiteWise 時，資料串流會自動在 中建立。如果第一次使用別名識別序列資料，則 AWS IoT SiteWise 會使用該別名建立新的資料串流，前提是尚未指派該別名的資產屬性。或者，如果第一次使用資產 ID 和屬性 ID 識別序列資料，則 AWS IoT SiteWise 會建立新的資料串流，並將該資料串流與資產屬性建立關聯。

有兩種方式可將別名指派給資產屬性。所使用的方法取決於資料是 AWS IoT SiteWise 先傳送到，還是先建立資產。

- 如果資料 AWS IoT SiteWise 先傳送到 `TimeSeries`，這會自動使用指派的別名建立資料串流。稍後建立資產時，請使用 [AssociateTimeSeriesToAssetProperty](#) API 將資料串流及其別名與資產屬性建立關聯。
- 如果先建立資產，請使用 [UpdateAssetProperty](#) API 將別名指派給資產屬性。稍後傳送資料至 `TimeSeries` 時，AWS IoT SiteWise 會自動建立資料串流並與資產屬性建立關聯。

目前，您只能將資料串流與測量建立關聯。測量是一種資產屬性類型，代表裝置的原始感應器資料串流，例如時間戳記溫度值或每分鐘時間戳記輪換 (RPM) 值。

當這些測量定義指標或轉換時，傳入的資料會觸發特定的計算。請務必注意，資產屬性一次只能連結到一個資料串流。

AWS IoT SiteWise 會將 `TimeSeries` 用於 Amazon Resource Name (ARN) 資源，以判斷您的儲存費用。如需詳細資訊，請參閱 [AWS IoT SiteWise 定價](#)。

下列各節說明如何使用 AWS IoT SiteWise 主控台或 API 來管理資料串流。

主題

- [設定許可和設定](#)
- [將資料串流與資產屬性建立關聯](#)
- [取消資料串流與資產屬性的關聯](#)
- [刪除資料串流](#)
- [更新資產屬性別名](#)
- [常用案例](#)

設定許可和設定

收到第一次序列資料 AWS IoT SiteWise 時，資料串流會自動在 `TimeSeries` 中建立。如果擷取的資料未與資產屬性建立關聯，AWS IoT SiteWise 會建立新的取消關聯資料串流，可設定為與資產屬性建立關聯。使用 IAM 政策指定要擷取的資料類型 AWS IoT SiteWise，設定閘道傳送資料的存取控制。

下列 IAM 政策會停用與閘道取消關聯的資料擷取，同時仍然允許資料擷取至與資產屬性相關聯的資料串流：

Example IAM 使用者政策，可停用與閘道的取消關聯資料擷取

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

```

    "Statement": [
    {
      "Sid": "AllowPutAssetPropertyValuesUsingAssetIdAndPropertyId",
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "arn:aws:iotsitewise:*:*:asset/*"
    },
    {
      "Sid": "AllowPutAssetPropertyValuesUsingAliasWithAssociatedAssetProperty",
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "arn:aws:iotsitewise:*:*:time-series/*",
      "Condition": {
        "StringLikeIfExists": {
          "iotsitewise:isAssociatedWithAssetProperty": "true"
        }
      }
    },
    {
      "Sid": "DenyPutAssetPropertyValuesUsingAliasWithNoAssociatedAssetProperty",
      "Effect": "Deny",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "arn:aws:iotsitewise:*:*:time-series/*",
      "Condition": {
        "StringLikeIfExists": {
          "iotsitewise:isAssociatedWithAssetProperty": "false"
        }
      }
    }
  ]
}

```

Example 停用閘道所有資料擷取的 IAM 使用者政策

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyPutAssetPropertyValues",
      "Effect": "Deny",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": {
        "arn:aws:iotsitewise:*:*:asset/*",

```

```
        "arn:aws:iotsitewise:*:*:time-series/*"
    }
}
]
```

將資料串流與資產屬性建立關聯

使用 AWS IoT SiteWise 主控台 或 管理您的資料串流 AWS CLI。

Console

使用 AWS IoT SiteWise 主控台管理您的資料串流。

管理資料串流（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇資料串流。
3. 篩選資料串流別名，或在篩選條件下拉式選單中選取取消關聯資料串流，以選擇資料串流。
4. 選取要更新的資料串流。您可以選取多個資料串流。按一下右上角的管理資料串流。
5. 從更新資料串流關聯選取要關聯的資料串流，然後按一下選擇測量按鈕。
6. 在選擇測量區段中，尋找對應的資產測量屬性。選取測量，然後按一下選擇。
7. 針對步驟 3 中選取的其他資料串流執行步驟 4 和 5。將資產屬性指派給所有資料串流。
8. 選擇更新以遞交變更。隨即顯示成功的確認橫幅，以確認更新。

AWS CLI

若要將資料串流（透過其別名識別）與資產屬性（透過其 IDs 識別）建立關聯，請執行下列命令：

```
aws iotsitewise associate-time-series-to-asset-property \
  --alias <data-stream-alias> \
  --assetId <asset-ID> \
  --propertyId <property-ID>
```

取消資料串流與資產屬性的關聯

Console

使用 AWS IoT SiteWise 主控台取消資料串流與資產屬性的關聯。

取消資料串流與資產屬性的關聯（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇資料串流。
3. 篩選資料串流別名，或在篩選條件下拉式選單中選取相關聯的資料串流，以選擇資料串流。
4. 選取要取消關聯的資料串流。資料串流別名欄必須包含別名。資產名稱和資產屬性名稱資料欄必須包含資料串流相關聯之資產屬性的值。您可以選取多個資料串流。
5. 按一下右上角的管理資料串流。
6. 在更新資料串流關聯區段中，按一下測量名稱欄中的 X。submitted 狀態應該會出現在狀態欄中。
7. 選擇更新以遞交變更。資料串流現在與資產屬性取消關聯，且別名現在用於識別資料串流。

AWS CLI

若要取消資料串流與資產屬性的關聯（由其 ID 及其別名識別），請執行下列命令：

```
aws iotsitewise disassociate-time-series-from-asset-property \  
  --alias <asset-property-alias> \  
  --assetId <asset-ID> \  
  --propertyId <property-ID>
```

資料串流現在會與資產屬性取消關聯，並使用別名來識別資料串流。別名不再與資產屬性相關聯，因為它現在與資料串流相關聯。

刪除資料串流

從資產模型移除屬性時，會從資產模型管理的所有資產 AWS IoT SiteWise 中刪除屬性及其資料串流。刪除資產時，也會刪除資產的所有屬性及其資料串流。如果必須保留資料串流資料，必須先將其與資產屬性取消關聯，才能將其刪除。

Warning

從資產刪除屬性時，也會刪除相關聯的資料串流。若要保留資料串流，請先將其與資產屬性取消關聯，然後再從資產模型刪除屬性，或刪除資產。

Console

使用 AWS IoT SiteWise 主控台取消資料串流與資產屬性的關聯。

刪除資料串流（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇資料串流。
3. 透過篩選資料串流別名來選擇資料串流。
4. 選取要刪除的資料串流。您可以選取多個資料串流。
5. 選擇刪除按鈕以刪除資料串流。

AWS CLI

使用 [DeleteTimeSeries](#) API 依其別名刪除特定資料串流。

```
aws iotsitewise delete-time-series \  
  --alias <data-stream-alias>
```

更新資產屬性別名

別名在 AWS 區域內必須是唯一的。這包括資產屬性和資料串流的別名。如果另一個屬性或資料串流使用該別名，請勿將別名指派給資產屬性。

Console

使用 AWS IoT SiteWise 主控台更新資產屬性別名。

更新資產屬性別名（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。

2. 在導覽窗格中，選擇 Assets (資產)。
3. 從資料表中選取資產。
4. 按一下 Edit (編輯) 按鈕。
5. 在屬性資料表中選取屬性類型。
6. 尋找 屬性，然後在屬性別名文字欄位中輸入新的別名。
7. 按一下儲存按鈕以儲存變更。

AWS CLI

若要更新資產屬性上的別名，請執行下列命令：

```
aws iotsitewise update-asset-property \  
  --asset-id <asset-ID> \  
  --property-id <property-ID> \  
  --property-alias <asset-property-alias> \  
  --property-notification-state <ENABLED|DISABLED>
```

Note

如果屬性通知目前已啟用，則必須再次提供，以確保它繼續啟用。

常用案例

移動資料串流

若要變更資料串流與另一個資產屬性的關聯，請先取消資料串流與目前資產屬性的關聯。取消資料串流與資產屬性的關聯時，必須指派別名至該資產屬性。

```
aws iotsitewise disassociate-time-series-from-asset-property \  
  --alias <asset-property-alias> \  
  --assetId <asset-ID> \  
  --propertyId <property-ID>
```

現在將資料串流重新指派給新的資產屬性。

```
aws iotsitewise associate-time-series-from-asset-property \  
  --alias <data-stream-alias> \  
  --assetId <new-asset-ID> \  
  --propertyId <new-property-ID>
```

將別名指派給資產屬性時發生錯誤

使用 UpdateAssetProperty API 將別名指派給屬性時，您可能會看到下列錯誤訊息：

```
Given alias <data-stream-alias> for property <property-name> with ID <property-ID>  
already in use by another property or data stream
```

此錯誤訊息表示別名未指派給 屬性，因為目前由另一個屬性或資料串流使用。

如果 AWS IoT SiteWise 使用別名擷取資料至 ，就會發生這種情況。當資料以另一個資料串流或資產屬性未使用的別名傳送時，會使用該別名建立新的資料串流。以下兩個選項可解決問題。

- 使用 AssociateTimeSeriesToAssetProperty API 將資料串流與其別名與資產屬性建立關聯。
- 暫時停止資料擷取並刪除資料串流。使用 UpdateAssetProperty API 將別名指派給資產屬性，然後重新開啟資料擷取。

將資料串流與資產屬性建立關聯時發生錯誤

將資料串流與資產屬性建立關聯時，會看到下列錯誤訊息。

```
assetProperty <property-name> with assetId <asset-ID> propertyId <property-ID> contains  
data
```

此錯誤訊息表示資產屬性已與包含資料的資料串流相關聯。在將其他資料串流與該資產屬性建立關聯之前，該資料串流必須取消關聯或刪除。

Note

取消資料串流與資產屬性的關聯時，指派給屬性的別名會提供給資料串流。若要讓該別名保持指派給 屬性，請先將新的別名指派給該屬性，再取消資料串流的關聯。

若要保留儲存在資產屬性中的資料，請執行下列動作：

- 確保資產屬性未擷取任何資料，以防止建立新的資料串流。
- 使用 `UpdateAssetProperty` API 設定提供給目前指派資料串流的新別名。
- 使用 `DisassociateTimeSeriesFromAssetProperty` API 取消目前資料串流與資產屬性的關聯。
- 使用 `AssociateTimeSeriesToAssetProperty` API 將所需的資料串流與資產屬性建立關聯。

如果必須刪除存放在資產屬性中的資料，請執行下列動作：

- 確保資產屬性未擷取任何資料，以防止建立新的資料串流。
- 使用 `DeleteTimeSeries` API 刪除目前指派的資料串流。
- 使用 `AssociateTimeSeriesToAssetProperty` API 將所需的資料串流與資產屬性建立關聯。

使用 AWS IoT SiteWise APIs 擷取資料

使用 AWS IoT SiteWise APIs 將時間戳記工業資料傳送至資產的屬性和測量屬性。APIs 接受包含 timestamp-quality-value (TQV) 結構的承載。

BatchPutAssetPropertyValue API

使用 [BatchPutAssetPropertyValue](#) 操作上傳您的資料。透過此操作，您可以一次上傳多個資料項目，以從多個裝置收集資料，並在單一請求中傳送所有資料。

Important

[BatchPutAssetPropertyValue](#) 操作受下列配額限制：

- 每個請求最多 10 個 [項目](#)。
- 每個項目最多 10 個 [屬性值](#) (TQV 資料點)。
- AWS IoT SiteWise 會拒絕時間戳記為過去 7 天以上或未來超過 10 分鐘的任何資料。

如需這些配額的詳細資訊，請參閱 AWS IoT SiteWise API 參考中的 [BatchPutAssetPropertyValue](#)。

若要識別資產屬性，請指定下列其中一項：

- 資料傳送至的資產屬性propertyId的 assetId和。
- propertyAlias，這是資料串流別名（例如 /company/windfarm/3/turbine/7/temperature）。若要使用這個選項，您必須先設定您的資產屬性別名。若要設定屬性別名，請參閱[管理的資料串流 AWS IoT SiteWise](#)。

以下範例示範如何從儲存在 JSON 檔案的承載中，設定風力渦輪機的溫度和每分鐘旋轉 (RPM) 讀數。

```
aws iotsitewise batch-put-asset-property-value --cli-input-json file://batch-put-payload.json
```

中的範例承載batch-put-payload.json具有下列內容。

```
{
  "enablePartialEntryProcessing": true,
  "entries": [
    {
      "entryId": "unique entry ID",
      "propertyAlias": "/company/windfarm/3/turbine/7/temperature",
      "propertyValues": [
        {
          "value": {
            "integerValue": 38
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    },
    {
      "entryId": "unique entry ID",
      "propertyAlias": "/company/windfarm/3/turbine/7/rpm",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 15.09
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          },
          "quality": "GOOD"
        }
      ]
    }
  ]
}
```

```

    }
  ]
},
{
  "entryId": "unique entry ID",
  "propertyAlias": "/company/windfarm/3/turbine/7/rpm",
  "propertyValues": [
    {
      "value": {
        "nullValue":{"valueType": "D"}
      },
      "timestamp": {
        "timeInSeconds": 1575691200
      },
      "quality": "BAD"
    }
  ]
}
]
}
}

```

將指定 `enablePartialEntryProcessing` 為 `true` 允許擷取不會導致失敗的所有值。預設行為是 `false`。如果值無效，則整個項目無法擷取。

承載中的每個項目都包含 `entryId`，您可以將其定義為任何獨特的字串。如果有任何請求項目失敗，每個錯誤都會包含對應請求的 `entryId`，這樣您就可以知道要重試哪些請求。

`propertyValues` 清單中的每個結構都是時間戳記品質值 (TQV) 結構，其中包含一個 `value`、一個 `timestamp`，以及選擇性地包含一個 `quality`。

- `value` – 包含下列其中一個欄位的結構，取決於所設定屬性的類型：
 - `booleanValue`
 - `doubleValue`
 - `integerValue`
 - `stringValue`
 - `nullValue`
- `nullValue` – 具有下列欄位的結構，表示值為 `Null` 且品質為 `BAD` 或 的屬性值類型 `UNCERTAIN`。
 - `valueType` – {"B"、"D"、"S"、"I"} 的列舉

- **timestamp** – 包含目前 Unix epoch 時間的結構，以秒為單位 `timeInSeconds`，。如果您有時間精確的資料，也可以在 `timestamp` 結構中設定 `offsetInNanos` 金鑰。AWS IoT SiteWise 會拒絕過去 7 天或未來 10 分鐘以上時間戳記的任何資料點。
- **quality** – (選用) 下列其中一個品質字串：
 - **GOOD** – (預設) 資料不會受到任何問題的影響。
 - **BAD** – 資料會受到感應器故障等問題的影響。
 - **UNCERTAIN** – 資料受到感應器不準確等問題的影響。

如需 如何處理運算中 AWS IoT SiteWise 資料品質的詳細資訊，請參閱[公式表達式中的資料品質](#)。

CreateBulkImportJob API

使用 `CreateBulkImportJob` API 從 Amazon S3 匯入大量資料。您的資料必須以 CSV 格式儲存在 Amazon S3 中。資料檔案可以有下列資料欄。

Note

不支援早於 1970 年 1 月 1 日 00:00:00 UTC 的資料。

若要識別資產屬性，請指定下列其中一項。

- 您要將資料傳送到其中 `PROPERTY_ID` 的資產屬性的 `ASSET_ID` 和。
- `ALIAS`，這是資料串流別名（例如 `/company/windfarm/3/turbine/7/temperature`）。若要使用這個選項，您必須先設定您的資產屬性別名。若要了解如何設定屬性別名，請參閱[the section called “管理資料串流”](#)。

- `ALIAS` – 識別 屬性的別名，例如 OPC UA 伺服器資料串流路徑（例如 `/company/windfarm/3/turbine/7/temperature`）。如需詳細資訊，請參閱[管理 的資料串流 AWS IoT SiteWise](#)。
- `ASSET_ID` – 資產的 ID。
- `PROPERTY_ID` – 資產屬性的 ID。
- `DATA_TYPE` – 屬性的資料類型可以是下列其中一項。
 - `STRING` – 最多 1024 個位元組的字串。
 - `INTEGER` – 範圍為 `[-2,147,483,648, 2,147,483,647]` 的帶正負號 32 位元整數。
 - `DOUBLE` – 範圍為 `[-10^100, 10^100]` 和 IEEE 754 雙精度的浮點數。
 - `BOOLEAN` – `true` 或 `false`。

- `TIMESTAMP_SECONDS` – 資料點的時間戳記，以 Unix epoch 時間表示。
- `TIMESTAMP_NANO_OFFSET` – 從涵蓋的奈秒位移 `TIMESTAMP_SECONDS`。
- `QUALITY` – (選用) 資產屬性值的品質。值可以是下列其中一項。
 - `GOOD` – (預設) 資料不會受到任何問題的影響。
 - `BAD` – 資料會受到感應器故障等問題的影響。
 - `UNCERTAIN` – 資料受到感應器不準確等問題的影響。

如需 如何處理運算中 AWS IoT SiteWise 資料品質的詳細資訊，請參閱[公式表達式中的資料品質](#)。

- `VALUE` – 資產屬性的值。

Example .csv 格式的資料檔案 (s)

```
asset_id,property_id,DOUBLE,1635201373,0,GOOD,1.0
asset_id,property_id,DOUBLE,1635201374,0,GOOD,2.0
asset_id,property_id,DOUBLE,1635201375,0,GOOD,3.0
```

```
unmodeled_alias1,DOUBLE,1635201373,0,GOOD,1.0
unmodeled_alias1,DOUBLE,1635201374,0,GOOD,2.0
unmodeled_alias1,DOUBLE,1635201375,0,GOOD,3.0
unmodeled_alias1,DOUBLE,1635201376,0,GOOD,4.0
unmodeled_alias1,DOUBLE,1635201377,0,GOOD,5.0
unmodeled_alias1,DOUBLE,1635201378,0,GOOD,6.0
unmodeled_alias1,DOUBLE,1635201379,0,GOOD,7.0
unmodeled_alias1,DOUBLE,1635201380,0,GOOD,8.0
unmodeled_alias1,DOUBLE,1635201381,0,GOOD,9.0
unmodeled_alias1,DOUBLE,1635201382,0,GOOD,10.0
```

AWS IoT SiteWise 提供下列 API 操作來建立大量匯入任務，並取得現有任務的相關資訊。

- [CreateBulkImportJob](#) – 建立新的大量匯入任務。
- [DescribeBulkImportJob](#) – 擷取大量匯入任務的相關資訊。
- [ListBulkImportJob](#) – 擷取所有大量匯入任務摘要的分頁清單。

建立 AWS IoT SiteWise 大量匯入任務 (AWS CLI)

使用 [CreateBulkImportJob](#) API 操作，將資料從 Amazon S3 傳輸到 AWS IoT SiteWise。 [CreateBulkImportJob](#) API 可擷取大量歷史資料，並以小批次緩衝擷取分析資料串流。它為資料擷取提供了經濟實惠的基礎。下列為使用 AWS CLI 的範例。

Important

在建立大量匯入任務之前，您必須啟用 AWS IoT SiteWise 暖層或 AWS IoT SiteWise 冷層。如需詳細資訊，請參閱 [在中設定儲存體設定 AWS IoT SiteWise](#)。

[CreateBulkImportJob](#) API 支援將歷史資料擷取至 `adaptive-ingestion-flag` 參數。

- 設為 `false`，API 會擷取歷史資料，而不會觸發運算或通知。
- 設為 `true`，API 會擷取新資料、計算指標和轉換資料，以在七天內最佳化持續的分析和通知。

執行下列命令。將 *file-name* 取代為包含大量匯入任務組態的檔案名稱。

```
aws iotsitewise create-bulk-import-job --cli-input-json file://file-name.json
```

Example 大量匯入任務組態

以下是組態設定的範例：

- 將 *adaptive-ingestion-flag* 取代為 `true` 或 `false`。
 - 如果設定為 `false`，大量匯入任務會將歷史資料擷取至 AWS IoT SiteWise。
 - 如果設定為 `true`，大量匯入任務會執行下列動作：
 - 擷取新資料 AWS IoT SiteWise。
 - 計算指標和轉換，並支援在七天內時間戳記的資料通知。
- 將 *delete-files-after-import-flag* 取代為 `true`，以在擷取至 AWS IoT SiteWise 暖層儲存之後從 Amazon S3 資料儲存貯體中刪除資料。
- 將 *amzn-s3-demo-bucket-for-errors* 取代為 Amazon S3 儲存貯體的名稱，此儲存貯體會傳送與此大量匯入任務相關聯的錯誤。
- 將 *amzn-s3-demo-bucket-for-errors-prefix* 取代為 Amazon S3 儲存貯體的字首，其會傳送與此大量匯入任務相關聯的錯誤。

Amazon S3 使用字首做為資料夾名稱，來組織儲存貯體中的資料。每個 Amazon S3 物件都有一個金鑰，其為儲存貯體中的唯一識別符。儲存貯體中的每個物件只能有一個金鑰。前綴必須以正斜線 (/) 結尾。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[使用字首組織物件](#)。

- 將 `amzn-s3-demo-bucket-data` 取代為從中匯入資料的 Amazon S3 儲存貯體名稱。
- 將 `data-bucket-key` 取代為包含您資料的 Amazon S3 物件的金鑰。每個物件都有一個唯一識別碼的金鑰。每個物件只有一個金鑰。
- 將 `data-bucket-version-id` 取代為版本 ID，以識別包含您資料的特定 Amazon S3 物件版本。此為選用參數。
- 將 `column-name` 取代為 .csv 檔案中指定的資料欄名稱。
- 將 `job-name` 取代為識別大量匯入任務的唯一名稱。
- 將 `job-role-arn` 取代為允許 AWS IoT SiteWise 讀取 Amazon S3 資料的 IAM 角色。

Note

請確定您的角色具有下列範例所示的許可。將 `amzn-s3-demo-bucket-data` 取代為包含您資料的 Amazon S3 儲存貯體名稱。此外，請將 `amzn-s3-demo-bucket-for-errors` 取代為 Amazon S3 儲存貯體的名稱，其會傳送與此大量匯入任務相關聯的錯誤。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket-data",
        "arn:aws:s3:::amzn-s3-demo-bucket-data/*",
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
```

```

        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket-for-errors",
        "arn:aws:s3:::amzn-s3-demo-bucket-for-errors/*"
      ],
      "Effect": "Allow"
    }
  ]
}

```

```

{
  "adaptiveIngestion": adaptive-ingestion-flag,
  "deleteFilesAfterImport": delete-files-after-import-flag,
  "errorReportLocation": {
    "bucket": "amzn-s3-demo-bucket-for-errors",
    "prefix": "amzn-s3-demo-bucket-for-errors-prefix"
  },
  "files": [
    {
      "bucket": "amzn-s3-demo-bucket-data",
      "key": "data-bucket-key",
      "versionId": "data-bucket-version-id"
    }
  ],
  "jobConfiguration": {
    "fileFormat": {
      "csv": {
        "columnNames": [ "column-name" ]
      }
    }
  },
  "jobName": "job-name",
  "jobRoleArn": "job-role-arn"
}

```

Example response

```

{
  "jobId": "f8c031d0-01d1-4b94-90b1-afe8bb93b7e5",
  "jobStatus": "PENDING",

```



```
"jobName": "myBulkImportJob"
}
```

描述 AWS IoT SiteWise 大量匯入任務 (AWS CLI)

使用 [DescribeBulkImportJob](#) API 操作來擷取特定大量匯入任務的相關資訊 AWS IoT SiteWise。此操作會傳回詳細資訊，例如任務的狀態、建立時間和任務失敗時的錯誤資訊。您可以使用此操作來監控任務進度和疑難排解問題。若要使用 `DescribeBulkImportJob`，您需要 `CreateBulkImportJob` 操作中的任務 ID。API 會傳回下列資訊：

- 匯入的檔案清單，包括其 Amazon S3 儲存貯體位置和金鑰
- 錯誤報告位置（如適用）
- 任務組態詳細資訊，例如檔案格式和 CSV 資料欄名稱
- 任務建立和上次更新時間戳記
- 目前的任務狀態（例如，任務正在進行中、已完成或失敗）
- 用於匯入任務的 IAM 角色 ARN

對於已完成的任務，請檢閱結果以確認資料整合成功。如果任務失敗，請檢查錯誤詳細資訊以診斷和解決問題。

將 *job-ID* 取代為您要擷取的大量匯入任務 ID。

```
aws iotsitewise describe-bulk-import-job --job-id job-ID
```

Example response

```
{
  "files": [
    {
      "bucket": "amzn-s3-demo-bucket1",
      "key": "100Tags12Hours.csv"
    },
    {
      "bucket": "amzn-s3-demo-bucket2",
      "key": "BulkImportData1MB.csv"
    },
    {
      "bucket": " amzn-s3-demo-bucket3",
```

```
        "key": "UnmodeledBulkImportData1MB.csv"
      },
    ],
    "errorReportLocation": {
      "prefix": "errors/",
      "bucket": "amzn-s3-demo-bucket-for-errors"
    },
    "jobConfiguration": {
      "fileFormat": {
        "csv": {
          "columnNames": [
            "ALIAS",
            "DATA_TYPE",
            "TIMESTAMP_SECONDS",
            "TIMESTAMP_NANO_OFFSET",
            "QUALITY",
            "VALUE"
          ]
        }
      }
    },
    "jobCreationDate": 1645745176.498,
    "jobStatus": "COMPLETED",
    "jobName": "myBulkImportJob",
    "jobLastUpdateDate": 1645745279.968,
    "jobRoleArn": "arn:aws:iam::123456789012:role/DemoRole",
    "jobId": "f8c031d0-01d1-4b94-90b1-afe8bb93b7e5"
  }
}
```

列出 AWS IoT SiteWise 大量匯入任務 (AWS CLI)

使用 [ListBulkImportJobs](#) API 操作來擷取大量匯入任務的摘要清單 AWS IoT SiteWise。此操作提供監控和管理資料匯入程序的有效方法。它會傳回每個任務的下列金鑰資訊：

- 任務 ID。每個大量匯入任務的唯一識別符
- 任務名稱。您在建立任務時指派給任務的名稱
- 目前狀態。任務的目前狀態（例如 COMPLETED、RUNNING、FAILED）

ListBulkImportJobs 特別適用於全面了解所有大量匯入任務。這可協助您追蹤多個資料匯入、識別任何需要注意的任務，以及維護有組織的工作流程。操作支援分頁，可讓您有效率地擷取大量任務摘要。您可以搭配 [DescribeBulkImportJob](#) 操作使用此操作傳回的任務 IDs，以擷取特定任務的詳細

資訊。這個兩步驟程序可讓您先取得所有任務的高階檢視，然後深入了解感興趣的任務詳細資訊。使用時 `ListBulkImportJobs`，您可以套用篩選條件來縮小結果範圍。例如，您可以根據任務狀態篩選任務，以僅擷取已完成的任務或僅執行中的任務。此功能可協助您專注於目前任務的最相關資訊。`nextToken` 如果有更多結果可用，操作也會傳回。您可以在後續呼叫中使用此權杖來擷取下一組任務摘要，即使您有大量任務，也能逐一查看所有大量匯入任務。下列範例示範如何使用 `ListBulkImportJobs` 搭配 AWS CLI 來擷取已完成任務的清單。

```
aws iotsitewise list-bulk-import-jobs --filter COMPLETED
```

Example 已完成任務的回應篩選條件

```
{
  "jobSummaries":[
    {
      "id":"bdbbfa52-d775-4952-b816-13ba1c7cb9da",
      "name":"myBulkImportJob",
      "status":"COMPLETED"
    },
    {
      "id":"15fffc641-dbd8-40c6-9983-5cb3b0bc3e6b",
      "name":"myBulkImportJob2",
      "status":"COMPLETED"
    }
  ]
}
```

此命令示範如何使用 `ListBulkImportJobs` 來擷取已完成但失敗的任務清單。最大值設定為 50 個結果，我們使用下一個字符來分頁結果。

```
aws iotsitewise list-bulk-import-jobs --filter COMPLETED_WITH_FAILURES --max-results 50
--next-token "string"
```

AWS IoT SiteWise 使用 AWS IoT Core 規則將資料擷取至

使用中的規則 AWS IoT SiteWise，從 AWS IoT 實物和其他 AWS 服務將資料傳送至 AWS IoT Core。規則會轉換 MQTT 訊息並執行動作以與服務互動 AWS。AWS IoT SiteWise 規則動作會從 AWS IoT SiteWise API 轉送訊息資料至 [BatchPutAssetPropertyValue](#) 操作。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的 [規則](#) 和 [AWS IoT SiteWise 動作](#)。

若要遵循教學課程逐步完成設定透過裝置影子擷取資料之規則所需的步驟，請參閱 [AWS IoT SiteWise 從 AWS IoT 實物擷取資料至](#)。

您也可以從 將資料 AWS IoT SiteWise 傳送至其他 AWS 服務。如需詳細資訊，請參閱[與其他 AWS 服務互動](#)。

主題

- [授予 AWS IoT 必要的存取權](#)
- [設定 AWS IoT SiteWise 規則動作](#)
- [在 中使用基本擷取降低成本 AWS IoT SiteWise](#)

授予 AWS IoT 必要的存取權

您可以使用 IAM 角色來控制每個規則可存取 AWS 的資源。建立規則之前，您必須建立具有政策的 IAM 角色，該政策允許規則對所需的 AWS 資源執行動作。在執行規則時 AWS IoT 擔任此角色。

如果您在 AWS IoT 主控台中建立規則動作，您可以選擇根資產來建立可存取所選資產階層的角色。如需如何手動定義規則角色的詳細資訊，請參閱《AWS IoT 開發人員指南》中的[授予 AWS IoT 必要的存取和傳遞角色許可](#)。

對於 AWS IoT SiteWise 規則動作，您必須定義角色，允許 `iotsitewise:BatchPutAssetPropertyValue` 存取規則傳送資料的資產屬性。若要改善安全性，您可以在 Condition 屬性中指定 AWS IoT SiteWise 資產階層路徑。

下列範例信任政策可允許存取特定資產及其子系。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iotsitewise:assetHierarchyPath": [
            "/root node asset ID",
            "/root node asset ID/*"
          ]
        }
      }
    }
  ]
}
```

```
}  
}  
]  
}
```

Condition 從政策中移除，以允許存取您的所有資產。下列範例信任政策允許存取您目前「Region」(區域) 中的所有資產。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "iotsitewise:BatchPutAssetPropertyValue",  
      "Resource": "*"   
    }  
  ]  
}
```

設定 AWS IoT SiteWise 規則動作

AWS IoT SiteWise 規則動作會從啟動規則的 MQTT 訊息將資料傳送至其中的資產屬性 AWS IoT SiteWise。您可以同時將多個資料項目上傳至不同的資產屬性，以一則訊息傳送裝置所有感應器的更新。您也可以針對每個資料項目一次上傳多個資料點。

Note

當您 AWS IoT SiteWise 使用規則動作將資料傳送至時，您的資料必須符合BatchPutAssetPropertyValue操作的所有要求。例如，您的資料的時間戳記不能早於目前 Unix epoch 時間的 7 天。如需詳細資訊，請參閱[使用 AWS IoT SiteWise API 導入資料](#)。

針對規則動作中的每個資料項目，您需要識別資產屬性並指定時間戳記、品質，以及該資產屬性的每個資料點的值。規則動作預期會接受所有參數的字串。

若要識別項目中的資產屬性，請指定下列其中一個項目：

- 您要傳送資料到其中的資產屬性的 Asset ID (資產 ID) (assetId) 和 Property ID (屬性 ID) (propertyId)。您可以使用 [尋找資產 ID 和屬性 ID](#) AWS IoT SiteWise 主控台。如果您知道資產 ID，您可以使用 AWS CLI 呼叫 [DescribeAsset](#) 來尋找屬性 ID。

- Property alias (屬性別名) (propertyAlias)，這是資料串流別名 (例如 /company/windfarm/3/turbine/7/temperature)。若要使用這個選項，您必須先設定您的資產屬性別名。若要了解如何設定屬性別名，請參閱[管理的資料串流 AWS IoT SiteWise](#)。

對於每個項目中的時間戳記，請使用設備報告的時間戳記或 提供的時間戳記 AWS IoT Core。時間戳記有兩個參數：

- 以秒為單位的時間 (timeInSeconds) – Unix epoch 以秒為單位的時間，感應器或設備在此時間回報資料。
- 以奈米為單位的位移 (offsetInNanos) – (選用) 以秒為單位的奈秒位移。

Important

如果您的時間戳記是字串、有小數部分，或不是以秒為單位，AWS IoT SiteWise 會拒絕請求。您必須將時間戳記轉換為秒和奈秒位移。使用 AWS IoT 規則引擎的功能來轉換時間戳記。如需詳細資訊，請參閱下列內容：

- [取得未報告準確時間之裝置的時間戳記](#)
- [轉換字串格式的時間戳記](#)

您可以在 動作中使用多個參數的替代範本來執行計算、叫用函數，以及從訊息承載提取值。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[替代範本](#)。

Note

因為替代範本中的運算式與 SELECT 陳述式是分開計算的，所以您不能使用替代範本來參考使用 AS 子句所建立的別名。除了支援的函數與運算子以外，您可以僅參照原始承載中出現的資訊。

主題

- [取得未報告準確時間之裝置的時間戳記](#)
- [轉換字串格式的時間戳記](#)
- [轉換奈秒精度時間戳記字串](#)
- [範例規則組態](#)

- [針對 規則動作進行故障診斷](#)

取得未報告準確時間之裝置的時間戳記

如果您的感應器或設備未報告準確的時間資料，請從具有 [timestamp\(\)](#) 的 AWS IoT 規則引擎取得目前的 Unix epoch 時間。此函數會以毫秒為單位輸出時間，因此您必須將值轉換為以秒為單位的時間，並以奈秒為單位偏移。若要這麼做，請使用下列轉換：

- 針對 Time in seconds (以秒為單位的時間) (timeInSeconds)，使用 `${floor(timestamp() / 1E3)}` 將時間從毫秒轉換為秒。
- 針對 Offset in nanos (以奈米為單位的位移) (offsetInNanos)，使用 `${(timestamp() % 1E3) * 1E6}` 計算時間戳記的奈米秒位移。

轉換字串格式的時間戳記

如果您的感應器或設備以字串格式（例如 2020-03-03T14:57:14.699Z）報告時間資料，請使用 [time_to_epoch\(String, String\)](#)。此函數會將時間戳記和格式模式輸入為參數，並以毫秒為單位輸出時間。然後，您必須以秒為單位將時間轉換為時間，並以奈秒為單位偏移。若要這麼做，請使用下列轉換：

- 對於以秒為單位的時間 (timeInSeconds)，使用 將時間戳記字串 `${floor(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z'") / 1E3)}` 轉換為毫秒，然後轉換為秒。
- 對於以奈米為單位的位移 (offsetInNanos)，請使用 `${(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z'") % 1E3) * 1E6}` 計算時間戳記字串的奈秒位移。

Note

`time_to_epoch` 函數支援高達毫秒精確度的時間戳記字串。若要轉換微秒或奈秒精確度的字串，請設定您的規則呼叫的 AWS Lambda 函數，將時間戳記轉換為數值。如需詳細資訊，請參閱 [轉換奈秒精度時間戳記字串](#)。

轉換奈秒精度時間戳記字串

如果您的裝置以奈秒精確度（例如 2020-03-03T14:57:14.699728491Z）傳送字串格式的時間戳記資訊，請使用下列程序來設定規則動作。您可以建立 AWS Lambda 函數，將時間戳記從字串轉換為以秒為單位的時間 (timeInSeconds) 和以奈米為單位的位移 (offsetInNanos)。然後，在規則動作參數中使用 [aws_lambda\(functionArn, inputJson\)](#) 來叫用該 Lambda 函數，並在規則中使用輸出。

Note

本節包含進階說明，會假設您已熟悉如何建立以下資源：

- Lambda 函數。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的[建立您的第一個 Lambda 函數](#)。
- AWS IoT 規則與 AWS IoT SiteWise 規則動作。如需詳細資訊，請參閱[AWS IoT SiteWise 使用 AWS IoT Core 規則將資料擷取至](#)。

建立剖析時間戳記字串的 AWS IoT SiteWise 規則動作

1. 使用下列屬性建立 Lambda 函數：

- 函數名稱 – 使用描述性函數名稱（例如 **ConvertNanosecondTimestampFromString**）。
- 執行時間 – 使用 Python 3 執行時間，例如 Python 3.11 (python3.11)。
- 許可 – 建立具有基本 Lambda 許可 (AWS LambdaBasicExecutionRole) 的角色。
- Layers – 新增 AWS SDKPandas-Python311 layer，讓 Lambda 函數使用 numpy。
- 函數程式碼 – 使用下列函數程式碼，這會使用名為 `timestamp` 的字串引數，並針對該時間戳記輸出 `timeInSeconds` 和 `offsetInNanos` 值。

```
import json
import math
import numpy

# Converts a timestamp string into timeInSeconds and offsetInNanos in Unix epoch
time.
# The input timestamp string can have up to nanosecond precision.
def lambda_handler(event, context):
    timestamp_str = event['timestamp']
    # Parse the timestamp string as nanoseconds since Unix epoch.
    nanoseconds = numpy.datetime64(timestamp_str, 'ns').item()
```



```

time_in_seconds = math.floor(nanoseconds / 1E9)
# Slice to avoid precision issues.
offset_in_nanos = int(str(nanoseconds)[-9:])
return {
    'timeInSeconds': time_in_seconds,
    'offsetInNanos': offset_in_nanos
}

```

此 Lambda 函數會使用 NumPy 中的 [datetime64](#)，以 [ISO 8601](#) 格式輸入時間戳記字串。

Note

如果您的時間戳記字串不是 ISO 8601 格式，您可以使用實作 pandas 定義時間戳記格式的解決方案。如需詳細資訊，請參閱 [pandas.to_datetime](#)。

- 當您為規則設定 AWS IoT SiteWise 動作時，請針對以秒為單位的時間 (timeInSeconds) 和以奈米為單位的位移 ()，使用下列替代範本 offsetInNanos。這些替代範本會假設您的訊息承載包含 timestamp 中的時間戳記字串。aws_lambda 函數會針對其秒數參數取用 JSON 結構，因此您可視需要修改以下替代範本。

- 針對 Time in seconds (以秒為單位的時間) (timeInSeconds)，使用以下替代範本。

```

${aws_lambda('arn:aws:lambda:region:account-id:function:ConvertNanosecondTimestampFromString', {'timestamp': timestamp}).timeInSeconds}

```

- 針對 Offset in nanos (以奈米為單位的位移) (offsetInNanos)，使用以下替代範本。

```

${aws_lambda('arn:aws:lambda:region:account-id:function:ConvertNanosecondTimestampFromString', {'timestamp': timestamp}).offsetInNanos}

```

對於每個參數，請將 **region** 和 **account-id** 取代為您的區域和 AWS 帳戶 ID。如果您為 Lambda 函數使用不同的名稱，請一併變更。

- 授予使用 AWS IoT 許可叫用函數的 lambda:InvokeFunction 許可。如需詳細資訊，請參閱 [aws_lambda\(functionArn, inputJson\)](#)。
- 測試您的規則（例如，使用 AWS IoT MQTT 測試用戶端），並確認 AWS IoT SiteWise 接收您傳送的資料。

如果規則無法正常運作，請參閱 [對 AWS IoT SiteWise 規則動作進行故障診斷](#)。

Note

此解決方案會為每個時間戳記字串調用 Lambda 函數兩次。如果您的規則處理每個承載中具有相同時間戳記的多個資料點，您可以建立另一個規則來減少 Lambda 函數叫用的數量。若要這樣做，請使用重新發佈動作來建立規則，該動作會叫用 Lambda，並使用轉換為 `timeInSeconds` 和 的時間戳記字串來發佈原始承載 `offsetInNanos`。然後，使用規則動作建立 AWS IoT SiteWise 規則，以使用轉換後的承載。透過此方法，您可以減少規則叫用 Lambda 的次數，但會增加 AWS IoT 規則動作執行的次數。如果您要將此解決方案套用到您的使用案例，請考慮定價。

範例規則組態

本節包含使用 AWS IoT SiteWise 動作建立規則的範例規則組態。

Example 使用屬性別名做為訊息主題的範例規則動作

下列範例會建立具有 AWS IoT SiteWise 動作的規則，該動作使用 主題（透過 `topic()`）做為屬性別名來識別資產屬性。使用此範例來定義一個規則，以將雙類型資料擷取至所有風力發電廠中的所有風力發電機。此範例需要您在所有渦輪機資產的屬性上定義屬性別名。您需要定義第二個類似擷取整數類型資料的規則。

```
aws iot create-topic-rule \  
  --rule-name SiteWiseWindFarmRule \  
  --topic-rule-payload file://sitewise-rule-payload.json
```

`sitewise-rule-payload.json` 中的範例裝載包含以下內容。

```
{  
  "sql": "SELECT * FROM '/company/windfarm/+/turbine/+/+' WHERE type = 'double'",  
  "description": "Sends data to the wind turbine asset property with the same alias as  
the topic",  
  "ruleDisabled": false,  
  "awsIotSqlVersion": "2016-03-23",  
  "actions": [  
    {
```

```

    "iotSiteWise": {
      "putAssetPropertyValueEntries": [
        {
          "propertyAlias": "${topic()}",
          "propertyValues": [
            {
              "timestamp": {
                "timeInSeconds": "${timeInSeconds}"
              },
              "value": {
                "doubleValue": "${value}"
              }
            }
          ]
        }
      ],
      "roleArn": "arn:aws:iam::account-id:role/MySiteWiseActionRole"
    }
  }
}

```

使用此規則動作，將下列訊息傳送至風力渦輪機屬性別名（例如 /company/windfarm/3/turbine/7/temperature），做為擷取資料的主題。

```

{
  "type": "double",
  "value": "38.3",
  "timeInSeconds": "1581368533"
}

```

Example 使用 timestamp() 定義時間的範例規則動作

下列範例會建立具有 AWS IoT SiteWise 動作的規則，該動作會依 IDs 識別資產屬性，並使用 [timestamp\(\)](#) 來判斷目前時間。

```

aws iot create-topic-rule \
  --rule-name SiteWiseAssetPropertyRule \
  --topic-rule-payload file://sitewise-rule-payload.json

```

sitewise-rule-payload.json 中的範例裝載包含以下內容。

```
{
  "sql": "SELECT * FROM 'my/asset/property/topic'",
  "description": "Sends device data to an asset property",
  "ruleDisabled": false,
  "awsIotSqlVersion": "2016-03-23",
  "actions": [
    {
      "iotSiteWise": {
        "putAssetPropertyValueEntries": [
          {
            "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
            "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
            "propertyValues": [
              {
                "timestamp": {
                  "timeInSeconds": "${floor(timestamp() / 1E3)}",
                  "offsetInNanos": "${(timestamp() % 1E3) * 1E6}"
                },
                "value": {
                  "doubleValue": "${value}"
                }
              }
            ]
          }
        ],
        "roleArn": "arn:aws:iam::account-id:role/MySiteWiseActionRole"
      }
    }
  ]
}
```

使用此規則動作，將下列訊息傳送至 my/asset/property/topic 以擷取資料。

```
{
  "type": "double",
  "value": "38.3"
}
```

針對 規則動作進行故障診斷

若要對 中的 AWS IoT SiteWise 規則動作進行故障診斷 AWS IoT Core，請設定 CloudWatch Logs 或設定規則的重新發佈錯誤動作。如需詳細資訊，請參閱 [對 AWS IoT SiteWise 規則動作進行故障診斷](#)。

在 中使用基本擷取降低成本 AWS IoT SiteWise

AWS IoT Core 提供稱為基本擷取的功能，您可以用來透過 傳送資料，AWS IoT Core 而不會產生 [AWS IoT 簡訊費用](#)。基本擷取可從擷取路徑移除發佈/訂閱訊息代理程式，以最佳化大量資料擷取工作負載的資料流程。如果您知道應將訊息路由到哪些規則，您可以使用基本擷取。

若要使用基本擷取，請使用 `$aws/rules/rule-name` 特殊主題，將訊息直接傳送到特定規則。例如，若要將訊息傳送到名為 SiteWiseWindFarmRule 的規則，請將訊息傳送到 `$aws/rules/SiteWiseWindFarmRule` 主題。

如果規則動作使用包含 [topic\(Decimal\)](#) 的替代範本，您可以在基本擷取特殊主題的結尾處 (例如 `$aws/rules/rule-name/original-topic`) 傳遞原始主題。例如，若要使用基本擷取搭配上一節中的風力發電廠屬性別名範例，您可以傳送訊息到下列主題。

```
$aws/rules/SiteWiseWindFarmRule//company/windfarm/3/turbine/7/temperature
```

Note

上述範例包含第二個斜線 (//)，因為 會從規則動作可見的主題 AWS IoT 中移除基本擷取字首 (`$aws/rules/rule-name/`)。在此範例中，規則會收到 `/company/windfarm/3/turbine/7/temperature` 主題。

如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的 [使用基本擷取降低簡訊成本](#)。

AWS IoT SiteWise 從 擷取資料至 AWS IoT Events

透過 AWS IoT Events，您可以在 AWS 雲端中為 IoT 機群建置複雜的事件監控應用程式。使用 中的 IoT SiteWise 動作 AWS IoT Events，在事件發生 AWS IoT SiteWise 時將資料傳送至 中的資產屬性。

Note

終止支援通知：將於 2026 AWS 年 5 月 20 日結束對 的支援 AWS IoT Events。2026 年 5 月 20 日之後，您將無法再存取 AWS IoT Events 主控台或 AWS IoT Events 資源。如需詳細資訊，請參閱 [AWS IoT Events 終止支援](#)。

AWS IoT Events 旨在簡化 AWS 雲端內 IoT 裝置和系統的事件監控應用程式的開發。使用 AWS IoT Events，您可以：

- 偵測並回應 IoT 機群的變更、異常或特定條件。
- 提升您的營運效率，並主動管理 IoT 生態系統。

透過 AWS IoT SiteWise 動作與整合，AWS IoT Events 擴展其功能，可讓您自動更新中的資產屬性 AWS IoT SiteWise 以回應特定事件。此互動可以簡化資料擷取和管理。它也可以為您提供可行的洞見。

如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的下列主題：

- [什麼是 AWS IoT Events？](#)
- [AWS IoT Events 動作](#)
- [IoT SiteWise 動作](#)

在 AWS IoT Greengrass 中使用串流管理員 AWS IoT SiteWise

AWS IoT Greengrass 串流管理員是一種整合功能，可促進將資料串流從本機來源傳輸到 AWS 雲端。它充當管理資料流程的中介層，讓在邊緣操作的裝置能夠在資料傳送到之前收集和存放資料 AWS IoT SiteWise，以供進一步分析和處理。

在 AWS IoT SiteWise 主控台上設定本機來源，以新增資料目的地。您也可以在自訂 AWS IoT Greengrass 解決方案中使用串流管理員，將資料擷取至 AWS IoT SiteWise。

Note

若要從 OPC UA 來源擷取資料，請設定在 上執行的 AWS IoT SiteWise Edge 閘道 AWS IoT Greengrass。如需詳細資訊，請參閱[使用 AWS IoT SiteWise Edge 閘道](#)。

如需如何設定本機來源資料目的地的詳細資訊，請參閱 [了解 AWS IoT SiteWise Edge 目的地](#)。

如需如何在自訂 AWS IoT Greengrass 解決方案中使用串流管理員擷取資料的詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的下列主題：

- [什麼是 AWS IoT Greengrass？](#)
- [管理核心上的 AWS IoT Greengrass 資料串流](#)
- [將資料匯出至 AWS IoT SiteWise 資產屬性](#)

使用 AWS IoT SiteWise Edge 閘道

AWS IoT SiteWise Edge 將雲端功能擴展到工業邊緣環境，實現本機資料處理、分析和決策。SiteWise Edge 與 AWS IoT SiteWise 和其他 AWS 服務整合，以提供全方位的工業 IoT 解決方案。閘道可做為工業設備與之間的媒介 AWS IoT SiteWise。

SiteWise Edge 閘道會在兩個不同的部署目標上執行：

- AWS IoT Greengrass V2
- Siemens Industrial Edge

您可以使用 SiteWise Edge 閘道在邊緣收集資料，並將其發佈至雲端。對於在上執行的閘道 AWS IoT Greengrass，您也可以使用資產模型和資產處理邊緣的資料。

上的 AWS IoT SiteWise Edge 應用程式 Siemens Industrial Edge 支援工業設備與之間的整合，AWS IoT SiteWise 讓您可以彙總和處理原始機器資料，並在本機執行分析，然後再將精簡的資料傳送至 AWS 雲端。

SiteWise Edge 閘道的關鍵概念

SiteWise Edge 有數個有用的功能，可用於工業環境中的邊緣運算。

本機資料收集和處理

支援使用 OPC-UA 和 MQTT 等通訊協定從工業資產收集資料。閘道在 AWS IoT Greengrass Core 裝置或上執行 Siemens Industrial Edge。

離線操作

在網際網路中斷期間繼續收集和處理資料，並在恢復連線時與雲端同步。

含 AWS IoT Greengrass 元件的邊緣運算

使用 IoT SiteWise 發佈者將資料轉送至雲端和 AWS IoT SiteWise 處理器，以進行本機轉換和計算。發佈者和處理器都是 AWS IoT Greengrass V2 元件。如需 AWS IoT Greengrass 元件的詳細資訊，請參閱 [AWS提供的元件](#)。

與整合 AWS IoT SiteWise 以擴展雲端功能

使用 AWS IoT SiteWise 雲端功能，將資產模型、分析和儀表板擴展到邊緣。

對於已啟用資料處理套件的閘道，您可以使用 AWS OpsHub 的 AWS IoT SiteWise 來集中管理 SiteWise Edge 閘道。AWS OpsHub 提供遠端管理和監控功能。如需詳細資訊，請參閱[使用 AWS OpsHub for 管理 SiteWise Edge 閘道 AWS IoT SiteWise](#)。

合作夥伴資料來源整合

將合作夥伴資料來源連接至您的閘道，並從 SiteWise Edge 閘道和 AWS 雲端中的合作夥伴接收資料。如需詳細資訊，請參閱[SiteWise Edge 閘道上的合作夥伴資料來源](#)。

邊緣上的本機視覺化

提供邊緣即時洞見的自訂儀表板。

在本機裝置上使用 SiteWise Monitor 入口網站在設施中本機監控資料。如需詳細資訊，請參閱[在邊緣啟用您的 AWS IoT SiteWise 入口網站](#)。

實作 SiteWise Edge 的優勢

SiteWise Edge 提供多種優勢，可大幅改善工業營運和決策程序。

- 即時營運洞見，不會延遲雲端處理
- 中斷連線環境中的操作持續性
- 透過邊緣預先處理降低頻寬和儲存成本
- 提高可靠性，並能夠做出本機、資料驅動的決策

使用 自我託管 AWS IoT SiteWise Edge 閘道 AWS IoT Greengrass V2

設定 AWS IoT SiteWise Edge 以在本機收集、處理和視覺化工業設備的資料，然後再傳送至雲端。使用 自行託管 AWS IoT Greengrass Version 2。

AWS IoT SiteWise Edge 閘道可做為工業設備和 之間的媒介 AWS IoT SiteWise。在 上執行時 AWS IoT Greengrass Version 2，SiteWise Edge 閘道支援現場部署的資料收集和處理。在已啟用並 AWS OpsHub 安裝資料處理套件的情況下，透過本機裝置上的 SiteWise Monitor 入口網站在設施內本機監控資料。

自我託管閘道有兩種類型：

啟用 MQTT 的 V3 閘道

啟用 MQTT 的 V3 閘道架構可改善資料擷取功能。它利用 MQTT 通訊協定進行有效率的資料通訊，並提供可設定的資料目的地。其中包括使用 Amazon S3 緩衝資料擷取的選項，以及即時資料擷取。您可以實作路徑篩選條件來訂閱特定 MQTT 主題，以啟用目標資料收集。請注意，已啟用 MQTT 的 V3 閘道不支援 Data Processing Pack 功能。如需詳細資訊，請參閱[啟用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge](#)。

Classic 串流、V2 閘道

Classic 串流 V2 閘道代表傳統的 AWS IoT SiteWise Edge 閘道架構。它非常適合現有的 SiteWise Edge 部署和熟悉已建立工作流程的使用者。雖然 Classic 串流、V2 閘道支援資料處理套件，但請注意，資料處理套件產生的資料無法透過 Amazon S3 擷取。如果您需要維持與現有部署的相容性，或需要資料處理套件功能，請使用 Classic 串流 V2 閘道。如需詳細資訊，請參閱[傳統串流、適用於 AWS IoT SiteWise Edge 的 V2 閘道](#)。

主題

- [AWS IoT SiteWise Edge 自我託管閘道需求](#)
- [建立自我託管 SiteWise Edge 閘道](#)
- [在本機裝置上安裝 AWS IoT SiteWise Edge 閘道軟體](#)
- [啟用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge](#)
- [傳統串流、適用於 AWS IoT SiteWise Edge 的 V2 閘道](#)
- [將資料來源新增至 AWS IoT SiteWise Edge 閘道](#)
- [AWS IoT GreengrassAWS IoT SiteWise Edge 的 元件](#)
- [在 SiteWise Edge 閘道上篩選資產](#)
- [設定 AWS IoT SiteWise Edge 的代理支援和管理信任存放區](#)
- [邊緣上的 AWS IoT SiteWise APIs](#)

AWS IoT SiteWise Edge 自我託管閘道需求

AWS IoT SiteWise 邊緣閘道以一組元件 AWS IoT Greengrass V2 的形式在 上執行，這些 AWS IoT Greengrass 元件支援現場部署的資料收集、處理和發佈。若要設定在 上執行的 SiteWise Edge 閘道 AWS IoT Greengrass V2，請在 中建立閘道，AWS 雲端 並執行 SiteWise Edge 閘道軟體來設定本機裝置。當您使用 AWS Management Console 建立 SiteWise Edge 閘道時，會提供安裝指令碼。在目標閘道裝置上執行此指令碼，以設定必要的軟體和相依性。

本機裝置需求

本機裝置必須符合下列要求，才能安裝和執行 SiteWise Edge 閘道軟體。

- 支援 AWS IoT Greengrass V2 核心軟體版本 [v2.3.0](#) 或更新版本。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的[需求](#)。
- 下列其中一個支援的平台：
 - 作業系統：Ubuntu 20.04 或更新版本
架構：x86_64 (AMD64) 或 ARMv8 (Aarch64)
 - 作業系統：Red Hat Enterprise Linux (RHEL) 8
架構：x86_64 (AMD64) 或 ARMv8 (Aarch64)
 - 作業系統：Amazon Linux 2
架構：x86_64 (AMD64) 或 ARMv8 (Aarch64)
 - 作業系統：Debian 11
架構：x86_64 (AMD64) 或 ARMv8 (Aarch64)
 - 作業系統：Windows Server 2019 及更新版本
架構：x86_64 (AMD64)

Note

ARM 平台僅支援具有 Data Collection Pack 的 SiteWise Edge 閘道。不支援資料處理套件。

- 最低 4 GB RAM。
- SiteWise Edge 閘道軟體的最低可用磁碟空間為 10 GB。
- 設定您的本機裝置，以確保可存取適當的連接埠。如需所需傳出服務端點的完整清單，請參閱 [AWS IoT SiteWise Edge 閘道所需的服務端點](#)。
- Java 執行期環境 (JRE) 第 11 版或更新版本。Java 必須在裝置上的 PATH 環境變數上可用。若要使用 Java 開發自訂元件，您必須安裝 Java 開發套件 (JDK)。我們建議您使用 [Amazon Corretto](#) 或 [OpenJDK](#)。

要允許列出本機裝置的 Amazon S3 儲存貯體

設定您的本機裝置以提供防火牆存取下列 Amazon S3 儲存貯體。根據裝置的個別區域設定存取權。

區域	端點
亞太區域 (東京)	https://iot-sitewise-gateway-ap-northeast-1-785558802005.s3.ap-northeast-1.amazonaws.com
亞太區域 (首爾)	https://iot-sitewise-gateway-ap-northeast-2-310055672453.s3.ap-northeast-2.amazonaws.com
亞太區域 (孟買)	https://iot-sitewise-gateway-ap-south-1-677656657204.s3.ap-south-1.amazonaws.com
亞太區域 (新加坡)	https://iot-sitewise-gateway-ap-southeast-1-475191558554.s3.ap-southeast-1.amazonaws.com
亞太區域 (悉尼)	https://iot-sitewise-gateway-ap-southeast-2-396319432685.s3.ap-southeast-2.amazonaws.com
加拿大 (中部)	https://iot-sitewise-gateway-ca-central-1-842060018567.s3.ca-central-1.amazonaws.com
中國 (北京)	https://iot-sitewise-gateway-cn-north-1-237124890262.s3.cn-north-1.amazonaws.com.cn
歐洲 (法蘭克福)	https://iot-sitewise-gateway-eu-central-1-748875242063.s3.eu-central-1.amazonaws.com
歐洲 (愛爾蘭)	https://iot-sitewise-gateway-eu-west-1-383414315062.s3.eu-west-1.amazonaws.com
美國東部 (維吉尼亞北部)	https://iot-sitewise-gateway-us-east-1-223558168232.s3.us-east-1.amazonaws.com 和 https://iot-sitewise-gateway-us-east-1-223558168232.s3.amazonaws.com/
美國東部 (俄亥俄)	https://iot-sitewise-gateway-us-east-2-005072661813.s3.us-east-2.amazonaws.com
AWS GovCloud (美國西部)	https://iot-sitewise-gateway-us-gov-west-1-599984565679.s3.us-gov-west-1.amazonaws.com/

區域	端點
美國西部 (奧勒岡)	https://iot-sitewise-gateway-us-west-2-502577205460.s3.us-west-2.amazonaws.com

資料處理套件需求

- 如果您計劃使用邊緣的資料處理套件 AWS IoT SiteWise，您的本機裝置也必須符合下列要求：
 - 具有 x86 64 位元四核心處理器。
 - 至少有 16 GB 的 RAM。
 - 如果使用，RAM 至少要有 32 GB Microsoft Windows。
 - 至少有 256 GB 的可用磁碟空間。
 - 本機裝置必須允許連接埠 443 的網路傳入流量。
 - 下列連接埠保留供使用 AWS IoT SiteWise：
 - 80、443、3001、4569、4572、8000、8081、8082、8084、8085、8445、8086、9000、9500、11080 和 50010。使用保留連接埠處理流量可能會導致連線終止。

Note

AWS IoT Greengrass V2 串流管理員元件有自己的需求。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的[組態](#)。

- 最小磁碟空間和運算容量需求取決於實作和使用案例特有的各種因素。
 - 在間歇性網際網路連線狀態下，快取資料所需的磁碟空間取決於下列因素：
 - 上傳的資料串流數目
 - 每個資料串流每秒的資料點
 - 每個資料點的大小
 - 通訊速度
 - 預期的網路停機時間
 - 輪詢和上傳資料所需的運算容量取決於下列因素：
 - 上傳的資料串流數目
 - 每個資料串流每秒的資料點

設定許可以使用 SiteWise Edge 閘道

您必須具有下列許可才能使用 SiteWise Edge 閘道：

Note

如果您使用 AWS IoT SiteWise 主控台建立 SiteWise Edge 閘道，則會為您新增這些許可。

- SiteWise Edge 閘道的 IAM 角色必須允許您在 AWS IoT Greengrass V2 裝置上使用 SiteWise Edge 閘道來處理資產模型資料和資產資料。

角色允許下列服務擔任該角色：`credentials.iot.amazonaws.com`。

許可詳細資訊

角色必須具有下列許可：

- `iotsitewise` – 允許主體擷取邊緣的資產模型資料和資產資料。
- `iot` – 允許 AWS IoT Greengrass V2 您的裝置與 互動 AWS IoT。
- `logs` – 允許 AWS IoT Greengrass V2 您的裝置將日誌傳送至 Amazon CloudWatch Logs。
- `s3` – 允許 AWS IoT Greengrass V2 您的裝置從 Amazon S3 下載自訂元件成品。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:BatchPutAssetPropertyValue",
        "iotsitewise:List*",
        "iotsitewise:Describe*",
        "iotsitewise:Get*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeCertificate",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
```

```

        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation",
        "s3:GetObject",
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:DescribeEndpoint"
    ],
    "Resource": "*"
}
]
}

```

建立自我託管 SiteWise Edge 閘道

使用 AWS IoT SiteWise 主控台或 AWS CLI 建立自我託管 SiteWise Edge 閘道。此程序詳細說明如何建立將安裝在自有硬體上的自我託管 SiteWise Edge 閘道。如需建立在 Siemens Industrial Edge 上執行的 SiteWise Edge 閘道的詳細資訊，請參閱 [在上託管 SiteWise Edge 閘道 Siemens Industrial Edge](#)。

建立 SiteWise Edge 閘道

Console

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇 Create gateway (建立閘道)。
4. 針對選擇部署目標，選擇自我託管閘道。
5. 選取啟用 MQTT 的 V3 閘道或 Classic 串流 V2 閘道。如需每個選項的詳細資訊，請參閱 [使用自我託管 AWS IoT SiteWise Edge 閘道 AWS IoT Greengrass V2](#)。建議使用啟用 MQTT 的 V3 閘道做為未來就緒功能。
6. 在閘道組態區段中，輸入 SiteWise Edge 閘道的名稱，或使用產生的名稱 AWS IoT SiteWise。
7. 在 Greengrass 裝置作業系統下，選取您要安裝此 SiteWise Edge 閘道之裝置的作業系統。

 Note

資料處理套件僅適用於 x86 平台。它僅適用於 Classic 串流、V2 閘道

8. (選用) 若要在邊緣處理和組織資料，請在邊緣功能下選取資料處理套件。

 Note

若要授予您公司目錄中使用者群組存取此 SiteWise Edge 閘道的權限，請參閱 [在 SiteWise Edge 中設定邊緣功能](#)

9. (選用) 在進階組態下，執行下列動作：

- 針對 Greengrass 核心裝置，選擇下列其中一個選項：
 - 預設設定 – AWS 自動使用預設設定在其中建立 Greengrass 核心裝置 AWS IoT Greengrass V2。
 - 1. 輸入 Greengrass 核心裝置的名稱，或使用 產生的名稱 AWS IoT SiteWise。
 - 進階設定 – 如果您想要使用現有的 Greengrass 核心裝置或手動建立，請選擇此選項。
 - 1. 選擇 Greengrass 核心裝置，或選擇建立 Greengrass 核心裝置以在 AWS IoT Greengrass V2 主控台中建立。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的 [設定 AWS IoT Greengrass V2 核心裝置](#)。

10. 選擇 Create gateway (建立閘道)。

11. 在產生 SiteWise Edge 閘道安裝程式對話方塊中，選擇產生和下載。AWS IoT SiteWise 會自動產生安裝程式，供您用來設定本機裝置。

 Important

您無法重新產生此檔案。請務必將安裝程式檔案儲存在安全的位置，因為您稍後將使用檔案。

AWS CLI

若要使用 建立自我託管閘道 AWS CLI，請提供閘道的名稱、指定平台和閘道版本。您可以在建立閘道時指定許多其他選項。如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [create-gateway](#)。

AWS IoT SiteWise

若要使用此範例，請以您自己的資訊取代使用者輸入預留位置。

```
aws iotsitewise create-gateway \  
  --gateway-name your-gateway-name \  
  --gateway-platform greengrassV2={coreDeviceThingName=your-core-device-thing-name, coreDeviceOperatingSystem=LINUX_AMD64} \  
  --gateway-version 3 \  
  [--cli-input-json your-configuration]
```

- gateway-name – 閘道的唯一名稱。
- gateway-platform – 指定閘道平台組態。對於自我託管閘道，輸入 greengrassV2。如需詳細資訊，請參閱 AWS CLI 命令參考之 create-gateway 區段中的[選項](#) AWS IoT SiteWise。
 - coreDeviceThingName – AWS IoT Greengrass V2 核心裝置的 AWS IoT 物件名稱。
 - coreDeviceOperatingSystem – 核心裝置的作業系統 AWS IoT Greengrass V2。需要指定作業系統，gateway-version 3 不適用於 gateway-version 2。選項包括：LINUX_AMD64、LINUX_AARCH64 和 WINDOWS_AMD64
- gateway-version – 閘道的版本。
 - 若要建立已啟用 MQTT 的 V3 閘道，請將 3 用於閘道版本。
 - 若要建立 Classic 串流、V2 閘道，請將 2 用於閘道版本。
- cli-input-json – 包含請求參數的 JSON 檔案。

現在您已建立 SiteWise Edge 閘道 [在本機裝置上安裝 AWS IoT SiteWise Edge 閘道軟體](#)。

在本機裝置上安裝 AWS IoT SiteWise Edge 閘道軟體

建立 AWS IoT SiteWise Edge 閘道之後，請在本機裝置上安裝 SiteWise Edge 閘道軟體。SiteWise Edge 閘道軟體可以安裝在已安裝 Linux 或 Microsoft Windows 伺服器作業系統的本機裝置上。

Important

請確定您的本機裝置連線至網際網路。

Linux

下列程序使用 SSH 連線到您的本機裝置。或者，您可以使用 USB 快閃磁碟機或其他工具，將安裝程式檔案傳輸至本機裝置。如果您不想使用 SSH，請跳到下面的步驟 2：安裝 SiteWise Edge 閘道軟體。

SSH 先決條件

使用 SSH 連線至裝置之前，請先完成下列先決條件。

- Linux 和 macOS - 下載並安裝 OpenSSH。如需詳細資訊，請參閱 <https://www.openssh.com>。

步驟 1：將安裝程式複製到 SiteWise Edge 閘道裝置

下列指示說明如何使用 SSH 用戶端連線至本機裝置。

- 若要連線至您的裝置，請在電腦上的終端機視窗中執行下列命令，以具有更高權限和 *IP* 地址的使用者名稱取代 *#####* 和 IP。

```
ssh username@IP
```

- 若要將 AWS IoT SiteWise 產生的安裝程式檔案傳輸至 SiteWise Edge 閘道裝置，請執行下列命令。

Note

- 將 *path-to-saved-installer* 取代為您用來儲存安裝程式檔案和安裝程式檔案名稱之電腦上的路徑。
- 將 *IP ##* 取代為您本機裝置的 IP 地址。
- 將 *directory-to-receive-installer* 取代為您用來接收安裝程式檔案之本機裝置上的路徑。

```
scp path-to-saved-installer.sh user-name@IP-address:directory-to-receive-installer
```

步驟 2：安裝 SiteWise Edge 閘道軟體

在下列程序中，在 SiteWise Edge 閘道裝置上的終端機視窗中執行命令。

1. 給予安裝程式檔案執行許可。

```
chmod +x path-to-installer.sh
```

2. 執行安裝程式。

```
sudo ./path-to-installer.sh
```

Windows Server

先決條件

您必須具備下列先決條件，才能安裝 SiteWise Edge 閘道軟體：

- Microsoft Windows Server 2019 或更新版本已安裝
- 管理員權限
- 已安裝 PowerShell 5.1 版或更新版本
- SiteWise Edge 閘道安裝程式下載至要佈建的 Windows Server

步驟 1：以管理員身分執行 PowerShell

1. 在您要安裝 SiteWise Edge 閘道的 Windows 伺服器上，以管理員身分登入。
2. 在 Windows 搜尋列中輸入 PowerShell。
3. 在搜尋結果中，開啟 Windows PowerShell 應用程式上的內容（按一下滑鼠右鍵）選單。選擇以管理員身分執行。

步驟 2：安裝 SiteWise Edge 閘道軟體

在 SiteWise Edge Gateway 裝置上的終端機視窗中執行下列命令。

1. 解除封鎖 SiteWise Edge 閘道安裝程式。

```
unblock-file path-to-installer.ps1
```

2. 執行安裝程式。

```
./path-to-installer.ps1
```

Note

如果系統已停用指令碼執行，請將指令碼執行政策變更為 RemoteSigned。

```
Set-ExecutionPolicy RemoteSigned
```

下一個步驟取決於您需要的自我託管閘道類型。繼續前往 [啟用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge](#) 或 [傳統串流、適用於 AWS IoT SiteWise Edge 的 V2 閘道](#)。

啟用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge

AWS IoT SiteWise 可以使用啟用 MQTT 的 V3 閘道，代表 SiteWise Edge 閘道架構的重大進展。此閘道類型利用 MQTT（訊息佇列遙測傳輸）通訊協定進行資料通訊，在工業 IoT 部署中提供增強的靈活性和效率。

啟用 MQTT 的 V3 閘道使用 MQTT 進行資料傳輸，啟用輕量型的發佈訂閱網路通訊協定，可在裝置和雲端之間有效率地傳輸訊息。您可以設定各種資料目的地，包括使用 Amazon S3 直接將即時資料擷取至 AWS IoT SiteWise 和緩衝資料擷取。若要啟用精確的資料收集，您可以實作路徑篩選條件來訂閱特定的 MQTT 主題。

啟用 MQTT 的 V3 閘道隨附預先設定的即時目的地，並將篩選條件設定為 "#"（所有主題），您可以視需要自訂或移除。為了簡化資料管理，每個閘道中只能存在一個即時目的地。

啟用 MQTT 的架構與 Classic 串流 V2 閘道明顯不同。雖然 V2 使用以串流為基礎的方法，但 V3 採用 MQTT，提供更可設定的資料目的地和篩選選項。不過，請注意，V3 不支援資料處理套件，該套件可在 V2 中使用。

啟用 MQTT 的 V3 閘道提供數種優點：

- 由於 MQTT 輕量型本質，改善了可擴展性，可更妥善處理許多裝置和高頻率資料傳輸。
- 透過路徑篩選條件增強資料控制，可精細管理資料收集並減少不必要的資料傳輸和處理。
- 靈活的資料處理，允許根據特定需求進行即時處理和緩衝儲存之間的組態。
- 符合現代 IoT 通訊標準，為未來的增強功能和整合做好準備。

考慮為新部署採用啟用 MQTT 的 V3 閘道，尤其是當您需要靈活的資料擷取選項和對資料收集的精確控制時。

Note

對於需要資料處理套件的現有部署或案例，Classic 串流、V2 閘道仍然是可行的選項。

透過提供這兩種閘道類型，AWS IoT SiteWise 確保您可以選擇最符合您特定工業 IoT 需求的解決方案，無論您是優先考慮進階 MQTT 功能還是與現有系統的相容性。

目的地和路徑篩選條件

檢視下列主題，進一步了解啟用 MQTT 閘道中的目的地和路徑篩選條件：

- [了解 AWS IoT SiteWise Edge 目的地](#)
- [新增 AWS IoT SiteWise Edge 即時目的地](#)
- [使用 Amazon S3 新增 AWS IoT SiteWise 緩衝目的地](#)
- [了解 AWS IoT SiteWise Edge 目的地的路徑篩選條件](#)
- [將路徑篩選條件新增至 AWS IoT SiteWise Edge 目的地](#)
- [管理 AWS IoT SiteWise Edge 目的地](#)

將外部應用程式連接至 EMQX 代理程式

本指南說明如何透過部署已啟用 MQTT 的 V3 閘道上的 EMQX 代理程式，將外部應用程式連接至 AWS IoT SiteWise Edge 閘道。外部應用程式可能包括自訂監控工具、第三方視覺化軟體，或需要與邊緣工業資料互動的舊版系統。

我們將涵蓋 Linux 和 Microsoft Windows 環境的組態步驟，包括 EMQX 部署組態、安全連線的 TLS 設定，以及控制特定主題存取的授權規則。

Note

EMQX 不是 AWS IoT SiteWise Edge 的廠商或供應商。

Important

為了保護您的閘道連線，我們強烈建議透過 AWS IoT Greengrass 用戶端裝置身分驗證功能使用憑證型身分驗證。此方法透過交互 TLS (mTLS) 身分驗證提供強大的安全性。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的[將用戶端裝置連接至核心裝置](#)。

如果您無法使用憑證型身分驗證，請依照本指南使用使用者名稱和密碼設定身分驗證。

先決條件

- 已啟用 SiteWise Edge MQTT 的 V3 閘道，已部署且在線上
- 存取閘道主機
- 存取 AWS IoT SiteWise 和 AWS IoT Greengrass 主控台

主題

- [AWS IoT SiteWise Edge 上的 EMQX 代理程式的訊息承載格式](#)
- [設定 EMQX 代理程式](#)
- [將應用程式連接到 AWS IoT SiteWise Edge 上的 EMQX 代理程式](#)
- [在 EMQX 中設定 AWS IoT SiteWise Edge 的授權規則](#)

AWS IoT SiteWise Edge 上的 EMQX 代理程式的訊息承載格式

若要讓 IoT SiteWise 發佈者元件取用外部應用程式的資料並將其發佈至 AWS IoT SiteWise 雲端，傳送給代理程式的承載必須符合特定要求。

了解承載格式是與 AWS IoT SiteWise Edge 成功進行 MQTT 通訊的關鍵。雖然稍後章節涵蓋連線設定程序，但我們會先提供承載需求，協助您規劃實作。

MQTT 主題需求

MQTT 主題結構沒有限制，包括使用的關卡或字元數。不過，我們建議主題符合承載中的 `propertyAlias` 欄位。

Example 屬性別名範例

如果 MQTT 主題是 `site1/line1/compressor1/temperature`，請確保 `propertyAlias` 相符。

```
{
  "assetId": "compressor_asset_01",
  "propertyAlias": "site1/line1/compressor1/temperature",
  "propertyId": "temperature_sensor_01",
  "propertyValues": [
    {
      "quality": "GOOD",
      "timestamp": {
        "offsetInNanos": 0,
        "timeInSeconds": 1683000000
      },
      "value": {
        "doubleValue": 23.5
      }
    }
  ]
}
```

JSON 承載結構

MQTT 訊息承載是以 JSON 撰寫，並遵循 [AWS IoT SiteWise API 參考](#) 中定義的 PutAssetPropertyValueEntry 訊息格式。

```
{
  "assetId": "string",
  "propertyAlias": "string",
  "propertyId": "string",
  "propertyValues": [
    {
      "quality": "string",
      "timestamp": {
        "offsetInNanos": number,
        "timeInSeconds": number
      },
      "value": {
        "booleanValue": boolean,
        "doubleValue": number,
        "integerValue": number,
        "stringValue": "string"
      }
    }
  ]
}
```

```
}
```

Note

若要讓訊息視為有效，只有下列其中一個條件可以是 true：

- `propertyAlias` 已設定，或
- `assetId` 和 `propertyId` 都已設定

`PutAssetPropertyValueEntry` 具有在此內容中不需要 `entryId` 的欄位。

設定 EMQX 代理程式

本節說明如何新增使用者名稱和密碼。它還涵蓋如何使用新增的使用者名稱和密碼從外部來源建立 TLS 連線。您可以使用 Linux 或 設定 EMQX 代理程式 Microsoft Windows。

Note

若要設定代理程式，您需要在啟用 MQTT 的 V3 閘道中使用預設 EMQX 組態設定的核心裝置。

Important

完成此程序後，強烈建議您設定授權規則。如需詳細資訊，請參閱[在 EMQX 中設定 AWS IoT SiteWise Edge 的授權規則](#)。新增使用者的授權規則可增強安全性。

更新 EMQX 部署組態以進行身分驗證

更新 EMQX 部署組態以進行身分驗證

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選擇要設定的閘道。
4. 在 Edge 閘道組態區段中，複製您的 Greengrass 核心裝置值。儲存它以供日後使用。
5. 開啟 [AWS IoT 主控台](#)。

6. 在左側導覽的管理區段下，選擇 Greengrass 裝置，然後選擇部署。
7. 尋找您先前儲存的核心裝置值，然後選擇該連結以開啟部署。
8. 選擇動作下拉式選單按鈕，然後修訂。
9. 閱讀出現的訊息，然後選擇修訂部署。指定目標頁面隨即出現。
10. 選擇下一步，直到您到達設定元件步驟為止。
11. 選取aws.greengrass.clientdevices.mqtt.EMQX選項按鈕。
12. 選擇設定元件按鈕。元件的組態頁面隨即出現。
13. 在組態更新下，選擇重設為元件版本的預設組態：2.*.*。
14. 根據您的作業系統，在要合併的組態區段中輸入下列組態。

Linux

```
{
  "emqxConfig": {
    "authorization": {
      "no_match": "allow"
    },
    "listeners": {
      "tcp": {
        "default": {
          "enabled": true,
          "enable_authn": false
        }
      },
      "ssl": {
        "default": {
          "enabled": true,
          "enable_authn": true,
          "ssl_options": {
            "verify": "verify_none",
            "fail_if_no_peer_cert": false
          }
        }
      }
    },
    "authentication": {
      "enable": true,
      "backend": "built_in_database",
      "mechanism": "password_based",
      "password_hash_algorithm": {
```



```

        "iterations": 210000,
        "mac_fun": "sha512",
        "name": "pbkdf2"
    },
    "user_id_type": "username"
},
"dashboard": {
    "listeners": {
        "http": {
            "bind": 18083
        }
    }
}
},
"authMode": "bypass",
"dockerOptions": "-p 8883:8883 -p 127.0.0.1:1883:1883\n-p 127.0.0.1:18083:18083 -v emqx-data:/opt/emqx/data -e\nEMQX_NODE__NAME=emqx@local",
"requiresPrivilege": "true"
}

```

Windows

```

{
  "emqxConfig": {
    "authorization": {
      "no_match": "allow"
    },
    "listeners": {
      "tcp": {
        "default": {
          "enabled": true,
          "enable_authn": false
        }
      },
      "ssl": {
        "default": {
          "enabled": true,
          "enable_authn": true,
          "ssl_options": {
            "verify": "verify_none",
            "fail_if_no_peer_cert": false
          }
        }
      }
    }
  }
}

```

```

        }
    },
    "authentication": {
        "enable": true,
        "backend": "built_in_database",
        "mechanism": "password_based",
        "password_hash_algorithm": {
            "iterations": 210000,
            "mac_fun": "sha512",
            "name": "pbkdf2"
        },
        "user_id_type": "username"
    },
    "dashboard": {
        "listeners": {
            "http": {
                "bind": 18083
            }
        }
    },
    "authMode": "bypass",
    "requiresPrivilege": "true"
}

```

dockerOptions 欄位僅適用於 Linux 閘道。

15. 選擇確認。
16. 選擇下一步，直到您到達檢閱步驟為止。
17. 選擇部署。
18. 部署成功後，請繼續下一個步驟。

啟用使用者名稱和密碼身分驗證

本節說明如何透過 EMQX 儀表板 GUI 新增使用者名稱和密碼。

Note

提供的 EMQX 相關指示僅供參考。由於 EMQX 文件和功能可能會隨著時間而變更，而且我們不會維護其文件，我們建議您參閱 [EMQX 的官方文件](#) 以取得最新資訊。

EMQX Dashboard

透過 EMQX 儀表板啟用使用者名稱和密碼身分驗證

1. 確定您位於閘道主機中。
2. 開啟瀏覽器視窗並造訪 <http://localhost:18083/>。
3. 輸入預設使用者名稱 **admin**和預設密碼 **public**。如需詳細資訊，請參閱 [EMQX 文件中的 EMQX 儀表板](#)。
4. 登入後，系統會提示您變更密碼。更新您的密碼以繼續前往 EMQX 儀表板。
5. 在左側導覽中，選擇遮罩圖示，然後選擇身分驗證。
6. 在內建資料庫列中，選擇使用者按鈕。
7. 選擇加號圖示按鈕以新增使用者。隨即出現新增畫面。
8. 輸入外部應用程式的 使用者的使用者名稱和密碼。
9. 選擇儲存。您選擇的使用者名稱會顯示在身分驗證頁面的資料表中。

Note

現有或預設的授權規則適用於新使用者。建議您檢閱並根據您的外部應用程式需求進行調整。

EMQX Management with Linux

使用位於的 AWS IoT SiteWise EMQX CLI 工具 `/greengrass/v2/bin/swe-emqx-cli`。

使用 Linux 透過 EMQX Management 啟用使用者名稱和密碼身分驗證

1. 執行下列命令來變更管理員密碼：

```
/greengrass/v2/bin/swe-emqx-cli admin change-pwd
```

2. 出現提示時，請執行下列動作：
 1. 輸入您目前的管理員使用者（預設為 `admin`）和密碼（預設為 `public`）。
 2. 輸入並確認新密碼。

如果成功，您會看到下列訊息：

```
admin password changed successfully
```

3. 執行下列命令，為外部應用程式新增使用者：

```
/greengrass/v2/bin/swe-emqx-cli users add
```

4. 出現提示時，請執行下列動作：

1. 輸入新使用者的使用者名稱。
2. 輸入並確認新使用者的密碼。

如果成功，您會看到下列訊息：

```
User '[username]' created successfully
```

5. 執行下列命令來驗證使用者組態：

```
/greengrass/v2/bin/swe-emqx-cli users list
```

輸出會顯示所有設定的使用者：

```
Users:  
- [your-added-username]  
  
Total users: 1
```

EMQX Management with Windows

在下列其中一個位置使用 AWS IoT SiteWise EMQX CLI 工具：

- PowerShell：C:\greengrass\v2\bin\swe-emqx-cli.ps1
- 命令提示字元：C:\greengrass\v2\bin\swe-emqx-cli.bat

使用 Windows 透過 EMQX Management 啟用使用者名稱和密碼身分驗證

1. 執行下列命令來變更管理員密碼：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 admin change-pwd
```

2. 出現提示時，請執行下列動作：

1. 輸入您目前的管理員使用者（預設為 admin）和密碼（預設為 public）。
2. 輸入並確認新密碼。

如果成功，您會看到下列訊息：

```
admin password changed successfully
```

3. 執行下列命令，為外部應用程式新增使用者：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 users add
```

4. 出現提示時，請執行下列動作：

1. 輸入新使用者的使用者名稱。
2. 輸入並確認新使用者的密碼。

如果成功，您會看到下列訊息：

```
User '[username]' created successfully
```

5. 執行下列命令來驗證使用者組態：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 users list
```

輸出會顯示所有設定的使用者：

```
Users:
- [your-added-username]

Total users: 1
```

將應用程式連接到 AWS IoT SiteWise Edge 上的 EMQX 代理程式

EMQX 代理程式會在連接埠 8883 上使用 Transport Layer Security (TLS) 來加密所有通訊，確保您的資料在傳輸期間仍受到保護。本節會逐步解說在應用程式與 EMQX 代理程式之間建立連線的步驟。遵循這些步驟有助於維護工業資料的完整性和機密性。連線程序涉及兩種主要方法：透過元件使用自動 IP 探索，或在 TLS 憑證中手動將 DNS 名稱和 IP 地址設定為主體別名 (SANs)。每個方法都有自己的優勢，取決於您的網路設定和安全需求。本文件將引導您完成這兩個選項。

主題

- [設定 TLS 以安全連線至 AWS IoT SiteWise Edge 上的 EMQX 代理程式](#)
- [在 AWS IoT SiteWise Edge 上測試 EMQX 代理程式連線](#)
- [使用您自己的 CA](#)
- [為外部防火牆連線開啟連接埠 8883](#)

設定 TLS 以安全連線至 AWS IoT SiteWise Edge 上的 EMQX 代理程式

根據預設，會為核心裝置憑證授權單位 (CA) 簽署的 EMQX 代理程式 AWS IoT Greengrass 產生 TLS 伺服器憑證。如需詳細資訊，請參閱[使用 MQTT 代理程式將用戶端裝置連接到 AWS IoT Greengrass 核心裝置](#)。

擷取 TLS 憑證

若要取得 CA 憑證，請在閘道主機上執行下列命令：

Linux

在閘道主機的 shell 工作階段中執行下列命令：

```
/greengrass/v2/bin/swe-emqx-cli cert
```

此命令會顯示憑證位置並列印憑證的內容。

或者，您也可以使用此命令將憑證儲存至檔案：

```
/greengrass/v2/bin/swe-emqx-cli cert --output /path/to/certificate.pem
```

Windows

在閘道主機的 PowerShell 工作階段中執行下列命令：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 cert
```

此命令會顯示憑證位置並列印憑證的內容。

或者，您也可以使用此命令將憑證儲存至檔案：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 cert --output C:\path\to\certificate.pem
```

無論系統上的確切路徑為何，CLI 都會自動尋找憑證。

將 ca.pem 檔案的內容複製到您要連線至代理程式的外部應用程式。另存為 BrokerCoreDeviceCA.pem。

將自訂 DNS 名稱/IP 地址新增至 TLS 伺服器憑證

產生的憑證上的主體替代名稱 (SAN) AWS IoT Greengrass 為 localhost。從閘道主機外部建立 TLS 連線時，TLS 驗證步驟會失敗，因為代理程式的主機名稱與伺服器憑證 localhost 上的主機名稱不相符。

為了解決不相符的主機名稱問題，AWS IoT Greengrass 提供兩種管理核心裝置端點的方式。本節涵蓋兩個選項。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的[管理核心裝置端點](#)。

- 若要使用核心裝置的 IP 地址連線至 EMQX 代理程式，請使用自動 IP 探索區段。
- 若要使用 DNS 名稱而非 IP 地址連線至 EMQX 代理程式，請使用手動管理區段。

Automated IP discovery

此選項可讓您的核心裝置自動探索其 IP 地址，並將其新增為代理程式憑證的主體別名 (SAN)。

1. 將 `aws.greengrass.clientdevices.IPDetector` 元件新增至核心裝置的部署。
2. 將變更部署至您的裝置
3. 等待部署完成。

部署完成後，您可以使用代理程式的 IP 地址建立安全的 TLS 連線。

IP 地址會自動新增為代理程式憑證的 SAN。

Manual DNS and IP Configuration

您可以將 DNS 名稱和 IP 地址手動新增為 TLS 憑證的主體別名 (SANs)。當您為閘道主機設定 DNS 名稱時，此方法非常有用。

Important

如果您使用的是 IPDetector 元件，請先將其從部署中移除再繼續。IPDetector 元件會覆寫手動端點組態。

手動設定端點

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選擇要設定的閘道。
4. 在 Edge 閘道組態區段中，選擇您的 Greengrass 核心裝置 URL。核心裝置的頁面隨即出現。
5. 選擇用戶端裝置索引標籤。
6. 選擇管理端點。
7. 在管理端點對話方塊中，輸入 DNS 名稱和您要新增為 SANs 的任何 IP 地址 (SAN)。使用連接埠 8883。
8. 選擇更新。

代理程式的 TLS 伺服器憑證會自動更新，以包含您的新端點。

使用 Linux 驗證 TLS 伺服器憑證更新

1. 在閘道主機上啟動 shell 工作階段。

```
docker exec emqx openssl x509 -in ./data/cert.pem -text -noout | grep -A1  
"Subject Alternative Name"
```

2. 命令會傳回類似下列的輸出：

```
X509v3 Subject Alternative Name:  
DNS:endpoint_you_added, DNS:localhost
```

3. 確認您的端點出現在 SANs 清單中。

使用 Windows 驗證 TLS 伺服器憑證更新

1. 在閘道主機上啟動 shell 工作階段。

```
(Get-PfxCertificate -FilePath "C:\greengrass\v2\work
\aws.greengrass.clientdevices.mqtt.EMQX\v2\data\cert.pem").Extensions | Where-
Object { $_.Oid.FriendlyName -eq "Subject Alternative Name" } | ForEach-Object
{ "Subject Alternative Name:", ($_.Format($true) -split "`n")[0..1] }
```

2. 命令會傳回類似下列的輸出：

```
Subject Alternative Name:
DNS Name=your-endpoint
DNS Name=localhost
```

3. 確認您新增的端點在 SANs 清單中。

在 AWS IoT SiteWise Edge 上測試 EMQX 代理程式連線

使用 TLS 憑證和身分驗證憑證設定 EMQX 代理程式之後，請務必確認您的設定是否正常運作。測試連線有助於確保您的安全組態已正確運作，且用戶端可以成功建立與代理程式的加密連線。本節示範如何使用 Mosquitto 命令列界面 (CLI) 用戶端測試您的代理程式連線，這是一種廣泛使用的 MQTT 用戶端工具，支援 TLS 加密和身分驗證。

使用 Mosquitto CLI 用戶端來測試 EMQX 代理程式連線

在此步驟中，我們將使用 mosquitto CLI 用戶端來測試我們的設定，並確保我們可以使用先前建立的使用者名稱和密碼成功連線到代理程式。若要取得步驟 3：設定 TLS 下的 BrokerCoreDeviceCA.pem 下列步驟。

```
mosquitto_sub -h hostname/ip address \
-p 8883 \
-t "#" \
-q 1 \
-u username -P password \
--cafile BrokerCoreDeviceCA.pem
```

Note

如果您連線的主機名稱/IP 地址與您傳遞給用戶端的 CA 憑證上的主體別名 (SAN) 不相符，您可能會收到 SSL：驗證錯誤。請參閱「將自訂 DNS 名稱/IP 地址新增至 TLS 伺服器憑證」的步驟 3：設定 TLS，了解如何取得具有正確 SAN 的憑證。

此時，所有使用者都有權發佈和訂閱代理程式上的所有主題。繼續執行「[在 EMQX 中設定 AWS IoT SiteWise Edge 的授權規則](#)」。

使用您自己的 CA

AWS IoT Greengrass 概述如何設定自己的用戶端裝置身分驗證元件，以使用您自己的憑證授權機構 (CA)。用戶端裝置身分驗證元件 (aws.greengrass.clientdevices.Auth) 會驗證用戶端裝置，並授權用戶端裝置動作。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的[使用您自己的憑證授權機構](#)。

若要使用您自己的 CA，請將 aws.greengrass.clientdevices.Auth 元件新增至部署，以便您可以指定自訂組態。

為外部防火牆連線開啟連接埠 8883

Linux

在您的 Linux 主機防火牆規則中，新增連接埠 8883 的傳入規則，以允許來自閘道主機外部的傳入連線。如果有任何防火牆，請確保允許連接埠 8883 上的傳入 TLS 連線。

Windows

在 Microsoft Windows 主機防火牆規則中，新增連接埠 8883 的傳入規則，以允許來自閘道主機外部的傳入連線。確保規則是允許規則，類型為連接埠，指定連接埠 8883。您可以根據您的網路組態來設定，以允許從外部應用程式到代理程式的連線。

在 EMQX 中設定 AWS IoT SiteWise Edge 的授權規則

EMQX 支援根據使用者名稱、IP 地址或用戶端 ID 等識別符新增授權規則。如果您想要限制連接到各種操作或主題的外部應用程式數量，這會很有用。

主題

- [使用內建資料庫搭配 Linux 設定授權](#)

- [使用內建資料庫搭配 Windows 設定授權](#)
- [更新授權的 EMQX 部署組態](#)
- [透過 EMQX 儀表板為使用者新增授權規則](#)

使用內建資料庫搭配 Linux 設定授權

當您設定授權規則時，有兩個組態選項取決於您的部署設定。

- Docker – 如果您要在沒有 的情況下執行標準Docker安裝Litmus Edge，請使用 Docker 橋接器閘道組態。這通常是您只部署 AWS IoT SiteWise 元件的情況。
- Litmus Edge – 如果您已Litmus Edge在閘道上安裝，請使用 Litmus Edge 網路子網路組態。

Note

如果您最初設定Docker橋接閘道並稍後安裝 Litmus Edge，請使用Litmus Edge網路子網路選項重新設定授權規則，以確保所有元件之間進行適當的通訊。

新增基本授權規則

1. 確認已部署並執行 EMQX 代理程式。
2. 在閘道主機上啟動 shell 工作階段。

Docker without Litmus Edge

對於沒有 的標準Docker安裝Litmus Edge，請執行：

```
/greengrass/v2/bin/swe-emqx-cli acl init
```

Litmus Edge network subnet

如果您使用的是 Litmus Edge，請判斷 Litmus Edge 網路子網路 IP：

```
docker network inspect LitmusNetwork | grep IPAM -A9
```

請注意輸出中的子網路值，並執行下列命令。litmus_subnet_ip 將取代為上一個步驟的子網路值。

```
/greengrass/v2/bin/swe-emqx-cli acl init litmus_subnet_ip
```

此工具會自動建立並套用授權規則，以允許從提供的 IP 地址連線到代理程式。它允許存取所有主題。這包括 IoT SiteWise OPC UA 收集器和 IoT SiteWise 發佈者。

3. 繼續執行「[更新授權的 EMQX 部署組態](#)」。

使用內建資料庫搭配 Windows 設定授權

本節涵蓋使用 Windows 部署的內建資料庫來設定授權規則。

新增基本授權規則

1. 確認已部署並執行 EMQX 代理程式。
2. 執行 AWS IoT SiteWise EMQX CLI 工具：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 acl init
```

此工具會自動建立並套用 ACL 規則，允許從 localhost (127.0.0.1) 連線到代理程式。它允許存取所有主題。這包括 IoT SiteWise OPC UA 收集器和 IoT SiteWise 發佈者。

3. 繼續執行「[更新授權的 EMQX 部署組態](#)」。

更新授權的 EMQX 部署組態

更新授權的 EMQX 部署組態

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選擇要設定的閘道。
4. 在 Edge 閘道組態區段中，複製您的 Greengrass 核心裝置值。儲存它以供日後使用。
5. 開啟 [AWS IoT 主控台](#)。
6. 在左側導覽的管理區段下，選擇 Greengrass 裝置，然後選擇部署。
7. 尋找您先前儲存的核心裝置值，然後選擇該連結以開啟部署。
8. 選擇動作下拉式選單按鈕，然後修訂。
9. 閱讀出現的訊息，然後選擇修訂部署。指定目標頁面隨即出現。

10. 選擇下一步，直到您到達設定元件步驟為止。
11. 選取`aws.greengrass.clientdevices.mqtt.EMQX`選項按鈕。
12. 選擇設定元件按鈕。隨即顯示元件的組態頁面。
13. 在組態更新下，選擇重設為元件版本的預設組態：`2.*.*`。
14. 根據您的作業系統，在要合併的組態區段中貼上下列內容。

Linux

```
{
  "emqxConfig": {
    "authorization": {
      "no_match": "deny",
      "sources": [
        {
          "type": "built_in_database"
        },
        {
          "type": "file",
          "path": "data/authz/acl.conf"
        }
      ]
    },
    "listeners": {
      "tcp": {
        "default": {
          "enabled": true,
          "enable_authn": false
        }
      },
      "ssl": {
        "default": {
          "enabled": true,
          "enable_authn": true,
          "ssl_options": {
            "verify": "verify_none",
            "fail_if_no_peer_cert": false
          }
        }
      }
    },
    "authentication": {
      "enable": true,
```

```

        "backend": "built_in_database",
        "mechanism": "password_based",
        "password_hash_algorithm": {
            "iterations": 210000,
            "mac_fun": "sha512",
            "name": "pbkdf2"
        },
        "user_id_type": "username"
    },
    "dashboard": {
        "listeners": {
            "http": {
                "bind": 18083
            }
        }
    }
},
"authMode": "bypass",
"dockerOptions": "-p 8883:8883 -p 127.0.0.1:1883:1883\n-p 127.0.0.1:18083:18083 -v emqx-data:/opt/emqx/data -e\nEMQX_NODE__NAME=emqx@local",
"requiresPrivilege": "true"
}

```

Windows

```

{
  "emqxConfig": {
    "authorization": {
      "no_match": "deny",
      "sources": [
        {
          "type": "built_in_database"
        },
        {
          "type": "file",
          "path": "C:\\greengrass\\v2\\work\\aws.greengrass.clientdevices.mqtt.EMQX\\v2\\data\\authz\\acl.conf"
        }
      ]
    },
    "listeners": {
      "tcp": {

```

```
        "default": {
            "enabled": true,
            "enable_authn": false
        }
    },
    "ssl": {
        "default": {
            "enabled": true,
            "enable_authn": true,
            "ssl_options": {
                "verify": "verify_none",
                "fail_if_no_peer_cert": false
            }
        }
    },
    "authentication": {
        "enable": true,
        "backend": "built_in_database",
        "mechanism": "password_based",
        "password_hash_algorithm": {
            "iterations": 210000,
            "mac_fun": "sha512",
            "name": "pbkdf2"
        },
        "user_id_type": "username"
    },
    "dashboard": {
        "listeners": {
            "http": {
                "bind": 18083
            }
        }
    },
    "authMode": "bypass",
    "requiresPrivilege": "true"
}
```

15. 選擇確認。
16. 選擇下一步，直到您到達檢閱步驟為止。
17. 選擇部署。

Note

從此時開始，您無法編輯 ACL 檔案來更新授權規則。或者，您可以在成功部署[透過 EMQX 儀表板為使用者新增授權規則](#)後繼續執行。

透過 EMQX 儀表板為使用者新增授權規則

您可以使用 EMQX 儀表板或 AWS IoT SiteWise EMQX CLI 工具新增或更新授權規則。AWS IoT SiteWise EMQX CLI 工具會使用 EMQX 的內建資料庫管理授權。

Note

新增授權規則是進階組態步驟，需要了解 MQTT 主題模式和存取控制。如需使用 EMQX 內建資料庫建立授權規則的詳細資訊，請參閱 EMQX 文件中的[使用內建資料庫](#)。

Note

提供的 EMQX 相關指示僅供參考。由於 EMQX 文件和功能可能會隨著時間而變更，而且我們不會維護其文件，我們建議您參閱[EMQX 的官方文件](#)以取得最新資訊。

EMQX dashboard

此程序說明如何在 EMQX 儀表板上新增授權規則。

EMQX 儀表板只能從閘道主機內存取。如果您嘗試從閘道主機外部連線，則無法存取儀表板。

使用 EMQX 儀表板新增授權規則

1. 確定您位於閘道主機中。
2. 開啟瀏覽器視窗並造訪 <http://localhost:18083/>。
3. 登入 EMQX 儀表板。此程序假設您已將預設登入憑證變更為您選擇的登入資料。如需初始設定的詳細資訊，請參閱[啟用使用者名稱和密碼身分驗證](#)。
4. 從下拉式選單中選擇遮罩圖示，然後選擇授權。
5. 選擇內建資料庫列上的許可按鈕。

- 在內建資料庫授權區段中，為您的業務需求新增或更新使用者授權規則。如需建立規則的更多指引，請參閱 EMQX 文件中的[使用內建資料庫](#)一節。

AWS IoT SiteWise CLI tool using Linux

若要在 Linux 中使用 AWS IoT SiteWise EMQX CLI 工具管理授權規則：

- 使用下列格式為使用者新增授權規則：

```
/greengrass/v2/bin/swe-emqx-cli auth add your-username your-action your-permission your-topic [your-action-permission-topic]
```

Example 新增使用者的授權規則

此範例說明如何為名為 `system1` 的使用者新增規則：

```
/greengrass/v2/bin/swe-emqx-cli auth add system1 \  
  publish allow "sensors/#" \  
  subscribe allow "control/#" \  
  all deny "#"
```

Example 檢視使用者的授權規則

若要檢視 `system1` 使用者的授權規則，請執行下列命令：

```
/greengrass/v2/bin/swe-emqx-cli auth list system1
```

Example 檢視所有現有的授權規則

若要檢視您目前擁有的所有授權規則，請執行下列命令：

```
/greengrass/v2/bin/swe-emqx-cli auth list
```

Example 刪除使用者的所有授權規則

若要刪除套用至特定使用者的所有授權規則，請執行下列命令：

```
/greengrass/v2/bin/swe-emqx-cli auth delete system1
```

系統會提示您確認刪除。

AWS IoT SiteWise CLI tool using Windows

若要使用 中的 AWS IoT SiteWise EMQX CLI 工具管理授權規則Windows PowerShell：

- 使用下列格式為使用者新增授權規則：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 auth add your-username your-action your-permission your-topic [your-action-permission-topic]
```

Example：新增使用者的授權規則

此範例說明如何為名為 的使用者新增規則system1：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 auth add system1 `
  publish allow "sensors/#" `
  subscribe allow "control/#" `
  all deny "#"
```

Example：檢視使用者的授權規則

若要檢視system1使用者的授權規則，請執行下列命令：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 auth list system1
```

Example：檢視所有現有的授權規則

若要檢視您目前擁有的所有授權規則，請執行下列命令：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 auth list
```

Example：刪除使用者的所有授權規則

若要刪除套用至特定使用者的所有授權規則，請執行下列命令：

```
C:\greengrass\v2\bin\swe-emqx-cli.ps1 auth delete system1
```

系統會提示您確認刪除。

使用 SiteWise Edge 和開放原始碼工具處理和視覺化資料

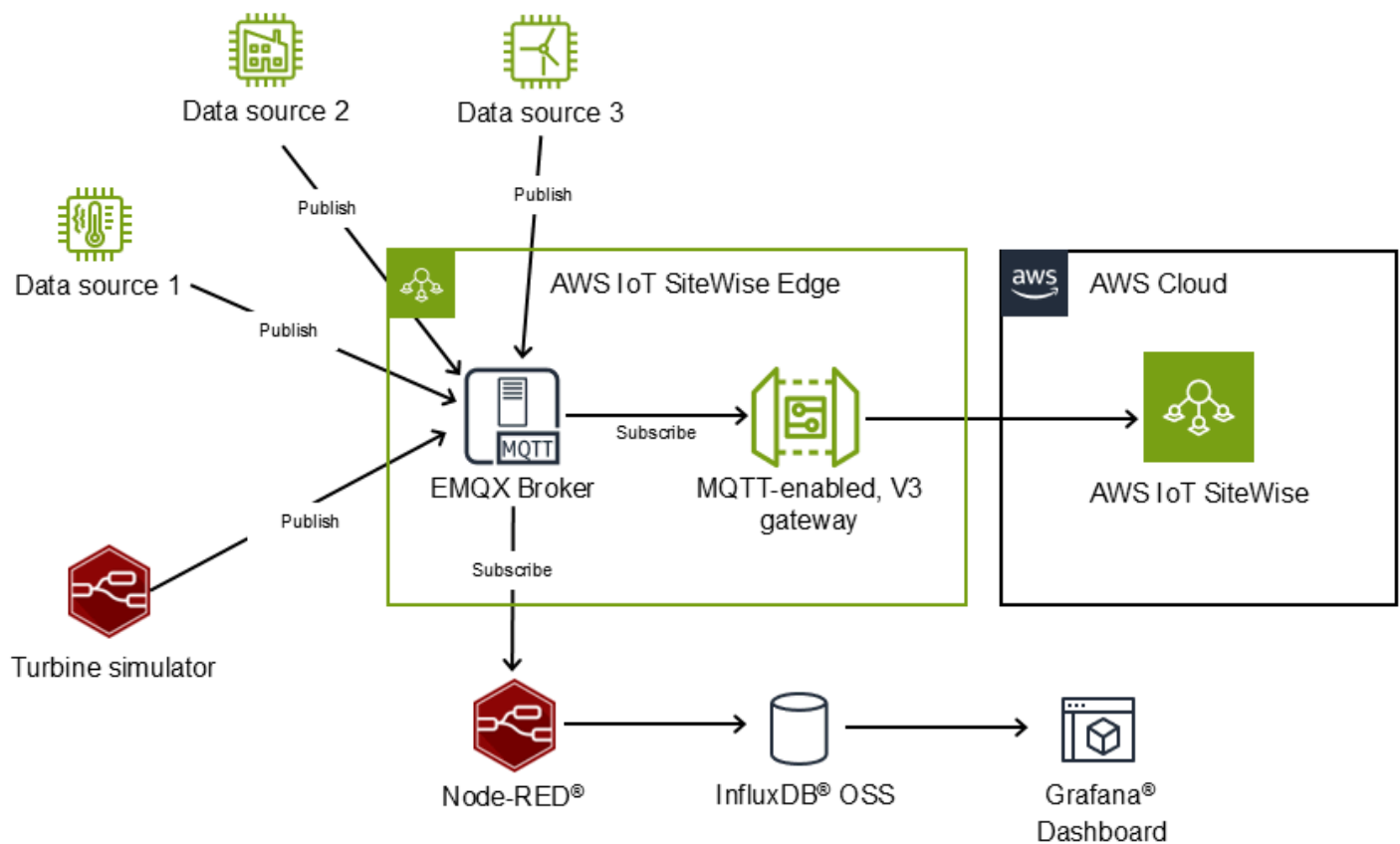
使用開放原始碼工具設定啟用 AWS IoT SiteWise Edge MQTT 的閘道，以進行本機處理和視覺化，以增強您的工業資料管理功能。

使用 SiteWise Edge，您可以使用外部開放原始碼工具建立本機資料處理管道。使用 [Node-RED®](#) 以 [InfluxDB®](#) 存放時間序列資料，並透過 [Grafana®](#) 儀表板監控操作。

Node-RED 會處理和轉換資料流程，而 InfluxDB 則提供時間序列資料儲存。Grafana 會顯示您的即時操作資料。將這些工具與 SiteWise Edge 搭配使用，以同步本機環境和之間的資料 AWS 雲端，為您提供即時本機洞見和長期雲端型分析功能。

Note

Node-RED®、InfluxDB® 和 Grafana® 不是 SiteWise Edge 的廠商或供應商。



Note

在本指南中，我們使用 [Grafana](#) for SiteWise Edge 的開放原始碼版本，而不是 [Amazon Managed Grafana](#) 服務。

部署選項

您可以使用兩種方法之一部署此解決方案。透過 Microsoft Windows 手動設定，您可以控制元件組態並與基礎設施整合。透過 Linux，您可以使用 Docker 在容器中部署預先設定的元件。

選擇符合您操作需求的方法。

- [手動設定開放原始碼整合 \(Windows\)](#) – 適用於自訂組態或現有基礎設施
- [設定與 Docker \(Linux\) 的開放原始碼整合](#) – 使用預先設定的元件快速部署

風力發電廠範例概觀

本指南使用風力發電廠範例，示範如何監控風力發電廠上渦輪機的風速。此實際案例說明常見的工業監控需求，其中本機和雲端型可見性對於營運效率都很有價值。

透過此整合，您可以：

- 使用 AWS IoT SiteWise Edge 閘道從工業設備收集資料
- 使用 Node-RED、InfluxDB 和 Grafana 在本機處理資料
- 使用 InfluxDB 在本機儲存資料
- 使用 Grafana 儀表板即時監控資料

在本指南中，我們使用風槍的範例。我們使用 Node-RED 來模擬產生風速資料的渦輪機。Node-RED 會轉譯資料承載、將資料發佈至 SiteWise Edge MQTT 代理程式、訂閱以從代理程式接收資料，並將資料儲存在本機 InfluxDB 中。此方法可確保在本機提供所有操作資料以供立即存取，並在雲端提供進一步分析。透過實作此模式，您可以獲得網路中斷的彈性，同時維持在中執行進階分析的能力 AWS 雲端。Grafana 會連線至 InfluxDB 進行本機監控，讓操作員即時查看指標，無需雲端相依性。啟用 SiteWise Edge MQTT 的閘道會連線至相同的 MQTT 代理程式，以將資料傳送至 AWS IoT SiteWise，在您的邊緣操作與雲端服務之間建立橋接。

無論您要監控製造設備、公用程式基礎設施或其他工業資產，都可以使用自己的資料和組態來建立專為您特定工業需求量身打造的類似工作流程。

開放原始碼整合的需求

在實作與 SiteWise Edge 的開放原始碼整合之前，請確定您的環境符合必要的需求。

- 硬體需求 - 您的閘道硬體必須符合 SiteWise Edge 閘道的需求。如需詳細資訊，請參閱[AWS IoT SiteWise Edge 自我託管閘道需求](#)啟用 MQTT 的 V3 閘道和 [AWS IoT SiteWise Edge 應用程式的需求](#)。

Important

部署其他開放原始碼元件時，請確定您的硬體符合 [InfluxDB](#)、[Node-RED](#) 和 [Grafana](#) 的需求。

- 您的網路組態必須支援 SiteWise Edge 元件與雲端連線之間的本機通訊。
- 所有 服務都必須在相同的主機上執行。

安全考量

建議您加密元件之間的所有通訊，尤其是從非本機網路存取介面時。為每個元件實作適當的存取控制，並遵循 AWS IoT SiteWise Edge 閘道組態和 AWS 帳戶安全的 AWS 最佳實務。

開發環境

本指南示範在閘道主機上執行和本機存取的 Node-RED、InfluxDB 和 Grafana。對於需要外部存取的生產部署，請實作安全措施，包括 TLS 加密、身分驗證和授權。遵循每個應用程式的安全最佳實務。

第三方軟體

此解決方案使用非 維護的第三方軟體 AWS，包括 InfluxDB、Node-RED、Grafana 和 node-red-contrib-influxdb 外掛程式。在部署之前，請確定這些元件符合您組織的安全需求、合規標準和控管政策。

Important

本指南參考 並使用非 擁有或維護的第三方軟體 AWS。在實作之前，請確保所有元件都符合您的安全、合規和控管要求。使用最新的安全修補程式讓所有軟體保持最新狀態，並遵循最佳實務來保護您的邊緣部署。

InfluxDB、Node-RED、Grafana 不是 SiteWise Edge 的廠商或供應商。

其他考量

實作與 SiteWise Edge 的開放原始碼整合時，請考慮這些其他因素。

- 使用所有服務、工具和元件的最新版本。
- 在雲端傳輸之前，在本機篩選和彙總資料，以降低 AWS IoT SiteWise 資料擷取成本。在 InfluxDB 中設定適當的資料保留期間，並正確調整閘道硬體的大小。如需詳細資訊，請參閱 [AWS IoT SiteWise 定價](#)。
- 實作所有資料的定期備份程序。
- 監控閘道上的資源用量，並為每個元件設定適當的資源限制。在 InfluxDB 中實作資料保留政策，以管理磁碟用量。

手動設定開放原始碼整合 (Windows)

使用本指南手動建立與 Grafana® 和 Node-RED® 連線的風速資料的時間序列儲存貯體。

在上手動安裝和設定 Node-RED、InfluxDB® 和 Grafana，Microsoft Windows 以控制您的部署組態。您可以使用 InfluxDB 從裝置存放和管理時間序列資料。

手動設定先決條件

開始之前，請先完成下列要求：

Note

在相同的主機上執行所有服務 (SiteWise Edge、InfluxDB、Node-RED 和 Grafana)。

- 安裝已啟用 MQTT 的 V3 閘道。如需詳細資訊，請參閱 [啟用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge](#)。
- 在本機安裝和執行這些服務：
 - InfluxDB OSS 第 2 版。如需安裝步驟，請參閱 [安裝 InfluxDB](#)。
 - Node-RED。如需安裝步驟，請參閱 [在本機安裝 Node-RED](#)。
 - Grafana。如需安裝步驟，請參閱 [安裝 Grafana](#)。

使用 InfluxDB 設定本機儲存

使用 InfluxDB®，您可以在本機儲存來自裝置的時間序列資料。本機儲存功能的目的是在網路中斷期間維持操作可見性，並減少時間關鍵型應用程式的延遲。您可以在邊緣執行分析和視覺化，同時仍可選擇選擇性地將資料轉送至雲端。

在本節中，您會為渦輪風速資料建立時間序列儲存貯體，並為 Grafana® 和 Node-RED® 連線產生 API 字符。InfluxDB 儲存貯體可做為時間序列資料的專用儲存容器，類似於傳統系統中的資料庫。API 權杖可讓您安全地以程式設計方式存取您的資料。

設定 InfluxDB

1. 完成先決條件步驟並確保所有工具都在相同的主機上執行後，請開啟您的 Web 瀏覽器並前往 <https://http://127.0.0.1:8086>。
2. (選用) 啟用 TLS 加密以增強安全性。如需詳細資訊，請參閱《》中的[啟用 TLS 加密](#)InfluxData Documentation。
3. 建立時間序列 InfluxDB 儲存貯體以存放 Node-RED 中的資料。儲存貯體將做為風力發電廠資料的專用容器，可讓您組織和管理此資料集特有的保留政策。如需詳細資訊，請參閱《》中的[管理儲存貯體](#)InfluxData Documentation。
4. (選用) 設定節點的資料保留期間。設定適當的保留期有助於有效管理儲存資源，方法是自動移除本機操作不再需要的舊資料。

如需資料保留的相關資訊，請參閱《》中的[InfluxDB 中的資料保留](#)InfluxData Documentation。

5. 產生儲存貯體的 API 字符。此字符將啟用 InfluxDB 與 Node-RED 和 Grafana 等其他元件之間的安全通訊。如此一來，只有授權的服務才能讀取或寫入您的資料存放區。如需詳細資訊，請參閱《》中的[建立權杖](#)InfluxData Documentation。

完成這些步驟後，您可以將時間序列資料儲存在 InfluxDB 執行個體中，為邊緣環境中的本機資料持久性和分析提供基礎。

設定 Node-RED 流程以進行 AWS IoT SiteWise 資料整合

使用 Node-RED®，您可以實作兩個流程來管理裝置和之間的資料 AWS IoT SiteWise。這些流程可協同運作，建立可處理本機和雲端資料流程的完整資料管理解決方案。

- 資料發佈流程 – 發佈至雲端。資料發佈流程會將資料傳送至 AWS IoT SiteWise。此流程透過產生感應器資料、將其轉換為 AWS IoT SiteWise 格式，以及發佈至 SiteWise Edge MQTT 代理程式，來模擬渦輪機裝置。這可讓您利用 AWS IoT SiteWise 的雲端功能來儲存、分析，以及與其他 AWS 服務的整合。

如需詳細資訊，請參閱[設定資料發佈流程](#)。

- 資料保留流程 – 將資料存放在邊緣。資料保留流程會訂閱 SiteWise Edge MQTT 代理程式以接收資料、將其翻譯為 InfluxDB® 格式，並將其存放在本機以進行監控。此本機儲存可讓您立即存取操作資料、降低關鍵時間應用程式的延遲，以及確保網路中斷期間的持續性。

如需詳細資訊，請參閱[設定資料保留流程](#)。

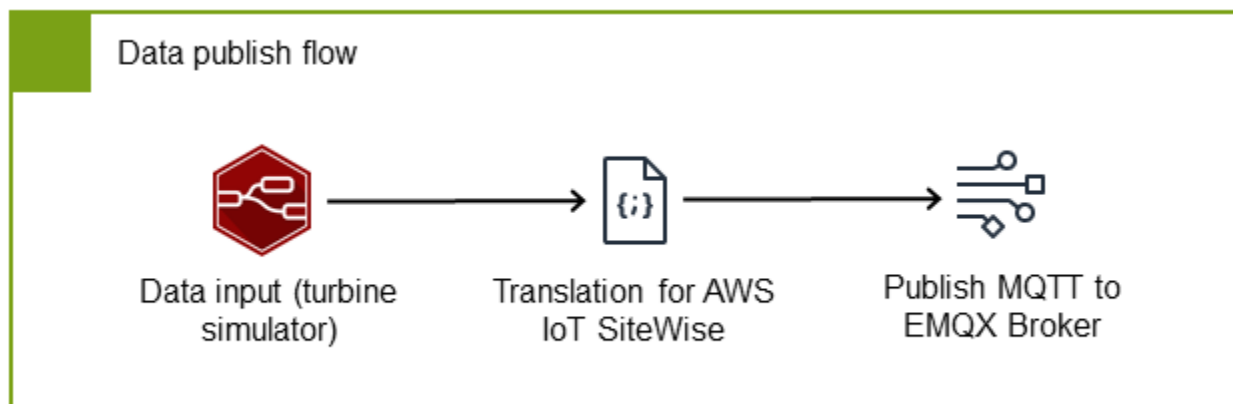
這兩個流程可一起運作，確保資料同時傳送至 AWS IoT SiteWise 並儲存在本機以供立即存取。

若要存取 Node-RED 主控台，請前往 <https://http://127.0.0.1:1880>。如需啟用 TLS 的資訊，請參閱[啟用 TLS 加密](#)。

設定資料發佈流程

資料發佈流程使用三個節點來建立管道，將工業資料傳送至雲端。此流程對於啟用雲端分析、長期儲存以及與其他 AWS 服務的整合至關重要。首先，模擬的裝置資料會傳送至 SiteWise Edge MQTT 代理程式。閘道會從代理程式取得資料，允許傳輸到 AWS IoT SiteWise 雲端，您可以在其中利用強大的分析和視覺化功能。

- 資料輸入 - 從工業設備或模擬器接收裝置資料
- 適用於 的資料轉譯器 AWS IoT SiteWise - 將資料轉譯為 AWS IoT SiteWise 格式，以確保與 SiteWise Edge 閘道的相容性
- MQTT 發佈者 - 將資料發佈至 SiteWise Edge MQTT 代理程式，使其可供本機和雲端消費者使用



設定資料輸入節點

在此範例中，資料輸入節點使用模擬風力渦輪機裝置來產生風速資料。此節點可做為工業資料的進入點，無論是來自模擬來源（如範例所示），還是來自生產環境中的實際工業設備。

我們為資料承載使用自訂 JSON 格式，以提供標準化結構，可有效率地與本機處理工具和 AWS IoT SiteWise 雲端服務搭配使用。此格式包含基本中繼資料，例如時間戳記和品質指標，以及實際的測量值，可在整個管道中實現全面的資料管理和品質追蹤。匯入注入節點，以接收此標準化 JSON 格式的模擬資料，其中包含時間戳記、品質指標和值。

如需 Node-RED 注入節點的詳細資訊，請參閱 Node-RED 文件中的[注入](#)一節。

渦輪模擬器會以此標準化 JSON 格式每秒產生風速資料：

Example：渦輪資料承載

```
{
  name: string,          // Property name/identifier
  timestamp: number,     // Epoch time in nanoseconds
  quality: "GOOD" | "UNCERTAIN" | "BAD",
  value: number | string | boolean
}
```

此格式提供數種優點：

- name 欄位可識別特定屬性或測量，可讓您從相同裝置追蹤多個資料點
- 以奈秒timestamp為單位的 可確保精確的時間追蹤，以進行準確的歷史分析
- quality 指標可協助您根據資料的可靠性篩選和管理資料
- 彈性value欄位支援不同的資料類型，以容納各種感應器輸出

Example：注入渦輪模擬器的節點

```
[
  {
    "id": "string",
    "type": "inject",
    "z": "string",
    "name": "Turbine Simulator",
    "props": [
      {
```

```

        "p": "payload.timestamp",
        "v": "",
        "vt": "date"
    },
    {
        "p": "payload.quality",
        "v": "GOOD",
        "vt": "str"
    },
    {
        "p": "payload.value",
        "v": "$random()",
        "vt": "jsonata"
    },
    {
        "p": "payload.name",
        "v": "/Renton/WindFarm/Turbine/WindSpeed",
        "vt": "str"
    }
],
"repeat": "1",
"crontab": "",
"once": false,
"onceDelay": "",
"topic": "",
"x": 270,
"y": 200,
"wires": [
    [
        "string"
    ]
]
}
]

```

設定節點以進行資料轉譯

SiteWise Edge 閘道需要特定格式的資料，以確保與 AWS IoT SiteWise 雲端的相容性。轉譯器節點是將您的輸入資料轉換為所需 AWS IoT SiteWise 承載格式的重要元件。此轉譯步驟可確保在雲端環境中正確處理、儲存及稍後分析 AWS IoT SiteWise 工業資料。

透過在此階段標準化資料格式，您可以啟用邊緣裝置與雲端服務之間的整合，您可以在其中使用資產建模、分析和視覺化功能。使用此結構：

Example : SiteWise Edge 資料剖析的承載結構

```
{
  "propertyAlias": "string",
  "propertyValues": [
    {
      "value": {
        "booleanValue": boolean,
        "doubleValue": number,
        "integerValue": number,
        "stringValue": "string"
      },
      "timestamp": {
        "timeInSeconds": number,
        "offsetInNanos": number
      },
      "quality": "GOOD" | "UNCERTAIN" | "BAD",
    }
  ]
}
```

Note

將 `propertyAlias` 與您的 MQTT 主題階層相符 (例如 `/Renton/WindFarm/Turbine/WindSpeed`)。這可確保您的資料與中的正確資產屬性正確關聯 AWS IoT SiteWise。如需詳細資訊, 請參閱 中的「資料串流別名」概念 [AWS IoT SiteWise 概念](#)。

1. 匯入用於 AWS IoT SiteWise 承載轉譯的範例函數節點。此函數會處理從標準化輸入格式到 AWS IoT SiteWise 相容格式的轉換, 確保適當的時間戳記格式、品質指標和值輸入。

```
[
  {
    "id": "string",
    "type": "function",
    "z": "string",
    "name": "Translate to SiteWise payload",
    "func": "let input = msg.payload;\nlet output = {};\n\noutput[\"propertyAlias\"] = input.name;\n\nlet propertyVal = {}\n\nlet timeInSeconds = Math.floor(input.timestamp / 1000);\nlet offsetInNanos = (input.timestamp % 1000) * 1000000;\n\npropertyVal[\"timestamp\"] = {\n  \"timeInSeconds\": timeInSeconds,\n  \"offsetInNanos\": offsetInNanos,\n};\n\npropertyVal[\"quality\"] = input.quality\n\nlet typeNameConverter = {\n
```

```

    \"number\": (x) => Number.isInteger(x) ? \"integerValue\" : \"doubleValue\",
    \"boolean\": (x) => \"booleanValue\",
    \"string\": (x) => \"stringValue\",
  },
  let typeName = typeNameConverter[typeof input.value](input.value)
  let propertyVal = {}
  propertyVal[\"value\"] = input.value
  let output = {
    \"propertyValues\": [propertyVal]
  }
  return {
    payload: JSON.stringify(output)
  }
}
]

```

2. 驗證 JavaScript 程式碼是否正確轉譯風速資料。函數會執行數個重要的任務：

- 從輸入擷取屬性名稱，並將其設定為 propertyAlias
- 將時間戳記從毫秒轉換為所需的秒和奈秒格式
- 保留資料可靠性追蹤的品質指標
- 自動偵測值類型並根據需求格式化 AWS IoT SiteWise

3. 將節點連接到您的流程，將其在資料輸入節點和 MQTT 發佈者之間連結。

如需撰寫專屬於您業務需求之函數的指引，請參閱 Node-RED 文件中的[撰寫函數](#)

設定 MQTT 發佈者

翻譯後，資料已準備好發佈至 SiteWise Edge MQTT 代理程式。

使用這些設定來設定 MQTT 發佈者，將資料傳送至 SiteWise Edge MQTT 代理程式：

匯入 MQTT 輸出節點

1. 使用 匯入 MQTT 輸出組態節點 "type": "mqtt out"。MQTT 輸出節點可讓您共用代理程式的組態。

2. 輸入鍵/值對，以取得 MQTT 代理程式連線和訊息路由的相關資訊。

匯入範例mqtt out節點。

Example

```
[
  {
    "id": "string",
    "type": "mqtt out",
    "z": "string",
    "name": "Publish to MQTT broker",
    "topic": "/Renton/WindFarm/Turbine/WindSpeed",
    "qos": "1",
    "retain": "",
    "respTopic": "",
    "contentType": "",
    "userProps": "",
    "correl": "",
    "expiry": "",
    "broker": "string",
    "x": 830,
    "y": 200,
    "wires": []
  },
  {
    "id": "string",
    "type": "mqtt-broker",
    "name": "emqx",
    "broker": "127.0.0.1",
    "port": "1883",
    "clientId": "",
    "autoConnect": true,
    "usetls": false,
    "protocolVersion": "5",
    "keepalive": 15,
    "cleansession": true,
    "autoUnsubscribe": true,
    "birthTopic": "",
    "birthQos": "0",
    "birthPayload": "",
    "birthMsg": {},
    "closeTopic": "",
  }
]
```

```
        "closePayload": "",
        "closeMsg": {},
        "willTopic": "",
        "willQos": "0",
        "willPayload": "",
        "willMsg": {},
        "userProps": "",
        "sessionExpiry": ""
    }
]
```

MQTT 輸出節點範例會使用下列資訊建立 MQTT 連線：

- 伺服器：127.0.0.1
- 連接埠：1883
- 通訊協定：MQTT V5

然後，MQTT 輸出節點會使用下列資訊設定訊息路由：

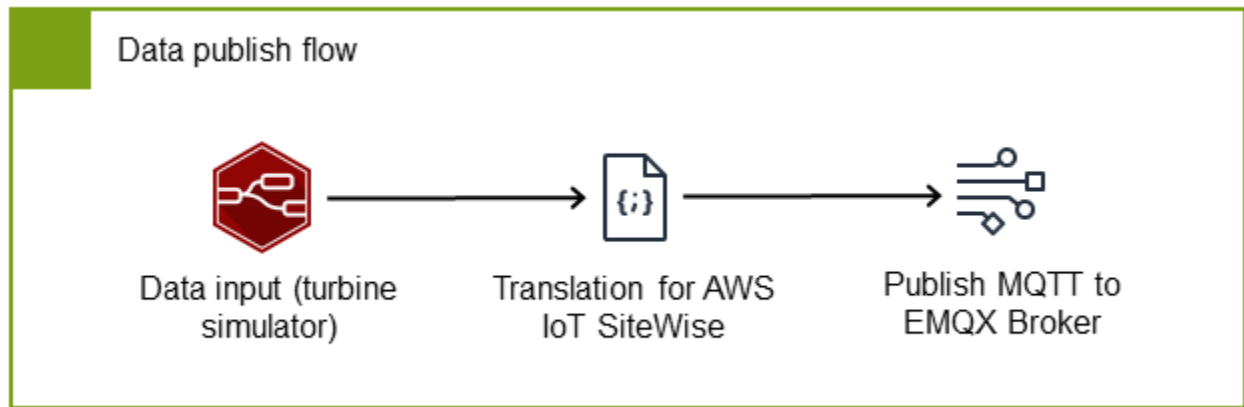
- 主題：/Renton/WindFarm/Turbine/WindSpeed
- QoS：1

部署和驗證節點

設定三個資料發佈流程節點之後，請依照下列步驟部署流程，並確認資料已正確傳輸至 AWS IoT SiteWise

部署和驗證連線

1. 連接三個節點，如資料發佈流程中所示。



2. 選擇部署以套用所有節點連線變更。
3. 導覽至 [AWS IoT SiteWise 主控台](#)，然後選擇資料串流。
4. 確定已在下拉式功能表中選取別名字首。然後，搜尋/Renton/WindFarm/Turbine/WindSpeed別名。

如果您在搜尋中看到正確的別名，表示您已部署流程和已驗證的資料傳輸。

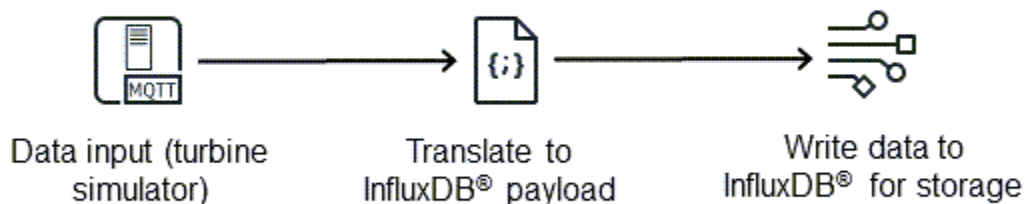
設定資料保留流程

資料保留流程可用來維持邊緣的操作可見性。這在網路中斷期間或當您需要立即存取資料時非常有用。此流程會訂閱 MQTT 代理程式以接收裝置資料、將其轉換為 InfluxDB® 格式，並將其存放在本機。透過實作此流程，您可以建立彈性本機資料存放區，讓操作員無需雲端相依性即可存取，從而在邊緣進行即時監控和決策。

流程由三個關鍵元件組成，共同確保正確擷取和存放您的資料：

- MQTT 訂閱用戶端 - 從代理程式接收資料，確保您擷取所有相關工業資料
- InfluxDB 轉譯器 - 將 AWS IoT SiteWise 承載轉換為 InfluxDB 格式，準備資料以實現高效的時間序列儲存
- InfluxDB 寫入器 - 處理本機儲存，確保本機應用程式的資料持久性和可用性

Data retention flow



設定 MQTT 訂閱用戶端

- AWS IoT SiteWise 匯入以下範例，在 Node-RED 中設定 MQTT 訂閱用戶端，以接收來自中 MQTT EMQX 代理程式的資料。

Example：節點中的 MQTT

```
[
  {
    "id": "string",
    "type": "mqtt in",
    "z": "string",
    "name": "Subscribe to MQTT broker",
    "topic": "/Renton/WindFarm/Turbine/WindSpeed",
    "qos": "1",
    "datatype": "auto-detect",
    "broker": "string",
    "nl": false,
    "rap": true,
    "rh": 0,
    "inputs": 0,
    "x": 290,
    "y": 340,
    "wires": [
      [
        "string"
      ]
    ]
  },
  {
```



```

    "id": "string",
    "type": "mqtt-broker",
    "name": "emqx",
    "broker": "127.0.0.1",
    "port": "1883",
    "clientId": "",
    "autoConnect": true,
    "usetls": false,
    "protocolVersion": "5",
    "keepalive": 15,
    "cleansession": true,
    "autoUnsubscribe": true,
    "birthTopic": "",
    "birthQos": "0",
    "birthPayload": "",
    "birthMsg": {},
    "closeTopic": "",
    "closePayload": "",
    "closeMsg": {},
    "willTopic": "",
    "willQos": "0",
    "willPayload": "",
    "willMsg": {},
    "userProps": "",
    "sessionExpiry": ""
  }
]

```

此訂閱可確保為本機儲存擷取發佈至代理程式的所有相關資料，並提供工業操作的完整記錄。節點使用與 [設定 MQTT 發佈者](#) 區段相同的 MQTT 連線參數，具有下列訂閱設定：

- 主題 – /Renton/WindFarm/Turbine/WindSpeed
- QoS – 1

如需詳細資訊，請參閱 Node-RED 文件中的 [連線至 MQTT 中介](#) 裝置。

設定 InfluxDB 轉譯器

InfluxDB 使用索引的 [標籤](#) 和值的 [欄位](#) 來組織資料。此組織會最佳化時間序列資料的查詢效能和儲存效率。匯入包含 JavaScript 程式碼的範例函數節點，將 AWS IoT SiteWise 承載轉換為 InfluxDB 格式。轉譯器會將屬性分割成兩個群組：

- 標籤 – 有效率索引的品質和名稱屬性
- 欄位 – 時間戳記（自 epoch 起以毫秒為單位）和值

Example：轉換為 InfluxDB 承載的函數節點

```
[
  {
    "id": "string",
    "type": "function",
    "z": "string",
    "name": "Translate to InfluxDB payload",
    "func": "let data = msg.payload;\n\nlet timeInSeconds =
data.propertyValues[0].timestamp.timeInSeconds;\nlet offsetInNanos =
data.propertyValues[0].timestamp.offsetInNanos;\nlet timestampInMilliseconds =
(timeInSeconds * 1000) + (offsetInNanos / 1000000);\n\nmsg.payload = [\n  {\n    \"timestamp(milliseconds_since_epoch)\": timestampInMilliseconds,\n    \"value\": data.propertyValues[0].value.doubleValue\n  },\n  {\n    \"name\": data.propertyAlias,\n    \"quality\": data.propertyValues[0].quality\n  }\n]\n\nreturn msg",
    "outputs": 1,
    "timeout": "",
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 560,
    "y": 340,
    "wires": [
      [
        "string"
      ]
    ]
  }
]
```

如需其他組態選項，請參閱 Node-RED GitHub 儲存庫中的 [node-red-contrib-influxdb](#)。GitHub

設定 InfluxDB 寫入器

InfluxDB 寫入器節點是資料保留流程中的最終元件，負責將工業資料儲存在本機 InfluxDB 資料庫中。此本機儲存對於在網路中斷期間維持操作可見性，並提供對時間關鍵型應用程式資料的立即存取至關重要。

1. 透過管理調色盤選項安裝 node-red-contrib-influxdb 套件。此套件提供將 Node-RED 與 InfluxDB 連線所需的節點。
2. 將 InfluxDB 輸出節點新增至您的流程。此節點將處理資料的實際寫入您的 InfluxDB 資料庫。
3. 設定伺服器屬性以建立與 InfluxDB 執行個體的安全連線：
 - a. 將版本設定為 2.0 - 這會指定您要連線至 InfluxDB v2.x，這會使用與舊版不同的 API
 - b. 將 URL 設定為 `http://127.0.0.1:8086` - 這指向您的本機 InfluxDB 執行個體
 - c. 輸入您的 InfluxDB 身分驗證字符。此安全字符會授權與資料庫的連線。您在[使用 InfluxDB 設定本機儲存](#)程序期間產生權杖。
4. 指定儲存位置參數，以定義資料的存放位置和方式：
 - a. 輸入您的 InfluxDB 組織名稱 – 組織是一組使用者的工作區，您的儲存貯體和儀表板屬於其中。如需詳細資訊，請參閱《》中的[管理組織](#)InfluxData Documentation。
 - b. 指定 InfluxDB 儲存貯體（例如 WindFarmData）– 儲存貯體等同於傳統系統中的資料庫，做為時間序列資料的容器
 - c. 設定 InfluxDB 測量（例如 TurbineData）– 測量類似於關聯式資料庫中的資料表，可組織相關資料點

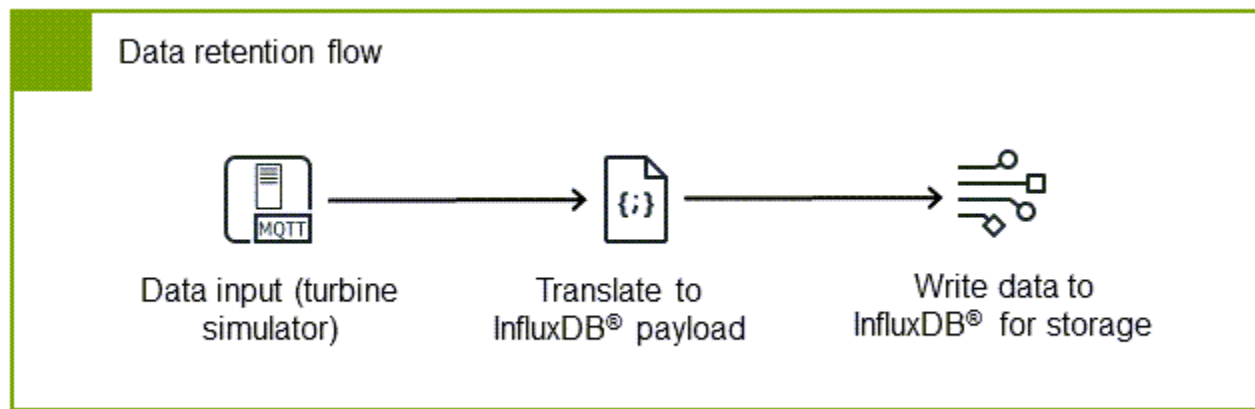
Note

在 InfluxDB 執行個體的左側邊欄中尋找您的組織名稱。組織、儲存貯體和測量概念是 InfluxDB 資料組織模型的基礎。如需詳細資訊，請參閱 [InfluxDB 文件](#)。

部署和驗證保留流程

設定資料保留流程的所有元件之後，您需要部署並驗證系統是否正常運作。此驗證可確保您的工業資料在本機正確存放，以便立即存取和分析。

1. 連接三個節點，如資料保留流程圖所示。這會建立從資料訂閱到本機儲存的完整管道。



2. 選擇部署以套用變更並啟用流程。這會啟動資料收集和儲存程序。
3. 使用 InfluxDB Data Explorer 查詢和視覺化您的資料。此工具可讓您驗證資料是否已正確儲存，並建立時間序列資料的初始視覺化。

在 Data Explorer 中，您應該能夠查看隨時間記錄的風速測量結果，確認從產生資料到本機儲存的整個管道都正常運作。

如需詳細資訊，請參閱《[中的在 Data Explorer 中查詢](#) InfluxData Documentation。

部署資料發佈流程和資料保留流程後，您現在擁有將資料傳送至 AWS IoT SiteWise 雲端的完整系統，同時保留本機複本以立即存取和恢復能力。這種雙路徑方法可確保您獲得雲端型分析和儲存的優勢，同時保持邊緣的操作可見性。

設定 Grafana for SiteWise Edge

Grafana® 可讓您為工業資料建立本機即時監控儀表板。透過視覺化儲存在 InfluxDB® 中的資料，您可以為操作員提供設備效能、程序效率和潛在問題的即時洞見。在邊緣的這種可見性對於時間敏感的操作和在網路中斷期間維持持續性至關重要。

設定資料來源

將 Grafana 連接到 InfluxDB 資料庫會為您的工業資料建立強大的視覺化層。此連線可啟用即時監控儀表板，讓運算子在沒有雲端相依性的情況下做出明智的決策。

1. 在瀏覽器中導覽至 <http://127.0.0.1:3000>，在本機存取您的 Grafana 執行個體。如果需要啟用 TLS，請參閱 Grafana Labs 文件中的[設定 Grafana HTTPS 以取得安全 Web 流量](#)。
2. 新增 InfluxDB 資料來源，指向 Node-RED 寫入資料的 InfluxDB 時間序列儲存貯體。例如 WindFarmData。此連線會在儲存的資料與視覺化平台之間建立連結。

3. 如需詳細說明，請參閱 Grafana Labs 文件中的[設定 InfluxDB 資料來源](#)。

為 SiteWise Edge 資料建立 Grafana 儀表板

建立儀表板是建置本機監控解決方案的最後一步。儀表板提供工業資料的視覺化呈現，讓您更輕鬆地一目了然地識別趨勢、異常和潛在問題。

- 依照指南建立儀表板。如需詳細資訊，請參閱 Grafana Labs 文件中的[建置您的第一個儀表板](#)。此範本假設您的儲存貯體已命名，WindFarmData且測量結果為 TurbineData。

您也可以透過匯入提供的範例儀表板範本來使用快速入門指南，為 Node-RED 在上一節中產生的資料快速建立具有時間序列圖的儀表板。此範本提供一個起點，您可以自訂以滿足特定的監控需求。

```
{
  "__inputs": [
    {
      "name": "DS_WINDFARM-DEMO",
      "label": "windfarm-demo",
      "description": "",
      "type": "datasource",
      "pluginId": "influxdb",
      "pluginName": "InfluxDB"
    }
  ],
  "__elements": {},
  "__requires": [
    {
      "type": "grafana",
      "id": "grafana",
      "name": "Grafana",
      "version": "11.6.0-pre"
    },
    {
      "type": "datasource",
      "id": "influxdb",
      "name": "InfluxDB",
      "version": "1.0.0"
    },
    {
      "type": "panel",
      "id": "timeseries",
```

```
    "name": "Time series",
    "version": ""
  }
],
"annotations": {
  "list": [
    {
      "builtIn": 1,
      "datasource": {
        "type": "grafana",
        "uid": "-- Grafana --"
      },
      "enable": true,
      "hide": true,
      "iconColor": "rgba(0, 211, 255, 1)",
      "name": "Annotations & Alerts",
      "type": "dashboard"
    }
  ]
},
"editable": true,
"fiscalYearStartMonth": 0,
"graphTooltip": 0,
"id": null,
"links": [],
"panels": [
  {
    "datasource": {
      "type": "influxdb",
      "uid": "${DS_WINDFARM-DEMO}"
    },
    "fieldConfig": {
      "defaults": {
        "color": {
          "mode": "palette-classic"
        },
        "custom": {
          "axisBorderShow": false,
          "axisCenteredZero": false,
          "axisColorMode": "text",
          "axisLabel": "",
          "axisPlacement": "auto",
          "barAlignment": 0,
          "barWidthFactor": 0.6,
```

```
    "drawStyle": "line",
    "fillOpacity": 0,
    "gradientMode": "none",
    "hideFrom": {
      "legend": false,
      "tooltip": false,
      "viz": false
    },
    "insertNulls": false,
    "lineInterpolation": "linear",
    "lineWidth": 1,
    "pointSize": 5,
    "scaleDistribution": {
      "type": "linear"
    },
    "showPoints": "auto",
    "spanNulls": false,
    "stacking": {
      "group": "A",
      "mode": "none"
    },
    "thresholdsStyle": {
      "mode": "off"
    }
  },
  "mappings": [],
  "thresholds": {
    "mode": "absolute",
    "steps": [
      {
        "color": "green"
      },
      {
        "color": "red",
        "value": 80
      }
    ]
  }
},
"overrides": []
},
"gridPos": {
  "h": 8,
  "w": 12,
```

```

        "x": 0,
        "y": 0
    },
    "id": 1,
    "options": {
        "legend": {
            "calcs": [],
            "displayMode": "list",
            "placement": "bottom",
            "showLegend": true
        },
        "tooltip": {
            "hideZeros": false,
            "mode": "single",
            "sort": "none"
        }
    },
    "pluginVersion": "11.6.0-pre",
    "targets": [
        {
            "datasource": {
                "type": "influxdb",
                "uid": "${DS_WINDFARM-DEMO}"
            },
            "query": "from(bucket: \"WindFarmData\")\n  |> range(start:
v.timeRangeStart, stop: v.timeRangeStop)\n  |> filter(fn: (r) => r[\"_measurement
\"] == \"TurbineData\")\n  |> filter(fn: (r) => r[\"_field\"] == \"value\")\n
  |> filter(fn: (r) => r[\"name\"] == \"/Renton/WindFarm/Turbine/WindSpeed\")\n
  |> filter(fn: (r) => r[\"quality\"] == \"GOOD\")\n  |> aggregateWindow(every:
v.windowPeriod, fn: mean, createEmpty: false)\n  |> yield(name: \"mean\")",
            "refId": "A"
        }
    ],
    "title": "Panel Title",
    "type": "timeseries"
}
],
"schemaVersion": 41,
"tags": [],
"templating": {
    "list": []
},
"time": {
    "from": "now-6h",

```



```
    "to": "now"
  },
  "timepicker": {},
  "timezone": "browser",
  "title": "demo dashboard",
  "uid": "fejc0t08o6d4wb",
  "version": 1,
  "weekStart": ""
}
```

設定與 Docker (Linux) 的開放原始碼整合

若要簡化部署程序，您可以使用 Docker 在 Linux 環境中設定 Node-RED®、InfluxDB® 和 Grafana®。此方法使用預先設定的容器，以便快速部署和更輕鬆地管理元件。

Docker 設定先決條件

開始之前，請確認 具有下列項目：

- 啟用 MQTT 的 V3 閘道。如需詳細資訊，請參閱[啟用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge](#)。
- Docker Compose 外掛程式。如需安裝步驟，請參閱[Docker 手冊文件中的安裝 Docker Compose 外掛程式](#)。

部署 服務

此部署會在相同的主機上執行 SiteWise Edge、InfluxDB、Node-RED 和 Grafana。

設定環境

1. 取得根存取權：

```
sudo -i
```

2. 建立 .env 檔案或匯出這些環境變數：

```
export INFLUXDB_PASSWORD=your-secure-influxdb-password
export INFLUXDB_TOKEN=your-secure-influxdb-token
export GRAFANA_PASSWORD=your-secure-grafana-password
```

設定 Docker 網路

- 使用名稱 建立橋接網路SiteWiseEdgeNodeRedDemoNetwork。

```
docker network create --driver=bridge SiteWiseEdgeNodeRedDemoNetwork
```

更新 SiteWise Edge 部署

1. 導覽至 [AWS IoT 主控台](#)
2. 在管理區段下的左側導覽功能表中選擇 Greengrass 裝置，然後選擇核心裝置。
3. 選取連接至 SiteWise Edge Gateway 的核心裝置。
4. 選擇部署索引標籤，然後選取部署 ID 值。
5. 選擇動作，然後選取修訂。
6. 讀取快顯訊息，然後選擇修訂部署。
7. 在步驟 2 - 選取元件中，選取下列元件，然後選擇下一步。
 - aws.greengrass.clientdevices.mqtt.EMQX
 - aws.iot.SiteWiseEdgePublisher
8. 在步驟 3 - 設定元件中，選取aws.greengrass.clientdevices.mqtt.EMQX元件值並新增下列網路組態：

```
{
  "emqxConfig": {
    "authorization": {
      "no_match": "allow"
    },
    "listeners": {
      "tcp": {
        "default": {
          "enabled": true,
          "enable_authn": false
        }
      }
    }
  },
  "authMode": "bypass",
  "dockerOptions": "-p 127.0.0.1:1883:1883 --
network=SiteWiseEdgeNodeRedDemoNetwork",
```

```
"requiresPrivilege": "true"
}
```

9. 選擇下一步。
10. 在步驟 4 - 設定進階設定中，選擇下一步。
11. 選擇部署

啟動 服務

1. 使用 Docker Compose 檔案啟動服務。在包含 `compose.yaml` 檔案的目錄下執行下列命令。

```
docker compose up -d
```

2. 建立 SSH 通道以存取 服務：

```
ssh -i path_to_your_ssh_key -L 1880:127.0.0.1:1880 -L 3000:127.0.0.1:3000 -L  
8086:127.0.0.1:8086 username@gateway_ip_address
```

此部署會在 中建立下列服務SiteWiseEdgeNodeRedDemoNetwork network：

InfluxDB v2 (連接埠 8086)

包括預先設定的組織 (iot-sitewise-edge)、WindFarmData InfluxDB 儲存貯體和管理登入資料

Node-RED (連接埠 1880)

包含 InfluxDB 節點和預先設定的 AWS IoT SiteWise 整合流程

Grafana (連接埠 3000)

包括管理員使用者、InfluxDB 資料來源和監控儀表板

存取 服務

部署之後，請使用下列 URLs和登入資料存取 服務：

Note

您可以從主機或閘道機器存取每個服務。

服務存取詳細資訊

服務	URL	登入資料
Node-RED	http://127.0.0.1:1880	不需要登入資料
InfluxDB	http://127.0.0.1:8086	使用者名稱：admin 密碼：\$INFLUXDB_PASSWORD
格拉法納	http://127.0.0.1:3000	使用者名稱：admin 密碼：\$GRAFANA_PASSWORD

驗證部署

若要確保您的部署成功，請執行下列檢查：

- 對於 Node-RED，驗證是否存在兩個預先載入的流程：
 - 資料發佈流程
 - 資料保留流程
- 對於 AWS IoT SiteWise 主控台中 AWS IoT SiteWise 具有別名的資料串流是否存在 /Renton/WindFarm/Turbine/WindSpeed。
- 對於 InfluxDB，請使用 Data Explorer 驗證 WindFarmData 儲存貯體內 TurbineData 測量中的資料儲存。
- 對於 Grafana，檢視儀表板以確認顯示從 Node-RED 產生的時間序列資料。

開放原始碼整合的程序資料

資料可以在不同的階段使用各種工具進行處理（例如轉換或彙總），每個工具都提供不同的監控需求。

使用 Node-RED 節點處理資料

使用 Node-RED® 內建處理節點即時轉換您的資料。透過 Node-RED 主控台設定這些節點，以建立您的資料管道。

資料轉換節點

AWS IoT SiteWise 使用這些節點轉換個別資料點，類似於 中的轉換：

- 變更節點 - 對資料執行簡單的值修改。
- 函數節點 - 為複雜的資料處理啟用自訂 JavaScript 轉換。

指標計算節點

使用這些節點 AWS IoT SiteWise，將多個資料點合併為單一輸出，類似於 中的指標：

- 批次節點 - 將多個訊息分組以進行批次處理。
- 聯結節點 - 將多個資料串流合併為單一輸出。
- 彙總工具節點 - 從多個資料點計算彙總指標。

如需其他節點選項，請參閱 [Node-RED 程式庫](#)。

建立 InfluxDB 任務

雖然 Node-RED 擅長快速設定的基本資料處理，但複雜的指標計算在流程型程式設計中可能會變得具有挑戰性。InfluxDB® 任務透過排程 Flux 指令碼提供替代方案，以滿足進階處理需求。

將 InfluxDB 任務用於：

- 跨大型資料集的統計彙總
- 多個屬性的數學操作
- 從多個來源衍生的測量

任務功能

- 排程執行 - 根據 Cron 表達式執行任務
- 批次處理 - 最佳化時間序列資料的操作
- 錯誤復原 - 自動重試失敗的操作
- 監控 - 透過詳細日誌追蹤執行

透過 InfluxDB UI、API 或 CLI 管理任務。如需詳細資訊，請參閱 [使用 InfluxDB 任務處理資料](#)。

使用 Grafana 轉換

轉換 Grafana® 中的資料視覺化，而不修改 InfluxDB 中的來源資料。Grafana 轉換僅適用於視覺化層。

- Visual Builder - 建立轉換而不撰寫程式碼
- 即時預覽 - 即時檢視轉換結果
- 多來源 - 處理來自多個資料庫來源的資料
- 儲存效率 - 在視覺化時間轉換資料，而不儲存中間結果

如需詳細資訊，請參閱[轉換資料](#)。

開放原始碼整合的故障診斷

如需 SiteWise Edge 閘道開放原始碼整合相關主題的疑難排解詳細資訊，請參閱 [對 Edge 的開放原始碼整合進行故障診斷](#)。

傳統串流、適用於 AWS IoT SiteWise Edge 的 V2 閘道

了解 Classic 串流、適用於 AWS IoT SiteWise Edge 的 V2 閘道的功能和限制。

在推出啟用 MQTT 的 V3 閘道之前，V2 閘道會維護先前 AWS IoT SiteWise 部署中熟悉的傳統功能。這些 SiteWise Edge 閘道被視為 Classic 串流、V2 閘道。它們可保持回溯相容性，並與資料處理套件相容。雖然 Classic 串流 V2 閘道為現有設定提供可靠的效能，但與較新的閘道選項相比，它有限制。具體而言，此閘道類型與啟用 MQTT 的 V3 閘道目的地中可用的進階功能不完全相容。若要使用 MQTT 訊息通訊協定，您可以建立新的啟用 MQTT 的 V3 閘道。如需詳細資訊，請參閱[啟用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge](#)。

主題

- [使用套件在 SiteWise Edge 中收集和處理資料](#)
- [設定 AWS IoT SiteWise 發佈者元件](#)
- [目的地和 AWS IoT Greengrass 串流管理員](#)
- [在 Edge 上設定 AWS IoT SiteWise 邊緣功能](#)
- [設定 AWS IoT SiteWise 模型和資產的邊緣資料處理](#)

使用套件在 SiteWise Edge 中收集和處理資料

AWS IoT SiteWise 邊緣閘道使用不同的套件來判斷如何收集和處理您的資料。

目前，可使用下列套件：

- 資料收集套件 – 使用此套件收集工業資料，並將其路由至 AWS 雲端目的地。根據預設，系統會自動為您的 SiteWise Edge 閘道啟用此套件。
- 資料處理套件 – 使用此套件啟用 SiteWise Edge 閘道與邊緣設定資產模型和資產的通訊。您可以使用邊緣組態來控制要在現場運算和處理哪些資產資料。然後，您可以將資料傳送至 AWS IoT SiteWise 或其他 AWS 服務。如需資料處理套件的詳細資訊，請參閱 [the section called “設定邊緣資料處理”](#)。

升級套件

Important

從 2.0.x 之前的資料處理套件版本（包括）升級至 2.1.x 版會導致資料遺失本機儲存的測量。

SiteWise Edge 閘道使用不同的套件來判斷如何收集和處理您的資料。您可以使用 AWS IoT SiteWise 主控台升級套件。

升級套件（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 在閘道清單中，選擇 SiteWise Edge 閘道與您要升級的套件。
4. 在閘道組態區段中，選擇可用的軟體更新。
5. 在編輯軟體版本頁面上，選擇更新。

Note

您只能升級已啟用的套件。若要尋找為此 SiteWise Edge 閘道啟用的套件清單，請選擇概觀，然後參閱 Edge 功能區段。

6. 在編輯軟體版本頁面的閘道元件更新區段中，執行下列動作：
 - 若要更新 OPC UA 收集器，請選擇版本，然後選擇部署。
 - 若要更新發佈者，請選擇版本，然後選擇部署。
 - 若要更新資料處理套件，請選擇版本，然後選擇部署。

7. 當您完成部署新版本時，請選擇完成。

如果您在升級套件時遇到問題，請參閱 [無法將套件部署至 SiteWise Edge 閘道](#)。

設定 AWS IoT SiteWise 發佈者元件

建立 AWS IoT SiteWise Edge 閘道並安裝軟體後，您可以設定發佈者元件，以便 SiteWise Edge 閘道可以將資料匯出至 AWS 雲端。使用發佈者元件來啟用其他功能或設定預設設定。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的 [AWS IoT SiteWise 發佈者](#)。

Note

發佈者組態會根據您使用的閘道類型而有所不同。對於 Classic 串流、V2 閘道，請使用 `iotsitewise:publisher:2` 命名空間。對於啟用 MQTT 的 V3 閘道，請使用 `iotsitewise:publisher:3` 命名空間。

Console

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取您要為其設定發佈者的 SiteWise Edge 閘道。
4. 在發佈者組態區段中，選擇編輯
5. 針對發佈順序，選擇下列其中一項：
 - 先發佈最舊的資料 – SiteWise Edge 閘道預設會先將最舊的資料發佈至雲端。
 - 先發佈最新資料 – SiteWise Edge 閘道會先將最新資料發佈至雲端。
6. （選用）如果您不希望 SiteWise Edge 閘道壓縮您的資料，請在上傳資料時取消選取啟用壓縮。
7. （選用）如果您不想發佈舊資料，請選擇排除過期的資料並執行下列動作：
 - 針對截斷期間，輸入值並選擇單位。截止期間必須介於 5 分鐘到 7 天之間。例如，如果截止期間為三天，則超過三天的資料不會發佈到雲端。
8. （選用）若要設定在本機裝置上處理資料的自訂設定，請選擇本機儲存設定並執行下列動作：

- a. 針對保留期間，輸入數字並選擇單位。保留期間必須介於一分鐘到 30 天之間，且大於或等於輪換期間。例如，如果保留期間為 14 天，則 SiteWise Edge 閘道會在儲存 14 天後，刪除邊緣早於指定截止期間的任何資料。
- b. 針對輪換期間，輸入數字並選擇單位。輪換期間必須大於一分鐘，且等於或小於保留期間。例如，假設輪換期間為兩天，SiteWise Edge 閘道會批次處理，並將早於截止期間的資料儲存到單一檔案。對於透過的自我託管閘道 AWS IoT Greengrass V2，SiteWise Edge 閘道每兩天會將一批資料傳輸到下列本機目錄：`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`。
- c. 針對儲存容量，輸入大於或等於 1 的值。如果儲存容量為 2 GB，則 SiteWise Edge 閘道會在本機儲存超過 2 GB 的資料時開始刪除資料。

9. 選擇儲存。

AWS CLI

使用 [UpdateGatewayCapabilityConfiguration](#) API 來設定發佈者。

將 `capabilityNamespace` 參數設為 `iotsitewise:publisher:2`。

Example：Classic Stream、V2 閘道的發佈者組態

發佈者命名空間：`iotsitewise:publisher:2`

```
{
  "SiteWisePublisherConfiguration": {
    "publishingOrder": "TIME_ORDER",
    "enableCompression": true,
    "dropPolicy": {
      "cutoffAge": "7d",
      "exportPolicy": {
        "retentionPeriod": "7d",
        "rotationPeriod": "6h",
        "exportSizeLimitGB": 10
      }
    }
  },
  "SiteWiseS3PublisherConfiguration": {
    "accessRoleArn": "arn:aws:iam:123456789012:role/roleName",
    "streamToS3ConfigMapping": [
      {
        "streamName": "S3_OPC-UA_Data_Collector",
```

```

        "targetBucketArn": "arn:aws:s3:::amzn-s3-demo-bucket/dataCollector",
        "publishPolicy": {
            "publishFrequency": "10m",
            "localSizeLimitGB": 10
        },
        "siteWiseImportPolicy": {
            "enableSiteWiseStorageImport": true,
            "enableDeleteAfterImport": true
        }
    }
]
}
}

```

發佈者提供下列您可以自訂的組態參數：

SiteWisePublisherConfiguration

publishingOrder

資料發佈至雲端的順序。此參數的值可以是下列其中一項：

- TIME_ORDER (先發佈最舊的資料) – 預設情況下，最早的資料會先發佈到雲端。
- RECENT_DATA (先發佈最新資料) – 最新資料會先發佈至雲端。

enableCompression

將此設為 true 以在發佈之前壓縮資料。資料壓縮可減少頻寬使用量。

dropPolicy

(選用) 控制發佈至雲端的資料的政策。

cutoffAge

要以天、小時和分鐘為單位發佈的資料最長存留期。例如 7d 或 1d7h16m。不會傳送早於您指定內容的資料 AWS IoT SiteWise。

早於截止期間的資料不會發佈至雲端。截止時間必須介於 5 分鐘到 7 天之間。

當您指定截止存留期 d 時 m，可以使用 h、和。請注意，m 代表分鐘，h 代表小時，d 代表天數。

exportPolicy

(選用) 管理邊緣資料儲存的政策。此政策適用於早於截止期限的資料。

retentionPeriod

儲存指定保留期間之後，您的 SiteWise Edge 閘道會從本機儲存體刪除邊緣早於截止期間的任何資料。保留期間必須介於一分鐘到 30 天之間，且大於或等於輪換期間。

您可以在指定保留期間d時使用 mh、 和 。請注意， m代表分鐘， h 代表小時， d代表天數。

rotationPeriod

將早於截止期間的資料批次儲存到單一檔案的時間間隔。SiteWise Edge 閘道會在每個輪換期間結束時，將一批資料傳輸到下列本機目錄：`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`。輪換期間必須大於一分鐘，且等於或小於保留期間。

您可以在指定輪換期間d時使用 mh、 和 。請注意， m代表分鐘， h 代表小時， d代表天數。

exportSizeLimitGB

本機儲存的資料允許大小上限，以 GB 為單位。如果違反此配額，SiteWise Edge 閘道會開始刪除最早的資料，直到本機存放的資料大小等於或小於配額為止。此參數的值必須大於或等於 1。

SiteWiseS3PublisherConfiguration

accessRoleArn

授予管理您要發佈之 Amazon S3 儲存貯體 AWS IoT SiteWise 許可的存取角色。

streamToS3ConfigMapping

將串流映射至 Amazon S3 組態的組態陣列。

streamName

要從 Amazon S3 組態讀取和發佈的串流。

targetBucketArn

ARN 要發佈的儲存貯體。

publishPolicy

publishFrequency

SiteWise Edge 閘道發佈至 Amazon S3 儲存貯體的頻率。

localSizeLimitGB

寫入本機磁碟的檔案大小上限。如果超過此閾值，發佈者會將所有緩衝資料發佈至其目的地。

siteWiseImportPolicy

enableSiteWiseStorageImport

將此設定為 true，將資料從 Amazon S3 儲存貯體匯入至 AWS IoT SiteWise 儲存貯體。

enableDeleteAfterImport

將此設定為 true 以在擷取至 AWS IoT SiteWise 儲存貯體後刪除 Amazon S3 儲存貯體中的檔案。

目的地和 AWS IoT Greengrass 串流管理員

AWS IoT Greengrass 串流管理員可讓您將資料傳送至下列 AWS 雲端 目的地：中的頻道 AWS IoT Analytics、Amazon Kinesis Data Streams 中的串流、中的資產屬性 AWS IoT SiteWise，或 Amazon Simple Storage Service (Amazon S3) 中的物件。如需詳細資訊，請參閱 [《AWS IoT Greengrass 開發人員指南》中的管理 核心上的資料串流](#)。AWS IoT Greengrass Version 2

Example：資料串流訊息結構

下列範例顯示串流管理員傳輸的必要資料 AWS IoT Greengrass 串流訊息結構。

```

{
  "assetId": "string",
  "propertyAlias": "string",
  "propertyId": "string",
  "propertyValues": [
    {
      "quality": "string",
      "timestamp": {
        "offsetInNanos": number,
        "timeInSeconds": number
      },
      "value": {
        "booleanValue": boolean,
        "doubleValue": number,
        "integerValue": number,

```

```

        "stringValue": "string"
      }
    }
  ]
}

```

Note

資料串流訊息必須在其結構propertyAlias中包含 (assetId 和 propertyId) 或。

assetId

(選用) 要更新的資產 ID。

propertyAlias

(選用) 識別 屬性的別名，例如 OPC UA 伺服器資料串流路徑。例如：

```
/company/windfarm/3/turbine/7/temperature
```

如需詳細資訊，請參閱AWS IoT SiteWise 《[使用者指南](#)》中的[管理資料串流](#)。

propertyId

(選用) 此項目的資產屬性 ID。

propertyValues

(必要) 要上傳的屬性值清單。您最多可以指定 10 個propertyValues陣列元素。

quality

(選用) 資產屬性值的品質。

timestamp

(必要) 資產屬性值的時間戳記。

offsetInNanos

(選用) 與 timeInSeconds 的奈秒位移。

timeInSeconds

(必要) 時間戳記日期，以秒為單位，以 Unix epoch 格式顯示。分數奈秒資料由 提供offsetInNanos。

value

(必要) 資產屬性的值。

Note

value 欄位中只能存在下列其中一個值。

booleanValue

(選用) 布林值 (true 或) 類型的資產屬性資料false。

doubleValue

(選用) 類型為雙的資產屬性資料 (浮點編號)。

integerValue

(選用) 類型整數的資產屬性資料 (整數)。

stringValue

(選用) 類型字串的資產屬性資料 (字元序列)。

在 Edge 上設定 AWS IoT SiteWise 邊緣功能

您可以使用 AWS IoT SiteWise Edge 收集和暫時存放資料，以便在本機組織和處理裝置資料。透過啟用邊緣處理，您可以選擇僅將彙總資料傳送至 AWS 雲端，以最佳化頻寬用量和雲端儲存成本。搭配使用 AWS IoT SiteWise 元件 AWS IoT Greengrass，您可以在將資料傳送到 之前收集和處理邊緣的資料 AWS 雲端，或使用 SiteWise Edge APIs 管理內部部署的資料。

資料收集會透過在 上執行的資料套件和 AWS IoT SiteWise 元件進行 AWS IoT Greengrass。

Note

- AWS IoT SiteWise 會在 SiteWise Edge 閘道上保留您的邊緣資料最多 30 天。資料的保留期間取決於裝置的可用磁碟空間。
- 如果您的 SiteWise Edge 閘道已從 中斷連線 AWS 雲端 30 天，[資料處理套件](#)會自動停用。

主題

- [在 SiteWise Edge 中設定邊緣功能](#)

在 SiteWise Edge 中設定邊緣功能

AWS IoT SiteWise 提供下列套件，您的 SiteWise Edge 閘道可用來判斷如何收集和處理您的資料。選取套件以啟用 SiteWise Edge 閘道的邊緣功能。

- 資料收集套件可讓 SiteWise Edge 閘道從多個 OPC UA 伺服器收集資料，然後將資料從邊緣匯出至 AWS 雲端。將資料來源新增至 SiteWise Edge 閘道後，它會變成作用中。
- 資料處理套件可讓您的 SiteWise Edge 閘道在邊緣處理您的設備資料。例如，您可以使用資產模型來計算指標和轉換。如需資產模型和資產的詳細資訊，請參閱 [模型工業資產](#)。

Note

- 資料處理套件僅適用於 x86 平台。

設定邊緣功能

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取您要為其啟用邊緣功能的 SiteWise Edge 閘道。
4. 在 Edge 功能區段中，選擇編輯
5. 在 Edge 功能區段中，選取啟用資料處理套件（會產生額外費用）。
6. （選用）在 Edge LDAP 連線區段中，您可以授予公司目錄中使用者群組存取此 SiteWise Edge 閘道的權限。使用者群組可以使用輕量型目錄存取通訊協定 (LDAP) 登入資料來存取 SiteWise Edge 閘道。然後，他們可以使用 AWS OpsHub AWS IoT SiteWise 應用程式、AWS IoT SiteWise API 操作或其他工具來管理 SiteWise Edge 閘道。如需詳細資訊，請參閱[管理 SiteWise Edge 閘道](#)。

Note

您也可以使用 Linux 或 Microsoft Windows 登入資料來存取 SiteWise Edge 閘道。如需詳細資訊，請參閱[使用 Linux 作業系統登入資料存取 SiteWise Edge 閘道](#)。

- a. 選取已啟用。

- b. 針對提供者名稱，輸入 LDAP 提供者的名稱。
- c. 針對主機名稱或 IP 地址，輸入 LDAP 伺服器的主機名稱或 IP 地址。
- d. 在連接埠中，輸入連接埠號碼。
- e. 針對基本辨別名稱 (DN)，輸入基本的辨別名稱 (DN)。

支援下列屬性類型：commonName (CN)、localityName (L)、stateOrProvinceName (ST)、organizationName (O)、organizationalUnitName (OU)、countryName (C)、streetAddress (STREET)、domainComponent (DC) 和 userid (UID)。

- f. 針對管理員群組 DN，輸入 DN。
- g. 針對使用者群組 DN，輸入 DN。

7. 選擇儲存。

現在您已在 SiteWise Edge 閘道上啟用邊緣功能，您需要為邊緣設定資產模型。您的資產模型邊緣組態會指定資產屬性的計算位置。您可以在邊緣運算所有屬性，也可以分別設定資產模型屬性。資產模型屬性包括[指標](#)、[轉換](#)和[測量](#)。

如需資產屬性的詳細資訊，請參閱 [the section called “定義資料屬性”](#)。

建立資產模型之後，您就可以針對邊緣進行設定。如需為邊緣設定資產模型的詳細資訊，請參閱 [the section called “建立資產模型（主控台）”](#)。

Note

每 10 分鐘會自動同步 Cloud 和 SiteWise Edge 閘道之間的 AWS 資產模型和儀表板。您也可以從本機 SiteWise Edge 閘道應用程式手動同步。

設定 AWS IoT SiteWise 模型和資產的邊緣資料處理

您可以使用 AWS IoT SiteWise Edge 在本機收集、存放、整理和監控設備資料。您可以使用 SiteWise Edge 來建立工業資料模型，並使用 SiteWise Monitor 建立儀表板，讓營運人員在本機視覺化資料。您可以在本機處理資料並將其傳送至 AWS 雲端，或使用 AWS IoT SiteWise API 在內部部署處理。

使用 AWS IoT SiteWise Edge，您可以在本機處理原始資料，並選擇僅將彙總資料傳送至 AWS 雲端，以最佳化頻寬用量和雲端儲存成本。

Note

- AWS IoT SiteWise 會在 SiteWise Edge 閘道上保留您的邊緣資料最多 30 天。資料的保留期間取決於裝置的可用磁碟空間。
- 如果您的 SiteWise Edge 閘道已從 AWS 雲端中斷連線 30 天，[在 SiteWise Edge 中設定 OPC UA 來源](#)則 會自動停用。

設定用於 SiteWise Edge 上資料處理的資產模型

您必須先為邊緣設定資產模型，才能在邊緣處理 SiteWise Edge 閘道資料。您的資產模型邊緣組態會指定資產屬性的計算位置。您可以選擇在邊緣運算所有屬性，並將結果 AWS 傳送至 雲端，或自訂要分別運算每個資產屬性的位置。如需詳細資訊，請參閱[設定 AWS IoT SiteWise 模型和資產的邊緣資料處理](#)。

資產屬性包括指標、轉換和測量：

- 指標是指定期間內資產的彙總資料。您可以使用現有的指標資料來計算新的指標。AWS IoT SiteWise 一律將您的指標傳送至 AWS 雲端，以進行長期儲存。預設情況下，會在 AWS 雲端上 AWS IoT SiteWise 運算指標。您可以設定資產模型在邊緣運算指標。會將處理的結果 AWS IoT SiteWise 傳送至 AWS 雲端。
- 轉換是數學表達式，可將資產屬性的資料點從一種形式對應到另一種形式。轉換可以使用指標做為輸入資料，而且必須計算並存放在與其輸入相同的位置。如果您將指標輸入設定為在邊緣運算，AWS IoT SiteWise 也會在邊緣運算其相關聯的轉換。
- 根據預設，測量會格式化為您的裝置收集並傳送至 AWS 雲端的原始資料。您可以設定資產模型，將此資料存放在本機裝置上。

如需資產屬性的詳細資訊，請參閱 [the section called “定義資料屬性”](#)。

建立資產模型之後，您就可以針對邊緣進行設定。如需為邊緣設定資產模型的詳細資訊，請參閱 [the section called “建立資產模型（主控台）”](#)。

Note

每 10 分鐘會自動同步 Cloud 和 SiteWise Edge 閘道之間的 AWS 資產模型和儀表板。您也可以從 手動同步[管理 SiteWise Edge 閘道](#)。

您可以使用 AWS IoT SiteWise REST APIs 和 AWS Command Line Interface (AWS CLI) 來查詢 SiteWise Edge 閘道的邊緣資料。在查詢 SiteWise Edge 閘道的邊緣資料之前，您必須符合下列先決條件：

- 您必須為 REST APIs 設定登入資料。如需設定登入資料的詳細資訊，請參閱 [the section called “管理閘道”](#)。
- SDK 端點必須指向 SiteWise Edge 閘道的 IP 地址。您可以在 SDK 的文件中找到更多資訊。例如，請參閱《AWS SDK for Java 2.x 開發人員指南》中的[指定自訂端點](#)。
- 您的 SiteWise Edge 閘道憑證必須註冊。您可以在 SDK 的文件中找到有關註冊 SiteWise Edge 閘道憑證的詳細資訊。例如，請參閱《AWS SDK for Java 2.x 開發人員指南》中的在 [Node.js 中註冊憑證套件](#)。

如需使用 查詢資料的詳細資訊 AWS IoT SiteWise，請參閱 [從 查詢資料 AWS IoT SiteWise](#)。

將資料來源新增至 AWS IoT SiteWise Edge 閘道

設定 AWS IoT SiteWise Edge 閘道之後，您可以新增和設定資料來源，以從本機工業設備擷取資料 AWS IoT SiteWise。SiteWise Edge 支援各種通訊協定，包括 OPC UA，以及許多可透過合作夥伴資料來源取得的其他通訊協定。這些來源可讓您的閘道與本機伺服器連線，並擷取工業資料。透過設定資料來源，您可以從各種資料來源擷取資料，然後將資料串流與資產屬性建立關聯，從而實現全面的工業資產建模和資料映射 AWS IoT SiteWise。

主題

- [AWS IoT SiteWise Edge 閘道的 OPC UA 資料來源](#)
- [SiteWise Edge 閘道上的合作夥伴資料來源](#)

AWS IoT SiteWise Edge 閘道的 OPC UA 資料來源

設定 AWS IoT SiteWise Edge 閘道之後，您可以設定資料來源，以便 SiteWise Edge 閘道可以從本機工業設備擷取資料 AWS IoT SiteWise。每個來源代表本機伺服器，例如您的 SiteWise Edge 閘道連線和擷取工業資料串流的 OPC UA 伺服器。如需設定 SiteWise Edge 閘道的詳細資訊，請參閱 [建立自我託管 SiteWise Edge 閘道](#)。

閘道類型、啟用 MQTT 的 V3 閘道與傳統串流 V2 閘道，會影響 OPC UA 資料的處理方式。在 Classic 串流中，V2 閘道會將 OPC UA 資料來源直接新增至閘道 IoT SiteWise 發佈者組態。每個資料來源都會與閘道結合，並針對每個來源個別設定資料路由。相反地，使用啟用 MQTT 的 V3 閘道，OPC UA 資料來源會轉換為 MQTT 主題，並透過集中式目的地進行管理。如需每種類型的詳細資訊，請參閱 [啟](#)

用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge 和 傳統串流、適用於 AWS IoT SiteWise Edge 的 V2 閘道。

 Note

AWS IoT SiteWise 每次新增或編輯來源時，都會重新啟動 SiteWise Edge 閘道。您的 SiteWise Edge 閘道在更新來源組態時不會擷取資料。重新啟動 SiteWise Edge 閘道的時間取決於 SiteWise Edge 閘道來源上的標籤數量。重新啟動時間的範圍可以從幾秒鐘（對於具有數個標籤的 SiteWise Edge 閘道）到幾分鐘（對於具有許多標籤的 SiteWise Edge 閘道）。

建立來源之後，您可以將資料串流與資產屬性建立關聯。如需建立和使用運作資產的詳細資訊，請參閱 [模型工業資產](#)。

您可以檢視 CloudWatch 指標，以驗證資料來源是否已連線 AWS IoT SiteWise。如需詳細資訊，請參閱 [AWS IoT Greengrass Version 2 閘道指標](#)。

目前，AWS IoT SiteWise 支援下列資料來源通訊協定：

- [OPC UA](#) – 用於工業自動化的 machine-to-machine (M2M) 通訊協定。

支援其他工業通訊協定

SiteWise Edge 透過與資料來源合作夥伴整合，支援各種工業通訊協定。這些合作夥伴關係可與超過 200 種不同的通訊協定連線，以容納各種工業系統和裝置。

如需可用資料來源合作夥伴的清單，請參閱 [SiteWise Edge 閘道合作夥伴資料來源選項](#)。

在 SiteWise Edge 中設定 OPC UA 來源

您可以使用 AWS IoT SiteWise 主控台或 SiteWise Edge 閘道功能來定義 OPC UA 來源，並將其新增至 SiteWise Edge 閘道，以代表本機 OPC UA 伺服器。

主題

- [設定 OPC UA 來源（主控台）](#)
- [設定 OPC UA 來源 \(AWS CLI\)](#)

設定 OPC UA 來源（主控台）

您可以使用 主控台，透過下列程序設定 OPC UA 來源。

 Note

警告：重複TQVs 可能會導致重複充電。

使用 AWS IoT SiteWise 主控台設定 OPC UA 來源

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選取 SiteWise Edge 閘道以新增 OPC UA 來源。
4. 選擇新增資料來源。
5. 輸入來源的名稱。
6. 輸入資料來源伺服器的 Local endpoint (本機端點)。端點可以是 IP 地址或主機名稱。您也可以將連接埠號碼新增至本機端點。例如，您的本機端點可能如下所示：
opc.tcp://203.0.113.0:49320
7. (選用) 對於用於選取的節點 ID，請新增節點篩選條件，以限制要擷取至 AWS 雲端的資料串流。根據預設，SiteWise Edge 閘道會使用伺服器的根節點來擷取所有資料串流。您可以使用節點篩選條件來減少 SiteWise Edge 閘道的啟動時間和 CPU 使用量，方法是只包含您建模資料的路徑 AWS IoT SiteWise。根據預設，SiteWise Edge 閘道會上傳開頭為 的所有 OPC UA 路徑/Server/。若要定義 OPC UA 節點篩選條件，您可以使用節點路徑和 * 和 ** 萬用字元。如需詳細資訊，請參閱 [在 SiteWise Edge 中使用 OPC UA 節點篩選條件](#)。
8. 目的地會因啟用 MQTT 的 V3 閘道和 Classic 串流、V2 閘道而有所不同。
 - 傳統蒸汽、V2 閘道目的地與來源有 1：1 的關係。每個來源都會將資料傳送至特定目的地。
 - 啟用 MQTT 的 V3 閘道目的地會分別設定，因為中樞和輻條模型可讓您集中設定和管理不同閘道的多個資料來源。若要在 V3 閘道中設定目的地，請參閱 [了解 AWS IoT SiteWise Edge 目的地](#)。

Classic streams, V2 gateway destinations

- AWS IoT SiteWise 即時 – 選擇此選項以直接將資料傳送至 AWS IoT SiteWise 儲存體。在邊緣即時擷取和監控資料。
- AWS IoT SiteWise 使用 Amazon S3 緩衝 – 以 Parquet 格式將資料傳送至 Amazon S3，然後匯入儲存 AWS IoT SiteWise 體。選擇此選項以批次擷取資料，並以經濟實惠的方式存放歷史資料。您可以設定偏好的 Amazon S3 儲存貯體位置，以及您希望將資料上傳至

Amazon S3 的頻率。您也可以選擇在擷取資料之後如何處理資料 AWS IoT SiteWise。您可以選擇在 AWS IoT SiteWise 和 Amazon S3 中同時提供資料，也可以選擇在匯入 Amazon S3 之後自動將其刪除 AWS IoT SiteWise。

- Amazon S3 儲存貯體是一種預備和緩衝機制，並支援 Parquet 格式的檔案。
- 如果您選取核取方塊將 AWS IoT SiteWise 資料匯入儲存體，資料會先上傳到 Amazon S3，然後再上傳到 AWS IoT SiteWise 儲存體。
 - 如果您選取核取方塊從 Amazon S3 刪除資料，資料會在匯入 SiteWise 儲存體後從 Amazon S3 刪除。
 - 如果您清除核取方塊從 Amazon S3 刪除資料，資料會同時儲存在 Amazon S3 和 SiteWise 儲存中。
- 如果您清除核取方塊將 AWS IoT SiteWise 資料匯入儲存，資料只會儲存在 Amazon S3 中。它不會匯入 SiteWise 儲存體。

[管理資料儲存](#) 如需各種儲存選項 AWS IoT SiteWise 的詳細資訊，請造訪。若要進一步了解定價選項，請參閱 [AWS IoT SiteWise 定價](#)。

- AWS IoT Greengrass 串流管理員 – AWS IoT Greengrass 使用串流管理員將資料傳送至下列 AWS 雲端目的地：中的頻道 AWS IoT Analytics、Amazon Kinesis Data Streams 中的串流、中的資產屬性 AWS IoT SiteWise，或 Amazon Simple Storage Service (Amazon S3) 中的物件。如需詳細資訊，請參閱 [《開發人員指南》中的管理 AWS IoT Greengrass 核心上的資料串流](#)。AWS IoT Greengrass Version 2

輸入 AWS IoT Greengrass 串流的名稱。

MQTT-enabled, V3 gateway destinations

1. [啟用 MQTT 的 V3 Gateways for AWS IoT SiteWise Edge](#) 如需新增相關目的地的資訊，請參閱。
 2. 新增來源目的地後，請返回此程序。
9. 在進階組態窗格中，您可以執行下列動作：
- a. 針對來源伺服器和 SiteWise Edge 閘道之間的連線和傳輸中的資料，選擇訊息安全模式。此欄位是 OPC UA 安全政策和訊息安全模式的組合。選擇您為 OPC UA 伺服器指定的相同安全政策和訊息安全模式。
 - b. 如果您的來源需要身分驗證，請從身分驗證組態清單中選擇 AWS Secrets Manager 秘密。SiteWise Edge 閘道會在連線至此資料來源時，使用此秘密中的身分驗證憑證。您必須將

秘密連接至 SiteWise Edge 閘道的 AWS IoT Greengrass 元件，以將其用於資料來源身分驗證。如需詳細資訊，請參閱[the section called “設定資料來源身分驗證”](#)。

 Tip

您的資料伺服器可能具有稱為 Allow anonymous login (允許匿名登入) 的選項。如果此選項為 Yes (是)，則您的來源不需要身分驗證。

- c. (選用) 您可以選取啟用資料串流字首 - 選用，以啟用資料串流字首。
 - 輸入資料串流字首。SiteWise Edge 閘道會將此字首新增至來自此來源的所有資料串流。使用資料流前置詞以區分來自不同來源但具有相同名稱的資料串流。您帳戶中的每個資料串流應該都有唯一的名稱。
- d. (選用) 選擇資料類型轉換選項，將不支援的 OPC UA 資料類型轉換為字串，然後再將其擷取至 AWS IoT SiteWise。將具有簡單資料類型的陣列值轉換為 JSON 字串，並將 DateTime 資料類型轉換為 ISO 8601 字串。如需詳細資訊，請參閱[轉換不支援的資料類型](#)。
- e. (選用) 對於屬性群組，選擇新增群組。
 - i. 輸入屬性群組的名稱。
 - ii. 對於屬性：
 - 1. 對於節點路徑，新增 OPC UA 節點篩選條件以限制要上傳哪些 OPC UA 路徑 AWS IoT SiteWise。格式類似於用於選取的節點 ID。
 - iii. 對於群組設定，請執行下列動作：
 - 1. 針對資料品質設定，選擇您希望 AWS IoT SiteWise 收集器擷取的資料品質類型。
 - 2. 對於掃描模式設定，使用掃描模式設定標準訂閱屬性。您可以選取訂閱或輪詢。如需掃描模式的詳細資訊，請參閱 [the section called “篩選資料擷取範圍”](#)。

Subscribe

傳送每個資料點

- i. 選擇訂閱並設定下列項目：
 - A. [資料變更觸發](#) – 啟動資料變更提醒的條件。
 - B. [訂閱佇列大小](#) – OPC-UA 伺服器上特定指標的佇列深度，其中監控項目的通知會排入佇列。
 - C. [訂閱發佈間隔](#) – 建立訂閱時指定的發佈週期間隔（以毫秒為單位）。

- D. 快照間隔 - 選用 - 快照頻率逾時設定，以確保 AWS IoT SiteWise Edge 擷取穩定的資料串流。
- E. 掃描速率 - 您希望 SiteWise Edge 閘道讀取暫存器的速率。AWS IoT SiteWise 會自動計算 SiteWise Edge 閘道的最低允許掃描速率。
- F. 時間戳記 - 要包含在 OPC UA 資料點中的時間戳記。您可以使用伺服器時間戳記或裝置的時間戳記。

 Note

使用 IoT SiteWise OPC UA 收集器元件的 2.5.0 版或更新版本。如果您搭配舊版使用時間戳記功能，組態更新會失敗。如需詳細資訊，請參閱[更新 AWS IoT SiteWise 元件的版本](#)。

- ii. 在無效頻帶設定中，設定無效頻帶類型。無效頻帶類型會控制來源傳送至的資料 AWS IoT SiteWise，以及其捨棄的資料。如需無效頻帶設定的詳細資訊，請參閱 [the section called “篩選資料擷取範圍”](#)。
 - 無 - 關聯的伺服器會傳送此屬性群組的所有資料點。
 - 百分比 - 關聯的伺服器只會傳送超出資料範圍指定百分比的資料。此範圍是由伺服器根據為每個節點定義的工程單位最小值和最大值來計算。如果伺服器不支援百分比無效頻帶或缺少定義的工程單位，閘道會使用下列提供的最小值和最大值來計算範圍。
 - 絕對 - 相關聯的伺服器只會傳送落在特定範圍之外的資料。
- A. 將 Deadband 值設定為 Deadband 的資料範圍百分比。
- B. (選用) 使用最小範圍 - 選用和最大範圍 - 選用，指定無效頻帶範圍的最小值和最大值。

Poll

在特定間隔傳送資料點

- 選擇輪詢並設定下列項目：
 - A. 掃描速率 - 您希望 SiteWise Edge 閘道讀取暫存器的速率。AWS IoT SiteWise 會自動計算 SiteWise Edge 閘道的最低允許掃描速率。

- B. 時間戳記 – 要包含在 OPC UA 資料點中的時間戳記。您可以使用伺服器時間戳記或裝置的時間戳記。

 Note

使用 IoT SiteWise OPC UA 收集器元件的 2.5.0 版或更新版本。如果您搭配舊版使用時間戳記功能，組態更新會失敗。如需詳細資訊，請參閱[更新 AWS IoT SiteWise 元件的版本](#)。

 Note

當您在掃描模式設定中選取訂閱時，就會套用無效頻帶設定。

10. 選擇儲存。

設定 OPC UA 來源 (AWS CLI)

您可以使用 為 SiteWise Edge 閘道定義 OPC UA 資料來源 AWS CLI。若要這樣做，請建立 OPC UA 功能組態 JSON 檔案，並使用 [update-gateway-capability-configuration](#) 命令來更新 SiteWise Edge 閘道組態。您必須在單一功能組態中定義所有 OPC UA 來源。

MQTT-enabled, V3 gateway

此功能具有下列命名空間。

- `iotsitewise:opcuacollector:3`

```
{
  "sources": [
    {
      "name": "string",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny" | "X509",
          "certificateBody": "string",
          "certificateChain": "string",
        },
        "endpointUri": "string",
      }
    }
  ]
}
```



```

    "securityPolicy": "NONE" | "BASIC128_RSA15" | "BASIC256" | "BASIC256_SHA256"
    | "AES128_SHA256_RSA0AEP" | "AES256_SHA256_RSAPSS",
    "messageSecurityMode": "NONE" | "SIGN" | "SIGN_AND_ENCRYPT",
    "identityProvider": {
        "type": "Anonymous" | "Username",
        "usernameSecretArn": "string"
    },
    "nodeFilterRules": [
        {
            "action": "INCLUDE",
            "definition": {
                "type": "OpcUaRootPath",
                "rootPath": "string"
            }
        }
    ],
    "measurementDataStreamPrefix": "string",
    "typeConversions": {
        "array": "JsonArray",
        "datetime": "ISO8601String"
    },
    "destination": {
        {
            "type": "MQTT"
        }
    },
    "propertyGroups": [
        {
            "name": "string",
            "nodeFilterRuleDefinitions": [
                {
                    "type": "OpcUaRootPath",
                    "rootPath": "string"
                }
            ]
        },
        "deadband": {
            "type": "PERCENT" | "ABSOLUTE",
            "value": double,
            "eguMin": double,
            "eguMax": double,
            "timeoutMilliseconds": integer
        },
        "scanMode": {

```

```

        "type": "EXCEPTION" | "POLL",
        "rate": integer,
        "timestampToReturn": "SOURCE_TIME" | "SERVER_TIME"
    },
    "dataQuality": {
        "allowGoodQuality": true | false,
        "allowBadQuality": true | false,
        "allowUncertainQuality": true | false
    },
    "subscription": {
        "dataChangeTrigger": "STATUS" | "STATUS_VALUE" |
"STATUS_VALUE_TIMESTAMP",
        "queueSize": integer,
        "publishingIntervalMilliseconds": integer,
        "snapshotFrequencyMilliseconds": integer
    }
}
]
}
]
}

```

Classic streams, V2 gateway

此功能具有下列命名空間。

- iotsitewise:opcuacollector:2

請求語法

```

{
  "sources": [
    {
      "name": "string",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny" | "X509",
          "certificateBody": "string",
          "certificateChain": "string",
        },
        "endpointUri": "string",
        "securityPolicy": "NONE" | "BASIC128_RSA15" | "BASIC256" | "BASIC256_SHA256"
| "AES128_SHA256_RSA0AEP" | "AES256_SHA256_RSAPSS",

```

```

"messageSecurityMode": "NONE" | "SIGN" | "SIGN_AND_ENCRYPT",
"identityProvider": {
  "type": "Anonymous" | "Username",
  "usernameSecretArn": "string"
},
"nodeFilterRules": [
  {
    "action": "INCLUDE",
    "definition": {
      "type": "OpcUaRootPath",
      "rootPath": "string"
    }
  }
],
"measurementDataStreamPrefix": "string",
"typeConversions": {
  "array": "JsonArray",
  "datetime": "ISO8601String"
},
"destination": {
  "type": "StreamManager",
  "streamName": "string",
  "streamBufferSize": integer,
},
"propertyGroups": [
  {
    "name": "string",
    "nodeFilterRuleDefinitions": [
      {
        "type": "OpcUaRootPath",
        "rootPath": "string"
      }
    ],
    "deadband": {
      "type": "PERCENT" | "ABSOLUTE",
      "value": double,
      "eguMin": double,
      "eguMax": double,
      "timeoutMilliseconds": integer
    },
    "scanMode": {
      "type": "EXCEPTION" | "POLL",
      "rate": integer,

```

```

        "timestampToReturn": "SOURCE_TIME" | "SERVER_TIME"
    },
    "dataQuality": {
        "allowGoodQuality": true | false,
        "allowBadQuality": true | false,
        "allowUncertainQuality": true | false
    },
    "subscription": {
        "dataChangeTrigger": "STATUS" | "STATUS_VALUE" |
"STATUS_VALUE_TIMESTAMP",
        "queueSize": integer,
        "publishingIntervalMilliseconds": integer,
        "snapshotFrequencyMilliseconds": integer
    }
}
]
}
]
}

```

請求內文

sources

OPC UA 來源定義結構的清單，每個結構都包含下列資訊：

name

來源的唯一且易記名稱。

endpoint

包含下列資訊的端點結構：

certificateTrust

包含下列資訊的憑證信任政策結構：

type

來源的憑證信任模式。選擇下列其中一項：

- **TrustAny** – SiteWise Edge 閘道會在連線到 OPC UA 來源時信任任何憑證。

- X509 – SiteWise Edge 閘道會在連線到 OPC UA 來源時信任 X.509 憑證。如果您選擇此選項，則必須在 `certificateTrust` 中定義 `certificateBody`。您也可以 `certificateTrust` 在中定義 `certificateChain`。

`certificateBody`

(選用性) X.509 憑證的主體。

如果您在 `certificateTrust` 中為 `type` 選擇 X509，則此欄位為必要。

`certificateChain`

(選用) X.509 憑證的信任鏈。

僅當您在 `certificateTrust` 中為 `type` 選擇 X509 的情況下，才會使用此欄位。

`endpointUri`

OPC UA 來源的本機端點。例如，您的本機端點可能看起來會像 `opc.tcp://203.0.113.0:49320`。

`securityPolicy`

要使用的安全政策，讓您可以保護從 OPC UA 來源讀取的訊息。選擇下列其中一項：

- NONE – SiteWise Edge 閘道無法保護來自 OPC UA 來源的訊息。我們建議您選擇不同的安全政策。如果您選擇此選項，您也必須為 `messageSecurityMode` 選擇 NONE。
- BASIC256_SHA256 – Basic256Sha256安全政策。
- AES128_SHA256_RSA0AEP – Aes128_Sha256_Rsa0aep安全政策。
- AES256_SHA256_RSAPSS – Aes256_Sha256_RsaPss安全政策。
- BASIC128_RSA15 – (已棄用) OPC UA 規格中的Basic128Rsa15安全政策已棄用，因為它不再被視為安全。我們建議您選擇不同的安全政策。如需詳細資訊，請參閱[設定檔 SecurityPolicy – Basic128Rsa15](#)。
- BASIC256 – (已棄用) OPC UA 規格中的Basic256安全政策已棄用，因為它不再被視為安全。我們建議您選擇不同的安全政策。如需詳細資訊，請參閱[SecurityPolicy – Basic256](#)。

 Important

如果您選擇 以外的安全政策NONE，您必須SIGN_AND_ENCRYPT為 選擇 SIGN或 `messageSecurityMode`。您也必須將來源伺服器設定為信任 SiteWise Edge 閘

道。如需詳細資訊，請參閱[設定 OPC UA 伺服器以信任 AWS IoT SiteWise Edge 閘道](#)。

messageSecurityMode

用於保護 OPC UA 來源連線的訊息安全模式。選擇下列其中一項：

- NONE – SiteWise Edge 閘道無法保護與 OPC UA 來源的連線。我們建議您選擇不同的訊息安全模式。如果您選擇此選項，您也必須為 securityPolicy 選擇 NONE。
- SIGN – SiteWise Edge 閘道與 OPC UA 來源之間傳輸中的資料已簽署但未加密。
- SIGN_AND_ENCRYPT – 閘道與 OPC UA 來源之間傳輸中的資料會經過簽署和加密。

Important

如果您選擇 以外的訊息安全模式 NONE，則必須選擇 securityPolicy 以外的 NONE。您也必須將來源伺服器設定為信任 SiteWise Edge 閘道。如需詳細資訊，請參閱[設定 OPC UA 伺服器以信任 AWS IoT SiteWise Edge 閘道](#)。

identityProvider

包含下列資訊的身分提供者結構：

type

來源所需的身分驗證登入資料類型。選擇下列其中一項：

- Anonymous – 來源不需要身分驗證即可連線。
- Username – 來源需要使用者名稱和密碼才能連線。如果您選擇此選項，則必須在 identityProvider 中定義 usernameSecretArn。

usernameSecretArn

(選用) AWS Secrets Manager 秘密的 ARN。SiteWise Edge 閘道會在連線至此來源時，使用此秘密中的身分驗證憑證。您必須將秘密連接到 SiteWise Edge 閘道的 IoT SiteWise 連接器，以將其用於來源身分驗證。如需詳細資訊，請參閱[設定 SiteWise Edge 的資料來源身分驗證](#)。

如果您在 identityProvider 中為 type 選擇 Username，則此欄位為必要。

nodeFilterRules

定義傳送至 AWS 雲端之 OPC UA 資料串流路徑的節點篩選規則結構清單。您可以使用節點篩選條件來減少 SiteWise Edge 閘道的啟動時間和 CPU 使用量，方法是只包含您建模資料的路徑 AWS IoT SiteWise。根據預設，SiteWise Edge 閘道會上傳所有 OPC UA 路徑，但以開頭的路徑除外/Server/。若要定義 OPC UA 節點篩選條件，您可以使用節點路徑和 * 和 ** 萬用字元。如需詳細資訊，請參閱[在 SiteWise Edge 中使用 OPC UA 節點篩選條件](#)。

清單中的每個結構都必須包含下列資訊：

action

此節點篩選規則的動作。您可以選擇下列選項：

- INCLUDE – SiteWise Edge 閘道僅包含符合此規則的資料串流。

definition

包含下列資訊的節點篩選規則結構：

type

此規則的節點篩選路徑類型。您可以選擇下列選項：

- OpcUaRootPath – SiteWise Edge 閘道會根據 OPC UA 路徑階層的根目錄評估此節點篩選路徑。

rootPath

針對 OPC UA 路徑階層的根評估的節點篩選路徑。此路徑必須以開頭/。

measurementDataStreamPrefix

從來源附加至所有資料串流的字串。SiteWise Edge 閘道會將此字首新增至來自此來源的所有資料串流。使用資料流前置詞以區分來自不同來源但具有相同名稱的資料串流。您帳戶中的每個資料串流應該都有唯一的名稱。

typeConversions

不支援的 OPC UA 資料類型可用的轉換類型。每個資料類型都會轉換為字串。如需詳細資訊，請參閱[轉換不支援的資料類型](#)。

array

轉換為字串的簡單陣列資料類型。您可以選擇下列選項：

- `JSONArray` – 表示您選擇將簡易陣列資料類型轉換為字串。

`datetime`

轉換為字串的 `DateTime` 資料類型。您可以選擇下列選項：

- `ISO8601String` – 表示您選擇將 ISO 8601 資料類型轉換為字串。

`destination`

OPC UA 標籤目的地的組態。傳統串流、v2 和啟用 MQTT 的 V3 閘道具有不同的目的地組態。

`type`

目的地的類型。

`streamName` – 僅適用於 Classic 串流、V2 閘道

串流的名稱。串流名稱應該是唯一的。

`streamBufferSize` – 僅適用於 Classic 串流、V2 閘道

串流的緩衝區大小。這對於管理來自 OPC UA 來源的資料流程非常重要。

`propertyGroups`

(選用) 定義通訊協定 `scanMode` 請求的 `deadband` 和 的屬性群組清單。

`name`

屬性群組的名稱。這應該是唯一的識別符。

`deadband`

該 `deadband` 值定義資料點值的最低變更，必須在資料傳送到雲端之前發生。其中包含下列資訊：

`type`

支援的無效頻帶類型。您可以選擇下列選項：

- `ABSOLUTE` – 固定值，指定將資料點視為足夠重要以傳送至雲端所需的最低絕對變更。
- `PERCENT` – 動態值，以上次傳送資料點值的百分比指定所需的最低變更。當資料值隨時間大幅變化時，這種類型的無效頻帶很有用。

`value`

無效頻帶的值。當 `type` 為 `ABSOLUTE`，此值為無單位雙。當 `type` 為 `PERCENT`，此值是介於 1 和 之間的兩倍 100。

eguMin

(選用) 使用PERCENT無效頻帶時的工程單位最小值。如果 OPC UA 伺服器未設定工程單位，您可以設定此項目。

eguMax

(選用) 使用PERCENT無效頻帶時的工程單位上限。如果 OPC UA 伺服器未設定工程單位，您可以設定此項目。

timeoutMilliseconds

逾時前的持續時間，以毫秒為單位。最小值為 100。

scanMode

包含下列資訊的scanMode結構：

type

支援的 類型scanMode。接受的值為 POLL和 EXCEPTION。

rate

掃描模式的取樣間隔。

timestampToReturn

時間戳記的來源。您可以選擇下列選項：

- SOURCE_TIME – 使用您裝置的時間戳記。
- SERVER_TIME – 使用您伺服器的時間戳記。

Note

TimestampToReturn 搭配 IoT SiteWise OPC UA 收集器元件的 2.5.0 版或更新版本使用。如果您將此功能與舊版搭配使用，組態更新會失敗。如需詳細資訊，請參閱[更新 AWS IoT SiteWise 元件的版本](#)。

nodeFilterRuleDefinitions

(選用) 屬性群組中要包含的節點路徑清單。屬性群組不能重疊。如果您未指定此欄位的值，則群組會包含根目錄下的所有路徑，而且您無法建立其他屬性群組。nodeFilterRuleDefinitions 結構包含以下資訊：

type

OpcUaRootPath 是唯一支援的類型。這會指定的值rootPath是相對於 OPC UA 瀏覽空間根目錄的路徑。

rootPath

逗號分隔清單，指定要包含在屬性群組中的路徑（相對於根）。

Classic 串流、V2 閘道 (AWS CLI) 的其他功能組態範例

下列範例會從存放在 JSON 檔案中的承載定義 OPC UA SiteWise Edge 閘道功能組態。

```
aws iotsitewise update-gateway-capability-configuration \  
--capability-namespace "iotsitewise:opcuacollector:2" \  
--capability-configuration file://opc-ua-configuration.json
```

Example : OPC UA 來源組態

下列opc-ua-configuration.json檔案定義了基本、不安全的 OPC UA 來源組態。

```
{  
  "sources": [  
    {  
      "name": "Wind Farm #1",  
      "endpoint": {  
        "certificateTrust": {  
          "type": "TrustAny"  
        },  
        "endpointUri": "opc.tcp://203.0.113.0:49320",  
        "securityPolicy": "NONE",  
        "messageSecurityMode": "NONE",  
        "identityProvider": {  
          "type": "Anonymous"  
        },  
        "nodeFilterRules": []  
      },  
      "measurementDataStreamPrefix": ""  
    },  
  ],  
}
```

Example : OPC UA 來源組態與定義的屬性群組

下列opc-ua-configuration.json檔案使用定義的屬性群組來定義基本、不安全的 OPC UA 來源組態。

```
{
  "sources": [
    {
      "name": "source1",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny"
        },
        "endpointUri": "opc.tcp://10.0.0.9:49320",
        "securityPolicy": "NONE",
        "messageSecurityMode": "NONE",
        "identityProvider": {
          "type": "Anonymous"
        },
        "nodeFilterRules": [
          {
            "action": "INCLUDE",
            "definition": {
              "type": "OpcUaRootPath",
              "rootPath": "/Utilities/Tank"
            }
          }
        ]
      },
      "measurementDataStreamPrefix": "propertyGroups",
      "propertyGroups": [
        {
          "name": "Deadband_Abs_5",
          "nodeFilterRuleDefinitions": [
            {
              "type": "OpcUaRootPath",
              "rootPath": "/Utilities/Tank/Temperature/TT-001"
            },
            {
              "type": "OpcUaRootPath",
              "rootPath": "/Utilities/Tank/Temperature/TT-002"
            }
          ],
          "deadband": {
```

```

        "type": "ABSOLUTE",
        "value": 5.0,
        "timeoutMilliseconds": 120000
    },
    {
        "name": "Polling_10s",
        "nodeFilterRuleDefinitions": [
            {
                "type": "OpcUaRootPath",
                "rootPath": "/Utilities/Tank/Pressure/PT-001"
            }
        ],
        "scanMode": {
            "type": "POLL",
            "rate": 10000
        }
    },
    {
        "name": "Percent_Deadband_Timeout_90s",
        "nodeFilterRuleDefinitions": [
            {
                "type": "OpcUaRootPath",
                "rootPath": "/Utilities/Tank/Flow/FT-*"
            }
        ],
        "deadband": {
            "type": "PERCENT",
            "value": 5.0,
            "eguMin": -100,
            "eguMax": 100,
            "timeoutMilliseconds": 90000
        }
    }
]
}

```

Example : OPC UA 來源組態與屬性

的下列 JSON 範例 `opc-ua-configuration.json` 定義具有下列屬性的 OPC UA 來源組態：

- 信任任何憑證。

- 使用BASIC256安全政策來保護訊息。
- 使用 SIGN_AND_ENCRYPT 模式來保護連線。
- 使用存放在 Secrets Manager 秘密中的身分驗證憑證。
- 篩選資料串流，路徑以 /WindFarm/2/WindTurbine/ 開頭的資料串流除外。
- 將 /Washington 新增至每個資料串流路徑的開頭，以區分此「風力發電廠 #2」和另一個區域中的「風力發電廠 #2」。

```
{
  "sources": [
    {
      "name": "Wind Farm #2",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny"
        },
        "endpointUri": "opc.tcp://203.0.113.1:49320",
        "securityPolicy": "BASIC256",
        "messageSecurityMode": "SIGN_AND_ENCRYPT",
        "identityProvider": {
          "type": "Username",
          "usernameSecretArn":
            "arn:aws:secretsmanager:region:123456789012:secret:greenrass-windfarm2-auth-1ABCDE"
        },
        "nodeFilterRules": [
          {
            "action": "INCLUDE",
            "definition": {
              "type": "OpcUaRootPath",
              "rootPath": "/WindFarm/2/WindTurbine/"
            }
          }
        ]
      },
      "measurementDataStreamPrefix": "/Washington"
    }
  ]
}
```

Example : OPC UA 來源組態與憑證信任

的下列 JSON 範例 `opc-ua-configuration.json` 定義具有下列屬性的 OPC UA 來源組態：

- 信任指定的 X.509 憑證。
- 使用 BASIC256 安全政策來保護訊息。
- 使用 SIGN_AND_ENCRYPT 模式來保護連線。

```
{
  "sources": [
    {
      "name": "Wind Farm #3",
      "endpoint": {
        "certificateTrust": {
          "type": "X509",
          "certificateBody": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w
0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZ
WF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAwTC0lBTSBDb25zb2x1MRIw
EAYDVQQDEw1UZXRhbnQ2L2sYWMxH2AdBgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5
jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTEwNDI1MjA0NTIxWjCBiDELMAkGA1UEBh
MCVVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBb
WF6b24xFDASBgNVBAwTC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnQ2L2sYWMx
H2AdBgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5jb20wZ8wDQYJKoZIhvcNAQEE
BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLygVI
k60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
ITx0USQv7c7ugFFDzQGBzZswY6786m86gpEIbb30hjZnzcvcQAaRHhd1QWIMm2nr
AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEATCu4nUhVVxYUntneD9+h8Mg9q6q+auN
KyExzyLwax1Aoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJI1J00zbhNYS5f6Guo
EDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
          "certificateChain": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w
0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZ
WF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAwTC0lBTSBDb25zb2x1MRIw
EAYDVQQDEw1UZXRhbnQ2L2sYWMxH2AdBgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5
jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTEwNDI1MjA0NTIxWjCBiDELMAkGA1UEBh
MCVVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBb
WF6b24xFDASBgNVBAwTC0lBTSBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnQ2L2sYWMx
H2AdBgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5jb20wZ8wDQYJKoZIhvcNAQEE
BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLygVI
```

```

k60CpiwsZ3G93vUEI03IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
ITx0USQv7c7ugFFDzQGBzZswY6786m86gpEIbb30hjZnzcvQAaRHhd1QWIMm2nr
AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJI1J00zbhNYS5f6Guo
EDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
3rrszlaEXAMPLE=
-----END CERTIFICATE-----"
    },
    "endpointUri": "opc.tcp://203.0.113.2:49320",
    "securityPolicy": "BASIC256",
    "messageSecurityMode": "SIGN_AND_ENCRYPT",
    "identityProvider": {
        "type": "Anonymous"
    },
    "nodeFilterRules": []
  },
  "measurementDataStreamPrefix": ""
}
]
}

```

設定 OPC UA 伺服器以信任 AWS IoT SiteWise Edge 閘道

如果您在設定 OPC UA 來源時選擇 `messageSecurityMode` 非無，則必須啟用來源伺服器信任 AWS IoT SiteWise Edge 閘道。SiteWise Edge 閘道會產生來源伺服器可能需要的憑證。程序會根據您的來源伺服器而有所不同。如需詳細資訊，請參閱您伺服器的文件。

下列程序概述基本步驟。

讓 OPC UA 伺服器信任 SiteWise Edge 閘道

1. 開啟用於設定 OPC UA 伺服器的界面。
2. 輸入 OPC UA 伺服器管理員的使用者名稱和密碼。
3. 在界面中尋找信任的用戶端，然後選擇 AWS IoT SiteWise 閘道用戶端。
4. 選擇 Trust (信任)。

匯出 OPC UA 用戶端憑證

有些 OPC UA 伺服器需要存取 OPC UA 用戶端憑證檔案，才能信任 SiteWise Edge 閘道。如果這適用於您的 OPC UA 伺服器，您可以使用下列程序從 SiteWise Edge 閘道匯出 OPC UA 用戶端憑證。然後，您可以在 OPC UA 伺服器上匯入憑證。

匯出來源的 OPC UA 用戶端憑證檔案

1. 執行下列命令，以變更至包含憑證檔案的目錄。將 *sitewise-work* 取代為 *aws.iot.SiteWiseEdgeCollectorOpcua* Greengrass 工作資料夾的本機儲存路徑，並將 *source-name* 取代為資料來源的名稱。

根據預設，Greengrass 工作資料夾在 Linux 上為 */greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua*，在上為 *C#/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua* Microsoft Windows。

```
cd /sitewise-work/source-name/opcua-certificate-store
```

2. 此來源的 SiteWise Edge 閘道 OPC UA 用戶端憑證位於 *aws-iot-opcua-client.pfx* 檔案中。

執行下列命令，將憑證匯出至名為 *aws-iot-opcua-client-certificate.pem* 的 .pem 檔案。

```
keytool -exportcert -v -alias aws-iot-opcua-client -keystore aws-iot-opcua-client.pfx -storepass amazon -storetype PKCS12 -rfc > aws-iot-opcua-client-certificate.pem
```

3. 將憑證檔案 *aws-iot-opcua-client-certificate.pem*，從 SiteWise Edge 閘道傳輸到 OPC UA 伺服器。

若要這麼做，您可以使用一般軟體，例如 *scp* 程式來傳輸使用 SSH 通訊協定的檔案。如需詳細資訊，請參閱 Wikipedia 上的[安全複製](#)。

Note

如果您的 SiteWise Edge 閘道在 Amazon Elastic Compute Cloud (Amazon EC2) 上執行，而且您第一次連線到它，您必須設定先決條件才能連線。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的[使用 SSH 連線至 Linux 執行個體](#)。

4. 在 OPC UA 伺服器上匯入憑證檔案 `aws-iot-opcua-client-certificate.pem`，以信任 SiteWise Edge 閘道。步驟可能有所不同，取決於您使用的來源伺服器。請參閱伺服器文件。

使用 OPC UA 篩選資料擷取範圍

您可以使用掃描模式和無效頻帶範圍，控制使用 OPC UA 來源擷取資料的方式。這些功能可讓您控制要擷取的資料類型，以及伺服器和 SiteWise Edge 閘道交換此資訊的方式和時間。

根據品質收集或篩選資料

您可以設定資料品質設定，以控制從 OPC UA 來源收集哪些資料。資料來源在傳送時包含品質評分做為中繼資料。您可以選取下列其中一個或所有選項：

- Good
- Bad
- Uncertain

處理 NaN 或 null 值

SiteWise Edge 支援收集和處理 NaN 和 null 值。

- NaN（非數字）：代表未定義或無法表示的數值結果。
- Null：表示遺失資料。

IoT SiteWise OPC UA 收集器會擷取具有 BAD 或 UNCERTAIN 品質的 NaN 和 Null 值。這些特殊值會寫入本機串流，讓資料收集更為完整。

使用掃描模式控制資料收集頻率

您可以設定 OPC UA 掃描模式，以控制從 OPC UA 來源收集資料的方式。您可以選擇訂閱或輪詢模式。

- 訂閱模式 – OPC UA 來源會依掃描速率定義的頻率，收集資料以傳送至 SiteWise Edge 閘道。伺服器只會在值變更時傳送資料，因此這是 SiteWise Edge 閘道接收資料的最大頻率。
- 輪詢模式 – SiteWise Edge 閘道會以掃描速率定義的設定頻率輪詢 OPC UA 來源。無論值是否已變更，伺服器都會傳送資料，因此 SiteWise Edge 閘道一律會在此間隔接收資料。

Note

輪詢模式選項會覆寫此來源的無效頻帶設定。

使用無效頻帶範圍篩選 OPC UA 資料擷取

您可以將無效頻帶套用至 OPC UA 來源屬性群組，以篩選和捨棄特定資料，而不是將其傳送至 AWS 雲端。無效頻帶會指定 OPC UA 來源傳入資料值中預期波動的時段。如果值落在此視窗中，您的 OPC UA 伺服器不會將其傳送至 AWS 雲端。您可以使用無效頻帶篩選來減少您正在處理和傳送至 AWS 雲端的資料量。若要了解如何設定 SiteWise Edge 閘道的 OPC UA 來源，請參閱 [AWS IoT SiteWise Edge 閘道的 OPC UA 資料來源](#)。

Note

伺服器會刪除位於無效頻帶所指定視窗內的所有資料。您無法復原此捨棄的資料。

無效頻帶的類型

您可以為 OPC UA 伺服器屬性群組指定兩種類型的無效頻帶。這些選項可讓您選擇傳送至 AWS 雲端的資料量，以及捨棄的資料量。

- 百分比 – 您可以使用測量值中的預期波動百分比來指定視窗。伺服器會從此百分比計算確切的時段，並將超過的資料傳送至超出該時段的 AWS 雲端。例如，在感應器上指定 2% 的無效頻帶值，範圍從華氏 -100 度到華氏 +100 度，指示伺服器在值變更 4 度以上時，將資料傳送至 AWS 雲端。

Note

如果您的來源伺服器未定義工程單位，您可以選擇為此時段指定最小和最大無效頻帶值。如果未提供工程單位範圍，OPC UA 伺服器會預設為測量資料類型的完整範圍。

- 絕對 – 您可以使用確切的單位來指定視窗。例如，在感應器上指定無效頻帶值 2，會指示伺服器在其值變更至少 2 個單位時，將資料傳送至 AWS 雲端。您可以在正常操作期間定期預期波動的動態環境中使用絕對無效頻帶。

Deadband 逾時

您可以選擇性地設定無效頻帶逾時設定。在此逾時之後，OPC UA 伺服器會傳送目前的測量值，即使它在預期的無效頻帶波動內。您可以使用逾時設定來確保始終 AWS IoT SiteWise 擷取穩定的資料串流，即使值未超過定義的無效頻帶時段也一樣。

在 SiteWise Edge 中使用 OPC UA 節點篩選條件

當您定義 SiteWise Edge 閘道的 OPC UA 資料來源時，您可以定義節點篩選條件。節點篩選條件可讓您限制 SiteWise Edge 閘道傳送至雲端的資料串流路徑。您可以使用節點篩選條件來減少 SiteWise Edge 閘道的啟動時間和 CPU 使用量，方法是只包含您建模資料的路徑 AWS IoT SiteWise。根據預設，SiteWise Edge 閘道會上傳所有 OPC UA 路徑，但以開頭的路徑除外/Server/。您可以在節點篩選器中使用 * 和 ** 萬用字元，將多個資料串流路徑包含在一個篩選器中。若要了解如何設定 SiteWise Edge 閘道的 OPC UA 來源，請參閱 [AWS IoT SiteWise Edge 閘道的 OPC UA 資料來源](#)。

Note

AWS IoT SiteWise 每次新增或編輯來源時，都會重新啟動 SiteWise Edge 閘道。您的 SiteWise Edge 閘道在更新來源組態時不會擷取資料。重新啟動 SiteWise Edge 閘道的時間取決於 SiteWise Edge 閘道來源上的標籤數量。重新啟動時間的範圍可以從幾秒鐘（對於具有幾個標籤的 SiteWise Edge 閘道）到幾分鐘（對於具有許多標籤的 SiteWise Edge 閘道）。

下表列出可用來篩選 OPC UA 資料來源的萬用字元。

OPC UA 節點篩選條件萬用字元

萬用字元	描述
*	比對資料串流路徑中的單一層級。
**	比對資料串流路徑中的多個層級。

Note

如果您使用廣泛的篩選條件設定來源，然後稍後將來源變更為使用更嚴格的篩選條件，會 AWS IoT SiteWise 停止儲存不符合新篩選條件的資料。

Example：使用節點篩選條件的案例

考慮以下的假設性資料串流：

- /WA/Factory 1/Line 1/PLC1
- /WA/Factory 1/Line 1/PLC2
- /WA/Factory 1/Line 2/Counter1
- /WA/Factory 1/Line 2/PLC1
- /OR/Factory 1/Line 1/PLC1
- /OR/Factory 1/Line 2/Counter2

您可以使用先前的資料串流定義節點篩選條件，以限制要從 OPC UA 來源包含的資料。

- 若要選取此範例中的所有節點，請使用 /或 /**/。使用 ** 萬用字元可以包含多個目錄或資料夾。
- 若要選取所有 PLC 資料串流，請使用 /*/*/PLC* 或 /**/PLC*。
- 若要選取此範例中的所有計數器，請使用 /**/Counter*或 /*/*/Counter*。
- 若要從 Line 2 中選取所有計數器，請使用 /**/Line 2/Counter*。

轉換不支援的資料類型

針對簡單陣列和 DateTime 資料類型，選擇性地 AWS IoT SiteWise 在 中啟用資料類型轉換。AWS IoT SiteWise 不支援所有 OPC UA 資料類型。當您將不支援的資料傳送到 AWS IoT Greengrass 資料串流時，該資料會遺失。不過，透過將不支援的原生資料類型轉換為字串，您可以將資料擷取至 AWS IoT SiteWise，而不是將其捨棄。AWS IoT SiteWise 會序列化轉換後的資料，以便您稍後可以視需要使用自己的函數，將字串轉換回其原始資料類型下游。

您可以隨時更新資料來源的資料類型轉換設定，而且每個資料來源都可以有自己的設定。

當您在 中新增資料來源時 AWS IoT SiteWise 主控台，進階組態中的資料類型轉換下有兩個核取方塊。您可以指出要轉換為字串的資料類型。

此外，IoT SiteWise OPC UA 收集器可以在邊緣接受 NaN 或 null 值。

- 將具有簡單資料類型的陣列值轉換為 JSON 字串
- 將 DateTime 值轉換為 ISO 8601 字串

先決條件

- 使用 [IoT SiteWise OPC UA 收集器的 2.5.0 版或更新版本](#)。

限制

這些是將 OPC UA 資料類型轉換為 中字串的限制 AWS IoT SiteWise。

- 不支援複雜的資料類型轉換。
- 轉換後的字串限制為 1024 個位元組。如果字串超過 1024 個位元組，則字串會遭到 拒絕 AWS IoT SiteWise。

設定 SiteWise Edge 的資料來源身分驗證

如果您的 OPC UA 伺服器需要身分驗證憑證才能連線，您可以使用 來 AWS Secrets Manager 建立秘密並將其部署到您的 SiteWise Edge 閘道。會在裝置上 AWS Secrets Manager 加密秘密，以保持使用者名稱和密碼的安全，直到您需要使用它們為止。如需 AWS IoT Greengrass 秘密管理員元件的詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的[秘密管理員](#)。

如需管理 Secrets Manager 秘密存取權的資訊，請參閱：

- [誰具有您的 AWS Secrets Manager 秘密的許可](#)。
- [判斷 帳戶內是否允許或拒絕請求](#)。

步驟 1：建立來源身分驗證秘密

您可以使用 AWS Secrets Manager 為您的資料來源建立身分驗證秘密。在秘密中，定義包含資料來源身分驗證詳細資訊的 **username** 和 **password** 鍵值對。

若要建立機密 (控制台)

1. 導覽至 [AWS Secrets Manager 主控台](#)。
2. 選擇儲存新機密。
3. 在秘密類型下，選擇其他類型的秘密。
4. 在鍵/值對下，執行下列動作：
 1. 在第一個輸入方塊中，輸入 **username**，然後在第二個輸入方塊中輸入使用者名稱。
 2. 選擇新增列。

3. 在第一個輸入方塊中，輸入 **password**，然後在第二個輸入方塊中輸入密碼。
5. 針對加密金鑰，選取 `aws/secretsmanager`，然後選擇下一步。
6. 在儲存新的秘密頁面上，輸入秘密名稱。
7. （選用）輸入可協助您識別此秘密的描述，然後選擇下一步。
8. （選用）在儲存新的秘密頁面上，開啟自動輪換。如需詳細資訊，請參閱AWS Secrets Manager 《使用者指南》中的[輪換秘密](#)。
9. 指定輪換排程。
10. 選擇可輪換此秘密的 Lambda 函數，然後選擇下一步。
11. 檢閱您的秘密組態，然後選擇儲存。

若要授權 SiteWise Edge 閘道與 互動 AWS Secrets Manager，SiteWise Edge 閘道的 IAM 角色必須允許 `secretsmanager:GetSecretValue` 動作。您可以使用 Greengrass 核心裝置來搜尋 IAM 政策。如需更新 IAM 政策的詳細資訊，請參閱AWS Identity and Access Management 《使用者指南》中的[編輯 IAM 政策](#)。

Example 政策

將 *secret-arn* 取代為您在上一步驟中建立之秘密的 Amazon Resource Name (ARN)。如需如何取得秘密 ARN 的詳細資訊，請參閱AWS Secrets Manager 《使用者指南》中的在 [中尋找秘密 AWS Secrets Manager](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "secret-arn"
      ]
    }
  ]
}
```

步驟 2：將秘密部署到您的 SiteWise Edge 閘道裝置

您可以使用 AWS IoT SiteWise 主控台將秘密部署到您的 SiteWise Edge 閘道。

部署秘密（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Gateways (網際網路閘道)。
3. 從閘道清單中，選擇目標 SiteWise Edge 閘道。
4. 在閘道組態區段中，選擇 Greengrass 核心裝置連結以開啟與 SiteWise Edge 閘道相關聯的 AWS IoT Greengrass 核心。
5. 在導覽窗格中，選擇部署。
6. 選擇目標部署，然後選擇修訂。
7. 在指定目標頁面上，選擇下一步。
8. 在選取元件頁面的公有元件區段中，關閉僅顯示選取的元件。
9. 搜尋並選擇 `aws.greengrass.SecretManager` 元件，然後選擇下一步。
10. 從選取的元件清單中，選擇 `aws.greengrass.SecretManager` 元件，然後選擇設定元件。
11. 在要合併的組態欄位中，新增下列 JSON 物件。

Note

將 `secret-arn` 取代為您在上一步驟中建立之秘密的 ARN。如需如何取得秘密 ARN 的詳細資訊，請參閱 AWS Secrets Manager 《使用者指南》中的在 [中尋找秘密 AWS Secrets Manager](#)。

```
{
  "cloudSecrets":[
    {
      "arn":"secret-arn"
    }
  ]
}
```

12. 選擇確認。
13. 選擇 Next (下一步)。

14. 在設定進階設定頁面上，選擇下一步。
15. 檢閱您的部署組態，然後選擇部署。

步驟 3：新增身分驗證組態

您可以使用 AWS IoT SiteWise 主控台將身分驗證組態新增至 SiteWise Edge 閘道。

新增身分驗證組態（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 從閘道清單中，選擇目標 SiteWise Edge 閘道。
3. 從資料來源清單中，選擇目標資料來源，然後選擇編輯。
4. 在新增資料來源頁面上，選擇進階組態。
5. 針對身分驗證組態，選擇您在上一個步驟中部署的秘密。
6. 選擇 Save (儲存)。

SiteWise Edge 閘道上的合作夥伴資料來源

使用 AWS IoT SiteWise Edge 閘道時，您可以將合作夥伴資料來源連線至 SiteWise Edge 閘道，並從 SiteWise Edge 閘道和 AWS 雲端中的合作夥伴接收資料。這些合作夥伴資料來源是在 AWS 和合作夥伴之間共同開發的 AWS IoT Greengrass 元件。當您新增合作夥伴資料來源時，AWS IoT SiteWise 會建立此元件，並將其部署到您的 SiteWise Edge 閘道。

Note

您可以為每個閘道中的每個合作夥伴新增一個資料來源。

若要新增合作夥伴資料來源，請執行下列動作：

1. [在 SiteWise Edge 中新增合作夥伴資料來源](#)
2. 前往合作夥伴的 Web 入口網站，如適用，並設定合作夥伴資料來源，使其連線至 SiteWise Edge 閘道。

主題

- [安全](#)

- [在 SiteWise Edge 閘道 Docker 上設定](#)
- [在 SiteWise Edge 中新增合作夥伴資料來源](#)
- [SiteWise Edge 閘道合作夥伴資料來源選項](#)

安全

作為 AWS 我們客戶和合作夥伴之間[共同責任模型](#)的一部分，以下說明負責不同安全層面的人員：

客戶責任

- 審核合作夥伴。
- 設定提供給合作夥伴的網路存取權。
- 監控 SiteWise Edge 閘道機器資源 (CPU、記憶體和檔案系統) 的合理使用情況。

AWS 責任

- 將合作夥伴與客戶 AWS 雲端資源隔離，但合作夥伴所需的資源除外。在此情況下，即為 AWS IoT SiteWise 擷取。
- 限制合作夥伴解決方案合理使用 SiteWise Edge 閘道機器資源 (CPU 和記憶體)。

合作夥伴責任

- 使用安全預設值。
- 透過修補程式和其他適當的更新，在一段時間內保護解決方案的安全。
- 將客戶資料保密。

在 SiteWise Edge 閘道 Docker 上設定

AWS IoT SiteWise 提供的 Docker 映像可讓您在各種平台和環境中執行 SiteWise Edge 應用程式。此 Docker 映像會封裝從工業設備收集、處理和傳送資料至 AWS 雲端所需的所有必要元件和相依性。透過使用 Docker 映像，您可以在 Docker 相容主機上部署和執行 SiteWise Edge 應用程式，例如伺服器、邊緣裝置或雲端容器服務。

若要新增合作夥伴資料來源，您的本機裝置上必須安裝 [Docker 引擎](#) 1.9.1 或更新版本。

Note

20.10 版是經過驗證可與 SiteWise Edge 閘道軟體搭配使用的最新版本。

確認 Docker 已安裝

若要驗證 Docker 是否已安裝，請從連線至 SiteWise Edge 閘道的終端機執行下列命令：

```
docker info
```

如果命令傳回 `docker is not recognized` 結果，或已安裝較舊版本的 Docker，請在繼續之前[安裝 Docker 引擎](#)。

設定 Docker

執行 Docker 容器元件的系統使用者必須具有根或管理員許可，或者您必須將 Docker 設定為以非根或非註冊者使用者身分執行。

在 Linux 裝置上，您必須將 `ggc_user` 使用者新增至 `docker` 群組，才能在沒有 `sudo` 的情況下呼叫 Docker 命令。

若要將 `ggc_user` 或您用來執行 Docker 容器元件的非根使用者新增至 `docker` 群組，請執行下列命令：

```
sudo usermod -aG docker ggc_user
```

如需詳細資訊，請參閱 [Docker 引擎的 Linux 安裝後步驟](#)。

在 SiteWise Edge 中新增合作夥伴資料來源

若要將合作夥伴資料來源連線至 SiteWise Edge 閘道，請將其新增為資料來源。當您將其新增為資料來源時，AWS IoT SiteWise 會將私有 AWS IoT Greengrass 元件部署到您的 SiteWise Edge 閘道。

先決條件

若要新增合作夥伴資料來源，您必須執行下列動作：

- 對於 EasyEdge 和 CloudRail，與合作夥伴建立帳戶，然後繫結帳戶。
- [在 SiteWise Edge 閘道 Docker 上設定](#)

使用合作夥伴資料來源建立 SiteWise Edge 閘道

如果您想要建立新的 SiteWise Edge 閘道，請完成中的步驟[建立自我託管 SiteWise Edge 閘道](#)。建立 SiteWise Edge 閘道之後，請依照中的步驟[將合作夥伴資料來源新增至現有的 SiteWise Edge 閘道](#)新增合作夥伴資料來源。

將合作夥伴資料來源新增至現有的 SiteWise Edge 閘道

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選擇您要連接合作夥伴資料來源的 SiteWise Edge 閘道。
4. 在資料來源下，選擇新增資料來源。
5. 在新增資料來源畫面上，選擇來源類型，以選取連接 SiteWise Edge 閘道的合作夥伴。每個資料來源都有自己的組態選項。資料來源有兩種類別：AWS 來源和合作夥伴來源。

使用合作夥伴資料來源，您可以為每個閘道選擇一個來源。如需資料來源合作夥伴整合選項的清單，請參閱 [SiteWise Edge 閘道合作夥伴資料來源選項](#)。請注意，您最多可以新增 100 個 OPC UA 資料來源 (AWS 來源)。若要開始使用 OPC UA 資料來源，請參閱 [AWS IoT SiteWise Edge 閘道的 OPC UA 資料來源](#)。

6. 輸入來源的名稱。
7. 選取下方的資料來源標籤，然後遵循組態程序。

CloudRail

在儲存 SiteWise Edge 閘道的資料來源 CloudRail 之後，入口網站中會完成大部分的 CloudRail 組態。不過，需要授權連線。

Note

CloudRail 連線僅適用於 Linux。

1. [建立 CloudRail 帳戶](#) 以開始使用 連線 AWS IoT SiteWise。
2. 確定 Docker 已安裝在您的閘道上。如需詳細資訊，請參閱 [在 SiteWise Edge 閘道上設定 Docker](#)。
3. 閱讀授權存取和部署協議，然後選擇授權。勾選方塊會授予 AWS 合作夥伴存取您資料來源的權限 AWS，並允許在合作夥伴的元件上部署。

Note

測量字首 - 選用是在 CloudRail 入口網站中設定。

 Note

合作夥伴軟體是由 AWS 合作夥伴開發、維護和支援。AWS 不負責介面、組態或軟體。

如需詳細資訊，請參閱[CloudRail](#)。

EasyEdge

在儲存 SiteWise Edge 閘道的資料來源EasyEdge之後，入口網站中會完成大部分的EasyEdge組態。不過，需要授權連線。

 Note

EasyEdge 連線僅適用於 Linux。

1. [建立 EasyEdge帳戶](#)以開始使用 連線 AWS IoT SiteWise。
2. 確定 Docker 已安裝在您的閘道上。如需詳細資訊，請參閱[在 SiteWise Edge 閘道Docker 上設定](#)。
3. 閱讀授權存取和部署協議，然後選擇授權。勾選方塊會授予 AWS 合作夥伴存取您資料來源的權限 AWS ，並允許 在合作夥伴的元件上部署 。

 Note

測量字首 - 選用是在EasyEdge入口網站中設定。

 Note

合作夥伴軟體是由 AWS 合作夥伴開發、維護和支援。AWS 不負責介面、組態或軟體。

如需詳細資訊，請參閱[EasyEdge](#)。

Litmus Edge

您可以透過兩種方式啟用Litmus組態。使用Litmus Edge Manager入口網站的資訊Litmus Edge直接啟用 AWS IoT SiteWise 。或者，您可以透過 手動啟用 Litmus Edge 的 AWS IoT SiteWise Litmus Edge Manager。

Note

Litmus Edge 連線僅適用於 Linux。

在 上使用啟用碼Litmus Edge啟用 AWS IoT SiteWise

在 新增具有Litmus Edge啟用碼的Litmus Edge資料來源時，請使用此程序 AWS IoT SiteWise 主控台。

1. 選取立即使用程式碼啟用。其他組態選項隨即出現。
2. 輸入 Litmus Edge Manager 以Litmus Edge連線至 SiteWise Edge 閘道。如需詳細資訊，請參閱 Litmus Edge Manager 文件中的[步驟 3a：設定資料和裝置管理端點](#)。
3. 提供要在 Litmus Edge上啟用的Litmus Edge Manager啟用碼 AWS IoT SiteWise
4. 或者，AWS IoT SiteWise 提供 Litmus Edge Manager CA 憑證。憑證Litmus Edge可防止在未經授權的 上啟用 Litmus Edge Manager。
5. 確定 Docker 已安裝在您的閘道上。如需詳細資訊，請參閱[在 SiteWise Edge 閘道Docker 上設定](#)。

Note

AWS IoT SiteWise 會將合作夥伴應用程式部署為 Docker 容器。應用程式使用 NET_ADMIN功能部署，以便透過 管理 Litmus Edge Docker 容器Litmus Edge Manager。Litmus Edge需要此特殊權限存取才能在您的裝置上執行。如需 Litmus Edge Docker 需求的詳細資訊，請參閱 Litmus Edge 文件中的 QuickStart 指南中的 [Docker 安裝](#)。

6. 閱讀授權存取和部署協議，然後選擇授權。勾選方塊會授予 AWS 合作夥伴存取您資料來源的權限 AWS ，並允許 在合作夥伴的元件上部署。

透過 手動啟用 Litmus Edge

1. 選取稍後在 上啟用Litmus Edge。
2. 確定 Docker 已安裝在您的閘道上。如需詳細資訊，請參閱[在 SiteWise Edge 閘道Docker 上設定](#)。

Note

AWS IoT SiteWise 會將合作夥伴應用程式部署為 Docker 容器。應用程式使用 NET_ADMIN功能部署，以便透過 管理 Litmus Edge Docker 容器Litmus Edge Manager。 Litmus Edge需要此特殊權限存取才能在您的裝置上執行。如需 Litmus Edge Docker 需求的詳細資訊，請參閱 Litmus Edge 文件中的 QuickStart 指南中的 [Docker 安裝](#)。

3. 閱讀授權存取和部署協議，然後選擇授權。勾選方塊會授予 AWS 合作夥伴存取您資料來源的權限 AWS ，並允許 在合作夥伴的元件上部署 。
4. 部署完成後，請遵循 Litmus Edge QuickStart Guide 文件中的[存取 Litmus Edge Web UI](#) 說明。

Note

合作夥伴軟體是由 AWS 合作夥伴開發、維護和支援。AWS 不負責介面、組態或軟體。

如需詳細資訊，請參閱[Litmus Edge](#)。

8. 選擇 Save (儲存)。

SiteWise Edge 閘道合作夥伴資料來源選項

AWS IoT SiteWise 可讓您從各種合作夥伴資料來源連接和擷取資料，例如工業設備、感應器和其他第三方系統。若要連接合作夥伴資料來源，您需要遵循幾個步驟，包括設定要傳送資料的資料來源 AWS

IoT SiteWise、設定必要的許可和身分驗證，以及將資料映射到您的資產模型。此程序可確保合作夥伴資料無縫整合至您的 AWS IoT SiteWise 環境，讓您與其他資料來源一起監控和分析。

本節列出 SiteWise Edge 閘道上第三方資料來源整合的可用合作夥伴。使用以下資訊來設定合作夥伴資料來源。

 Note

您可以為每個閘道中的每個合作夥伴新增一個資料來源

CloudRail

入口網站：

<https://devices.cloudrail.com/>

需求

如需CloudRail需求的詳細資訊，請參閱 CloudRail 網站上的[常見問答集](#)。

CloudRail 文件：

[邊緣運算：SiteWise Edge](#)

EasyEdge

入口網站：

<https://studio.easyedge.io/>

需求

[EasyEdge 需求](#) – EasyEdge需求的相關資訊，包括設定防火牆所需的端點和連接埠。注意：您將需要 EasyEdge帳戶才能存取此文件。

EasyEdge 文件：

[EasyEdge 適用於的 AWS](#)

Litmus Edge

存取 Litmus Edge Manager：

若要存取 Litmus Edge，請設定 [Litmus Edge Manager 帳戶](#)。

需求

[Litmus Edge 需求](#) – 部署 的建議組態和系統需求Litmus Edge。

Litmus 文件：

- [整合到 AWS IoT SiteWise](#)
- [Litmus Edge 文件](#)

AWS IoT GreengrassAWS IoT SiteWise Edge 的 元件

SiteWise Edge 使用 AWS IoT Greengrass 元件來收集、處理和傳輸邊緣的工業資料。這些元件可共同運作，以啟用本機資料處理，並與 AWS IoT SiteWise 雲端服務無縫整合。

IoT SiteWise 發佈者

IoT SiteWise 發佈者元件 (`aws.iot.SiteWiseEdgePublisher`) 負責：

- 將收集的資料安全地傳輸到 AWS IoT SiteWise 雲端服務
- 在連線問題期間管理資料緩衝和重試

如需設定 SiteWise Edge 發佈者的詳細資訊，請參閱 [設定 AWS IoT SiteWise 發佈者元件](#)。此外，如需發佈者元件的詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的 [IoT SiteWise 發佈者](#)。

IoT SiteWise 處理器

IoT SiteWise 處理器元件 (`aws.iot.SiteWiseEdgeProcessor`) 會執行下列任務：

- 在邊緣執行資料轉換和計算
- 在本機實作資產屬性定義和運算
- 在傳輸之前彙總或篩選資料，以減少資料量

如需處理器元件的詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的 [IoT SiteWise 處理器](#)。

IoT SiteWise OPC UA 收集器

IoT SiteWise OPC UA 收集器 (`aws.iot.SiteWiseEdgeCollectorOpcua`) 元件旨在：

- 連線至工業環境中的 OPC UA 伺服器
- 有效率地從 OPC UA 資料來源收集資料
- 將 OPC UA 資料轉換為與 相容的格式 AWS IoT SiteWise

如需 OPC UA 收集器元件的詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的 [IoT SiteWise OPC UA 收集器](#)。

IoT SiteWise OPC UA 資料來源模擬器

IoT SiteWise OPC UA 資料來源模擬器元件

(`aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator`) 提供下列功能：

- 啟動產生範例資料的本機 OPC UA 伺服器
- 模擬閘道上 AWS IoT SiteWise OPC UA 收集器元件可以讀取的資料來源 AWS IoT SiteWise
- 使用產生的範例資料來探索 AWS IoT SiteWise 功能

此元件特別適用於測試和開發目的，可讓您模擬工業資料來源，而不需要實體設備。

如需資料來源模擬元件的詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的 [IoT SiteWise OPC UA 資料來源模擬器](#)。

這些 AWS IoT Greengrass 元件可用來啟用 SiteWise Edge 功能。IoT SiteWise 發佈者可確保資料可靠地傳送到雲端，IoT SiteWise 處理器會處理本機運算和資料最佳化，而 IoT SiteWise OPC UA 收集器可促進與常見工業通訊協定的整合。

Note

若要使用這些元件，您必須在邊緣裝置上安裝 AWS IoT Greengrass V2 或更新版本。每個元件的適當組態對於 SiteWise Edge 的最佳效能至關重要。

在 SiteWise Edge 閘道上篩選資產

您可以使用邊緣篩選，透過僅將一部分資產傳送至特定的 SiteWise Edge 閘道，以更有效地管理資產。如果您的資產配置在樹狀目錄或父子系結構中，您可以設定連接到 SiteWise Edge 閘道 IAM 角色的 IAM 政策，僅允許將樹狀目錄或父系及其子系傳送至特定的 SiteWise Edge 閘道。

Note

如果您要將現有資產配置為樹狀結構，請在建立結構之後，前往您新增至結構的每個現有資產，然後選擇編輯，然後選擇儲存以確保 AWS IoT SiteWise 辨識新結構。

設定邊緣篩選

在 SiteWise Edge 閘道上設定邊緣篩選，方法是將下列 IAM 政策新增至 SiteWise Edge 閘道的 IAM 角色，以您想要傳送至 SiteWise Edge 閘道的根資產 ID 取代 `<root-asset-id>`。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "iotsitewise:DescribeAsset",
        "iotsitewise:ListAssociatedAssets"
      ],
      "Resource": "arn:aws:iotsitewise:*:*:asset/*",
      "Condition": {
        "StringNotLike": {
          "iotsitewise:assetHierarchyPath": "/<root-asset-id>*"
        }
      }
    }
  ]
}
```

如果您想要移除 SiteWise Edge 閘道上目前有資產，請登入 SiteWise Edge 閘道並執行下列命令，AWS IoT SiteWise 透過刪除快取強制 SiteWise Edge 閘道與 同步。

```
sudo rm /greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/sync-app/
sync_resource_bundles/edge.json
```

設定 AWS IoT SiteWise Edge 的代理支援和管理信任存放區

在 AWS IoT SiteWise Edge 中，設定和管理信任存放區，為您的邊緣裝置設定代理支援。首先，設定代理組態，然後設定信任存放區。您可以在閘道安裝期間或建立閘道後手動設定信任存放區。

- 代理 – 促進各種網路環境中的邊緣裝置 AWS 和服務之間的連線。
- 信任存放區 – 透過管理信任的憑證來確保安全連線。適當的組態可協助您遵守網路安全政策、在受限的網路環境中啟用通訊，以及最佳化邊緣裝置和雲端服務之間的資料傳輸。

SiteWise Edge 會針對不同的元件類型使用多個信任存放區，確保從邊緣裝置到雲端的安全且有效率資料流程。您可以在現有閘道上或在建立新閘道的安裝程序期間設定信任存放區和代理。

信任存放區和代理組態的需求

在使用代理設定設定信任存放區或安裝 SiteWise Edge 之前，請確定您符合先決條件。根據您的元件用量和功能需求，有不同的實作需求。

Proxy 支援需求

- 代理伺服器的 URL。URL 應包含使用者資訊、主機的连接埠號碼。例如 `scheme://[userinfo@]host[:port]`。
 - `scheme` – 必須是 HTTP 或 HTTPS
 - (選用) `userinfo` – 使用者名稱和密碼資訊
 - `host` – 代理伺服器的主機名稱或 IP 地址
 - `port` – 連接埠號碼
- 略過代理的地址清單。
- (選用) 如果您使用 HTTPS 代理搭配自我簽署憑證，則為代理 CA 憑證檔案。

信任存放區需求

- 如需使用 HTTPS 代理的完整資料處理套件功能，您應該更新這三個信任存放區。
- 如果您只使用 IoT SiteWise OPC UA 收集器和 IoT SiteWise 發佈者，請將 AWS IoT Greengrass Core 和 Java 信任存放區憑證更新為最新版本。

信任存放區和代理伺服器邊緣組態的最佳實務

針對持續維護和在邊緣環境中維持最高層級的安全性：

- 定期檢閱和更新代理設定，以符合您的網路安全需求。
- 監控閘道連線和資料流程，以確保適當的代理通訊

- 根據組織的憑證管理政策來維護和更新信任存放區
- 您可以實作並遵循我們在邊緣環境中安全通訊的建議最佳實務，例如：
- 記錄您的代理和信任存放區組態，以實現操作可見性
- 遵循組織的登入資料管理安全實務

這些實務有助於維持 SiteWise Edge 閘道的安全和可靠操作，同時與更廣泛的安全政策保持一致。

在 AWS IoT SiteWise Edge 閘道安裝期間設定代理設定

您可以設定 AWS IoT SiteWise Edge 在閘道安裝期間使用代理伺服器。安裝指令碼同時支援 HTTP 和 HTTPS 代理，並可以自動設定安全代理連線的信任存放區。

當您使用代理設定執行安裝指令碼時，它會執行數個重要的任務：

- 驗證代理 URL 格式和參數，以確保正確指定它們。
- 透過設定的代理下載並安裝必要的相依性。
- 如果提供代理 CA 憑證，它會附加到 AWS IoT Greengrass 根 CA 憑證並匯入到 Java KeyStore。
- 設定 AWS IoT Greengrass (SiteWise Edge 使用的) 為所有傳出連線使用代理。
- 使用適當的代理和信任存放區組態完成 SiteWise Edge 安裝。

在安裝閘道軟體時設定代理設定

1. 建立 SiteWise Edge 閘道。如需詳細資訊，請參閱[建立自我託管 SiteWise Edge 閘道](#)及[在本機裝置上安裝 AWS IoT SiteWise Edge 閘道軟體](#)。
2. 為您的環境使用適當的代理設定執行安裝指令碼。將預留位置取代為您的特定代理資訊

取代下列每個項目：

- -p, --proxy-url – 代理伺服器的 URL。URL 必須是 http 或 https。
- -n, --no-proxy – 以逗號分隔的地址清單，以略過代理。
- (選用) -c: 代理 CA 憑證檔案的 --proxy-ca-cert 路徑。
- (選用) -j, --javastorepass – Java KeyStore 密碼。預設密碼為 changeit。

Linux

對於 Linux 系統，請使用下列命令結構：

```
sudo ./install.sh -p proxy-url -n no-proxy-addresses [-c proxy-ca-cert-path] [-j javastorepass]
```

Windows

對於使用 PowerShell Microsoft Windows 的系統，請使用下列命令結構：

```
.\install.ps1 -ProxyUrl proxy-url -NoProxyAddresses no-proxy-addresses [-ProxyCaCertPath proxy-ca-cert-path] [-JavaStorePass javastorepass]
```

在啟用代理的安裝期間進行故障診斷

如需解決 SiteWise Edge 閘道相關信任存放區問題的詳細資訊，請參閱 [啟用代理的安裝問題](#)。

在 AWS IoT SiteWise Edge 中手動設定 HTTPS 代理支援的信任存放區

設定 AWS IoT SiteWise Edge 元件以透過 HTTPS 代理連線時，請將代理伺服器的憑證新增至適當的信任存放區。SiteWise Edge 使用多個信任存放區來保護通訊。有三個信任存放區，您使用它們取決於閘道實作中的 SiteWise Edge 元件類型。

提供代理設定時，信任存放區會在安裝程序期間自動更新。

- [設定 AWS IoT Greengrass 核心元件信任存放區](#) – AWS IoT Greengrass 根 CA 憑證包含在信任存放區中，以驗證 AWS 服務的真實性。

此信任存放區可協助 AWS IoT Greengrass 元件透過代理安全地與服務 AWS 通訊，同時驗證這些服務的真實性。

- [設定以 Java 為基礎的元件信任存放區](#) – Java KeyStore (JKS) 是 Java 型元件用於 SSL/TLS 連線的主要信任存放區。

Java 應用程式依賴 JKS 來建立安全連線。例如，如果您使用 IoT SiteWise 發佈者或以 Java 為基礎的 IoT SiteWise OPC UA 收集器，則需要設定此信任存放區。這可確保這些元件在將資料傳送至雲端或從 OPC UA 伺服器收集資料時，可以透過 HTTPS 代理安全地通訊。

- [系統層級元件信任存放區組態](#) – 使用 HTTPS 代理時，必須將憑證新增至適當的信任存放區，以啟用安全連線。

使用 HTTPS 代理時，其憑證必須新增至適當的信任存放區，才能啟用安全連線。這是必要的，因為通常以 Rust 或 Go 等語言撰寫的系統層級元件依賴系統的信任存放區，而不是 Java 的 JKS。例

如，如果您使用的是需要透過代理通訊的系統公用程式（例如軟體更新或時間同步），則需要設定系統層級信任存放區。這可確保這些元件和公用程式可以透過代理建立安全連線。

設定 AWS IoT Greengrass 核心元件信任存放區

對於使用 Amazon 根 CA AWS IoT Greengrass 的核心函數：

1. 在 找到憑證檔案 `/greengrass/v2/AmazonRootCA1.pem`
2. 將 HTTPS 代理根憑證（自我簽署）附加至此檔案。

```
-----BEGIN CERTIFICATE-----
MIIEFTCCA v2gAwIQWgIVAMHSAzWG/5YVRYtRQ0xXUTEpHuEmApzGCSqGSIb3DQEK
\nCwUAhuL9MQswCQwJVUzEPMAVUzEYMBYGA1UECgwP1hem9uLmNvbSBjbMUMRww
... content of proxy CA certificate ...
+vHIR1t0e5JAm5\noTIZGoFbK82A0/n07f/t5PSIDAim9V3Gc3pSXxCCAQoFYnui
GaPULGk1gCE84a0X\n7Rp/1ND/PuMZ/s8YjlK2NmYmNjMCAXDTE5MTEyN2cM216
gJMIADggEPADf2/m45hzEXAMPLE=
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIDQTCCAimgF6AwIBAgITBmyfz/5mjAo54vB4ikPm1jZKyjANJmApzyMZFo6qBg
ADA5MQswCQYDVQQGEwJVUzEPMA0tMVT8QtPHRh8jrdkGA1UEChMGDV3QQDExBKw
... content of root CA certificate ...
o/ufQJQWUCyziar1hem9uMRkwFwYVPSHCb2XV4cdFyQzR1KldZwgJcIQ6XUDgHaa
5MsI+yMRQ+hDaXJioblXgUka642M4UwtBV8oK2xJNDd2ZhwLnoQdeXeGADKkpy
rqXRfKoQnoZsG4q5WTP46EXAMPLE
-----END CERTIFICATE-----
```

在已建立的閘道上設定 HTTPS 代理

您可以將代理支援新增至已建立的閘道，方法是連線至連接埠 443，而不是連接埠 8883。如需使用代理伺服器的詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的在[連接埠 443 上或透過網路代理連線](#)。如果您建立新的閘道，您可以在閘道安裝期間設定代理組態。如需詳細資訊，請參閱在[AWS IoT SiteWise Edge 閘道安裝期間設定代理設定](#)。

當您在 SiteWise Edge AWS IoT Greengrass 上使用 HTTPS 代理搭配時，軟體會根據提供的 URL 在 HTTP 和 HTTPS 之間自動選擇代理連線。

 Important

在嘗試透過 HTTPS 代理連線之前，更新所有必要的信任存放區。

設定以 Java 為基礎的元件信任存放區

對於資料處理套件中的 IoT SiteWise 發佈者、IoT SiteWise OPC UA 收集器和 Java 服務，預設 Java 信任存放區位置為 `$JAVA_HOME/jre/lib/security/cacerts`

新增憑證

1. 建立 檔案以存放代理伺服器的憑證，例如 `proxy.crt`。

 Note

使用代理伺服器的憑證提前建立檔案。

2. 使用下列命令將檔案新增至 Java 的信任存放區：

```
sudo keytool -import -alias proxyCert -keystore /usr/lib/jvm/java-11-openjdk-amd64/lib/security/cacerts -file proxy.crt
```

3. 出現提示時，請使用預設密碼：`changeit`

系統層級元件信任存放區組態

對於以 Rust、Go 和使用系統信任存放區的其他語言撰寫的元件：

Linux

Linux 系統：將憑證新增至 `/etc/ssl/certs/ca-certificates.crt`

Windows

Microsoft Windows 系統：若要設定信任存放區，請遵循 Microsoft Ignite 文件中的[憑證存放區](#)程序。

Windows 提供多個憑證存放區，包括使用者和電腦範圍的個別存放區，每個都有數個子存放區。對於大多數 SiteWise Edge 設定，我們建議將憑證新增至 `COMPUTER | Trusted Root`

Certification Authorities 存放區。不過，根據您的特定組態和安全性需求，您可能需要使用不同的存放區。

對信任存放區問題進行故障診斷

如需解決 SiteWise Edge 閘道相關信任存放區問題的詳細資訊，請參閱 [信任存放區問題](#)。

邊緣上的 AWS IoT SiteWise APIs

AWS IoT SiteWise 提供其 APIs 的子集，以及邊緣特定的 APIs，可與部署在邊緣的資產模型及其關聯資產進行無縫互動。這些資產模型必須設定為在邊緣執行。如需詳細資訊，請參閱 [設定用於 SiteWise Edge 上資料處理的資產模型](#) 以取得此設定程序的詳細說明。

設定這些 APIs 之後，您可以擷取資產模型和個別資產的完整資料。擷取資產模型、資產、儀表板、入口網站和專案資訊可協助您監控部署的入口網站和儀表板，以及存取在邊緣層級收集的資產資料。這會在您的網路中提供與互動的中央主機，AWS IoT SiteWise 而不需要 Web API 呼叫。

主題

- [所有可用的 AWS IoT SiteWise Edge 裝置 APIs](#)
- [用於邊緣裝置的僅限 AWS IoT SiteWise 邊緣 APIs](#)
- [在 AWS IoT SiteWise Edge APIs 上啟用 CORS](#)
- [設定 AWS IoT SiteWise Edge 的工作階段逾時](#)
- [教學課程：列出 AWS IoT SiteWise Edge 閘道上的資產模型](#)

所有可用的 AWS IoT SiteWise Edge 裝置 APIs

AWS IoT SiteWise 提供各種 APIs 以在邊緣裝置上使用，讓您可以在裝置上於本機完成任務。一些可用的邊緣 APIs 包括擷取資產模型、建立和更新資產屬性，以及將資料串流傳送至雲端。透過利用這些 APIs，您可以建置可在間歇性或有限網路連線的環境中操作的解決方案。

AWS IoT SiteWise APIs

下列 AWS IoT SiteWise APIs 可在邊緣裝置上使用：

- [ListAssetModels](#)
- [DescribeAssetModel](#)
- [ListAssets](#)

- [DescribeAsset](#)
- [DescribeAssetProperty](#)
- [ListAssociatedAssets](#)
- [GetAssetPropertyAggregates](#)
- [GetAssetPropertyValue](#)
- [GetAssetPropertyValueHistory](#)
- [ListDashboards](#)
- [ListPortals](#)
- [ListProjectAssets](#)
- [ListProjects](#)
- [DescribeDashboard](#)
- [DescribePortal](#)
- [DescribeProject](#)

可用的僅限邊緣 APIs

下列 APIs 會在本機用於邊緣的裝置：

- [驗證](#) – 使用此 API 取得您將用來進行 API 呼叫的 SigV4 臨時登入資料。

用於邊緣裝置的僅限 AWS IoT SiteWise 邊緣 APIs

除了邊緣上可用的 AWS IoT SiteWise APIs 之外，還有特定邊緣的 API。這些邊緣規格 APIs 如下所述。

驗證

從 SiteWise Edge 閘道取得登入資料。您需要新增本機使用者，或使用 LDAP 或 Linux 使用者集區連線到您的系統。如需新增使用者的詳細資訊，請參閱 [LDAP](#) 或 [Linux 使用者集區](#)。

請求語法

```
POST /authenticate HTTP/1.1
Content-type: application/json
{
  "username": "string",
  "password": "string",
```

```
"authMechanism": "string"  
}
```

URI 請求參數

請求不會使用任何 URI 參數。

請求內文

請求接受採用 JSON 格式的下列資料。

使用者名稱

用來驗證請求呼叫的使用者名稱。

類型：字串

必要：是

密碼

請求登入資料之使用者的密碼。

類型：字串

必要：是

authMechanism

在主機中驗證此使用者的身分驗證方法。

類型：字串

有效值：ldap、linux、winnt

必要：是

回應語法

```
HTTP/1.1 200  
Content-type: application/json  
{  
  "accessKeyId": "string",  
  "secretAccessKey": "string",  
  "sessionToken": "string",  
}
```

```
"region": "edge"  
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

下列資料會以 JSON 格式傳回。

accessKeyId

識別臨時安全登入資料的存取金鑰 ID。

長度限制：長度下限為 16。長度上限為 128。

模式：`[\w]*`

secretAccessKey

可用來簽署請求的私密存取金鑰。

類型：字串

sessionToken

使用者必須傳遞至服務 API 才能使用臨時登入資料的字符。

類型：字串

region

您為 API 呼叫設定目標的區域。

類型：CONSTANT - edge

錯誤

IllegalArgumentException

請求被拒絕，因為提供的內文文件格式不正確。錯誤訊息說明特定錯誤。

HTTP 狀態碼：400

AccessDeniedException

根據目前的身分提供者，使用者沒有有效的登入資料。錯誤訊息說明身分驗證機制。

HTTP 狀態碼：403

TooManyRequestsException

請求已達到其身分驗證嘗試的限制。錯誤訊息包含等待直到進行新的身分驗證嘗試的時間量。

HTTP 狀態碼：429

在 AWS IoT SiteWise Edge APIs 上啟用 CORS

在 AWS IoT SiteWise Edge APIs 上啟用 CORS（跨來源資源共用）可讓 Web 應用程式直接與不同網域的 APIs 通訊。這可實現無縫整合、即時資料交換和跨網域資料存取，無需中介伺服器或解決方法。您可以將 CORS 設定設定為指定允許的原始伺服器，以確保受控制的跨來源存取。

Note

CORS 適用於 3.3.1 版和更新版本 此功能適用於 3.3.1 版和更新版本 `aws.iot.SiteWiseEdgeProcessor` 元件。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的 [AWS IoT SiteWise 處理器](#)。

在 SiteWise Edge APIs 上啟用 CORS

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取您要為其啟用 CORS 的 SiteWise Edge 閘道。您可以在 AWS IoT Greengrass V2 部署類型上啟用 CORS。
4. 在閘道組態區段中，選擇相關聯的 Greengrass 核心裝置。
5. 在部署索引標籤的 Greengrass 裝置下，選取適當的部署連結。
6. 在動作下選擇修訂，然後選擇修訂部署。

Important

建立已啟用修訂 CORS 的組態會取代裝置目前的組態。

7. 在步驟 1 指定目標中，提供選用的名稱來識別部署。
8. 在步驟 2 中，選取元件 - 選用，您可以將所有目前的選擇保持原狀，然後選擇下一步。

9. 在步驟 3 中，設定元件 - 選用，選取 `aws.iot.SiteWiseEdgeProcessor`，然後選擇設定元件。
10. 在組態更新區段要合併的組態下，輸入下列 JSON：

```
{  
  "AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN": "*"   
}
```

 Note

使用 * 做為 的值 `AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN` 允許所有原始伺服器。對於生產環境，建議指定確切的原始URLs 以提高安全性。

11. 選擇確認。
12. 選擇下一步繼續其餘步驟，直到您抵達Step5，檢閱為止。
13. 檢閱您的組態變更，然後選擇部署，將變更套用至 SiteWise Edge 閘道。

 Note

或者，您可以透過將全域環境變數設定為 AWS IoT SiteWise 閘道* 上的 `AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN` 來啟用 CORS。

 Note

對於已驗證的代理，`userinfo` 必須包含在代理組態的 `url` 欄位中，而不是做為分隔的 `username` 和 `password` 欄位。

部署完成後，會在 SiteWise Edge API 上啟用 CORS，允許指定的原始伺服器向 API 提出跨來源請求。

設定 AWS IoT SiteWise Edge 的工作階段逾時

SiteWise Edge 可讓您設定 SiteWise Edge API 的工作階段逾時。此功能透過在指定的時段後自動終止非作用中工作階段來增強安全性。本節會引導您使用 設定工作階段逾時的程序 AWS IoT SiteWise 主控台。

 Note

工作階段逾時組態適用於 `aws.iot.SiteWiseEdgeProcessor` 元件的 3.4.0 版和更新版本。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的 [AWS IoT SiteWise 處理器](#)。

設定 SiteWise Edge 閘道的工作階段逾時

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇您要設定工作階段逾時的 SiteWise Edge 閘道。

 Note

您可以在 AWS IoT Greengrass V2 部署類型上設定工作階段逾時。

4. 在閘道組態區段中，選擇相關聯的 Greengrass 核心裝置。
5. 在部署索引標籤的 Greengrass 裝置下，選取適當的部署連結。
6. 在動作下選擇修訂。閱讀警告，然後選擇修訂部署。

 Important

建立修訂的工作階段逾時組態會取代裝置目前的組態。

7. 在步驟 1 指定目標中，提供選用的名稱以識別修訂後的部署，然後選擇下一步。
8. 在步驟 2 中，選取元件 - 選用，您可以依原狀保留所有目前的選擇，然後選擇下一步。
9. 在步驟 3 中，設定元件 - 選用，選取 `aws.iot.SiteWiseEdgeProcessor`，然後選擇設定元件。
10. 在組態更新區段的要合併的組態下，輸入下列 JSON：

```
{
  "AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES": "240"
}
```

11. 以分鐘 `AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES` 為單位設定 的值。工作階段逾時值可以是 1 分鐘到 10080 分鐘 (7 天)。預設值為 240 分鐘 (4 小時)。

12. 選擇確認。
13. 選擇下一步繼續其餘步驟，直到您抵達步驟 5 檢閱為止。
14. 檢閱您的組態變更，然後選擇部署，將變更套用至 SiteWise Edge 閘道。

 Note

或者，您可以將全域環境變數 `AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES` 設定為 SiteWise Edge 閘道上所需的值（以分鐘為單位），以設定工作階段逾時。

部署完成後，新的工作階段逾時組態會套用至 SiteWise Edge API。

教學課程：列出 AWS IoT SiteWise Edge 閘道上的資產模型

您可以使用一部分的可用 AWS IoT SiteWise APIs 以及邊緣特定的 APIs，與邊緣上的資產模型及其資產互動。本教學課程將逐步引導您取得 AWS IoT SiteWise Edge 閘道的臨時登入資料，以及 SiteWise Edge 閘道上的資產模型清單。

先決條件

在本教學課程中，您可以使用各種工具。若要使用這些工具，請確定已安裝對應的先決條件。

為了完成本教學，您需要以下項目：

- 部署和執行中的 [AWS IoT SiteWise Edge 自我託管閘道需求](#)
- 透過連接埠 443 存取相同網路中的 SiteWise Edge 閘道。
- 已安裝 [OpenSSL](#)
- (AWS OpsHub for AWS IoT SiteWise) [AWS OpsHubAWS IoT SiteWise 應用程式的](#)
- 已安裝 (curl) [curl](#)
- (Python) [urllib3](#) 已安裝
- (Python) 已安裝 [Python3](#)
- (Python) 已安裝 [Boto3](#)
- (Python) 已安裝 [BotoCore](#)

步驟 1：取得 SiteWise Edge 閘道服務簽章憑證

若要建立與 SiteWise Edge 閘道上可用 APIs 的 TLS 連線，您需要信任的憑證。您可以使用 OpenSSL 或 AWS OpsHub 產生此憑證 AWS IoT SiteWise。

OpenSSL

Note

您需要安裝 [OpenSSL](#) 才能執行此命令。

開啟終端機並執行下列命令，從 SiteWise Edge 閘道取得已簽署的憑證。<sitewise_gateway_ip> 將取代為 SiteWise Edge 閘道的 IP。

```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | openssl  
x509 -outform PEM > GatewayCert.pem
```

AWS OpsHub for AWS IoT SiteWise

您可以使用 AWS OpsHub 的 AWS IoT SiteWise。如需詳細資訊，請參閱[管理 SiteWise Edge 閘道](#)。

本教學課程使用已下載 SiteWise Edge 閘道憑證的絕對路徑。執行下列命令以匯出憑證的完整路徑，將取代<absolute_path_to_certificate>為憑證的路徑：

```
export PATH_TO_CERTIFICATE='<absolute_path_to_certificate>'
```

步驟 2：取得 SiteWise Edge 閘道主機名稱

Note

您需要安裝 [OpenSSL](#) 才能執行此命令。

若要完成教學課程，您需要 SiteWise Edge 閘道的主機名稱。若要取得 SiteWise Edge 閘道的主機名稱，請執行下列動作，<sitewise_gateway_ip>將取代為 SiteWise Edge 閘道的 IP：


```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | grep -Po  
'CN = \K.*' | head -1
```

執行下列命令以匯出主機名稱以供日後使用，<your_edge_gateway_hostname>將 取代為 SiteWise Edge 閘道的主機名稱：

```
export GATEWAY_HOSTNAME='<your_edge_gateway_hostname>'
```

步驟 3：取得 SiteWise Edge 閘道的臨時登入資料

現在您已擁有已簽署的憑證和 SiteWise Edge 閘道的主機名稱，您需要取得暫時憑證，才能在閘道上執行 APIs。您可以透過 取得這些登入資料，AWS OpsHub AWS IoT SiteWise 或使用 APIs 直接從 SiteWise Edge 閘道取得這些登入資料。

Important

登入資料每隔 4 小時過期，因此您應該在 SiteWise Edge 閘道上使用 APIs 之前取得登入資料。請勿快取登入資料超過 4 小時。

使用 AWS OpsHub for 取得臨時登入資料 AWS IoT SiteWise

Note

您需要已安裝[AWS OpsHub AWS IoT SiteWise 應用程式的](#)。

若要使用 AWS OpsHub 讓 AWS IoT SiteWise 應用程式取得您的臨時登入資料，請執行下列動作：

1. 登入應用程式。
2. 選擇設定。
3. 針對身分驗證，選擇複製登入資料。
4. 展開適合您環境的選項，然後選擇複製。
5. 儲存登入資料以供日後使用。

使用 SiteWise Edge 閘道 API 取得臨時憑證

若要使用 SiteWise Edge 閘道 API 取得您可以使用 Python 指令碼或 curl 的臨時登入資料，您必須先擁有 SiteWise Edge 閘道的使用者名稱和密碼。SiteWise Edge 閘道使用 SigV4 身分驗證和授權。如需新增使用者的詳細資訊，請參閱 [LDAP](#) 或 [Linux 使用者集區](#)。這些登入資料將用於下列步驟，以在 SiteWise Edge 閘道上取得使用 AWS IoT SiteWise APIs。

Python

Note

您需要安裝 [urllib3](#) 和 [Python3](#)。

使用 Python 取得登入資料

1. 建立名為 `get_credentials.py` 的檔案，並將下列程式碼複製到其中。

```
'''
The following demonstrates how to get the credentials from the SiteWise Edge
gateway. You will need to add local users or connect your system to LDAP/AD
https://docs.aws.amazon.com/iot-sitewise/latest/userguide/manage-gateways-
ggv2.html#create-user-pool

Example usage:
    python3 get_credentials.py -e https://<gateway_hostname> -c
    <path_to_certificate> -u '<gateway_username>' -p '<gateway_password>' -m
    '<method>'
'''

import urllib3
import json
import urllib.parse
import sys
import os
import getopt

'''
This function retrieves the AWS IoT SiteWise Edge gateway credentials.
'''

def get_credentials(endpoint, certificatePath, user, password, method):
    http = urllib3.PoolManager(cert_reqs='CERT_REQUIRED', ca_certs=
    certificatePath)
```

```

    encoded_body = json.dumps({
        "username": user,
        "password": password,
        "authMechanism": method,
    })

    url = urllib.parse.urljoin(endpoint, "/authenticate")

    response = http.request('POST', url,
        headers={'Content-Type': 'application/json'},
        body=encoded_body)

    if response.status != 200:
        raise Exception(f'Failed to authenticate! Response status
{response.status}')

    auth_data = json.loads(response.data.decode('utf-8'))

    accessKeyId = auth_data["accessKeyId"]
    secretAccessKey = auth_data["secretAccessKey"]
    sessionToken = auth_data["sessionToken"]
    region = "edge"

    return accessKeyId, secretAccessKey, sessionToken, region

def print_help():
    print('Usage:')
    print(f'{os.path.basename(__file__)} -e <endpoint> -c <path/to/certificate>
-u <user> -p <password> -m <method> -a <alias>')
    print('')
    print('-e, --endpoint    edge gateway endpoint. Usually the Edge gateway
hostname.')
    print('-c, --cert_path path to downloaded gateway certificate')
    print('-u, --user        Edge user')
    print('-p, --password    Edge password')
    print('-m, --method      (Optional) Authentication method (linux, winnt,
ldap), default is linux')
    sys.exit()

def parse_args(argv):
    endpoint = ""
    certificatePath = None
    user = None

```

```

password = None
method = "linux"

try:
    opts, args = getopt.getopt(argv, "he:c:u:p:m:",
["endpoint=", "cert_path=", "user=", "password=", "method="])
except getopt.GetoptError:
    print_help()

for opt, arg in opts:
    if opt == '-h':
        print_help()
    elif opt in ("-e", "--endpoint"):
        endpoint = arg
    elif opt in ("-u", "--user"):
        user = arg
    elif opt in ("-p", "--password"):
        password = arg
    elif opt in ("-m", "--method"):
        method = arg.lower()
    elif opt in ("-c", "--cert_path"):
        certificatePath = arg

if method not in ['ldap', 'linux', 'winnt']:
    print("not valid method parameter, required are ldap, linux, winnt")
    print_help()

if (user == None or password == None):
    print("To authenticate against edge user, password have to be passed
together, and the region has to be set to 'edge'")
    print_help()

if(endpoint == ""):
    print("You must provide a valid and reachable gateway hostname")
    print_help()

return endpoint,certificatePath, user, password, method

def main(argv):
    # get the command line args
    endpoint, certificatePath, user, password, method = parse_args(argv)

```

```

    accessKeyId, secretAccessKey, sessionToken, region=get_credentials(endpoint,
certificatePath, user, password, method)

    print("Copy and paste the following credentials into the shell, they are
valid for 4 hours:")
    print(f"export AWS_ACCESS_KEY_ID={accessKeyId}")
    print(f"export AWS_SECRET_ACCESS_KEY={secretAccessKey}")
    print(f"export AWS_SESSION_TOKEN={sessionToken}")
    print(f"export AWS_REGION={region}")
    print()

if __name__ == "__main__":
    main(sys.argv[1:])

```

2. 從終端機使用您建立 <gateway_password> 的登入資料取代 <gateway_username> 和 , 執行 get_credentials.py。

```

python3 get_credentials.py -e https://$GATEWAY_HOSTNAME -c $PATH_TO_CERTIFICATE
-u '<gateway_username>' -p '<gateway_password>' -m 'linux'

```

curl

Note

您需要安裝 [curl](#)。

使用 curl 取得登入資料

1. 從終端機執行下列命令，將 <gateway_username> 和 <gateway_password> 取代為您建立的登入資料。

```

curl --cacert $PATH_TO_CERTIFICATE --location \
-X POST https://$GATEWAY_HOSTNAME:443/authenticate \
--header 'Content-Type: application/json' \
--data-raw '{
    "username": "<gateway_username>",
    "password": "<gateway_password>",

```

```
"authMechanism": "linux"
}'
```

回應如下所示：

```
{
  "username": "sweuser",
  "accessKeyId": "<accessKeyId>",
  "secretAccessKey": "<secretAccessKey>",
  "sessionToken": "<sessionToken>",
  "sessionExpiryTime": "2022-11-17T04:51:40.927095Z",
  "authMechanism": "linux",
  "role": "edge-user"
}
```

2. 從您的終端機中執行下列命令。

```
export AWS_ACCESS_KEY_ID=<accessKeyId>
export AWS_SECRET_ACCESS_KEY=<secretAccessKey>
export AWS_SESSION_TOKEN=<sessionToken>
export AWS_REGION=edge
```

步驟 4：取得 SiteWise Edge 閘道上的資產模型清單

現在您已擁有已簽署的憑證、SiteWise Edge 閘道主機名稱和 SiteWise Edge 閘道的臨時憑證，您可以使用 ListAssetModels API 取得 SiteWise Edge 閘道上的資產模型清單。

Python

Note

您需要安裝 [Python3](#)、[Boto3](#) 和 [BotoCore](#)。

使用 Python 取得資產模型的清單

1. 建立名為 list_asset_model.py 的檔案，並將下列程式碼複製到其中。

```
import json
import boto3
import botocore
```

```
import os

# create the client using the credentials
client = boto3.client("iotsitewise",
    endpoint_url= "https://" + os.getenv("GATEWAY_HOSTNAME"),
    region_name=os.getenv("AWS_REGION"),
    aws_access_key_id=os.getenv("AWS_ACCESS_KEY_ID"),
    aws_secret_access_key=os.getenv("AWS_SECRET_ACCESS_KEY"),
    aws_session_token=os.getenv("AWS_SESSION_TOKEN"),
    verify=os.getenv("PATH_TO_CERTIFICATE"),
    config=botocore.config.Config(inject_host_prefix=False))

# call the api using local credentials
response = client.list_asset_models()
print(response)
```

2. 從終端機執行 list_asset_model.py。

```
python3 list_asset_model.py
```

curl

 Note

您需要安裝 [curl](#)。

使用 curl 取得資產模型的清單

從終端機執行下列命令。

```
curl \
  --request GET https://$GATEWAY_HOSTNAME:443/asset-models \
  --cacert $PATH_TO_CERTIFICATE \
  --aws-sigv4 "aws:amz:edge:iotsitewise" \
  --user "$AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY" \
  -H "x-amz-security-token:$AWS_SESSION_TOKEN"
```

回應應如下所示：

```
{
```

```
"assetModelSummaries": [
  {
    "arn": "arn:aws:iotsitewise:{region}:{account-id}:asset-model/{asset-model-id}",
    "creationDate": 1.669245291E9,
    "description": "This is a small example asset model",
    "id": "{asset-model-id}",
    "lastUpdateDate": 1.669249038E9,
    "name": "Some Metrics Model",
    "status": {
      "error": null,
      "state": "ACTIVE"
    }
  },
  .
  .
  .
],
"nextToken": null
}
```

在上託管 SiteWise Edge 閘道 Siemens Industrial Edge

使用 Edge 應用程式在 Siemens Industrial AWS IoT SiteWise Edge 上託管您的閘道。如同一樣 AWS IoT Greengrass V2，您可以使用上的 SiteWise Edge 來最佳化製造程序或改善操作工作流程Siemens Industrial Edge。

您可以在裝置上執行 SiteWise Edge 閘道，將資料從 Siemens Industrial Edge 裝置擷取至 AWS 您的帳戶。若要這樣做，請向 SiteWise AWS IoT SiteWise Edge 支援團隊請求存取 Edge 應用程式。然後，使用 Siemens Industrial Edge 裝置的部署目標建立新的 SiteWise Edge 閘道資源。接著，下載組態檔案，並透過Siemens Industrial Edge Management入口網站將其上傳至您的應用程式。如需在上執行應用程式的詳細資訊Siemens Industrial Edge，包括如何設定所需Siemens資源，請參閱 Siemens 文件中的[什麼是 Industrial Edge ?](#)。

Note

Siemens 不是 SiteWise Edge 的廠商或供應商。Siemens Industrial Edge Marketplace 是一個獨立的市場。

主題

- [安全](#)
- [Siemens Secure Storage 和 AWS IoT SiteWise Edge 應用程式](#)
- [Siemens Industrial Edge 裝置目的地](#)
- [從預覽應用程式遷移](#)
- [故障診斷](#)
- [AWS IoT SiteWise Edge 應用程式變更日誌](#)
- [AWS IoT SiteWise Edge 應用程式的需求](#)
- [建立 的閘道 Siemens Industrial Edge](#)
- [Siemens Databus user 為應用程式建立](#)
- [存取 AWS IoT SiteWise Edge 應用程式](#)
- [在Siemens裝置上安裝應用程式](#)
- [更新 AWS IoT SiteWise Edge 應用程式組態](#)

安全

作為 AWS 我們客戶和合作夥伴之間[共同責任模型](#)的一部分，以下說明誰負責不同的安全層面：

客戶責任

- 審核合作夥伴。
- 設定提供給合作夥伴的網路存取權。
- 實體保護執行 SiteWise Edge 的裝置。

AWS 責任

- 將合作夥伴與客戶 AWS 雲端資源隔離。

合作夥伴責任

- 使用安全預設值。
- 透過修補程式和其他適當的更新，在一段時間內保護解決方案的安全。
- 將客戶資料保密。
- 審核合作夥伴市集中可用的其他應用程式。

Siemens Secure Storage 和 AWS IoT SiteWise Edge 應用程式

為了保護執行 AWS IoT SiteWise Edge 應用程式所需的登入資料和秘密，Siemens Industrial Edge 提供機制來安全地將登入資料存放在裝置上。如果 AWS IoT SiteWise Edge 應用程式不支援安全存放這些登入資料，就無法在裝置上執行。缺少安全儲存支援所造成的執行失敗會記錄在日誌檔案中。

安裝和執行 AWS IoT SiteWise Edge 應用程式需要下列最低作業系統版本。將您的裝置升級至最新版本以安裝應用程式。

- 對於虛擬裝置：EVD 1.19 版或更新版本
- 對於實體裝置：IED-OS 2.2 版或更新版本

在您升級裝置之前，上的 AWS IoT SiteWise Edge 應用程式 Siemens Industrial Edge 不會執行。

Siemens Industrial Edge 裝置目的地

在上使用 AWS IoT SiteWise Edge 應用程式時 Siemens Industrial Edge，目的地會先協助準備資料，再傳送至 AWS IoT SiteWise 進行進一步分析和分發。您可以使用 Amazon S3 設定緩衝資料擷取的資料目的地設定，或使用即時資料擷取。兩者都可讓您使用 Siemens Industrial Edge 裝置部署目標上的路徑篩選條件來訂閱 MQTT 主題。

閘道上的 Siemens Industrial Edge 部署目標支援兩種主要資料處理方法：

- AWS IoT SiteWise 即時設定 - 資料在收集 AWS IoT SiteWise 時直接傳送到
- AWS IoT SiteWise 使用 Amazon S3 設定緩衝 - 在傳送至 Amazon S3 之前，會批次暫時收集和存放資料

如需設定這些選項的詳細資訊，請參閱 [使用 Amazon S3 新增 AWS IoT SiteWise 緩衝目的地](#) 和 [新增 AWS IoT SiteWise Edge 即時目的地](#)。

路徑篩選條件的字首

使用 Siemens Industrial Edge 部署目標的閘道路徑篩選條件會結合主題和資料串流名稱，為您的資料建立唯一識別符。具有資料串流名稱的組合主題在 Siemens Industrial Edge 閘道中稱為字首。這與自我託管閘道不同，其中路徑篩選條件僅以 MQTT 主題為基礎。

Example Siemens 資料串流的路徑篩選結構

Siemens 資料串流的典型路徑篩選條件包含主題路徑和資料串流名稱：

```
ie/d/device1/application1/datastream1
```

其中：

- `ie/d/` 是 Siemens 資料串流所需的字首
- `device1/application1` 代表階層路徑
- `datastream1` 是特定的資料串流名稱

Note

使用 Siemens Industrial Edge 資料串流時，請確定您在路徑篩選條件中同時包含中繼資料 (`ie/m/`) 和資料 (`ie/d/`) 主題，以接收資料串流的完整資訊。

目的地和路徑篩選條件

檢視下列主題，進一步了解啟用 MQTT 閘道中的目的地和路徑篩選條件：

- [了解 AWS IoT SiteWise Edge 目的地](#)
- [新增 AWS IoT SiteWise Edge 即時目的地](#)
- [使用 Amazon S3 新增 AWS IoT SiteWise 緩衝目的地](#)
- [了解 AWS IoT SiteWise Edge 目的地的路徑篩選條件](#)
- [將路徑篩選條件新增至 AWS IoT SiteWise Edge 目的地](#)
- [管理 AWS IoT SiteWise Edge 目的地](#)

從預覽應用程式遷移

如果您在預覽階段 Siemens Industrial Edge 於上執行 SiteWise Edge，您將需要從預覽版本 1.0.1 升級至最新版本。執行下列動作來遷移：

1. 建立新的 SiteWise Edge 閘道。如需詳細資訊，請參閱[建立的閘道 Siemens Industrial Edge](#)。
2. Siemens Databus user 為每個新閘道建立新的。如需詳細資訊，請參閱[Siemens Databus user 為應用程式建立](#)。
3. 在您的 IED 上解除安裝 1.0.1 AWS IoT SiteWise Edge 版閘道應用程式。

 Note

當您重新設定 AWS IoT SiteWise Edge 應用程式預覽版本先前使用的 AWS IoT SiteWise 資產時，請準備資料流程中斷。保留資料歷史記錄時，當您重新安裝新的閘道時，可能會遺失資料。

4. 在 中刪除您在預覽期間建立的 SiteWise Edge 閘道[AWS IoT SiteWise 主控台](#)。
5. 使用新的閘道組態檔案在 IED 上安裝 AWS IoT SiteWise Edge 閘道應用程式。如需詳細資訊，請參閱[在 Siemens 裝置上安裝應用程式](#)。

 Important

安裝新的閘道會覆寫 SiteWise Edge 應用程式的預覽版本。安裝 2.0.0 版後，無法返回 1.0.1 版。

設定新的閘道和 之後 Siemens Databus user，您的資料會流向 屬性。

您也可以直接從版本升級 SiteWise Edge 應用程式。不過，仍然需要新的閘道組態。

故障診斷

若要對 Siemens Industrial Edge 裝置上的 SiteWise Edge 閘道進行故障診斷，請參閱 [對上的 AWS IoT SiteWise Edge 應用程式進行故障診斷 Siemens Industrial Edge](#)。

您也可以存取 [AWS re : Post](#) 來尋找問題的答案。

AWS IoT SiteWise Edge 應用程式變更日誌

下表說明 AWS IoT SiteWise Edge 應用程式每個版本中的變更。

版本	變更	
3.0.0	新功能 在 AWS IoT SiteWise 主控台中為安裝在 上的閘道新增其他組態選項 Siemens Industrial Edge：	

版本	變更	
	• 新增和設定 Amazon S3 目的地。 • 移除 AWS IoT SiteWise 即時目的地。 • 為每個目的地新增篩選條件，以控制資料路由。 • 新增全域字首組態的選項，以從Siemens IE安裝程式中的 IoT SiteWise 發佈者傳送所有資料。字首會在篩選後套用。 錯誤修正和改善 • 新增在 AWS IoT SiteWise Edge 應用程式初次嘗試失敗時重試 Siemens Industrial Edge Databus 連線的功能。	
2.0.1	錯誤修正和改善 • 修正應用程式在啟動時無法擷取 AWS 登入資料時，會進入故障狀態並結束的問題。新增支援以重試，直到成功擷取登入資料為止。	
2.0.0	• AWS IoT SiteWise Edge 應用程式現已正式推出。 • 應用程式需要 Siemens IEVD 1.19 版或 Siemens IED-OS 2.2 版。 • 效能改善：減少記憶體和 CPU 使用量。 • 偵錯改進：您現在可以上傳選用的組態檔案，以啟用偵錯日誌。 • 安全性增強功能：應用程式使用 SecureStorage API將登入資料安全地存放在裝置上。 • Docker 摘要值： sha256:4a960f29234a190ebb5224c1fd0f3e99faafccc4cb3d93ca13fef247b6656d18	
1.0.1	初始版本	

AWS IoT SiteWise Edge 應用程式的需求

若要在 上執行 AWS IoT SiteWise EdgeSiemens Industrial Edge，您需要下列項目：

- [Siemens Digital Exchange Platform](#) 帳戶。
- Siemens Industrial Edge Hub (iehub) 帳戶。
- Siemens Industrial Edge Management 執行個體。
 - IE 應用程式組態服務。若要進一步了解，請參閱 Siemens Industrial Edge Management 文件 [Installing the IE App Configuration Service manually](#) 中的。
- 存取 AWS IoT SiteWise Edge 應用程式的 2.0.1 版或更新版本。如需詳細資訊，請參閱 [存取 AWS IoT SiteWise Edge 應用程式](#)。
- Siemens Industrial Edge 裝置 (IED) 或 Siemens Industrial Edge 虛擬裝置 (IEVD)。
 - 至少 15 GB 的磁碟空間，以滿足硬體需求。
 - 1 GB 的 RAM 以及額外的 1 GB 交換記憶體。
 - 裝置組態，以允許連接埠 443 和 8883 上的傳出流量。
 - x86-64 位元處理器。
 - Siemens Industrial Edge Management 1.13.10 版或更新版本。
 - 裝置符合 Siemens Secure Storage 要求。
 - 在虛擬裝置上，EVD 1.19 版或更新版本。
 - 在實體裝置上，IED-OS 2.2 版或更新版本。
 - Docker Compose 的最新版本。
 - Docker Engine 18.091 版或更新版本。
- 必要的網域存取。如需詳細資訊，請參閱 [AWS IoT SiteWise 端點](#)。

建立的閘道 Siemens Industrial Edge

在您擁有適當的 Siemens 帳戶和 IEM 執行個體之後，您可以建立部署類型的 SiteWise Edge 閘道 Siemens Industrial Edge 裝置。

Note

請確定您符合在 上執行裝置的所有需求 Siemens Industrial Edge Management。如需詳細資訊，請參閱 [AWS IoT SiteWise Edge 應用程式的需求](#)。

建立組態檔案

1. 導覽至 [AWS IoT SiteWise 主控台](#)。

2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇 Create gateway (建立閘道)。
4. 針對部署類型，選擇 Siemens Industrial Edge 裝置。
5. 輸入 SiteWise Edge 閘道的名稱，或使用 產生的名稱 AWS IoT SiteWise。
6. (選用) 在進階組態下，執行下列動作：
 - 輸入 AWS IoT Core 物件的名稱，或使用 產生的名稱 AWS IoT SiteWise。
7. 選擇 Create gateway (建立閘道)。
8. 在產生 SiteWise Edge 閘道組態檔案對話方塊中，選擇產生並下載。AWS IoT SiteWise 自動產生組態檔案，您將用來設定 AWS IoT SiteWise Edge 應用程式。

Important

您可以使用閘道組態檔案來備份和還原 AWS IoT SiteWise Edge 應用程式。將 SiteWise Edge 閘道組態檔案儲存在 [AWS Secrets Manager](#)，以安全地存放和管理檔案。Secrets Manager 可安全地存放、管理和擷取敏感資訊。

Siemens Databus user 為應用程式建立

AWS IoT SiteWise Edge on 從Siemens Databus應用程式Siemens Industrial Edge擷取資料。若要將 SiteWise Edge 連線至 Siemens Databus，您需要Siemens Databus user提供存取您要安全傳輸之資料的 AWS IoT SiteWise。若要開始，請建立 [Siemens Databus user](#)，然後將登入資料提供給 SiteWise Edge 應用程式。

建立 Siemens Databus user

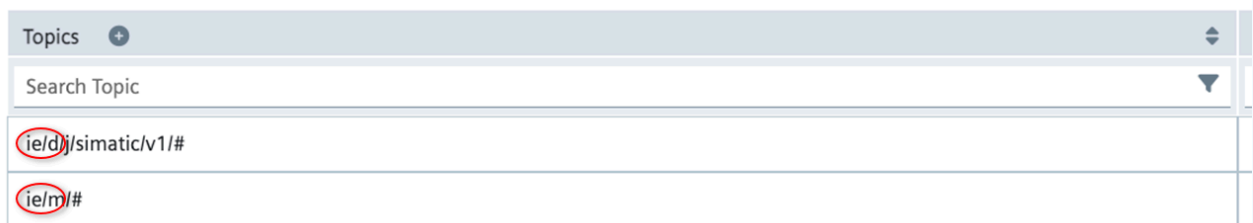
1. 在Siemens Industrial Edge Management執行個體中，選擇平台應用程式區段中的邊緣管理。
2. 選擇 資料連線 圖示。
3. 選取 Databus。您的連線裝置清單隨即顯示。
4. 選取要連線至 AWS IoT SiteWise Edge 應用程式的裝置。
5. 選擇啟動。Databus Configurator 您所選裝置的 隨即出現。
6. 在使用者下為您的 Edge 裝置建立使用者。如需建立使用者的詳細資訊，請參閱 Siemens Industrial Edge Management 文件中的[使用者](#)。
7. 選取Siemens Databus應可存取的主題。這些主題會限制 AWS IoT SiteWise Edge 可存取的內容。

⚠ Important

Siemens Databus user 可存取 的所有主題都會發佈至 AWS IoT SiteWise。

ℹ Note

Siemens Databus user需要同時存取資料和中繼資料主題。開頭為 的主題ie/d是資料主題。開頭為 的主題ie/m是中繼資料主題。以對方式共用主題，讓 SiteWise Edge 可以存取每個個別主題的資料和中繼資料。



8. 為您的Siemens Databus組態設定適當的許可。

建立Siemens Databus組態後，您可以在 上安裝 AWS IoT SiteWise Edge 應用程式Siemens Industrial Edge Management。如需詳細資訊，請參閱[在Siemens裝置上安裝應用程式](#)。

您也可以選擇性地設定Siemens Industrial Edge閘道的目的地和路徑篩選條件。如需詳細資訊，請參閱[目的地和路徑篩選條件](#)。

存取 AWS IoT SiteWise Edge 應用程式

若要存取 上的 AWS IoT SiteWise Edge 應用程式Siemens Industrial Edge，[請傳送電子郵件](#)請求 SiteWise Edge 支援團隊的存取權。

在您的電子郵件中包含下列資訊：

- 您的姓名和聯絡資訊
- 公司名稱
- Siemens Industrial Edge 租戶 ID

在Siemens裝置上安裝應用程式

傳送電子郵件給 SiteWise Edge 支援團隊以取得 AWS IoT SiteWise Edge 應用程式的存取權後 Siemens Industrial Edge，請將應用程式指派給 的執行個體Siemens Industrial Edge Management。然後，您可以在裝置上安裝 AWS IoT SiteWise Edge 應用程式。

安裝 AWS IoT SiteWise Edge 應用程式

1. 確認 中提供的Docker摘要Siemens Industrial Edge Management符合 中列出的最新版本[AWS IoT SiteWise Edge 應用程式變更日誌](#)。

如需尋找 Siemens Docker 摘要值的詳細資訊，請參閱 Siemens 文件Siemens Industrial Edge的裝置中的[管理應用程式](#)。

Siemens Industrial Edge Management 一次支援一個版本的 AWS IoT SiteWise Edge 應用程式。執行此步驟，以確保您在Siemens Industrial Edge裝置上安裝 AWS IoT SiteWise Edge 應用程式之前，使用最新版本的應用程式。

2. 將 AWS IoT SiteWise Edge 應用程式指派給 Siemens Industrial Edge Management。如需詳細資訊，請參閱 Siemens 文件的工業邊緣管理一節中的[管理應用程式](#)。
3. 在 Edge Management 中，瀏覽 AWS IoT SiteWise Edge 的目錄，然後選擇它。
4. 選擇 Install (安裝)。

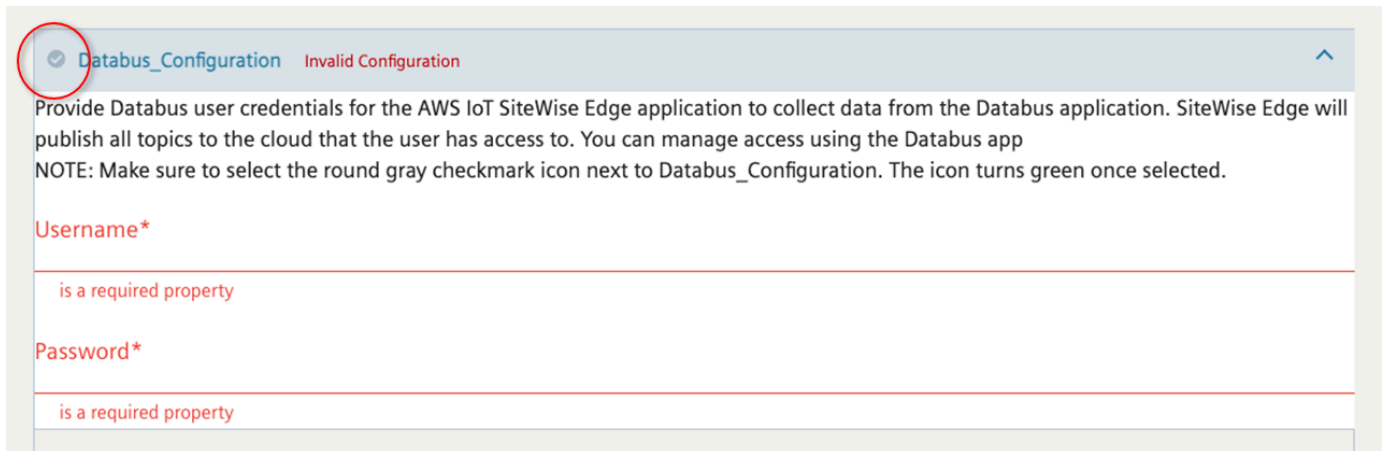
Note

如果顯示聯絡我們按鈕，請選擇該按鈕，然後依照步驟請求存取 上的 AWS IoT SiteWise Edge 應用程式Siemens Industrial Edge。如需詳細資訊，請參閱[存取 AWS IoT SiteWise Edge 應用程式](#)。

5. 在結構描述組態選項中選取 Databus_Configuration。
6. 輸入 Databus 組態的使用者名稱和密碼。如需建立 的詳細資訊Siemens Databus user，請參閱[Siemens Databus user 為應用程式建立](#)。
7. 選擇 Databus_Configuration 旁的小型圓形灰色核取記號圖示，將圖示變成綠色。

Note

輸入組態只有在核取記號圖示從灰色變更為綠色時才適用。否則，會忽略輸入組態。



 **Databus_Configuration** Invalid Configuration 

Provide Databus user credentials for the AWS IoT SiteWise Edge application to collect data from the Databus application. SiteWise Edge will publish all topics to the cloud that the user has access to. You can manage access using the Databus app

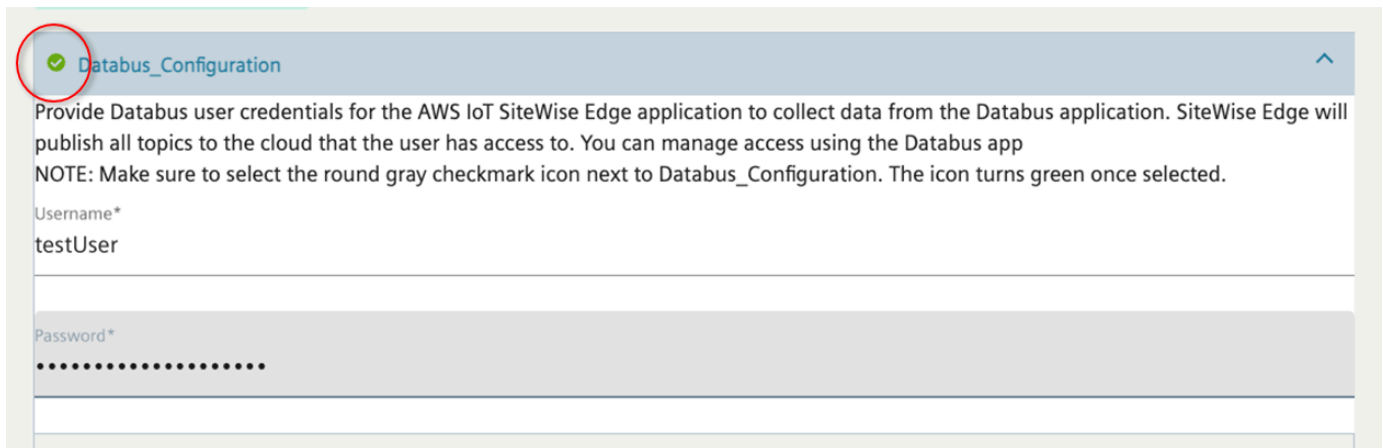
NOTE: Make sure to select the round gray checkmark icon next to Databus_Configuration. The icon turns green once selected.

Username*

is a required property

Password*

is a required property



 **Databus_Configuration** 

Provide Databus user credentials for the AWS IoT SiteWise Edge application to collect data from the Databus application. SiteWise Edge will publish all topics to the cloud that the user has access to. You can manage access using the Databus app

NOTE: Make sure to select the round gray checkmark icon next to Databus_Configuration. The icon turns green once selected.

Username*

testUser

Password*

.....

8. 選擇下一步以移至其他組態，您可以在其中上傳閘道組態檔案。
9. 選擇 SiteWise_Edge_Gateway_Config 作為上傳閘道組態檔案的位置。

 Note

請確定您選擇 SiteWise_Edge_Gateway_Config，而不是 SiteWise_Edge_Support_Config_Optional。

10. 選取裝置以安裝應用程式。
11. 選擇立即安裝。

您可以選擇性地設定發佈者元件，將資料匯出至 AWS 雲端。如需詳細資訊，請參閱[設定 AWS IoT SiteWise 發佈者元件](#)。

若要設定Siemens Industrial Edge閘道的目的地，請參閱 [目的地和路徑篩選條件](#)。

更新 AWS IoT SiteWise Edge 應用程式組態

在 上更新 AWS IoT SiteWise Edge 應用程式組態時，有幾件事需要考慮 Siemens Industrial Edge。

Note

AWS IoT SiteWise Edge 應用程式組態的任何變更都需要重新啟動應用程式。

重新啟動 AWS IoT SiteWise Edge 應用程式的原因

- Edge 應用程式的新 Siemens Databus user AWS IoT SiteWise 。
- 閘道組態檔案（您的 SiteWise_Edge_Gateway_Config 檔案）的變更。
- 代理組態更新（也需要完整 IEVD 重新啟動）
- 啟用偵錯日誌以偵錯問題

重新啟動應用程式

1. 在 Siemens Industrial Edge Management 執行個體中，選擇平台應用程式區段中的邊緣管理。
2. 選擇我的已安裝應用程式。
3. 選取 AWS IoT SiteWise Edge 應用程式。
4. 選擇重新啟動。

目的地和路徑篩選條件

AWS IoT SiteWise Edge 中的目的地提供靈活且有效率的方式來管理工業資料如何從邊緣裝置流向雲端。本節說明如何設定目的地、使用路徑篩選條件路由特定資料串流，以及為您的使用案例選擇正確的目的地類型。

您可以在啟用自我託管 MQTT 的 V3 閘道和與上託管的 AWS IoT SiteWise Edge 應用程式搭配使用的閘道上使用目的地和路徑篩選條件 Siemens Industrial Edge。目的地和路徑篩選條件不適用於 Classic Streams、V2 閘道。

主題

- [了解 AWS IoT SiteWise Edge 目的地](#)
- [了解 AWS IoT SiteWise Edge 目的地的路徑篩選條件](#)

- [新增 AWS IoT SiteWise Edge 即時目的地](#)
- [使用 Amazon S3 新增 AWS IoT SiteWise 緩衝目的地](#)
- [將路徑篩選條件新增至 AWS IoT SiteWise Edge 目的地](#)
- [管理 AWS IoT SiteWise Edge 目的地](#)

了解 AWS IoT SiteWise Edge 目的地

使用 AWS IoT SiteWise Edge 目的地來判斷傳送來源資料的位置。您可以根據所需的特定特性來選擇資料目的地，例如成本效益、低延遲或儲存需求。整合由 AWS IoT SiteWise、我們的合作夥伴或自訂應用程式擷取的裝置資料，以在邊緣發佈和訂閱路徑篩選條件（主題）。然後，您可以將裝置資料建模、傳輸和存放在雲端。

Note

若要完整使用自我託管閘道上的所有目的地功能，請升級至最新版本的 IoT SiteWise 發佈者和 IoT SiteWise OPC UA 收集器。在 Classic 串流、V2 閘道上繼續支援串流，以維持與現有設定的相容性。如需詳細資訊，請參閱[傳統串流、適用於 AWS IoT SiteWise Edge 的 V2 閘道](#)

主題

- [SiteWise Edge 目的地如何增強資料管理](#)
- [目的地類型](#)
- [比較閘道版本之間的目的地功能](#)
- [目的地限制](#)
- [SiteWise Edge 目的地的使用案例](#)

SiteWise Edge 目的地如何增強資料管理

AWS IoT SiteWise 即時將資料從邊緣匯出到 [Amazon S3](#) 分批匯出。

目的地可增強您 AWS IoT SiteWise 環境中的靈活性和可擴展性。目的地會實作集中式資料管理模型，其中來源會將資料發佈至中央系統。目的地會使用路徑篩選條件來判斷資料的傳送位置。目的地可以訂閱多個路徑篩選條件。

啟用 MQTT 的閘道，無論是自我託管還是在 [AWS IoT SiteWise Edge](#) 上執行 Siemens Industrial Edge，請使用 MQTT 進行本機通訊，並隨附預設即時目的地，其篩選條件設定為 #。這表示根據預設，所有主題上的所有訊息都會發

佈到 AWS IoT SiteWise 即時目的地。如需詳細資訊，請參閱[了解 AWS IoT SiteWise Edge 目的地的路徑篩選條件](#)。您可以在每個閘道中新增一個即時目的地。

目的地類型

為閘道設定目的地時，您有兩個主要選項：使用的即時組態 AWS IoT SiteWise，以及使用 Amazon S3 的緩衝組態。每個目的地類型都有自己的一組設定和考量事項。

AWS IoT SiteWise 即時設定

選擇此選項可將資料直接傳送到 AWS IoT SiteWise 熱層儲存，以方便即時擷取和監控資料。即時設定會管理資料流程，特別是當閘道與雲端發生連線問題時。在連線中斷期間，資料會暫時儲存在本機的閘道上。重新建立連線後，儲存的資料會自動傳送至雲端。

您可以調整資料發佈程序的各個層面，例如要在本機存放的最大資料量、重新連線時將資料傳送至雲端的速率，以及在儲存達到其容量後何時刪除資料。

如需 AWS IoT SiteWise 儲存層的詳細資訊，請參閱[在中管理資料儲存 AWS IoT SiteWise](#)。

AWS IoT SiteWise 使用 Amazon S3 設定緩衝

此目的地類型可讓您在本地緩衝閘道上的資料，並定期將其批次傳送至 Amazon S3 儲存貯體。資料會以有效的 Parquet 格式儲存，並針對分析工作負載進行最佳化。資料在 Amazon S3 中後，您可以將其匯入至 AWS IoT SiteWise 以進行儲存、處理和分析。

選擇此選項以批次擷取資料，並以經濟實惠的方式存放歷史資料。您可以設定偏好的 Amazon S3 儲存貯體位置，以及您希望將資料上傳至 Amazon S3 的頻率。您也可以選擇在擷取資料之後如何處理資料 AWS IoT SiteWise。您可以選擇在 SiteWise 和 Amazon S3 中同時提供資料，也可以選擇從 Amazon S3 自動刪除資料。

比較閘道版本之間的目的地功能

啟用 MQTT 的閘道中的目的地功能可簡化資料流程管理。目的地透過將資料路由至各種端點的集中式組態來簡化資料管理。這種方法不需要複雜的個別串流設定，讓整體系統更具彈性且更容易管理。

相較之下，Classic 串流、V2 閘道、SiteWise Edge AWS IoT Greengrass 會透過串流將資料從資料來源傳輸到發佈者，為每個資料來源個別設定資料目的地。

使用 AWS IoT SiteWise 目的地功能時，發佈者路由組態會合併。目的地組態可讓您集中管理目的地和路徑篩選條件。您可以輕鬆新增目的地、管理路徑篩選條件、刪除不必要的篩選條件或目的地，視您的需求而定。

此外，目的地功能使用 MQTT（訊息佇列遙測傳輸），這是一種產業標準通訊協定，廣泛用於工業 IoT 應用程式。這種採用 MQTT AWS IoT SiteWise 有助於簡化與各種裝置和系統的整合。

目的地限制

SiteWise Edge 閘道上目的地的目前限制包括：

- 啟用 MQTT 的閘道不支援資料處理套件。
- 資料類型支援僅限於 AWS IoT SiteWise 資料類型。如需啟用資料類型轉換的資訊，請參閱 [轉換不支援的資料類型](#)。

SiteWise Edge 目的地的使用案例

SiteWise Edge 目的地用於各種應用程式。以下是一些關鍵範例：

工業自動化, 即時監控和預測性維護

在工業設定中，工廠現場的感應器和裝置可以將資料發佈到 SiteWise Edge。您可以設定目的地來篩選和路由相關資料，以便即時監控和分析機器效能。您可以使用路徑篩選條件訂閱相關 MQTT 主題、處理資料，然後發佈已處理的資料。透過這種方式，您可以選擇性地將處理的資料路由到 AWS 雲端分析服務或內部部署系統。然後，製造商可以實作預測性維護策略、最佳化生產程序，並減少停機時間。

智慧建物, 能源效率和佔用率最佳化

建置自動化系統會產生資料串流，以監控和控制建築物的各個層面，例如 HVAC 系統、照明和存取控制。使用 SiteWise Edge，這些資料串流可以擷取、處理和路由到不同的目的地。設施管理員可以設定目的地來篩選和轉送相關資料，啟用能源效率測量和佔用最佳化等進階功能，同時確保資料隱私權和合規性。

這些使用案例示範如何跨各種產業利用 SiteWise Edge 中的目的地功能，以有效率地擷取、處理和路由資料。這可啟用進階功能，例如即時監控、預測性維護、能源效率和遠端診斷，同時確保資料隱私權和合規性。

了解 AWS IoT SiteWise Edge 目的地的路徑篩選條件

主題

- [路徑篩選條件需求](#)
- [路徑篩選條件的最佳實務](#)

- [OPC UA 伺服器的路徑篩選條件](#)
- [路徑篩選條件名稱中的特殊字元](#)

每個目的地都設定為將資料路由到 AWS IoT SiteWise 或 Amazon S3。路徑篩選條件可讓您選取接收目的地的 MQTT 訊息時要篩選的特定資料。路徑篩選條件代表資料串流的邏輯名稱，做為所需 MQTT 主題的訂閱。

在 MQTT 中，資料會組織成主題，這些主題是以斜線 (/) 分隔的階層式字串。例如，裝置可能會將溫度資料發佈至主題 `home/livingroom/sensor1/temperature`。在這裡，`home/livingroom/sensor1` 代表感應器的路徑或邏輯名稱，而 `temperature` 是要發佈的資料類型。

您可以使用路徑篩選條件，使用萬用字元 (+ 和 *) 訂閱特定主題或一系列主題 #。+ 萬用字元符合主題階層中的單一層級。例如，`home/+/sensor1/temperature` 會比對 `home/livingroom/sensor1/temperature` 和 `home/bedroom/sensor1/temperature`。萬 # 用字元在篩選條件結尾使用時，會符合多個層級。

您也可以使用路徑篩選條件名稱中 MQTT 規格中通常不允許的各種字元。在 name 中使用時，這些字元不做為萬用字元。會使用編碼 AWS IoT SiteWise 來轉換這些字元，以確保 MQTT 合規，同時保留原始命名結構。此功能對於容納來自其他系統的現有命名慣例特別有用。如需詳細資訊，請參閱[路徑篩選條件名稱中的特殊字元](#)。

透過仔細選取適當的路徑篩選條件，您可以控制要傳送到特定目的地的資料。使用路徑篩選條件，根據 IoT 系統的需求量身打造資料流程。

路徑篩選條件需求

使用 輸入路徑篩選條件時 AWS IoT SiteWise 主控台，請記住下列事項：

- 路徑篩選條件以新行分隔，每行代表單獨的路徑篩選條件。
- 個別路徑篩選條件可以有 1 到 65,535 個位元組。
- 路徑篩選條件不能空白。
- 不允許空值 (U+0000)。
- 您一次最多可輸入 100 個路徑篩選條件或 65,535 個字元，以先達到限制為準。
- 閘道上所有目的地的總限制為 20,000 個路徑篩選條件。
- 您可以在路徑篩選條件名稱中使用 %、+、# 和 \$ 字元，但 AWS IoT SiteWise 會自動將其轉換為 URI 編碼。

路徑篩選條件的最佳實務

為您的 AWS IoT SiteWise 目的地建立路徑篩選條件時，請考慮下列策略以有效管理您的資料。

- 建構您的篩選條件以鏡射您的裝置階層。例如，在製造設定中，`factory/+ /machine/#`會從不同生產線的所有機器擷取資料。
- 針對裝置類型、位置或函數使用特定層級。例如 `factory/assembly-line/robot/temperature`。或者，在智慧農業中，`farm/+ /crop/+ /moisture`會監控不同欄位間各種作物的濕度等級。
- 策略性地利用萬用字元：`+`用於單一層級的變化`#`，並擷取所有後續層級。例如，`building/+ /+ /energy-consumption`會追蹤建築物中不同區域和地板的能源使用量。這會假設第一個`+`擷取所有樓層，第二個`+`擷取所有區域。
- 透過建立足以擷取相關資料的篩選條件，但足以適應未來變更的彈性，來平衡特異性和靈活性。例如，`site/+ /equipment-type/+ /measurement`允許新增新的站點或設備類型，而不變更篩選條件結構。

徹底測試您的篩選條件，以確保它們擷取預期資料，並與 IoT 系統的架構和目標保持一致。

OPC UA 伺服器的路徑篩選條件

對於 OPC UA 伺服器，您的路徑篩選條件必須對應至 OPC UA 標籤名稱。路徑篩選條件的最終層級必須完全符合 OPC UA 標籤名稱。例如，如果您的 OPC UA 標籤為 `Device1.Temperature`，您的路徑篩選條件可能為 `factory/line1/Device1.Temperature`。您可以在上述關卡中使用萬用字元，例如 `factory/+ /Device1.Temperature`跨多個生產線擷取標籤。如果您的路徑篩選條件名稱中有特殊字元，請參閱 [路徑篩選條件名稱中的特殊字元](#) 以取得詳細資訊。

路徑篩選條件名稱中的特殊字元

AWS IoT SiteWise 可容納 OPC UA 等工業通訊協定中常用的字元，通常在標準 MQTT 主題名稱中是不允許的。此功能有助於更順暢地整合工業系統與 MQTT 型架構。

Note

雖然我們的特殊字元處理有助於整合和遷移，但建議盡可能符合新實作的標準 MQTT 命名慣例，以確保更廣泛的相容性。

從工業來源接收資料時，會使用特殊字元的 URI 編碼 AWS IoT SiteWise 標準化主題名稱：

- % 變成 %25 (先編碼為逸出字元)
- # 成為 %23
- + 成為 %2B
- \$ 變成 %24 (僅在主題開始時)

此編碼可確保包含這些特殊 MQTT 字元的來源資料可安全地用作 MQTT 主題名稱，同時保留原始工業命名慣例。

Example：路徑篩選條件名稱中的特殊字元

以下是工業主題名稱如何在 AWS IoT SiteWise 路徑篩選條件中顯示的範例：

- Factory1/Line#2/Sensor+3 成為 Factory1/Line%232/Sensor%2B3
- Plant%A/Unit\$1/Temp 成為 Plant%25A/Unit%241/Temp
- Site1/#Section/+Node 成為 Site1/%23Section/%2BNode

在 中建立訂閱或檢視主題名稱時 AWS IoT SiteWise，您會看到原始、未編碼的版本。系統會自動處理編碼，以確保 MQTT 合規。

新增 AWS IoT SiteWise Edge 即時目的地

即時目的地類型可讓您將 IoT 資料直接從裝置和閘道串流到 AWS IoT SiteWise 儲存體。此選項非常適合需要立即擷取和處理產生的資料的使用案例，而不需要批次處理或緩衝。每個閘道中只能設定一個即時目的地，因為它會持續串流資料 AWS IoT SiteWise。

Note

重複TQVs 可能會導致重複充電。

新增即時目的地

使用 AWS IoT SiteWise 主控台或 AWS CLI 將即時目的地新增至已啟用 SiteWise Edge MQTT 的閘道。

Console

1. 開啟 [AWS IoT SiteWise 主控台](#)。

2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取您要新增目的地的閘道。
4. 在目的地區段中，選擇新增目的地。
5. 在新增目的地頁面上，輸入目的地詳細資訊：
 - a. 目的地名稱欄位中目的地的名稱。
 - b. 選取目的地類型的AWS IoT SiteWise 即時。
6. 透過將發佈順序設定為先發佈較舊的資料或先發佈最新資料，來設定閘道發佈順序。根據預設，閘道會先發佈最舊的資料。
7. 使用最大批次等待時間來設定發佈者在傳送一批資料之前等待的最長時間 AWS IoT SiteWise。此設定適用於每個別名。資料會存放在本機，直到：
 - 設定的時間已過，或
 - 別名收到 10 個time-quality-value (TQV) 項目

無論符合何種條件，都會先觸發要傳送至雲端的批次。

8. 若要壓縮上傳的資料，請選取上傳資料時啟用壓縮核取方塊。讓閘道在上傳資料到雲端之前壓縮資料，可降低頻寬使用量。
9. 若要篩選出過期的發佈者資料，請選取排除過期的資料核取方塊。此選擇只會將作用中和目前的資料傳送至 AWS IoT SiteWise。
10. 在截止日期欄位中，輸入資料在您的資料集內應視為過期的頻率。您可以判斷資料是以分鐘或天數來計算。最短截止期間為五分鐘。最大截止期間為 7 天。
11. 選擇性地設定本機儲存設定：
 - a. 設定保留期間頻率 – 閘道在本機存放超過截止期間的資料的時間。最短保留期間為一分鐘。

最長保留期間為 30 天，且大於或等於輪換期間。
 - b. 設定輪換期間 – 儲存超過單一檔案之截止期間的資料時所指定的時間間隔。閘道會在每個輪換期間結束時，將一批資料傳輸到下列本機目錄：`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`。

保留時間必須大於一分鐘且等於保留期間。

- c. 提供儲存容量 (GB) 值，以設定本機以 GB 儲存的資料大小上限。如果資料超過確定的本機儲存大小上限，閘道會先開始刪除最舊的資料。閘道會持續刪除，直到本機存放的資料大小等於或小於配額為止。

儲存容量必須大於或等於 1 GB。

12. 將路徑篩選條件新增至目的地。如需詳細資訊，請參閱 [將路徑篩選條件新增至 AWS IoT SiteWise Edge 目的地](#)。

如需詳細資訊，請參閱[目的地類型](#)。

AWS CLI

Example：建立新的 AWS IoT SiteWise 即時目的地

使用 [UpdateGatewayCapabilityConfiguration](#) API 來設定發佈者。

將 `capabilityNamespace` 參數設為 `iotsitewise:publisher:3`。

```
{
  "sources": [
    {
      "type": "MQTT"
    }
  ],
  "destinations": [
    {
      "type": "SITEWISE_REALTIME",
      "name": "your-destination-name",
      "config": {
        "publishingOrder": "TIME_ORDER",
        "enableCompression": true,
        "maxBatchWaitTime": "10s"
      },
      "filters": [
        {
          "type": "PATH",
          "config": {
            "paths": [
              "#"
            ]
          }
        }
      ]
    }
  ]
}
```

```

    ]
  }
]
}

```

若要更新現有的 AWS IoT SiteWise 即時目的地，請先使用 `DescribeGatewayCapabilityConfiguration` API 尋找 `destinationId`。

Example：更新 AWS IoT SiteWise 即時目的地

使用 [UpdateGatewayCapabilityConfiguration](#) API 來設定發佈者。

將 `capabilityNamespace` 參數設為 `iotsitewise:publisher:3`。

```

{
  "sources": [
    {
      "type": "MQTT"
    }
  ],
  "destinations": [
    {
      "id": "your-existing-destination-id",
      "type": "SITEWISE_REALTIME",
      "name": "your-destination-name",
      "config": {
        "publishingOrder": "TIME_ORDER",
        "enableCompression": true,
        "dropPolicy": {
          "cutoffAge": "7d",
          "exportPolicy": {
            "retentionPeriod": "7d",
            "rotationPeriod": "6h",
            "exportSizeLimitGB": 10
          }
        },
        "maxBatchWaitTime": "10s"
      },
      "filters": [
        {
          "type": "PATH",
          "config": {

```

```

    "paths": [
      "#"
    ]
  }
}
]
}
]
}
}

```

下列組態選項專屬於使用 `iotsitewise:publisher:3` 命名空間的閘道。

`sources`

定義將資料從工業設備傳輸到 的資料來源 AWS IoT SiteWise。對於啟用 MQTT 的閘道，請使用 MQTT。

類型：物件陣列

必要：是

`destinations`

定義傳送資料的位置。目的地是即時的或使用 Amazon S3 緩衝。至少需要一個目的地物件，但您可以新增空陣列。您可以為每個閘道有一個即時目的地。如需詳細資訊，請參閱[了解 AWS IoT SiteWise Edge 目的地](#)。

類型：物件陣列

必要：是

`id`

目的地的唯一識別符。您可以提供現有的目的地 ID 或將其保留空白。如果您未指定 ID，則預設會產生 UUID。

類型：字串

必要：否

`type`

目的地類型。選項包括：SITEWISE_REALTIME和 SITEWISE_BUFFERED。

- SITEWISE_REALTIME – 即時將資料直接傳送到 AWS IoT SiteWise 儲存體。
- SITEWISE_BUFFERED – 以 Parquet 格式批次傳送資料至 Amazon S3，然後匯入至 AWS IoT SiteWise 儲存體。

類型：字串

必要：是

name

目的地的唯一名稱。

類型：字串

必要：是

config

JSON 格式的目的地類型特定組態。組態會因即時和緩衝目的地而有所不同。

類型：物件

必要：是

publishingOrder

決定資料發佈的順序。資料會根據其時間戳記發佈。選項包括 TIME_ORDER 和 RECENT_DATA。

- TIME_ORDER（預設）– 先發佈較舊的資料。
- RECENT_DATA – 先發佈最新的資料。

類型：字串

必要：否

enableCompression

設定為 true，會在傳送至 之前啟用資料壓縮 AWS IoT SiteWise。讓閘道在上傳資料到雲端之前壓縮資料，可降低頻寬使用量。預設值為 true。

類型：布林值

必要：否

dropPolicy

定義如何處理較舊的資料。

類型：物件

必要：否

- `cutoffAge`

以天、小時和分鐘為單位指定要發佈的資料最長存留期。例如 7d 或 1d7h16m。不會傳送比您指定內容更舊的資料 AWS IoT SiteWise。

早於截止期間的資料不會發佈至雲端。截止時間必須介於 5 分鐘到 7 天之間。

當您指定截止存留期d時m，可以使用 h、 和 。請注意， m代表分鐘， h 代表小時， d代表天數。

類型：字串

必要：是

- `exportPolicy`

定義如何處理超過截止期限的資料。

類型：物件

必要：否

- `retentionPeriod`

儲存指定的保留期間之後，您的 SiteWise Edge 閘道會從本機儲存體刪除邊緣早於截止期間的任何資料。保留期間必須介於一分鐘到 30 天之間，且大於或等於輪換期間。

您可以在指定保留期間d時使用 mh、 和 。請注意， m代表分鐘， h 代表小時， d代表天數。

類型：字串

必要：否

- `rotationPeriod`

將早於截止期間的資料批次儲存到單一檔案的時間間隔。SiteWise Edge 閘道會在每個輪換期間結束時，將一批資料傳輸到下列本機目錄：`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`。輪換期間必須大於一分鐘，且等於或小於保留期間。

您可以在指定輪換期間d時使用 `mh`、`h` 和 `d`。請注意，`m`代表分鐘，`h` 代表小時，`d`代表天數。

類型：字串

必要：否

- `exportSizeLimitGB`

允許的資料大小上限，以 GB 為單位儲存在本機。如果違反此配額，SiteWise Edge 閘道會開始刪除最早的資料，直到本機存放的資料大小等於或小於配額為止。此參數的值必須大於或等於 1。

類型：整數

必要：否

`maxBatchWaitTime`

設定發佈者在傳送一批資料之前等待的最長時間 AWS IoT SiteWise。此設定適用於每個別名。資料會存放在本機，直到：

- 設定的時間已過，或
- 別名收到 10 個time-quality-value(TQV) 項目

使用 `m`、`h`和 `d` 來指定截止時間。請注意，`m`代表分鐘，`h` 代表小時，`d`代表天數。

類型：字串

必要：否

`filters`

要套用至資料的篩選條件。至少需要一個篩選條件。

類型：字串

必要：是

type

篩選條件類型。請使用 PATH。

類型：字串

必要：是

config

篩選條件類型特定的 JSON 格式組態。至少需要一個物件，但陣列可以是空的。

類型：物件

必要：是

- paths

路徑篩選條件的陣列。如需詳細資訊，請參閱[了解 AWS IoT SiteWise Edge 目的地的路徑篩選條件](#)。預設路徑為 #。

類型：字串陣列

必要：是

使用 Amazon S3 新增 AWS IoT SiteWise 緩衝目的地

如果您 AWS IoT SiteWise 不需要即時資料，緩衝目的地類型可讓您節省 的擷取成本。它可讓您暫時將 IoT 資料存放在 Amazon S3 儲存貯體中，然後再匯入 AWS IoT SiteWise。或者，您可以直接將資料上傳至 S3 進行儲存，無論您是否打算將其匯入 AWS IoT SiteWise。這在擷取資料之前，對於從裝置和閘道批次處理和緩衝資料非常有用 AWS IoT SiteWise。使用此選項時，資料會以設定的頻率以 Parquet 格式上傳至指定的 S3 儲存貯體。然後，您可以將此資料匯入 AWS IoT SiteWise 儲存體，以供進一步分析和處理。

使用 Amazon S3 新增緩衝的目的地

使用 AWS IoT SiteWise 主控台或 AWS CLI 將使用 Amazon S3 緩衝資料的目的地新增至 SiteWise Edge 閘道。

Console

使用 AWS Management Console 新增使用 Amazon S3 緩衝的 AWS IoT SiteWise 目的地。

1. 開啟 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取您要新增目的地的閘道。
4. 在目的地區段中，選擇新增目的地。
5. 在新增目的地頁面上，輸入目的地詳細資訊：
 - a. 目的地名稱欄位中目的地的名稱。
 - b. 選取AWS IoT SiteWise 使用 Amazon S3 for Destination type 緩衝。使用 Amazon S3 AWS IoT SiteWise 緩衝會以 Parquet 格式批次將資料傳送至 Amazon Simple Storage Service，然後將資料 AWS IoT SiteWise 匯入儲存。
6. 輸入您要存放閘道資料的位置的 Amazon S3 URL。您可以選擇瀏覽 S3 來瀏覽路徑。新增儲存貯體後，您也可以選擇檢視來檢視儲存貯體。
7. 輸入時間範圍並選取資料上傳頻率的時間增量，以指定閘道將資料上傳至 Amazon S3 的頻率。頻率值應大於 0 且小於或等於 30 天。
8. 在資料儲存設定中，決定將閘道資料匯入至 後如何處理 AWS IoT SiteWise。需要對資料儲存做出兩個決策：
 - 如果您想要將匯入的資料複製到 AWS IoT SiteWise 儲存體，請選取將資料複製到儲存體核取方塊。此選項會將從您設定的 Amazon S3 儲存貯體匯入的資料複製到 AWS IoT SiteWise 儲存體。
 - 如果您選擇將資料從 Amazon S3 儲存貯體匯入 AWS IoT SiteWise 儲存體，您也可以指定是否應在匯入完成後刪除匯入的資料。選取從 Amazon S3 刪除資料核取方塊，以在匯入至 AWS IoT SiteWise 儲存體後，從設定的 Amazon S3 儲存貯體刪除匯入的日期。
9. 將路徑篩選條件新增至目的地。如需詳細資訊，請參閱 [將路徑篩選條件新增至 AWS IoT SiteWise Edge 目的地](#)。

AWS CLI

Example：使用 Amazon S3 建立新的緩衝 AWS IoT SiteWise 目的地

使用 [UpdateGatewayCapabilityConfiguration](#) API 來設定發佈者。

將 capabilityNamespace 參數設為 iotsitewise:publisher:3。

```
{
  "sources": [
    {
```

```

        "type": "MQTT"
      }
    ],
    "destinations": [
      {
        "type": "SITEWISE_BUFFERED",
        "name": "your-s3-destination-name",
        "config": {
          "targetBucketArn": "arn:aws:s3:::amzn-s3-demo-bucket/Optional/SomeFolder",
          "publishPolicy": {
            "publishFrequency": "15m",
            "localSizeLimitGB": 10
          },
          "siteWiseImportPolicy": {
            "enableSiteWiseStorageImport": true,
            "enableDeleteAfterImport": true,
            "bulkImportJobRoleArn": "arn:aws:iam::123456789012:role/your-role-name"
          }
        }
      },
      {
        "filters": [
          {
            "type": "PATH",
            "config": {
              "paths": [
                "#"
              ]
            }
          }
        ]
      }
    ]
  }
}

```

Example：使用 Amazon S3 更新緩衝的 AWS IoT SiteWise 目的地

若要更新現有的 AWS IoT SiteWise 即時目的地，請先使用 DescribeGatewayCapabilityConfiguration API 尋找 destinationId。

發佈者命名空間：iotsitewise:publisher:3

```

{
  "sources": [

```

```

    {
      "type": "MQTT"
    }
  ],
  "destinations": [
    {
      "id": "your-existing-destination-id",
      "type": "SITEWISE_BUFFERED",
      "name": "your-s3-destination-name",
      "config": {
        "targetBucketArn": "arn:aws:s3:::amzn-s3-demo-bucket/Optional/SomeFolder",
        "publishPolicy": {
          "publishFrequency": "15m",
          "localSizeLimitGB": 10
        },
        "siteWiseImportPolicy": {
          "enableSiteWiseStorageImport": true,
          "enableDeleteAfterImport": true,
          "bulkImportJobRoleArn": "arn:aws:iam::123456789012:role/your-role-name"
        }
      },
    },
    {
      "filters": [
        {
          "type": "PATH",
          "config": {
            "paths": [
              "#"
            ]
          }
        }
      ]
    }
  ]
}

```

下列組態選項專屬於使用 `iotsitewise:publisher:3` 命名空間啟用 MQTT 的閘道。

sources

定義將資料從工業設備傳輸到 的資料來源 AWS IoT SiteWise。對於啟用 MQTT 的閘道，請使用 MQTT。

類型：物件陣列

必要：是

destinations

定義傳送資料的位置。目的地是即時的或使用 Amazon S3 緩衝。至少需要一個目的地物件，但您可以新增空陣列。您可以為每個閘道有一個即時目的地。如需詳細資訊，請參閱[了解 AWS IoT SiteWise Edge 目的地](#)。

類型：物件陣列

必要：是

id

目的地的唯一識別符。您可以提供現有的目的地 ID，或將其保留空白，讓目的地自動產生新的 ID。

類型：字串

必要：否

type

目的地類型。選項包括：SITEWISE_REALTIME和 SITEWISE_BUFFERED。選擇 SITEWISE_BUFFERED。

- SITEWISE_REALTIME（預設）– 即時將資料直接傳送到 AWS IoT SiteWise 儲存體。如需詳細資訊，請參閱[新增 AWS IoT SiteWise Edge 即時目的地](#)。
- SITEWISE_BUFFERED – 以 Parquet 格式批次傳送資料至 Amazon S3，然後匯入至 AWS IoT SiteWise 儲存體。

類型：字串

必要：是

name

目的地的唯一名稱。

類型：字串

必要：是

config

JSON 格式的目的地類型特定組態。組態會因即時和緩衝目的地而有所不同。

類型：物件

必要：是

targetBucketArn

ARN 要發佈的儲存貯體。AWS 區域 為 AWS IoT SiteWise 和 Amazon S3 選擇相同的。如果選擇字首，它必須介於 1-255 個字元之間。

 Note

AWS IoT SiteWise，包括閘道，將可存取整個指定的 S3 儲存貯體。建議使用專用儲存貯體進行緩衝資料擷取。

類型：字串

必要：是

publishPolicy

發佈政策的詳細資訊。

類型：物件

必要：是

publishFrequency

SiteWise Edge 閘道發佈至 Amazon S3 儲存貯體的頻率。資料上傳至 Amazon S3 的頻率必須大於 0 分鐘，且小於或等於 30 天。當您指定發佈頻率存留期d時m，可以使用 h、和。請注意，m代表分鐘，h 代表小時，d代表天數。預設值為 15 分鐘。

類型：字串

必要：是

localSizeLimitGB

寫入本機磁碟的檔案大小上限，以 GB 為單位。如果超過此閾值，發佈者會將所有緩衝資料發佈至其目的地。

類型：整數

必要：是

`siteWiseImportPolicy`

用於匯入資料之匯入政策的詳細資訊 AWS IoT SiteWise。

類型：物件

必要：是

`enableSiteWiseStorageImport`

將此設定為 `true`，將資料從 Amazon S3 儲存貯體匯入至 AWS IoT SiteWise 儲存體。它最初會在 中複製資料 AWS IoT SiteWise。然後，如果您 `enableDeleteAfterImport` 將 設定為 `true`，S3 中的資料會在複製到 後刪除 AWS IoT SiteWise。定價含意適用。預設值為 `true`。

類型：布林值

必要：是

`enableDeleteAfterImport`

將此設定為 `true` 以在擷取至 AWS IoT SiteWise 儲存體後刪除 Amazon S3 儲存貯體中的檔案。預設值為 `true`。

類型：布林值

必要：是

`bulkImportJobRoleArn`

AWS IoT SiteWise 擔任 在資料擷取期間從 Amazon S3 讀取緩衝資料的 IAM 角色 ARN。當邊緣裝置呼叫 AWS IoT SiteWise APIs 啟動大量匯入程序時，會使用此角色。

 Note

如果 `enableSiteWiseStorageImport` 設定為 `true`，則需要此參數。

類型：字串

必要：否

新增目的地的路徑篩選條件。如需詳細資訊，請參閱[將路徑篩選條件新增至 AWS IoT SiteWise Edge 目的地](#)。

將路徑篩選條件新增至 AWS IoT SiteWise Edge 目的地

將路徑篩選條件新增至目的地。路徑篩選條件使用 MQTT 主題語法，其中 # 是符合任意數量關卡的萬用字元，而 + 是符合單一關卡的萬用字元。您可以將多個目的地新增至閘道，每個目的地都有自己的一組路徑篩選條件訂閱您的設備遙測。

Siemens Industrial Edge 閘道使用字首來實現相容性。如需詳細資訊，請參閱[路徑篩選條件的字首](#)。

Console

新增路徑篩選條件

1. 開啟 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取您要新增路徑篩選條件的閘道。
4. 在新增目的地的路徑篩選條件區段中，選擇新增路徑篩選條件。
5. 輸入您希望此目的地訂閱的路徑篩選條件。您可以使用萬用字元 (# 和 +) 來訂閱多個路徑。
6. 選擇新增路徑篩選條件，將路徑篩選條件新增至清單。
7. 如有需要，請重複步驟來新增其他路徑篩選條件。
8. 新增所有必要路徑篩選條件後，請選擇建立。

AWS CLI for self-hosted gateways

Example：路徑篩選條件組態

```
{
  "destinations": [
    {
      ...
    }
  ],
  "filters": [
```



```
{
  "type": "PATH",
  "config": {
    "paths": [
      "home/+/sensor1/temperature",
      "home/livingroom/sensor1/temperature",
      "home/livingroom/sensor1/temperature",
      "building/#"
    ]
  }
}
```

AWS CLI for 西門子 IEGateways

Example：路徑篩選條件的字首組態

使用每個路徑篩選條件的資料 (ie/d) 和中繼資料 (ie/m) 字首來擷取所有資料。

```
{
  "destinations": [
    {
      ...
    }
  ],
  "filters": [
    {
      "type": "PATH",
      "config": {
        "paths": [
          "ie/d/home/+/sensor12/temperature",
          "ie/m/home/livingroom/sensor12/temperature",
          "ie/d/home/livingroom/sensor13/temperature2",
          "ie/m/home/livingroom/sensor13/temperature2",
          "ie/d/building/#",
          "ie/m/building/#"
        ]
      }
    }
  ]
}
```

Note

透過下載路徑篩選條件清單，在目的地之間複製路徑篩選條件。如需詳細資訊，請參閱[下載目的地中的所有路徑篩選條件（主控台）](#)。

大量上傳路徑篩選條件

若要大量上傳路徑篩選條件，請使用 CSV 或文字檔案。當您上傳檔案時 AWS IoT SiteWise，會自動移除確切的重複項目。例如，windfarm/site1/和 windfarm/site1/ 是 AWS IoT SiteWise 擷取的确切重複項目，因為字串完全相同。部分重複項目不會移除，且會產生額外費用。例如，windfarm/#和 windfarm/site1 是重疊的主題，因為 windfarm/site1 已包含 windfarm/#。

Note

避免重複，以避免產生額外費用。上傳的檔案必須是 .csv 或 .txt 格式。它不能包含任何標頭，且應由單一資料欄組成。在 欄中，列出您的路徑篩選條件，每個篩選條件位於單獨的一行。檔案中不應包含任何其他資訊。

檔案上傳需求

這些是額外的路徑篩選條件需求。

- 您可以上傳一個 .csv 或 .txt 檔案。不支援其他檔案格式。
- CSV (.csv) 檔案不能有標頭，而且應該只包含一個資料欄。
- 您可以在每行上有一個路徑篩選條件。
- 上傳的檔案不能為空白。
- 使用 # 做為萬用字元時，必須是主題篩選條件中的最後一個字元。例如，topic/# 或 作為特定主題層級的獨立字元。不過，請注意，#也可以做為主題層級名稱中的一般字元，例如 factory/machine#1/topic。如需詳細資訊，請參閱[路徑篩選條件名稱中的特殊字元](#)
- 您也可以使用 + 字元。例如，使用 factory/+/temp取得工廠的所有溫度，而非factory/machine3/temp個別的 factory/machine2/temp和。

管理 AWS IoT SiteWise Edge 目的地

新增目的地之後，您可以執行各種操作來管理它們，例如編輯目的地組態、刪除目的地和管理路徑篩選條件。

編輯目的地

選取資料表中目的地旁的選項按鈕，然後選擇編輯按鈕來編輯目的地。

Console

使用 AWS IoT SiteWise 主控台編輯目的地

1. 開啟 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選取適當的閘道。
4. 在目的地區段中，選擇您要編輯的目的地，然後選擇編輯。
5. 修改目的地，然後選擇儲存。

AWS CLI

使用 編輯目的地 AWS CLI

- 您可以透過修改 JSON 功能組態資訊來編輯目的地。

```
aws iotsitewise update-gateway-capability-configuration \
--gateway-id your-gateway-id \
--capability-namespace "iotsitewise:publisher:3" \
--capability-configuration '{
  "sources": [
    {
      "type": "MQTT"
    }
  ],
  "destinations": [
    {
      "id": "your-existing-destination-id",
      "type": "SITEWISE_REALTIME",
      "name": "your-updated-destination-name",
      "config": {
```

```

        "publishingOrder": "TIME_ORDER",
        "enableCompression": true,
        "dropPolicy": {
            "cutoffAge": "10d",
            "exportPolicy": {
                "retentionPeriod": "10d",
                "rotationPeriod": "6h",
                "exportSizeLimitGB": 10
            }
        },
        "maxBatchWaitTime": "15s"
    },
    "filters": [
        {
            ...
        }
    ]
}
]
}'

```

Note

您無法更新目的地type或 capability-namespace。例如，您無法從 類型切換SITEWISE_REALTIME到 SITEWISE_BUFFERED。您可以為每個啟用 MQTT 的閘道有一個即時目的地。

刪除目的地

如果您不再需要目的地，您可以從 SiteWise Edge 閘道將其刪除。

Console

使用 AWS IoT SiteWise 主控台刪除目的地

1. 開啟 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選取適當的閘道。
4. 在目的地區段中，選擇您要刪除的目的地，然後選擇刪除。確認畫面隨即出現。

- 若要確認選擇刪除目的地，請在確認方塊中輸入「刪除」。

AWS CLI

使用 刪除目的地 AWS CLI

- 指定閘道 ID 並修改功能組態以移除您要刪除的目的地，以刪除閘道功能組態。

```
aws iotsitewise update-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
  --capability-configuration '{
    "sources": [
      {
        "type": "MQTT"
      }
    ],
    "destinations": []
  }'
```

Note

目的地陣列可以是空的 ([])，但目的地物件本身必須包含在功能組態中。

下載目的地中的所有路徑篩選條件（主控台）

在 AWS IoT SiteWise 主控台中下載包含所有路徑篩選條件的 CSV 檔案。您可以使用下載的路徑篩選條件清單，在閘道目的地之間輕鬆共用路徑篩選條件清單。

使用 AWS IoT SiteWise 主控台下載所有路徑篩選條件的 CSV 檔案

- 開啟 [AWS IoT SiteWise 主控台](#)。
- 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
- 選取包含路徑篩選條件的閘道。
- 選擇新增目的地或編輯目的地。
- 導覽至路徑篩選條件區段，然後選擇下載 CSV。

 Note

CSV 檔案包含特定目的地中的所有路徑篩選條件，無論您從路徑篩選條件清單中選擇了哪些路徑篩選條件。

編輯路徑篩選條件

您可以編輯個別路徑篩選條件，以精簡目的地接收到的資料。

Console

使用 AWS IoT SiteWise 主控台，您可以編輯每個個別文字方塊中的每個個別路徑篩選條件。

使用 AWS IoT SiteWise 主控台編輯路徑篩選條件

1. 開啟 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選取包含路徑篩選條件的閘道。
4. 選取適當的目的地。
5. 選擇編輯。
6. 針對包含您要編輯之路徑篩選條件的資料列，選擇文字方塊。
7. 更新路徑篩選條件的文字，確保已選取編輯的路徑篩選條件的核取方塊。
8. 選擇儲存。

AWS CLI

若要使用 編輯目的地的路徑篩選條件 AWS CLI，請先擷取目前的組態、修改組態，然後使用 `update-gateway-capability-configuration` 命令進行更新。

使用 編輯路徑篩選條件 AWS CLI

1. 擷取目前的功能組態：

```
aws iotsitewise describe-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
```

```
--query "capabilityConfiguration"
```

2. 編輯 JSON 以視需要修改路徑篩選條件。
3. 使用修改過的路徑篩選條件更新功能組態：

```
aws iotsitewise update-gateway-capability-configuration \  
  --gateway-id your-gateway-id \  
  --capability-namespace "iotsitewise:publisher:3" \  
  --capability-configuration json-containing-your-updated-path-filters
```

刪除路徑篩選條件

您可以刪除目的地的路徑篩選條件，以控制從 MQTT 來源和資料處理管道接收的資料。

Console

使用 AWS IoT SiteWise 主控台刪除路徑篩選條件

1. 開啟 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽中，選擇 Edge 區段中的 Edge 閘道。
3. 選取包含路徑篩選條件的閘道。
4. 選取適當的目的地。
5. 選擇編輯。
6. 在編輯目的地畫面的路徑篩選條件區段中，選取要刪除的一或多個路徑篩選條件。
7. 選擇 刪除。隨即出現刪除確認訊息。如果想要繼續刪除路徑篩選條件，請在確認畫面上選擇刪除。

AWS CLI

使用 刪除目的地 AWS CLI

- 從功能組態中移除路徑篩選條件，以將其刪除。

```
aws iotsitewise update-gateway-capability-configuration \  
  --gateway-id your-gateway-id \  
  --capability-namespace "iotsitewise:publisher:3" \  
  --capability-configuration '{  
    "sources": [  
      {  
        "pathFilter": "your-path-filter"  
      }  
    ]  
  }'
```

```

    {
      "type": "MQTT"
    }
  ],
  "destinations": [
    {
      "id": "your-destination-id",
      "type": "SITEWISE_REALTIME",
      "name": "your-destination-name",
      "config": {
        ...
      },
      "filters": [
        {
          "type": "PATH",
          "config": {
            "paths": [
              "/path1",
              "/path2",
              "/delete-a-path-to-remove-it"
            ]
          }
        }
      ]
    }
  ]
}

```

Note

篩選條件陣列可以是空的 ([])，但篩選條件物件本身必須包含在功能組態中。

管理 SiteWise Edge 閘道

您可以使用 AWS IoT SiteWise 主控台和 API 操作來管理 AWS IoT SiteWise Edge 閘道。您也可以使用 [AWS OpsHubAWS IoT SiteWise for Windows](#) 應用程式，從本機裝置管理 SiteWise Edge 閘道的某些層面。

強烈建議您 AWS OpsHub 針對 AWS IoT SiteWise 應用程式使用 來監控本機裝置上的磁碟用量。您也可以監控 Gateway.AvailableDiskSpace 和 Gateway.UsedPercentageDiskSpace Amazon

CloudWatch 指標，並建立警示，以便在磁碟空間不足時收到通知。如需 Amazon CloudWatch 警示的詳細資訊，請參閱[根據靜態閾值建立 CloudWatch 警示](#)。

請確定您的裝置有足夠的空間容納即將來臨的資料。當您即將用盡本機裝置上的空間時，服務會自動刪除具有最舊時間戳記的少量資料，為即將來臨的資料騰出空間。

若要檢查服務是否刪除您的資料，請執行下列動作：

1. 登入 AWS OpsHub AWS IoT SiteWise 應用程式適用的。
2. 選擇設定。
3. 針對日誌，指定時間範圍，然後選擇下載。
4. 解壓縮日誌檔案。
5. 如果日誌檔案包含下列訊息，服務會刪除您的資料：已刪除資料的位元組#，以防止 SiteWise Edge 閘道儲存體空間不足。

使用 AWS IoT SiteWise 主控台管理您的 SiteWise Edge 閘道

您可以使用 AWS IoT SiteWise 主控台來設定、更新和監控 AWS 帳戶中的所有 SiteWise Edge 閘道。

您可以透過導覽至[AWS IoT SiteWise 主控台](#)中的 Edge Gateways 頁面來檢視 SiteWise Edge 閘道。若要存取特定閘道的 Edge 閘道詳細資訊頁面，請選擇 Edge 閘道的名稱。

從 Edge 閘道詳細資訊頁面的概觀索引標籤中，您可以執行下列動作：

- 在資料來源區段中，更新資料來源組態並設定其他資料來源
- 選擇開啟 CloudWatch 指標以檢視 CloudWatch 指標主控台中每個資料來源擷取的資料點數量
- 在 Edge 功能區段中，按一下編輯，將資料套件新增至 SiteWise Edge 閘道
- 在閘道組態區段中，檢視 SiteWise Edge 閘道的連線狀態
- 在發佈者組態區段中，檢視發佈者元件的 SiteWise Edge AWS IoT SiteWise 閘道同步狀態和組態

從 Edge 閘道詳細資訊頁面的更新索引標籤中，您可以查看部署到 Edge 閘道的目前元件和套件版本。當新版本可用時，您也可以在此部署新版本。

使用 AWS OpsHub for 管理 SiteWise Edge 閘道 AWS IoT SiteWise

您可以將 AWS OpsHub 用於 AWS IoT SiteWise 應用程式，以管理和監控自我託管的 SiteWise Edge 閘道。此應用程式提供下列監控和管理選項：

- 在概觀下，您可以執行下列動作：
 - 檢視 SiteWise Edge 閘道詳細資訊，協助您深入了解 SiteWise Edge 閘道裝置資料、識別問題並改善 SiteWise Edge 閘道的效能。
 - 檢視 SiteWise Monitor 入口網站，以監控來自本機伺服器 and 設備邊緣的資料。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[什麼是 AWS IoT SiteWise Monitor](#)。
- 在運作狀態下，有一個儀表板可顯示來自 SiteWise Edge 閘道的資料。程序工程師等網域專家可以使用儀表板來查看 SiteWise Edge 閘道行為的概觀。
- 在資產下，檢視部署至本機裝置的資產，以及為資產屬性收集或計算的最後一個值。
- 在設定下，您可以執行下列動作：
 - 如果已安裝 Data Processing Pack，請檢視 SiteWise Edge 閘道組態資訊，AWS 並將資源與雲端同步。
 - 下載您可以使用其他工具來存取 SiteWise Edge 閘道的身分驗證檔案。
 - 下載可用於對 SiteWise Edge 閘道進行故障診斷的日誌。
 - 檢視部署到 SiteWise Edge 閘道的 AWS IoT SiteWise 元件。

Important

使用 AWS OpsHub 時需要下列項目 AWS IoT SiteWise：

- 您的本機裝置和 AWS OpsHub AWS IoT SiteWise 應用程式的 必須連接到相同的網路。
- 必須啟用資料處理套件。

使用 管理 SiteWise Edge 閘道 AWS OpsHub

1. 下載並安裝 [AWS OpsHubAWS IoT SiteWise for Windows](#) 應用程式的。
2. 開啟 應用程式。
3. 如果您沒有為閘道設定本機登入資料，請依照下的步驟[使用本機作業系統登入資料存取 SiteWise Edge 閘道](#)進行設定。
4. 您可以使用 Linux 或輕量型目錄存取通訊協定 (LDAP) 登入資料登入 SiteWise Edge 閘道。若要登入 SiteWise Edge 閘道，請執行下列其中一項操作：

Linux

1. 針對主機名稱或 IP 地址，輸入本機裝置的主機名稱或 IP 地址。

2. 針對身分驗證，選擇 Linux。
3. 針對使用者名稱，輸入 Linux 作業系統的使用者名稱。
4. 針對密碼，輸入 Linux 作業系統的密碼。
5. 選擇 Sign In (登入)。

LDAP

1. 針對主機名稱或 IP 地址，輸入本機裝置的主機名稱或 IP 地址。
2. 針對身分驗證，選擇 LDAP。
3. 針對使用者名稱，輸入 LDAP 的使用者名稱。
4. 針對密碼，輸入 LDAP 的密碼。
5. 選擇 Sign In (登入)。

使用本機作業系統登入資料存取 SiteWise Edge 閘道

除了輕量型目錄存取通訊協定 (LDAP) 之外，您還可以使用 Linux 或 Windows 登入資料來存取自我託管的 SiteWise Edge 閘道。

Important

若要使用 Linux 登入資料存取 SiteWise Edge 閘道，您必須啟用 SiteWise Edge 閘道的資料處理套件。

使用 Linux 作業系統登入資料存取 SiteWise Edge 閘道

下列步驟假設您使用裝置搭配 Ubuntu。如果您使用不同的 Linux 發行版本，請參閱裝置的相關文件。

建立 Linux 使用者集區

1. 若要建立管理員群組，請執行下列命令。

```
sudo groupadd --system SWE_ADMIN_GROUP
```

SWE_ADMIN_GROUP 群組中的使用者可以允許 SiteWise Edge 閘道的管理員存取。

2. 若要建立使用者群組，請執行下列命令。

```
sudo groupadd --system SWE_USER_GROUP
```

SWE_USER_GROUP 群組中的使用者可以允許 SiteWise Edge 閘道的唯讀存取。

- 若要將使用者新增至管理員群組，請執行下列命令。以您要新增的使用者名稱和密碼取代使用者名稱和密碼。

```
sudo useradd -p $(openssl passwd -1 password) user-name
```

- 若要將使用者新增至 SWE_ADMIN_GROUP 或 SWE_USER_GROUP，請以您在上一個步驟中新增的使用者名稱取代 *user-name*。

```
sudo usermod -a -G SWE_ADMIN_GROUP user-name
```

您現在可以使用使用者名稱和密碼來登入 AWS OpsHub AWS IoT SiteWise 應用程式上的 SiteWise Edge 閘道。

使用 Windows 登入資料存取 SiteWise Edge 閘道

下列步驟假設您使用裝置搭配 Windows。

Important

安全性是 AWS 與您之間共同責任。建立至少 12 個字元的強式密碼政策，以及大寫、小寫、數字和符號的組合。此外，設定 Windows 防火牆規則以允許連接埠 443 上的傳入流量，並封鎖所有其他連接埠上的傳入流量。

建立 Windows Server 使用者集區

- 以系統管理員身分執行 PowerShell。
 - 在您要安裝 SiteWise Edge Gateway 的 Windows 伺服器上，以管理員身分登入。
 - 在 Windows 搜尋列中輸入 PowerShell。
 - 在搜尋結果中，用滑鼠右鍵按一下 Windows PowerShell 應用程式。選擇以管理員身分執行。
- 若要建立管理員群組，請執行下列命令。

```
net localgroup SWE_ADMIN_GROUP /add
```

您必須是 SWE_ADMIN_GROUP 群組中的使用者，才能允許 SiteWise Edge 閘道的管理員存取。

- 若要建立使用者群組，請執行下列命令。

```
net localgroup SWE_USER_GROUP /add
```

您必須是 SWE_USER_GROUP 群組中的使用者，才能允許 SiteWise Edge 閘道的唯讀存取。

- 若要新增使用者，請執行下列命令。以您要建立的 ##### 取代使用者名稱和密碼。

```
net user user-name password /add
```

- 若要將使用者新增至管理員群組，請執行下列命令。將 *user-name* 取代為您要新增的使用者名稱。

```
net localgroup SWE_ADMIN_GROUP user-name /add
```

您現在可以使用使用者名稱和密碼來登入 AWS OpsHub AWS IoT SiteWise 應用程式 上的 SiteWise Edge 閘道。

管理 SiteWise Edge 閘道憑證

您可以在 SiteWise Edge 閘道裝置上使用 SiteWise Monitor 和第三方應用程式，例如 Grafana。這些應用程式需要與服務的 TLS 連線。SiteWise Edge 閘道目前使用自我簽署憑證。如果您使用瀏覽器開啟應用程式，例如 SiteWise Monitor 入口網站，您可能會收到不受信任憑證的警告。

以下說明如何從 AWS IoT SiteWise 應用程式 AWS OpsHub 下載信任的憑證。

- 登入應用程式。
- 選擇設定。
- 針對身分驗證，選擇下載憑證。

下列假設您使用 Google Chrome 或 FireFox。如果您使用不同的瀏覽器，請參閱瀏覽器的相關文件。若要將您在上一個步驟中下載的憑證新增至瀏覽器，請執行下列其中一項操作：

- 如果您使用 Google Chrome，請遵循 Google Chrome Enterprise 說明文件中[的設定憑證](#)。

- 如果您使用 Firefox，請遵循 Oracle 文件中的[將憑證載入 Mozilla 或 Firefox 瀏覽器](#)。

變更 SiteWise Edge 閘道元件套件的版本

您可以使用 AWS IoT SiteWise 主控台來變更 SiteWise Edge 閘道上的元件套件版本。

變更 SiteWise Edge 閘道元件套件的版本

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中，選擇 Gateways (閘道)。
3. 選取您要變更套件版本的 SiteWise Edge 閘道。
4. 在閘道組態下，選擇檢視軟體版本。
5. 在編輯軟體版本頁面上，針對您要更新版本的套件，選取您要部署的版本，然後選擇部署。
6. 選擇完成。

列出 SiteWise Edge 閘道

Console

列出 SiteWise Edge 閘道

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 檢視所有 SiteWise Edge 閘道的清單。

AWS CLI

若要使用 列出您的閘道 AWS CLI，請遵循下列步驟：

- 使用 list-gateways 命令來檢視所有閘道：

```
aws iotsitewise list-gateways
```

此命令會傳回閘道的清單，其中包含其 IDs、名稱和其他資訊。

您也可以指定分頁參數：

```
aws iotsitewise list-gateways --max-results 10 --next-token your-token
```

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [list-gateways](#)。

描述 SiteWise Edge 閘道

Console

檢視閘道詳細資訊

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇您要檢視詳細資訊的閘道名稱。
4. 在 Edge 閘道詳細資訊頁面上檢視閘道詳細資訊。

AWS CLI

若要使用 取得特定閘道的詳細資訊 AWS CLI，請遵循下列步驟：

- 使用 describe-gateway 命令搭配閘道 ID：

```
aws iotsitewise describe-gateway --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

此命令會傳回閘道的詳細資訊。

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [describe-gateway](#)。

建立 SiteWise Edge 閘道

Console

建立 SiteWise Edge 閘道

1. 導覽至 [AWS IoT SiteWise 主控台](#)。

2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇 Create gateway (建立閘道)。
4. 輸入閘道的名稱。
5. 選取閘道的 Greengrass 群組。
6. 或者，將標籤新增至閘道。
7. 選擇建立。

AWS CLI

若要使用 建立新的 IoT SiteWise 閘道 AWS CLI，請遵循下列步驟：

- 使用 create-gateway 命令建立新的閘道：

```
aws iotsitewise create-gateway \
  --gateway-name "NewSiteWiseGateway" \
  --gateway-platform '{
    "greengrass": {
      "groupArn": "arn:aws:greengrass:us-east-1:123456789012:group/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
    }
  }' \
  --tags '{
    "Environment": "Production",
    "Location": "Factory1"
  }'
```

此命令會傳回新閘道的 ID 和 ARN：

```
{
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayArn": "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
}
```

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [create-gateway](#)。

更新 SiteWise Edge 閘道

Console

更新 SiteWise Edge 閘道

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取您要更新的閘道。
4. 選擇編輯。
5. 視需要更新閘道名稱或其他設定。
6. 選擇儲存。

AWS CLI

若要使用 更新現有的閘道 AWS CLI，請遵循下列步驟：

- 使用 update-gateway 命令更新閘道的名稱：

```
aws iotsitewise update-gateway \  
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE \  
  --gateway-name "UpdatedGatewayName"
```

此命令成功後就不會產生輸出。

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [update-gateway](#)。

更新閘道功能組態

Console

更新閘道功能組態

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇您要更新的閘道名稱。

4. 在資料來源區段中，選擇編輯。
5. 視需要更新資料來源組態。
6. 選擇儲存。

AWS CLI

若要使用 更新閘道的功能組態 AWS CLI，請遵循下列步驟：

- 使用 `update-gateway-capability-configuration` 命令來更新功能組態：

```
aws iotsitewise update-gateway-capability-configuration \
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE \
  --capability-namespace "iotsitewise:opcuacollector:1" \
  --capability-configuration '{
    "sources": [
      {
        "name": "OPC-UA Server",
        "endpoint": {
          "certificateTrust": {
            "type": "TrustAny"
          },
          "endpointUri": "opc.tcp://10.0.0.1:4840",
          "securityPolicy": "NONE",
          "messageSecurityMode": "NONE",
          "identityProvider": {
            "type": "Anonymous"
          }
        },
        "measurementDataStreamPrefix": ""
      }
    ]
  }'
```

此命令會傳回功能命名空間和同步狀態：

```
{
  "capabilityNamespace": "iotsitewise:opcuacollector:1",
  "capabilitySyncStatus": "CONFIGURING"
}
```

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [update-gateway-capability-configuration](#)。

標記閘道資源

Console

標記閘道資源

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇您要標記的閘道名稱。
4. 選擇 Tags (標籤) 索引標籤。
5. 選擇管理標籤。
6. 選擇新增標籤，然後為每個標籤輸入索引鍵和值。
7. 選擇儲存。

AWS CLI

若要使用 將標籤新增至閘道 AWS CLI，請遵循下列步驟：

- 使用 tag-resource 命令將標籤新增至閘道：

```
aws iotsitewise tag-resource \
  --resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE" \
  --tags '{
    "Department": "Operations",
    "Project": "FactoryAutomation"
  }'
```

此命令成功後就不會產生輸出。

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [tag-resource](#)。

列出閘道的標籤

Console

列出閘道的標籤

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇您要檢視標籤的閘道名稱。
4. 選擇 Tags (標籤) 索引標籤。
5. 檢視與閘道相關聯的標籤清單。

AWS CLI

若要使用 列出與閘道相關聯的標籤 AWS CLI，請遵循下列步驟：

- 使用 `list-tags-for-resource` 命令列出閘道的標籤：

```
aws iotsitewise list-tags-for-resource \
    --resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
```

此命令會傳回與閘道相關聯的標籤：

```
{
  "tags": {
    "Environment": "Production",
    "Location": "Factory1",
    "Department": "Operations",
    "Project": "FactoryAutomation"
  }
}
```

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [list-tags-for-resource](#)。

從閘道移除標籤

Console

從閘道移除標籤

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇您要從中移除標籤的閘道名稱。
4. 選擇 Tags (標籤) 索引標籤。
5. 選擇管理標籤。
6. 選擇您要移除的每個標籤旁邊的移除圖示 (X)。
7. 選擇儲存。

AWS CLI

若要使用 從閘道移除標籤 AWS CLI，請遵循下列步驟：

- 使用 `untag-resource` 命令從閘道移除標籤：

```
aws iotsitewise untag-resource \
  --resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE" \
  --tag-keys '["Project", "Department"]'
```

此命令成功後就不會產生輸出。

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [untag-resource](#)。

更新 AWS IoT SiteWise 元件的版本

更新 AWS IoT Greengrass 核心裝置上的 AWS IoT SiteWise 閘道元件，以確保您存取最新的功能、效能改善和安全性修補程式。

若要更新 上的 AWS IoT SiteWise 元件 AWS IoT Greengrass

1. 導覽至 [AWS IoT SiteWise 主控台](#)。

2. 在左側導覽窗格中，選擇 Edge 閘道。
3. 選取要編輯的閘道，然後選擇編輯。
4. 在 Edge 功能中，在軟體版本下，選擇可用的軟體更新。隨即顯示編輯軟體版本頁面。
5. 選擇元件版本。

 Note

建議您選取可用的最新版本。讓閘道元件保持up-to-date，可協助您維持工業資料收集和處理的最佳功能。

6. 選擇部署。這會啟動 AWS IoT Greengrass V2 部署，以更新閘道上的 AWS IoT SiteWise 元件。

刪除 SiteWise Edge 閘道

Console

刪除 SiteWise Edge 閘道

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇您要刪除的閘道。
4. 選擇 刪除。
5. 若要確認要刪除閘道，請輸入「刪除」，然後在出現的視窗中選擇刪除。

AWS CLI

若要使用 刪除閘道 AWS CLI，請遵循下列步驟：

1. 列出您的閘道，以識別您要刪除之閘道的閘道 ID。

```
aws iotsitewise list-gateways
```

此命令會傳回閘道的清單，其中包含其 IDs、名稱和其他資訊：

```
{
  "gatewaySummaries": [
    {
```

```
{
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayName": "ExampleCorpGateway",
  "gatewayCapabilitySummaries": [
    {
      "capabilityNamespace": "iotsitewise:opcuacollector:1",
      "capabilitySyncStatus": "IN_SYNC"
    }
  ],
  "creationDate": 1588369971.457,
  "lastUpdateDate": 1588369971.457
}
```

2. 透過指定閘道的 ID 來刪除閘道：

```
aws iotsitewise delete-gateway --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

此命令成功後就不會產生輸出。

 Note

當您刪除閘道時，閘道的某些檔案會保留在閘道的檔案系統中。

3. 若要確認閘道已刪除，您可以再次列出閘道：

```
aws iotsitewise list-gateways
```

已刪除的閘道不應再出現在清單中。

如需詳細資訊，請參閱《AWS CLI 命令參考》中的 [delete-gateway](#)。

備份和還原 SiteWise Edge 閘道

本主題涵蓋如何還原 SiteWise Edge 閘道和備份指標資料。如果您在相同機器上遇到故障的 SiteWise Edge 閘道問題，且需要對問題進行故障診斷，請閱讀 AWS IoT SiteWise 文件 [對 SiteWise Edge 閘道問題進行故障診斷](#)。

 Note

本主題涵蓋的指引適用於安裝在 2.1.0 AWS IoT Greengrass V2 版或更新版本上的 SiteWise Edge 閘道。

指標資料的每日備份

如果您想要在新機器上傳輸或還原資料，建立備份非常重要。備份資料可大幅降低傳輸或還原過程中遺失操作資料的風險。

本節適用於使用資料處理套件的閘道。如需資料處理套件的詳細資訊，請參閱 [設定用於 SiteWise Edge 上資料處理的資產模型](#)。

influxdb 資料夾路徑如下：

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/influxdb
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeProcessor\influxdb
```

建議您將整個資料夾備份到其中的所有內容。

建議您定期將指標資料從 1.0 SiteWise Edge 備份至外部硬碟或 AWS 雲端。

還原 SiteWise Edge 閘道

嘗試還原 SiteWise Edge 閘道之前，請確定所有連線至閘道的邊緣裝置都已停止或中斷連線。

使用下列程序還原 SiteWise Edge 閘道：

1. 使用建立 SiteWise Edge 閘道時下載的安裝指令碼，在新機器上還原 SiteWise Edge 閘道。閱讀 [在本機裝置上安裝 SiteWise Edge 閘道軟體](#)，以設定 SiteWise Edge 閘道。

如果您遺失或找不到安裝指令碼，請聯絡 [AWS 客戶支援](#)。

2. 安裝 SiteWise Edge 閘道後，請登入 [AWS IoT Greengrass 主控台](#)。

- 若要重新部署元件，請導覽至管理，然後在AWS IoT Greengrass 裝置下選取核心裝置。
- 在AWS IoT Greengrass 核心裝置資料表中，選取對應至 SiteWise Edge 閘道的核心裝置。
- 在裝置頁面上，開啟部署索引標籤並選取您的部署 ID，這會開啟包含您所選 ID 的部署頁面。
- 進入部署頁面後，按下右上角的動作按鈕，然後選取修訂選項。以啟動新的部署。設定部署。如果您想要保持部署不變，請跳至檢閱和部署。
- 等待部署狀態變成 Completed。

 Note

SiteWise Edge 上的所有元件還需要幾分鐘的時間才能完全設定和執行。

還原 AWS IoT SiteWise 資料

使用下列程序在新機器上還原資料。

- 將 influxdb 資料夾複製到新機器。
- 在終端機中執行下列命令，以停止 SiteWise EdgeProcessor 元件：

Linux

```
sudo /greengrass/v2/bin/greengrass-cli component stop -n  
aws.iot.SiteWiseEdgeProcessor
```

Windows

```
C:\greengrass\v2\bin\greengrass-cli component stop -n  
aws.iot.SiteWiseEdgeProcesso
```

- 找到您備份資料的路徑，並執行下列命令：

Linux

```
sudo yes | sudo cp -rf <influxdb_backup_path> /greengrass/v2/work/  
aws.iot.SiteWiseEdgeProcessor/influxdb
```

PowerShell

```
Copy-Item -Recurse -Force <influxdb_backup_path>\* C:\greengrass  
\v2\work\aws.iot.SiteWiseEdgeProcessor\
```

Windows

```
robocopy <influxdb_backup_path> C:\greengrass\v2\work  
\aws.iot.SiteWiseEdgeProcessor\ /E
```

4. 重新啟動 SiteWiseEdgeProcessor 元件：

Linux

```
sudo /greengrass/v2/bin/greengrass-cli component restart -n  
aws.iot.SiteWiseEdgeProcessor
```

Windows

```
C:\greengrass\v2\bin\greengrass-cli component restart -n  
aws.iot.SiteWiseEdgeProcessor
```

驗證成功的備份和還原

使用此程序驗證備份資料和 SiteWise Edge 閘道還原。

Note

此程序需要您已為 安裝 AWS OpsHub AWS IoT SiteWise。如需詳細資訊，請參閱[使用 AWS OpsHub 管理 SiteWise Edge 閘道 AWS IoT SiteWise](#)。

1. 適用於 的 Open AWS OpsHub AWS IoT SiteWise。
2. 在 SiteWise Edge Gateway 設定頁面上，檢查元件資料表中列出的每個元件的狀態。確認狀態顏色為綠色，且讀取結果顯示正在執行。
3. 在入口網站儀表板上驗證您過去的資料，以檢查過去的資料和新資料是否正確設定。過去和新的資料之間將會有停機時間。除了 之外，您應該查看未收集任何資料點的持續時間。

如果您在備份或還原 SiteWise Edge 閘道時遇到問題，請參閱下列故障診斷主題 [故障診斷 AWS IoT SiteWise Edge 閘道](#)。

舊版閘道 (AWS IoT Greengrass Version 1)

Note

AWS IoT Greengrass V1 只有在 2021 年 7 月 29 日之前開始使用此功能時，才能使用在 上執行的 SiteWise Edge 閘道。如需使用 執行 AWS IoT SiteWise 閘道的詳細資訊 AWS IoT Greengrass V2，請參閱 [使用 自我託管 AWS IoT SiteWise Edge 閘道 AWS IoT Greengrass V2](#)。

SiteWise Edge 閘道現在專門在 上執行 AWS IoT Greengrass V2，為您的工業 IoT 應用程式提供增強的功能並改善效能。此最新版本 AWS IoT Greengrass V2 代表架構演進，以現代元件型架構為基礎，可啟用模組化軟體部署。它透過統一的安裝程式簡化安裝，同時為開發人員提供部署自訂元件和進行本機測試的更大靈活性。元件型模型允許更有效率的資源管理，並透過元件配方引入簡化的組態方法。此設計有助於元件之間的更佳相依性處理、支援持續部署實務，並提供增強型 CLI 功能以進行本機開發。此外，會透過 AWS IoT Greengrass V2 集中管理組態，AWS IoT Core 並提供改善的記錄和監控功能，所有功能都受到更精細的安全許可模型保護。

如需使用 開始使用 SiteWise Edge 閘道的詳細資訊 AWS IoT Greengrass V2，[AWS IoT SiteWise Edge 自我託管閘道需求](#)請參閱。這些資源提供step-by-step說明。IoT

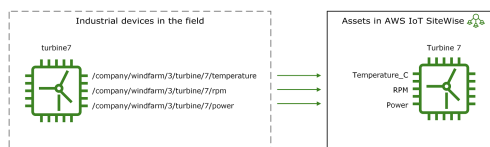
Note

隨著 AWS 持續創新並改善其 IoT 服務，建議您隨時掌握最新的功能和增強功能。定期檢查 AWS IoT SiteWise 和 AWS IoT Greengrass 文件，以取得可進一步最佳化工業 IoT 解決方案的新功能。

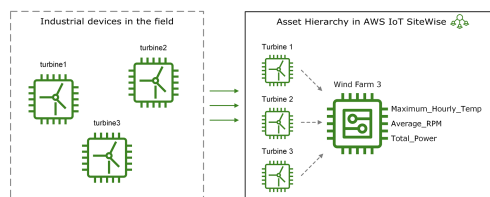
模型工業資產

您可以使用 AWS IoT SiteWise 資產建立工業操作的虛擬表示法。資產代表將一或多個資料串流上傳至 AWS 雲端的裝置、設備片段或程序。例如，資產裝置可以是風力渦輪機，將空氣溫度、螺旋槳旋轉速度和功率輸出時間序列測量傳送至 AWS IoT SiteWise 中的資產屬性。

每個資料串流都會對應到唯一的屬性別名。例如，別名 `/company/windfarm/3/turbine/7/temperature` 僅辨識風力發電廠 #3 中渦輪 #7 的溫度資料流。您可以設定 AWS IoT SiteWise 資產，以使用數學表達式轉換傳入的測量資料，例如將溫度資料從攝氏轉換為華氏。



資產也可以代表裝置的邏輯分組，例如整個風力發電場。您可以將資產與其他資產建立關聯，以建立代表複雜工業操作的資產階層。資產可以存取其相關聯子資產中的資料。如此一來，您可以使用 AWS IoT SiteWise 表達式來計算彙總指標，例如風力發電廠的淨功率輸出。



您必須從資產模型建立每個資產。資產模型是宣告式結構，可將資產格式標準化。資產模型會跨相同類型的多個資產強制執行一致的資訊，以便您可以在代表裝置群組的資產中處理資料。在上圖中，三個渦輪機都使用相同的資產模型，因為所有渦輪機都共用一組通用的屬性。

您也可以建立元件模型。元件模型是一種特殊類型的資產模型，您可以包含在資產模型或其他元件模型中。您可以使用元件模型來定義可在多個資產模型之間共用的通用可重複使用子組件，例如感應器、馬達等。

定義資產模型後，您可以建立工業操作。若要建立資產，請選取 ACTIVE 資產模型以從該模型建立資產。然後，您可以填入資產特定資訊，例如資料串流別名和屬性。在上圖中，您可以從一個資產模型建立三個渦輪資產，然後將資料串流別名，如 `/company/windfarm/3/turbine/7/temperature`，和每個渦輪建立關聯。

您也可以更新和刪除現有的資產、資產模型和元件模型。更新資產模型時，以該資產模型為基礎的每個資產都會反映您對基礎模型所做的任何變更。當您更新元件模型時，這會根據參考元件模型的每個資產模型，套用至每個資產。

您的資產模型可能非常複雜，例如在建模具有許多子元件的複雜設備時。為了協助讓這類資產模型保持組織和可維護，您可以使用自訂複合模型來將相關屬性分組或重複使用共用元件。如需詳細資訊，請參閱[自訂複合模型（元件）](#)。

主題

- [資產和模型狀態](#)
- [資產模型版本](#)
- [自訂複合模型（元件）](#)
- [設定 AWS IoT SiteWise 物件 IDs](#)
- [建立的資產和編譯模型 AWS IoT SiteWise](#)
- [在中建立資產模型的資產 AWS IoT SiteWise](#)
- [在上搜尋資產 AWS IoT SiteWise 主控台](#)
- [更新屬性值](#)
- [關聯和取消關聯資產](#)
- [更新資產和模型](#)
- [在中刪除資產和模型 AWS IoT SiteWise](#)
- [使用資產和模型進行大量操作](#)

資產和模型狀態

當您建立、更新或刪除資產、資產模型或元件模型時，變更需要一些時間才能傳播。AWS IoT SiteWise 會以非同步方式解決這些操作，並更新每個資源的狀態。每個資產、資產模型和元件模型都有一個狀態欄位，其中包含資源的狀態和任何錯誤訊息，如果適用的話。狀態可以是下列其中一個值：

- ACTIVE – 資源處於作用中狀態。這是您可以查詢資產、資產模型和元件模型並與之互動的唯一狀態。
- CREATING – 正在建立資源。
- UPDATING – 正在更新資源。
- DELETING – 正在刪除資源。
- PROPAGATING – （僅限資產模型和元件模型）變更正在傳播到所有相依資源（從資產模型到資產，或從元件模型到資產模型）。
- FAILED – 資源無法在建立或更新操作期間驗證，可能是因為表達式中有循環參考。您可以刪除處於 FAILED 狀態的資源。

某些建立、更新和刪除操作會將資產、資產模型或元件模型 AWS IoT SiteWise 置於操作解析ACTIVE時以外的狀態。若要在執行其中一個操作後查詢資源或與之互動，您必須等到狀態變更為ACTIVE。否則，您的請求會失敗。

主題

- [檢查資產的狀態](#)
- [檢查資產或元件模型的狀態](#)

檢查資產的狀態

您可以使用 AWS IoT SiteWise 主控台或 API 來檢查資產的狀態。

主題

- [檢查資產的狀態（主控台）](#)
- [檢查資產的狀態 \(AWS CLI\)](#)

檢查資產的狀態（主控台）

使用下列程序來檢查 AWS IoT SiteWise 主控台中資產的狀態。

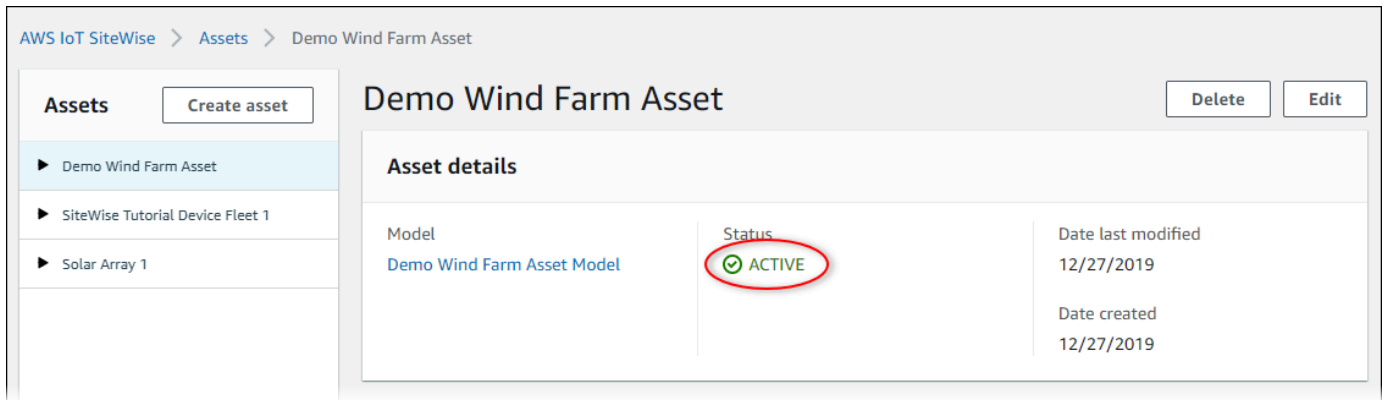
檢查資產的狀態 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇要檢查的資產。

Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 在資產詳細資訊面板中尋找狀態。



檢查資產的狀態 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來檢查資產的狀態。

若要檢查資產狀態，請使用 [DescribeAsset](#) 操作搭配 `assetId` 參數。

檢查資產的狀態 (AWS CLI)

- 輸入以下命令以描述 Pod。以資產的 ID 或外部 ID 取代 *asset-id*。外部 ID 是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

```
aws iotsitewise describe-asset --asset-id asset-id
```

此操作會傳回包含資產詳細資料的回應。回應包含具有下列結構的 `assetStatus` 物件：

```
{
  ...
  "assetStatus": {
    "state": "String",
    "error": {
      "code": "String",
      "message": "String"
    }
  }
}
```

資產的狀態在 JSON 物件的 `assetStatus.state` 中。

檢查資產或元件模型的狀態

您可以使用 AWS IoT SiteWise 主控台或 API 來檢查資產模型或元件模型的狀態。

主題

- [檢查資產模型或元件模型的狀態（主控台）](#)
- [檢查資產模型或元件模型的狀態 \(AWS CLI\)](#)

檢查資產模型或元件模型的狀態（主控台）

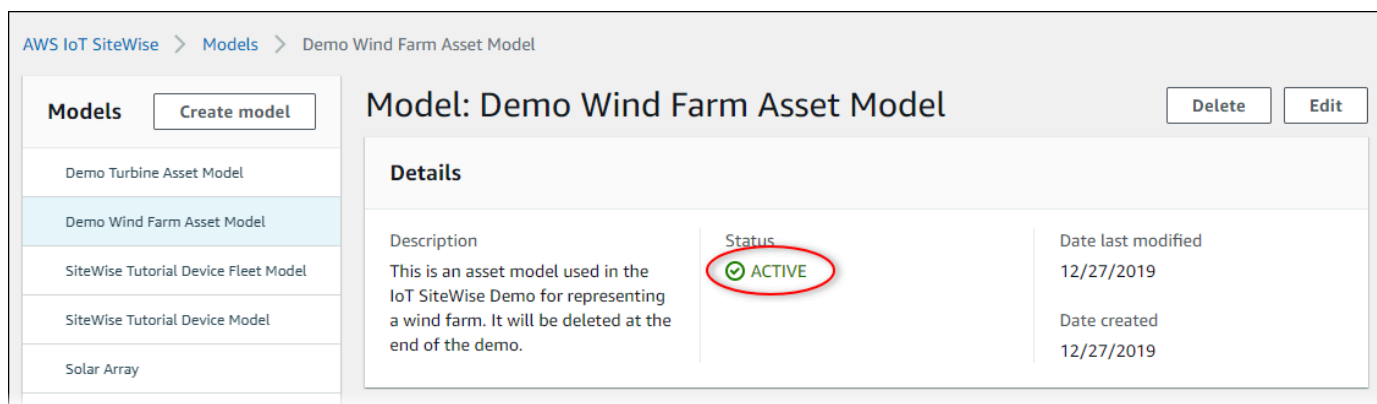
使用下列程序，在 AWS IoT SiteWise 主控台中檢查資產模型或元件模型的狀態。

Tip

資產模型和元件模型都列在導覽窗格中的模型下。所選資產模型或元件模型的詳細資訊面板會指出其類型。

檢查資產模型或元件模型的狀態（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇要檢查的模型。
4. 在詳細資訊面板中尋找狀態。



檢查資產模型或元件模型的狀態 (AWS CLI)

您可以使用 AWS CLI 來檢查資產模型或元件模型的狀態。

若要檢查資產模型或元件模型的狀態，請使用 [DescribeAssetModel](#) 操作搭配 `assetModelId` 參數。

 Tip

將元件模型 AWS CLI 定義為資產模型的類型。因此，您針對這兩種類型的模型使用相同的 [DescribeAssetModel](#) 操作。回應中的 `assetModelType` 欄位指出它是 `ASSET_MODEL` 或 `COMPONENT_MODEL`。

檢查資產模型或元件模型的狀態 (AWS CLI)

- 執行下列命令來描述模型。以資產模型或元件模型的 ID 或外部 ID 取代 *asset-model-id*。外部 ID 是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

操作會傳回包含模型詳細資訊的回應。回應包含具有以下結構的 `assetModelStatus` 物件。

```
{
  ...
  "assetModelStatus": {
    "state": "String",
    "error": {
      "code": "String",
      "message": "String"
    }
  }
}
```

模型的狀態在 JSON 物件 `assetModelStatus.state` 中為。

資產模型版本

AWS IoT SiteWise 支援非同步處理資產模型和元件模型上的建立和更新操作。它也會更新模型的狀態。

AWS IoT SiteWise 在建立中傳播有效的模型變更，並將請求更新至其相依資源（從資產模型到資產，或從元件模型到資產模型）。然後，它會將模型置於 `ACTIVE` 狀態。

如果提供的模型定義無效，會將模型 AWS IoT SiteWise 置於 FAILED 狀態。變更不會傳播到相依資源。相依資源是指模型處於 ACTIVE 狀態時傳播的最後一個模型定義。

根據上述資訊，模型定義有兩種類型的模型版本：

1. 最新版本 – 在建立或更新請求中接受的最新定義。
2. 作用中版本 – 最新定義已成功處理，模型狀態為 ACTIVE。

根據預設，當在資產模型或元件模型上呼叫描述 APIs 時，會傳回模型最新版本的詳細資訊。在某些情況下，需要資產模型或元件模型的作用中版本。請參閱以下範例案例：

- 具有無效定義的更新操作會使您的資產模型處於 FAILED 狀態。您必須擷取資產模型的作用中版本，並建立參考此有效定義的另一個更新請求，以還原變更。
- 上的應用程式 AWS IoT SiteWise 存在，客戶可以在其中檢視資產及其對應的資產模型。當使用者參考對應至特定資產的資產模型定義，且資產模型處於暫時性 UPDATING、PROPAGATING 或 FAILED 狀態時，最新版本會傳回尚未傳播至其資產的資產模型定義。在此情況下，您必須將資產模型的作用中版本擷取給客戶。

主題

- [擷取資產模型或元件模型的作用中版本（主控台）](#)
- [擷取資產模型或元件模型的作用中版本 \(AWS CLI\)](#)

擷取資產模型或元件模型的作用中版本（主控台）

請依照此程序，在 AWS IoT SiteWise 主控台中擷取資產模型或元件模型的作用中版本。

Tip

資產模型和元件模型都列在導覽窗格中的模型下。所選資產模型或元件模型的詳細資訊面板會指出其類型。

擷取資產模型或元件模型的作用中版本（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。

3. 選擇模型以擷取其作用中版本。
 - a. 如果模型處於 ACTIVE 狀態，則您正在檢視其作用中版本。
 - b. 如果模型處於暫時性 UPDATING、PROPAGATING 或 FAILED 狀態，請在詳細資訊面板中的狀態下尋找作用中版本。

擷取資產模型或元件模型的作用中版本 (AWS CLI)

使用 AWS CLI 擷取資產模型或元件模型的作用中版本。

若要擷取資產模型或元件模型的作用中版本，請使用 [DescribeAssetModel](#) 操作搭配 `assetModelVersion` 參數。

Tip

將元件模型 AWS CLI 定義為資產模型的類型。因此，您針對這兩種類型的模型使用相同的 [DescribeAssetModel](#) 操作。回應中的 `assetModelType` 欄位指出它是 `ASSET_MODEL` 或 `COMPONENT_MODEL`。

擷取資產模型或元件模型的作用中版本 (AWS CLI)

- 執行下列命令來描述模型。以資產模型或元件模型的 ID 或外部 ID 取代 *asset-model-id*。外部 ID 是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id --asset-model-version ACTIVE
```

操作會傳回包含模型詳細資訊的回應。回應包含具有下列結構的 `assetModelStatus` 物件。

```
{
  ...
  "assetModelName": "string",
  "assetModelProperties": [ ... ],
  ...,
  "assetModelVersion": "string"
}
```

自訂複合模型（元件）

當您為特別複雜的工業資產建模時，例如具有許多部分的複雜機器，讓資產模型保持井然有序且可維護可能成為一項挑戰。

在這種情況下，如果您使用 主控台，您可以將自訂複合模型或元件新增至現有的資產模型和元件模型。這些可協助您透過分組相關屬性和重複使用子元件定義來保持組織狀態。

自訂複合模型有兩種類型：

- 內嵌自訂複合模型定義一組分組屬性，適用於自訂複合模型所屬的資產模型或元件模型。您可以使用它們來分組相關屬性。它們由名稱、描述和一組資產模型屬性組成。它們不可重複使用。
- Component-model-based 自訂複合模型會參考您想要包含在資產模型或元件模型中的元件模型。您可以使用它們在模型中包含標準子組件。它們由名稱、描述及其參考的元件模型 ID 組成。它們沒有自己的屬性；參考的元件模型會將其關聯的屬性提供給任何建立的資產。

以下各節說明如何在設計中使用自訂複合模型。

主題

- [內嵌自訂複合模型](#)
- [Component-model-based 自訂複合模型](#)
- [使用路徑來參考自訂複合模型屬性](#)

內嵌自訂複合模型

內嵌自訂複合模型提供透過分組相關屬性來組織資產模型的方法。

例如，假設您想要建立機器人資產的模型。機器人包含伺服馬達、電源供應器和電池。每個元件都有自己的屬性，您想要包含在模型中。您可以定義名為 的資產模型 `robot_model`，其屬性如下。

- `robot_model`
 - `servo_status` （整數）
 - `servo_position` （雙重）
 - `powersupply_status` （整數）
 - `powersupply_temperature` （雙重）
 - `battery_status` （整數）

- battery_charge (雙重)

不過，在某些情況下，可能有許多子組件，或者子組件本身可能有許多屬性。在這些情況下，可能會有太多屬性，使其變得難以在模型根目錄的單一平面清單中參考和維護，如上述範例所示。

若要處理這類情況，您可以使用內嵌自訂複合模型來將屬性分組。內嵌自訂複合模型是一種自訂複合模型，可定義自己的屬性。例如，您可以建立機器人的模型，如下所示。

- robot_model
 - servo
 - status (整數)
 - position (雙重)
 - powersupply
 - status (整數)
 - temperature (雙重)
 - battery
 - status (整數)
 - charge (雙重)

在上述範例中，servo、powersupply和 battery是robot_model資產模型中定義的內嵌自訂複合模型名稱。然後，每個複合模型都會定義自己的屬性。

Note

在此情況下，每個自訂複合模型都會定義自己的屬性，因此所有屬性都是資產模型本身的一部分 (robot_model在此案例中為)。這些屬性不會與任何其他資產模型或元件模型共用。例如，如果您建立的其他一些資產模型也有稱為的內嵌自訂複合模型servo，則在 servo內變更 robot_model 不會影響其他資產模型servo的定義。

如果您想要實作這類共用（例如，對於所有資產模型都可以共用的伺服只有一個定義），您可以改為為其建立元件模型，然後建立參考該模型的component-model-based複合模型。如需詳細資訊，請參閱下一節。

如需如何建立內嵌自訂複合模型的詳細資訊，請參閱 [建立自訂複合模型 \(元件\)](#)。

Component-model-based 自訂複合模型

您可以在 中建立元件模型，AWS IoT SiteWise 以定義標準的可重複使用子組件。建立元件模型後，您可以在其他資產模型和元件模型中為其新增參考。您可以透過將component-model-based自訂複合模型新增至您想要參考元件的任何模型來執行此操作。您可以從許多模型或相同模型中多次新增元件的參考。

透過這種方式，您可以避免跨模型複製相同的定義。它還簡化了維護模型，因為您對元件模型所做的任何變更都會反映在使用該模型的所有資產模型中。

例如，假設您的工業安裝有許多類型的設備，而這些設備都使用相同類型的伺服馬達。其中一些在單一設備中有許多 伺服馬達。您可以為每個設備類型建立資產模型，但不想servo每次都重複 的定義。您想要將其建模一次，並在各種資產模型中使用。如果您稍後變更 的定義servo，則會在所有模型和資產中更新。

若要以這種方式建立上一個範例中的機器人模型，您可以將 伺服馬達、電源供應器和電池定義為元件模型，如下所示。

- servo_component_model

- status (整數)
- position (雙重)

- powersupply_component_model

- status (整數)
- temperature (雙重)

- battery__component_model

- status (整數)
- charge (雙重)

然後，您可以定義參考這些元件的資產模型robot_model，例如 。多個資產模型可以參考相同的元件模型。您也可以一個資產模型中多次參考相同的元件模型，例如，如果您的機器人在其中有多個伺服馬達。

- robot_model
 - servo1 (參考 : servo_component_model)
 - servo2 (參考 : servo_component_model)
 - servo3 (參考 : servo_component_model)
 - powersupply (參考 : powersupply_component_model)
 - battery (參考 : battery_component_model)

如需如何建立元件模型的資訊，請參閱 [建立元件模型](#)。

如需如何在其他模型中參考元件模型的資訊，請參閱 [建立自訂複合模型（元件）](#)。

使用路徑來參考自訂複合模型屬性

當您在資產模型、元件模型或自訂複合模型上建立屬性時，您可以從使用其值的其他屬性參考它，例如 [轉換](#) 和 [指標](#)。

AWS IoT SiteWise 提供不同的方法來參考您的屬性。最簡單的方式通常是使用其屬性 ID。不過，如果您要參考的 屬性位於自訂複合模型上，您可能會發現改為透過路徑參考它會更有用。

路徑是路徑區段的排序序列，根據其在資產模型和複合模型中巢狀複合模型之間的位置指定屬性。

取得屬性路徑

您可以從其 [AssetModelProperty](#) 的 path 欄位取得屬性的路徑。

例如，假設您有一個資產模型 robot_model，其中包含具有屬性 servo 的自訂複合模型 position。如果您在 上呼叫 [DescribeAssetModelCompositeModel](#) servo，則 position 屬性會列出如下所示 path 的欄位：

```
"path": [  
  {  
    "id": "asset model ID",  
    "name": "robot_model"  
  },  
  {  
    "id": "composite model ID",  
    "name": "servo"  
  },  
  {
```

```

    "id": "property ID",
    "name": "position"
  }
]

```

使用屬性路徑

當您定義參考其他屬性的屬性時，可以使用屬性路徑，例如轉換或指標。

屬性使用變數來參考另一個屬性。如需使用變數的詳細資訊，請參閱 [在公式表達式中使用變數](#)。

當您定義變數以參考屬性時，您可以使用屬性的 ID 或其路徑。

若要定義使用參考屬性路徑的變數，請指定其值propertyPath的欄位。

例如，若要定義具有使用路徑參考屬性之指標的資產模型，您可以像這樣將承載傳遞給 [CreateAssetModel](#)：

```

{
  ...
  "assetModelProperties": [
    {
      ...
      "type": {
        "metric": {
          ...
          "variables": [
            {
              "name": "variable name",
              "value": {
                "propertyPath": [
                  path segments
                ]
              }
            }
          ]
        }
      },
      ...
    },
    ...
  ],
  ...
}

```



```
}
```

設定 AWS IoT SiteWise 物件 IDs

AWS IoT SiteWise 定義各種類型的持久性物件，例如資產、資產模型、屬性和階層。所有這類物件都有唯一識別符，您可以用來擷取、更新和刪除它們。

AWS IoT SiteWise 有不同的選項可供客戶建立 ID。預設會在物件建立時為您 AWS IoT SiteWise 產生一個選項。使用者也可以將自己的 IDs 提供給物件。

主題

- [使用物件 UUIDs](#)
- [使用外部 IDs](#)

使用物件 UUIDs

中的每個持久性物件 AWS IoT SiteWise 都有 [UUID](#) 來識別它。例如，資產模型具有資產模型 ID，資產具有資產 ID，以此類推。此 ID 會在您建立物件時指派，並在物件的生命週期內保持不變。

當您建立新的物件時，預設會為您 AWS IoT SiteWise 產生唯一的 ID。您也可以在建立時以 UUID 格式提供自己的 ID。

Note

UUIDs 建立 AWS 該 UUID 的區域中，以及針對相同物件類型，必須是全域唯一的。當 AWS IoT SiteWise 為您自動產生 ID 時，一律是唯一的。如果您選擇自己的 ID，請確定其是唯一的。

例如，如果您透過呼叫 [CreateAssetModel](#) 建立新的資產模型，您可以在請求的選用 `assetModelId` 欄位中提供自己的 UUID。

相反地，如果您 `assetModelId` 省略請求，會為新的資產模型 AWS IoT SiteWise 產生 UUID。

使用外部 IDs

若要以 UUID 以外的某些格式定義您自己的 ID，您可以指派外部 ID。例如，如果您重複使用在非系統中使用的 ID AWS，或更易於人類閱讀，則可以執行此操作。外部 IDs 的格式更靈活。您可以使用它們在 API 操作中 AWS IoT SiteWise 參考物件，否則會使用 UUID。

如同 UUIDs，每個外部 ID 在其內容中必須是唯一的。例如，您不能有兩個具有相同外部 ID 的資產模型。此外，與 UUIDs 一樣，物件在其生命週期內只能有一個外部 ID，這無法變更。

外部 IDs 和 UUIDs 之間的差異

外部 IDs 與 UUIDs 的差別如下：

- 每個物件都有 UUID，但外部 IDs 是選用的。
- AWS IoT SiteWise 永遠不會產生外部 IDs。您自行提供這些項目。
- 如果物件還沒有，您可以隨時指派外部 ID。

外部 IDs 的格式

有效的外部 ID 具有下列屬性：

- 長度介於 2 到 128 個字元之間。
- 第一個和最後一個字元必須是英數字元 (A-Z、a-z、0-9)。
- 第一個和最後一個以外的字元必須是英數字元，或下列其中一項：_ - . :

例如，外部 ID 必須符合下列規則表達式：

```
[a-zA-Z0-9][a-zA-Z0-9_\-.:]*[a-zA-Z0-9]+
```

參考具有外部 IDs 物件

在許多您可以使用其 UUID 參考物件的地方，您可以改用其外部 ID，如果有的話。若要這樣做，請將外部 ID 附加到字串 `externalId`。

例如，假設您有一個資產模型的 UUID (資產模型 ID) 為 `a1b2c3d4-5678-90ab-cdef-11111EXAMPLE`，其外部 ID 也是 `myExternalId`。呼叫 [DescribeAssetModel](#) 以取得詳細資訊。您可以使用下列其中一項做為的值 `assetModelId`：

- 使用資產模型 ID (UUID) 本身：`a1b2c3d4-5678-90ab-cdef-11111EXAMPLE`
- 使用外部 ID：`externalId:myExternalId`

```
aws iotsitewise describe-asset-model --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
aws iotsitewise describe-asset-model --asset-model-id externalId:myExternalId
```

Note

`externalId`: 字首本身不是外部 ID 的一部分。只有在您將外部 ID 提供給接受 UUIDs 或外部 IDs 的 API 操作時，才需要提供字首。例如，當您查詢或更新現有物件時，請提供字首。當您定義物件的外部 ID 時，例如當您建立資產模型時，請勿包含字首。

您可以將外部 IDs 以這種方式取代 UUIDs，用於中的許多 API 操作 AWS IoT SiteWise，但並非全部。例如，[GetAssetPropertyValue](#) 必須使用 UUIDs 它不支援外部 ID 用量。

若要判斷特定 API 操作是否支援此用量，請參閱 [API 參考](#)。

建立的資產和編譯模型 AWS IoT SiteWise

AWS IoT SiteWise 資產模型和元件模型可推動工業資料的標準化。資產和元件模型代表工業資產及其元件的結構和屬性。資產模型定義整體資產，例如風力渦輪機或生產線。元件模型代表組成資產的個別元件，例如葉片、產生器或感應器。透過建立這些模型，您可以透過反映工業設備真實世界關係和階層的方式組織和建構資產資料，讓您更輕鬆地監控、分析和維護。

資產模型或元件模型包含名稱、描述、資產屬性和（選擇性）自訂複合模型，可將屬性分組在一起，或參考子組件的元件模型。

在中 AWS IoT SiteWise，您可以建立資產模型和元件模型，以代表工業資產及其元件的結構和屬性。

- 您可以使用資產模型來建立資產。除了上述功能之外，資產模型也可以包含階層定義，以定義資產之間的關係。
- 元件模型代表資產模型或其他元件模型中的子組件。建立元件模型時，您可以在資產模型和其他元件模型中為其新增參考。不過，您無法直接從元件模型建立資產。

建立資產模型或元件模型之後，您可以為其建立自訂複合模型，以將屬性分組在一起或參考現有的元件模型。

如需如何建立資產模型和元件模型的詳細資訊，請參閱下列各節。

主題

- [在中建立資產模型 AWS IoT SiteWise](#)
- [建立元件模型](#)

- [定義資料屬性](#)
- [建立自訂複合模型 \(元件\)](#)

在 中建立資產模型 AWS IoT SiteWise

AWS IoT SiteWise 資產模型可推動工業資料的標準化。資產模型包含名稱、說明、資產屬性和資產階層定義。例如，您可以定義具有溫度、每分鐘旋轉次數 (RPM) 和功率屬性的風力渦輪機模型。然後，您可以使用淨功率輸出性質和風力發電機階層定義來定義風力發電場模型。

Note

- 建議您從最低層級的節點開始建立操作模型。例如，在建立風力發電場模型之前，先建立風力發電機模型。資產階層定義包含現有資產模型的參考。利用此方法，您可以在建立模型時定義資產階層。
- 資產模型不能包含其他資產模型。如果您必須將可參考的模型定義為另一個模型中的子組件，則應改為建立 component--> 模型。如需詳細資訊，請參閱[建立元件模型](#)。

下列各節說明如何使用 AWS IoT SiteWise 主控台或 API 來建立資產模型。下列各節也會說明您可用來建立模型的不同類型資產屬性和資產階層。

主題

- [建立資產模型 \(主控台\)](#)
- [建立資產模型 \(AWS CLI\)](#)
- [範例資產模型](#)
- [定義資產模型階層](#)

建立資產模型 (主控台)

您可以使用 AWS IoT SiteWise 主控台來建立資產模型。AWS IoT SiteWise 主控台提供各種功能，例如公式自動完成，可協助您定義有效的資產模型。

建立資產模型 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。

3. 選擇建立模型。
4. 在 Create model (建立模型) 頁面上，執行下列動作：
 - a. 輸入資產模型的 Name (名稱)，例如 **Wind Turbine** 或 **Wind Turbine Model**。此名稱在您於此區域中帳戶的所有模型間都必須是唯一的。
 - b. (選用) 新增模型的外部 ID。這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。
 - c. (選擇性) 新增模型的 Measurement definitions (衡量值定義)。測量代表來自您設備的資料串流。如需詳細資訊，請參閱[從設備定義資料串流 \(度量\)](#)。
 - d. (選擇性) 新增模型的 Transform definitions (轉換定義)。轉換是將資料從一個表單映射到另一個表單的公式。如需詳細資訊，請參閱[轉換資料 \(轉換\)](#)。
 - e. (選擇性) 新增模型的 Metric definitions (指標定義)。指標是跨時間間隔彙總資料的公式。指標可以從相關聯的資產輸入資料，讓您可以計算代表操作或操作子集的值。如需詳細資訊，請參閱[從屬性和其他資產彙總資料 \(指標\)](#)。
 - f. (選擇性) 新增模型的 Hierarchy definitions (階層定義)。階層是資產之間的關係。如需詳細資訊，請參閱[定義資產模型階層](#)。
 - g. (選擇性) 新增資產模型的標籤。如需詳細資訊，請參閱[標記您的 AWS IoT SiteWise 資源](#)。
 - h. 選擇建立模型。

當您建立資產模型時，AWS IoT SiteWise 主控台會導覽至新模型的頁面。在此頁面上，您可以看到模型的 Status (狀態)，最初為 CREATING (建立中)。此頁面會自動更新，因此您可以等候模型的狀態更新。

 Note

複雜模型的資產模型建立程序可能需要幾分鐘的時間。在資產模型狀態為 ACTIVE 之後，您可以使用資產模型來建立資產。如需詳細資訊，請參閱[資產和模型狀態](#)。

5. (選用) 建立資產模型後，您可以為邊緣設定資產模型。如需 SiteWise Edge 的詳細資訊，請參閱 [在 Edge 上設定 AWS IoT SiteWise 邊緣功能](#)。
 - a. 在模型頁面上，選擇設定邊緣。
 - b. 在模型組態頁面上，選擇模型的邊緣組態。這會控制 AWS IoT SiteWise 可以運算和存放與此資產模型相關聯屬性的位置。如需為邊緣設定模型的詳細資訊，請參閱 [在 SiteWise Edge 中設定 OPC UA 來源](#)。

- c. 針對自訂節點組態，選擇您要 AWS IoT SiteWise 運算的位置，並存放每個資產模型屬性。

 Note

必須針對相同的位置設定相關聯的轉換和指標。如需為邊緣設定模型的詳細資訊，請參閱 [在 SiteWise Edge 中設定 OPC UA 來源](#)。

- d. 選擇儲存。在模型頁面上，您的 Edge 組態現在應該已設定。

建立資產模型 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來建立資產模型。

使用 [CreateAssetModel](#) 操作建立具有屬性和階層的資產模型。此操作預期會有具備下列結構的承載。

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition
}
```

建立資產模型 (AWS CLI)

1. 建立名為 `asset-model-payload.json` 的檔案，然後將下列 JSON 物件複製到檔案。

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ],
  "assetModelHierarchies": [

  ],
  "assetModelCompositeModels": [

  ]
}
```

```
}
```

2. 使用您偏好的 JSON 文字編輯器來編輯下列項目的 `asset-model-payload.json` 檔案：
 - a. 輸入資產模型的名稱 (`assetModelName`)，例如 **Wind Turbine** 或 **Wind Turbine Model**。此名稱在帳戶中的所有資產模型和元件模型中必須是唯一的 AWS 區域。
 - b. (選用) 輸入資產模型的外部 ID (`assetModelExternalId`)。這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。
 - c. (選擇性) 輸入資產模型的說明 (`assetModelDescription`)，或移除 `assetModelDescription` 鍵值組。
 - d. (選擇性) 定義模型的資產屬性 (`assetModelProperties`)。如需詳細資訊，請參閱[定義資料屬性](#)。
 - e. (選擇性) 定義模型的資產階層 (`assetModelHierarchies`)。如需詳細資訊，請參閱[定義資產模型階層](#)。
 - f. (選用) 定義模型的警示。警示會監控其他屬性，讓您可以識別設備或程序何時需要注意。每個警示定義都是複合模型 (`assetModelCompositeModels`)，可標準化警示使用的屬性集。如需詳細資訊，請參閱[在中使用警示監控資料 AWS IoT SiteWise](#)及[在中定義資產模型的警示 AWS IoT SiteWise](#)。
 - g. (選擇性) 新增資產模型的標籤 (`tags`)。如需詳細資訊，請參閱[標記您的 AWS IoT SiteWise 資源](#)。
3. 執行以下命令，透過 JSON 文件中的定義建立資產模型。

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-payload.json
```

操作會傳回回應，其中包含建立資產時您參考的 `assetModelId`。回應還包含模型 (`assetModelStatus.state`) 的狀態，起初為 `CREATING`。在變更傳播之前，資產模型狀態會保持為 `CREATING`。

Note

複雜模型的資產模型建立程序可能需要幾分鐘的時間。若要檢查資產模型的目前狀態，請指定來使用 [DescribeAssetModel](#) 操作 `assetModelId`。資產模型狀態變成 `ACTIVE` 後，您便能使用資產模型來建立資產。如需詳細資訊，請參閱[資產和模型狀態](#)。

4. (選用) 為您的資產模型建立自訂複合模型。使用自訂複合模型，您可以在模型中將屬性分組，或透過參考元件模型來包含子組件。如需詳細資訊，請參閱[建立自訂複合模型 \(元件\)](#)。

範例資產模型

本節包含資產模型定義範例，可用來使用 AWS CLI 和 AWS IoT SiteWise SDKs 建立資產模型。這些資產模型代表風力渦輪機和風力發電廠。風力渦輪機資產會擷取原始感應器資料，並計算功率和平均風速等值。風力發電廠資產會計算數值，例如風力發電廠中所有風力發電機的總功率。

主題

- [風力發電機資產模型](#)
- [風力發電場資產模型](#)

風力發電機資產模型

下列資產模型可呈現風力發電場中的風力發電機。風力發電機會擷取感應器資料，以計算功率和平均風速等值。

Note

此範例模型類似於 AWS IoT SiteWise 示範中的風力渦輪機模型。如需詳細資訊，請參閱[使用 AWS IoT SiteWise 示範](#)。

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "Wind Turbine Asset Model",
  "assetModelDescription": "Represents a turbine in a wind farm.",
  "assetModelProperties": [
    {
      "name": "Location",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Renton"
        }
      }
    },
    {
      "name": "Make",
      "dataType": "STRING",
      "type": {
        "attribute": {
```



```

        "defaultValue": "Amazon"
    }
}
},
{
    "name": "Model",
    "dataType": "INTEGER",
    "type": {
        "attribute": {
            "defaultValue": "500"
        }
    }
},
{
    "name": "Torque (KiloNewton Meter)",
    "dataType": "DOUBLE",
    "unit": "kNm",
    "type": {
        "measurement": {}
    }
},
{
    "name": "Wind Direction",
    "dataType": "DOUBLE",
    "unit": "Degrees",
    "type": {
        "measurement": {}
    }
},
{
    "name": "RotationsPerMinute",
    "dataType": "DOUBLE",
    "unit": "RPM",
    "type": {
        "measurement": {}
    }
},
{
    "name": "Wind Speed",
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
        "measurement": {}
    }
}

```

```

    },
    {
      "name": "RotationsPerSecond",
      "dataType": "DOUBLE",
      "unit": "RPS",
      "type": {
        "transform": {
          "expression": "rpm / 60",
          "variables": [
            {
              "name": "rpm",
              "value": {
                "propertyId": "RotationsPerMinute"
              }
            }
          ]
        }
      }
    },
    {
      "name": "Overdrive State",
      "dataType": "DOUBLE",
      "type": {
        "transform": {
          "expression": "gte(torque, 3)",
          "variables": [
            {
              "name": "torque",
              "value": {
                "propertyId": "Torque (KiloNewton Meter)"
              }
            }
          ]
        }
      }
    },
    {
      "name": "Average Power",
      "dataType": "DOUBLE",
      "unit": "Watts",
      "type": {
        "metric": {
          "expression": "avg(torque) * avg(rps) * 2 * 3.14",
          "variables": [

```

```

        {
            "name": "torque",
            "value": {
                "propertyId": "Torque (Newton Meter)"
            }
        },
        {
            "name": "rps",
            "value": {
                "propertyId": "RotationsPerSecond"
            }
        }
    ],
    "window": {
        "tumbling": {
            "interval": "5m"
        }
    }
}
},
{
    "name": "Average Wind Speed",
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
        "metric": {
            "expression": "avg(windspeed)",
            "variables": [
                {
                    "name": "windspeed",
                    "value": {
                        "propertyId": "Wind Speed"
                    }
                }
            ]
        },
        "window": {
            "tumbling": {
                "interval": "5m"
            }
        }
    }
}
},

```

```

{
  "name": "Torque (Newton Meter)",
  "dataType": "DOUBLE",
  "unit": "Nm",
  "type": {
    "transform": {
      "expression": "knm * 1000",
      "variables": [
        {
          "name": "knm",
          "value": {
            "propertyId": "Torque (KiloNewton Meter)"
          }
        }
      ]
    }
  }
},
{
  "name": "Overdrive State Time",
  "dataType": "DOUBLE",
  "unit": "Seconds",
  "type": {
    "metric": {
      "expression": "statetime(overdrive_state)",
      "variables": [
        {
          "name": "overdrive_state",
          "value": {
            "propertyId": "Overdrive State"
          }
        }
      ]
    },
    "window": {
      "tumbling": {
        "interval": "5m"
      }
    }
  }
}
],
"assetModelHierarchies": []

```

```
}
```

風力發電場資產模型

下列資產模型可呈現包括多個風力發電機的風力發電場。此資產模型會定義風力渦輪機模型的階層。這可讓風力發電廠從風力發電廠中所有風力發電機的資料計算值（例如平均功率）。

Note

此範例模型類似於 AWS IoT SiteWise 示範中的風力發電廠模型。如需詳細資訊，請參閱[使用 AWS IoT SiteWise 示範](#)。

此資產模型取決於 [風力發電機資產模型](#)。將 `propertyId` 和 `childAssetModelId` 值取代為來自現有風力發電機資產模型中的值。

```
{
  "assetModelName": "Wind Farm Asset Model",
  "assetModelDescription": "Represents a wind farm.",
  "assetModelProperties": [
    {
      "name": "Code",
      "dataType": "INTEGER",
      "type": {
        "attribute": {
          "defaultValue": "300"
        }
      }
    },
    {
      "name": "Location",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Renton"
        }
      }
    },
    {
      "name": "Reliability Manager",
      "dataType": "STRING",
      "type": {
```

```

        "attribute": {
            "defaultValue": "Mary Major"
        }
    },
    {
        "name": "Total Overdrive State Time",
        "dataType": "DOUBLE",
        "unit": "seconds",
        "type": {
            "metric": {
                "expression": "sum(overdrive_state_time)",
                "variables": [
                    {
                        "name": "overdrive_state_time",
                        "value": {
                            "propertyId": "ID of Overdrive State Time property in Wind Turbine
Asset Model",
                            "hierarchyId": "Turbine Asset Model"
                        }
                    }
                ],
                "window": {
                    "tumbling": {
                        "interval": "5m"
                    }
                }
            }
        }
    },
    {
        "name": "Total Average Power",
        "dataType": "DOUBLE",
        "unit": "Watts",
        "type": {
            "metric": {
                "expression": "sum(turbine_avg_power)",
                "variables": [
                    {
                        "name": "turbine_avg_power",
                        "value": {
                            "propertyId": "ID of Average Power property in Wind Turbine Asset
Model",
                            "hierarchyId": "Turbine Asset Model"
                        }
                    }
                ]
            }
        }
    }
]

```

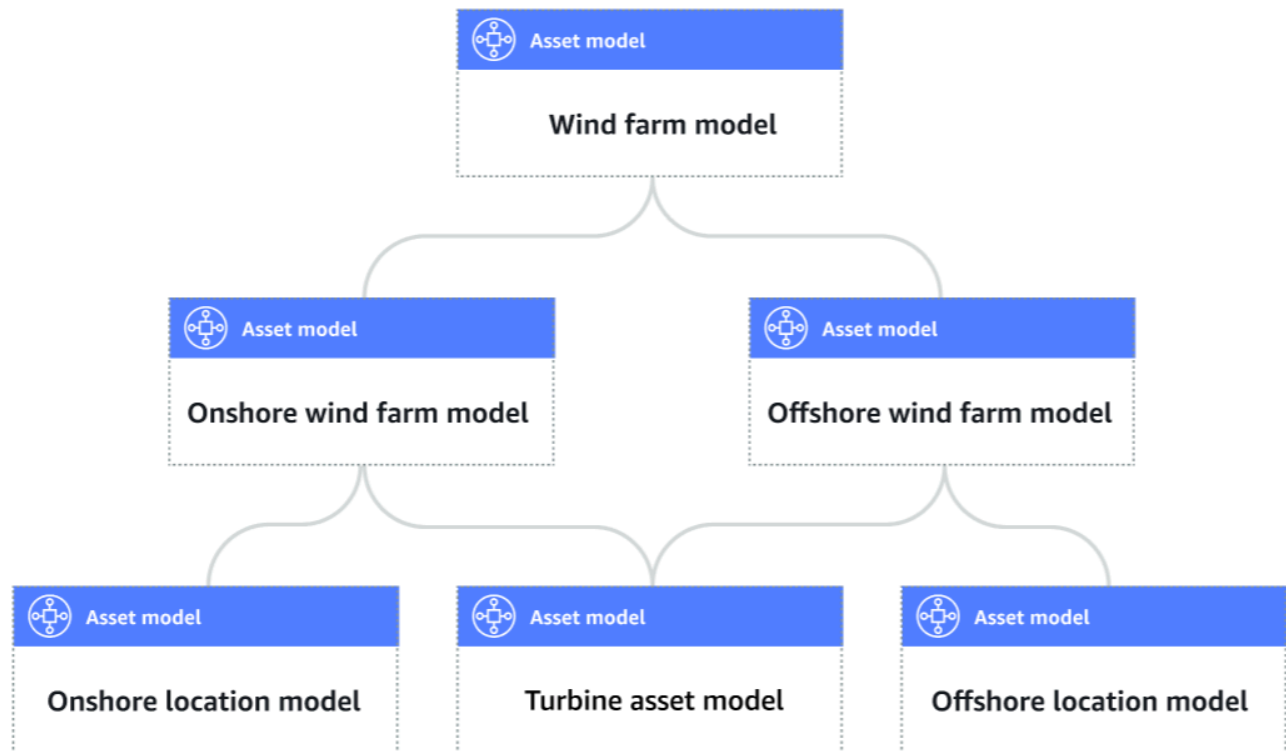
```
        }
      }
    ],
    "window": {
      "tumbling": {
        "interval": "5m"
      }
    }
  }
},
"assetModelHierarchies": [
  {
    "name": "Turbine Asset Model",
    "childAssetModelId": "ID of Wind Turbine Asset Model"
  }
]
```

定義資產模型階層

您可以定義資產模型階層，在工業操作中的資產模型之間建立邏輯關聯。例如，您可以定義由陸上和海上風力發電廠組成的風力發電廠。陸上風力發電廠包含渦輪機和陸上位置。海上風力發電廠包含渦輪機和海上位置。



Asset model hierarchy



當您透過階層將子資產模型與父資產模型建立關聯時，父資產模型的指標可以從子資產模型的指標輸入資料。您可以使用資產模型階層和指標來計算統計資料，以深入了解您的操作或操作的子集。如需詳細資訊，請參閱[從屬性和其他資產彙總資料（指標）](#)。

每個階層都會定義父資產模型與子資產模型之間的關係。在父資產模型中，您可以為相同的子資產模型定義多個階層。例如，如果您的風力發電廠中有兩個不同類型的風力發電機，其中所有風力發電機都由相同的資產模型表示，您可以為每個類型定義階層。然後，您可以在風力發電廠模型中定義指標，以計算每種風力發電機類型的獨立和合併統計資料。

父資產模型可以與多個子資產模型建立關聯。例如，如果您有 上岸風力發電廠和一個由兩種不同的資產模型表示的離岸風力發電廠，您可以將這些資產模型與相同的父風力發電廠資產模型建立關聯。

子資產模型也可以與多個父資產模型建立關聯。例如，如果您有兩個不同類型的風力發電廠，其中所有風力發電機都由相同的資產模型表示，您可以將風力發電機資產模型與不同的風力發電廠資產模型建立關聯。

Note

當您定義資產模型階層時，子資產模型必須是ACTIVE或具有先前的ACTIVE版本。如需詳細資訊，請參閱[資產和模型狀態](#)。

定義階層式資產模型並建立資產之後，您可以將資產關聯，以完成父子關係。如需詳細資訊，請參閱[在中建立資產模型的資產 AWS IoT SiteWise](#)及[關聯和取消關聯資產](#)。

主題

- [定義資產模型階層（主控台）](#)
- [定義資產階層 \(AWS CLI\)](#)

定義資產模型階層（主控台）

當您在 AWS IoT SiteWise 主控台中定義資產模型的階層時，您可以指定下列參數：

- 階層名稱 – 階層的名稱，例如 **Wind Turbines**。
- 階層模型 – 子資產模型。
- 階層外部 ID（選用） – 這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

如需詳細資訊，請參閱[建立資產模型（主控台）](#)。

定義資產階層 (AWS CLI)

當您使用 AWS IoT SiteWise API 定義資產模型的階層時，您可以指定下列參數：

- name – 階層的名稱，例如 **Wind Turbines**。
- childAssetModelId – 階層子資產模型的 ID 或外部 ID。您可以使用 [ListassetModel](#) 操作來尋找現有資產模型的 ID。

Example 範例階層定義

下列範例示範資產模型階層，其代表風力發電廠與風力發電機的關係。此物件是 [AssetModelHierarchy](#) 的範例。如需詳細資訊，請參閱[建立資產模型 \(AWS CLI\)](#)。

```
{
  ...
  "assetModelHierarchies": [
    {
      "name": "Wind Turbines",
      "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
    },
  ],
}
```

建立元件模型

使用 AWS IoT SiteWise 元件模型來定義您可以從資產模型或其他元件模型參考的子組件。如此一來，您可以在多個其他模型中重複使用元件的定義，或在相同的模型內多次使用元件。

定義元件模型的程序與定義資產模型非常類似。如同資產模型，元件模型具有名稱、描述和資產屬性。不過，元件模型不能包含資產階層定義，因為元件模型本身無法直接用來建立資產。元件模型也無法定義警示。

例如，您可以定義具有馬達溫度、編碼器溫度和隔離電阻屬性的 伺服馬達元件。然後，您可以為包含伺服馬達的設備定義資產模型，例如 CNC 機器。

Note

- 建議您從最低層級的節點開始建立操作模型。例如，在建立 CNC 機器的資產模型之前，請先建立您的伺服馬達元件。資產模型包含現有元件模型的參考。
- 您無法直接從元件模型建立資產。若要建立使用您的元件的資產，您必須為資產建立資產模型。然後，您可以為其建立參考元件的自訂複合模型。如需建立資產模型的詳細資訊，請參閱[在中建立資產模型 AWS IoT SiteWise](#)。如需建立自訂複合模型的詳細資訊，請參閱[建立自訂複合模型（元件）](#)。

下列各節說明如何使用 AWS IoT SiteWise API 來建立元件模型。

主題

- [建立元件模型 \(AWS CLI\)](#)
- [範例元件模型](#)

建立元件模型 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來建立元件模型。

使用 [CreateAssetModel](#) 操作建立具有 屬性的元件模型。此操作預期承載具有下列結構：

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
}
```

建立元件模型 (AWS CLI)

1. 建立名為 的檔案，component-model-payload.json然後將下列 JSON 物件複製到 檔案：

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ]
}
```

2. 使用您偏好的 JSON 文字編輯器來編輯下列項目的 component-model-payload.json 檔案：
 - a. 輸入元件模型的名稱 (assetModelName)，例如 **Servo Motor**或 **Servo Motor Model**。此名稱在帳戶中的所有資產模型和元件模型中必須是唯一的 AWS 區域。
 - b. （選用）輸入元件模型的外部 ID (assetModelExternalId)。這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs物件](#)。
 - c. (選擇性) 輸入資產模型的說明 (assetModelDescription)，或移除 assetModelDescription 鍵值組。
 - d. （選用）定義元件模型的資產屬性 (assetModelProperties)。如需詳細資訊，請參閱[定義資料屬性](#)。
 - e. (選擇性) 新增資產模型的標籤 (tags)。如需詳細資訊，請參閱[標記您的 AWS IoT SiteWise 資源](#)。
3. 執行下列命令，從 JSON 檔案中的定義建立元件模型。

```
aws iotsitewise create-asset-model --cli-input-json file://component-model-payload.json
```

操作會傳回回應，其中包含assetModelId您在資產模型或其他元件模型中將參考新增至元件模型時所參考的。回應還包含模型(assetModelStatus.state)的狀態，起初為CREATING。元件模型的狀態為，CREATING直到變更傳播為止。

Note

複雜模型的元件模型建立程序最多可能需要幾分鐘的時間。若要檢查元件模型的目前狀態，請指定來使用 [DescribeAssetModel](#) 操作assetModelId。在元件模型狀態為之後ACTIVE，您可以在資產模型或其他元件模型中新增元件模型的參考。如需詳細資訊，請參閱[資產和模型狀態](#)。

4. (選用) 為您的元件模型建立自訂複合模型。使用自訂複合模型，您可以在模型中分組屬性，或透過參考另一個元件模型來包含子組件。如需詳細資訊，請參閱[建立自訂複合模型 \(元件\)](#)。

範例元件模型

本節包含元件模型定義範例，可用來建立具有 AWS CLI 和 AWS IoT SiteWise SDKs元件模型。此元件模型代表可在 CNC 機器等其他設備中使用的伺服馬達。

主題

- [伺服馬達元件模型](#)

伺服馬達元件模型

下列元件模型代表可在 CNC 機器等設備中使用的伺服馬達。伺服馬達提供各種測量，例如溫度和電阻。這些衡量值可作為參考 伺服馬達元件模型之資產模型所建立資產的屬性。

```
{
  "assetModelName": "ServoMotor",
  "assetModelType": "COMPONENT_MODEL",
  "assetModelProperties": [
    {
      "dataType": "DOUBLE",
      "name": "Servo Motor Temperature",
      "type": {
```

```

        "measurement": {},
      },
      "unit": "Celsius"
    },
    {
      "dataType": "DOUBLE",
      "name": "Spindle speed",
      "type": {
        "measurement": {}
      },
      "unit": "rpm"
    }
  ]
}

```

定義資料屬性

資產屬性是每個資產中包含資產資料的結構。資產屬性可以是下列任一類型：

- **屬性** – 資產的一般靜態屬性，例如裝置製造商或地理區域。如需詳細資訊，請參閱[定義靜態資料 \(屬性\)](#)。
- **測量** – 資產的原始裝置感應器資料串流，例如時間戳記的旋轉速度值或以攝氏為單位的時間戳記溫度值。衡量值由資料串流別名定義。如需詳細資訊，請參閱[從設備定義資料串流 \(度量\)](#)。
- **轉換** – 資產的轉換時間序列值，例如 Fahrenheit 中的時間戳記溫度值。轉換由表達式和變數定義，以使用該表達式。如需詳細資訊，請參閱[轉換資料 \(轉換\)](#)。
- **指標** – 在指定時間間隔內彙總的資產資料，例如每小時平均溫度。指標由時間間隔、表達式以及與該表示式搭配使用的變數所定義。指標表達式可以輸入相關聯資產的指標屬性，以便您可以計算代表操作或操作子集的指標。如需詳細資訊，請參閱[從屬性和其他資產彙總資料 \(指標\)](#)。

如需詳細資訊，請參閱[在中建立資產模型 AWS IoT SiteWise](#)。

如需如何使用衡量值、轉換和指標來計算「整體設備效率」(OEE) 的範例，請參閱[在中計算 OEE AWS IoT SiteWise](#)。

主題

- [定義靜態資料 \(屬性\)](#)
- [從設備定義資料串流 \(度量\)](#)
- [轉換資料 \(轉換\)](#)

- [從屬性和其他資產彙總資料（指標）](#)
- [使用公式表達式](#)

定義靜態資料（屬性）

資產屬性代表通常為靜態的資訊，例如裝置製造商或地理位置。您從資產模型建立的每個資產，皆會包含該模型的屬性。

主題

- [定義屬性（主控台）](#)
- [定義屬性 \(AWS CLI\)](#)

定義屬性（主控台）

當您在 AWS IoT SiteWise 主控台中定義資產模型的屬性時，您可以指定下列參數：

- 名稱 – 屬性的名稱。
- 預設值 –（選用）此屬性的預設值。從模型建立的資產具有此屬性值。對於從模型建立的資產，如需如何覆寫預設值的詳細資訊，請參閱 [更新屬性值](#)。
- 資料類型 – 屬性的資料類型，其為下列其中一項：
 - 字串 – 最多 1024 個位元組的字串。
 - 整數 – 範圍為 **【-2,147,483,648, 2,147,483,647】** 的帶正負號 32 位元整數。
 - Double – 範圍為 **【-10¹⁰⁰, 10¹⁰⁰】** 和 IEEE 754 雙精度的浮點數。
 - 布林值 – true 或 false。
- 外部 ID –（選用）這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

如需詳細資訊，請參閱 [建立資產模型（主控台）](#)。

定義屬性 (AWS CLI)

當您使用 AWS IoT SiteWise API 定義資產模型的屬性時，您可以指定下列參數：

- name – 屬性的名稱。
- defaultValue –（選用）此屬性的預設值。從模型建立的資產具有此屬性值。對於從模型建立的資產，如需如何覆寫預設值的詳細資訊，請參閱 [更新屬性值](#)。

- `dataType` – 屬性的資料類型，其為下列其中一項：
 - `STRING` – 最多 1024 個位元組的字串。
 - `INTEGER` – 範圍為 `[-2, 147, 483, 648, 2, 147, 483, 647]` 的帶正負號 32 位元整數。
 - `DOUBLE` – 範圍為 `[-10^100, 10^100]` 和 IEEE 754 雙精度的浮點數。
 - `BOOLEAN` – `true`或 `false`。
- `externalId` – (選用) 這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs物件](#)。

Example 範例屬性定義

下列範例示範會將資產的模型編號以預設值呈現的屬性。此物件是包含 [屬性的 `AssetModelProperty`](#) 範例。您可以將此物件指定為 [CreateAssetModel](#) 請求承載的一部分，以建立 屬性。如需詳細資訊，請參閱[建立資產模型 \(AWS CLI\)](#)。

```
{
  ...
  "assetModelProperty": [
    {
      "name": "Model number",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "BLT123"
        }
      }
    }
  ],
  ...
}
```

從設備定義資料串流 (度量)

測量代表裝置的原始感應器資料串流，例如時間戳記溫度值或每分鐘時間戳記輪換 (RPM) 值。

主題

- [定義測量 \(主控台\)](#)
- [定義測量 \(AWS CLI\)](#)

定義測量（主控台）

當您在 AWS IoT SiteWise 主控台中定義資產模型的測量結果時，您可以指定下列參數：

- 名稱 – 屬性的名稱。
- 單位 – （選用）屬性的科學單位，例如 mm 或 Celsius。
- 資料類型 – 屬性的資料類型，其為下列其中一項：
 - 字串 – 最多 1024 個位元組的字串。
 - 整數 – 範圍為 **【-2, 147, 483, 648, 2, 147, 483, 647】** 的帶正負號 32 位元整數。
 - Double – 範圍為 **【-10¹⁰⁰、10¹⁰⁰】** 和 IEEE 754 雙精度的浮點數。
 - 布林值 – true 或 false。
- 外部 ID – （選用）這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

如需詳細資訊，請參閱[建立資產模型（主控台）](#)。

定義測量 (AWS CLI)

當您使用 AWS IoT SiteWise API 定義資產模型的測量時，您可以指定下列參數：

- name – 屬性的名稱。
- dataType – 屬性的資料類型，其為下列其中一項：
 - STRING – 最多 1024 個位元組的字串。
 - INTEGER – 範圍為 **【-2, 147, 483, 648, 2, 147, 483, 647】** 的帶正負號 32 位元整數。
 - DOUBLE – 範圍為 **【-10¹⁰⁰、10¹⁰⁰】** 和 IEEE 754 雙精度的浮點數。
 - BOOLEAN – true 或 false。
- unit – （選用）屬性的科學單位，例如 mm 或 Celsius。
- externalId – （選用）這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

Example 範例衡量值定義

下列範例示範代表資產溫度感測器讀數的衡量值。此物件是包含測量的 [AssetModelProperty](#) 範例。您可以將此物件指定為 [CreateAssetModel](#) 請求承載的一部分，以建立測量屬性。如需詳細資訊，請參閱[建立資產模型 \(AWS CLI\)](#)。

當您定義資產模型時，[測量](#)結構是空的結構，因為您稍後會將每個資產設定為使用唯一的裝置資料串流。如需如何將資產的測量屬性連接至裝置感應器資料串流的詳細資訊，請參閱 [管理的資料串流 AWS IoT SiteWise](#)。

```
{
  ...
  "assetModelProperties": [
    {
      "name": "Temperature C",
      "dataType": "DOUBLE",
      "type": {
        "measurement": {}
      },
      "unit": "Celsius"
    }
  ],
  ...
}
```

轉換資料（轉換）

轉換是數學表達式，可將資產屬性的資料點從一種形式映射到另一種形式。轉換表達式由資產屬性變數、常值、運算子和函數組成。轉換的資料點與輸入資料點保持one-to-one的關係。會在每次任何輸入屬性接收到新的資料點時 AWS IoT SiteWise 計算新的轉換資料點。

Note

對於具有相同時間戳記的屬性更新，來自其他傳入屬性的更新可能會覆寫輸出值。

例如，如果您的資產具有名為 Temperature_C、以攝氏單位為單位的溫度衡量值串流，您可以使用公式 $\text{Temperature\_F} = 9/5 * \text{Temperature\_C} + 32$ 將每個資料點轉換為華氏。每次在 Temperature_C 測量串流中 AWS IoT SiteWise 收到資料點時，都會在幾秒鐘內計算對應的 Temperature_F 值，並以 Temperature_F 屬性的形式提供。

如果您的轉換包含多個變數，則稍早抵達的資料點會立即啟動運算。考慮一個零件製造商使用轉換來監控產品品質的範例。製造商會根據組件類型使用不同的標準，使用以下測量來代表程序：

- Part_Number - 識別組件類型的字串。
- Good_Count - 如果組件符合標準，則會增加 1 的整數。

- `Bad_Count` - 如果部分不符合標準，則會增加 1 的整數。

製造商也會建立等於 `Quality_Monitor` 的轉換 `if(eq(Part_Number, "BLT123") and (Bad_Count / (Good_Count + Bad_Count) > 0.1), "Caution", "Normal")`。

此轉換會監控針對特定組件類型產生的不良組件百分比。如果組件編號為 BLT123，且不良組件的百分比超過 10% (0.1)，則轉換會傳回 "Caution"。否則，轉換會傳回 "Normal"。

Note

- 如果在其他測量之前 `Part_Number` 收到新的資料點，`Quality_Monitor` 轉換會使用新的 `Part_Number` 值和最新的 `Good_Count` 和 `Bad_Count` 值。若要避免錯誤，請在下一次製造執行 `Bad_Count` 之前重設 `Good_Count` 和。
- 如果您只想在所有變數接收到新的資料點之後評估表達式，請使用 [指標](#)。

主題

- [定義轉換（主控台）](#)
- [定義轉換 \(AWS CLI\)](#)

定義轉換（主控台）

當您在 AWS IoT SiteWise 主控台中定義資產模型的轉換時，您可以指定下列參數：

- 名稱 – 屬性的名稱。
- 單位 –（選用）屬性的科學單位，例如 mm 或 Celsius。
- 資料類型 – 轉換的資料類型，可以是 Double 或 String。
- 外部 ID –（選用）這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。
- 公式 – 轉換表達式。轉換表達式無法使用彙總函數或暫時函數。若要開啟自動完成功能，請開始輸入或按向下箭頭鍵。如需詳細資訊，請參閱 [使用公式表達式](#)。

Important

轉換可以輸入整數、雙數、布林值或字串類型的屬性。布林值轉換為 0(false) 和 1(true)。

轉換必須輸入一或多個非屬性的屬性，以及任意數量的屬性屬性。每次非屬性的輸入屬性收到新的資料點時，都會 AWS IoT SiteWise 計算新的轉換資料點。新的屬性值不會啟動轉換更新。資產屬性資料 API 操作的相同請求率適用於轉換運算結果。

公式表達式只能輸出雙值或字串值。巢狀運算式可以輸出其他資料類型，例如字串，但整體公式必須評估為數字或字串。您可以使用 [jp 函數](#) 將字串轉換為數字。布林值必須為 1 (true) 或 0 (false)。如需詳細資訊，請參閱[未定義、無限和溢出值](#)。

如需詳細資訊，請參閱[建立資產模型（主控台）](#)。

定義轉換 (AWS CLI)

當您使用 AWS IoT SiteWise API 定義資產模型的轉換時，您可以指定下列參數：

- name – 屬性的名稱。
- unit – (選用) 屬性的科學單位，例如 mm 或 Celsius。
- dataType – 轉換的資料類型，必須是 DOUBLE 或 STRING。
- externalId – (選用) 這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。
- expression – 轉換表達式。轉換表達式無法使用彙總函數或暫時函數。如需詳細資訊，請參閱[使用公式表達式](#)。
- variables – 變數清單，定義要在表達式中使用的資產其他屬性。每個變數結構都包含可在表達式中使用的簡單名稱，以及一個 value 結構，用來識別要連結到該變數的屬性。value 結構包含以下資訊：
 - propertyId – 要輸入值之屬性的 ID。您可以使用屬性的名稱而非其 ID。

Important

轉換可以輸入整數、雙數、布林值或字串類型的屬性。布林值轉換為 0(false) 和 1(true)。轉換必須輸入一或多個非屬性的屬性，以及任意數量的屬性屬性。每次非屬性的輸入屬性收到新的資料點時，都會 AWS IoT SiteWise 計算新的轉換資料點。新的屬性值不會啟動轉換更新。資產屬性資料 API 操作的相同請求率適用於轉換運算結果。

公式表達式只能輸出雙值或字串值。巢狀運算式可以輸出其他資料類型，例如字串，但整體公式必須評估為數字或字串。您可以使用 [jp 函數](#) 將字串轉換為數字。布林值必須為 1 (true) 或 0 (false)。如需詳細資訊，請參閱[未定義、無限和溢出值](#)。

Example 轉換定義

下列範例示範一種轉換屬性，可將資產的溫度衡量值從攝氏轉換為華氏。此物件是包含[轉換](#)的 [AssetModelProperty](#) 範例。您可以將此物件指定為 [CreateAssetModel](#) 請求承載的一部分，以建立轉換屬性。如需詳細資訊，請參閱[建立資產模型 \(AWS CLI\)](#)。

```
{
  ...
  "assetModelProperty": [
    ...
    {
      "name": "Temperature F",
      "dataType": "DOUBLE",
      "type": {
        "transform": {
          "expression": "9/5 * temp_c + 32",
          "variables": [
            {
              "name": "temp_c",
              "value": {
                "propertyId": "Temperature C"
              }
            }
          ]
        }
      },
      "unit": "Fahrenheit"
    }
  ],
  ...
}
```

Example 包含三個變數的轉換定義

下列範例示範轉換屬性，如果超過 10% 的 BLT123 組件不符合標準，則會傳回警告訊息 ("Caution")。否則，它會傳回資訊訊息 ("Normal")。

```
{
  ...
  "assetModelProperty": [
    ...
    {
      "name": "Quality_Monitor",
```

```

"dataType": "STRING",
"type": {
  "transform": {
    "expression": "if(eq(Part_Number,\"BLT123\") and (Bad_Count / (Good_Count +
Bad_Count) > 0.1), \"Caution\", \"Normal\")",
    "variables": [
      {
        "name": "Part_Number",
        "value": {
          "propertyId": "Part Number"
        }
      },
      {
        "name": "Good_Count",
        "value": {
          "propertyId": "Good Count"
        }
      },
      {
        "name": "Bad_Count",
        "value": {
          "propertyId": "Bad Count"
        }
      }
    ]
  }
}
}
...
}

```

從屬性和其他資產彙總資料（指標）

指標是數學表達式，使用彙總函數來處理所有輸入資料點，並在每個指定的時間間隔輸出單一資料點。例如，指標可以從溫度資料串流計算每小時平均溫度。

指標可以從關聯資產的指標輸入資料，因此您可以計算統計資料，以提供您的操作或操作子集的深入解析。例如，指標可以計算風力發電場中所有風力發電機的平均每小時溫度。如需如何定義資產之間關聯的詳細資訊，請參閱 [定義資產模型階層](#)。

指標也可以從其他屬性輸入資料，而無需在每個時間間隔內彙總資料。如果您在公式中指定[屬性](#)，AWS IoT SiteWise 會在計算公式時使用該屬性的[最新值](#)。如果您在公式中指定指標，AWS IoT SiteWise 會使用其計算公式之時間間隔的[最後一個值](#)。這表示您可以定義等指標OEE

= Availability * Quality * Performance，其中 Availability、Quality 和 Performance 是相同資產模型上的所有其他指標。

AWS IoT SiteWise 也會自動運算所有資產屬性的一組基本彙總指標。若要降低運算成本，您可以使用這些彙總，而不要為基本運算定義自訂指標。如需詳細資訊，請參閱[在中查詢資產屬性彙總 AWS IoT SiteWise](#)。

主題

- [定義指標（主控台）](#)
- [定義指標 \(AWS CLI\)](#)

定義指標（主控台）

當您在 AWS IoT SiteWise 主控台中定義資產模型的指標時，您可以指定下列參數：

- 名稱 – 屬性的名稱。
- 資料類型 – 轉換的資料類型，可以是 Double 或 String。
- 外部 ID – （選用）這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的[參考具有外部 IDs 物件](#)。
- 公式 – 指標表達式。指標表達式可以使用[彙總函數](#)，從階層中所有相關聯資產的屬性輸入資料。開始輸入或按向下箭頭鍵開啟自動完成功能。如需詳細資訊，請參閱[使用公式表達式](#)。

Important

指標只能是整數、雙數、布林值或字串類型的屬性。布林值轉換為 0(false) 和 1(true)。如果您在指標表達式中定義任何指標輸入變數，則這些輸入必須和輸出指標有相同的時間間隔。

公式表達式只能輸出雙值或字串值。巢狀運算式可以輸出其他資料類型，例如字串，但整體公式必須評估為數字或字串。您可以使用 [jp 函數](#) 將字串轉換為數字。布林值必須為 1 (true) 或 0 (false)。如需詳細資訊，請參閱[未定義、無限和溢出值](#)。

- 時間間隔 – 指標時間間隔。AWS IoT SiteWise 支援下列輪轉時段時間間隔，其中每個間隔會在前一個時間間隔結束時開始：
 - 1 分鐘 – 1 分鐘，在每分鐘結束時（上午 12 : 00 : 00、上午 12 : 01 : 00、上午 12 : 02 : 00 等）計算。
 - 5 分鐘 – 5 分鐘，從小時開始每 5 分鐘計算一次（上午 12 : 00 : 00、上午 12 : 05 : 00、上午 12 : 10 : 00 等）。

- 15 分鐘 – 15 分鐘，從小時開始每 15 分鐘的結尾開始計算（上午 12 : 00 : 00、上午 12 : 15 : 00、上午 12 : 30 : 00 等）。
- 1 小時 – 1 小時 (60 分鐘)，每小時結束時以 UTC 計算（上午 12 : 00 : 00、上午 01 : 00 : 00、上午 02 : 00 : 00 等）。
- 1 天 – 1 天 (24 小時)，在每天結束時以 UTC 計算（星期一上午 12 : 00 : 00、星期二上午 12 : 00 : 00，以此類推）。
- 1 週 – 1 週 (7 天)，在 UTC 的每個星期日結束時計算（星期一上午 12 : 00 : 00）。
- 自訂間隔 – 您可以輸入一分鐘到一週之間的任何時間間隔。
- 位移日期 –（選用）要彙總資料的參考日期。
- 位移時間 –（選用）要彙總資料的參考時間。偏移時間必須介於 00 : 00 : 00 和 23 : 59 : 59 之間。
- 位移時區 –（選用）位移的時區。如果未指定，則預設位移時區為國際標準時間 (UTC)。

支援的時區

- (UTC+00 : 00) 國際標準時間
- (UTC+01 : 00) 歐洲中部時間
- (UTC+02 : 00) 東歐
- (UTC+03 : 00) 東非時間
- (UTC+04 : 00) 接近東部時間
- (UTC+05 : 00) 巴基斯坦拉合爾時間
- (UTC+05 : 30) 印度標準時間
- (UTC+06 : 00) 孟加拉標準時間
- (UTC+07 : 00) 越南標準時間
- (UTC+08 : 00) 中國台灣時間
- (UTC+09 : 00) 日本標準時間
- (UTC+09 : 30) 澳洲中部時間
- (UTC+10 : 00) 澳洲東部時間
- (UTC+11 : 00) 索羅門標準時間
- (UTC+12 : 00) 紐西蘭標準時間
- (UTC-11 : 00) 中途群島時間
- (UTC-10 : 00) 夏威夷標準時間
- (UTC-09 : 00) 阿拉斯加標準時間

- (UTC-08 : 00) 太平洋標準時間
- (UTC-07 : 00) 鳳凰城標準時間
- (UTC-06 : 00) 中部標準時間
- (UTC-05 : 00) 東部標準時間
- (UTC-04 : 00) 波多黎各和美屬維京群島時間
- (UTC-03 : 00) 阿根廷標準時間
- (UTC-02 : 00) 南喬治亞時間
- (UTC-01 : 00) 中非時間

Example 具有位移的自訂時間間隔（主控台）

下列範例說明如何在 2021 年 2 月 20 日下午 6 : 30 : 30 (PST) 定義偏移量的 12 小時時間間隔。

定義具有位移的自訂間隔

1. 針對時間間隔，選擇自訂間隔。
2. 針對時間間隔，執行下列其中一項操作：
 - 輸入 **12**，然後選擇小時。
 - 輸入 **720**，然後選擇分鐘。
 - 輸入 **43200**，然後選擇秒。

 Important

無論單位為何，時間間隔都必須是整數。

3. 針對位移日期，選擇 2021/02/20。
4. 針對位移時間，輸入 **18:30:30**。
5. 針對位移時區，選擇 (UTC-08 : 00) 太平洋標準時間。

如果您在 2021 年 7 月 1 日下午 06 : 30 : 30 (PST) 之前或當天建立指標，您會在 2021 年 7 月 1 日下午 06 : 30 : 30 (PST) 取得第一個彙總結果。第二個彙總結果是 2021 年 7 月 2 日上午 06 : 30 : 30 (PST)，以此類推。

定義指標 (AWS CLI)

當您使用 AWS IoT SiteWise API 定義資產模型的指標時，您可以指定下列參數：

- `name` – 屬性的名稱。
- `dataType` – 指標的資料類型，可以是 `DOUBLE` 或 `STRING`。
- `externalId` – (選用) 這是使用者定義的 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。
- `expression` – 指標表達式。指標表達式可以使用 [彙總函數](#)，從屬性輸入階層中所有相關聯資產的資料。如需詳細資訊，請參閱 [使用公式表達式](#)。
- `window` – 指標輪轉時段的时间間隔和位移，其中每個間隔會在前一個間隔結束時開始：
 - `interval` – 輪轉時段的时间間隔。時間間隔必須介於一分鐘和一週之間。
 - `offsets` – 輪轉時段的位移。

如需詳細資訊，請參閱 AWS IoT SiteWise API 參考中的 [TumblingWindow](#)。

Example 具有位移的自訂時間間隔 (AWS CLI)

下列範例說明如何在 2021 年 2 月 20 日下午 6 : 30 : 30 (PST) 以偏移量定義 12 小時的時間間隔。

```
{
  "window": {
    "tumbling": {
      "interval": "12h",
      "offset": " 2021-07-23T18:30:30-08"
    }
  }
}
```

如果您在 2021 年 7 月 1 日下午 06 : 30 : 30 (PST) 之前或當天建立指標，您會在 2021 年 7 月 1 日下午 06 : 30 : 30 (PST) 取得第一個彙總結果。第二個彙總結果是 2021 年 7 月 2 日上午 06 : 30 : 30 (PST)，以此類推。

- `variables` – 變數清單，定義要在表達式中使用的資產或子資產的其他屬性。每個變數結構都包含可在表達式中使用的簡單名稱，以及一個 `value` 結構，用來識別要連結到該變數的屬性。`value` 結構包含以下資訊：
 - `propertyId` – 要從中提取值的屬性 ID。如果屬性已在目前的模型中定義 (而非在階層中的模型內定義)，您可以使用屬性的名稱而非其 ID。

- `hierarchyId` – (選用) 從中查詢屬性子資產的階層 ID。您可以使用階層定義的名稱，而非其 ID。如果您省略此值，會在目前的模型中 AWS IoT SiteWise 尋找屬性。

⚠ Important

指標只能是整數、雙數、布林值或字串類型的屬性。布林值轉換為 0(false) 和 1(true)。如果您在指標表達式中定義任何指標輸入變數，則這些輸入必須和輸出指標有相同的時間間隔。

公式表達式只能輸出雙值或字串值。巢狀運算式可以輸出其他資料類型，例如字串，但整體公式必須評估為數字或字串。您可以使用 [jp 函數](#) 將字串轉換為數字。布林值必須為 1 (true) 或 0 (false)。如需詳細資訊，請參閱[未定義、無限和溢出值](#)。

- `unit` – (選用) 屬性的科學單位，例如 mm 或 Celsius。

Example 指標定義範例

下列範例示範彙總資產溫度衡量值的指標屬性，以計算每小時華氏溫度上限。此物件是包含[指標的 AssetModelProperty](#) 範例。您可以將此物件指定為 [CreateAssetModel](#) 請求承載的一部分，以建立指標屬性。如需詳細資訊，請參閱[建立資產模型 \(AWS CLI\)](#)。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "Max temperature",
      "dataType": "DOUBLE",
      "type": {
        "metric": {
          "expression": "max(temp_f)",
          "variables": [
            {
              "name": "temp_f",
              "value": {
                "propertyId": "Temperature F"
              }
            }
          ]
        },
        "window": {
```

```

        "tumbling": {
            "interval": "1h"
        }
    },
    "unit": "Fahrenheit"
},
...
}

```

Example 從相關聯資產輸入資料的範例指標定義

下列範例示範的指標屬性會彙總多個風力發電機的平均功率資料，以計算風力發電廠的總平均功率。此物件是包含指標的 [AssetModelProperty](#) 範例。您可以將此物件指定為 [CreateAssetModel](#) 請求承載的一部分，以建立指標屬性。

```

{
    ...
    "assetModelProperties": [
        ...
        {
            "name": "Total Average Power",
            "dataType": "DOUBLE",
            "type": {
                "metric": {
                    "expression": "avg(power)",
                    "variables": [
                        {
                            "name": "power",
                            "value": {
                                "propertyId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
                                "hierarchyId": "Turbine Asset Model"
                            }
                        }
                    ]
                },
                "window": {
                    "tumbling": {
                        "interval": "5m"
                    }
                }
            }
        }
    ]
}

```

```
    },  
    "unit": "kWh"  
  }  
],  
  ...  
}
```

使用公式表達式

使用公式表達式，您可以定義數學函數來轉換和彙整您的原始產業資料，以取得您操作的詳情。公式表達式結合常值、運算子、函數和變數來處理資料。如需如何定義使用公式表達式之資產屬性的詳細資訊，請參閱 [轉換資料（轉換）](#) 和 [從屬性和其他資產彙總資料（指標）](#)。轉換和指標是公式屬性。

主題

- [在公式表達式中使用變數](#)
- [在公式表達式中使用常值](#)
- [在公式表達式中使用運算子](#)
- [在公式表達式中使用常數](#)
- [在公式表達式中使用函數](#)
- [公式表達式教學課程](#)

在公式表達式中使用變數

變數代表公式表達式中的 AWS IoT SiteWise 資產屬性。使用變數來輸入運算式中其他資產屬性的值，以便處理來自常數屬性 ([屬性](#))、原始資料串流 ([度量](#)) 和其他公式屬性的資料。

變數可以代表來自相同資產模型或相關聯子資產模型的資產屬性。只有指標公式可以從子資產模型輸入變數。

您可以在 主控台 和 API 中依不同名稱識別變數。

- AWS IoT SiteWise 主控台 – 在表達式中使用資產屬性名稱做為變數。
- AWS IoT SiteWise API (AWS CLI, AWS SDKs) – 使用 [ExpressionVariable](#) 結構定義變數，這需要變數名稱和資產屬性的參考。變數名稱可以包含小寫字母、數字和底線。然後，使用變數名稱來參考表達式中的資產屬性。

變數名稱區分大小寫。

如需詳細資訊，請參閱[定義轉換](#)和[定義指標](#)。

使用變數來參考屬性

變數的值會定義其參考的屬性。AWS IoT SiteWise 提供執行此操作的不同方式。

- 依屬性 ID：您可以指定屬性的唯一 ID (UUID) 來識別它。
- 依名稱：如果 屬性位於相同的資產模型上，您可以在屬性 ID 欄位中指定其名稱。
- 依路徑：變數值可以依其路徑參考屬性。如需詳細資訊，請參閱[使用路徑來參考自訂複合模型屬性](#)。

Note

AWS IoT SiteWise 主控台不支援變數。AWS IoT SiteWise API 會使用它們，包括 AWS Command Line Interface (AWS CLI) 和 AWS SDKs。

您在 的回應中收到的變數 AWS IoT SiteWise 包含該值的完整資訊，包括 ID 和路徑。

不過，當您將變數傳遞至 AWS IoT SiteWise（例如，在「建立」或「更新」呼叫中）時，您只需要指定其中一個。例如，如果您指定路徑，則不需要提供 ID。

在公式表達式中使用常值

AWS IoT SiteWise 支援在表達式和公式中使用常值。文字是代表特定資料類型的固定值。在 中 AWS IoT SiteWise，您可以在公式表達式中定義數字和字串常值。文字可用於各種內容，包括資料轉換、警示條件和視覺化計算。

- ### 數字

使用數字和科學記號來定義整數和變數。您可以使用 [E 表示法](#) 來表達具有科學表示法的數字。

範例：1、2.0、.9、-23.1、7.89e3、3.4E-5

- ### 字串

使用 '（引號）和 "（雙引號）字元來定義字串。開始和結束的引號類型必須相符。若要逸出與您用來宣告字串的引號相符的引號，請包含該引號字元兩次。這是 AWS IoT SiteWise 字串中唯一的逸出字元。

範例：'active'、'inactive'、'{"temp": 52}'、'{"temp": "high"}'

在公式表達式中使用運算子

您可以在公式表達式中使用下列常見運算子。

運算子	描述
+	如果兩個運算元都是數字，則此運算子會新增左側和右側運算元。 如果其中一個運算元是字串，此運算子會將左側和右側運算元串連為字串。例如，表達式會 <code>1 + 2 + " is three"</code> 評估為 <code>"3 is three"</code>。串連字串最多可有 1024 個字元。如果字串超過 1024 個字元，則 AWS IoT SiteWise 不會輸出該運算的資料點。
-	從左運算元中減去右運算元。 您只能將此運算子與數值運算元搭配使用。
/	將左運算元除以右運算元。 您只能將此運算子與數值運算元搭配使用。
*	將左側和右側的運算元相乘。 您只能將此運算子與數值運算元搭配使用。
^	將左側的運算元乘上右側運算元的次方數 (指數)。 您只能將此運算子與數值運算元搭配使用。
%	傳回左運算元除以右運算元的餘數。結果與左側運算元具有相同的符號。此行為與模數操作不同。

運算子	描述
	您只能將此運算子與數值運算元搭配使用。
$x < y$	1 如果 x 小於 y ，則傳回 y ，否則傳回 0。
$x > y$	1 如果 x 大於 y ，則傳回 y ，否則傳回 0。
$x \leq y$	1 如果 x 小於或等於 y ，則傳回 y ，否則傳回 0。
$x \geq y$	1 如果 x 大於或等於 y ，則傳回 y ，否則傳回 0。
$x == y$	1 如果 x 等於 y ，則傳回 y ，否則傳回 0。
$x != y$	1 如果 x 不等於 y ，則傳回 y ，否則傳回 0。
$!x$	1 如果評估 x 為 0(false)，則傳回 1，否則傳回 0。 x 評估為 false，如果： x 是數值運算元，並評估為 0。 x 會評估為空字串。 x 會評估為空陣列。 x 會評估為 None。
$x \text{ and } y$	0 如果評估 x 為 0(false)，則傳回 0。否則，會傳回 x 的評估結果 y。 x 或 y 評估為 false，如果： x 或 y 是數值運算元，且評估為 0。 x 或 y 會評估為空字串。 x 或 y 會評估為空陣列。 x 或 y 會評估為 None。

運算子	描述
<code>x or y</code>	1 如果評估<code>x</code>為 1(true)，則傳回 。否則， 會傳回的評估結果<code>y</code>。 <code>x</code> 或 <code>y</code> 評估為 false，如果： • <code>x</code> 或 <code>y</code>是數值運算元，且評估為 0。 • <code>x</code> 或 <code>y</code> 會評估為空字串。 • <code>x</code> 或 <code>y</code> 會評估為空陣列。 • <code>x</code> 或 <code>y</code> 會評估為 None。
<code>not x</code>	1 如果評估<code>x</code>為 0(false)，則傳回 ，否則傳回 0。 <code>x</code> 評估為 false，如果： • <code>x</code> 是數值運算元，並評估為 0。 • <code>x</code> 會評估為空字串。 • <code>x</code> 會評估為空陣列。 • <code>x</code> 會評估為 None。
<code>[]</code> <code>s[index]</code>	傳回字串 <code>index</code> 索引的字元<code>s</code>。這相當於 Python 中的索引語法。 Example 範例 • <code>"Hello!"[1]</code> 傳回：e • <code>"Hello!"[-2]</code> 傳回：o

運算子	描述
[] s[start:end:step]	傳回字串 的配量s。這相當於 Python 中的配量語法。此運算子具有下列引數： • start – (選用) 配量的包含開始索引。預設為 0。 • end – (選用) 配量的專屬結束索引。預設為字串的長度。 • step – (選用) 配量中每個步驟要遞增的數字。例如，您可以指定 2 傳回具有任何其他字元的配量，或指定 -1 來反轉配量。預設為 1。 您可以省略引step數以使用其預設值。例如，s[1:4:1] 等同於 s[1:4]。 引數必須是整數或無常數。如果您指定 none，AWS IoT SiteWise 會使用該引數的預設值。 Example 範例 • "Hello!"[1:4] 傳回："ell" • "Hello!"[:2] 傳回："He" • "Hello!"[3:] 傳回："lo!" • "Hello!"[:-4] 傳回："He" • "Hello!"[::2] 傳回："Hlo" • "Hello!"[::-1] 傳回："!olleH"

在公式表達式中使用常數

在 中 AWS IoT SiteWise，您可以在表達式和公式中使用常數來表示固定值或預先定義的參數。常數可用於各種內容，例如資料轉換、警示條件或視覺化計算。透過使用常數，您可以簡化表達式，使它們更易於讀取和維護。

您可以在您的表達式中使用下列常見的數學常數。所有常數不區分大小寫。

Note

如果您使用與常數相同的名稱定義變數，則變數會覆寫常數。

常數	描述
pi	數字 pi (π) : 3.141592653589793
e	數字 e : 2.718281828459045
true	相當於數字 1。在 中 AWS IoT SiteWise，布林值會轉換為其數量對等項目。
false	相當於數字 0。在 中 AWS IoT SiteWise，布林值會轉換為其數量對等項目。
none	等同於無值。您可以使用此常數來輸出條件 表達式 的結果。

在公式表達式中使用函數

您可以使用下列函數來操作公式表達式中的資料。

轉換和指標支援不同的 函數。下表指出哪些類型的函數與每種類型的公式屬性相容。

Note

您可以在公式表達式中包含最多 10 個函數。

函數類型	轉換	指標
在公式表達式中使用常見函數	 是	 是

函數類型	轉換	指標
在公式表達式中使用比較函數	 是	 是
在公式表達式中使用條件式函數	 是	 是
在公式表達式中使用字串函數	 是	 是
在公式表達式中使用彙總函數	 否	 是
在公式表達式中使用暫時函數	 是	 是
在公式表達式中使用日期和時間函數	 是	 是

函數語法

您可以使用下列語法來建立函數：

一般語法

使用一般語法時，函數名稱後面接著具有零或多個引數的括號。

function_name(argument1, argument2, argument3, ...)。例如，具有一般語法的函數可能看起來像 `log(x)` 和 `contains(s, substring)`。

統一函數呼叫語法 (UFCS)

UFCS 可讓您在物件導向程式設計中使用 方法呼叫的語法來呼叫函數。使用 UFCS，第一個引數後面接著點 (.)，然後是函數名稱和括號內剩餘的引數（如果有的話）。

argument1.function_name(argument2, argument3, ...)。例如，使用 UFCS 的函數可能看起來像 `x.log()` 和 `s.contains(substring)`。

您也可以使用 UFCS 鏈結後續函數。AWS IoT SiteWise 會使用目前函數的評估結果做為下一個函數的第一個引數。

例如，您可以使用 `message.jp('$$.status').lower().contains('fail')` 而非 `contains(lower(jp(message, '$$.status')), 'fail')`。

如需詳細資訊，請造訪 [D 程式設計語言](#) 網站。

Note

您可以對所有 AWS IoT SiteWise 函數使用 UFCS。

AWS IoT SiteWise 函數不區分大小寫。例如，您可以 `Lower(s)` 交替使用 `lower(s)` 和 `lower(s)`。

在公式表達式中使用常見函數

在 [轉換](#) 和 [指標](#) 中，您可以使用下列函數來計算轉換和指標中的常見數學函數。

函式	描述
<code>abs(x)</code>	傳回 <code>x</code> 的絕對值。
<code>acos(x)</code>	傳回 <code>x</code> 的反餘弦。
<code>asin(x)</code>	傳回 <code>x</code> 的反正弦。

函式	描述
<code>atan(x)</code>	傳回 x 的反正切。
<code>cbrt(x)</code>	傳回 x 的立方根。
<code>ceil(x)</code>	傳回大於 x 的最接近整數。
<code>cos(x)</code>	傳回 x 的餘弦。
<code>cosh(x)</code>	傳回 x 的雙曲餘弦。
<code>cot(x)</code>	傳回 x 的餘切。
<code>exp(x)</code>	傳回乘以 x 次方的 e 。
<code>expm1(x)</code>	傳回 $\exp(x) - 1$ 。使用此函數可更準確地計算的小 $\exp(x) - 1$ 值 x 。
<code>floor(x)</code>	傳回小於 x 的最接近整數。
<code>log(x)</code>	傳回 x 的 $\log_e$ (基底 e)。
<code>log10(x)</code>	傳回 x 的 $\log_{10}$ (基底 10)。
<code>log1p(x)</code>	傳回 $\log(1 + x)$ 。使用此函數可更準確地計算的小 $\log(1 + x)$ 值 x 。
<code>log2(x)</code>	傳回 x 的 $\log_2$ (基底 2)。
<code>pow(x, y)</code>	傳回乘以 y 次方的 x 。這相當於 x^y 。
<code>signum(x)</code>	傳回 x 的正負號 (輸入負值為 -1，輸入零為 0，輸入正值則為 +1)。
<code>sin(x)</code>	傳回 x 的正弦。
<code>sinh(x)</code>	傳回 x 的雙曲正弦。
<code>sqrt(x)</code>	傳回 x 的平方根。

函式	描述
<code>tan(x)</code>	傳回 x 的正切。
<code>tanh(x)</code>	傳回 x 的雙曲正切。

在公式表達式中使用比較函數

在[轉換](#)和[指標](#)中，您可以使用下列比較函數來比較兩個值和輸出 1(true) 或 0(false)。AWS IoT SiteWise compares strings by [Lexicographic order](#)。

函式	描述
<code>gt(x, y)</code>	如果 x 大於 y，則傳回 1，否則傳回 0 ($x > y$)。 如果 x 和 y 是不相容的類型，例如數字和字串，則此函數不會傳回值。
<code>gte(x, y)</code>	如果 x 大於或等於 y，則傳回 1，否則傳回 0 ($x \geq y$)。 AWS IoT SiteWise 如果引數在 的相對容錯範圍內，則 會將引數視為相等$1E-9$。這的行為類似於 Python 中的 isclose 函數。 如果 x 和 y 是不相容的類型，例如數字和字串，則此函數不會傳回值。
<code>eq(x, y)</code>	如果 x 等於 y，則傳回 1，否則傳回 0 ($x == y$)。 AWS IoT SiteWise 如果引數在 的相對容錯範圍內，則 會將引數視為相等$1E-9$。這的行為類似於 Python 中的 isclose 函數。 如果 x 和 y 是不相容的類型，例如數字和字串，則此函數不會傳回值。

函式	描述
<code>lt(x, y)</code>	如果 x 小於 y，則傳回 1，否則傳回 0 ($x < y$)。 如果 x 和 y 是不相容的類型，例如數字和字串，則此函數不會傳回值。
<code>lte(x, y)</code>	如果 x 小於或等於 y，則傳回 1，否則傳回 0 ($x \leq y$)。 AWS IoT SiteWise 如果引數在 的相對容錯範圍內，則 會將引數視為相等$1E-9$。這的行為類似於 Python 中的 isclose 函數。 如果 x 和 y 是不相容的類型，例如數字和字串，則此函數不會傳回值。
<code>isnan(x)</code>	1 如果 x 等於 <code>NaN</code>，則傳回 NaN，否則傳回 0。 如果 x 是字串，則此函數不會傳回值。

在公式表達式中使用條件式函數

在[轉換](#)和[指標](#)中，您可以使用下列函數來檢查條件並傳回不同的結果，無論條件評估為 true 還是 false。

函式	描述
<code>if(condition, result_if_true, result_if_false)</code>	<code>result_if_true</code> 如果條件評估為 true 或 <code>result_if_false</code> 條件評估為 <code>condition</code> 和 傳回 false。 <code>condition</code> 必須是數字。此函數會將 0 和空字串視為 false，並將其他所有項目（包括 NaN）視為 true。布林值轉換為 0(false) 和 1(true)。

函式	描述
	您可以從此函數傳回無常數，以捨棄特定條件的輸出。這表示您可以篩選出不符合條件的資料點。如需詳細資訊，請參閱篩選資料點。 Example 範例 • <code>if(0, x, y)</code> 會傳回變數 <code>y</code>。 • <code>if(5, x, y)</code> 會傳回變數 <code>x</code>。 • <code>if(gt(temp, 300), x, y)</code> <code>x</code> 如果變數 <code>temp</code> 大於 <code>300</code>，則 會傳回變數 <code>300</code>。 • <code>if(gt(temp, 300), temp, none)</code> <code>temp</code> 如果變數大於或等於 <code>300</code>，則傳回變數 <code>300</code>；如果 小於 <code>temp</code>，則傳回 <code>none</code> (無值) <code>300</code>。 我們建議您將 UFCS 用於巢狀條件函數，其中一或多個引數是條件函數。您可以使用 <code>if(condition, result_if_true)</code> 來評估條件 <code>elif(condition, result_if_true, result_if_false)</code>，以及評估其他條件。 例如，您可以使用 <code>if(condition1, result1_if_true).elif(condition2, result2_if_true, result2_if_false)</code> 而非 <code>if(condition1, result1_if_true, if(condition2, result2_if_true, result2_if_false))</code>。 您也可以鏈結其他中繼條件函數。例如，您可以使用 <code>if(condition1, result1_if_true).elif(condition2, result2_if_true).elif(condition3, result3_if_true, result3_i</code>

函式	描述
	f_false) ，而不是巢狀多個if陳述式，例如 <code>if(condition1, result1_if_true, if(condition2, result2_if_true, if(condition3, result3_if_true result3_if_false)))</code> 。  Important 您必須<code>elif(condition, result_if_true, result_if _false)</code> 搭配 UFCS 使用。

在公式表達式中使用字串函數

在[轉換](#)和[指標](#)中，您可以使用下列函數在字串上操作。如需詳細資訊，請參閱[在公式中使用字串](#)。

Important

公式表達式只能輸出雙值或字串值。巢狀運算式可以輸出其他資料類型，例如字串，但整體公式必須評估為數字或字串。您可以使用 [jp 函數](#) 將字串轉換為數字。布林值必須為 1 (true) 或 0 (false)。如需詳細資訊，請參閱[未定義、無限和溢出值](#)。

函數	描述
<code>len(s)</code>	傳回字串 的長度s。
<code>find(s, substring)</code>	傳回字串 substring 中字串的索引s。
<code>contains(s, substring)</code>	1 如果字串s包含字串 ，則傳回 substring ，否則傳回 0。
<code>upper(s)</code>	傳回s大寫形式的字串。
<code>lower(s)</code>	以s小寫形式傳回字串。

函數	描述
<code>jp(s, json_path)</code>	s 使用 JsonPath 運算式評估字串 <code>json_path</code> 並傳回結果。 使用此函數執行下列動作： • 從序列化 JSON 結構擷取值、陣列或物件。 • 將字串轉換為數字。例如，公式 <code>jp('111', '\$')</code> 會以數字 111 傳回。 若要從 JSON 結構擷取字串值並將其傳回為數字，您必須使用多個巢狀 <code>jp</code> 函數。外部 <code>jp</code> 函數會從 JSON 結構擷取字串，而內部 <code>jp</code> 函數會將字串轉換為數字。 字串 <code>json_path</code> 必須包含字串常值。這表示 <code>json_path</code> 不能是評估為字串的表達式。 Example 範例 • <code>jp('{ "status": "active", "value": 15 }', '\$.value')</code> 傳回：15 • <code>jp('{ "measurement": { "reading": 25, "confidence": 0.95 } }', '\$.measurement.reading')</code> 傳回：25 • <code>jp('[2,8,23]', '\$[2]')</code> 傳回：23 • <code>jp('{ "values": [3,6,7] }', '\$.values[1]')</code> 傳回：6 • <code>jp('111', '\$')</code> 傳回：111 • <code>jp(jp('{ "measurement": { "reading": 25, "confidence": "0.95" } }', '\$.measurement.confidence'), '\$')</code> 傳回：0.95

函數	描述
<code>join(s0, s1, s2, s3, ...)</code>	傳回具有分隔符號的串連字串。此函數使用第一個輸入字串做為分隔符號，並將剩餘的輸入字串聯結在一起。這的行為類似於 Java 中的 join(CharSequence delimiter, CharSequence... 元素) 函數。 Example 範例 • <code>join("-", "aa", "bb", "cc")</code> 傳回 <code>aa-bb-cc</code>
<code>format(expression: "format")</code> 或 <code>format("format", expression)</code>	傳回指定格式的字串。此函數會評估 <code>expression</code> 為值，然後以指定的格式傳回值。這的行為類似於 Java 中的 format (字串格式、物件... args) 函數。如需支援格式的詳細資訊，請參閱 Java 平台 Standard Edition 7 API 規格中的 類別格式化器 下的轉換。 Example 範例 • <code>format(100+1: "d")</code> 傳回字串 101。 • <code>format("The result is %d", 100+1)</code> 傳回字串 The result is 101。

函數	描述
<code>f'expression'</code>	傳回串連字串。使用此格式化函數，您可以使用簡單的表達式來串連和格式化字串。這些函數可能包含巢狀運算式。您可以使用 <code>{}</code> (大括號) 插入運算式。這的行為類似於 Python 中的格式化字串常值。 Example 範例 <code>f'abc{1+2: "f"}d'</code> 傳回：<code>abc3.000000d</code> 若要評估此範例表達式，請執行下列動作： <code>format(1+2: "f")</code> 會傳回浮點數 <code>3.000000</code>。 <code>join('', "abc", 1+2, 'd')</code> 傳回字串 <code>abc3.000000d</code>。 您也可以以下列方式撰寫表達式：<code>join('', "abc", format(1+2: "f"), 'd')</code>。

在公式表達式中使用彙總函數

僅在[指標](#)中，您可以使用下列函數來彙總每個時間間隔的輸入值，並計算單一輸出值。彙總函數可以彙總來自關聯資產的資料。

彙總函數引數可以是[變數](#)、[數字常值](#)、[時間函數](#)、巢狀運算式或彙總函數。公式`max(latest(x), latest(y), latest(z))`使用彙總函數做為引數，並傳回 `x`、`y` 和 `z` 屬性的最大目前值。

您可以在彙總函數中使用巢狀運算式。當您使用巢狀運算式時，適用下列規則：

- 每個引數只能有一個變數。

Example

例如，支援 `sum(x/2 )/avg(y^2 ) avg(x*(x-1))` 和。

例如，`min(x/y)` 不支援。

- 每個引數都可以有多層巢狀運算式。

Example

例如，支援 $\text{sum}(\text{avg}(x^2)/2)$ 。

- 不同的引數可以有不同的變數。

Example

例如，支援 $\text{sum}(x/2, y*2)$ 。

Note

- 如果您的表達式包含測量值，AWS IoT SiteWise 會使用目前時間間隔內的最後一個值，讓測量值運算彙總。
- 如果您的表達式包含屬性，AWS IoT SiteWise 會使用屬性的最新值來計算彙總。

函式	描述
$\text{avg}(x_0, \dots, x_n)$	傳回目前時間間隔指定變數值的平均數。 只有在指定的變數在目前時間間隔內至少有一個資料點時，此函數才會輸出資料點。
$\text{sum}(x_0, \dots, x_n)$	傳回目前時間間隔指定變數值的總和。 只有在指定的變數在目前時間間隔內至少有一個資料點時，此函數才會輸出資料點。
$\text{min}(x_0, \dots, x_n)$	傳回目前時間間隔指定變數值的最小值。 只有在指定的變數在目前時間間隔內至少有一個資料點時，此函數才會輸出資料點。
$\text{max}(x_0, \dots, x_n)$	傳回目前時間間隔指定變數值的最大值。 只有在指定的變數在目前時間間隔內至少有一個資料點時，此函數才會輸出資料點。

函式	描述
<code>count(x₀, ..., x_n)</code>	傳回目前時間間隔內給定變數的資料點總數。若要取得有關如何計算符合條件之資料點數目的更多資訊，請參閱 計數符合條件的資料點 。 此函數會計算每個時間間隔的資料點。
<code>stdev(x₀, ..., x_n)</code>	傳回指定變數值在目前時間間隔內的標準差。 只有在指定的變數在目前時間間隔內至少有一個資料點時，此函數才會輸出資料點。

在公式表達式中使用暫時函數

使用暫時函數，根據資料點的時間戳記傳回值。

在指標中使用暫時函數

僅在 [指標](#) 中，您可以使用下列函數，根據資料點的時間戳記傳回值。

暫時函數引數必須是來自本機資產模型或巢狀運算式的屬性。這表示您無法在暫時函數中使用子資產模型的屬性。

您可以在時間函數中使用巢狀運算式。當您使用巢狀運算式時，適用下列規則：

- 每個引數只能有一個變數。

例如，支援 `latest( t*9/5 + 32 )`。

- 引數不能是彙總函數。

例如，`first( sum(x) )` 不支援。

函式	描述
<code>first(x)</code>	傳回目前時間間隔指定變數值中時間戳記最早的值。

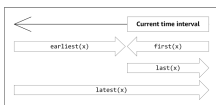
函式	描述
<code>last(x)</code> <code>earliest(x)</code>	傳回目前時間間隔指定變數值中時間戳記最晚的值。 傳回指定變數在目前時間間隔開始前的最後一個值。 如果輸入屬性在其歷史記錄中至少有一個資料點，此函數會計算每個時間間隔的資料點。如需詳細資訊，請參閱 time-range-defintion。
<code>latest(x)</code>	傳回指定變數的最後一個值，並在目前時間間隔結束前加上最新的時間戳記。 如果輸入屬性在其歷史記錄中至少有一個資料點，此函數會計算每個時間間隔的資料點。如需詳細資訊，請參閱 time-range-defintion。
<code>statetime(x)</code>	傳回目前時間間隔中指定變數為正 (positive) 的秒數。您可以使用比較函數來建立轉換屬性，以供statetime 函數使用。 例如，如果您有一個為 0 或 1 的 Idle 屬性，您可以使用此表達式來計算每個時間間隔的閒置時間：<code>IdleTime = statetime(Idle)</code>。 如需詳細資訊，請參閱 example statetime scenario (狀態時間案例範例)。 此函數不支援使用指標屬性做為輸入變數。 如果輸入屬性在其歷史記錄中至少有一個資料點，此函數會計算每個時間間隔的資料點。

函式	描述
<code>TimeWeightedAvg(x, [interpolation])</code>	傳回以點之間的時間間隔加權的輸入資料平均值。 如需運算和間隔詳細資訊，請參閱時間加權函數參數。 選用引數<code>interpolation</code> 必須是字串常數： • <code>locf</code> – 這是預設值。計算會針對資料點之間的時間間隔使用最後觀察的轉移運算演算法。在此方法中，資料點會計算為最後一個觀察值，直到下一個輸入資料點時間戳記為止。 良好資料點之後的值會推斷為其值，直到下一個資料點時間戳記為止。 • <code>linear</code> – 計算使用線性插補運算演算法來計算資料點之間的時間間隔。 兩個良好資料點之間的值會推斷為這些資料點值之間的線性插補。 良好和不良資料點之間的值，或最後一個良好資料點之後的值，將會推斷為良好資料點。

函式	描述
TimeWeightedStDev(x, [algo])	傳回以點之間的時間間隔加權的輸入資料標準差。 如需運算和間隔詳細資訊，請參閱時間加權函數參數。 計算會針對資料點之間的時間間隔使用最後觀察的轉移運算演算法。在此方法中，資料點會計算為最後一個觀察值，直到下一個輸入資料點時間戳記為止。權重的計算方式為資料點或視窗邊界之間的時間間隔，以秒為單位。 選用引數algo必須是字串常數： - f – 這是預設值。它會傳回具有頻率權重的無偏差加權範例變異數，其中 TimeWeight 以秒為單位計算。此演算法通常採用標準差，稱為貝塞爾校正加權樣本的標準差。 - p – 傳回偏差加權範例變異數，也稱為母體變異數。 下列公式用於運算，其中： S_p = 母群體標準差 S_f = 頻率標準差 X_i = 傳入資料 ω_i = 等於時間間隔的權重，以秒為單位 μ^* = 傳入資料的加權平均值 母群體標準差的方程式： $S_p^2 = \frac{\sum_{i=1}^N \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^N \omega_i}$

函式	描述
	頻率標準差的方程式： $S_f^2 = \frac{\sum_{i=1}^N \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^N \omega_i - 1}$

下圖顯示 如何計算 AWS IoT SiteWise 相對於目前時間間隔的時間函數 latest、first、lastearliest和。



Note

- first(x)的時間範圍last(x)為【目前時段開始、目前時段結束】。
- 的時間範圍latest(x)是【時間開始，目前時段結束】。
- 的時間範圍earliest(x)是【時間開始，上一個時段結束】。

時間加權函數參數

針對彙總時段計算的時間加權函數會考量下列事項：

- 視窗內的資料點
- 資料點之間的時間間隔
- 視窗之前的最後一個資料點
- 視窗後的第一個資料點（適用於某些演算法）

術語：

- 資料點錯誤 – 任何品質不佳或非數值的資料點。這不會在視窗結果運算中考慮。
- 錯誤的間隔 – 資料點錯誤後的間隔。第一個已知資料點之前的間隔也會被視為錯誤的間隔。
- 良好的資料點 – 具有良好品質和數值的任何資料點。

Note

- AWS IoT SiteWise 只會在運算轉換和指標時耗用GOOD品質資料。它會忽略 UNCERTAIN和BAD 資料點。
- 第一個已知資料點之前的間隔會被視為錯誤的間隔。如需詳細資訊，請參閱[the section called “公式表達式教學課程”](#)。

最後一個已知資料點之後的間隔會無限期持續，影響下列所有時段。當新的資料點到達時，函數會重新計算間隔。

遵循上述規則，會計算彙總視窗結果，並限於視窗邊界。根據預設，函數只會在整個時段為良好間隔時傳送時段結果。

如果視窗良好間隔小於視窗長度，則函數不會傳送視窗。

當影響視窗結果的資料點變更時，即使資料點位於視窗之外，函數也會重新計算視窗。

如果輸入屬性在其歷史記錄中至少有一個資料點，且已啟動運算，則函數會計算每個時間間隔的時間加權彙總函數。

Example 狀態時間案例範例

假設您具有以下屬性的資產：

- Idle – 為 0 或 1 的測量。值為 1 時，機器處於閒置狀態。
- Idle Time – 使用公式 `statetime(Idle)` 計算機器閒置時間的指標，以秒為單位，間隔為 1 分鐘。

該 Idle 屬性具有以下資料點。

Timestamp	2:00:00 PM	2:00:30 PM	2:01:15 PM	2:02:45 PM	2:04:00 PM
Idle	0	1	1	0	0

AWS IoT SiteWise 每分鐘會從 0 或 1 的值計算 Idle Time 屬性 Idle。此計算完成之後，Idle Time 屬性會具有下列資料點。

Timestamp	2:00:00 PM	2:01:00 PM	2:02:00 PM	2:03:00 PM	2:04:00 PM
Idle Time	N/A	30	60	45	0

AWS IoT SiteWise 會在每分鐘 Idle Time 結束時對 執行下列計算。

- 下午 2:00 (下午 1:59 至下午 2:00)
 - 下午 2:00 之前 Idle 沒有資料，因此不會計算任何資料點。
- 下午 2:01 (下午 2:00 至下午 2:01)
 - 下午 2:00:00，機器處於作用中狀態 (Idle 為 0)。
 - 下午 2:00:30，機器處於閒置狀態 (Idle 為 1)。
 - 在下午 2:01:00 的間隔結束之前 Idle 不會再次改變，所以 Idle Time 是 30 秒。
- 下午 2:02 (下午 2:01 至下午 2:02)
 - 下午 2:01:00，機器處於閒置狀態 (即最後一個資料點位於下午 2:00:30)。
 - 下午 2:01:15，機器仍處於閒置狀態。
 - 在下午 2:02:00 的間隔結束之前 Idle 不會再次改變，所以 Idle Time 是 60 秒。
- 下午 2:03 (下午 2:02 至下午 2:03)
 - 下午 2:02:00，機器處於閒置狀態 (即最後一個資料點位於下午 2:01:15)。
 - 下午 2:02:45，機器處於作用中狀態。
 - 在下午 2:03:00 的間隔結束之前 Idle 不會再次改變，所以 Idle Time 是 45 秒。
- 下午 2:04 (下午 2:03 至下午 2:04)
 - 下午 2:03:00，機器處於作用中狀態 (即最後一個資料點位於下午 2:02:45)。
 - 在下午 2:04:00 的間隔結束之前 Idle 不會再次改變，所以 Idle Time 是 0 秒。

Example TimeWeightedAvg 和 TimeWeightedStDev 案例範例

下表提供這些一分鐘時段指標的範例輸入和輸出：Avg(x)，TimeWeightedAvg(x)，TimeWeightedAvg(x, "linear")，stDev(x)，timeWeightedStDev(x)，timeWeightedStDev(x, 'p')。

一分鐘彙總時段的範例輸入：

Note

這些資料點都具有GOOD品質。

03 : 00 : 00	4.0
03 : 01 : 00	2.0
03 : 01 : 10	8.0
03 : 01 : 50	20.0
03 : 02 : 00	14.0
03 : 02 : 05	10.0
03 : 02 : 10	3.0
03 : 02 : 30	20.0
03 : 03 : 30	0.0

彙總結果輸出：

Note

無 – 未為此時段產生結果。

時間	Avg(x)	TimeWeightedAvg(x)	TimeWeightedAvg(X, "linear")	stDev(X)	timeWeightedStDev(x)	timeWeightedStDev(x, 'p')
3 : 00 : 00	4	無	無	0	無	無
3 : 01 : 00	2	4	3	0	0	0

時間	Avg(x)	TimeWeightedAvg(x)	TimeWeightedAvg(X, "linear")	stDev(X)	timeWeightedStDev(x)	timeWeightedStDev(x, 'p')
3 : 02 : 00	14	9	13	6	5.430610041581775	5.385164807134504
3 : 03 : 00	11	13	12.875	8.54400374531753	7.724054437220943	7.659416862050705
3 : 04 : 00	0	10	2.5	0	10.084389681792215	10
3 : 05 : 00	無	0	0	無	0	0

在轉換中使用暫時函數

僅在[轉換](#)中，您可以使用 `pretrigger()` 函數，在啟動目前轉換計算的屬性更新之前擷取變數GOOD的品質值。

請考慮製造商用來 AWS IoT SiteWise 監控機器狀態的範例。製造商使用下列測量和轉換來代表程序：

- 測量 `current_state`，可以是 0 或 1。
 - 如果機器處於清除狀態，則 `current_state` 等於 1。
 - 如果機器處於製造狀態，則 `current_state` 等於 0。
- 等於 `cleaning_state_duration` 的轉換 `if(pretrigger(current_state) == 1, timestamp(current_state) - timestamp(pretrigger(current_state)), none)`。此轉換會以秒為單位傳回機器處於清除狀態的時間，格式為 Unix epoch。如需詳細資訊，請參閱 [在公式表達式中使用條件式函數](#) 和 [timestamp\(\)](#) 函數。

如果機器停留在清除狀態的時間超過預期，製造商可能會調查機器。

您也可以在多變量轉換中使用 `pretrigger()` 函數。例如，您有兩個名為 `x` 和 `z` 的測量 `y`，以及一個等於 `x + y + pretrigger(y)` 的轉換。下表顯示 `x`、`y` 和 `z` 的值，從上午 9 : 00 到上午 9 : 15。

Note

- 此範例假設測量的值依時間順序到達。例如，上午 09：00 x 的值會在上午 09：05 x 的值之前到達。
- 如果上午 9：05 的資料點在上午 9：00 的資料點到達，z 則不會在上午 9：05 計算。
- 如果上午 9：05 x 的值在上午 09：00 x 的值之前到達，且依時間順序 y 抵達的值，則 z 等於上午 22 = 20 + 1 + 1 9：05。

	上午 09：00	上午 09：05	上午 09：10	上午 09：15
x	10	20		30
y	1	2	3	
$z = x + y + \text{pretrigger}(y)$	y 在上午 09：00 之前不會收到任何資料點。因此，z 不會在上午 09：00 計算。	$23 = 20 + 2 + 1$ pretrigger(y) 等於 1。	$25 = 20 + 3 + 2$ x 不會收到新的資料點。pretrigger(y) 等於 2。	$36 = 30 + 3 + 3$ y 不會收到新的資料點。因此，pretrigger(y) 等於上午 09：15 的 3。

在公式表達式中使用日期和時間函數

在[轉換](#)和[指標](#)中，您可以透過下列方式使用日期和時間函數：

- 擷取 UTC 或本機時區中資料點的目前時間戳記。
- 使用引數建構時間戳記，例如 year、month 和 day_of_month。
- 使用 unix_time 引數擷取期間，例如年份或月份。

函式	描述
now()	以秒為單位，以 Unix epoch 格式傳回目前的日期和時間。

函式	描述
timestamp()	- 在轉換中，函數會以秒為單位傳回 Unix epoch 格式輸入訊息的時間戳記。 僅在轉換中，您可以執行下列其中一項操作： - 提供變數做為函數的引數。timestamp (<i>variable-name</i>) 函數會以秒為單位傳回 Unix epoch 格式指定變數的最新 GOOD品質值的時間戳記。 例如，如果您的資產有一個名為的轉換屬性Temperature_F，使用 $9/5 * \text{Temperature\_C}$ 公式將每個溫度資料點從攝氏轉換為華氏，您可以使用 timestamp(Temperature_F) 函數來取得 Temperature_F 屬性最新GOOD品質值的時間戳記。 - 使用 pretrigger() 函數做為函數的引數。timestamp(pretrigger(<i>variable-name</i>)) 函數會在以 Unix epoch 格式啟動目前轉換計算的屬性更新之前，傳回指定變數GOOD品質值的時間戳記，以秒為單位。如需詳細資訊，請參閱在轉換中使用暫時函數。 - 在指標中，函數會以秒為單位，傳回在目前視窗結束時以 Unix epoch 格式擷取的時間戳記。

函式	描述
<code>mktime(time_zone, year, month, day_of_month, hour, minute, second)</code>	以秒為單位，以 Unix epoch 格式傳回輸入時間。 下列需求適用於使用此函數： - 時區引數必須是引號字串 ('UTC')。如果未指定，則預設時區為 UTC。 時區引數可以是第一個或最後一個引數。 年、月、月、日、小時、分鐘和第二個引數必須依序排列。 - 年份、月份和日期引數是必要的。 下列限制適用於使用此函數： - year - 有效值介於 1970 到 2250 之間。 - month - 有效值介於 1 到 12 之間。 - day-of-month - 有效值介於 1 到 31 之間。 - hour - 有效值介於 0 到 23 之間。 - minute - 有效值介於 0 到 59 之間。 - second - 有效值介於 0 到 60 之間。它可以是浮點數。 範例： <code>mktime(2020, 2, 29)</code> <code>mktime('UTC+3', 2021, 12, 31, 22)</code> <code>mktime(2022, 10, 13, 2, 55, 13.68, 'PST')</code>

函式	描述
<code>localtime(unix_time, time_zone)</code>	傳回從 Unix 時間起算指定時區中的年份、日期、星期幾、星期幾、小時、分鐘或秒。 下列需求適用於使用此函數： - 時區引數必須是引號字串 ('UTC')。如果未指定，則預設時區為 UTC。 - Unix 時間引數是 Unix epoch 格式的時間，以秒為單位。有效範圍介於 1 與 315568898 64403199 之間。它可以是浮點數。 回應範例：2007-12-03T10:15:30+01:00[Europe/Paris] <code>localtime(unix_time, time_zone)</code> 不是獨立的函數。<code>year()</code>、<code>mon()</code>、<code>mday</code>、<code>wday()</code>、<code>yday()</code>、<code>hour()</code>、<code>minute()</code>和 <code>sec()</code>函數以引數<code>localtime(unix_time, time_zone)</code> 表示。 範例： <code>year(localtime('GMT', 1605898608.8113723))</code> <code>now().localtime().year()</code> <code>timestamp().localtime('PST').year()</code> <code>localtime(1605289736, 'Europe/London').year()</code>
<code>year(localtime(unix_time, time_zone))</code>	從 傳回年份 <code>localtime(unix_time, time_zone)</code> 。
<code>mon(localtime(unix_time, time_zone))</code>	從 傳回月份 <code>localtime(unix_time, time_zone)</code> 。

函式	描述
<code>mday(localtime(unix_time, time_zone))</code>	從 傳回當月的日期 <code>localtime(unix_time, time_zone)</code> 。
<code>wday(localtime(unix_time, time_zone))</code>	從 傳回星期幾 <code>localtime(unix_time, time_zone)</code> 。
<code>yday(localtime(unix_time, time_zone))</code>	從 傳回一年中的日期 <code>localtime(unix_time, time_zone)</code> 。
<code>hour(localtime(unix_time, time_zone))</code>	從 傳回小時 <code>localtime(unix_time, time_zone)</code> 。
<code>minute(localtime(unix_time, time_zone))</code>	從 傳回分鐘 <code>localtime(unix_time, time_zone)</code> 。
<code>sec(localtime(unix_time, time_zone))</code>	從 傳回第二個 <code>localtime(unix_time, time_zone)</code> 。

支援的時區格式

您可以透過下列方式指定時區引數：

- 時區位移 - 'Z' 指定 UTC 或位移 ('+2' 或 '-5')。
- 位移 IDs - 結合時區縮寫和位移。例如，'GMT+2' 和 'UTC-01:00'。時區縮寫只能包含三個字母。
- 區域型 IDs - 例如 'Etc/GMT+12' 和 'Pacific/Pago_Pago'。

支援的時區縮寫

日期和時間函數支援下列三個字母的時區縮寫：

- EST - -05 : 00
- HST - -10 : 00
- MST - -07 : 00
- ACT - 澳洲/達爾文

- AET - 澳洲/雪梨
- AGT - 美洲/阿根廷/布宜諾斯_Aires
- ART - 非洲/開羅
- AST - 美洲/安克拉治
- BET - 美洲/聖保羅
- BST - 亞洲/達卡
- CAT - 非洲/哈拉雷
- CET - 歐洲/巴黎
- CNT - 美洲/聖約翰
- CST - 美洲/芝加哥
- CTT - 亞洲/上海
- EAT - 非洲/Addis_Ababa
- IET - America/Indiana/Indianapolis
- IST - 亞洲/加爾各答
- JST - 亞洲/東京
- MIT - 太平洋/非洲
- NET - 亞洲/葉文
- NST - 太平洋/奧克蘭
- PLT - 亞洲/喀拉蚩
- PRT - 美洲/波多黎各
- PST - 美洲/Los_Angeles
- SST - 太平洋/瓜達爾卡納爾
- VST - 亞洲/Ho_Chi_Minh

支援的區域型 IDs

日期和時間函數支援下列區域型 IDs，依其與 UTC+00 : 00 的關係進行組織：

- Etc/GMT+12 (UTC-12 : 00)
- Pacific/Pago_Pago (UTC-11 : 00)

- Pacific/Samoa (UTC-11 : 00)
- Pacific/Niue (UTC-11 : 00)
- 美國/薩摩亞 (UTC-11 : 00)
- Etc/GMT+11 (UTC-11 : 00)
- Pacific/Midway (UTC-11 : 00)
- Pacific/Honolulu (UTC-10 : 00)
- 太平洋/羅克頓加 (UTC-10 : 00)
- 太平洋/大溪地 (UTC-10 : 00)
- Pacific/Johnston (UTC-10 : 00)
- 美國/夏威夷 (UTC-10 : 00)
- SystemV/HST10 (UTC-10 : 00)
- Etc/GMT+10 (UTC-10 : 00)
- Pacific/Marquesas (UTC-09 : 30)
- Etc/GMT+9 (UTC-09 : 00)
- Pacific/Gambier (UTC-09 : 00)
- America/Atka (UTC-09 : 00)
- SystemV/YST9 (UTC-09 : 00)
- America/Adak (UTC-09 : 00)
- 美國/阿留申文 (UTC-09 : 00)
- Etc/GMT+8 (UTC-08 : 00)
- 美國/阿拉斯加 (UTC-08 : 00)
- America/Juneau (UTC-08 : 00)
- America/Metlakatla (UTC-08 : 00)
- America/Yakutat (UTC-08 : 00)
- Pacific/Pitcairn (UTC-08 : 00)
- America/Sitka (UTC-08 : 00)
- 美洲/安克拉治 (UTC-08 : 00)
- SystemV/PST8 (UTC-08 : 00)

- 美洲/北美洲 (UTC-08 : 00)
- SystemV/YST9YDT (UTC-08 : 00)
- 加拿大/伊克隆 (UTC-07 : 00)
- 美國/太平洋-新 (UTC-07 : 00)
- Etc/GMT+7 (UTC-07 : 00)
- 美國/亞利桑那州 (UTC-07 : 00)
- America/Dawson_Creek (UTC-07 : 00)
- 加拿大/太平洋 (UTC-07 : 00)
- PST8PDT (UTC-07 : 00)
- SystemV/MST7 (UTC-07 : 00)
- America/Dawson (UTC-07 : 00)
- 墨西哥/BajaNorte (UTC-07 : 00)
- 美洲/提華納 (UTC-07 : 00)
- America/Creston (UTC-07 : 00)
- America/Hermosillo (UTC-07 : 00)
- America/Santa_Isabel (UTC-07 : 00)
- America/Vancouver (UTC-07 : 00)
- America/Ensenada (UTC-07 : 00)
- 美洲/鳳凰城 (UTC-07 : 00)
- 美洲/白馬 (UTC-07 : 00)
- America/Fort_Nelson (UTC-07 : 00)
- SystemV/PST8PDT (UTC-07 : 00)
- America/Los_Angeles (UTC-07 : 00)
- 美國/太平洋 (UTC-07 : 00)
- America/El_Salvador (UTC-06 : 00)
- 美洲/瓜地馬拉 (UTC-06 : 00)
- 美洲/貝里斯 (UTC-06 : 00)
- 美洲/馬拿瓜 (UTC-06 : 00)
- 美洲/德古西加巴 (UTC-06 : 00)

- Etc/GMT+6 (UTC-06 : 00)
- Pacific/Easter (UTC-06 : 00)
- 墨西哥/BajaSur (UTC-06 : 00)
- 美洲/區域 (UTC-06 : 00)
- 美洲/丹佛 (UTC-06 : 00)
- 太平洋/加拉巴哥 (UTC-06 : 00)
- America/Yellowknife (UTC-06 : 00)
- America/Swift_Current (UTC-06 : 00)
- America/Inuvik (UTC-06 : 00)
- 美洲/馬薩特蘭 (UTC-06 : 00)
- America/Boise (UTC-06 : 00)
- America/Costa_Rica (UTC-06 : 00)
- MST7MDT (UTC-06 : 00)
- SystemV/CST6 (UTC-06 : 00)
- 美洲/吉瓦瓦 (UTC-06 : 00)
- America/Ojinaga (UTC-06 : 00)
- 智利/EasterIsland (UTC-06 : 00)
- US/Mountain (UTC-06 : 00)
- America/Edmonton (UTC-06 : 00)
- 加拿大/山區 (UTC-06 : 00)
- America/Cambridge_Bay (UTC-06 : 00)
- Navajo (UTC-06 : 00)
- SystemV/MST7MDT (UTC-06 : 00)
- 加拿大/薩克其萬文 (UTC-06 : 00)
- 美洲/船舶 (UTC-06 : 00)
- 美洲/巴拿馬 (UTC-05 : 00)
- 美洲/芝加哥 (UTC-05 : 00)
- America/Eirunepe (UTC-05 : 00)
- Etc/GMT+5 (UTC-05 : 00)

- 墨西哥/一般 (UTC-05 : 00)
- America/Porto_Acre (UTC-05 : 00)
- 美洲/蓋亞基爾 (UTC-05 : 00)
- America/Rankin_Inlet (UTC-05 : 00)
- 美國/中部 (UTC-05 : 00)
- America/Rainy_River (UTC-05 : 00)
- America/Indiana/Knox (UTC-05 : 00)
- America/North_Dakota/Beulah (UTC-05 : 00)
- America/Monterrey (UTC-05 : 00)
- 美洲/牙買加 (UTC-05 : 00)
- America/Atikokan (UTC-05 : 00)
- America/Coral_Harbour (UTC-05 : 00)
- America/North_Dakota/Center (UTC-05 : 00)
- 美洲/開曼島 (UTC-05 : 00)
- America/Indiana/Tell_City (UTC-05 : 00)
- 美洲/墨西哥_城市 (UTC-05 : 00)
- America/Matamoros (UTC-05 : 00)
- CST6CDT (UTC-05 : 00)
- America/Knox_IN (UTC-05 : 00)
- 美洲/波哥大 (UTC-05 : 00)
- America/Menominee (UTC-05 : 00)
- America/Resolute (UTC-05 : 00)
- SystemV/EST5 (UTC-05 : 00)
- 加拿大/中部 (UTC-05 : 00)
- 巴西/希臘 (UTC-05 : 00)
- America/Cancun (UTC-05 : 00)
- 美洲/利馬 (UTC-05 : 00)
- America/Bahia_Banderas (UTC-05 : 00)
- US/Indiana-Starke (UTC-05 : 00)

- America/Rio_Branco (UTC-05 : 00)
- SystemV/CST6CDT (UTC-05 : 00)
- 牙買加 (UTC-05 : 00)
- 美洲/米蘭 (UTC-05 : 00)
- America/North_Dakota/New_Salem (UTC-05 : 00)
- 美洲/溫尼伯 (UTC-05 : 00)
- America/Cuiaba (UTC-04 : 00)
- America/Marigot (UTC-04 : 00)
- America/Indiana/Petersburg (UTC-04 : 00)
- 智利/洲 (UTC-04 : 00)
- America/Grand_Turk (UTC-04 : 00)
- 古巴 (UTC-04 : 00)
- Etc/GMT+4 (UTC-04 : 00)
- 美洲/馬納斯 (UTC-04 : 00)
- America/Fort_Wayne (UTC-04 : 00)
- America/St_Thomas (UTC-04 : 00)
- 美洲/安圭拉 (UTC-04 : 00)
- 美洲/哈瓦那 (UTC-04 : 00)
- 美國/密西根州 (UTC-04 : 00)
- 美洲/巴貝多 (UTC-04 : 00)
- 美洲/路易斯維爾 (UTC-04 : 00)
- 美洲/庫拉索 (UTC-04 : 00)
- 美洲/蓋亞那 (UTC-04 : 00)
- America/Martinique (UTC-04 : 00)
- America/Puerto_Rico (UTC-04 : 00)
- America/Port_of_Spain (UTC-04 : 00)
- SystemV/AST4 (UTC-04 : 00)
- America/Indiana/Vevay (UTC-04 : 00)
- America/Indiana/Vincennes(UTC-04 : 00)

- America/Kralendijk (UTC-04 : 00)
- 美洲/安提瓜 (UTC-04 : 00)
- 美洲/印地安那波利斯 (UTC-04 : 00)
- America/Iqaluit (UTC-04 : 00)
- America/St_Vincent (UTC-04 : 00)
- America/Kentucky/Louisville (UTC-04 : 00)
- 美洲/多明尼加 (UTC-04 : 00)
- America/Asuncion (UTC-04 : 00)
- EST5EDT (UTC-04 : 00)
- 美洲/拿索 (UTC-04 : 00)
- America/Kentucky/Monticello (UTC-04 : 00)
- 巴西/西部 (UTC-04 : 00)
- 美洲/阿魯巴 (UTC-04 : 00)
- America/Indiana/Indianapolis (UTC-04 : 00)
- 美洲/聖地牙哥 (UTC-04 : 00)
- America/La_Paz (UTC-04 : 00)
- America/Thunder_Bay (UTC-04 : 00)
- America/Indiana/Marengo (UTC-04 : 00)
- America/Blanc-Sablon (UTC-04 : 00)
- America/Santo_Domingo (UTC-04 : 00)
- 美國/東部 (UTC-04 : 00)
- 加拿大/東部 (UTC-04 : 00)
- America/Port-au-Prince (UTC-04 : 00)
- America/St_Barthelemy (UTC-04 : 00)
- America/Nipigon (UTC-04 : 00)
- 美國/印度東部 (UTC-04 : 00)
- America/St_Lucia (UTC-04 : 00)
- 美洲/蒙特塞拉特 (UTC-04 : 00)
- America/Lower_Princes (UTC-04 : 00)

- America/Detroit (UTC-04 : 00)
- America/Tortola (UTC-04 : 00)
- America/Porto_Velho (UTC-04 : 00)
- America/Campo_Grande (UTC-04 : 00)
- 美洲/維吉尼亞 (UTC-04 : 00)
- America/Pangnirtung (UTC-04 : 00)
- 美洲/蒙特婁 (UTC-04 : 00)
- America/Indiana/Winamac (UTC-04 : 00)
- America/Boa_Vista (UTC-04 : 00)
- 美洲/格瑞那達 (UTC-04 : 00)
- America/New_York (UTC-04 : 00)
- America/St_Kitts (UTC-04 : 00)
- 美洲/卡拉卡斯 (UTC-04 : 00)
- America/Guadeloupe (UTC-04 : 00)
- 美洲/多倫多 (UTC-04 : 00)
- SystemV/EST5EDT (UTC-04 : 00)
- America/Argentina/Catamarca (UTC-03 : 00)
- 加拿大/大西洋 (UTC-03 : 00)
- America/Argentina/Cordoba (UTC-03 : 00)
- 美洲/阿拉圭 (UTC-03 : 00)
- America/Argentina/Salta (UTC-03 : 00)
- Etc/GMT+3 (UTC-03 : 00)
- America/Montevideo (UTC-03 : 00)
- 巴西/東部 (UTC-03 : 00)
- America/Argentina/Mendoza (UTC-03 : 00)
- America/Argentina/Rio_Gallegos (UTC-03 : 00)
- America/Catamarca (UTC-03 : 00)
- America/Cordoba (UTC-03 : 00)
- America/Sao_Paulo (UTC-03 : 00)

- America/Argentina/Jujuy (UTC-03 : 00)
- America/Cayenne (UTC-03 : 00)
- America/Recife (UTC-03 : 00)
- America/Buenos_Aires (UTC-03 : 00)
- America/Paramaribo (UTC-03 : 00)
- America/Moncton (UTC-03 : 00)
- 美洲/門多薩 (UTC-03 : 00)
- America/Santarem (UTC-03 : 00)
- Atlantic/Bermuda (UTC-03 : 00)
- America/Maceio (UTC-03 : 00)
- 大西洋/斯坦利 (UTC-03 : 00)
- America/Halifax (UTC-03 : 00)
- 南極/盧比 (UTC-03 : 00)
- America/Argentina/San_Luis (UTC-03 : 00)
- America/Argentina/Ushuaia(UTC-03 : 00)
- 南極/Palmer (UTC-03 : 00)
- America/Punta_Arenas (UTC-03 : 00)
- America/Glace_Bay (UTC-03 : 00)
- America/Fortaleza (UTC-03 : 00)
- America/Thule (UTC-03 : 00)
- America/Argentina/La_Rioja (UTC-03 : 00)
- America/Belem (UTC-03 : 00)
- America/Jujuy (UTC-03 : 00)
- 美洲/巴亞 (UTC-03 : 00)
- America/Goose_Bay (UTC-03 : 00)
- America/Argentina/San_Juan (UTC-03 : 00)
- America/Argentina/ComodRivadavia (UTC-03 : 00)
- America/Argentina/Tucuman (UTC-03 : 00)
- America/Rosario (UTC-03 : 00)

- SystemV/AST4ADT (UTC-03 : 00)
- America/Argentina/Buenos_Aires (UTC-03 : 00)
- America/St_Johns (UTC-02 : 30)
- 加拿大/紐芬蘭 (UTC-02 : 30)
- America/Miquelon (UTC-02 : 00)
- Etc/GMT+2 (UTC-02 : 00)
- America/Godthab (UTC-02 : 00)
- America/Noronha (UTC-02 : 00)
- 巴西/DeNoronha (UTC-02 : 00)
- 大西洋/南部_喬治亞 (UTC-02 : 00)
- Etc/GMT+1 (UTC-01 : 00)
- Atlantic/Cape_Verde (UTC-01 : 00)
- Pacific/Kiritimati (UTC+14 : 00)
- Etc/GMT-14 (UTC+14 : 00)
- Pacific/Fakaofu (UTC+13 : 00)
- Pacific/Enderbury (UTC+13 : 00)
- Pacific/Apia (UTC+13 : 00)
- Pacific/Tongatapu (UTC+13 : 00)
- Etc/GMT-13 (UTC+13 : 00)
- NZ-CHAT (UTC+12 : 45)
- Pacific/Chatham (UTC+12 : 45)
- Pacific/Kwajalein (UTC+12 : 00)
- 南極/McMurdo (UTC+12 : 00)
- Pacific/Wallis (UTC+12 : 00)
- Pacific/Fiji (UTC+12 : 00)
- Pacific/Funafuti (UTC+12 : 00)
- Pacific/Nauru (UTC+12 : 00)
- Kwajalein (UTC+12 : 00)
- NZ (UTC+12 : 00)

- Pacific/Wake (UTC+12 : 00)
- 南極/南極 (UTC+12 : 00)
- Pacific/Tarawa (UTC+12 : 00)
- 太平洋/奧克蘭 (UTC+12 : 00)
- Asia/Kamchatka (UTC+12 : 00)
- Etc/GMT-12 (UTC+12 : 00)
- 亞洲/安多爾 (UTC+12 : 00)
- Pacific/Majuro (UTC+12 : 00)
- Pacific/Ponape (UTC+11 : 00)
- Pacific/Bougainville (UTC+11 : 00)
- 南極/Macquarie (UTC+11 : 00)
- Pacific/Pohnpei (UTC+11 : 00)
- Pacific/Efate (UTC+11 : 00)
- Pacific/Norfolk (UTC+11 : 00)
- 亞洲/馬加達文 (UTC+11 : 00)
- Pacific/Kosrae (UTC+11 : 00)
- Asia/Sakhalin (UTC+11 : 00)
- Pacific/Noumea (UTC+11 : 00)
- Etc/GMT-11 (UTC+11 : 00)
- Asia/Srednekolymsk (UTC+11 : 00)
- Pacific/Guadalcanal (UTC+11 : 00)
- Australia/Lord_Howe (UTC+10 : 30)
- 澳洲/LHI (UTC+10 : 30)
- 澳洲/哈巴特 (UTC+10 : 00)
- Pacific/Yap (UTC+10 : 00)
- 澳洲/塔斯馬尼亞 (UTC+10 : 00)
- Pacific/Port_Moresby (UTC+10 : 00)
- 澳洲/ACT (UTC+10 : 00)
- 澳洲/維吉尼亞 (UTC+10 : 00)

- Pacific/Chuuk (UTC+10 : 00)
- 澳洲/芬蘭 (UTC+10 : 00)
- 澳洲/坎培拉 (UTC+10 : 00)
- 澳洲/馬來西亞 (UTC+10 : 00)
- 太平洋/關島 (UTC+10 : 00)
- Pacific/Truk (UTC+10 : 00)
- 澳洲/NSW (UTC+10 : 00)
- Asia/Vladivostok (UTC+10 : 00)
- 太平洋/塞班 (UTC+10 : 00)
- 南極/DumontDUrville (UTC+10 : 00)
- 澳洲/雪梨 (UTC+10 : 00)
- 澳洲/布里斯班 (UTC+10 : 00)
- Etc/GMT-10 (UTC+10 : 00)
- Asia/Ust-Nera (UTC+10 : 00)
- 澳洲/墨爾本 (UTC+10 : 00)
- 澳洲/林德曼 (UTC+10 : 00)
- 澳洲/北部 (UTC+09 : 30)
- 澳洲/雅科文納 (UTC+09 : 30)
- 澳洲/阿德雷德 (UTC+09 : 30)
- Australia/Broken_Hill (UTC+09 : 30)
- 澳洲/南部 (UTC+09 : 30)
- 澳洲/達爾文 (UTC+09 : 30)
- Etc/GMT-9 (UTC+09 : 00)
- Pacific/Palau (UTC+09 : 00)
- 亞洲/中國 (UTC+09 : 00)
- Asia/Dili (UTC+09 : 00)
- 亞洲/日本 (UTC+09 : 00)
- 亞洲/雅各斯克 (UTC+09 : 00)
- Asia/Pyongyang (UTC+09 : 00)

- ROK (UTC+09 : 00)
- 亞洲/首爾 (UTC+09 : 00)
- Asia/Khandyga (UTC+09 : 00)
- 日本 (UTC+09 : 00)
- 亞洲/東京 (UTC+09 : 00)
- Australia/Eucla (UTC+08 : 45)
- Asia/Kuching (UTC+08 : 00)
- 亞洲/北京 (UTC+08 : 00)
- Etc/GMT-8 (UTC+08 : 00)
- 澳洲/伯斯 (UTC+08 : 00)
- 亞洲/澳門 (UTC+08 : 00)
- 亞洲/澳門 (UTC+08 : 00)
- Asia/Choibalsan (UTC+08 : 00)
- 亞洲/上海 (UTC+08 : 00)
- 南極/凱西 (UTC+08 : 00)
- Asia/Ulan_Bator (UTC+08 : 00)
- 亞洲/重慶 (UTC+08 : 00)
- 亞洲/阿拉伯聯合大公國 (UTC+08 : 00)
- 亞洲/台北 (UTC+08 : 00)
- 亞洲/曼尼拉 (UTC+08 : 00)
- PRC (UTC+08 : 00)
- Asia/Ujung_Pandang (UTC+08 : 00)
- 亞洲/哈爾濱 (UTC+08 : 00)
- 新加坡 (UTC+08 : 00)
- 亞洲/布魯伊 (UTC+08 : 00)
- 澳洲/西部 (UTC+08 : 00)
- 亞洲/香港_香港 (UTC+08 : 00)
- 亞洲/馬克薩斯 (UTC+08 : 00)
- 香港 (UTC+08 : 00)

- Asia/Kuala_Lumpur (UTC+08 : 00)
- 亞洲/伊爾庫次克 (UTC+08 : 00)
- 亞洲/新加坡 (UTC+08 : 00)
- Asia/Pontianak (UTC+07 : 00)
- Etc/GMT-7 (UTC+07 : 00)
- Asia/Phnom_Penh (UTC+07 : 00)
- 亞洲/Novosibirsk (UTC+07 : 00)
- 南極/達維斯 (UTC+07 : 00)
- 亞洲/東京 (UTC+07 : 00)
- 亞洲/雅加達 (UTC+07 : 00)
- 亞洲/巴納爾 (UTC+07 : 00)
- 印度/聖誕 (UTC+07 : 00)
- 亞洲/Ho_Chi_Minh (UTC+07 : 00)
- Asia/Hovd (UTC+07 : 00)
- 亞洲/曼谷 (UTC+07 : 00)
- Asia/Vientiane (UTC+07 : 00)
- 亞洲/Novokuznetsk (UTC+07 : 00)
- 亞洲/克拉斯諾亞爾斯克 (UTC+07 : 00)
- Asia/Saigon (UTC+07 : 00)
- 亞洲/仰光 (UTC+06 : 30)
- Asia/Rangoon (UTC+06 : 30)
- 印度/可哥斯 (UTC+06 : 30)
- 亞洲/喀什加爾 (UTC+06 : 00)
- Etc/GMT-6 (UTC+06 : 00)
- Asia/Almaty (UTC+06 : 00)
- 亞洲/達卡 (UTC+06 : 00)
- 亞洲/大阪 (UTC+06 : 00)
- 亞洲/達卡 (UTC+06 : 00)
- 印度/芝加哥 (UTC+06 : 00)

- Asia/Qyzylorda (UTC+06 : 00)
- 亞洲/比什克文 (UTC+06 : 00)
- 南極/維斯托克 (UTC+06 : 00)
- 亞洲/烏魯木齊 (UTC+06 : 00)
- Asia/Thimbu (UTC+06 : 00)
- 亞洲/明湖 (UTC+06 : 00)
- 亞洲/加德滿都 (UTC+05 : 45)
- 亞洲/加德滿都 (UTC+05 : 45)
- 亞洲/加爾各答 (UTC+05 : 30)
- 亞洲/可倫坡 (UTC+05 : 30)
- 亞洲/卡庫塔 (UTC+05 : 30)
- Asia/Aqtau (UTC+05 : 00)
- Etc/GMT-5 (UTC+05 : 00)
- Asia/Samarkand (UTC+05 : 00)
- 亞洲/喀拉蚩 (UTC+05 : 00)
- 亞洲/葉卡特琳堡 (UTC+05 : 00)
- Asia/Dushanbe (UTC+05 : 00)
- 印度/馬爾地夫 (UTC+05 : 00)
- 亞洲/歐羅 (UTC+05 : 00)
- 亞洲/塔什干 (UTC+05 : 00)
- 南極/馬遜 (UTC+05 : 00)
- Asia/Aqtobe (UTC+05 : 00)
- Asia/Ashkhabad (UTC+05 : 00)
- Asia/Ashgabat (UTC+05 : 00)
- Asia/Atyrau (UTC+05 : 00)
- 印度/開加勒文 (UTC+05 : 00)
- 伊朗 (UTC+04 : 30)
- 亞洲/德黑蘭 (UTC+04 : 30)
- 亞洲/喀布爾 (UTC+04 : 30)

- 亞洲/葉文 (UTC+04 : 00)
- Etc/GMT-4 (UTC+04 : 00)
- Etc/GMT-4 (UTC+04 : 00)
- 亞洲/杜拜 (UTC+04 : 00)
- 印度/留尼旺 (UTC+04 : 00)
- 歐洲/薩拉托夫 (UTC+04 : 00)
- 歐洲/薩馬拉 (UTC+04 : 00)
- 印度/馬來西亞 (UTC+04 : 00)
- 亞洲/巴庫 (UTC+04 : 00)
- 亞洲/馬斯卡特 (UTC+04 : 00)
- 歐洲/伏爾加格勒 (UTC+04 : 00)
- 歐洲/阿斯特拉坎 (UTC+04 : 00)
- 亞洲/提比利斯 (UTC+04 : 00)
- 歐洲/阿拉伯聯合大公國斯克 (UTC+04 : 00)
- Asia/Aden (UTC+03 : 00)
- 非洲/奈洛比 (UTC+03 : 00)
- 歐洲/伊斯坦堡 (UTC+03 : 00)
- Etc/GMT-3 (UTC+03 : 00)
- 歐洲/Zaporozhye (UTC+03 : 00)
- 以色列 (UTC+03 : 00)
- Indian/Comoro (UTC+03 : 00)
- 南極/雪瓦 (UTC+03 : 00)
- 非洲/摩加迪休 (UTC+03 : 00)
- 歐洲/布加勒斯特 (UTC+03 : 00)
- 非洲/亞洲 (UTC+03 : 00)
- 歐洲/馬里漢文 (UTC+03 : 00)
- 亞洲/伊斯坦堡 (UTC+03 : 00)
- 歐洲/特拉索波爾 (UTC+03 : 00)
- 歐洲/莫斯科 (UTC+03 : 00)

- 歐洲/中國 (UTC+03 : 00)
- 歐洲/赫爾辛基 (UTC+03 : 00)
- 亞洲/貝魯特 (UTC+03 : 00)
- 亞洲/特拉維夫 (UTC+03 : 00)
- 非洲/吉布地 (UTC+03 : 00)
- 歐洲/塞維多爾 (UTC+03 : 00)
- 歐洲/索菲亞 (UTC+03 : 00)
- 亞洲/加薩 (UTC+03 : 00)
- 非洲/阿斯馬拉 (UTC+03 : 00)
- 歐洲/里加 (UTC+03 : 00)
- 亞洲/巴加達 (UTC+03 : 00)
- 亞洲/大馬士革 (UTC+03 : 00)
- Africa/Dar_es_Salaam (UTC+03 : 00)
- Africa/Addis_Ababa (UTC+03 : 00)
- 歐洲/烏茲哥羅德 (UTC+03 : 00)
- 亞洲/耶路撒冷 (UTC+03 : 00)
- 亞洲/利雅得 (UTC+03 : 00)
- 亞洲/科威特 (UTC+03 : 00)
- 歐洲/基羅夫 (UTC+03 : 00)
- 非洲/坎帕拉 (UTC+03 : 00)
- 歐洲/明斯克 (UTC+03 : 00)
- 亞洲/卡達 (UTC+03 : 00)
- 歐洲/基輔 (UTC+03 : 00)
- 亞洲/巴林 (UTC+03 : 00)
- 歐洲/維爾紐斯 (UTC+03 : 00)
- Indian/Antananarivo (UTC+03 : 00)
- 印度/ Mayotte (UTC+03 : 00)
- 歐洲/塔林 (UTC+03 : 00)
- 土耳其 (UTC+03 : 00)

- 非洲/古巴 (UTC+03 : 00)
- 亞洲/尼科西亞 (UTC+03 : 00)
- 亞洲/法蘭克福 (UTC+03 : 00)
- W-SU (UTC+03 : 00)
- EET (UTC+03 : 00)
- Asia/Hebron (UTC+03 : 00)
- 亞洲/安曼 (UTC+03 : 00)
- 歐洲/尼科西亞 (UTC+03 : 00)
- 歐洲/雅典 (UTC+03 : 00)
- 非洲/開羅 (UTC+02 : 00)
- 非洲/馬巴巴尼 (UTC+02 : 00)
- 歐洲/巴西 (UTC+02 : 00)
- 歐洲/華沙 (UTC+02 : 00)
- CET (UTC+02 : 00)
- 歐洲/盧森堡 (UTC+02 : 00)
- Etc/GMT-2 (UTC+02 : 00)
- 利比亞 (UTC+02 : 00)
- 非洲/基加利 (UTC+02 : 00)
- 非洲/Tripoli (UTC+02 : 00)
- 歐洲/卡寧格勒 (UTC+02 : 00)
- 非洲/溫荷克 (UTC+02 : 00)
- 歐洲/馬爾他 (UTC+02 : 00)
- 歐洲/北京 (UTC+02 : 00)
-
- 歐洲/斯科普耶 (UTC+02 : 00)
- 歐洲/薩拉傑沃 (UTC+02 : 00)
- 歐洲/羅馬 (UTC+02 : 00)
- 歐洲/蘇黎世 (UTC+02 : 00)
- 歐洲/加布羅陀 (UTC+02 : 00)

- 非洲/盧森堡 (UTC+02 : 00)
- 歐洲/維度 (UTC+02 : 00)
- 歐洲/盧比安那 (UTC+02 : 00)
- 歐洲/柏林 (UTC+02 : 00)
- 歐洲/斯德哥爾摩 (UTC+02 : 00)
- 歐洲/布達佩斯 (UTC+02 : 00)
- 歐洲/札格瑞布 (UTC+02 : 00)
- 歐洲/巴黎 (UTC+02 : 00)
- 非洲/捷克 (UTC+02 : 00)
- 歐洲/布拉格 (UTC+02 : 00)
- Antarctica/Troll (UTC+02 : 00)
- 非洲/加布隆 (UTC+02 : 00)
- 歐洲/哥本哈根 (UTC+02 : 00)
- 歐洲/維也納 (UTC+02 : 00)
- 歐洲/土耳其 (UTC+02 : 00)
- MET (UTC+02 : 00)
- 歐洲/阿姆斯特丹 (UTC+02 : 00)
- 非洲/馬普索 (UTC+02 : 00)
- 歐洲/桑_馬利諾 (UTC+02 : 00)
- 波蘭 (UTC+02 : 00)
- 歐洲/安道爾 (UTC+02 : 00)
- 歐洲/奧斯陸 (UTC+02 : 00)
- 歐洲/Podgorica (UTC+02 : 00)
- 非洲/布漢布拉 (UTC+02 : 00)
- Atlantic/Jan_Mayen (UTC+02 : 00)
- 非洲/馬斯魯 (UTC+02 : 00)
- 歐洲/馬德里 (UTC+02 : 00)
- 非洲/布林特爾 (UTC+02 : 00)
- 非洲/俄羅斯 (UTC+02 : 00)

- 非洲/哈拉雷 (UTC+02 : 00)
- 非洲/喀土木 (UTC+02 : 00)
- 非洲/約翰尼斯堡 (UTC+02 : 00)
- 歐洲/貝爾格勒 (UTC+02 : 00)
- 歐洲/布拉迪斯拉瓦 (UTC+02 : 00)
- 北極/長年比 (UTC+02 : 00)
- 埃及 (UTC+02 : 00)
- 歐洲/梵蒂岡 (UTC+02 : 00)
- 歐洲/摩納哥 (UTC+02 : 00)
- 歐洲/倫敦 (UTC+01 : 00)
- Etc/GMT-1 (UTC+01 : 00)
- 歐洲/澤西 (UTC+01 : 00)
- 歐洲/格恩西島 (UTC+01 : 00)
- 歐洲/Isle_of_Man (UTC+01 : 00)
- 非洲/突尼斯 (UTC+01 : 00)
- 非洲/馬拉波文 (UTC+01 : 00)
- GB-Eire (UTC+01 : 00)
- 非洲/拉哥斯 (UTC+01 : 00)
- 非洲/阿爾及爾 (UTC+01 : 00)
- GB (UTC+01 : 00)
- 葡萄牙 (UTC+01 : 00)
- Africa/Sao_Tome (UTC+01 : 00)
- 非洲/Ndjamena (UTC+01 : 00)
- 大西洋/法蘭克福 (UTC+01 : 00)
- 愛爾蘭 (UTC+01 : 00)
- 大西洋/法羅 (UTC+01 : 00)
- 歐洲/都柏林 (UTC+01 : 00)
- 非洲/利伯維爾 (UTC+01 : 00)
- Africa/El_Aaiun (UTC+01 : 00)

- Africa/EI_Aaiun (UTC+01 : 00)
- 非洲/多拉 (UTC+01 : 00)
- 非洲/布拉薩維爾 (UTC+01 : 00)
- 非洲/波多-Novo (UTC+01 : 00)
- 大西洋/馬德拉 (UTC+01 : 00)
- 歐洲/里斯本 (UTC+01 : 00)
- 大西洋/加那利 (UTC+01 : 00)
- 非洲/卡薩布蘭卡 (UTC+01 : 00)
- 歐洲/貝爾法斯特 (UTC+01 : 00)
- 非洲/盧安達 (UTC+01 : 00)
- 非洲/金夏沙 (UTC+01 : 00)
- 非洲/孟買 (UTC+01 : 00)
- WET (UTC+01 : 00)
- 非洲/尼亞米 (UTC+01 : 00)
- GMT (UTC+00 : 00)
- Etc/GMT-0 (UTC+00 : 00)
- Atlantic/St_Helena (UTC+00 : 00)
- Etc/GMT+0 (UTC+00 : 00)
- 非洲/班加爾 (UTC+00 : 00)
- Etc/GMT (UTC+00 : 00)
- 非洲/自由城 (UTC+00 : 00)
- 非洲/巴馬科 (UTC+00 : 00)
- 非洲/庫克里 (UTC+00 : 00)
- 通用 (UTC+00 : 00)
- 非洲/Nouakchott (UTC+00 : 00)
- UTC (UTC+00 : 00)
- Etc/Universal (UTC+00 : 00)
- 大西洋/亞述群島 (UTC+00 : 00)
- 非洲/Abidjan (UTC+00 : 00)

- 非洲/阿克拉 (UTC+00 : 00)
- Etc/UCT (UTC+00 : 00)
- GMT0 (UTC+00 : 00)
- Zulu (UTC+00 : 00)Zulu (UTC+00 : 00)
- 非洲/大阪 (UTC+00 : 00)
- 大西洋/雷克雅維克 (UTC+00 : 00)
- Etc/Zulu (UTC+00 : 00)
- 冰島 (UTC+00 : 00)
- 非洲/羅馬 (UTC+00 : 00)
- Greenwich (UTC+00 : 00)
- Etc/GMT0 (UTC+00 : 00)
- 美洲/丹麥夏文 (UTC+00 : 00)
- 非洲/達卡 (UTC+00 : 00)
- 非洲/比索 (UTC+00 : 00)
- Etc/Greenwich (UTC+00 : 00)
- 非洲/Timbuktu (UTC+00 : 00)
- UCT (UTC+00 : 00)
- 非洲/蒙羅維亞 (UTC+00 : 00)
- Etc/UTC (UTC+00 : 00)

公式表達式教學課程

您可以遵循這些教學課程來使用 中的公式表達式 AWS IoT SiteWise。

主題

- [在公式中使用字串](#)
- [篩選資料點](#)
- [計數符合條件的資料點](#)
- [公式中的延遲資料](#)
- [公式中的資料品質](#)

- [未定義、無限和溢出值](#)

在公式中使用字串

您可以在公式表達式中的字串上操作。您也可以從參考屬性和測量屬性的變數輸入字串。

Important

公式表達式只能輸出雙值或字串值。巢狀運算式可以輸出其他資料類型，例如字串，但整體公式必須評估為數字或字串。您可以使用 [jp 函數](#) 將字串轉換為數字。布林值必須為 1 (true) 或 0 (false)。如需詳細資訊，請參閱[未定義、無限和溢出值](#)。

AWS IoT SiteWise 提供下列公式表達式功能，可用於在字串上操作：

- [字串常值](#)
- [索引運算子](#) (s[index])
- [配量運算子](#) (s[start:end:step])
- [比較函數](#)，您可以使用依[語彙順序](#)比較字串
- [字串函數](#)，其中包含可剖析序列化 JSON 物件並將字串轉換為數字的jp函數

篩選資料點

您可以使用 [if 函數](#) 來篩選不符合條件的資料點。if 函數會評估條件，並傳回 true 和 false 結果的不同值。您可以使用[無常數](#)做為一個if函數案例的輸出，以捨棄該案例的資料點。

篩選出符合條件的資料點

- 建立使用 if 函數的轉換來定義條件，以檢查是否符合條件，並傳回 none result_if_true 或 result_if_false 值。

Example 範例：篩選出水未沸騰的資料點

假設您有提供機器中水溫（攝氏）的測量 temp_c。您可以定義下列轉換，以篩選出水未沸騰的資料點：

- 轉換：boiling_temps = if(gte(temp_c, 100), temp_c, none) – 如果溫度大於或等於攝氏 100 度，則傳回溫度，否則不會傳回任何資料點。

計數符合條件的資料點

您可以使用[比較函數](#)和 [sum\(\)](#) 來計算條件為 true 的資料點數目。

計算符合條件的資料點

1. 建立使用比較函數在另一個屬性上定義篩選條件的轉換。
2. 建立可加總符合該條件之資料點的指標。

Example 範例：計算水沸騰的資料點數量

假設您有提供機器中水溫（攝氏）的測量 temp_c。您可以定義下列轉換和指標屬性，以計算水沸騰的資料點數量：

- 轉換：is_boiling = gte(temp_c, 100) – 1 如果溫度大於或等於攝氏 100 度，則傳回 1，否則傳回 0。
- 指標：boiling_count = sum(is_boiling) – 傳回水沸騰的資料點數目。

公式中的延遲資料

AWS IoT SiteWise 支援延遲擷取長達 7 天的資料。當 AWS IoT SiteWise 收到延遲資料時，它會重新計算過去時段內輸入延遲資料的任何指標的現有值。這些重新計算會產生資料處理費用。

Note

當 AWS IoT SiteWise 運算輸入延遲資料的屬性時，它會使用每個屬性目前的公式表達式。

AWS IoT SiteWise 重新計算指標的過去時段後，它會取代該時段的先前值。如果您啟用該指標的通知，AWS IoT SiteWise 也會發出屬性值通知。這表示您可以針對先前收到通知的相同屬性和時間戳記，接收新的屬性值更新通知。如果您的應用程式或資料湖使用屬性值通知，您必須使用新的值來更新之前的值，使其資料正確無誤。

公式中的資料品質

在中 AWS IoT SiteWise，每個資料點都有一個品質碼，可以是下列其中一項：

- GOOD – 資料不會受到任何問題的影響。
- BAD – 資料會受到感應器故障等問題的影響。

- UNCERTAIN – 資料受到感應器不準確等問題的影響。

AWS IoT SiteWise 只會在運算轉換和指標時耗用GOOD品質資料。AWS IoT SiteWise 輸出只會輸出GOOD品質資料以成功運算。如果運算失敗，則 AWS IoT SiteWise 不會輸出該運算的資料點。如果計算導致未定義、無限或溢出值，則可能會發生這種情況。

如需如何查詢資料及依資料品質篩選的詳細資訊，請參閱 [從查詢資料 AWS IoT SiteWise](#)。

未定義、無限和溢出值

有些公式表達式（例如 $x / 0$ 、 $\text{sqrt}(-1)$ 或 $\log(0)$ ）會計算在實數系統中未定義、無限或超出支援範圍的值 AWS IoT SiteWise。當資產屬性的表達式運算未定義、無限或溢位值時，AWS IoT SiteWise 不會輸出該運算的資料點。

AWS IoT SiteWise 如果資料點運算非數值作為公式表達式的結果，也不會輸出資料點。這表示如果您定義計算字串、陣列或無常數的公式，則 AWS IoT SiteWise 不會輸出該運算的資料點。

Example 範例

下列每個公式表達式都會產生 AWS IoT SiteWise 無法表示為 number 的值。當計算這些公式表達式時，AWS IoT SiteWise 不會輸出資料點。

- $x / 0$ 未定義。
- $\log(0)$ 未定義。
- $\text{sqrt}(-1)$ 在實數系統中未定義。
- "hello" + " world" 是字串。
- `jp({'values':[3,6,7]}, '$.values')` 是陣列。
- `if(gte(temp, 300), temp, none)` 當小於 none 時 temp，為 300。

建立自訂複合模型（元件）

如果您使用 主控台，則可自訂複合模型或元件，為您的資產模型和元件模型提供另一層級的組織。您可以使用它們來透過分組屬性或參考其他模型來建構模型。如需使用自訂複合模型的詳細資訊，請參閱 [自訂複合模型（元件）](#)。

您可以在現有的資產模型或元件模型中建立自訂複合模型。自訂複合模型有兩種類型。若要將模型中的相關屬性分組，您可以建立內嵌自訂複合模型。若要在資產模型或元件模型中參考元件模型，您可以建立 component-model-based 自訂複合模型。

下列各節說明如何使用 AWS IoT SiteWise API 來建立自訂複合模型。

主題

- [建立內嵌元件（主控台）](#)
- [建立內嵌自訂複合模型 \(AWS CLI\)](#)
- [建立component-model-based元件（主控台）](#)
- [建立component-model-based自訂複合模型 \(AWS CLI\)](#)

建立內嵌元件（主控台）

您可以使用 AWS IoT SiteWise 主控台來建立定義其自身屬性的內嵌元件。

Note

由於這是內嵌元件，因此這些屬性僅適用於目前的資產模型，不會在其他地方共用。如果您需要產生可重複使用的模型（例如，要在多個資產模型之間共用，或在一個資產模型中包含多個執行個體），您應該改為根據元件模型建立元件。如需詳細資訊，請參閱下一節。

建立元件（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇您要新增元件的資產模型。
4. 在屬性索引標籤上，選擇元件。
5. 選擇建立元件。
6. 在建立元件頁面上，執行下列動作：
 - a. 輸入元件的名稱，例如 **ServoMotor** 或 **ServoMotor Model**。此名稱在此區域您帳戶中的所有元件中必須是唯一的。
 - b. (選擇性) 新增模型的 Attribute definitions (屬性定義)。屬性代表很少變更的資訊。如需詳細資訊，請參閱 [定義靜態資料（屬性）](#)。
 - c. (選擇性) 新增模型的 Measurement definitions (衡量值定義)。測量代表來自您設備的資料串流。如需詳細資訊，請參閱 [從設備定義資料串流（度量）](#)。
 - d. (選擇性) 新增模型的 Transform definitions (轉換定義)。轉換是將資料從一個表單映射到另一個表單的公式。如需詳細資訊，請參閱 [轉換資料（轉換）](#)。

- e. (選擇性) 新增模型的 Metric definitions (指標定義)。指標是可在時間間隔內彙總資料的公式。指標可以從相關聯的資產輸入資料，讓您可以計算代表操作或操作子集的值。如需詳細資訊，請參閱[從屬性和其他資產彙總資料 \(指標\)](#)。
- f. 選擇建立元件。

建立內嵌自訂複合模型 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來建立定義自己的屬性的內嵌自訂複合模型。

使用 [CreateAssetModelCompositeModel](#) 操作來建立具有 屬性的內嵌模型。此操作預期會有具備下列結構的承載。

Note

由於這是內嵌複合模型，因此這些屬性僅適用於目前的資產模型，不會在其他地方共用。讓它「內嵌」的原因在於它不會為 `composedAssetModelId` 欄位提供值。
如果您需要產生可重複使用的模型（例如，要在多個資產模型之間共用，或在一個資產模型中包含多個執行個體），您應該改為建立以 `component-model-based` 複合模型。如需詳細資訊，請參閱下一節。

```
{
  "assetModelCompositeModelName": "CNCLathe_ServoMotorA",
  "assetModelCompositeModelType": "CUSTOM",
  "assetModelCompositeModelProperties": [
    {
      "dataType": "DOUBLE",
      "name": "Servo Motor Temperature",
      "type": {
        "measurement": {}
      },
      "unit": "Celsius"
    },
    {
      "dataType": "DOUBLE",
      "name": "Spindle speed",
      "type": {
        "measurement": {}
      },
      "unit": "rpm"
    }
  ]
}
```

```
    }
  ]
}
```

建立component-model-based元件（主控台）

您可以使用 AWS IoT SiteWise 主控台，根據元件模型建立元件。

建立component-model-based元件（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇您要新增元件的資產模型。
4. 在屬性索引標籤上，選擇元件。
5. 選擇建立元件。
6. 在建立元件頁面上，執行下列動作：
 - a. 選取您要以元件為基礎的元件模型。
 - b. 輸入元件的名稱，例如 **ServoMotor**或 **ServoMotor Model**。此名稱在此區域您帳戶中的所有元件中必須是唯一的。
 - c. 選擇建立元件。

建立component-model-based自訂複合模型 (AWS CLI)

您可以使用 AWS CLI 在資產模型中建立component-model-based自訂複合模型。component-model-based自訂複合模型是您已在其他位置定義的元件模型參考。

使用 [CreateAssetModelCompositeModel](#) 操作來建立component-model-based自訂複合模型。此操作預期會有具備下列結構的承載。

Note

在此範例中，的值composedAssetModelId是資產模型 ID 或現有元件模型的外部 ID。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs物件](#)。如需如何建立元件模型的範例，請參閱 [建立元件模型 \(AWS CLI\)](#)。

```
{
```

```
"assetModelCompositeModelName": "CNCLathe_ServoMotorA",  
"assetModelCompositeModelType": "CUSTOM",  
"composedAssetModelId": component model ID  
]
```

由於它只是參考，component-model-based自訂複合模型除了名稱之外沒有自己的屬性。

如果您想要將相同元件的多個執行個體新增至資產模型（例如，具有多個伺服馬達的 CNC 機器），您可以新增多個component-model-based自訂複合模型，每個模型都有自己的名稱，但都參考相同的composedAssetModelId。

您可以在其他元件中巢狀化元件。若要這樣做，您可以將以component-model-based複合模型新增至其中一個元件模型，如本範例所示。

在 中建立資產模型的資產 AWS IoT SiteWise

您可以從資產模型建立資產。您必須先擁有資產模型，才能建立資產。若您尚未建立資產模型，請參閱 [在 中建立資產模型 AWS IoT SiteWise](#)。

Note

您只能從 ACTIVE 模型建立資產。如果模型狀態不是 ACTIVE，您可能需要等待幾分鐘，才能從該模型建立資產。如需詳細資訊，請參閱[資產和模型狀態](#)。

主題

- [建立資產（主控台）](#)
- [建立資產 \(AWS CLI\)](#)
- [設定新的資產](#)

建立資產（主控台）

您可以使用 AWS IoT SiteWise 主控台來建立資產。

建立資產 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。

3. 選擇 Create asset (建立資產)。
4. 在 Create asset (建立資產) 頁面上，執行下列動作：
 - a. 針對 Model (模型)，選擇要從中建立資產的資產模型。

Note

如果您的模型非 ACTIVE (作用中)，則必須等候模型處於作用中狀態，或如果模型為 FAILED (失敗)，則必須解決問題。

- b. 輸入資產的 Name (名稱)。
- c. (選用) 為您的資產新增標籤。如需詳細資訊，請參閱[標記您的 AWS IoT SiteWise 資源](#)。
- d. 選擇 Create asset (建立資產)。

當您建立資產時，AWS IoT SiteWise 主控台會導覽至新資產的頁面。在此頁面上，您可以看到資產的 Status (狀態)，最初為 CREATING (建立中)。此頁面會自動更新，因此您可以等候資產的狀態更新。

Note

資產建立程序可能需要一分鐘的時間。狀態為 ACTIVE 之後，您可以對資產執行更新操作。如需詳細資訊，請參閱[資產和模型狀態](#)。

建立資產後，請參閱 [設定新的資產](#)。

建立資產 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 從資產模型建立資產。

您必須具有 assetModelId 才能建立資產。如果您建立了資產模型，但不知道其 assetModelId，請使用 [ListAssetModels](#) API 來檢視所有資產模型。

若要從資產模型建立資產，請使用 [CreateAsset](#) API 搭配下列參數：

- assetName – 新資產的名稱。為您的資產命名，以協助您識別資產。
- assetModelId – 資產的 ID。這是 UUID 格式的實際 ID，externalId:myExternalId 如果有的話，則為 。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

建立資產 (AWS CLI)

- 執行下列命令以建立資產。將 *asset-name* 取代為資產的名稱，並將 *asset-model-id* 取代為資產模型的 ID 或外部 ID。

```
aws iotsitewise create-asset \  
  --asset-name asset-name \  
  --asset-model-id asset-model-id
```

此操作會以下列格式傳回包含新資產詳細資料和狀態的回應。

```
{  
  "assetId": "String",  
  "assetArn": "String",  
  "assetStatus": {  
    "state": "String",  
    "error": {  
      "code": "String",  
      "message": "String"  
    }  
  }  
}
```

資產的 state 會是 CREATING，直到資產建立為止。

Note

資產建立程序可能需要一分鐘的時間。若要檢查資產的狀態，請使用 [DescribeAsset](#) 操作搭配資產的 ID 做為 *assetId* 參數。在資產的 state 為 之後 ACTIVE，您可以對資產執行更新操作。如需詳細資訊，請參閱[資產和模型狀態](#)。

建立資產後，請參閱 [設定新的資產](#)。

設定新的資產

在 中建立資產之後 AWS IoT SiteWise，您可以採取幾個後續步驟，以充分利用資產及其資料。這些步驟可能包括設定資料串流從資產擷取資料、設定警示和通知以監控資產效能、建立視覺化和儀表板以顯示資產的資料，以及將資產與其他 AWS 服務或第三方應用程式整合以進行進一步分析或自動化。

使用下列選用動作完成資產的設定：

- [管理 的資料串流 AWS IoT SiteWise](#) 如果您的資產具有衡量值屬性。
- [更新屬性值](#) 如果您的資產具有唯一的屬性值。
- [關聯和取消關聯資產](#) 如果您的資產是父項資產。

在 上搜尋資產 AWS IoT SiteWise 主控台

使用 AWS IoT SiteWise 主控台 搜尋功能根據中繼資料和即時屬性值篩選條件來尋找資產。

先決條件

AWS IoT SiteWise 需要與 整合的許可 AWS IoT TwinMaker ，才能更好地組織和建立工業資料的模型。如果您已授予 的許可 AWS IoT SiteWise ，請使用 [ExecuteQuery](#) API。如果您尚未授予許可給 AWS IoT SiteWise ，且需要協助入門，請參閱 [整合 AWS IoT SiteWise 和 AWS IoT TwinMaker](#)。

上的進階搜尋 AWS IoT SiteWise 主控台

中繼資料搜尋

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇資產下的進階搜尋。
3. 在進階搜尋下，選擇中繼資料搜尋選項。
4. 填寫參數。填寫盡可能多的欄位以進行有效率的搜尋。
 - a. 資產名稱 — 輸入完整資產名稱或部分名稱以進行廣泛搜尋。
 - b. 屬性名稱 — 輸入完整屬性名稱或部分名稱以進行廣泛搜尋。
 - c. 運算子 — 從中選擇運算子：
 - =
 - <
 - >
 - <=
 - >=
 - d. 屬性值 — 此值會與屬性的最新值進行比較。

e. 屬性值類型 — 屬性的資料類型。請選擇下列項目：

- 雙重
- 整數
- : 字串
- : 布林值

5. 選擇 Search (搜尋)。

6. 從搜尋結果表格中，從名稱欄中選擇資產。這會帶您前往該資產的詳細資產頁面。

Assets

Assets represent Industrial devices and processes that send data streams to SiteWise. Models are structures that enforce a specific model of properties and hierarchies for all instances of each asset. You must create every asset from a model.

Advanced search

Use advanced search to find assets based on specific metadata. In addition, you can enter SQL queries directly in the query builder.

Metadata search | Query builder

Asset name: Level-2 | Property name: power_max | Operator: > | Property value: 20 | Property value type: Double

Clear | Search

Search results (2)

Name	Asset id	Description
Level-2-asset-1	d0e9019b-9c38-4316-b574-38317aa38143	
Level-2-asset-2	b9c0d2fc-1527-42ce-8ba2-d1a4e8ff43de	Example description

部分搜尋

資產搜尋不需要提供所有參數。以下是使用中繼資料搜尋選項的部分搜尋範例：

- 依資產名稱尋找資產：
 - 單獨在資產名稱欄位中輸入值。
 - 屬性名稱和屬性值欄位為空白。
- 尋找包含具有特定名稱屬性的資產：
 - 單獨在屬性名稱欄位中輸入值。

- 資產名稱和屬性值欄位為空白。
- 根據其屬性的最新值尋找資產：
 - 在屬性名稱和屬性值欄位中輸入值。
 - 選取運算子和屬性值類型。

查詢建置器搜尋

1. 導覽至 AWS IoT SiteWise 主控台。
2. 在導覽窗格中，選擇資產下的進階搜尋。
3. 在進階搜尋下，選擇查詢建置器選項。
4. 在查詢建置器窗格中，撰寫 SQL 查詢以擷取 asset_name、asset_id 和 asset_description。
5. 選擇 Search (搜尋)。
6. 從搜尋結果表格中，從名稱欄中選擇資產。這會帶您前往該資產的詳細資產頁面。

The screenshot shows the AWS IoT SiteWise console interface. At the top, there's a navigation bar with the AWS logo, 'Services', a search bar, and a region dropdown set to 'N. Virginia'. Below this, the 'Assets' page is displayed. It includes a 'Create asset' button and a description of assets. The 'Advanced search' section is active, showing the 'Query builder' tab. A SQL query is entered in the query builder: `SELECT a.asset_id, a.asset_name, a.asset_description FROM asset a, asset_property p, latest_value_time_series ts WHERE a.asset_name LIKE '%asset-2%' AND a.property_name = 'temperature_f' AND ts.double_value > 50.0`. Below the query builder, the 'Search results (2)' section shows a table with two results:

Name	Asset id	Description
Level-2a-asset-2	4fed596d-e903-4338-86db-34ca9301233a	Generator #3
Level-2b-asset-2	b4ac2b24-4fce-4a72-9fea-ef6d0f741e8d	Generator #2

Note

- SQL 查詢中的 SELECT 子句必須包含 `asset_name` 和 `asset_id` 欄位，以確保搜尋結果資料表中的有效資產。
- 查詢建置器只會在結果資料表中顯示名稱、資產 ID 和描述。將更多欄位新增至 SELECT 子句並不會將更多資料欄新增至結果資料表

更新屬性值

資產會繼承其資產模型的屬性，包括屬性的預設值。在某些情況下，您會想要保留資產模型的預設屬性，例如資產製造商屬性。在其他情況下，您會想要更新繼承的屬性，例如資產的緯度和經度。

Updating an attribute value (console)

您可以使用 AWS IoT SiteWise 主控台來更新屬性資產屬性的值。

更新屬性值 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要為其更新屬性的資產。

Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇編輯。
5. 尋找要更新的屬性，然後輸入其新值。

The screenshot shows the 'Attributes' section of the AWS IoT SiteWise console. It contains two main input areas. On the left, under the heading '"Location"', there is a text input field containing the word 'Renton'. This field is circled in red. Below the input field, a note states 'Must be less than 2048 characters.' On the right, under the heading 'Notification status', there is a dropdown menu currently showing 'DISABLED'. Below this dropdown, a detailed notification topic string is displayed: 'Notification will be published to topic \$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'.

6. 選擇儲存。

Updating an attribute value (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來更新屬性值。

您必須知道資產的 `assetId` 和財產的 `propertyId`，才能完成此程序。您也可以使用外部 ID。如果您建立資產但不知道其 `assetId`，請使用 [ListAssets](#) API 列出特定模型的所有資產。使用 [DescribeAsset](#) 操作來檢視資產的屬性，包括屬性 IDs。

使用 [BatchPutAssetPropertyValue](#) 操作將屬性值指派給您的資產。您可以使用此操作一次設定多個屬性。此操作的裝載包含項目清單，而每個項目都包含資產 ID、屬性 ID 和屬性值。

更新屬性的值 (AWS CLI)

1. 建立名為 `batch-put-payload.json` 的檔案，並將下列 JSON 物件複製到檔案。此範例承載示範如何設定風力發電機的緯度和經度。更新 ID、值和時間戳記，以修改您的使用案例的承載。

```
{
  "entries": [
    {
      "entryId": "windfarm3-turbine7-latitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 47.6204
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    },
    {
      "entryId": "windfarm3-turbine7-longitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 122.3491
          },

```

```

        "timestamp": {
            "timeInSeconds": 1575691200
        }
    }
]
}
]
}

```

- 承載中的每個項目都包含 `entryId`，您可以將其定義為任何獨特的字串。如果有任何請求項目失敗，每個錯誤都會包含對應請求的 `entryId`，這樣您就可以知道要重試哪些請求。
- 若要設定屬性值，您可以在每個屬性內容的 `propertyValues` 清單中包含一個時間戳記品質值 (TQV) 結構。此結構必須包含新的 `value` 和目前的 `timestamp`。
 - `value` – 包含下列其中一個欄位的結構，取決於所設定屬性的類型：
 - `booleanValue`
 - `doubleValue`
 - `integerValue`
 - `stringValue`
 - `nullValue`
 - `timestamp` – 包含目前 Unix epoch 時間的結構，以秒為單位，`timeInSeconds`。AWS IoT SiteWise 會拒絕過去超過 7 天或未來超過 5 分鐘的任何時間戳記資料點。

如需如何準備 [BatchPutAssetPropertyValue](#) 承載的詳細資訊，請參閱 [使用 AWS IoT SiteWise APIs 擷取資料](#)。

2. 執行下列命令，將屬性值傳送至 AWS IoT SiteWise：

```
aws iotsitewise batch-put-asset-property-value -\-cli-input-json file://batch-put-payload.json
```

關聯和取消關聯資產

如果您的資產模型定義了任何子資產模型階層，您可以將子資產與您的資產建立關聯。父資產可以存取並從關聯的資產彙總資料。如需階層式資產模型的詳細資訊，請參閱 [定義資產模型階層](#)。

主題

- [關聯和取消關聯資產 \(主控台\)](#)
- [關聯和取消關聯資產 \(AWS CLI\)](#)

關聯和取消關聯資產 (主控台)

您可以使用 AWS IoT SiteWise 主控台來關聯和取消關聯資產。

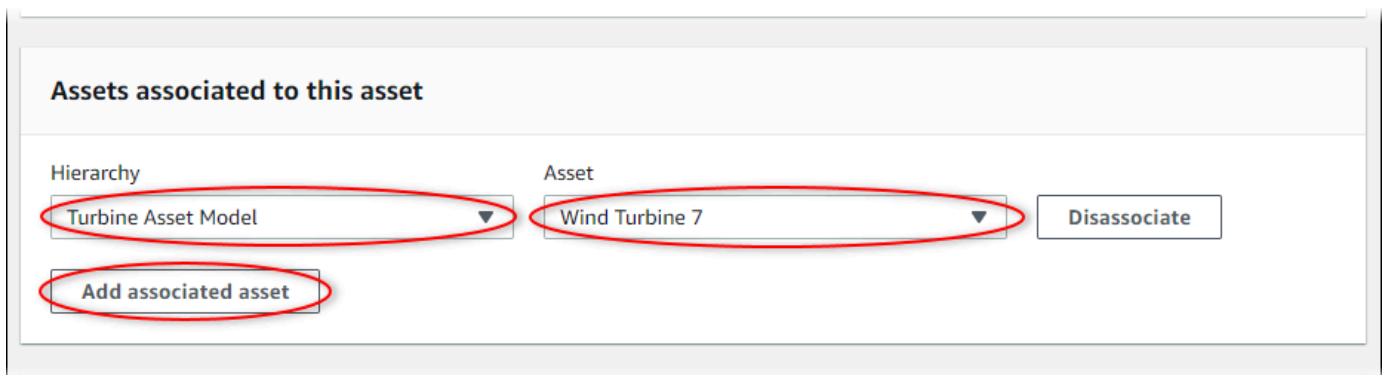
關聯資產 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要關聯子資產的父資產。

Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇編輯。
5. 在 Assets associated to this asset (與此資產相關聯的資產) 中，選擇 Add associated asset (新增相關聯的資產)。



6. 針對 Hierarchy (階層) 中，選擇定義主資產與子資產之間關係的階層。
7. 對於 Asset (資產)，選擇要關聯的子資產。
8. 選擇儲存。

取消關聯資產 (主控台)

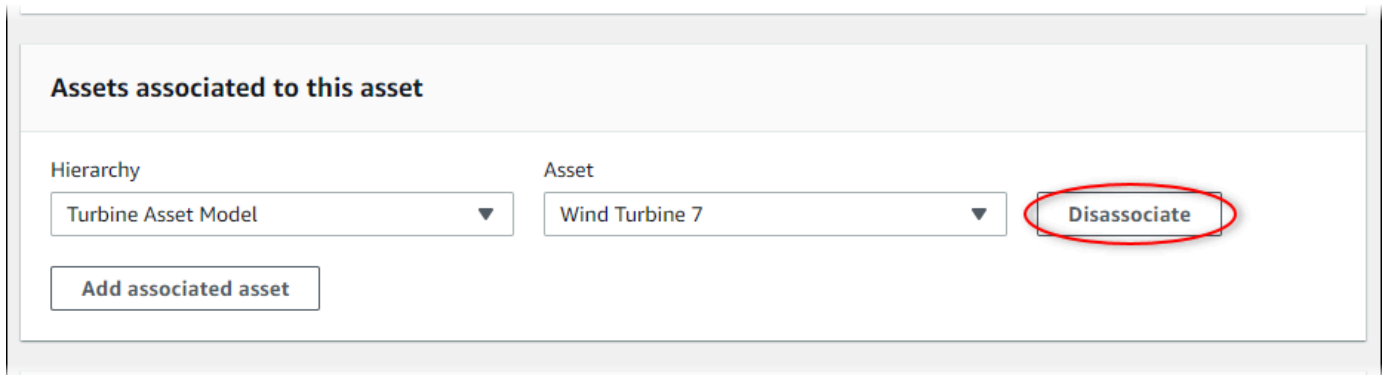
1. 導覽至 [AWS IoT SiteWise 主控台](#)。

2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要取消關聯子資產的父資產。

i Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇編輯。
5. 在 Assets associated to this asset (與此資產相關聯的資產) 中，選擇資產的 Disassociate (取消關聯)。



6. 選擇儲存。

關聯和取消關聯資產 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來關聯和取消關聯資產。

針對此程序，您必須知道定義與子資產模型之關係的父資產模型中階層 (hierarchyId) 的 ID。使用 [DescribeAsset](#) 操作來尋找回應中的階層 ID。

尋找階層 ID

- 執行下列命令來描述父資產。將 *parent-asset-id* 取代為父資產的 ID 或外部 ID。

```
aws iotsitewise describe-asset --asset-id parent-asset-id
```

此操作會傳回包含資產詳細資料的回應。回應包含具有下列結構的 assetHierarchies 清單：

```
{  
  ...  
}
```

```
"assetHierarchies": [  
  {  
    "id": "String",  
    "name": "String"  
  },  
  ...  
]
```

階層 ID 是資產階層清單中階層的 id 值。

取得階層 ID 之後，您可以將資產與該階層關聯或取消關聯。

若要將子資產與父項資產建立關聯，請使用 [AssociateAssets](#) 操作。若要將子資產與父資產取消關聯，請使用 [DisassociateAssets](#) 操作。指定下列參數，這兩個操作的參數相同：

- `assetId` – 父資產的 ID 或外部 ID。
- `hierarchyId` – 父資產中的階層 ID 或外部 ID。
- `childAssetId` – 子資產的 ID 或外部 ID。

建立資產的關聯 (AWS CLI)

- 執行下列命令來關聯子資產與父資產。將 *parent-asset-id*、*hierarchy-id* 和 *child-asset-id* 取代為個別 IDs：

```
aws iotsitewise associate-assets \  
  --asset-id parent-asset-id \  
  --hierarchy-id hierarchy-id \  
  --child-asset-id child-asset-id
```

取消資產的關聯 (AWS CLI)

- 執行下列命令來取消關聯子資產與父資產。將 *parent-asset-id*、*hierarchy-id* 和 *child-asset-id* 取代為個別 IDs：

```
aws iotsitewise disassociate-assets \  
  --asset-id parent-asset-id \  
  --hierarchy-id hierarchy-id \  
  --child-asset-id child-asset-id
```

```
--child-asset-id child-asset-id
```

更新資產和模型

您可以在 中更新資產、資產模型和元件模型 AWS IoT SiteWise ，以修改其名稱和定義。這些更新操作是非同步的，需要一些時間才能傳播 AWS IoT SiteWise。在進行其他變更之前，請檢查資產或模型的狀態。您必須先等到變更傳播，才能繼續使用更新的資產或模型。

主題

- [在 中更新資產 AWS IoT SiteWise](#)
- [更新資產模型和元件模型](#)
- [更新自訂複合模型（元件）](#)
- [資產模型寫入的樂觀鎖定](#)

在 中更新資產 AWS IoT SiteWise

您可以使用 AWS IoT SiteWise 主控台或 API 來更新資產的名稱。

當您更新資產時，資產的狀態會是 `UPDATING`直到變更傳播為止。如需詳細資訊，請參閱[資產和模型狀態](#)。

主題

- [更新資產（主控台）](#)
- [更新資產 \(AWS CLI\)](#)

更新資產（主控台）

您可以使用 AWS IoT SiteWise 主控台來更新資產詳細資訊。

更新資產 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇要更新的資產。

 Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇編輯。
5. 更新資產的 Name (名稱)。
6. (選擇性) 在此頁面上，更新資產的其他資訊。如需詳細資訊，請參閱下列內容：
 - [管理的資料串流 AWS IoT SiteWise](#)
 - [更新屬性值](#)
 - [與其他 AWS 服務互動](#)
7. 選擇儲存。

更新資產 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來更新資產的名稱。

使用 [UpdateAsset](#) 操作來更新資產。指定下列參數：

- `assetId` – 資產的 ID。這是 UUID 格式的實際 ID，`externalId:myExternalId`如果有的話，則為。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs物件](#)。
- `assetName` – 資產的新名稱。

更新資產的名稱 (AWS CLI)

- 執行以下命令來更新資產的名稱。以 `### ID ### ID ## asset-id`。使用 `##### asset-name`。

```
aws iotsitewise update-asset \  
  --asset-id asset-id \  
  --asset-name asset-name
```

更新資產模型和元件模型

使用 AWS IoT SiteWise 主控台或 API 更新資產模型或元件模型。

您無法變更現有屬性的類型或資料類型，或現有指標的視窗。您也無法將模型的類型從資產模型變更為元件模型，或反之亦然。

Important

- 如果您從資產模型或元件模型中移除屬性，會 AWS IoT SiteWise 刪除該屬性的所有先前資料。對於元件模型，這會影響使用該元件模型的所有資產模型，因此請特別小心，以了解您的變更可能適用的範圍。
- 如果您從資產模型中移除階層定義，會 AWS IoT SiteWise 取消該階層中的所有資產的關聯。

更新資產模型時，以該模型為基礎的每個資產都會反映您對基礎模型所做的任何變更。在變更傳播之前，每個資產都有 UPDATING 狀態。您必須等到這些資產變回 ACTIVE 狀態，才能與它們進行互動。在這段期間，更新的資產模型的狀態將是 PROPAGATING。

當您更新元件模型時，包含該元件模型的每個資產模型都會反映變更。在元件模型變更傳播之前，每個受影響的資產模型都會具有 UPDATING 狀態，然後在更新其相關聯的資產 PROPAGATING 時，如前段所述。您必須等到這些資產模型回到 ACTIVE 狀態，才能與其互動。在此期間，更新後的元件模型狀態將為 PROPAGATING。

如需詳細資訊，請參閱[資產和模型狀態](#)。

主題

- [更新資產或元件模型（主控台）](#)
- [更新資產或元件模型 \(AWS CLI\)](#)

更新資產或元件模型（主控台）

您可以使用 AWS IoT SiteWise 主控台來更新資產模型或元件模型。

更新資產模型或元件模型（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇要更新的資產模型或元件模型。
4. 選擇編輯。

5. 在 Edit model (編輯模型) 頁面上，執行以下任一項操作：
 - 在 Model details (模型詳細資料) 中，變更模型的 Name (名稱)。
 - 變更任何 Attribute definitions (屬性定義)。您無法變更現有屬性的 Data type (資料類型)。如需詳細資訊，請參閱[定義靜態資料 \(屬性\)](#)。
 - 變更任何 Measurement definitions (衡量值定義)。您無法變更現有衡量值的 Data type (資料類型)。如需詳細資訊，請參閱[從設備定義資料串流 \(度量\)](#)。
 - 變更任何 Transform definitions (轉換定義)。如需詳細資訊，請參閱[轉換資料 \(轉換\)](#)。
 - 變更任何 Metric definitions (指標定義)。您無法變更現有指標的 Time interval (時間間隔)。如需詳細資訊，請參閱[從屬性和其他資產彙總資料 \(指標\)](#)。
 - (僅限資產模型) 變更任何階層定義。您無法變更現有階層的 Hierarchy model (階層模型)。如需詳細資訊，請參閱[定義資產模型階層](#)。
6. 選擇 Save (儲存)。

Note

如果自您上次開啟編輯模型頁面以來，其他使用者成功更新資產模型，則會拒絕在主控台中提出的更新請求。主控台會提示使用者重新整理編輯模型頁面，以擷取更新的模型。您必須再次進行更新，然後重試儲存。如需詳細資訊，請參閱[資產模型寫入的樂觀鎖定](#)。

更新資產或元件模型 (AWS CLI)

使用 AWS Command Line Interface (AWS CLI) 更新資產模型或元件模型。

使用 [UpdateAssetModel](#) API 更新資產模型或元件模型的名稱、描述和屬性。僅針對資產模型，您可以更新階層。指定下列參數：

- `assetModelId` – 資產的 ID。這是 UUID 格式的實際 ID，`externalId:myExternalId` 如果有的話，則為 `externalId:myExternalId`。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs 物件](#)。

在承載中指定更新的模型。若要了解資產模型或元件模型的預期格式，請參閱 [在中建立資產模型 AWS IoT SiteWise](#)。

⚠ Warning

[UpdateAssetModel](#) API 會使用您在承載中提供的模型覆寫現有模型。若要避免刪除模型的屬性或階層，您必須在更新的模型承載中包含其 IDs 和定義。若要了解如何查詢模型的現有結構，請參閱 [DescribeAssetModel](#) 操作。

ℹ Note

下列程序只能更新類型 的複合模型AWS/ALARM。如果您想要更新CUSTOM複合模型，請改用 [UpdateAssetModelCompositeModel](#)。如需詳細資訊，請參閱[更新自訂複合模型（元件）](#)。

更新資產模型或元件模型 (AWS CLI)

1. 執行下列命令來擷取現有的模型定義。將 *asset-model-id* 取代為要更新的資產模型或元件模型的 ID 或外部 ID。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

上述命令會傳回對應至模型最新版本的模型定義。

對於資產模型處於 FAILED 狀態的使用案例，請擷取與其作用中版本對應的有效模型定義，以建置您的更新請求。如需詳細資訊，請參閱 [資產模型版本](#)。執行下列命令來擷取作用中模型定義：

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id --asset-model-version ACTIVE
```

操作會傳回包含模型詳細資訊的回應。回應有以下結構。

```
{
  "assetModelId": "String",
  "assetModelArn": "String",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition,
  "assetModelCompositeModels": Array of AssetModelCompositeModel,
  "assetModelCompositeModelSummaries": Array of AssetModelCompositeModelSummary,
```



```

    "assetModelCreationDate": "String",
    "assetModelLastUpdateDate": "String",
    "assetModelStatus": {
      "state": "String",
      "error": {
        "code": "String",
        "message": "String"
      },
    },
    "assetModelType": "String"
  },
  "assetModelVersion": "String",
  "eTag": "String"
}

```

如需詳細資訊，請參閱 [DescribeAssetModel](#) 操作。

2. 建立名為 `update-asset-model.json` 的文件，並將前一個命令的回應複製到檔案中。
3. 從 JSON 物件的 `update-asset-model.json` 中移除以下鍵值組：

- `assetModelId`
- `assetModelArn`
- `assetModelCompositeModelSummaries`
- `assetModelCreationDate`
- `assetModelLastUpdateDate`
- `assetModelStatus`
- `assetModelType`
- `assetModelVersion`
- `eTag`

[UpdateAssetModel](#) 操作預期承載具有下列結構：

```

{
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition,
  "assetModelCompositeModels": Array of AssetModelCompositeModel
}

```

4. 在 `update-asset-model.json` 中，執行下列任何一項：
 - 變更資產模型的名稱 (`assetModelName`)。
 - 變更、新增或移除資產模型的描述 (`assetModelDescription`)。
 - 變更、新增或移除任何資產模型的屬性 (`assetModelProperties`)。您無法變更現有屬性的 `dataType` 或現有指標的 `window`。如需詳細資訊，請參閱[定義資料屬性](#)。
 - 變更、新增或移除任何資產模型的階層 (`assetModelHierarchies`)。您無法變更現有階層的 `childAssetModelId`。如需詳細資訊，請參閱[定義資產模型階層](#)。
 - 變更、新增或移除任何資產模型類型 `AWS/ALARM()` 的複合模型 `assetModelCompositeModels`。警示會監控其他屬性，讓您可以識別設備或程序何時需要注意。每個警示定義都是複合模型，可標準化警示使用的屬性集。如需詳細資訊，請參閱[在中使用警示監控資料 AWS IoT SiteWise](#)及[在中定義資產模型的警示 AWS IoT SiteWise](#)。
5. 執行以下命令，使用儲存在 `update-asset-model.json` 中的定義更新資產模型。將 *`asset-model-id`* 取代為資產模型的 ID：

```
aws iotsitewise update-asset-model \  
  --asset-model-id asset-model-id \  
  --cli-input-json file://model-payload.json
```

Important

當多個使用者同時更新資產模型時，其他使用者可能會無意中覆寫使用者的變更。若要避免這種情況，您必須定義條件式更新請求。請參閱[資產模型寫入的樂觀鎖定](#)。

更新自訂複合模型（元件）

您可以使用 AWS IoT SiteWise API 來更新自訂複合模型，或使用 AWS IoT SiteWise 主控台來更新元件。

主題

- [更新元件（主控台）](#)
- [更新自訂複合模型 \(AWS CLI\)](#)

更新元件（主控台）

您可以使用 AWS IoT SiteWise 主控台來更新元件。

更新元件（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇元件所在的資產模型。
4. 在屬性索引標籤上，選擇元件。
5. 選擇您要更新的元件。
6. 選擇編輯。
7. 在編輯元件頁面上，執行下列任一動作：
 - 在 Model details (模型詳細資料) 中，變更模型的 Name (名稱)。
 - 變更任何 Attribute definitions (屬性定義)。您無法變更現有屬性的 Data type (資料類型)。如需詳細資訊，請參閱[定義靜態資料（屬性）](#)。
 - 變更任何 Measurement definitions (衡量值定義)。您無法變更現有衡量值的 Data type (資料類型)。如需詳細資訊，請參閱[從設備定義資料串流（度量）](#)。
 - 變更任何 Transform definitions (轉換定義)。如需詳細資訊，請參閱[轉換資料（轉換）](#)。
 - 變更任何 Metric definitions (指標定義)。您無法變更現有指標的 Time interval (時間間隔)。如需詳細資訊，請參閱[從屬性和其他資產彙總資料（指標）](#)。
8. 選擇 Save (儲存)。

更新自訂複合模型 (AWS CLI)

使用 AWS Command Line Interface (AWS CLI) 更新自訂複合模型。

若要更新名稱或描述，請使用 [UpdateAssetModelCompositeModel](#) 操作。僅適用於內嵌自訂複合模型，您也可以更新屬性。您無法更新以 component-model-based 自訂複合模型的屬性，因為其參考的元件模型提供其相關聯的屬性。

Important

如果您從自訂複合模型中移除屬性，會 AWS IoT SiteWise 刪除該屬性的所有先前資料。您無法變更現有屬性的類型或資料類型。

若要將現有的複合模型屬性取代為具有相同的新複合模型屬性 name，請執行下列動作：

1. 提交已移除整個現有屬性的UpdateAssetModelCompositeModel請求。
2. 提交包含新屬性的第二個UpdateAssetModelCompositeModel請求。新的資產屬性將與上一個資產屬性name相同，AWS IoT SiteWise 並會產生新的唯一 id。

更新自訂複合模型 (AWS CLI)

1. 若要擷取現有的複合模型定義，請執行下列命令。將 *composite-model-id* 取代為要更新的自訂複合模型的 ID 或外部 ID，並將 *asset-model-id* 取代為與自訂複合模型相關聯的資產模型。如需詳細資訊，請參閱「AWS IoT SiteWise 使用者指南」。

- a. 執行以下命令：

```
aws iotsitewise describe-asset-model-composite-model \
--asset-model-composite-model-id composite-model-id \
--asset-model-id asset-model-id
```

- b. 上述命令會傳回對應至相關聯模型最新版本的複合模型定義。對於資產模型處於 FAILED 狀態的使用案例，請擷取與其作用中版本對應的有效模型定義，以建置您的更新請求。如需詳細資訊，請參閱 [資產模型版本](#)。
- c. 執行下列命令來擷取作用中模型定義：

```
aws iotsitewise describe-asset-model-composite-model \
--asset-model-composite-model-id composite-model-id \
--asset-model-id asset-model-id \
--asset-model-version ACTIVE
```

- d. 如需詳細資訊，請參閱 [DescribeAssetModelCompositeModel](#) 操作。
2. 建立名為 `update-custom-composite-model.json` 的檔案，然後將上一個命令的回應複製到檔案。
 3. 從 `update-custom-composite-model.json` 的 JSON 物件移除每個鍵/值對，下列欄位除外：

- `assetModelCompositeModelName`
- `assetModelCompositeModelDescription` (如果有)
- `assetModelCompositeModelProperties` (如果有)

4. 在 `update-custom-composite-model.json` 中，執行下列任何一項：

- 變更 的值 `assetModelCompositeModelName`。
- 新增或移除 `assetModelCompositeModelDescription`，或變更其值。
- 僅適用於內嵌自訂複合模型：變更、新增或移除 中的任何資產模型屬性 `assetModelCompositeModelProperties`。

如需此檔案所需格式的詳細資訊，請參閱 [UpdateAssetModelCompositeModel](#) 的請求語法。

5. 執行下列命令，使用存放在 中的定義更新自訂複合模型 `update-custom-composite-model.json`。以複合模型的 ID 取代 *composite-model-id*，並以其所在的資產模型 ID 取代 *asset-model-id*。

```
aws iotsitewise update-asset-model-composite-model \  
--asset-model-composite-model-id composite-model-id \  
--asset-model-id asset-model-id \  
--cli-input-json file://update-custom-composite-model.json
```

Important

當多個使用者同時更新資產模型時，其他使用者可能會無意中覆寫使用者的變更。若要避免這種情況，您必須定義條件式更新請求。請參閱 [資產模型寫入的樂觀鎖定](#)。

資產模型寫入的樂觀鎖定

更新資產模型時，使用者會執行下列動作：

1. 讀取目前的資產模型定義。
2. 編輯具有必要變更的資產模型定義。
3. 使用新定義更新資產模型。

在有兩個使用者更新模型的情況下，下列是可能的：

- 使用者 A 讀取資產模型 X 定義。
- 使用者 B 讀取資產模型 X 定義並遞交變更，修改 X 的定義。
- 使用者 A 會遞交並覆寫使用者 B 針對資產模型 X 所做的變更，而無需驗證或整合使用者 B 的變更。

樂觀鎖定是用來防止意外覆寫 AWS IoT SiteWise 的機制，如上述案例所示。樂觀鎖定是確保更新或刪除資產模型目前版本的策略，與其目前版本相同 AWS IoT SiteWise。這可防止資產模型寫入被意外更新覆寫。

請依照下列步驟執行具有樂觀鎖定的資產模型寫入：

主題

- [使用樂觀鎖定執行資產模型寫入（主控台）](#)
- [使用樂觀鎖定執行資產模型寫入 \(AWS CLI\)](#)

使用樂觀鎖定執行資產模型寫入（主控台）

以下程序說明如何在 主控台中對資產模型的作用中版本執行具有樂觀鎖定的資產模型寫入。

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇要更新的資產模型或元件模型。
4. 選擇編輯。
5. 在編輯模型頁面上進行變更。
6. 選擇儲存。

Note

有時候，當使用者開始編輯模型，並將所做的編輯儲存至模型時，就會發生一或多個成功的模型更新。

為了確保使用者不會意外覆寫新的成功更新，會拒絕使用者的寫入。主控台會停用儲存按鈕，並提示使用者重新整理編輯模型頁面。使用者必須再次更新模型的新作用中版本。使用者必須執行下列額外步驟：

7. 選擇 Refresh (重新整理)。
8. 再次遵循步驟 5 和 6。

使用樂觀鎖定執行資產模型寫入 (AWS CLI)

以下程序說明如何在 中執行具有樂觀鎖定的資產模型寫入 AWS CLI。

1. 擷取與目前模型定義相關聯的 ETag

ETag 是針對資產模型的每個新表示所產生的唯一字符。呼叫 [DescribeAssetModel](#) API 以擷取目前的資產模型定義，並從回應ETag建立關聯。

在並行更新期間，使用者會執行成功的更新（處於 ACTIVE 狀態的模型）或失敗的更新（處於 FAILED 狀態的模型）。為了確保使用者不會意外覆寫成功更新，您必須從 擷取資產模型的作用中版本[資產模型版本](#)，並取得 ETag值。

執行以下命令：

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id \
--asset-model-version ACTIVE
```

回應會傳回下列結構：

```
{
  "assetModelId": "String",
  "assetModelArn": "String",
  "assetModelName": "String",
  ...
  "eTag": "String"
}
```

Note

您必須擷取資產模型的最新版本及其 ETag，才不會覆寫任何更新。

2. 使用寫入條件執行 UPDATE 和 DELETE 操作

下列資產模型 APIs支援樂觀鎖定：

- [UpdateAssetModel](#)
- [DeleteAssetModel](#)
- [CreateAssetModelCompositeModel](#)
- [UpdateAssetModelCompositeModel](#)
- [DeleteAssetModelCompositeModel](#)

 Note

以下案例使用 UpdateAssetModel API 做為參考。這些條件適用於上述所有操作。

以下案例根據並行控制需求描述不同的寫入條件：

- 執行下列命令，以免覆寫任何成功的更新。自上次讀取作用中版本以來，新的作用中版本不得存在。e-tag 將 取代為在讀取作用中版本時使用的 API 操作中ETag傳回的 。

```
aws iotsitewise update-asset-model \  
  --asset-model-id asset-model-id \  
  --if-match e-tag \  
  --match-for-version-type ACTIVE \  
  --cli-input-json file://model-payload.json
```

- 當模型建立失敗時，作用中版本尚不存在，因為它處於 FAILED 狀態。在遞交變更之前，仍然可以覆寫存在的新作用中版本。當您上次讀取期間沒有作用中版本時，請執行下列命令來不要覆寫新的作用中版本。

```
aws iotsitewise update-asset-model \  
  --asset-model-id asset-model-id \  
  --if-none-match "*" \  
  --match-for-version-type ACTIVE \  
  --cli-input-json file://model-payload.json
```

- 執行下列命令，以避免覆寫任何成功或失敗的更新。此命令會定義寫入條件，以確保自您上次讀取最新版本以來，不會建立最新版本。e-tag 將 取代為在讀取作用中版本時使用的 API 操作中ETag傳回的 。

```
aws iotsitewise update-asset-model \  
  --asset-model-id asset-model-id \  
  --if-match eTag \  
  --match-for-version-type LATEST \  
  --cli-input-json file://model-payload.json
```


如果寫入條件評估為 FALSE，寫入請求會因 `而失敗PreconditionFailedException`。

在 中刪除資產和模型 AWS IoT SiteWise

當您完成資產和模型 AWS IoT SiteWise 時，可以從 刪除這些資產和模型。刪除操作是非同步的，需要一些時間才能傳播 AWS IoT SiteWise。

主題

- [在 中刪除資產 AWS IoT SiteWise](#)
- [在 中刪除資產模型 AWS IoT SiteWise](#)

在 中刪除資產 AWS IoT SiteWise

您可以使用 AWS IoT SiteWise 主控台或 API 來刪除環境中不再需要的資產。刪除資產模型也會刪除所有相關聯的資產和元件模型。不過，請務必注意，刪除資產或模型是永久動作，而且也會移除與已刪除資源相關聯的任何資料。在刪除資產或模型之前，建議您檢閱任何可能受影響的相依性或整合，並確保您擁有任何重要資料的備份。

您必須先將資產與其子資產取消關聯，才能將其從其父資產中刪除。如需詳細資訊，請參閱[關聯和取消關聯資產](#)。如果您使用 AWS Command Line Interface (AWS CLI)，您可以使用 [ListAssociatedAssets](#) 操作來列出資產的子系。

刪除資產時，於變更傳播之前，其狀態會保持為 DELETING。如需詳細資訊，請參閱[資產和模型狀態](#)。刪除資產後，您無法查詢該資產。如果您這樣做，API 會傳回 HTTP 404 回應。

Important

AWS IoT SiteWise 會刪除已刪除資產的所有屬性資料。

主題

- [刪除資產（主控台）](#)
- [刪除資產 \(AWS CLI\)](#)

刪除資產 (主控台)

您可以使用 AWS IoT SiteWise 主控台來刪除資產。

刪除資產 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇要刪除的資產。

Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 如果資產有任何 Associated assets (關聯的資產)，請刪除每個資產。您可以選擇資產的名稱，以導覽至其頁面，您可以在其中刪除資產。
5. 在資產頁面上，選擇 Delete (刪除)。
6. 在刪除資產對話方塊中，執行下列動作：
 - a. 輸入 **Delete** 以確認刪除。
 - b. 選擇 刪除。

刪除資產 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來刪除資產。

使用 [DeleteAsset](#) 操作來刪除資產。指定下列參數：

- `assetId` – 資產的 ID。這是 UUID 格式的實際 ID，`externalId:myExternalId`如果有的話，則為。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs物件](#)。

刪除資產 (AWS CLI)

1. 執行下列命令來列出資產的階層。以 `### ID ### ID ## asset-id`：

```
aws iotsitewise describe-asset --asset-id asset-id
```

此操作會傳回包含資產詳細資料的回應。回應包含具有下列結構的 `assetHierarchies` 清單：

```
{
  ...
  "assetHierarchies": [
    {
      "id": "String",
      "name": "String"
    }
  ],
  ...
}
```

如需詳細資訊，請參閱 [DescribeAsset](#) 操作。

- 針對每個階層，執行下列命令來列出與該階層相關聯的資產子系。將 *asset-id* 取代為資產的 ID 或外部 ID，並將 *hierarchy-id* 取代為階層的 ID 或外部 ID。

```
aws iotsitewise list-associated-assets \
  --asset-id asset-id \
  --hierarchy-id hierarchy-id
```

如需詳細資訊，請參閱 [ListAssociatedAssets](#) 操作。

- 執行下列命令來刪除每個關聯的資產，然後刪除資產。以 *### ID ### ID ## asset-id*。

```
aws iotsitewise delete-asset --asset-id asset-id
```

在中刪除資產模型 AWS IoT SiteWise

您可以使用 AWS IoT SiteWise 主控台或 API 來刪除資產模型。

您必須先刪除從資產模型建立的所有資產，才能刪除資產模型。

刪除資產模型時，於變更傳播之前，其狀態會保持為 DELETING。如需詳細資訊，請參閱 [資產和模型狀態](#)。刪除資產模型後，您無法查詢該資產模型。如果您這樣做，API 會傳回 HTTP 404 回應。

主題

- [刪除資產模型（主控台）](#)
- [刪除資產模型 \(AWS CLI\)](#)

刪除資產模型（主控台）

您可以使用 AWS IoT SiteWise 主控台來刪除資產模型。

刪除資產模型 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇要刪除的資產模型。
4. 如果模型有任何 Assets (資產)，請刪除每個資產。選擇資產的名稱，以導覽至其頁面，您可以在其中刪除資產。如需詳細資訊，請參閱[刪除資產（主控台）](#)。
5. 在模型頁面上，選擇 Delete (刪除)。
6. 在刪除模型對話方塊中，執行下列動作：
 - a. 輸入 **Delete** 以確認刪除。
 - b. 選擇 刪除。

刪除資產模型 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來刪除資產模型。

使用 [DeleteAssetModel](#) 操作來刪除資產模型。指定下列參數：

- `assetModelId` – 資產的 ID。這是 UUID 格式的實際 ID，`externalId:myExternalId`如果有的話，則為。如需詳細資訊，請參閱《AWS IoT SiteWise 使用者指南》中的 [參考具有外部 IDs物件](#)。

刪除資產模型 (AWS CLI)

1. 執行以下命令來列出從模型建立的所有資產。以資產模型的 ID 或外部 ID 取代 *asset-model-id*。

```
aws iotsitewise list-assets --asset-model-id asset-model-id
```

如需詳細資訊，請參閱 [ListAssets](#) 操作。

2. 如果上一個命令傳回模型中的任何資產，請刪除每個資產。如需詳細資訊，請參閱[刪除資產 \(AWS CLI\)](#)。
3. 執行以下命令以刪除資產模型。將 *asset-model-id* 取代為資產模型的 ID 或外部 ID。

```
aws iotsitewise delete-asset-model --asset-model-id asset-model-id
```

Important

若要避免刪除自上次讀取操作後同時更新的資產模型，您必須定義條件式刪除請求。請參閱 [資產模型寫入的樂觀鎖定](#)。

使用資產和模型進行大量操作

若要使用大量資產或資產模型，請使用大量操作將資源大量匯入和匯出到不同的位置。例如，您可以建立資料檔案來定義 Amazon S3 儲存貯體中的資產或資產模型，並使用大量匯入來建立或更新它們 AWS IoT SiteWise。或者，如果您有大量的資產或資產模型 AWS IoT SiteWise，您可以將它們匯出至 Amazon S3。

Note

您可以透過在 AWS IoT TwinMaker API 中 AWS IoT SiteWise 呼叫 操作，在 中執行大量操作。您可以這樣做，而無需設定 AWS IoT TwinMaker 或建立 AWS IoT TwinMaker 工作區。您只需要一個 Amazon S3 儲存貯體，您可以在其中放置 AWS IoT SiteWise 內容。

主題

- [重要概念和術語](#)
- [受支援的功能](#)
- [大量操作先決條件](#)
- [執行大量匯入任務](#)
- [執行大量匯出任務](#)
- [任務進度追蹤和錯誤處理](#)
- [匯入中繼資料範例](#)
- [匯出中繼資料範例](#)
- [AWS IoT SiteWise 中繼資料傳輸任務結構描述](#)

重要概念和術語

AWS IoT SiteWise 大量匯入和匯出功能依賴下列概念和術語：

- 匯入：將資產或資產模型從 Amazon S3 儲存貯體中的檔案移至 的動作 AWS IoT SiteWise。
- 匯出：將資產或資產模型從 移至 Amazon S3 儲存貯 AWS IoT SiteWise 體的動作。
- 來源：您要從中移動內容的起始位置。

例如，Amazon S3 儲存貯體是匯入來源，AWS IoT SiteWise 也是匯出來源。

- 目的地：您要將內容移至其中的所需位置。

例如，Amazon S3 儲存貯體是匯出目的地，AWS IoT SiteWise 也是匯入目的地。

- AWS IoT SiteWise 結構描述：此結構描述用於從中匯入和匯出中繼資料 AWS IoT SiteWise。
- 最上層資源：您可以個別建立或更新 AWS IoT SiteWise 的資源，例如資產或資產模型。
- 子資源：最上層 AWS IoT SiteWise 資源中的巢狀資源。範例包括屬性、階層和複合模型。
- 中繼資料：成功匯入或匯出資源所需的金鑰資訊。中繼資料的範例是資產和資產模型的定義。
- metadataTransferJob：執行 時建立的物件CreateMetadataTransferJob。

受支援的功能

本主題說明執行大量操作時可以執行的動作。大量操作支援下列功能：

- 最上層資源建立：當您匯入的資產或資產模型未定義 ID，或其 ID 與現有資源的 ID 不相符時，將會建立新的資源。
- 頂層資源取代：當您匯入 ID 與已存在的資產或資產模型相符時，它會取代現有的資源。
- 子資源建立、取代或刪除：當您的匯入取代資產或資產模型等頂層資源時，新定義會取代所有子資源，例如屬性、階層或複合模型。

例如，如果您在大量匯入期間更新資產模型，且更新版本定義了原始屬性上不存在的屬性，則會建立新的屬性。如果定義已存在的屬性，則會更新現有的屬性。如果更新的資產模型省略存在於原始的屬性，則會刪除該屬性。

- 不刪除頂層資源：大量操作不會刪除資產或資產模型。大量操作只會建立或更新它們。

大量操作先決條件

本節說明大量操作先決條件，包括在 AWS 服務和本機機器之間交換資源的 AWS Identity and Access Management (IAM) 許可。開始大量操作之前，請先完成下列先決條件：

- 建立 Amazon S3 儲存貯體以存放資源。如需使用 Amazon S3 的詳細資訊，請參閱[什麼是 Amazon S3 ?](#)

IAM 許可

若要執行大量操作，您必須建立具有許可的 AWS Identity and Access Management (IAM) 政策 AWS IoT SiteWise，以允許 Amazon S3 與本機電腦之間的 AWS 資源交換。如需建立自訂 IAM 政策的詳細資訊，請參閱[建立 IAM 政策](#)。

若要執行大量操作，您需要下列政策。

AWS IoT SiteWise 政策

此政策允許存取大量操作所需的 AWS IoT SiteWise API 動作：

```
{
  "Sid": "SiteWiseApiAccess",
  "Effect": "Allow",
  "Action": [
    "iotsitewise:CreateAsset",
    "iotsitewise:CreateAssetModel",
    "iotsitewise:UpdateAsset",
    "iotsitewise:UpdateAssetModel",
    "iotsitewise:UpdateAssetProperty",
    "iotsitewise:ListAssets",
    "iotsitewise:ListAssetModels",
    "iotsitewise:ListAssetProperties",
    "iotsitewise:ListAssetModelProperties",
    "iotsitewise:ListAssociatedAssets",
    "iotsitewise:DescribeAsset",
    "iotsitewise:DescribeAssetModel",
    "iotsitewise:DescribeAssetProperty",
    "iotsitewise:AssociateAssets",
    "iotsitewise:DisassociateAssets",
    "iotsitewise:AssociateTimeSeriesToAssetProperty",
    "iotsitewise:DisassociateTimeSeriesFromAssetProperty",
```

```

        "iotsitewise:BatchPutAssetPropertyValue",
        "iotsitewise:BatchGetAssetPropertyValue",
        "iotsitewise:TagResource",
        "iotsitewise:UntagResource",
        "iotsitewise:ListTagsForResource",
        "iotsitewise:CreateAssetModelCompositeModel",
        "iotsitewise:UpdateAssetModelCompositeModel",
        "iotsitewise:DescribeAssetModelCompositeModel",
        "iotsitewise>DeleteAssetModelCompositeModel",
        "iotsitewise:ListAssetModelCompositeModels",
        "iotsitewise:ListCompositionRelationships",
        "iotsitewise:DescribeAssetCompositeModel"
    ],
    "Resource": "*"
}

```

AWS IoT TwinMaker 政策

此政策允許存取您用來處理大量操作的 AWS IoT TwinMaker API 操作：

```

{
    "Sid": "MetadataTransferJobApiAccess",
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:CreateMetadataTransferJob",
        "iottwinmaker:CancelMetadataTransferJob",
        "iottwinmaker:GetMetadataTransferJob",
        "iottwinmaker:ListMetadataTransferJobs"
    ],
    "Resource": "*"
}

```

Amazon S3 政策

此政策提供 Amazon S3 儲存貯體的存取權，以傳輸大量操作中繼資料。

For a specific Amazon S3 bucket

如果您使用一個特定的儲存貯體來使用大量操作中繼資料，則此政策會提供該儲存貯體的存取權：

```

{
    "Effect": "Allow",

```



```
"Action": [
    "s3:PutObject",
    "s3:GetObject",
    "s3:GetBucketLocation",
    "s3:ListBucket",
    "s3:AbortMultipartUpload",
    "s3:ListBucketMultipartUploads",
    "s3:ListMultipartUploadParts"
],
"Resource": [
    "arn:aws:s3:::bucket name",
    "arn:aws:s3:::bucket name/*"
]
}
```

To allow any Amazon S3 bucket

如果您將使用許多不同的儲存貯體來使用大量操作中繼資料，則此政策可讓您存取任何儲存貯體：

```
{
  "Effect": "Allow",
  "Action": [
    "s3:PutObject",
    "s3:GetObject",
    "s3:GetBucketLocation",
    "s3:ListBucket",
    "s3:AbortMultipartUpload",
    "s3:ListBucketMultipartUploads",
    "s3:ListMultipartUploadParts"
  ],
  "Resource": "*"
}
```

如需對匯入和匯出操作進行疑難排解的資訊，請參閱 [對大量匯入和匯出進行故障診斷](#)。

執行大量匯入任務

大量匯入是將中繼資料移至 AWS IoT SiteWise 工作區的動作。例如，大量匯入可以將中繼資料從本機檔案或 Amazon S3 儲存貯體中的檔案移至 AWS IoT SiteWise 工作區。

步驟 1：準備要匯入的檔案

下載 AWS IoT SiteWise 原生格式檔案以匯入資產和資產模型。如需詳細資訊，請參閱[AWS IoT SiteWise 中繼資料傳輸任務結構描述](#)。

步驟 2：將準備好的檔案上傳至 Amazon S3

將檔案上傳至 Amazon S3。如需詳細資訊，請參閱 [《Amazon Simple Storage Service 使用者指南》中的將檔案上傳至 Amazon S3](#)。

匯入中繼資料（主控台）

您可以使用 AWS IoT SiteWise 主控台 大量匯入中繼資料。遵循 [步驟 1：準備要匯入的檔案](#) 和 [步驟 2：將準備好的檔案上傳至 Amazon S3](#) 準備準備好匯入的檔案。

將資料從 Amazon S3 匯入至 AWS IoT SiteWise 主控台

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 從導覽窗格中選擇大量操作新增。
3. 選擇新增匯入以開始匯入程序。
4. 在匯入中繼資料頁面上：
 - 選擇瀏覽 Amazon S3 以檢視 Amazon S3 儲存貯體和檔案。
 - 導覽至包含預備匯入檔案的 Amazon S3 儲存貯體。
 - 選取要匯入的檔案。
 - 檢閱選取的檔案，然後選擇匯入。
5. SiteWise 中繼資料頁面上的大量操作會在任務進度表中 AWS IoT SiteWise 主控台 顯示新建立的匯入任務。

匯入中繼資料 (AWS CLI)

若要執行匯入動作，請使用下列程序：

將資料從 Amazon S3 匯入至 AWS CLI

1. 依照 建立中繼資料檔案，指定您要匯入的資源[AWS IoT SiteWise 中繼資料傳輸任務結構描述](#)。將此檔案存放在 Amazon S3 儲存貯體中。

如需要匯入的中繼資料檔案範例，請參閱 [匯入中繼資料範例](#)。

2. 現在使用請求內文建立 JSON 檔案。請求內文會指定傳輸任務的來源和目的地。此檔案與上一個步驟的檔案分開。請務必將 Amazon S3 儲存貯體指定為來源和*iotsitewise*目的地。

下列範例顯示請求內文：

```
{
  "metadataTransferJobId": "your-transfer-job-Id",
  "sources": [{
    "type": "s3",
    "s3Configuration": {
      "location": "arn:aws:s3:::amzn-s3-demo-bucket/
your_import_metadata.json"
    }
  ]},
  "destination": {
    "type": "iotSiteWise"
  }
}
```

3. 執行下列 AWS CLI 命令*CreateMetadataTransferJob*來叫用。在此範例中，上一個步驟的請求內文檔案名為 *createMetadataTransferJobExport.json*。

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
  --cli-input-json file://createMetadataTransferJobImport.json
```

這將建立中繼資料傳輸任務，並開始轉移所選資源的程序。

執行大量匯出任務

大量匯出是將中繼資料從 AWS IoT SiteWise 工作區移至 Amazon S3 儲存貯體的動作。

當您將 AWS IoT SiteWise 內容大量匯出至 Amazon S3 時，您可以指定篩選條件來限制您要匯出的特定資產模型和資產。

篩選條件必須在 JSON 請求來源*iotSiteWiseConfiguration*區段中的 區段中指定。

Note

您可以在請求中包含多個篩選條件。大量操作會匯出符合任何篩選條件的資產模型和資產。如果您未提供任何篩選條件，大量操作會匯出所有資產模型和資產。

Example 使用篩選條件請求內文

```
{
  "metadataTransferJobId": "your-transfer-job-id",
  "sources": [
    {
      "type": "iotsitewise",
      "iotSiteWiseConfiguration": {
        "filters": [
          {
            "filterByAssetModel": {
              "assetModelId": "asset model ID"
            }
          },
          {
            "filterByAssetModel": {
              "assetModelId": "asset model ID",
              "includeAssets": true
            }
          },
          {
            "filterByAssetModel": {
              "assetModelId": "asset model ID",
              "includeOffspring": true
            }
          }
        ]
      }
    }
  ],
  "destination": {
    "type": "s3",
    "s3Configuration": {
      "location": "arn:aws:s3:::amzn-s3-demo-bucket"
    }
  }
}
```

匯出中繼資料（主控台）

下列程序說明主控台匯出動作：

在 中建立匯出任務 AWS IoT SiteWise 主控台

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 從導覽窗格中選擇大量操作新增。
3. 選擇新增匯出以開始匯出程序。
4. 在匯出中繼資料頁面上：
 - 輸入匯出任務的名稱。這是用於 Amazon S3 儲存貯體中匯出檔案的名稱。
 - 選擇要匯出的資源，這會設定任務的篩選條件：
 - 匯出所有資產和資產模型。在資產和資產模型上使用篩選條件。
 - 匯出資產。篩選您的資產。
 - 選取要用於匯出篩選條件的資產。
 - (選用) 新增後代或相關聯的資產模型。
 - 匯出資產模型。篩選您的資產模型。
 - 選取要用於匯出篩選條件的資產模型。
 - (選用) 新增後代，或關聯的資產或兩者。
 - 選擇下一步。
 - 導覽至 Amazon S3 儲存貯體：
 - 選擇瀏覽 Amazon S3 以檢視 Amazon S3 儲存貯體和檔案。
 - 導覽至必須放置檔案的 Amazon S3 儲存貯體。
 - 選擇下一步。
 - 檢閱匯出任務，然後選擇匯出。
5. SiteWise 中繼資料頁面上的大量操作會在任務進度表中 AWS IoT SiteWise 主控台 顯示新建立的匯入任務。

如需匯出中繼資料時使用篩選條件的不同方式，請參閱 [匯出中繼資料範例](#)。

匯出中繼資料 (AWS CLI)

下列程序說明 AWS CLI 匯出動作：

從 匯出資料 AWS IoT SiteWise 至 Amazon S3

1. 使用請求內文建立 JSON 檔案。請求內文會指定傳輸任務的來源和目的地。下列範例顯示範例請求內文：

```
{
  "metadataTransferJobId": "your-transfer-job-Id",
  "sources": [{
    "type": "iotsitewise"
  }],
  "destination": {
    "type": "s3",
    "s3Configuration": {
      "location": "arn:aws:s3:::amzn-s3-demo-bucket"
    }
  }
}
```

請務必將 Amazon S3 儲存貯體指定為中繼資料傳輸任務的目的地。

Note

此範例會匯出所有資產模型和資產。若要限制匯出至特定資產模型或資產，您可以在請求內文中包含篩選條件。如需套用匯出篩選條件的詳細資訊，請參閱 [匯出中繼資料範例](#)。

2. 儲存您的請求內文檔案以用於下一個步驟。在此範例中，檔案命名為 `createMetadataTransferJobExport.json`。
3. 執行下列 AWS CLI 命令 `CreateMetadataTransferJob` 來叫用：

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
  --cli-input-json file://createMetadataTransferJobExport.json
```

以 `createMetadataTransferJobExport.json` 您自己的傳輸檔案名稱取代輸入 JSON 檔案。

任務進度追蹤和錯誤處理

大量程序任務需要時間來處理。每個任務都會按照 AWS IoT SiteWise 接收請求的順序進行處理。每個帳戶 one-at-a-time。當任務完成時，佇列中的下一個會自動開始處理。會以非同步方式 AWS IoT SiteWise 解決任務，並在任務進行時更新每個任務的狀態。如果適用，每個任務都有一個狀態欄位，其中包含資源的狀態和錯誤訊息。

狀態可以是下列其中一個值：

- VALIDATING – 驗證任務，包括提交的檔案格式及其內容。
- PENDING – 任務在佇列中。您可以從主控台取消處於此狀態 AWS IoT SiteWise 的任務，但所有其他狀態會持續到結束為止。
- RUNNING – 處理任務。它正在建立和更新匯入檔案定義的資源，或根據所選的匯出任務篩選條件匯出資源。如果取消，則不會刪除此任務匯入的任何資源。如需詳細資訊，請參閱[檢閱任務進度和詳細資訊（主控台）](#)。
- CANCELLING – 任務正在主動取消。
- ERROR – 一或多個資源無法處理。如需詳細資訊，請參閱詳細的任務報告。如需詳細資訊，請參閱[檢查錯誤詳細資訊（主控台）](#)。
- COMPLETED – 任務已完成，沒有錯誤。
- CANCELLED – 任務已取消且未排入佇列。如果您取消RUNNING任務，則不會從中刪除此任務在取消時已匯入的資源 AWS IoT SiteWise。

主題

- [任務進度追蹤](#)
- [檢查的錯誤 AWS IoT SiteWise](#)

任務進度追蹤

檢閱任務進度和詳細資訊（主控台）

請參閱 [匯入中繼資料（主控台）](#) 或 [匯出中繼資料（主控台）](#) 以啟動大量任務。

AWS IoT SiteWise 主控台中的任務進度概觀：

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 從導覽窗格中選擇大量操作新增。
3. AWS IoT SiteWise 主控台中的任務進度表會顯示大量操作任務的清單。
4. 任務類型欄說明是匯出還是匯入任務。匯入日期欄會顯示任務開始的日期。
5. 狀態欄會顯示任務的狀態。您可以選取任務以查看任務的詳細資訊。
6. 選取的任務會在成功時顯示成功，或在任務失敗時顯示失敗清單。每個資源類型也會顯示錯誤描述。

AWS IoT SiteWise 主控台其中的任務詳細資訊概觀：

AWS IoT SiteWise 主控台其中的任務進度表會顯示大量操作任務的清單。

1. 選擇任務以查看更多詳細資訊。
2. 對於匯入任務，Data source ARN代表匯入檔案的 Amazon S3 位置。
3. 對於匯出任務，Data destination ARN代表匯出後檔案的 Amazon S3 位置。
4. Status 和 Status reason提供目前任務的其他詳細資訊。如需詳細資訊，請參閱[任務進度追蹤和錯誤處理](#)。
5. Queued position 代表程序佇列中任務的位置。一次處理一個任務。排入佇列的位置為 1，表示接下來將處理任務。
6. 任務詳細資訊頁面也會顯示任務進度計數。
 - 任務進度計數類型為：
 - i. Total resources – 指出轉移程序中資產的總計數。
 - ii. Succeeded – 表示程序期間成功轉移的資產計數。
 - iii. Failed – 表示程序期間失敗的資產計數。
 - iv. Skipped – 表示程序期間略過的資產計數。
7. 任務狀態為 PENDING或時VALIDATING，會顯示所有任務進度計為 -。這表示正在評估任務進度計數。
8. 的任務狀態RUNNING會顯示Total resources計數，即提交處理的任務。詳細計數 (Succeeded、Failed和 Skipped) 會套用至已處理的資源。詳細計數的總和小於Total resources計數，直到任務的狀態為 COMPLETED或 為止ERROR。
9. 如果任務的狀態為 COMPLETED或 ERROR，則Total resources計數等於詳細計數 (Succeeded、Failed和) 的總和Skipped。
10. 如果任務的狀態為 ERROR，請檢查任務失敗資料表，以取得特定錯誤和失敗的詳細資訊。如需詳細資訊，請參閱[檢查錯誤詳細資訊 \(主控台\)](#)。

檢閱任務進度和詳細資訊 (AWS CLI)

啟動大量操作後，您可以使用下列 API 動作來檢查或更新其狀態：

- 若要擷取特定任務的資訊，請使用 [GetMetadataTransferJob](#) API 動作。

使用 **GetMetadataTransferJob** API 擷取資訊：

1. 建立並執行傳輸任務。呼叫 GetMetadataTransferJob API。

Example AWS CLI 命令：

```
aws iottwinmaker get-metadata-transfer-job \
    --metadata-transfer-job-id your_metadata_transfer_job_id \
    --region your_region
```

2. GetMetadataTransferJob API 會傳回具有下列參數的 MetadataTransferJobProgress 物件：

- succeededCount – 表示流程中成功傳輸的資產計數。
- failedCount – 表示程序期間失敗的資產計數。
- skippedCount – 表示程序期間略過的資產計數。
- totalCount – 表示轉移程序中資產的總計數。

這些參數表示任務進度狀態。如果狀態為 RUNNING，則有助於追蹤仍在處理的資源數量。

如果您遇到結構描述驗證錯誤，或如果 failedCount 大於或等於 1，任務進度狀態會變成 ERROR。任務的完整錯誤報告會放置在 Amazon S3 儲存貯體中。如需詳細資訊，請參閱[檢查的錯誤 AWS IoT SiteWise](#)。

- 若要列出目前的任務，請使用 [ListMetadataTransferJobs](#) API 動作。

使用 JSON 檔案，根據其目前狀態篩選傳回的任務。請參閱下列程序：

1. 若要指定要使用的篩選條件，請建立 AWS CLI 輸入 JSON 檔案。想要使用：

```
{
  "sourceType": "s3",
  "destinationType": "iottwinmaker",
  "filters": [{
    "state": "COMPLETED"
  }]
}
```

如需有效 state 值的清單，請參閱《AWS IoT TwinMaker API 參考指南》中的 [ListMetadataTransferJobsFilter](#)。

2. 在下列 AWS CLI 範例命令中使用 JSON 檔案做為引數：

```
aws iottwinmaker list-metadata-transfer-job --region your_region \  
--cli-input-json file://ListMetadataTransferJobsExample.json
```

- 若要取消任務，請使用 [CancelMetadataTransferJob](#) API 動作。此 API 會取消特定中繼資料傳輸任務，而不會影響任何已匯出或匯入的資源：

```
aws iottwinmaker cancel-metadata-transfer-job \  
--region your_region \  
--metadata-transfer-job-id job-to-cancel-id
```

檢查 的錯誤 AWS IoT SiteWise

檢查錯誤詳細資訊（主控台）

AWS IoT SiteWise 主控台中的錯誤詳細資訊：

- 導覽至 [AWS IoT SiteWise 主控台](#)。
- 如需大量操作任務的清單 AWS IoT SiteWise 主控台，請參閱 中的任務進度表。
- 選取任務以檢視任務詳細資訊。
- 如果任務的狀態為 COMPLETED或 ERROR，則Total resources計數等於詳細計數 (Succeeded、Failed和) 的總和Skipped。
- 如果任務的狀態為 ERROR，請檢查任務失敗資料表以取得特定錯誤和失敗的詳細資訊。
- 任務失敗資料表會顯示任務報告中的內容。Resource type 欄位指出錯誤或失敗的位置，如下所示：
 - 例如，Resource type 欄位中Bulk operations template的驗證錯誤表示匯入範本和中繼資料結構描述檔案格式不相符。如需詳細資訊，請參閱[AWS IoT SiteWise 中繼資料傳輸任務結構描述](#)。
 - Resource type 欄位Asset失敗表示資產未建立，因為與另一個資產發生衝突。如需 AWS IoT SiteWise 資源[錯誤和衝突的相關資訊](#)，請參閱常見錯誤。

檢查錯誤詳細資訊 (AWS CLI)

若要處理和診斷傳輸任務期間產生的錯誤，請參閱下列有關使用 `GetMetadataTransferJob` API 動作的程序：

1. 建立並執行傳輸任務之後，請呼叫 [GetMetadataTransferJob](#)：

```
aws iottwinmaker get-metadata-transfer-job \  
    --metadata-transfer-job-id your_metadata_transfer_job_id \  
    --region us-east-1
```

2. 一旦看到任務狀態變成 `COMPLETED`，您就可以開始驗證任務的結果。
3. 當您呼叫 `GetMetadataTransferJob`，它會傳回名為的物件 [MetadataTransferJobProgress](#)。

`MetadataTransferJobProgress` 物件包含下列參數：

- `failedCount`：表示轉移程序期間失敗的資產計數。
 - `skippedCount`：表示轉移程序期間略過的資產計數。
 - `succeededCount`：表示轉移程序期間成功的資產計數。
 - `totalCount`：表示轉移程序中涉及的資產總數。
4. 此外，API 呼叫會傳回元素 `reportUrl`，其中包含預先簽章的 URL。如果您的轉移任務有任何需要進一步調查的問題，請造訪此 URL。

匯入中繼資料範例

本節說明如何建立中繼資料檔案，以使用單一大量匯入操作匯入資產模型和資產。

大量匯入的範例

您可以使用單一大量匯入操作來匯入許多資產模型和資產。下列範例示範如何建立中繼資料檔案來執行此操作。

在此範例案例中，您有各種工作站點，其中包含工作單元中的工業機器人。

此範例定義兩個資產模型：

- `RobotModel1`：此資產模型代表您在工作站點中擁有的特定類型的機器人。機器人具有測量屬性 `Temperature`。

- **WorkCell**：此資產模型代表其中一個工作站點內的機器人集合。資產模型會定義階層 `robotHierarchyOEM1`，以代表工作區包含機器人的關係。

此範例也會定義一些資產：

- **WorkCell1**：您波士頓站點內的工作區
- **RobotArm123456**：該工作區內的機器人
- **RobotArm987654**：該工作區內的另一個機器人

下列 JSON 中繼資料檔案會定義這些資產模型和資產。使用此中繼資料執行大量匯入會在其中建立資產模型和資產 AWS IoT SiteWise，包括其階層關係。

用於匯入的中繼資料檔案

```
{
  "assetModels": [
    {
      "assetModelExternalId": "Robot.OEM1.3536",
      "assetModelName": "RobotModel1",
      "assetModelProperties": [
        {
          "dataType": "DOUBLE",
          "externalId": "Temperature",
          "name": "Temperature",
          "type": {
            "measurement": {
              "processingConfig": {
                "forwardingConfig": {
                  "state": "ENABLED"
                }
              }
            }
          },
          "unit": "fahrenheit"
        }
      ],
    },
    {
      "assetModelExternalId": "ISA95.WorkCell",
      "assetModelName": "WorkCell",
      "assetModelProperties": [],
    }
  ]
}
```

```

        "assetModelHierarchies": [
            {
                "externalId": "workCellHierarchyWithOEM1Robot",
                "name": "robotHierarchyOEM1",
                "childAssetModelExternalId": "Robot.OEM1.3536"
            }
        ]
    },
    ],
    "assets": [
        {
            "assetExternalId": "Robot.OEM1.3536.123456",
            "assetName": "RobotArm123456",
            "assetModelExternalId": "Robot.OEM1.3536"
        },
        {
            "assetExternalId": "Robot.OEM1.3536.987654",
            "assetName": "RobotArm987654",
            "assetModelExternalId": "Robot.OEM1.3536"
        },
        {
            "assetExternalId": "BostonSite.Area1.Line1.WorkCell1",
            "assetName": "WorkCell1",
            "assetModelExternalId": "ISA95.WorkCell",
            "assetHierarchies": [
                {
                    "externalId": "workCellHierarchyWithOEM1Robot",
                    "childAssetExternalId": "Robot.OEM1.3536.123456"
                },
                {
                    "externalId": "workCellHierarchyWithOEM1Robot",
                    "childAssetExternalId": "Robot.OEM1.3536.987654"
                }
            ]
        }
    ]
}

```

模型和資產初始加入的範例

在此範例案例中，您有各種工作站點，其中包含公司中的工業機器人。

此範例定義多個資產模型：

- Sample_Enterprise – 此資產模型代表網站所屬的公司。資產模型會定義階層 Enterprise to Site，以代表網站與企業的關係。
- Sample_Site – 此資產模型代表公司內的製造站點。資產模型會定義階層 Site to Line，以代表行與網站的關係。
- Sample_Welding Line – 此資產模型代表工作現場內的組裝線。資產模型會定義階層 Line to Robot，以代表機器人與線條的關係。
- Sample_Welding Robot – 此資產模型代表您工作站點中特定類型的機器人。

此範例也會根據資產模型定義資產。

- Sample_AnyCompany Motor – 此資產是從Sample_Enterprise資產模型建立的。
- Sample_Chicago – 此資產是從Sample_Site資產模型建立的。
- Sample_Welding Line 1 – 此資產是從Sample_Welding Line資產模型建立的。
- Sample_Welding Robot 1 – 此資產是從Sample_Welding Robot資產模型建立的。
- Sample_Welding Robot 2 – 此資產是從Sample_Welding Robot資產模型建立的。

下列 JSON 中繼資料檔案會定義這些資產模型和資產。使用此中繼資料執行大量匯入會在其中建立資產模型和資產 AWS IoT SiteWise，包括其階層關係。

用於加入資產和模型以進行匯入的 JSON 檔案

```
{
  "assetModels": [
    {
      "assetModelExternalId": "External_Id_Welding_Robot",
      "assetModelName": "Sample_Welding Robot",
      "assetModelProperties": [
        {
          "dataType": "STRING",
          "externalId": "External_Id_Welding_Robot_Serial_Number",
          "name": "Serial Number",
          "type": {
            "attribute": {
              "defaultValue": "-"
            }
          },
          "unit": "-"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "dataType": "DOUBLE",
      "externalId": "External_Id_Welding_Robot_Cycle_Count",
      "name": "CycleCount",
      "type": {
        "measurement": {}
      },
      "unit": "EA"
    },
    {
      "dataType": "DOUBLE",
      "externalId": "External_Id_Welding_Robot_Joint_1_Current",
      "name": "Joint 1 Current",
      "type": {
        "measurement": {}
      },
      "unit": "Amps"
    },
    {
      "dataType": "DOUBLE",
      "externalId": "External_Id_Welding_Robot_Joint_1_Max_Current",
      "name": "Max Joint 1 Current",
      "type": {
        "metric": {
          "expression": "max(joint1current)",
          "variables": [
            {
              "name": "joint1current",
              "value": {
                "propertyExternalId":
"External_Id_Welding_Robot_Joint_1_Current"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "5m"
            }
          }
        }
      },
      "unit": "Amps"
    }
  ]
}

```

```

    ]
  },
  {
    "assetModelExternalId": "External_Id_Welding_Line",
    "assetModelName": "Sample_Welding Line",
    "assetModelProperties": [
      {
        "dataType": "DOUBLE",
        "externalId": "External_Id_Welding_Line_Availability",
        "name": "Availability",
        "type": {
          "measurement": {}
        },
        "unit": "%"
      }
    ],
    "assetModelHierarchies": [
      {
        "externalId": "External_Id_Welding_Line_T0_Robot",
        "name": "Line to Robot",
        "childAssetModelExternalId": "External_Id_Welding_Robot"
      }
    ]
  },
  {
    "assetModelExternalId": "External_Id_Site",
    "assetModelName": "Sample_Site",
    "assetModelProperties": [
      {
        "dataType": "STRING",
        "externalId": "External_Id_Site_Street_Address",
        "name": "Street Address",
        "type": {
          "attribute": {
            "defaultValue": "-"
          }
        },
        "unit": "-"
      }
    ],
    "assetModelHierarchies": [
      {
        "externalId": "External_Id_Site_T0_Line",
        "name": "Site to Line",

```



```

        "childAssetModelExternalId": "External_Id_Welding_Line"
    }
]
},
{
    "assetModelExternalId": "External_Id_Enterprise",
    "assetModelName": "Sample_Enterprise",
    "assetModelProperties": [
        {
            "dataType": "STRING",
            "name": "Company Name",
            "externalId": "External_Id_Enterprise_Company_Name",
            "type": {
                "attribute": {
                    "defaultValue": "-"
                }
            },
            "unit": "-"
        }
    ],
    "assetModelHierarchies": [
        {
            "externalId": "External_Id_Enterprise_T0_Site",
            "name": "Enterprise to Site",
            "childAssetModelExternalId": "External_Id_Site"
        }
    ]
}
],
"assets": [
    {
        "assetExternalId": "External_Id_Welding_Robot_1",
        "assetName": "Sample_Welding Robot 1",
        "assetModelExternalId": "External_Id_Welding_Robot",
        "assetProperties": [
            {
                "externalId": "External_Id_Welding_Robot_Serial_Number",
                "attributeValue": "S1000"
            },
            {
                "externalId": "External_Id_Welding_Robot_Cycle_Count",
                "alias": "AnyCompany/Chicago/Welding Line/S1000/Count"
            }
        ]
    }
]

```

```

        "externalId": "External_Id_Welding_Robot_Joint_1_Current",
        "alias": "AnyCompany/Chicago/Welding Line/S1000/1/Current"
    }
]
},
{
    "assetExternalId": "External_Id_Welding_Robot_2",
    "assetName": "Sample_Welding Robot 2",
    "assetModelExternalId": "External_Id_Welding_Robot",
    "assetProperties": [
        {
            "externalId": "External_Id_Welding_Robot_Serial_Number",
            "attributeValue": "S2000"
        },
        {
            "externalId": "External_Id_Welding_Robot_Cycle_Count",
            "alias": "AnyCompany/Chicago/Welding Line/S2000/Count"
        },
        {
            "externalId": "External_Id_Welding_Robot_Joint_1_Current",
            "alias": "AnyCompany/Chicago/Welding Line/S2000/1/Current"
        }
    ]
},
{
    "assetExternalId": "External_Id_Welding_Line_1",
    "assetName": "Sample_Welding Line 1",
    "assetModelExternalId": "External_Id_Welding_Line",
    "assetProperties": [
        {
            "externalId": "External_Id_Welding_Line_Availability",
            "alias": "AnyCompany/Chicago/Welding Line/Availability"
        }
    ],
    "assetHierarchies": [
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_1"
        },
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_2"
        }
    ]
}
]

```

```

    },
    {
      "assetExternalId": "External_Id_Site_Chicago",
      "assetName": "Sample_Chicago",
      "assetModelExternalId": "External_Id_Site",
      "assetHierarchies": [
        {
          "externalId": "External_Id_Site_T0_Line",
          "childAssetExternalId": "External_Id_Welding_Line_1"
        }
      ]
    },
  ],
  {
    "assetExternalId": "External_Id_Enterprise_AnyCompany",
    "assetName": "Sample_AnyEnterprise Motor",
    "assetModelExternalId": "External_Id_Enterprise",
    "assetHierarchies": [
      {
        "externalId": "External_Id_Enterprise_T0_Site",
        "childAssetExternalId": "External_Id_Site_Chicago"
      }
    ]
  }
]
}
}

```

下列螢幕擷取畫面是在您執行先程式碼範例 AWS IoT SiteWise 主控台 後，顯示於 中的模型。

IoT SiteWise > Models

Models (4) Refresh Create component model Create asset model

Assets represent industrial devices and processes that send data streams to SiteWise. Models are structures that enforce a specific model of properties and hierarchies for all instances of each asset. You must create every asset from a model.

Name	Status	Model type	Date created	Date modified
Sample_Enterprise	ACTIVE	Asset model	November 10, 2023 at 11:22:13 (UT...)	November 10, 202...
Sample_Site	ACTIVE	Asset model	November 10, 2023 at 11:21:57 (UT...)	November 10, 202...
Sample_Welding Line	ACTIVE	Asset model	November 10, 2023 at 11:21:40 (UT...)	November 10, 202...
Sample_Welding Robot	ACTIVE	Asset model	November 10, 2023 at 11:21:24 (UT...)	November 10, 202...

下列螢幕擷取畫面是在您執行先程式碼範例 AWS IoT SiteWise 主控台 後，顯示於 中的模型、資產和階層。

IoT SiteWise > Assets

Assets (1) Refresh Create asset

Assets represent industrial devices and processes that send data streams to SiteWise. Models are structures that enforce a specific model of properties and hierarchies for all instances of each asset. You must create every asset from a model.

Filter top level assets

Name	Description	Status	Date created	Date modified
Sample_AnyEnterprise Motor		ACTIVE	November 10, 2023 at 11:23:06 (UTC-5:00)	November 10, 2023 at 11:23:06 (UTC-5:00)
Sample_Chicago		ACTIVE	November 10, 2023 at 11:22:57 (UTC-5:00)	November 10, 2023 at 11:22:57 (UTC-5:00)
Sample_Welding Line 1		ACTIVE	November 10, 2023 at 11:22:48 (UTC-5:00)	November 10, 2023 at 11:22:48 (UTC-5:00)
Sample_Welding Robot 1		ACTIVE	November 10, 2023 at 11:22:39 (UTC-5:00)	November 10, 2023 at 11:22:39 (UTC-5:00)
Sample_Welding Robot 2		ACTIVE	November 10, 2023 at 11:22:30 (UTC-5:00)	November 10, 2023 at 11:22:30 (UTC-5:00)

加入其他資產的範例

此範例定義要匯入至您帳戶中現有資產模型的其他資產：

- Sample_Welding Line 2 – 此資產是從Sample_Welding Line資產模型建立的。
- Sample_Welding Robot 3– 此資產是從Sample_Welding Robot資產模型建立的。
- Sample_Welding Robot 4– 此資產是從Sample_Welding Robot資產模型建立的。

若要建立此範例的初始資產，請參閱 [模型和資產初始加入的範例](#)。

下列 JSON 中繼資料檔案會定義這些資產模型和資產。使用此中繼資料執行大量匯入會在其中建立資產模型和資產 AWS IoT SiteWise，包括其階層關係。

加入其他資產的 JSON 檔案

```
{
  "assets": [
    {
      "assetExternalId": "External_Id_Welding_Robot_3",
      "assetName": "Sample_Welding Robot 3",
      "assetModelExternalId": "External_Id_Welding_Robot",
      "assetProperties": [
        {
          "externalId": "External_Id_Welding_Robot_Serial_Number",
          "attributeValue": "S3000"
        },
        {
```

```

        "externalId": "External_Id_Welding_Robot_Cycle_Count",
        "alias": "AnyCompany/Chicago/Welding Line/S3000/Count"
    },
    {
        "externalId": "External_Id_Welding_Robot_Joint_1_Current",
        "alias": "AnyCompany/Chicago/Welding Line/S3000/1/Current"
    }
]
},
{
    "assetExternalId": "External_Id_Welding_Robot_4",
    "assetName": "Sample_Welding Robot 4",
    "assetModelExternalId": "External_Id_Welding_Robot",
    "assetProperties": [
        {
            "externalId": "External_Id_Welding_Robot_Serial_Number",
            "attributeValue": "S4000"
        },
        {
            "externalId": "External_Id_Welding_Robot_Cycle_Count",
            "alias": "AnyCompany/Chicago/Welding Line/S4000/Count"
        },
        {
            "externalId": "External_Id_Welding_Robot_Joint_1_Current",
            "alias": "AnyCompany/Chicago/Welding Line/S4000/1/Current"
        }
    ]
},
{
    "assetExternalId": "External_Id_Welding_Line_1",
    "assetName": "Sample_Welding Line 1",
    "assetModelExternalId": "External_Id_Welding_Line",
    "assetHierarchies": [
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_1"
        },
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_2"
        },
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_3"
        }
    ]
}

```

```

        }
    ]
},
{
    "assetExternalId": "External_Id_Welding_Line_2",
    "assetName": "Sample_Welding Line 2",
    "assetModelExternalId": "External_Id_Welding_Line",
    "assetHierarchies": [
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_4"
        }
    ]
},
{
    "assetExternalId": "External_Id_Site_Chicago",
    "assetName": "Sample_Chicago",
    "assetModelExternalId": "External_Id_Site",
    "assetHierarchies": [
        {
            "externalId": "External_Id_Site_T0_Line",
            "childAssetExternalId": "External_Id_Welding_Line_1"
        },
        {
            "externalId": "External_Id_Site_T0_Line",
            "childAssetExternalId": "External_Id_Welding_Line_2"
        }
    ]
}
]
}

```

下列螢幕擷取畫面是在您執行先前的程式碼範例 AWS IoT SiteWise 主控台 後，顯示於 中的模型、資產和階層。

IoT SiteWise > Assets

Assets (1) Refresh Create asset

Assets represent industrial devices and processes that send data streams to SiteWise. Models are structures that enforce a specific model of properties and hierarchies for all instances of each asset. You must create every asset from a model.

Filter top level assets

Name	Description	Status	Date created	Date modified
Sample_AnyCompany Motor		ACTIVE	November 09, 2023 at 19:18:05 (UTC-5:00)	November 09, 2023 at 19:18:05 (UTC-5:00)
Sample_Chicago		ACTIVE	November 09, 2023 at 19:17:56 (UTC-5:00)	November 09, 2023 at 19:17:56 (UTC-5:00)
Sample_Welding Line 1		ACTIVE	November 09, 2023 at 19:17:48 (UTC-5:00)	November 09, 2023 at 19:17:48 (UTC-5:00)
Sample_Welding Robot 2		ACTIVE	November 09, 2023 at 19:17:39 (UTC-5:00)	November 09, 2023 at 19:51:05 (UTC-5:00)
Sample_Welding Robot 3		ACTIVE	November 09, 2023 at 20:40:02 (UTC-5:00)	November 09, 2023 at 20:40:02 (UTC-5:00)
Sample_Welding Robot 1		ACTIVE	November 09, 2023 at 19:17:30 (UTC-5:00)	November 09, 2023 at 19:51:05 (UTC-5:00)
Sample_Welding Line 2		ACTIVE	November 09, 2023 at 20:40:20 (UTC-5:00)	November 09, 2023 at 20:40:20 (UTC-5:00)
Sample_Welding Robot 4		ACTIVE	November 09, 2023 at 20:40:11 (UTC-5:00)	November 09, 2023 at 20:40:11 (UTC-5:00)

加入新屬性的範例

此範例定義現有資產模型的新屬性。請參閱 [加入其他資產的範例](#) 以加入其他資產和模型。

- Joint 1 Temperature – 此屬性會新增至Sample_Welding Robot資產模型。此新屬性也會傳播到從資產模型建立的每個Sample_Welding Robot資產。

若要將新屬性新增至現有資產模型，請參閱下列 JSON 中繼資料檔案範例。如 JSON 所示，整個現有Sample_Welding Robot資產模型定義必須與新屬性一起提供。如果未提供現有定義中的整個屬性清單，會 AWS IoT SiteWise 刪除省略的屬性。

加入新屬性的 JSON 檔案

此範例Joint 1 Temperature會將新的屬性新增至資產模型。

```
{
  "assetModels": [
    {
      "assetModelExternalId": "External_Id_Welding_Robot",
      "assetModelName": "Sample_Welding Robot",
      "assetModelProperties": [
        {
          "dataType": "STRING",
          "externalId": "External_Id_Welding_Robot_Serial_Number",
```

```

        "name": "Serial Number",
        "type": {
            "attribute": {
                "defaultValue": "-"
            }
        },
        "unit": "-"
    },
    {
        "dataType": "DOUBLE",
        "externalId": "External_Id_Welding_Robot_Cycle_Count",
        "name": "CycleCount",
        "type": {
            "measurement": {}
        },
        "unit": "EA"
    },
    {
        "dataType": "DOUBLE",
        "externalId": "External_Id_Welding_Robot_Joint_1_Current",
        "name": "Joint 1 Current",
        "type": {
            "measurement": {}
        },
        "unit": "Amps"
    },
    {
        "dataType": "DOUBLE",
        "externalId": "External_Id_Welding_Robot_Joint_1_Max_Current",
        "name": "Max Joint 1 Current",
        "type": {
            "metric": {
                "expression": "max(joint1current)",
                "variables": [
                    {
                        "name": "joint1current",
                        "value": {
                            "propertyExternalId":
"External_Id_Welding_Robot_Joint_1_Current"
                        }
                    }
                ],
                "window": {
                    "tumbling": {

```



```

        "interval": "5m"
      }
    }
  },
  "unit": "Amps"
},
{
  "dataType": "DOUBLE",
  "externalId": "External_Id_Welding_Robot_Joint_1_Temperature",
  "name": "Joint 1 Temperature",
  "type": {
    "measurement": {}
  },
  "unit": "degC"
}
]
}
]
}

```

管理資料串流的範例

此範例顯示管理與資產屬性相關聯之資料串流的兩種方式。重新命名資產屬性別名時，資產屬性資料串流中目前存放的歷史資料有兩個選項。

- 選項一：保留目前的資料串流並單獨重新命名別名，以便使用新的別名存取歷史資料。

在 JSON 中繼資料檔案範例中，具有 ID 的資產屬性會將其別名 `External_Id_Welding_Robot_Cycle_Count` 變更為 `AnyCompany/Chicago/Welding Line/S3000/Count-Updated`。此資產屬性的歷史資料在此變更後保持不變。

- 選項二：將新的資料串流指派給資產屬性，該屬性可透過新的別名存取。舊資料串流及其歷史資料仍可使用舊別名存取，但不會與任何資產屬性相關聯。

在 JSON 中繼資料檔案範例中，具有 ID 的資產屬性會將其別名 `External_Id_Welding_Robot_Joint_1_Current` 變更為 `AnyCompany/Chicago/Welding Line/S4999/1/Current`。此時會出現額外的值，`retainDataOnAliasChange` 並將設定為 `False`。使用此設定，原始資料串流會與資產屬性取消關聯，並建立新的資料串流，其中包含沒有歷史資料。

若要使用原始歷史資料存取舊資料串流，請在 `中` AWS Console Home，前往資料串流頁面並搜尋舊別名 `AnyCompany/Chicago/Welding Line/S3000/1/Current`。

更新屬性別名的 JSON 檔案

```
{
  "assetExternalId": "External_Id_Welding_Robot_3",
  "assetName": "Sample_Welding Robot 3",
  "assetModelExternalId": "External_Id_Welding_Robot",
  "assetProperties": [
    {
      "externalId": "External_Id_Welding_Robot_Serial_Number",
      "attributeValue": "S3000"
    },
    {
      "externalId": "External_Id_Welding_Robot_Cycle_Count",
      "alias": "AnyCompany/Chicago/Welding Line/S3000/Count-Updated"
    },
    {
      "externalId": "External_Id_Welding_Robot_Joint_1_Current",
      "alias": "AnyCompany/Chicago/Welding Line/S4999/1/Current",
      "retainDataOnAliasChange": "FALSE"
    }
  ]
}
```

匯出中繼資料範例

當您將 AWS IoT SiteWise 內容大量匯出至 Amazon S3 時，您可以指定篩選條件來限制您要匯出的特定資產模型和資產。

您可以在請求內文 `iotSiteWiseConfiguration` 區段的 `sources` 區段中指定篩選條件。

Note

您可以包含多個篩選條件。大量操作將匯出符合任何篩選條件的任何資產模型或資產。如果您不提供任何篩選條件，則操作會匯出所有資產模型和資產。

```
{
```

```

    "metadataTransferJobId": "your-transfer-job-id",
    "sources": [{
      "type": "iotsitewise",
      "iotSiteWiseConfiguration": {
        "filters": [{
          list of filters
        }]
      }
    }],
    "destination": {
      "type": "s3",
      "s3Configuration": {
        "location": "arn:aws:s3:::amzn-s3-demo-bucket"
      }
    }
  }
}

```

依資產模型篩選

您可以篩選特定的資產模型。您也可以包含使用該模型的所有資產，或其階層中的所有資產模型。您無法同時包含資產和階層。

如需有關階層的詳細資訊，請參閱[定義資產模型階層](#)。

Asset model

此篩選條件包含指定的資產模型：

```

"filterByAssetModel": {
  "assetModelId": "asset model ID"
}

```

Asset model and its assets

此篩選條件包含指定的資產模型，以及使用該資產模型的所有資產：

```

"filterByAssetModel": {
  "assetModelId": "asset model ID",
  "includeAssets": true
}

```

Asset model and its hierarchy

此篩選條件包含指定的資產模型，以及其階層中所有相關聯的資產模型：

```
"filterByAssetModel": {  
  "assetModelId": "asset model ID",  
  "includeOffspring": true  
}
```

依資產篩選

您可以篩選特定資產。您也可以在其階層中包含其資產模型，或所有相關資產。您無法同時包含資產模型和階層。

如需有關階層的詳細資訊，請參閱[定義資產模型階層](#)。

Asset

此篩選條件包含指定的資產：

```
"filterByAsset": {  
  "assetId": "asset ID"  
}
```

Asset and its asset model

此篩選條件包含指定的資產，以及其使用的資產模型：

```
"filterByAsset": {  
  "assetId": "asset ID",  
  "includeAssetModel": true  
}
```

Asset and its hierarchy

此篩選條件包含指定的資產，以及其階層中所有相關聯的資產：

```
"filterByAsset": {  
  "assetId": "asset ID",  
  "includeOffspring": true  
}
```

}

AWS IoT SiteWise 中繼資料傳輸任務結構描述

執行您自己的大量匯入和匯出操作時，請使用 AWS IoT SiteWise 中繼資料傳輸任務結構描述進行參考：

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "IoTSiteWise",
  "description": "Metadata transfer job resource schema for IoTSiteWise",
  "definitions": {
    "Name": {
      "type": "string",
      "minLength": 1,
      "maxLength": 256,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "Description": {
      "type": "string",
      "minLength": 1,
      "maxLength": 2048,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "ID": {
      "type": "string",
      "minLength": 36,
      "maxLength": 36,
      "pattern": "^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$"
    },
    "ExternalId": {
      "type": "string",
      "minLength": 2,
      "maxLength": 128,
      "pattern": "[a-zA-Z0-9_][a-zA-Z_\\-0-9\\.]*[a-zA-Z0-9_]+"
    },
    "AttributeValue": {
      "description": "The value of the property attribute.",
      "type": "string",
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "PropertyUnit": {
```

```

    "description": "The unit of measure (such as Newtons or RPM) of the asset
property.",
    "type": "string",
    "minLength": 1,
    "maxLength": 256,
    "pattern": "[^\\u0000-\\u001F\\u007F]+"
  },
  "PropertyAlias": {
    "description": "The property alias that identifies the property.",
    "type": "string",
    "minLength": 1,
    "maxLength": 1000,
    "pattern": "[^\\u0000-\\u001F\\u007F]+"
  },
  "AssetProperty": {
    "description": "The asset property's definition, alias, unit, and notification
state.",
    "type": "object",
    "additionalProperties": false,
    "anyOf": [
      {
        "required": [
          "id"
        ]
      },
      {
        "required": [
          "externalId"
        ]
      }
    ],
    "properties": {
      "id": {
        "description": "The ID of the asset property.",
        "$ref": "#/definitions/ID"
      },
      "externalId": {
        "description": "The ExternalID of the asset property.",
        "$ref": "#/definitions/ExternalId"
      },
      "alias": {
        "$ref": "#/definitions/PropertyAlias"
      },
      "unit": {

```

```

    "$ref": "#/definitions/PropertyUnit"
  },
  "attributeValue": {
    "$ref": "#/definitions/AttributeValue"
  },
  "retainDataOnAliasChange": {
    "type": "string",
    "default": "TRUE",
    "enum": [
      "TRUE",
      "FALSE"
    ]
  },
  "propertyNotificationState": {
    "description": "The MQTT notification state (ENABLED or DISABLED) for this
asset property.",
    "type": "string",
    "enum": [
      "ENABLED",
      "DISABLED"
    ]
  }
}
},
"AssetHierarchy": {
  "description": "A hierarchy specifies allowed parent/child asset relationships.",
  "type": "object",
  "additionalProperties": false,
  "anyOf": [
    {
      "required": [
        "id",
        "childAssetId"
      ]
    },
    {
      "required": [
        "externalId",
        "childAssetId"
      ]
    },
    {
      "required": [
        "id",

```

```
        "childAssetExternalId"
      ]
    },
    {
      "required": [
        "externalId",
        "childAssetExternalId"
      ]
    }
  ],
  "properties": {
    "id": {
      "description": "The ID of a hierarchy in the parent asset's model.",
      "$ref": "#/definitions/ID"
    },
    "externalId": {
      "description": "The ExternalID of a hierarchy in the parent asset's model.",
      "$ref": "#/definitions/ExternalId"
    },
    "childAssetId": {
      "description": "The ID of the child asset to be associated.",
      "$ref": "#/definitions/ID"
    },
    "childAssetExternalId": {
      "description": "The ExternalID of the child asset to be associated.",
      "$ref": "#/definitions/ExternalId"
    }
  }
},
"Tag": {
  "type": "object",
  "additionalProperties": false,
  "required": [
    "key",
    "value"
  ],
  "properties": {
    "key": {
      "type": "string"
    },
    "value": {
      "type": "string"
    }
  }
}
```



```

    },
    "AssetModelType": {
      "type": "string",
      "default": null,
      "enum": [
        "ASSET_MODEL",
        "COMPONENT_MODEL"
      ]
    },
  },
  "AssetModelCompositeModel": {
    "description": "Contains a composite model definition in an asset model. This composite model definition is applied to all assets created from the asset model.",
    "type": "object",
    "additionalProperties": false,
    "anyOf": [
      {
        "required": [
          "id"
        ]
      },
      {
        "required": [
          "externalId"
        ]
      }
    ],
    "required": [
      "name",
      "type"
    ],
  },
  "properties": {
    "id": {
      "description": "The ID of the asset model composite model.",
      "$ref": "#/definitions/ID"
    },
    "externalId": {
      "description": "The ExternalID of the asset model composite model.",
      "$ref": "#/definitions/ExternalId"
    },
    "parentId": {
      "description": "The ID of the parent asset model composite model.",
      "$ref": "#/definitions/ID"
    },
    "parentExternalId": {

```

```

        "description": "The ExternalID of the parent asset model composite model.",
        "$ref": "#/definitions/ExternalId"
    },
    "composedAssetModelId": {
        "description": "The ID of the composed asset model.",
        "$ref": "#/definitions/ID"
    },
    "composedAssetModelExternalId": {
        "description": "The ExternalID of the composed asset model.",
        "$ref": "#/definitions/ExternalId"
    },
    "description": {
        "description": "A description for the asset composite model.",
        "$ref": "#/definitions/Description"
    },
    "name": {
        "description": "A unique, friendly name for the asset composite model.",
        "$ref": "#/definitions/Name"
    },
    "type": {
        "description": "The type of the composite model. For alarm composite models,
this type is AWS/ALARM.",
        "$ref": "#/definitions/Name"
    },
    "properties": {
        "description": "The property definitions of the asset model.",
        "type": "array",
        "items": {
            "$ref": "#/definitions/AssetModelProperty"
        }
    }
},
"AssetModelProperty": {
    "description": "Contains information about an asset model property.",
    "type": "object",
    "additionalProperties": false,
    "anyOf": [
        {
            "required": [
                "id"
            ]
        }
    ],
    {

```

```

        "required": [
            "externalId"
        ]
    },
],
"required": [
    "name",
    "dataType",
    "type"
],
"properties": {
    "id": {
        "description": "The ID of the asset model property.",
        "$ref": "#/definitions/ID"
    },
    "externalId": {
        "description": "The ExternalID of the asset model property.",
        "$ref": "#/definitions/ExternalId"
    },
    "name": {
        "description": "The name of the asset model property.",
        "$ref": "#/definitions/Name"
    },
    "dataType": {
        "description": "The data type of the asset model property.",
        "$ref": "#/definitions/DataType"
    },
    "dataTypeSpec": {
        "description": "The data type of the structure for this property.",
        "$ref": "#/definitions/Name"
    },
    "unit": {
        "description": "The unit of the asset model property, such as Newtons or
RPM.",
        "type": "string",
        "minLength": 1,
        "maxLength": 256,
        "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "type": {
        "description": "The property type",
        "$ref": "#/definitions/PropertyType"
    }
}
}

```

```

    },
    "DataType": {
      "type": "string",
      "enum": [
        "STRING",
        "INTEGER",
        "DOUBLE",
        "BOOLEAN",
        "STRUCT"
      ]
    },
    "PropertyType": {
      "description": "Contains a property type, which can be one of attribute,
measurement, metric, or transform.",
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "attribute": {
          "$ref": "#/definitions/Attribute"
        },
        "transform": {
          "$ref": "#/definitions/Transform"
        },
        "metric": {
          "$ref": "#/definitions/Metric"
        },
        "measurement": {
          "$ref": "#/definitions/Measurement"
        }
      }
    },
    "Attribute": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "defaultValue": {
          "type": "string",
          "pattern": "[^\\u0000-\\u001F\\u007F]+"
        }
      }
    },
    "Transform": {
      "type": "object",
      "additionalProperties": false,

```

```

    "required": [
      "expression",
      "variables"
    ],
    "properties": {
      "expression": {
        "description": "The mathematical expression that defines the transformation function.",
        "type": "string",
        "minLength": 1,
        "maxLength": 1024
      },
      "variables": {
        "description": "The list of variables used in the expression.",
        "type": "array",
        "items": {
          "$ref": "#/definitions/ExpressionVariable"
        }
      },
      "processingConfig": {
        "$ref": "#/definitions/TransformProcessingConfig"
      }
    }
  },
  "TransformProcessingConfig": {
    "description": "The processing configuration for the given transform property.",
    "type": "object",
    "additionalProperties": false,
    "required": [
      "computeLocation"
    ],
    "properties": {
      "computeLocation": {
        "description": "The compute location for the given transform property.",
        "$ref": "#/definitions/ComputeLocation"
      },
      "forwardingConfig": {
        "description": "The forwarding configuration for a given property.",
        "$ref": "#/definitions/ForwardingConfig"
      }
    }
  },
  "Metric": {
    "type": "object",

```

```

    "additionalProperties": false,
    "required": [
      "expression",
      "variables",
      "window"
    ],
    "properties": {
      "expression": {
        "description": "The mathematical expression that defines the metric
aggregation function.",
        "type": "string",
        "minLength": 1,
        "maxLength": 1024
      },
      "variables": {
        "description": "The list of variables used in the expression.",
        "type": "array",
        "items": {
          "$ref": "#/definitions/ExpressionVariable"
        }
      },
      "window": {
        "description": "The window (time interval) over which AWS IoT SiteWise
computes the metric's aggregation expression",
        "$ref": "#/definitions/MetricWindow"
      },
      "processingConfig": {
        "$ref": "#/definitions/MetricProcessingConfig"
      }
    }
  },
  "MetricProcessingConfig": {
    "description": "The processing configuration for the metric.",
    "type": "object",
    "additionalProperties": false,
    "required": [
      "computeLocation"
    ],
    "properties": {
      "computeLocation": {
        "description": "The compute location for the given metric property.",
        "$ref": "#/definitions/ComputeLocation"
      }
    }
  }
}

```

```
    },
    "ComputeLocation": {
      "type": "string",
      "enum": [
        "EDGE",
        "CLOUD"
      ]
    },
    "ForwardingConfig": {
      "type": "object",
      "additionalProperties": false,
      "required": [
        "state"
      ],
      "properties": {
        "state": {
          "type": "string",
          "enum": [
            "ENABLED",
            "DISABLED"
          ]
        }
      }
    },
    "MetricWindow": {
      "description": "Contains a time interval window used for data aggregate computations (for example, average, sum, count, and so on).",
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "tumbling": {
          "description": "The tumbling time interval window.",
          "type": "object",
          "additionalProperties": false,
          "required": [
            "interval"
          ],
          "properties": {
            "interval": {
              "description": "The time interval for the tumbling window.",
              "type": "string",
              "minLength": 2,
              "maxLength": 23
            }
          }
        }
      }
    },
```

```

        "offset": {
            "description": "The offset for the tumbling window.",
            "type": "string",
            "minLength": 2,
            "maxLength": 25
        }
    }
},
"ExpressionVariable": {
    "type": "object",
    "additionalProperties": false,
    "required": [
        "name",
        "value"
    ],
    "properties": {
        "name": {
            "description": "The friendly name of the variable to be used in the
expression.",
            "type": "string",
            "minLength": 1,
            "maxLength": 64,
            "pattern": "^[a-z][a-z0-9_]*$"
        },
        "value": {
            "description": "The variable that identifies an asset property from which to
use values.",
            "$ref": "#/definitions/VariableValue"
        }
    }
},
"VariableValue": {
    "type": "object",
    "additionalProperties": false,
    "anyOf": [
        {
            "required": [
                "propertyId"
            ]
        },
        {
            "required": [

```



```

        "propertyExternalId"
      ]
    }
  ],
  "properties": {
    "propertyId": {
      "$ref": "#/definitions/ID"
    },
    "propertyExternalId": {
      "$ref": "#/definitions/ExternalId"
    },
    "hierarchyId": {
      "$ref": "#/definitions/ID"
    },
    "hierarchyExternalId": {
      "$ref": "#/definitions/ExternalId"
    }
  }
},
"Measurement": {
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "processingConfig": {
      "$ref": "#/definitions/MeasurementProcessingConfig"
    }
  }
},
"MeasurementProcessingConfig": {
  "type": "object",
  "additionalProperties": false,
  "required": [
    "forwardingConfig"
  ],
  "properties": {
    "forwardingConfig": {
      "description": "The forwarding configuration for the given measurement property.",
      "$ref": "#/definitions/ForwardingConfig"
    }
  }
},
"AssetModelHierarchy": {
  "description": "Contains information about an asset model hierarchy.",

```

```
"type": "object",
"additionalProperties": false,
"anyOf": [
  {
    "required": [
      "id",
      "childAssetModelId"
    ]
  },
  {
    "required": [
      "id",
      "childAssetModelExternalId"
    ]
  },
  {
    "required": [
      "externalId",
      "childAssetModelId"
    ]
  },
  {
    "required": [
      "externalId",
      "childAssetModelExternalId"
    ]
  }
],
"required": [
  "name"
],
"properties": {
  "id": {
    "description": "The ID of the asset model hierarchy.",
    "$ref": "#/definitions/ID"
  },
  "externalId": {
    "description": "The ExternalID of the asset model hierarchy.",
    "$ref": "#/definitions/ExternalId"
  },
  "name": {
    "description": "The name of the asset model hierarchy.",
    "$ref": "#/definitions/Name"
  }
},
```

```

    "childAssetModelId": {
      "description": "The ID of the asset model. All assets in this hierarchy must
be instances of the child AssetModelId asset model.",
      "$ref": "#/definitions/ID"
    },
    "childAssetModelExternalId": {
      "description": "The ExternalID of the asset model. All assets in this
hierarchy must be instances of the child AssetModelId asset model.",
      "$ref": "#/definitions/ExternalId"
    }
  },
  "AssetModel": {
    "type": "object",
    "additionalProperties": false,
    "anyOf": [
      {
        "required": [
          "assetModelId"
        ]
      },
      {
        "required": [
          "assetModelExternalId"
        ]
      }
    ],
    "required": [
      "assetModelName"
    ],
    "properties": {
      "assetModelId": {
        "description": "The ID of the asset model.",
        "$ref": "#/definitions/ID"
      },
      "assetModelExternalId": {
        "description": "The ID of the asset model.",
        "$ref": "#/definitions/ExternalId"
      },
      "assetModelName": {
        "description": "A unique, friendly name for the asset model.",
        "$ref": "#/definitions/Name"
      },
      "assetModelDescription": {

```

```

    "description": "A description for the asset model.",
    "$ref": "#/definitions/Description"
  },
  "assetModelType": {
    "description": "The type of the asset model.",
    "$ref": "#/definitions/AssetModelType"
  },
  "assetModelProperties": {
    "description": "The property definitions of the asset model.",
    "type": "array",
    "items": {
      "$ref": "#/definitions/AssetModelProperty"
    }
  },
  "assetModelCompositeModels": {
    "description": "The composite asset models that are part of this asset model.
Composite asset models are asset models that contain specific properties.",
    "type": "array",
    "items": {
      "$ref": "#/definitions/AssetModelCompositeModel"
    }
  },
  "assetModelHierarchies": {
    "description": "The hierarchy definitions of the asset model. Each hierarchy
specifies an asset model whose assets can be children of any other assets created from
this asset model.",
    "type": "array",
    "items": {
      "$ref": "#/definitions/AssetModelHierarchy"
    }
  },
  "tags": {
    "description": "A list of key-value pairs that contain metadata for the asset
model.",
    "type": "array",
    "items": {
      "$ref": "#/definitions/Tag"
    }
  }
},
"Asset": {
  "type": "object",
  "additionalProperties": false,

```

```
"anyOf": [
  {
    "required": [
      "assetId",
      "assetModelId"
    ]
  },
  {
    "required": [
      "assetExternalId",
      "assetModelId"
    ]
  },
  {
    "required": [
      "assetId",
      "assetModelExternalId"
    ]
  },
  {
    "required": [
      "assetExternalId",
      "assetModelExternalId"
    ]
  }
],
"required": [
  "assetName"
],
"properties": {
  "assetId": {
    "description": "The ID of the asset",
    "$ref": "#/definitions/ID"
  },
  "assetExternalId": {
    "description": "The external ID of the asset",
    "$ref": "#/definitions/ExternalId"
  },
  "assetModelId": {
    "description": "The ID of the asset model from which to create the asset.",
    "$ref": "#/definitions/ID"
  },
  "assetModelExternalId": {
```

```

        "description": "The ExternalID of the asset model from which to create the
asset.",
        "$ref": "#/definitions/ExternalId"
    },
    "assetName": {
        "description": "A unique, friendly name for the asset.",
        "$ref": "#/definitions/Name"
    },
    "assetDescription": {
        "description": "A description for the asset",
        "$ref": "#/definitions/Description"
    },
    "assetProperties": {
        "type": "array",
        "items": {
            "$ref": "#/definitions/AssetProperty"
        }
    },
    "assetHierarchies": {
        "type": "array",
        "items": {
            "$ref": "#/definitions/AssetHierarchy"
        }
    },
    "tags": {
        "description": "A list of key-value pairs that contain metadata for the
asset.",
        "type": "array",
        "uniqueItems": false,
        "items": {
            "$ref": "#/definitions/Tag"
        }
    }
},
"additionalProperties": false,
"properties": {
    "assetModels": {
        "type": "array",
        "uniqueItems": false,
        "items": {
            "$ref": "#/definitions/AssetModel"
        }
    }
}

```

```
    },  
    "assets": {  
      "type": "array",  
      "uniqueItems": false,  
      "items": {  
        "$ref": "#/definitions/Asset"  
      }  
    }  
  }  
}
```

在 中使用警示監控資料 AWS IoT SiteWise

您可以設定資料的警示，在設備或程序以次最佳化方式執行時提醒您的團隊。機器或程序的最佳效能，表示特定指標的值應該在高低限制範圍內。當這些指標超出其操作範圍時，則必須通知設備操作員以便修正問題。使用警示來快速識別問題並通知操作員，以最大限度地提高設備和程序的效能。

主題

- [警示類型](#)
- [警示狀態](#)
- [警示狀態屬性](#)
- [在中定義資產模型的警示 AWS IoT SiteWise](#)
- [在中設定資產警示 AWS IoT SiteWise](#)
- [在中回應警示 AWS IoT SiteWise](#)
- [在中擷取外部警示狀態 AWS IoT SiteWise](#)

警示類型

您可以定義在 AWS 雲端中偵測到的警示，以及使用外部程序偵測到的警示。AWS IoT SiteWise 支援下列類型的警示：

- AWS IoT Events 警示

Note

終止支援通知：將於 2026 AWS 年 5 月 20 日結束對的支援 AWS IoT Events。2026 年 5 月 20 日之後，您將無法再存取 AWS IoT Events 主控台或 AWS IoT Events 資源。如需詳細資訊，請參閱[AWS IoT Events 終止支援](#)。

AWS IoT Events 警示是偵測 in. AWS IoT Events AWS IoT SiteWise sends 資產屬性值到警示模型的警示 AWS IoT Events。然後，AWS IoT Events 傳送警示狀態至 AWS IoT SiteWise。您可以設定選項，例如警示偵測到時，以及警示狀態變更時要通知誰。您也可以定義警示狀態變更時所發生[AWS IoT Events 的動作](#)。

中的警示 AWS IoT Events 是警示模型的執行個體。警示模型會指定警示的閾值和嚴重性、警示狀態變更時該怎麼做等。當您設定警示模型的每個特徵時，您可以從警示監控的資產模型指定屬性屬性。當 AWS IoT Events 評估警示的特性時，所有以資產模型為基礎的資產都會使用 屬性的值。如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[使用警示](#)。

您可以在 AWS IoT Events 警示變更狀態時回應警示。例如，您可以在警示變成作用中時確認警示或將其休眠。您也可以啟用、停用和重設警示。

SiteWise Monitor 使用者可以在 SiteWise Monitor 入口網站中視覺化、設定和回應 AWS IoT Events 警示。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[使用警示進行監控](#)。

Note

AWS IoT Events 費用適用於評估這些警示，以及在 AWS IoT SiteWise 和 之間傳輸資料 AWS IoT Events。如需詳細資訊，請參閱 [AWS IoT Events 定價](#)。

• 外部警示

外部警示是您在 外部評估的警示 AWS IoT SiteWise。如果您有報告警示狀態的資料來源，請使用外部警示。外部警示包含您擷取警示狀態資料的測量屬性。

當外部警示變更狀態時，您無法確認或休眠外部警示。

SiteWise Monitor 使用者可以在 SiteWise Monitor 入口網站中查看外部警示的狀態，但無法設定或回應這些警示。

AWS IoT SiteWise 不會評估外部警示的狀態。

警示狀態

工業警示包含其監控之設備或程序狀態的相關資訊，以及（選用）運算子回應警示狀態的相關資訊。

當您定義 AWS IoT Events 警示時，您可以指定是否啟用確認流程。確認流程預設為啟用。當您啟用此選項時，運算子可以確認警示，並記下警示的詳細資訊或他們為了解決該警示而採取的動作。如果運算子在處於非作用中狀態之前未確認作用中警示，則警示會遭到鎖定。門鎖狀態表示警示變成作用中且未確認，因此操作員需要檢查設備或程序，並確認門鎖警示。

警示具有下列狀態：

- 正常 (Normal) – 警示已啟用但非作用中。工業程序或設備可如預期運作。
- 作用中 (Active) – 警示處於作用中狀態。工業程序或設備超出其操作範圍，需要注意。
- 已確認 (Acknowledged) – 操作員已確認警示的狀態。

此狀態僅適用於您啟用確認流程的警示。

- 鎖定 (Latched) – 警示傳回正常，但處於作用中狀態，且沒有操作員認可。工業程序或設備需要操作員的注意，才能將警示重設為正常。

此狀態僅適用於您啟用確認流程的警示。

- Snoozed (SnoozeDisabled) – 警示已停用，因為運算子將警示休眠。運算子定義警示休眠的持續時間。在該持續時間之後，警示會返回正常狀態。
- 停用 (Disabled) – 警示已停用且無法偵測。

警示狀態屬性

AWS IoT SiteWise 會將警示狀態資料儲存為序列化至字串的 JSON 物件。此物件包含警示的狀態和其他資訊，例如運算子回應動作和警示評估的規則。

您可以依警示狀態屬性的名稱和結構類型來識別警示狀態屬性AWS/ALARM_STATE。如需詳細資訊，請參閱[在 中定義資產模型的警示 AWS IoT SiteWise](#)。

警示狀態資料物件包含下列資訊：

stateName

警示的狀態。如需詳細資訊，請參閱[警示狀態](#)。

資料類型：STRING

customerAction

(選用) 包含運算子對警示回應相關資訊的物件。操作員可以啟用、停用、確認和暫停警示。當他們這樣做時，警示狀態資料包含其回應，以及他們可以在回應時離開的注意事項。此物件包含下列資訊：

actionName

運算子用來回應警示的動作名稱。此值包含下列其中一個字串：

- ENABLE

- DISABLE
- SNOOZE
- ACKNOWLEDGE
- RESET

資料類型：STRING

enable

(選用) 當運算子啟用警示customerAction時，存在於 中的物件。當 運算子啟用警示時，警示狀態會變更為 Normal。此物件包含下列資訊：

note

(選用) 客戶啟用警示時離開的備註。

資料類型：STRING

長度上限：128 個字元

disable

(選用) 當運算子停用警示customerAction時，存在於 中的物件。當 運算子啟用警示時，警示狀態會變更為 Disabled。此物件包含下列資訊：

note

(選用) 客戶停用警示時離開的備註。

資料類型：STRING

長度上限：128 個字元

acknowledge

(選用) 當運算子確認警示customerAction時，存在於 中的物件。當 運算子啟用警示時，警示狀態會變更為 Acknowledged。此物件包含下列資訊：

note

(選用) 客戶確認警示時離開的備註。

資料類型：STRING

長度上限：128 個字元

snooze

(選用) 當運算子休眠警示customerAction時，存在於 中的物件。當 運算子啟用警示時，警示狀態會變更為 SnoozeDisabled。此物件包含下列資訊：

snoozeDuration

運算子延遲警示的持續時間，以秒為單位。在此持續時間之後，警示會變更為 Normal 狀態。

資料類型：INTEGER

note

(選用) 客戶休眠警示時離開的注意事項。

資料類型：STRING

長度上限：128 個字元

ruleEvaluation

(選用) 包含評估警示之規則相關資訊的物件。此物件包含下列資訊：

simpleRule

包含簡單規則相關資訊的物件，該規則會將屬性值與閾值與比較運算子進行比較。此物件包含下列資訊：

inputProperty

此警示評估的 屬性值。

資料類型：DOUBLE

operator

此警示用來比較 屬性與閾值的比較運算子。此值包含下列其中一個字串：

- < – 小於
- <= – 小於或等於
- == – 等於
- != – 不等於
- >= – 大於或等於
- > – 大於

資料類型：STRING

threshold

此警示比較屬性值的閾值。

資料類型：DOUBLE

在 中定義資產模型的警示 AWS IoT SiteWise

資產模型可推動工業資料和警示的標準化。您可以在資產模型上定義警示定義，以根據資產模型標準化所有資產的警示。

您可以使用複合資產模型來定義資產模型的警示。複合資產模型是資產模型，可標準化另一個資產模型上的特定屬性集。複合資產模型可確保資產模型上存在特定屬性。警示具有類型、狀態和（選用）來源屬性，因此警示複合模型會強制執行這些屬性存在。

每個複合資產模型都有定義該複合模型屬性的類型。警示複合模型會定義警示類型、警示狀態和（選用）警示來源的屬性。當您從具有複合模型的資產模型建立資產時，資產會包含複合模型的屬性，以及您在資產模型中指定的屬性。

複合模型中的每個屬性都必須具有識別其複合模型類型的名稱。複合模型屬性支援具有複雜資料類型的屬性。這些屬性具有STRUCT資料類型和dataTypeSpec特性，可指定屬性的複雜資料類型。複雜資料類型屬性包含序列化為字串的JSON資料。

警示複合模型具有下列屬性。每個屬性都必須具有識別此類型複合模型的名稱。

警示類型

警示的類型。請指定下列其中一項：

- IOT_EVENTS – AWS IoT Events alarm. AWS IoT SiteWise sends data 至 AWS IoT Events 以評估此警示的狀態。您必須指定警示來源屬性，才能定義此 AWS IoT Events 警示定義的警示模型。

Note

終止支援通知：將於 2026 AWS 年 5 月 20 日結束對 的支援 AWS IoT Events。2026 年 5 月 20 日之後，您將無法再存取 AWS IoT Events 主控台或 AWS IoT Events 資源。如需詳細資訊，請參閱[AWS IoT Events 終止支援](#)。

- EXTERNAL – 外部警示。您可以將警示的狀態擷取為測量。

屬性名稱：AWS/ALARM_TYPE

屬性類型：[屬性](#)

資料類型：STRING

警示狀態

警示狀態的時間序列資料。這是序列化為字串的物件，其中包含狀態和警示的其他資訊。如需詳細資訊，請參閱[警示狀態屬性](#)。

屬性名稱：AWS/ALARM_STATE

屬性類型：[測量](#)

資料類型：STRUCT

資料結構類型：AWS/ALARM_STATE

警示來源

(選用) 評估警示狀態之資源的 Amazon Resource Name (ARN)。對於 AWS IoT Events 警示，這是警示模型的 ARN。

屬性名稱：AWS/ALARM_SOURCE

屬性類型：[屬性](#)

資料類型：STRING

Example 警示複合模型範例

下列資產模型代表具有警示以監控其溫度的鍋爐。會將溫度資料 AWS IoT SiteWise 傳送至 AWS IoT Events 以偵測警示。

```
{
  "assetModelName": "Boiler",
  "assetModelDescription": "A boiler that alarms when its temperature exceeds its limit.",
  "assetModelProperties": [
    {
      "name": "Temperature",
      "dataType": "DOUBLE",
      "unit": "Celsius",
      "type": {
```

```

        "measurement": {}
    }
},
{
    "name": "High Temperature",
    "dataType": "DOUBLE",
    "unit": "Celsius",
    "type": {
        "attribute": {
            "defaultValue": "105.0"
        }
    }
}
],
"assetModelCompositeModels": [
    {
        "name": "BoilerTemperatureHighAlarm",
        "type": "AWS/ALARM",
        "properties": [
            {
                "name": "AWS/ALARM_TYPE",
                "dataType": "STRING",
                "type": {
                    "attribute": {
                        "defaultValue": "IOT_EVENTS"
                    }
                }
            },
            {
                "name": "AWS/ALARM_STATE",
                "dataType": "STRUCT",
                "dataTypeSpec": "AWS/ALARM_STATE",
                "type": {
                    "measurement": {}
                }
            },
            {
                "name": "AWS/ALARM_SOURCE",
                "dataType": "STRING",
                "type": {
                    "attribute": {}
                }
            }
        ]
    }
]

```

```
}  
]  
}
```

主題

- [中的警示通知需求 AWS IoT SiteWise](#)
- [定義的 AWS IoT Events 警示 AWS IoT SiteWise](#)
- [在中定義外部警示 AWS IoT SiteWise](#)

中的警示通知需求 AWS IoT SiteWise

AWS IoT Events 會在您的帳戶中使用 AWS Lambda 函數 AWS 來傳送警示通知。您必須在與警示相同的 AWS 區域中建立此 Lambda 函數，才能啟用警示通知。此 Lambda 函數使用 [Amazon Simple Notification Service \(Amazon SNS\)](#) 傳送文字通知和 [Amazon Simple Email Service \(Amazon SES\)](#) 傳送電子郵件通知。建立 AWS IoT Events 警示時，您可以設定警示用來傳送通知的通訊協定和設定。

Note

終止支援通知：將於 2026 AWS 年 5 月 20 日結束對的支援 AWS IoT Events。2026 年 5 月 20 日之後，您將無法再存取 AWS IoT Events 主控台或 AWS IoT Events 資源。如需詳細資訊，請參閱[AWS IoT Events 終止支援](#)。

AWS IoT Events 提供 AWS CloudFormation 堆疊範本，可讓您用來在帳戶中建立此 Lambda 函數。如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[警示通知 Lambda 函數](#)。

定義的 AWS IoT Events 警示 AWS IoT SiteWise

Note

終止支援通知：將於 2026 AWS 年 5 月 20 日結束對的支援 AWS IoT Events。2026 年 5 月 20 日之後，您將無法再存取 AWS IoT Events 主控台或 AWS IoT Events 資源。如需詳細資訊，請參閱[AWS IoT Events 終止支援](#)。

當您建立 AWS IoT Events 警示時，AWS IoT SiteWise 會將資產屬性值傳送至 AWS IoT Events，以評估 alarm。AWS IoT Events alarm 定義的狀態，取決於您在 中定義的警示模型 AWS IoT Events。若

要在資產模型上定義 AWS IoT Events 警示，您可以定義警示複合模型，將 AWS IoT Events 警示模型指定為其警示來源屬性。

AWS IoT Events 警示取決於警示閾值和警示通知設定等輸入。您可以將這些輸入定義為資產模型上的屬性。然後，您可以根據模型在每個資產上自訂這些輸入。AWS IoT SiteWise 主控台可以為您建立這些屬性。如果您使用 AWS CLI 或 API 定義警示，您必須在資產模型上手動定義這些屬性。

您也可以定義警示偵測到時發生的其他動作，例如自訂警示通知動作。例如，您可以設定將推送通知傳送至 Amazon SNS 主題的動作。如需您可以定義之動作的詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[使用其他 AWS 服務](#)。

當您更新或刪除資產模型時，AWS IoT SiteWise 可以檢查 中的 AWS IoT Events 警示模型是否正在監控與此資產模型相關聯的資產屬性。這可防止您刪除 AWS IoT Events 警示目前正在使用的資產屬性。若要在 中啟用此功能 AWS IoT SiteWise，您必須擁有 `iotevents:ListInputRoutings` 許可。此許可允許 AWS IoT SiteWise 呼叫 支援的 [ListInputRoutings](#) API 操作 AWS IoT Events。如需詳細資訊，請參閱 [\(選用\) ListInputRoutings 許可](#)。

Note

警示通知功能不適用於中國（北京）區域。

主題

- [定義 AWS IoT Events 警示 \(AWS IoT SiteWise 主控台\)](#)
- [定義 AWS IoT Events 警示 \(AWS IoT Events 主控台\)](#)
- [定義 AWS IoT Events 警示 \(AWS CLI\)](#)

定義 AWS IoT Events 警示 (AWS IoT SiteWise 主控台)

您可以使用 AWS IoT SiteWise 主控台在現有資產模型上定義 AWS IoT Events 警示。若要在新的資產模型上定義 AWS IoT Events 警示，請建立資產模型，然後完成這些步驟。如需詳細資訊，請參閱[在 中建立資產模型 AWS IoT SiteWise](#)。

Important

每個警示都需要一個屬性，指定要與警示比較的閾值。您必須先在資產模型上定義閾值屬性，才能定義警示。

假設您想要定義警示，以偵測風力發電機何時超過其最大風速等級 50 mph 的範例。定義警示之前，您必須定義預設值為 的屬性 (最大風速)50。

在資產模型上定義 AWS IoT Events 警示

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇要為其定義警示的資產模型。
4. 選擇警示索引標籤。
5. 選擇新增警示。
6. 在警示類型選項區段中，選擇AWS IoT Events 警示。
7. 在警示詳細資訊區段中，執行下列動作：
 - a. 輸入警示的名稱。
 - b. (選用) 輸入警示的描述。
8. 在閾值定義區段中，您可以定義警示何時偵測到和警示的嚴重性。請執行下列操作：
 - a. 選取警示偵測到的屬性。每次此屬性收到新值時，都會將值傳送至 AWS IoT SiteWise AWS IoT Events，以評估警示的狀態。
 - b. 選取要用來比較 屬性與閾值的運算子。您可以從以下選項中選擇：
 - < 小於
 - <= 小於或等於
 - == 等於
 - != 不相等
 - >= 大於或等於
 - > 大於
 - c. 針對值，選取要用作閾值的屬性屬性。會 AWS IoT Events 比較 屬性的值與此屬性的值。
 - d. 輸入警示的嚴重性。使用您的團隊了解的數字來反映此警示的嚴重性。
9. (選用) 在通知設定 - 選用區段中，執行下列動作：
 - a. 選擇作用中。

 Note

如果您選擇非作用中，您和您的團隊將不會收到任何警示通知。

- b. 針對收件人，選擇收件人。

 Important

您可以傳送警示通知給 AWS IAM Identity Center 使用者。若要使用此功能，您必須啟用 IAM Identity Center。您 AWS 一次只能在一個區域中啟用 IAM Identity Center。這表示您只能在啟用 IAM Identity Center 的區域中定義警示通知。如需詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》中的「[入門](#)」。

- c. 針對通訊協定，從下列選項中選擇：

- 電子郵件和文字 – 警示會以簡訊和電子郵件訊息通知 IAM Identity Center 使用者。
- 電子郵件 – 警示會以電子郵件訊息通知 IAM Identity Center 使用者。
- 文字 – 警示會以簡訊通知 IAM Identity Center 使用者。

- d. 針對寄件者，選擇寄件者。

 Important

您必須在 Amazon Simple Email Service (Amazon SES) 中驗證寄件者電子郵件地址。如需詳細資訊，請參閱《Amazon Simple Email Service 開發人員指南》中的[驗證電子郵件地址身分](#)。

10. 在預設資產狀態區段中，您可以為從此資產模型建立的警示設定預設狀態。

 Note

您可以為稍後步驟中從此資產模型建立的資產啟用或停用此警示。

11. 在進階設定區段中，您可以設定許可、其他通知設定、警示狀態動作、SiteWise Monitor 中的警示模型，以及確認流程。

 Note

AWS IoT Events 警示需要下列服務角色：

- AWS IoT Events 擔任將警示狀態值傳送至其中的角色 AWS IoT SiteWise。
- AWS IoT Events 擔任將資料傳送至 Lambda 的角色。只有在警示傳送通知時，您才需要此角色。

在許可區段中，執行下列動作：

- 對於 AWS IoT Events 角色，請使用現有角色或建立具有所需許可的角色。此角色需要 `iotsitewise:BatchPutAssetPropertyValue` 許可和允許 `iotevents.amazonaws.com` 擔任該角色的信任關係。
- 對於 AWS IoT Events Lambda 角色，請使用現有角色或建立具有所需許可的角色。此角色需要 `lambda:InvokeFunction` 和 `sso-directory:DescribeUser` 許可，以及允許 `iotevents.amazonaws.com` 擔任角色的信任關係。

12. (選用) 在其他通知設定區段中，執行下列動作：

- 對於收件人屬性，您可以定義屬性，其值會指定通知的收件人。您可以選擇 IAM Identity Center 使用者做為收件人。

您可以在資產模型上建立屬性或使用現有的屬性。

- 如果您選擇建立新的收件人屬性，請指定收件人屬性名稱和收件人預設值 - 屬性為選用。
- 如果您選擇使用現有的收件人屬性，請在收件人屬性名稱中選擇屬性。警示會使用您選擇的屬性的預設值。

您可以覆寫從此資產模型建立的每個資產的預設值。

- 對於自訂訊息屬性，您可以定義屬性，其值指定除了預設狀態變更訊息之外要傳送的自訂訊息。例如，您可以指定訊息，以協助您的團隊了解如何處理此警示。

您可以選擇在資產模型上建立屬性或使用現有的屬性。

- 如果您選擇建立新的自訂訊息屬性，請為屬性指定自訂訊息屬性名稱和自訂訊息預設值 - 選用。

- 如果您選擇使用現有的自訂訊息屬性，請在自訂訊息屬性名稱中選擇屬性。警示會使用您選擇的屬性的預設值。

您可以覆寫從此資產模型建立的每個資產的預設值。

c. 針對管理您的 Lambda 函數，請執行下列其中一項操作：

- 若要 AWS IoT SiteWise 建立新的 Lambda 函數，請選擇從 AWS 受管範本建立新的 Lambda。
- 若要使用現有的 Lambda 函數，請選擇使用現有的 Lambda，然後選擇函數的名稱。

如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[管理警示通知](#)。

13. (選用) 在設定狀態動作區段中，執行下列動作：

- a. 選擇編輯動作。
- b. 在新增警示狀態動作下，新增動作。和 選擇儲存。

您最多可以新增 10 個動作。

AWS IoT Events 可以在警示作用中時執行動作。您可以定義內建動作來使用計時器或設定變數，或將資料傳送至其他 AWS 資源。如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[支援的動作](#)。

14. (選用) 在 SiteWise Monitor 中的管理警示模型下 - 選用，選擇作用中或非作用中。

使用此選項可讓您在 SiteWise Monitors 中更新警示模型。此選項預設為啟用。

15. 在確認流程下，選擇作用中或非作用中。如需確認流程的詳細資訊，請參閱[警示狀態](#)。

16. 選擇新增警示。

Note

AWS IoT SiteWise 主控台會發出多個 API 請求，將警示新增至資產模型。當您選擇新增警示時，主控台會開啟顯示這些 API 請求進度的對話方塊。請保留在此頁面上，直到每個 API 請求成功或 API 請求失敗為止。如果請求失敗，請關閉對話方塊，修正問題，然後選擇新增警示再試一次。

定義 AWS IoT Events 警示 (AWS IoT Events 主控台)

您可以使用 AWS IoT Events 主控台來定義現有資產模型上的 AWS IoT Events 警示。若要在新的資產模型上定義 AWS IoT Events 警示，請建立資產模型，然後完成這些步驟。如需詳細資訊，請參閱[在中建立資產模型 AWS IoT SiteWise](#)。

Important

每個警示都需要一個屬性，指定要與警示比較的閾值。您必須先在資產模型上定義閾值屬性，才能定義警示。

假設您想要定義警示，以偵測風力發電機何時超過其最大風速等級 50 mph 的範例。定義警示之前，您必須定義預設值為 的屬性 (最大風速)50。

在資產模型上定義 AWS IoT Events 警示

1. 導覽至 [AWS IoT Events 主控台](#)。
2. 在導覽窗格中，選擇警示模型。
3. 選擇建立警示模型。
4. 輸入警示的名稱。
5. (選用) 輸入警示的描述。
6. 在警示目標區段中，執行下列動作：
 - a. 針對目標選項，選擇AWS IoT SiteWise 資產屬性。
 - b. 選擇您要為其新增警示的資產模型。
7. 在閾值定義區段中，您可以定義警示何時偵測到和警示的嚴重性。請執行下列操作：
 - a. 選取警示偵測到的屬性。每次此屬性收到新值時，都會將值傳送至 AWS IoT SiteWise AWS IoT Events，以評估警示的狀態。
 - b. 選取要用來比較 屬性與閾值的運算子。您可以從以下選項中選擇：
 - < 小於
 - <= 小於或等於
 - == 等於
 - != 不相等
 - >= 大於或等於

- > 大於
 - c. 針對值，選取要用作閾值的屬性屬性。會將屬性的值與此屬性的值 AWS IoT Events 進行比較。
 - d. 輸入警示的嚴重性。使用您的團隊了解的數字來反映此警示的嚴重性。
8. (選用) 在通知設定 - 選用區段中，執行下列動作：
- a. 針對通訊協定，從下列選項中選擇：
 - 電子郵件和文字 – 警示會以簡訊和電子郵件訊息通知 IAM Identity Center 使用者。
 - 電子郵件 – 警示會以電子郵件訊息通知 IAM Identity Center 使用者。
 - 文字 – 警示會以簡訊通知 IAM Identity Center 使用者。
 - b. 針對寄件者，選擇寄件者。

 Important

您必須在 Amazon Simple Email Service (Amazon SES) 中驗證寄件者電子郵件地址。如需詳細資訊，請參閱 [《Amazon Simple Email Service 開發人員指南》中的在 Amazon SES 中驗證電子郵件地址](#)。

- c. 在收件人屬性中選擇屬性 - 選用。警示會使用您選擇的屬性的預設值。
 - d. 選擇自訂訊息屬性中的屬性 - 選用。警示會使用您選擇的屬性的預設值。
9. 在執行個體區段中，指定此警示的預設狀態。您可以為稍後步驟中從此資產模型建立的所有資產啟用或停用此警示。
10. 在進階設定設定中，您可以設定許可、其他通知設定、警示狀態動作、SiteWise Monitor 中的警示模型，以及確認流程。

 Note

AWS IoT Events 警示需要下列服務角色：

- AWS IoT Events 擔任將警示狀態值傳送至其中的角色 AWS IoT SiteWise。
- AWS IoT Events 擔任將資料傳送至 Lambda 的角色。只有在警示傳送通知時，您才需要此角色。

- a. 在確認流程區段中，選擇啟用或停用。如需確認流程的詳細資訊，請參閱 [警示狀態](#)。

- b. 在許可區段中，執行下列動作：
- 對於AWS IoT Events 角色，請使用現有角色或建立具有所需許可的角色。此角色需要*ioticsitewise:BatchPutAssetPropertyValue*許可和允許 *iotevents.amazonaws.com* 擔任該角色的信任關係。
 - 對於 Lambda 角色，請使用現有角色或建立具有必要許可的角色。此角色需要 *lambda:InvokeFunction*和 *sso-directory:DescribeUser*許可，以及允許 *iotevents.amazonaws.com*擔任該角色的信任關係。
- c. （選用）在其他通知設定窗格中，執行下列動作：
- 針對管理您的 Lambda 函數，請執行下列其中一項操作：
 - 若要 AWS IoT Events 建立新的 Lambda 函數，請選擇建立新的 Lambda 函數。
 - 若要使用現有的 Lambda 函數，請選擇使用現有的 Lambda 函數，然後選擇函數的名稱。
- 如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[管理警示通知](#)。
- d. （選用）在設定狀態動作 - 選用區段中，執行下列動作：
- 在警示狀態動作下，新增動作。，然後選擇儲存。

您最多可以新增 10 個動作。

AWS IoT Events 可以在警示處於作用中狀態時執行動作。您可以定義內建動作以使用計時器或設定變數，或將資料傳送至其他 AWS 資源。如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[支援的動作](#)。

11. 選擇建立。

Note

AWS IoT Events 主控台會發出多個 API 請求，將警示新增至資產模型。當您選擇新增警示時，主控台會開啟顯示這些 API 請求進度的對話方塊。請保留在此頁面上，直到每個 API 請求成功或 API 請求失敗為止。如果請求失敗，請關閉對話方塊，修正問題，然後選擇新增警示再試一次。

定義 AWS IoT Events 警示 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來定義監控資產屬性的 AWS IoT Events 警示。您可以在新的或現有的資產模型上定義警示。在資產模型上定義警示之後，您可以在 中建立警示，AWS IoT Events 並將其連接到資產模型。在此程序中，您會執行下列動作：

步驟

- [步驟 1：在資產模型上定義警示](#)
- [步驟 2：定義 AWS IoT Events 警示模型](#)
- [步驟 3：啟用 AWS IoT SiteWise 和 之間的資料流程 AWS IoT Events](#)

步驟 1：在資產模型上定義警示

將警示定義和相關聯的屬性新增至新的或現有的資產模型。

在資產模型上定義警示 (CLI)

1. 建立稱為 `asset-model-payload.json` 的檔案。請依照這些其他章節中的步驟，將資產模型的詳細資訊新增至 檔案，但不要提交建立或更新資產模型的請求。在本節中，您將警示定義新增至 `asset-model-payload.json` 檔案中的資產模型詳細資訊。
 - 如需如何建立資產模型的詳細資訊，請參閱 [建立資產模型 \(AWS CLI\)](#)。
 - 如需如何更新現有資產模型的詳細資訊，請參閱 [更新資產或元件模型 \(AWS CLI\)](#)。

Note

您的資產模型必須至少定義一個資產屬性，包括使用警示監控的資產屬性。

2. 將警示複合模型 (`assetModelCompositeModels`) 新增至資產模型。AWS IoT Events 警示複合模型會指定 `IOT_EVENTS` 類型，並指定警示來源屬性。您可以在建立警示模型後新增警示來源屬性 AWS IoT Events。

Important

警示複合模型的名稱必須與稍後建立的 AWS IoT Events 警示模型相同。警示模型名稱只能包含英數字元。指定唯一的英數名稱，以便您可以為警示模型使用相同的名稱。

```

{
  ...
  "assetModelCompositeModels": [
    {
      "name": "BoilerTemperatureHighAlarm",
      "type": "AWS/ALARM",
      "properties": [
        {
          "name": "AWS/ALARM_TYPE",
          "dataType": "STRING",
          "type": {
            "attribute": {
              "defaultValue": "IOT_EVENTS"
            }
          }
        },
        {
          "name": "AWS/ALARM_STATE",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/ALARM_STATE",
          "type": {
            "measurement": {}
          }
        }
      ]
    }
  ]
}

```

3. 將警示閾值屬性新增至資產模型。指定用於此閾值的預設值。您可以根據此模型覆寫每個資產上的此預設值。

Note

警示閾值屬性必須是 INTEGER 或 DOUBLE。

```

{
  ...
  "assetModelProperties": [

```

```

    ...
  {
    "name": "Temperature Max Threshold",
    "dataType": "DOUBLE",
    "type": {
      "attribute": {
        "defaultValue": "105.0"
      }
    }
  }
]
}

```

4. (選用) 將警示通知屬性新增至資產模型。這些屬性會指定 IAM Identity Center 收件人，以及警示變更狀態時 AWS IoT Events 用來傳送通知的其他輸入。您可以根據此模型覆寫每個資產上的這些預設值。

Important

您可以傳送警示通知給 AWS IAM Identity Center 使用者。若要使用此功能，您必須啟用 IAM Identity Center。您 AWS 一次只能在一個區域中啟用 IAM Identity Center。這表示您只能在啟用 IAM Identity Center 的區域中定義警示通知。如需詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》中的 [「入門」](#)。

請執行下列操作：

- a. 新增指定 IAM Identity Center 身分存放區的 ID 的屬性。您可以使用 IAM Identity Center [ListInstances](#) API 操作來列出您的身分存放區。此操作僅適用於您啟用 IAM Identity Center 的區域。

```
aws sso-admin list-instances
```

然後，指定身分存放區 ID (例如 d-123EXAMPLE) 做為屬性的預設值。

```

{
  ...
  "assetModelProperties": [
    ...
    {

```

```

    "name": "identityStoreId",
    "dataType": "STRING",
    "type": {
      "attribute": {
        "defaultValue": "d-123EXAMPLE"
      }
    }
  }
]
}

```

- b. 新增 屬性，指定接收通知的 IAM Identity Center 使用者 ID。若要定義預設通知收件人，請新增 IAM Identity Center 使用者 ID 做為預設值。執行下列其中一項操作以取得 IAM Identity Center 使用者 ID：
 - i. 您可以使用 IAM Identity Center [ListUsers](#) API 來取得您知道其使用者名稱的 ID。將 *d-123EXAMPLE* 取代為您身分存放區的 ID，並將 *Name* 取代為使用者名稱。

```

aws identitystore list-users \
  --identity-store-id d-123EXAMPLE \
  --filters AttributePath=UserName,AttributeValue=Name

```

- ii. 使用 [IAM Identity Center 主控台](#) 瀏覽您的使用者並尋找使用者 ID。

然後，指定使用者 ID（例如 123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE）做為 屬性的預設值，或定義沒有預設值的屬性。

```

{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "userId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE"
        }
      }
    }
  ]
}

```

```
}

```

- c. (選用) 新增 屬性，指定簡訊 (文字) 訊息通知的預設寄件者 ID。寄件者 ID 會在 Amazon Simple Notification Service (Amazon SNS) 傳送的訊息上顯示為訊息寄件者。如需詳細資訊，請參閱AWS End User Messaging SMS 《使用者指南》中的在 [中請求寄件者 ID AWS End User Messaging SMS](#)。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "senderId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "MyFactory"
        }
      }
    }
  ]
}
```

- d. (選用) 新增 屬性，指定要用作電子郵件通知中來源地址的預設電子郵件地址。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "fromAddress",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "my.factory@example.com"
        }
      }
    }
  ]
}
```

- e. (選用) 新增 屬性，指定要在電子郵件通知中使用的預設主旨。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "emailSubject",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "[ALERT] High boiler temperature"
        }
      }
    }
  ]
}
```

- f. (選用) 新增屬性，指定要包含在通知中的額外訊息。根據預設，通知訊息包含警示的相關資訊。您也可以包含額外的訊息，提供使用者更多資訊。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "additionalMessage",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Turn off the power before you check the alarm."
        }
      }
    }
  ]
}
```

5. 建立資產模型或更新現有的資產模型。執行以下任意一項：

- 若要建立資產模型，請執行下列命令。

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-
payload.json
```

- 若要更新現有的資產模型，請執行下列命令。將 *asset-model-id* 取代為資產模型的 ID。

```
aws iotsitewise update-asset-model \  
  --asset-model-id asset-model-id \  
  --cli-input-json file://asset-model-payload.json
```

執行命令之後，請在回應 `assetModelId` 中記下。

範例：Boiler 資產模型

下列資產模型代表報告溫度資料的鍋爐。此資產模型會定義警示，以偵測鍋爐何時過熱。

```
{  
  "assetModelName": "Boiler Model",  
  "assetModelDescription": "Represents a boiler.",  
  "assetModelProperties": [  
    {  
      "name": "Temperature",  
      "dataType": "DOUBLE",  
      "unit": "C",  
      "type": {  
        "measurement": {}  
      }  
    },  
    {  
      "name": "Temperature Max Threshold",  
      "dataType": "DOUBLE",  
      "type": {  
        "attribute": {  
          "defaultValue": "105.0"  
        }  
      }  
    },  
    {  
      "name": "identityStoreId",  
      "dataType": "STRING",  
      "type": {  
        "attribute": {  
          "defaultValue": "d-123EXAMPLE"  
        }  
      }  
    }  
  ],  
  {  
    "name": "identityStoreId",  
    "dataType": "STRING",  
    "type": {  
      "attribute": {  
        "defaultValue": "d-123EXAMPLE"  
      }  
    }  
  }  
}
```

```
    "name": "userId",
    "dataType": "STRING",
    "type": {
      "attribute": {
        "defaultValue": "123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE"
      }
    }
  },
  {
    "name": "senderId",
    "dataType": "STRING",
    "type": {
      "attribute": {
        "defaultValue": "MyFactory"
      }
    }
  },
  {
    "name": "fromAddress",
    "dataType": "STRING",
    "type": {
      "attribute": {
        "defaultValue": "my.factory@example.com"
      }
    }
  },
  {
    "name": "emailSubject",
    "dataType": "STRING",
    "type": {
      "attribute": {
        "defaultValue": "[ALERT] High boiler temperature"
      }
    }
  },
  {
    "name": "additionalMessage",
    "dataType": "STRING",
    "type": {
      "attribute": {
        "defaultValue": "Turn off the power before you check the alarm."
      }
    }
  }
}
```



```

],
"assetModelHierarchies": [

],
"assetModelCompositeModels": [
{
  "name": "BoilerTemperatureHighAlarm",
  "type": "AWS/ALARM",
  "properties": [
    {
      "name": "AWS/ALARM_TYPE",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "IOT_EVENTS"
        }
      }
    },
    {
      "name": "AWS/ALARM_STATE",
      "dataType": "STRUCT",
      "dataTypeSpec": "AWS/ALARM_STATE",
      "type": {
        "measurement": {}
      }
    }
  ]
}
]
}

```

步驟 2：定義 AWS IoT Events 警示模型

在 中建立警示模型 AWS IoT Events。在 中 AWS IoT Events，您可以使用表達式來指定警示模型中的值。您可以使用表達式指定來自 的值 AWS IoT SiteWise，以評估並使用 做為警示的輸入。當 AWS IoT SiteWise 將資產屬性值傳送至警示模型時，AWS IoT Events 會評估表達式以取得屬性的值或資產的 ID。您可以在警示模型中使用下列表達式：

- 資產屬性值

若要取得資產屬性的值，請使用下列表達式。將 *assetModelId* 取代為資產模型的 ID，並將 *propertyId* 取代為 屬性的 ID。

```
$sitewise.assetModel.`assetModelId`.`propertyId`.propertyValue.value
```

- 資產 IDs

若要取得資產的 ID，請使用下列表達式。將 *assetModelId* 取代為資產模型的 ID，並將 *propertyId* 取代為屬性的 ID。

```
$sitewise.assetModel.`assetModelId`.`propertyId`.assetId
```

 Note

建立警示模型時，您可以定義常值，而不是評估為 AWS IoT SiteWise 值的表達式。這可以減少您在資產模型上定義的屬性數目。不過，如果您將值定義為常值，則無法根據資產模型在資產上自訂該值。您的 AWS IoT SiteWise Monitor 使用者也無法自訂警示，因為他們只能在資產上設定警示設定。

建立 AWS IoT Events 警示模型 (CLI)

1. 在 中建立警示模型時 AWS IoT Events，您必須指定警示使用的每個屬性 ID，其中包括下列項目：
 - 複合資產模型中的警示狀態屬性
 - 警示監控的 屬性
 - 閾值屬性
 - (選用) IAM Identity Center 身分存放區 ID 屬性
 - (選用) IAM Identity Center 使用者 ID 屬性
 - (選用) SMS 寄件者 ID 屬性
 - (選用) 來自地址屬性的電子郵件
 - (選用) 電子郵件主旨屬性
 - (選用) 其他訊息屬性

執行下列命令來擷取資產模型上這些屬性IDs。將 *asset-model-id* 取代為上一個步驟的資產模型 ID。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

此操作會傳回包含資產模型詳細資料的回應。請注意警示使用的每個屬性 ID。在下一個步驟中建立 AWS IoT Events 警示模型時，您會使用這些 IDs。

2. 在 中建立警示模型 AWS IoT Events。請執行下列操作：

- a. 建立稱為 `alarm-model-payload.json` 的檔案。
- b. 將下列 JSON 物件複製到 檔案。
- c. 輸入警示的名稱 (`alarmModelName`)、描述 (`alarmModelDescription`) 和嚴重性 (`severity`)。針對嚴重性，請指定反映貴公司嚴重性等級的整數。

 Important

警示模型必須與您先前在資產模型上定義的警示複合模型具有相同的名稱。
警示模型名稱只能包含英數字元。

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3
}
```

- d. 將比較規則 (`alarmRule`) 新增至警示。此規則定義要監控的屬性 (`inputProperty`)、要比較的閾值 (`threshold`)，以及要使用的比較運算子 (`comparisonOperator`)。
 - 以資產模型的 ID 取代 *assetModelId*。
 - 以警示監控的 屬性 ID 取代 *alarmPropertyId*。
 - 將 *thresholdAttributeId* 取代為 `threshold` 屬性的 ID。
 - 將 *GREATER* 取代為運算子，以用來比較屬性值與閾值。您可以從以下選項中選擇：
 - LESS
 - LESS_OR_EQUAL
 - EQUAL
 - NOT_EQUAL
 - GREATER_OR_EQUAL

- GREATER

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  }
}
```

- e. 新增動作 (alarmEventActions)，以在警示變更狀態 AWS IoT SiteWise 時將警示狀態傳送至。

 Note

對於進階組態，您可以定義警示變更狀態時要執行的其他動作。例如，您可以呼叫 AWS Lambda 函數或發佈至 MQTT 主題。如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[使用其他 AWS 服務](#)。

- 以資產模型的 ID 取代 *assetModelId*。
- 以警示監控的 屬性 ID 取代 *alarmPropertyId*。
- 將 *alarmStatePropertyId* 取代為警示複合模型中警示狀態屬性的 ID。

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
```

```

        "comparisonOperator": "GREATER",
        "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    },
    "alarmEventActions": {
        "alarmActions": [
            {
                "iotSiteWise": {
                    "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
                    "propertyId": "'alarmStatePropertyId'"
                }
            }
        ]
    }
}

```

- f. (選用) 設定警示通知設定。警示通知動作會使用您帳戶中的 Lambda 函數來傳送警示通知。如需詳細資訊，請參閱[中的警示通知需求 AWS IoT SiteWise](#)。在警示通知設定中，您可以設定要傳送給 IAM Identity Center 使用者的簡訊和電子郵件通知。請執行下列操作：
 - i. 將警示通知組態 (alarmNotification) 新增至 中的承載alarm-model-payload.json。
 - 以處理警示通知的 Lambda 函數 ARN 取代 *alarmNotificationFunctionArn*。

```

{
    "alarmModelName": "BoilerTemperatureHighAlarm",
    "alarmModelDescription": "Detects when the boiler temperature is high.",
    "severity": 3,
    "alarmRule": {
        "simpleRule": {
            "inputProperty":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
            "comparisonOperator": "GREATER",
            "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
        }
    },
    "alarmEventActions": {
        "alarmActions": [

```

```

    {
      "iotSiteWise": {
        "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
        "propertyId": "'alarmStatePropertyId'"
      }
    }
  ],
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        }
      }
    ]
  }
}

```

- ii. (選用) 設定 SMS 通知 (smsConfigurations)，以在警示變更狀態時傳送給 IAM Identity Center 使用者。

- 將 *identityStoreIdAttributeId* 取代為包含 IAM Identity Center 身分存放區 ID 的屬性 ID。
- 將 *userIdAttributeId* 取代為包含 IAM Identity Center 使用者 ID 的屬性 ID。
- 將 *senderIdAttributeId* 取代為包含 Amazon SNS 寄件者 ID 的屬性 ID，或從 senderId 承載中移除。
- 以包含額外訊息的屬性 ID 取代 *additionalMessageAttributeId*，或從 additionalMessage 承載中移除。

```

{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",

```

```

        "comparisonOperator": "GREATER",
        "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    },
    "alarmEventActions": {
        "alarmActions": [
            {
                "iotSiteWise": {
                    "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
                    "propertyId": "'alarmStatePropertyId'"
                }
            }
        ]
    },
    "alarmNotification": {
        "notificationActions": [
            {
                "action": {
                    "lambdaAction": {
                        "functionArn": "alarmNotificationFunctionArn"
                    }
                },
                "smsConfigurations": [
                    {
                        "recipients": [
                            {
                                "ssoIdentity": {
                                    "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.va
                                    "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                                }
                            }
                        ],
                        "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
                        "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.
                    }
                ]
            }
        ]
    }
}

```

```

    }
  }

```

- iii. (選用) 設定電子郵件通知 (emailConfigurations), 以在警示變更狀態時傳送給 IAM Identity Center 使用者。
 - 將 *identityStoreIdAttributeId* 取代為 IAM Identity Center 身分存放區 ID 屬性的 ID。
 - 將 *userIdAttributeId* 取代為 IAM Identity Center 使用者 ID 屬性的 ID。
 - 以「寄件人」地址屬性屬性的 ID 取代 *fromAddressAttributeId*, 或從from承載中移除。
 - 將 *emailSubjectAttributeId* 取代為電子郵件主體屬性屬性的 ID, 或從subject承載中移除。
 - 以其他訊息屬性屬性的 ID 取代 *additionalMessageAttributeId*, 或從additionalMessage承載中移除。

```

{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
},

```



```

"alarmNotification": {
  "notificationActions": [
    {
      "action": {
        "lambdaAction": {
          "functionArn": "alarmNotificationFunctionArn"
        }
      },
      "smsConfigurations": [
        {
          "recipients": [
            {
              "ssoIdentity": {
                "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId` .propertyValue.value",
                "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId` .propertyValue.value"
              }
            }
          ],
          "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId` .propertyValue.value",
          "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId` .propertyValue.value"
        }
      ],
      "emailConfigurations": [
        {
          "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId` .propertyValue.value",
          "recipients": {
            "to": [
              {
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId` .propertyValue.value",
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId` .propertyValue.value"
                }
              }
            ]
          },
          "content": {

```

```

        "subject":
        "$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value",
        "additionalMessage":
        "$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.value"
    }
}
]
}
]
}
}

```

- g. (選用) 將警示功能 (alarmCapabilities) 新增至 `alarm-model-payload.json`。在此物件中，您可以根據資產模型指定確認流程是否已啟用，以及資產的預設啟用狀態。如需確認流程的詳細資訊，請參閱 [警示狀態](#)。

```

{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
        "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
        "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
            "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {

```

```

    "action": {
      "lambdaAction": {
        "functionArn": "alarmNotificationFunctionArn"
      }
    },
    "smsConfigurations": [
      {
        "recipients": [
          {
            "ssoIdentity": {
              "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId` .propertyValue.value"
              "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId` .propertyValue.value"
            }
          }
        ],
        "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId` .propertyValue.value",
        "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId` .propertyValue.value"
      }
    ],
    "emailConfigurations": [
      {
        "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId` .propertyValue.value",
        "recipients": {
          "to": [
            {
              "ssoIdentity": {
                "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId` .propertyValue.value"
                "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId` .propertyValue.value"
              }
            }
          ]
        },
        "content": {
          "subject":
"$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId` .propertyValue.value",
          "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId` .propertyValue.value"
        }
      }
    ]
  }
}

```

```

    }
  }
]
}
],
"alarmCapabilities": {
  "initializationConfiguration": {
    "disabledOnInitialization": false
  },
  "acknowledgeFlow": {
    "enabled": true
  }
}
}
}

```

- h. 新增 AWS IoT Events 可擔任傳送資料的 IAM 服務角色 (roleArn) AWS IoT SiteWise。此角色需要允許 `iotevents.amazonaws.com` 擔任角色的 `iotsitewise:BatchPutAssetPropertyValue` 許可和信任關係。若要傳送通知，此角色也需要 `lambda:InvokeFunction` 和 `sso-directory:DescribeUser` 許可。如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的 [警示服務角色](#)。

- 將取代 `roleArn` 為 AWS IoT Events 可擔任以執行這些動作之角色的 ARN。

```

{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
        "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
        "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {

```

```

        "assetId":
        "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
        "propertyId": "`alarmStatePropertyId`"
    }
}
],
},
"alarmNotification": {
    "notificationActions": [
        {
            "action": {
                "lambdaAction": {
                    "functionArn": "alarmNotificationFunctionArn"
                }
            },
            "smsConfigurations": [
                {
                    "recipients": [
                        {
                            "ssoIdentity": {
                                "identityStoreId":
                                "$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value",
                                "userId":
                                "$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                            }
                        }
                    ],
                    "senderId":
                    "$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
                    "additionalMessage":
                    "$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.value"
                }
            ],
            "emailConfigurations": [
                {
                    "from":
                    "$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value",
                    "recipients": {
                        "to": [
                            {
                                "ssoIdentity": {
                                    "identityStoreId":
                                    "$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                                }
                            }
                        ]
                    }
                }
            ]
        }
    ]
}

```

```

        "userId":
        "$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
      }
    }
  ],
  },
  "content": {
    "subject":
    "$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value",
    "additionalMessage":
    "$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.value"
  }
}
],
},
"alarmCapabilities": {
  "initializationConfiguration": {
    "disabledOnInitialization": false
  },
  "acknowledgeFlow": {
    "enabled": false
  }
},
"roleArn": "arn:aws:iam::123456789012:role/MyIoTEventsAlarmRole"
}

```

- i. 執行下列命令，從 中的承載建立 AWS IoT Events 警示模型alarm-model-payload.json。

```
aws iotevents create-alarm-model --cli-input-json file://alarm-model-payload.json
```

- j. 操作會傳回包含警示模型 ARN 的回應 alarmModelArn。複製此 ARN 以在下一個步驟的資產模型警示定義中設定。

步驟 3：啟用 AWS IoT SiteWise 和 之間的資料流程 AWS IoT Events

在 AWS IoT SiteWise 和 中建立所需的資源後 AWS IoT Events，您可以啟用資源之間的資料流程，以啟用警示。在本節中，您會更新資產模型中的警示定義，以使用您在上一個步驟中建立的警示模型。

啟用 AWS IoT SiteWise 和 AWS IoT Events (CLI) 之間的資料流程

- 在資產模型中將警示模型設定為警示的來源。請執行下列操作：
 - 執行以下命令來擷取現有的資產模型定義。將 *asset-model-id* 取代為資產模型的 ID。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

此操作會傳回包含資產模型詳細資料的回應。

- 建立名為 `update-asset-model-payload.json` 的文件，並將前一個命令的回應複製到檔案中。
- 從 `update-asset-model-payload.json` 檔案移除下列鍵/值對：
 - `assetModelId`
 - `assetModelArn`
 - `assetModelCreationDate`
 - `assetModelLastUpdateDate`
 - `assetModelStatus`
- 將警示來源屬性 (AWS/ALARM_SOURCE) 新增至您先前定義的警示複合模型。將 *alarmModelArn* 取代為警示模型的 ARN，這會設定警示來源屬性的值。

```
{
  ...
  "assetModelCompositeModels": [
    ...
    {
      "name": "BoilerTemperatureHighAlarm",
      "type": "AWS/ALARM",
      "properties": [
        {
          "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
          "name": "AWS/ALARM_TYPE",
          "dataType": "STRING",
          "type": {
            "attribute": {
              "defaultValue": "IOT_EVENTS"
            }
          }
        }
      ]
    }
  ],
}
```

```

    {
      "id": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "name": "AWS/ALARM_STATE",
      "dataType": "STRUCT",
      "dataTypeSpec": "AWS/ALARM_STATE",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "AWS/ALARM_SOURCE",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "alarmModelArn"
        }
      }
    }
  ]
}

```

- e. 執行下列命令，使用存放在 `update-asset-model-payload.json` 檔案中的定義來更新資產模型。將 *asset-model-id* 取代為資產模型的 ID。

```

aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://update-asset-model-payload.json

```

您的資產模型現在定義 偵測傳入的警示 AWS IoT Events。警示會根據此資產模型監控所有資產中的目標屬性。您可以在每個資產上設定警示，以自訂屬性，例如每個資產的閾值或 IAM Identity Center 收件人。如需詳細資訊，請參閱[在 中設定資產警示 AWS IoT SiteWise](#)。

在 中定義外部警示 AWS IoT SiteWise

外部警示包含您在 外部偵測到的警示狀態 AWS IoT SiteWise。

定義外部警示（主控台）

您可以使用 AWS IoT SiteWise 主控台，在現有資產模型上定義外部警示。若要在新的資產模型上定義外部警示，請建立資產模型，然後完成這些步驟。如需詳細資訊，請參閱[在 中建立資產模型 AWS IoT SiteWise](#)。

在資產模型上定義警示

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Models (模型)。
3. 選擇要為其定義警示的資產模型。
4. 選擇警示定義索引標籤。
5. 選擇新增警示。
6. 在警示類型選項中，選擇外部警示。
7. 輸入警示的名稱。
8. （選用）輸入警示的描述。
9. 選擇新增警示。

定義外部警示 (CLI)

您可以使用 AWS CLI 在新的或現有的資產模型上定義外部警示。

若要將外部警示新增至資產模型，請將警示複合模型新增至資產模型。外部警示複合模型會指定 EXTERNAL 類型，且不指定警示來源屬性。下列範例複合警示會定義外部溫度警示。

```
{
  ...
  "assetModelCompositeModels": [
    {
      "name": "BoilerTemperatureHighAlarm",
      "type": "AWS/ALARM",
      "properties": [
        {
          "name": "AWS/ALARM_TYPE",
          "dataType": "STRING",
          "type": {
            "attribute": {
              "defaultValue": "EXTERNAL"
            }
          }
        }
      ]
    }
  ]
}
```

```
    }
  },
  {
    "name": "AWS/ALARM_STATE",
    "dataType": "STRUCT",
    "dataTypeSpec": "AWS/ALARM_STATE",
    "type": {
      "measurement": {}
    }
  }
]
```

如需如何將複合模型新增至新資產模型或現有資產模型的詳細資訊，請參閱下列內容：

- [建立資產模型 \(AWS CLI\)](#)
- [更新資產或元件模型 \(AWS CLI\)](#)

定義外部警示之後，您可以根據資產模型將警示狀態擷取至資產。如需詳細資訊，請參閱[在中擷取外部警示狀態 AWS IoT SiteWise](#)。

在中設定資產警示 AWS IoT SiteWise

在資產模型上定義 AWS IoT Events 警示之後，您可以根據資產模型在每個資產上設定警示。您可以編輯警示的閾值和通知設定。這些值都是資產上的屬性，因此您可以更新屬性的預設值來設定這些值。

Note

您可以為 AWS IoT Events 警示設定這些值，但不能在外部警示上設定這些值。

主題

- [設定閾值（主控台）](#)
- [設定閾值 \(AWS CLI\)](#)
- [在中設定通知設定 AWS IoT SiteWise](#)

設定閾值（主控台）

您可以使用 AWS IoT SiteWise 主控台來更新指定警示閾值的 屬性值。

更新警示的閾值（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要更新警示閾值的資產。

Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇編輯。
5. 尋找警示用於其閾值的屬性，然後輸入其新值。
6. 選擇儲存。

設定閾值 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來更新指定警示閾值的 屬性值。

您必須知道資產的 `assetId` 和財產的 `propertyId`，才能完成此程序。您也可以使用外部 ID。如果您建立資產但不知道其 `assetId`，請使用 [ListAssets](#) API 列出特定模型的所有資產。使用 [DescribeAsset](#) 操作來檢視資產的屬性，包括屬性 IDs。

使用 [BatchPutAssetPropertyValue](#) 操作將屬性值指派給您的資產。您可以使用此操作一次設定多個屬性。此操作的裝載包含項目清單，而每個項目都包含資產 ID、屬性 ID 和屬性值。

若要更新屬性的值 (AWS CLI)

1. 建立名為 `batch-put-payload.json` 的檔案，並將下列 JSON 物件複製到檔案。此範例承載示範如何設定風力發電機的緯度和經度。更新 ID、值和時間戳記，以修改您的使用案例的承載。

```
{
  "entries": [
    {
      "entryId": "windfarm3-turbine7-latitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
```

```
"propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
"propertyValues": [
  {
    "value": {
      "doubleValue": 47.6204
    },
    "timestamp": {
      "timeInSeconds": 1575691200
    }
  }
],
{
  "entryId": "windfarm3-turbine7-longitude",
  "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE",
  "propertyValues": [
    {
      "value": {
        "doubleValue": 122.3491
      },
      "timestamp": {
        "timeInSeconds": 1575691200
      }
    }
  ]
}
]
```

- 承載中的每個項目都包含 `entryId`，您可以將其定義為任何獨特的字串。如果有任何請求項目失敗，每個錯誤都會包含對應請求的 `entryId`，這樣您就可以知道要重試哪些請求。
- 若要設定屬性值，您可以在每個屬性內容的 `propertyValues` 清單中包含一個時間戳記品質值 (TQV) 結構。此結構必須包含新的 `value` 和目前的 `timestamp`。
 - `value` – 包含下列其中一個欄位的結構，取決於所設定屬性的類型：
 - `booleanValue`
 - `doubleValue`
 - `integerValue`
 - `stringValue`
 - `nullValue`

- `timestamp` – 包含目前 Unix epoch 時間的結構，以秒為單位，`timeInSeconds`。AWS IoT SiteWise 會拒絕過去超過 7 天或未來超過 5 分鐘的任何時間戳記資料點。

如需如何準備 [BatchPutAssetPropertyValue](#) 承載的詳細資訊，請參閱 [使用 AWS IoT SiteWise APIs 擷取資料](#)。

2. 執行下列命令，將屬性值傳送至 AWS IoT SiteWise：

```
aws iotsitewise batch-put-asset-property-value -\cli-input-json file://batch-put-payload.json
```

在 中設定通知設定 AWS IoT SiteWise

您可以使用 AWS IoT SiteWise 主控台或 AWS Command Line Interface () 來設定警示通知設定AWS CLI。

設定通知設定（主控台）

您可以使用 AWS IoT SiteWise 主控台來更新指定警示通知設定的屬性值。

更新警示的通知設定（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要更新警示設定的資產。
4. 選擇編輯。
5. 尋找警示用於您要變更之通知設定的屬性，然後輸入其新值。
6. 選擇儲存。

設定通知設定 (CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 更新指定警示通知設定的 屬性值。

您必須知道資產的 `assetId` 和財產的 `propertyId`，才能完成此程序。您也可以使用外部 ID。如果您建立資產但不知道其 `assetId`，請使用 [ListAssets](#) API 列出特定模型的所有資產。使用 [DescribeAsset](#) 操作來檢視資產的屬性，包括屬性 IDs。

使用 [BatchPutAssetPropertyValue](#) 操作將屬性值指派給您的資產。您可以使用此操作一次設定多個屬性。此操作的裝載包含項目清單，而每個項目都包含資產 ID、屬性 ID 和屬性值。

若要更新屬性的值 (AWS CLI)

1. 建立名為 `batch-put-payload.json` 的檔案，並將下列 JSON 物件複製到檔案。此範例承載示範如何設定風力發電機的緯度和經度。更新 ID、值和時間戳記，以修改您的使用案例的承載。

```
{
  "entries": [
    {
      "entryId": "windfarm3-turbine7-latitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 47.6204
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    },
    {
      "entryId": "windfarm3-turbine7-longitude",
      "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE",
      "propertyValues": [
        {
          "value": {
            "doubleValue": 122.3491
          },
          "timestamp": {
            "timeInSeconds": 1575691200
          }
        }
      ]
    }
  ]
}
```

- 承載中的每個項目都包含 `entryId`，您可以將其定義為任何獨特的字串。如果有任何請求項目失敗，每個錯誤都會包含對應請求的 `entryId`，這樣您就可以知道要重試哪些請求。
- 若要設定屬性值，您可以在每個屬性內容的 `propertyValues` 清單中包含一個時間戳記品質值 (TQV) 結構。此結構必須包含新的 `value` 和目前的 `timestamp`。
 - `value` – 包含下列其中一個欄位的結構，取決於所設定屬性的類型：
 - `booleanValue`
 - `doubleValue`
 - `integerValue`
 - `stringValue`
 - `nullValue`
 - `timestamp` – 包含目前 Unix epoch 時間的結構，以秒為單位，`timeInSeconds`。AWS IoT SiteWise 會拒絕過去超過 7 天或未來超過 5 分鐘的任何時間戳記資料點。

如需如何準備 [BatchPutAssetPropertyValue](#) 承載的詳細資訊，請參閱 [使用 AWS IoT SiteWise APIs 擷取資料](#)。

2. 執行下列命令，將屬性值傳送至 AWS IoT SiteWise：

```
aws iotsitewise batch-put-asset-property-value -\-cli-input-json file://batch-put-payload.json
```

在中回應警示 AWS IoT SiteWise

當 AWS IoT Events 警示變更狀態時，您可以執行下列動作來回應警示：

- 確認警示，表示您正在處理問題。
- 暫停警示以暫時停用它。
- 停用警示以永久停用它，直到您再次啟用它為止。
- 啟用停用的警示以偵測警示狀態。
- 重設警示以清除其狀態和最新值。

您可以使用 AWS IoT SiteWise 主控台或 AWS IoT Events API 來回應警示。

 Note

您可以回應 AWS IoT Events 警示，但不能回應外部警示。

主題

- [回應警示（主控台）](#)
- [回應警示 \(API\)](#)

回應警示（主控台）

您可以使用 AWS IoT SiteWise 主控台來確認、暫停、停用或啟用警示。

主題

- [確認警示（主控台）](#)
- [暫停警示（主控台）](#)
- [停用警示（主控台）](#)
- [啟用警示（主控台）](#)
- [重設警示（主控台）](#)

確認警示（主控台）

您可以確認警示，表示您正在處理問題。

 Note

您必須啟用警示的確認流程，才能確認警示。如果您從主控台定義警示，則預設會 AWS IoT SiteWise 啟用此選項。

確認警示（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要確認警示的資產。

 Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇警示索引標籤。
5. 選取要確認的警示，然後選擇動作以開啟回應動作功能表。
6. 選擇確認。警示的狀態變更為已確認。

暫停警示（主控台）

您可以暫停警示以暫時停用它。指定休眠警示的持續時間。

暫停警示（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要讓警示休眠的資產。

 Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇警示索引標籤。
5. 選取要暫停的警示，然後選擇動作以開啟回應動作功能表。
6. 選擇暫停。模型隨即開啟，您可以在其中指定休眠的持續時間。
7. 選擇休眠長度或輸入自訂休眠長度。
8. 選擇儲存。警示的狀態會變更為暫停。

停用警示（主控台）

您可以停用警示，使其不再偵測。停用警示之後，如果您想要偵測警示，則必須再次啟用它。

停用警示（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。

3. 選擇您要停用警示的資產。

Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇警示索引標籤。
5. 選取要停用的警示，然後選擇動作以開啟回應動作功能表。
6. 選擇停用。警示的狀態會變更為已停用。

啟用警示（主控台）

您可以啟用警示，以便在停用或休眠後再次偵測。

啟用警示（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要為其啟用警示的資產。

Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇警示索引標籤。
5. 選取要啟用的警示，然後選擇動作以開啟回應動作功能表。
6. 選擇 啟用。警示的狀態會變更為「正常」。

重設警示（主控台）

您可以重設警示以清除其狀態和最新值。

重設警示（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要為其重設警示的資產。

 Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇警示索引標籤。
5. 選取要啟用的警示，然後選擇動作以開啟回應動作功能表。
6. 選擇 Reset (重設)。警示的狀態會變更為「正常」。

回應警示 (API)

您可以使用 AWS IoT Events API 來確認、暫停、停用、啟用或重設警示。如需詳細資訊，請參閱 AWS IoT Events API 參考中的下列操作：

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

如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的[回應警示](#)。

在中擷取外部警示狀態 AWS IoT SiteWise

外部警示是您在外部評估的警示 AWS IoT SiteWise。當您的資料來源報告您想要擷取的警示狀態時，您可以使用外部警示 AWS IoT SiteWise。

警示狀態屬性需要警示狀態資料值的特定格式。每個資料值必須是序列化為字串的 JSON 物件。然後，您將序列化字串擷取為字串值。如需詳細資訊，請參閱[警示狀態屬性](#)。

Example 警示狀態資料值範例（未序列化）

```
{
  "stateName": "Active"
}
```

Example 警示狀態資料值範例（序列化）

```
{\"stateName\":\"Active\"}
```

Note

如果您的資料來源無法以此格式報告資料，或者您無法在擷取資料之前將資料轉換為此格式，您可以選擇不使用警示屬性。反之，您可以將資料擷取為具有字串資料類型的測量屬性。如需詳細資訊，請參閱[從設備定義資料串流（度量）](#)及[將資料擷取至 AWS IoT SiteWise](#)。

在中映射外部警示狀態串流 AWS IoT SiteWise

您可以定義屬性別名，將資料串流映射至警示狀態屬性。這可協助您在擷取或擷取資料時輕鬆識別警示狀態屬性。如需屬性別名的詳細資訊，請參閱[管理的資料串流 AWS IoT SiteWise](#)。

主題

- [映射外部警示狀態串流（主控台）](#)
- [映射外部警示狀態串流 \(AWS CLI\)](#)

映射外部警示狀態串流（主控台）

您可以定義屬性別名，將資料串流映射至警示狀態屬性。這可協助您在擷取或擷取資料時輕鬆識別警示狀態屬性。如需屬性別名的詳細資訊，請參閱[管理的資料串流 AWS IoT SiteWise](#)。

您可以使用 AWS IoT SiteWise 主控台來設定警示狀態屬性的別名。

設定警示狀態屬性的屬性別名（主控台）

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇您要為其設定屬性別名的資產。

Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇編輯。

5. 捲動至警示並展開 區段。
6. 在外部警示下，在屬性別名中輸入別名 – 選用。
7. 選擇儲存。

映射外部警示狀態串流 (AWS CLI)

您可以定義屬性別名，將資料串流映射至警示狀態屬性。這可協助您在擷取或擷取資料時輕鬆識別警示狀態屬性。如需屬性別名的詳細資訊，請參閱 [管理的資料串流 AWS IoT SiteWise](#)。

您可以使用 AWS Command Line Interface (AWS CLI) 來設定警示狀態屬性的別名。

您必須知道資產的 `assetId` 和財產的 `propertyId`，才能完成此程序。您也可以使用外部 ID。如果您建立資產但不知道其 `assetId`，請使用 [ListAssets](#) API 列出特定模型的所有資產。使用 [DescribeAsset](#) 操作來檢視資產的屬性，包括屬性 IDs。

Note

[DescribeAsset](#) 回應包含資產的複合資產模型清單。每個警示都是複合模型。若要尋找 `propertyId`，請尋找警示的複合模型，然後在該複合模型中尋找 `AWS/ALARM_STATE` 屬性。

如需如何設定屬性別名的詳細資訊，請參閱 [更新資產屬性別名](#)。

在中擷取警示狀態資料 AWS IoT SiteWise

警示狀態屬性預期警示狀態為序列化 JSON 字串。若要將警示狀態擷取至 中的外部警示 AWS IoT SiteWise，您可以將此序列化字串擷取為時間戳記字串值。下列範例示範作用中警示的狀態資料值。

```
{"stateName\":\"Active\"}
```

若要識別警示狀態屬性，您可以指定下列其中一項：

- 您要傳送資料的警示屬性 `propertyId` 的 `assetId` 和。
- `propertyAlias`，這是資料串流別名（例如 `/company/windfarm/3/turbine/7/temperature/high`）。若要使用此選項，您必須先設定警示屬性的別名。若要了解如何設定警示狀態屬性的屬性別名，請參閱 [在中映射外部警示狀態串流 AWS IoT SiteWise](#)。

下列範例 [BatchPutAssetPropertyValue](#) API 承載示範如何格式化外部警示的狀態。此外部警示會在風力發電機每分鐘旋轉 (RPM) 讀數過高時回報。

Example 警示狀態資料的範例 BatchPutAssetPropertyValue 承載

```
{
  "entries": [
    {
      "entryId": "unique entry ID",
      "propertyAlias": "/company/windfarm/3/turbine/7/temperature/high",
      "propertyValues": [
        {
          "value": {
            "stringValue": "{\"stateName\":\"Active\"}"
          },
          "timestamp": {
            "timeInSeconds": 1607550262
          }
        }
      ]
    }
  ]
}
```

如需如何使用 BatchPutAssetPropertyValue API 擷取資料的詳細資訊，請參閱 [使用 AWS IoT SiteWise APIs 擷取資料](#)。

如需其他擷取資料方式的詳細資訊，請參閱 [將資料擷取至 AWS IoT SiteWise](#)。

AWS IoT SiteWise 助理

AWS IoT SiteWise 助理是生成式 AI 輔助。它可讓工廠經理、品質工程師和維護技術人員等使用者直接從其營運和企業資料中取得洞見、解決問題並採取動作。

AWS IoT SiteWise 助理會將 AWS IoT 資料、資產模型、手冊和文件的資訊合併為可理解的關鍵事件摘要。它還啟用互動式深入探索問答工作階段，以便於診斷、根本原因探索和引導建議。

主題

- [設定 AWS IoT SiteWise 助理](#)
- [建立資料集](#)
- [編輯資料集](#)
- [刪除資料集](#)
- [AWS IoT SiteWise 助理問題](#)

設定 AWS IoT SiteWise 助理

AWS IoT SiteWise 助理組態

1. 登入 [AWS IoT SiteWise 主控台](#)。

Note

授予許可可以啟用與服務的整合 AWS IoT TwinMaker。這是 AWS IoT SiteWise 助理和儀表板在 AWS IoT SiteWise 資源中執行 SQL 查詢的必要項目。請參閱[整合 AWS IoT SiteWise 和 AWS IoT TwinMaker](#)。

Grant permission

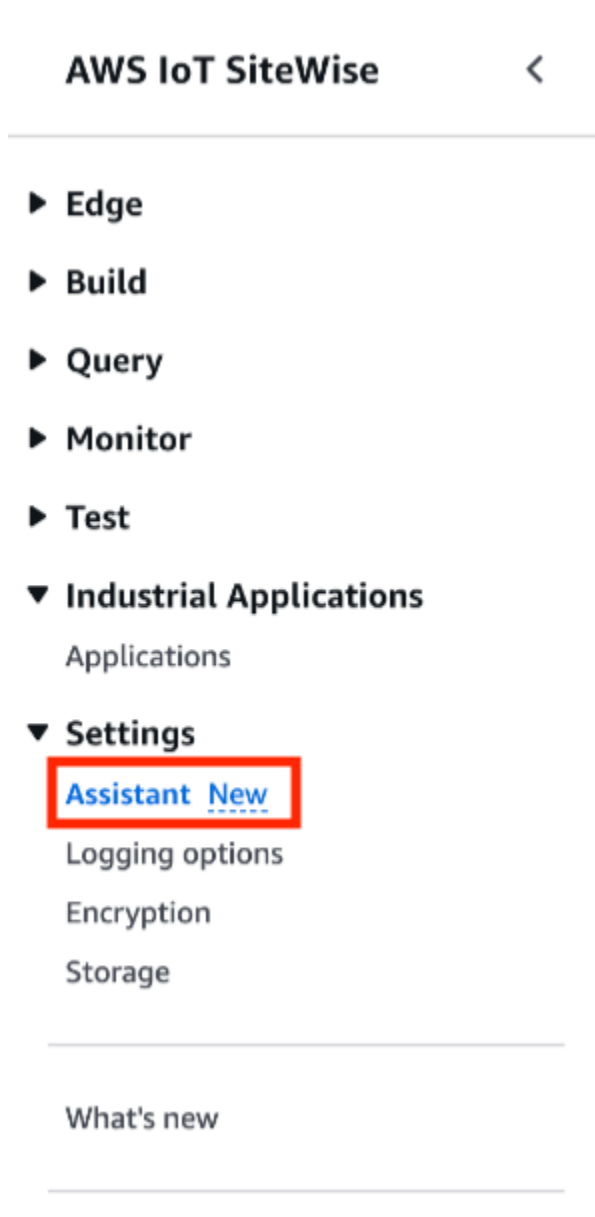


In order to better organize and model industrial data, AWS IoT SiteWise requires permission to integrate with AWS IoT TwinMaker. Choose **Grant permission** to enable this integration. This integration could incur costs. For more information, see [pricing](#) .

Cancel

Grant permission

2. 從左側導覽面板選擇助理。



建立資料集

Note

AWS IoT SiteWise 助理必須使用具有 [Amazon Kendra](#) 索引的資料集，以取得企業級知識和指導。如果您沒有 Amazon Kendra 索引，請參閱[建立索引](#)以建立索引。新增[資料集](#)可改善助理回應的品質，並將幻覺降至最低。

Console

在 AWS IoT SiteWise 主控台中建立資料集

1. 資料集會顯示在AWS IoT SiteWise 助理頁面的資料集區段中。
2. 如果沒有資料集，請選擇建立資料集。
3. 在資料集詳細資訊頁面中，從下拉式選單中選擇 Kendra 索引，以與資料集建立關聯。
4. 資料集名稱由步驟 3 中選取的 Kendra 索引填入。視需要編輯名稱。
5. （選用）資料集描述由步驟 3 中選取的 Kendra 索引填入。如有需要，請編輯描述。
6. 在許可區段中，從下方選擇：
 - a. 選擇建立並使用新的服務角色。根據預設，AWS IoT SiteWise 會自動建立服務角色。此角色允許 AWS IoT SiteWise 助理存取您的 Kendra 索引。
 - b. 選擇使用現有的服務角色，然後選擇目標角色。
7. 選擇 Create (建立)。

[AWS IoT SiteWise](#) > [Assistant](#) > Create dataset

Create dataset

Create a dataset for the Assistant.

Dataset details [Info](#)
Kendra index
Select a Kendra index for the Assistant dataset.

test-index ▼ [Amazon Kendra](#)

Dataset name
Dataset name is pre-populated from the Kendra index selected. You can modify the dataset name.

Dataset1

Dataset name must be 1-256 characters. Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen).

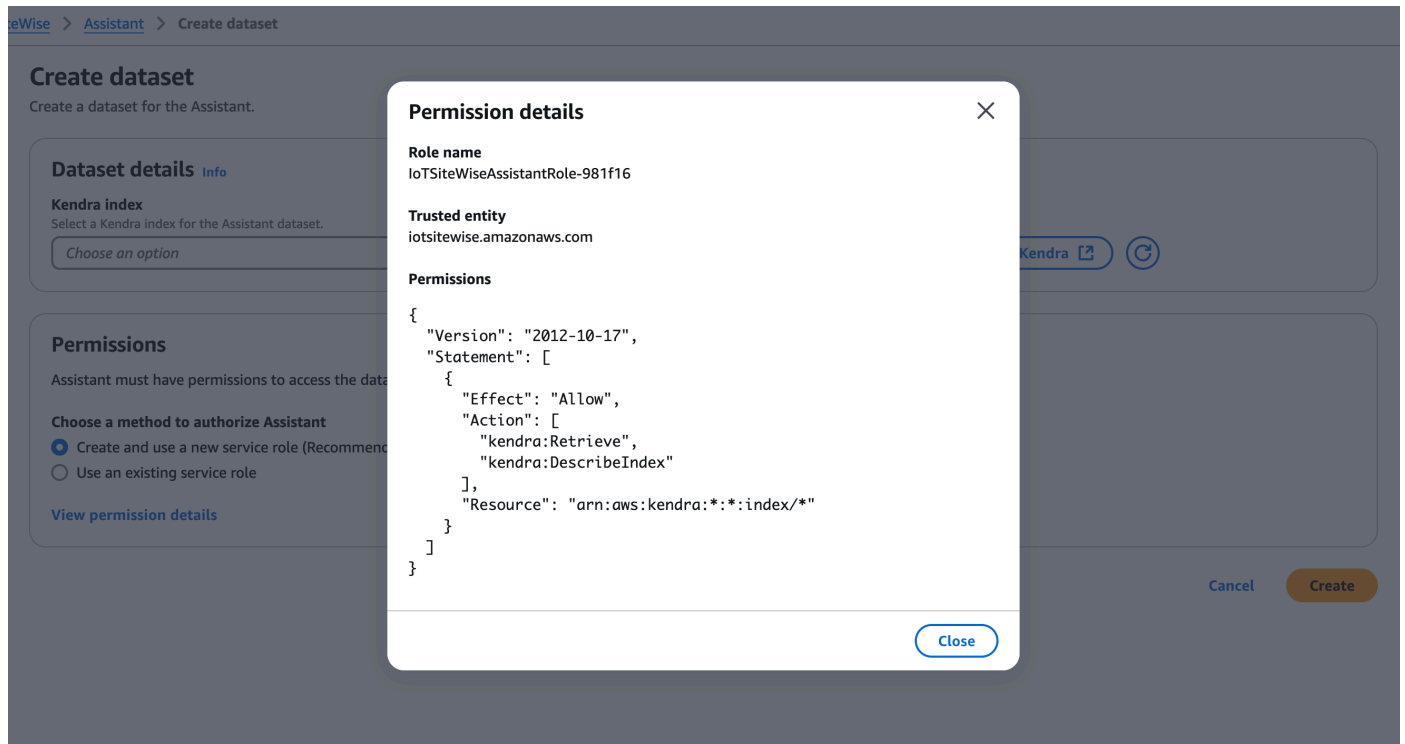
Dataset description - optional
Dataset description is pre-populated from the Kendra index selected. You can modify the dataset description.

Dataset for AWS IoT SiteWise Assistant

Dataset description must be 1-2048 characters.

Permissions
Assistant must have permissions to access the data. To create a custom role, visit the [IAM console](#).
Choose a method to authorize Assistant
☒ Create and use a new service role - *recommended*
☐ Use an existing service role
[View permission details](#)[Cancel](#) [Create](#)

如果使用者選擇建立和使用新的服務角色，則由 AWS IoT SiteWise 為使用者建立的服務角色。



AWS CLI

在 中建立資料集 AWS CLI

1. 建立用於建立資料集的 IAM 角色。使用下列許可政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kendra:Retrieve"
      ],
      "Resource": "arn:aws:kendra:*:*:index/*"
    }
  ]
}
```

使用下列信任關係：

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iotsitewise.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 使用範例中提供的範本建立檔案 `create-dataset.json`。填入 `datasetId`、`kendra knowledgeBaseArn` 和 `roleArn` 以與此資料集連線。

```
{
  "datasetId": "<UUID>",
  "datasetName": "DatasetForAssistant",
  "datasetSource": {
    "sourceType": "KENDRA",
    "sourceFormat": "KNOWLEDGE_BASE",
    "sourceDetail": {
      "kendra": {
        "knowledgeBaseArn": "arn:aws:kendra::%s:index/index",
        "roleArn": "arn:aws:iam::%s:role/role"
      }
    }
  }
}
```

3. 使用下列命令建立資料集：

```
aws iotsitewise create-dataset --cli-input-json file://create-dataset.json --
region us-east-1
```

編輯資料集

Console

編輯資料集

1. 資料集會顯示在助理頁面的資料集區段中。選擇要編輯的資料集。選擇編輯以開始編輯。
2. 在資料集詳細資訊頁面中，從下拉式選單中選擇 Kendra 索引，以與資料集建立關聯。
3. 資料集名稱由步驟 2 中選取的 Kendra 索引填入。視需要編輯名稱。
4. （選用）資料集描述由步驟 2 中選取的 Kendra 索引填入。如有需要，請編輯描述。
5. 在許可區段中，從下方選擇：
 - a. 選擇建立並使用新的服務角色。根據預設，AWS IoT SiteWise 會自動建立服務角色。此角色允許 AWS IoT SiteWise 助理存取您的 Kendra 索引。
 - b. 選擇使用現有的服務角色，然後選擇目標角色。
6. 選擇儲存變更以儲存您的選擇。

[AWS IoT SiteWise](#) > [Assistant](#) > Edit dataset

Edit dataset

Edit a dataset for the Assistant.

Dataset details [Info](#)

Kendra index

Select a Kendra index for the Assistant dataset.

test-index

[Amazon Kendra](#)



Dataset name

Dataset name is pre-populated from the Kendra index selected. You can modify the dataset name.

Dataset1

Dataset name must be 1-256 characters. Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen).

Dataset description - optional

Dataset description is pre-populated from the Kendra index selected. You can modify the dataset description.

Dataset for AWS IoT SiteWise Assistant

Dataset description must be 1-2048 characters.

Permissions

Assistant must have permissions to access the data. To create a custom role, visit the [IAM console](#).

Choose a method to authorize Assistant

- ☐ Create and use a new service role - *recommended*
- ☒ Use an existing service role

Existing role

IoTSiteWiseAssistantRole-40f994

[View the IoTSiteWiseAssistantRole-40f994 role in the IAM console](#)

[Cancel](#)

[Save changes](#)

AWS CLI

在中編輯資料集 AWS CLI

1. 使用範例中提供的範本建立檔案 update-dataset.json。填入 datasetId、kendra knowledgeBaseArn 和 roleArn 以與此資料集連線。

```
{
  "datasetId": "<UUID>",
  "datasetName": "DatasetForAssistant",
  "datasetSource": {
    "sourceType": "KENDRA",
    "sourceFormat": "KNOWLEDGE_BASE",
    "sourceDetail": {
      "kendra": {
        "knowledgeBaseArn": "arn:aws:kendra::%s:index/index",
        "roleArn": "arn:aws:iam::%s:role/role"
      }
    }
  }
}
```

```
}  
  }  
}  
}
```

2. 使用以下命令更新資料集：

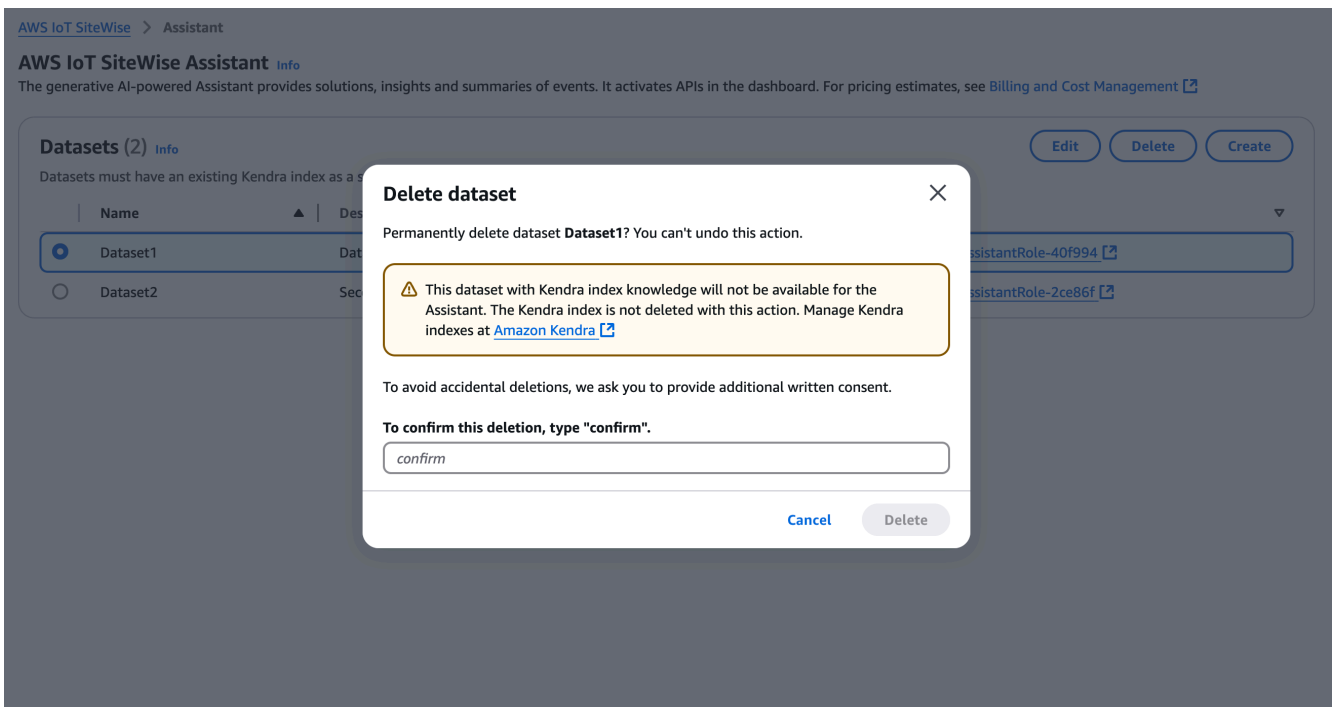
```
aws iotsitewise update-dataset --cli-input-json file://update-dataset.json --  
region us-east-1
```

刪除資料集

Console

刪除資料集

1. 資料集會顯示在助理頁面的資料集區段中。選擇資料集。選擇 **刪除**。
2. 在快顯視窗中輸入確認以確認刪除。



3. 選擇 **刪除**。

AWS CLI

刪除資料集

- 使用 `刪除資料集datasetId`。

```
aws iotsitewise delete-dataset --region us-east-1 --dataset-id <UUID>
```

AWS IoT SiteWise 助理問題

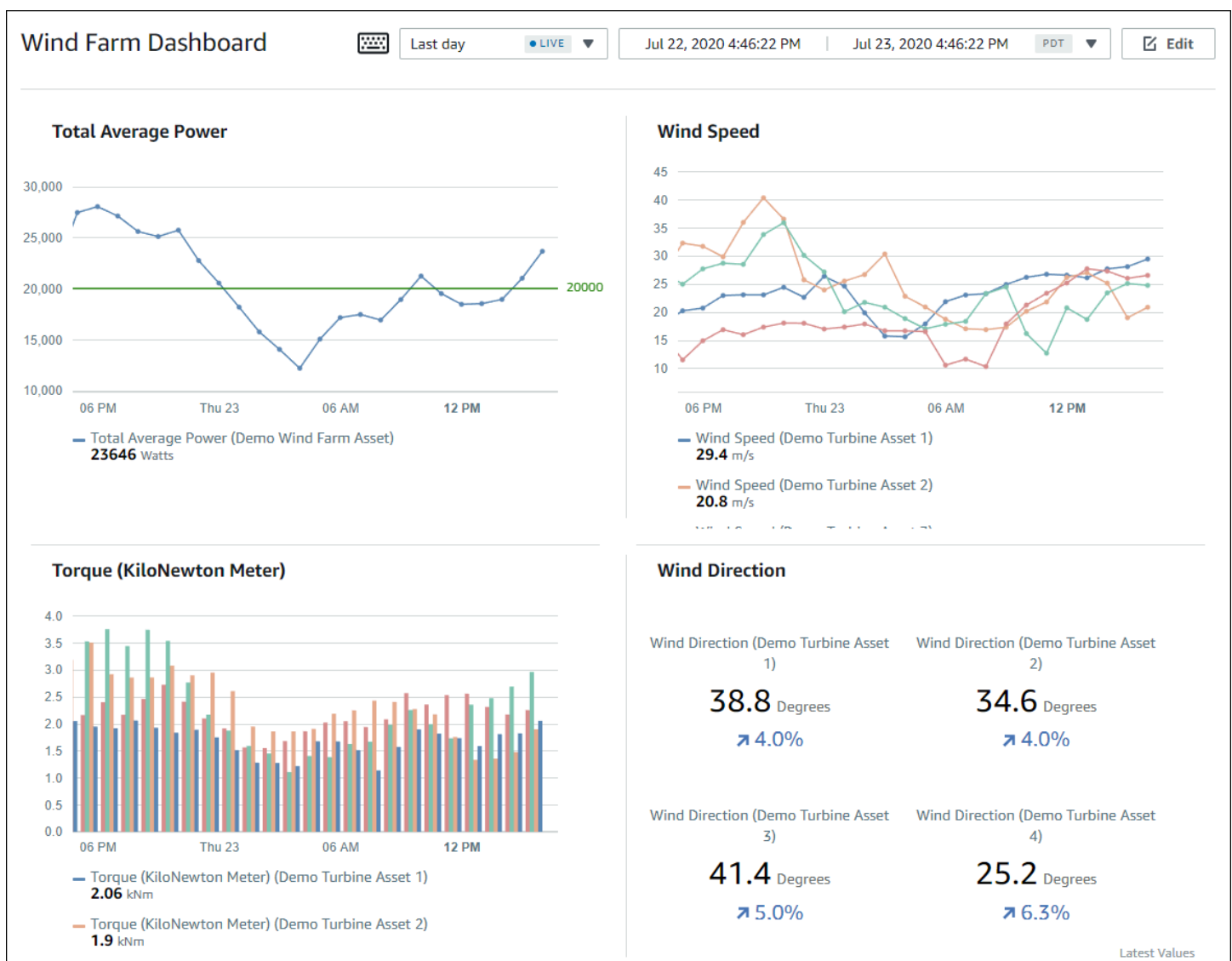
如需查詢 AWS IoT SiteWise 助理的詳細資訊[要詢問 AWS IoT SiteWise 助理的問題範例](#)，請參閱。

使用 監控資料 AWS IoT SiteWise Monitor

您可以使用 AWS IoT SiteWise 建立 SiteWise Monitor Web 入口網站，以監控程序、裝置和設備中的資料。SiteWise Monitor 是 的一項功能 AWS IoT SiteWise，您可以使用此功能，以受管 Web 應用程式的形式建立入口網站。然後，您可以使用這些入口網站來檢視和共用您的操作資料。您可以使用儀表板建立專案，將連接至 AWS IoT 的程序、裝置和設備的資料視覺化。

領域專家 (例如程序工程師) 可以使用這些入口網站，快速深入了解操作資料，以了解裝置和設備的行為。

以下是顯示風力發電場資料的範例儀表板。



由於會隨著時間 AWS IoT SiteWise 擷取資料，因此您可以使用 SiteWise Monitor 來檢視一段時間內的操作資料，或特定時間點的上次報告值。這可讓您發掘可能難以尋獲的洞見。

SiteWise Monitor 角色

四個角色與 SiteWise Monitor 互動：

AWS 管理員

AWS 管理員使用 AWS IoT SiteWise 主控台來建立入口網站。AWS 管理員也可以指派入口網站管理員及新增入口網站使用者。入口網站管理員之後可將入口網站使用者指派給專案，以做為擁有者或檢視者。AWS 管理員只能在 AWS 主控台中運作。

入口網站管理員

每個 SiteWise Monitor 入口網站都有一或多個入口網站管理員。入口網站管理員會使用入口網站來建立包含資產和儀表板集合的專案。接著，入口網站管理員會為每個專案指派資產和擁有者。入口網站管理員會透過控制對專案的存取權，指定專案擁有者和檢視者可以看到的資產。

專案擁有者

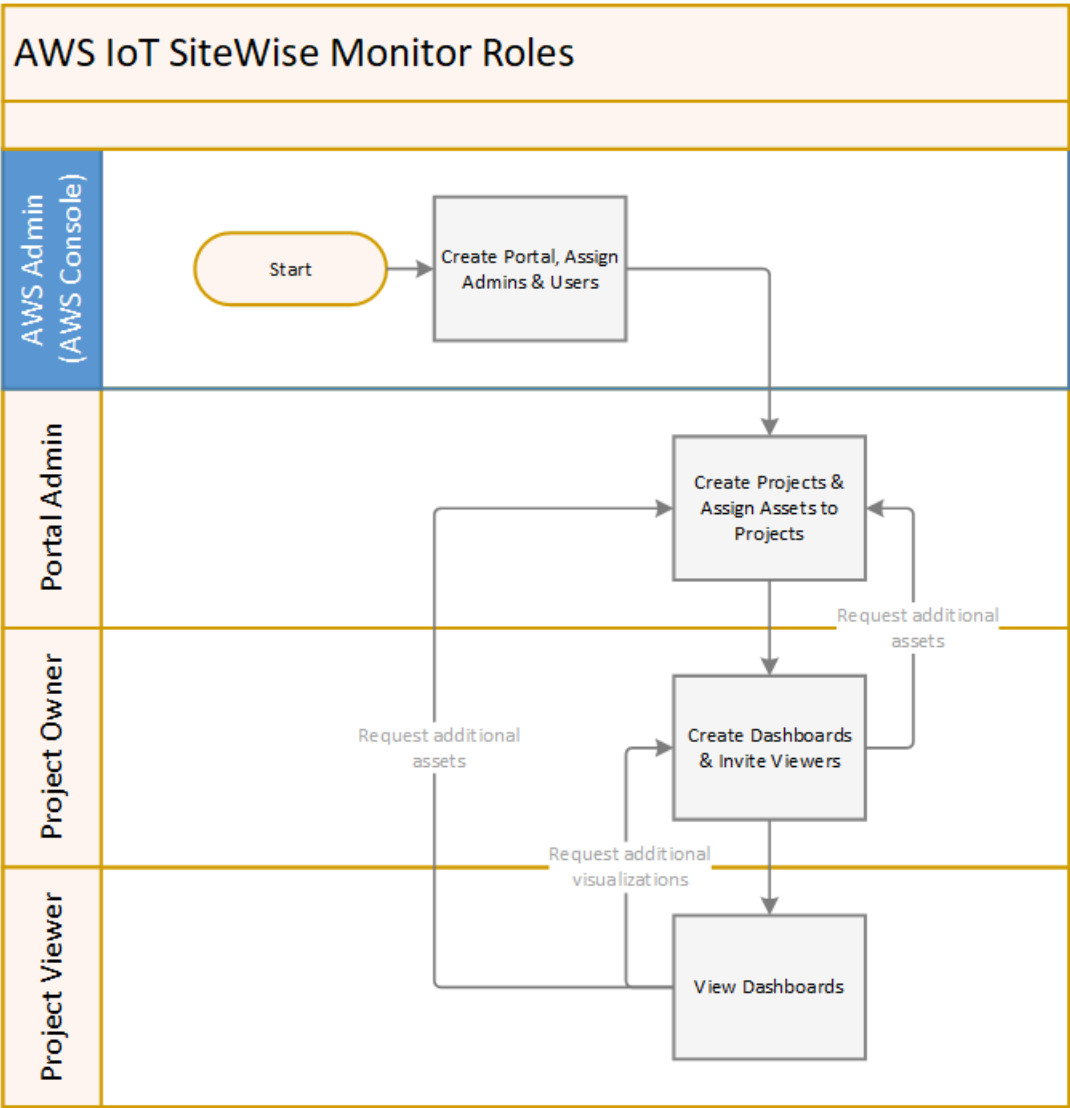
每個 SiteWise Monitor 專案都有擁有者。專案擁有者會以儀表板形式建立視覺效果，以一致的方式呈現營運資料。儀表板準備好共用時，專案擁有者可以將檢視者邀請到專案中。專案擁有者也可為專案指派其他擁有者。專案擁有者可以設定警示的閾值和通知設定。

專案檢視者

每個 SiteWise Monitor 專案都有檢視器。專案檢視者可以連線到入口網站，檢視專案擁有者建立的儀表板。在每個儀表板中，專案檢視器可以調整時間範圍，以更了解操作資料。專案檢視者只能檢視他們可存取之專案中的儀表板。專案檢視者可以確認和暫停警示。

視您的組織而定，同一個人可執行多個角色。

下圖說明這四個角色如何在 SiteWise Monitor 入口網站中互動。



您可以使用 AWS IAM Identity Center 或 IAM 來管理誰可以存取您的資料。您的資料使用者可以使用其 IAM Identity Center 或 IAM 登入資料，從桌面或行動瀏覽器登入 SiteWise Monitor。

SAML 聯合

IAM Identity Center 和 IAM 支援與 [SAML \(安全性聲明標記語言 \) 2.0](#) 的聯合身分。SAML 2.0 是許多外部身分提供者 (IdPs) 用來驗證使用者的開放標準，並將他們的身分和安全資訊傳遞給服務提供者 (SPs)。SPs 通常是應用程式或服務。SAML 聯合可讓您的 SiteWise Monitor 入口網站管理員和使用者使用外部登入資料登入其指派的入口網站，例如其公司使用者名稱和密碼。

您可以設定 IAM Identity Center 和 IAM 使用 SAML 型聯合來存取 SiteWise Monitor 入口網站。

IAM Identity Center

您的入口網站管理員和使用者可以使用其公司使用者名稱和密碼登入 AWS 存取入口網站。然後，他們可以導覽至其指派的 SiteWise Monitor 入口網站。IAM Identity Center 使用憑證來設定身分提供者與之間的 SAML 信任關係 AWS。如需詳細資訊，請參閱 AWS IAM Identity Center 《使用者指南》中的 [SCIM 設定檔和 SAML 2.0 實作](#)。

IAM

您的入口網站管理員和使用者可以請求臨時安全登入資料，以存取其指派的 SiteWise Monitor 入口網站。您可以在 IAM 中建立 SAML 身分提供者身分，以設定身分提供者與之間的信任關係 AWS。如需詳細資訊，請參閱《IAM 使用者指南》中的 [使用 SAML 型聯合進行 API 存取 AWS](#)。

您的入口網站管理員和使用者可以登入您公司的入口網站，然後選取前往 AWS 管理主控台的選項。然後，他們可以導覽至其指派的 SiteWise Monitor 入口網站。您公司的入口網站會處理身分提供者與之間的信任交換 AWS。如需詳細資訊，請參閱《IAM 使用者指南》中的 [啟用 SAML 2.0 聯合身分使用者存取 AWS 管理主控台](#)。

Note

將使用者或管理員新增至入口網站時，請避免建立限制使用者許可的 IAM 政策，例如有限的 IP。任何具有受限許可的連接政策將無法連線至 AWS IoT SiteWise 入口網站。

SiteWise Monitor 概念

若要使用 SiteWise Monitor，您應該熟悉下列概念：

Portal

AWS IoT SiteWise Monitor 入口網站是一種 Web 應用程式，可用來視覺化和共用 AWS IoT SiteWise 您的資料。入口網站有一或多個管理員，且包含零或多個專案。

專案

每個 SiteWise Monitor 入口網站都包含一組專案。每個專案都有一個與其相關聯的 AWS IoT SiteWise 資產子集。專案擁有者會建立一或多個儀表板，以便透過一致的方式檢視與這些資產相關聯的資料。專案擁有者可以將檢視者邀請到專案中，讓檢視者能夠檢視專案中的資產和儀表板。專案是 SiteWise Monitor 內共用的基本單位。專案擁有者可以邀請管理員授予入口網站存取權的使用者 AWS。必須先確保使用者具備入口網站存取權，才能與該使用者共用該入口網站中的專案。

資產

從 AWS IoT SiteWise 工業設備擷取資料時，您的裝置、設備和程序都會以資產表示。每個資產都有與其相關聯的屬性和警示。入口網站管理員會為每個專案指派資產集合。

屬性

屬性是與資產相關聯的時間序列資料。例如，一個設備可能有一個序號、一個位置、一個製造商和型號，以及一個安裝日期。它可能也有可用性、效能、品質、溫度、壓力等項目的時間序列值。

警示

警示會監控屬性，以識別設備何時超出其操作範圍。每個警示都會定義閾值和要監控的屬性。當屬性超過閾值時，警示會變成作用中，並指出您或您團隊中的某人應該解決問題。專案擁有者可以自訂警示的閾值和通知設定。專案檢視者可以確認和暫停警示，也可以留下訊息，其中包含警示的詳細資訊，或他們為了解決該警示而採取的動作。

儀表板

每個專案都包含一組儀表板。儀表板可提供一組資產集合值的視覺效果。專案擁有者會建立儀表板和包含的視覺效果。當專案擁有者準備好共用該組儀表板時，擁有者可以將檢視者邀請到專案中，讓他們能夠存取專案中的所有儀表板。如果您需要另一組適用於不同儀表板的檢視者，您必須分割專案之間的儀表板。當瀏覽者查看儀表板時，他們可以自訂時間範圍來查看特定資料。

視覺效果

在每個儀表板中，專案擁有者會決定如何顯示與專案相關聯的資產屬性和警示。可用性可能以折線圖表示，而其他值可能以長條圖或關鍵績效指標 (KPIs) 顯示。警示最好顯示為狀態網格和狀態時間表。專案擁有者會自訂每個視覺效果，讓使用者充分理解該資產的資料。

開始使用 AWS IoT SiteWise Monitor (傳統)

如果您是組織的 AWS 管理員，您可以從 AWS IoT SiteWise 主控台建立入口網站。請完成下列步驟來建立入口網站，以便您的組織成員可以檢視 AWS IoT SiteWise 您的資料：

1. 設定及建立入口網站
2. 新增入口網站管理員及傳送邀請電子郵件
3. 新增入口網站使用者

建立入口網站後，入口網站管理員可以檢視您的 AWS IoT SiteWise 資產，並將其指派給入口網站中的專案。然後，專案擁有者可以建立儀表板，將資產的屬性視覺化，協助專案檢視者了解裝置、程序和設備的執行情況。

Note

將使用者或管理員新增至入口網站時，請避免建立限制使用者許可的 AWS Identity and Access Management (IAM) 政策，例如有限的 IP。任何具有受限許可的附加政策將無法連線至 AWS IoT SiteWise 入口網站。

您可以按照教學，透過必要步驟來使用風力發電廠資料，針對特定情形建立具有專案、儀表板和多個使用者的入口網站。如需詳細資訊，請參閱[在 SiteWise Monitor 中視覺化和共用風力發電廠資料](#)。

主題

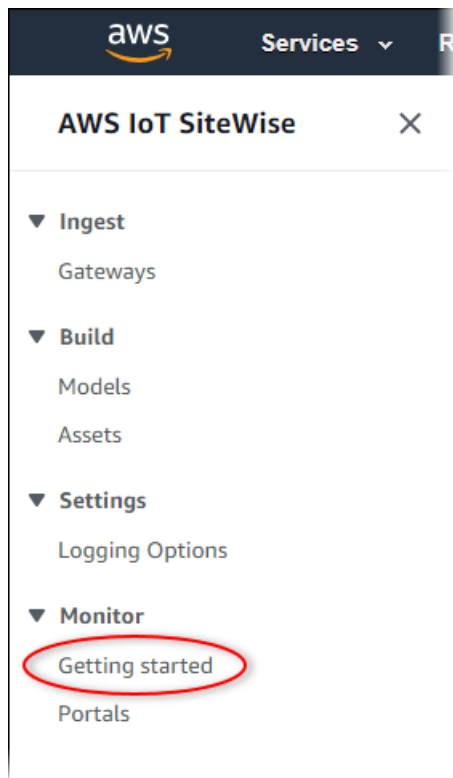
- [在 SiteWise Monitor 中建立入口網站](#)
- [在 SiteWise Monitor 中設定您的入口網站](#)
- [在 SiteWise Monitor 中邀請管理員](#)
- [在 SiteWise Monitor 中新增入口網站使用者](#)
- [建立 AWS IoT SiteWise 儀表板 \(AWS CLI\)](#)
- [在中開啟入口網站的警示 AWS IoT SiteWise](#)
- [在邊緣啟用您的 AWS IoT SiteWise 入口網站](#)
- [管理您的 SiteWise Monitor 入口網站](#)

在 SiteWise Monitor 中建立入口網站

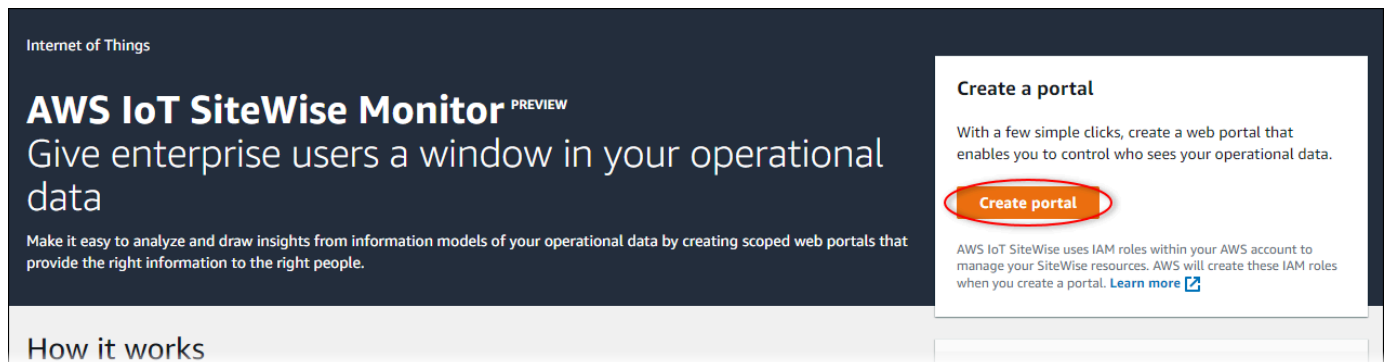
您可以在 AWS IoT SiteWise 主控台中建立 SiteWise Monitor 入口網站。

建立入口網站

1. 登入 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Monitor (監視器)、Getting started (入門)。



3. 選擇 Create Portal (建立入口網站)。



接下來，您必須提供一些基本資訊來設定入口網站。

在 SiteWise Monitor 中設定您的入口網站

使用者會使用入口網站檢視您的資料。您可以自訂入口網站的名稱、描述、品牌、使用者身分驗證、支援聯絡人電子郵件和許可。

Step 1
Portal configurationStep 2- optional
Additional featuresStep 3
Invite administratorsStep 4
Assign users

Portal configuration

Each web portal provides enterprise users with access to your IoT SiteWise assets. [Learn more](#)

Portal details

Portal name

Choose a portal name to identify the web portal to your users. Company name is recommended.

example-factory-1

Name should be 1-128 characters and only contain A-Z a-z 0-9 _ and -.

Description - optional

Create a description of your portal

Example Corp Factory #1 in Renton, WA

Description should contain a maximum of 2048 characters.

Portal branding

You can provide your logo image to display your brand in this web portal.

Logo image

Upload a square, high-resolution .png file. The image is displayed on a dark background.

Choose file

The file size must be less than 1 MB.

User authentication

Your users can sign in to this portal with their AWS Single Sign-On (AWS SSO) or AWS Identity and Access Management (IAM) credentials. If you choose AWS SSO, you must enable the service for your AWS account.

 You haven't enabled AWS SSO in your account yet. When you create your first portal user, this automatically enables AWS SSO in your AWS account.

Create user

☒ AWS SSO

Your users can sign in to the portal with their corporate usernames and passwords.

☐ IAM

Your users can sign in to the portal with their IAM credentials.

Support contact email

You can provide an email address for cases where there's a problem or issue with this portal and your users need to contact support to resolve.

Email

support@example.com

Tags

This resource doesn't have any tags.

Add tag

You can add up to 50 more tags.

Permissions

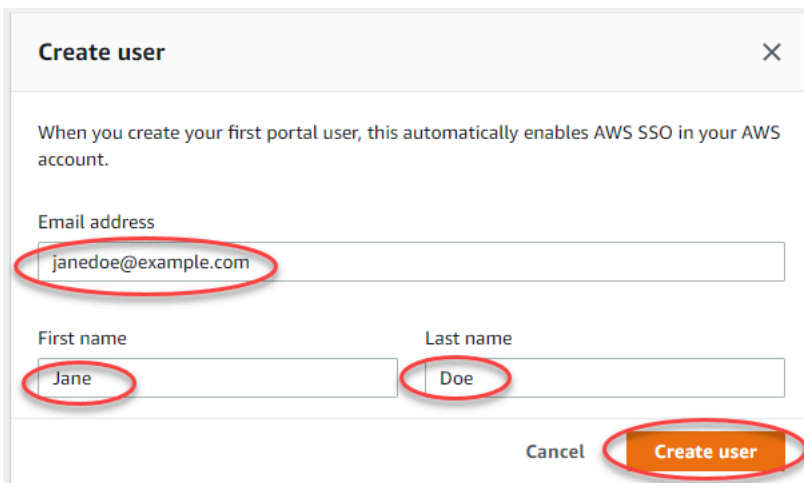
SiteWise Monitor assumes this role to give permissions to your federated users to access AWS IoT SiteWise resources. [Learn](#)

設定入口網站

1. 輸入入口網站的名稱。
2. (選擇性) 輸入入口網站的描述。如果您有多個入口網站，請使用有意義的描述來協助您追蹤每個入口網站包含的內容。
3. (選用) 上傳影像以在入口網站中顯示您的品牌。選擇正方形 PNG 影像。如果您上傳非正方形影像，入口網站會將影像縮小為正方形。
4. 請選擇下列其中一個選項：
 - 如果您的入口網站使用者使用其公司使用者名稱和密碼登入此入口網站，請選擇 IAM Identity Center。

如果您尚未在帳戶中啟用 IAM Identity Center，請執行下列動作：

- a. 選擇 Create user (建立使用者)。
- b. 在建立使用者頁面上，若要建立第一個入口網站，請輸入使用者的電子郵件地址、名字和姓氏，然後選擇建立使用者。



Note

- AWS 當您建立第一個入口網站使用者時，會自動在您的帳戶中啟用 IAM Identity Center。
- 您一次只能在一個區域中設定 IAM Identity Center。SiteWise Monitor 會連線至您為 IAM Identity Center 設定的區域。這表示您使用一個區域進行 IAM Identity Center 存取，但您可以在任何區域中建立入口網站。

- 如果您的入口網站使用者使用其 IAM 登入資料登入此入口網站，請選擇 IAM。

 Important

使用者或角色必須具有登入入口網站的 `iotsitewise:DescribePortal` 許可。

5. 輸入入口網站使用者在入口網站發生問題時可以聯絡並需要協助解決此問題的電子郵件地址。
6. (選用) 為您的入口網站新增標籤。如需詳細資訊，請參閱 [標記您的 AWS IoT SiteWise 資源](#)。
7. 請選擇下列其中一個選項：
 - 選擇建立並使用新的服務角色。根據預設，SiteWise Monitor 會自動為每個入口網站建立服務角色。此角色可讓您的入口網站使用者存取您的 AWS IoT SiteWise 資源。如需詳細資訊，請參閱 [使用的服務角色 AWS IoT SiteWise Monitor](#)。
 - 選擇使用現有的服務角色，然後選擇目標角色。
8. 選擇下一步
9. (選用) 啟用入口網站的警示。如需詳細資訊，請參閱 [在中開啟入口網站的警示 AWS IoT SiteWise](#)。
10. 選擇建立。AWS IoT SiteWise 將建立您的入口網站。

 Note

如果您關閉主控台，您可以透過新增管理員和使用者來完成安裝程序。如需詳細資訊，請參閱 [在中新增或移除入口網站管理員 AWS IoT SiteWise](#)。如果您不想保留此入口網站，請將其刪除，使其不使用資源。如需詳細資訊，請參閱 [在中刪除入口網站 AWS IoT SiteWise](#)。

狀態欄可以是下列其中一個值。

- CREATING - AWS IoT SiteWise 正在處理您的建立入口網站的請求。此程序可能需要幾分鐘的時間才能完成。
- 正在處理更新入口網站的 AWS IoT SiteWise 請求。此程序可能需要幾分鐘的時間才能完成。
- 待定 - AWS IoT SiteWise 正在等待 DNS 記錄傳播完成。此程序可能需要幾分鐘的時間才能完成。當狀態為待定时，您可以刪除入口網站。

- 刪除 - AWS IoT SiteWise 正在處理您的刪除入口網站的請求。此程序可能需要幾分鐘的時間才能完成。
- ACTIVE - 當入口網站變成作用中時，您的入口網站使用者可以存取它。
- 失敗 - AWS IoT SiteWise 無法處理您的建立、更新或刪除入口網站的請求。如果您啟用 AWS IoT SiteWise 將日誌傳送至 Amazon CloudWatch Logs，您可以使用這些日誌來疑難排解問題。如需詳細資訊，請參閱[AWS IoT SiteWise 使用 CloudWatch Logs 進行監控](#)。

建立入口網站時，會出現一則訊息。

☑ Successfully created portal URL at <https://a1b2c3d4-5678-90ab-cdef-1111EXAMPLE.app.iotsitewise.aws>



接下來，您必須邀請一或多位入口網站管理員前往入口網站。到目前為止，您建立了一個入口網站，但沒有人可以存取它。

在 SiteWise Monitor 中邀請管理員

若要開始使用新入口網站，您必須指派入口網站管理員。入口網站管理員會建立專案、選擇專案擁有者，以及將資產指派給專案。入口網站管理員可以查看您的所有 AWS IoT SiteWise 資產。

根據使用者身分驗證服務，選擇下列其中一個選項：

IAM Identity Center

如果您是第一次使用 SiteWise Monitor，您可以選擇您先前建立的使用者做為入口網站管理員。如果您想要將其他使用者新增為入口網站管理員，您可以從此頁面建立 IAM Identity Center 使用者。或者，您可以將外部身分提供者連線至 IAM Identity Center。如需詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》<https://docs.aws.amazon.com/singlesignon/latest/userguide/>。

邀請管理員

1. 選取您想要做為入口網站管理員之使用者的核取方塊。這會將使用者新增至入口網站管理員清單。

Note

如果您使用 IAM Identity Center 做為身分存放區，且已登入管理 AWS Organizations 帳戶，您可以選擇建立使用者來建立 IAM Identity Center 使用者。IAM Identity Center 會傳送電子郵件給新使用者，讓他們設定密碼。然後，您可以將使用者以管理員身分指派給入口網站。如需詳細資訊，請參閱[在 IAM Identity Center 中管理身分](#)。

2. (選擇性) 選擇 Send invite to selected users (將邀請傳送給選取的使用者)。您的電子郵件用戶端隨即開啟，並在訊息內文中填入邀請。

您可以先自訂電子郵件，然後再將其傳送給入口網站管理員。您也可以稍後將電子郵件傳送給入口網站管理員。如果您是第一次嘗試 SiteWise Monitor，並將新的 IAM Identity Center 或 IAM 使用者或角色新增為入口網站管理員，則不需要自行傳送電子郵件。

3. 如果您新增的使用者不是要做為系統管理員，請清除該使用者的核取方塊。
4. 當您完成入口網站管理員的邀請時，請選擇 Next (下一步)。

IAM

您可以選擇使用者或角色做為入口網站管理員。如果您想要將其他使用者或角色新增為入口網站管理員，您可以在 IAM 主控台中建立使用者或角色。如需詳細資訊，請參閱 [《IAM 使用者指南》中的在帳戶中 AWS 建立 IAM 使用者](#) 和 [建立 IAM 角色](#)。

邀請管理員

1. 請執行下列操作：
 - 選擇 IAM 使用者，將 IAM 使用者新增為您的入口網站管理員。
 - 選擇 IAM 角色，將 IAM 角色新增為您的入口網站管理員。
2. 選取您要做為入口網站管理員的使用者或角色的核取方塊。這會將使用者或角色新增至入口網站管理員清單。
3. 如果您新增不想要以管理員身分擔任的使用者或角色，請清除該使用者或角色的核取方塊。
4. 當您完成入口網站管理員的邀請時，請選擇 Next (下一步)。

Important

使用者或角色必須具有登入入口網站的 `iotsitewise:DescribePortal` 許可。

Note

如果您使用 IAM Identity Center 做為身分存放區，且已登入管理 AWS Organizations 帳戶，您可以選擇建立使用者以建立 IAM Identity Center 使用者。IAM Identity Center 會傳送

電子郵件給新使用者，讓他們設定密碼。然後，您可以將使用者以管理員身分指派給入口網站。如需詳細資訊，請參閱[在 IAM Identity Center 中管理身分](#)。

您可以稍後變更入口網站管理員的清單。如需詳細資訊，請參閱[在 中新增或移除入口網站管理員 AWS IoT SiteWise](#)。

Note

由於只有入口網站管理員可以建立專案並將資產指派給他們，因此您應該指定至少一個入口網站管理員。

在最後一個步驟中，您會新增可存取新入口網站的使用者。

在 SiteWise Monitor 中新增入口網站使用者

您可以控制哪些使用者可以存取您的入口網站。在每個入口網站中，入口網站管理員會建立一個或多個專案，並將入口網站使用者指派為每個專案的擁有者或檢視者。每個專案擁有者都可以邀請其他入口網站使用者來擁有或檢視專案。

根據使用者身分驗證服務，選擇下列其中一個選項：

IAM Identity Center

如果您想要將使用者新增至使用者清單，請完成下列步驟。

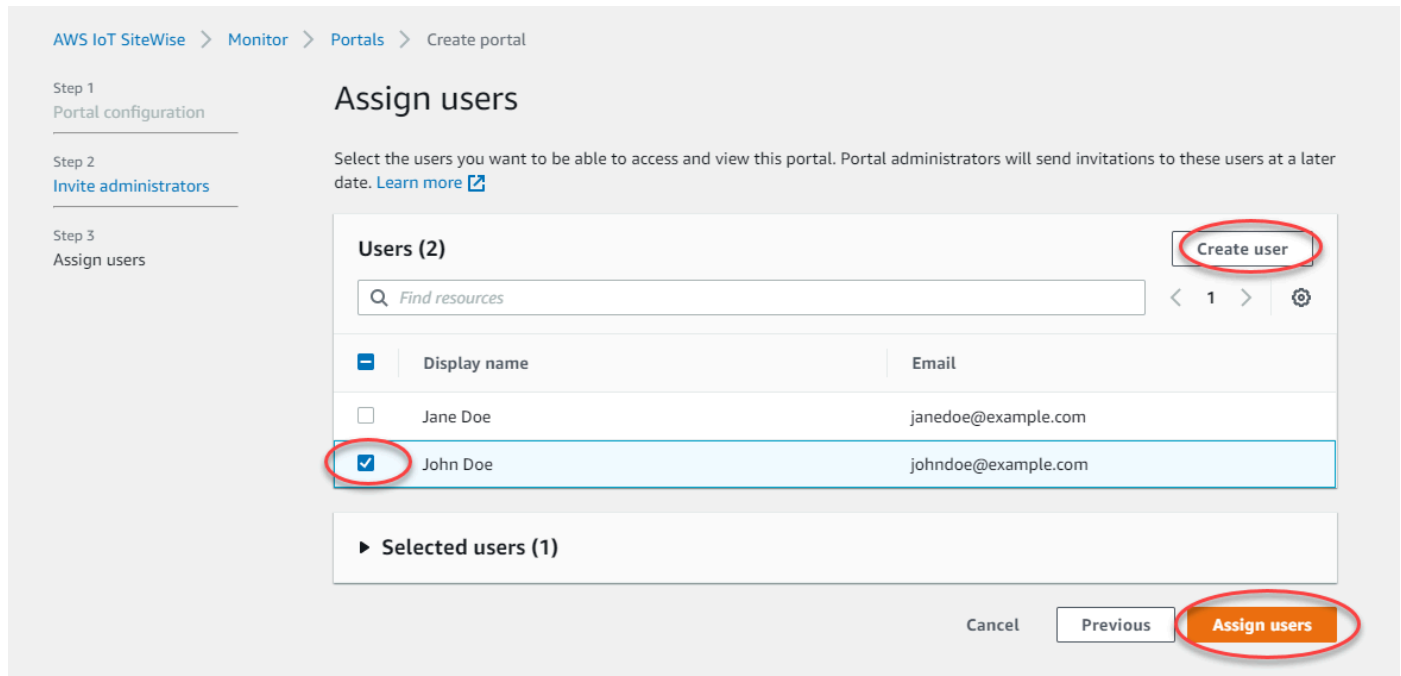
新增入口網站使用者

1. 從使用者清單中選擇使用者，以新增至入口網站。這會將使用者新增至入口網站使用者清單。如果您是第一次使用 SiteWise Monitor，則不需要將入口網站管理員新增為入口網站使用者。

Note

如果您使用 IAM Identity Center 做為身分存放區，且已登入管理 AWS Organizations 帳戶，您可以選擇建立使用者以建立 IAM Identity Center 使用者。IAM Identity Center 會傳送電子郵件給新使用者，讓他們設定密碼。然後，您可以將使用者以使用者身分指派給入口網站。如需詳細資訊，請參閱[在 IAM Identity Center 中管理身分](#)。

- 如果您新增使用者，而您不希望該使用者具有入口網站的存取權，請清除該使用者的核取方塊。
- 完成選取使用者後，請選擇指派使用者。



IAM

如果您看到要在 IAM 使用者或 IAM 角色清單中新增的使用者或角色，請完成以下步驟。

新增入口網站使用者

- 執行下列選項：
 - 選擇 IAM 使用者，將 IAM 使用者新增為入口網站使用者。
 - 選擇 IAM 角色，將 IAM 角色新增為入口網站使用者。

如果您是第一次使用 SiteWise Monitor，則不需要將入口網站管理員新增為入口網站使用者。

- 選取您想要以入口網站使用者身分使用之使用者或角色的核取方塊。這會將使用者或角色新增至入口網站使用者清單。
- 如果您新增使用者，而您不希望該使用者具有入口網站的存取權，請清除該使用者的核取方塊。
- 完成選取使用者後，請選擇指派使用者。

⚠ Important

使用者或角色必須具有登入入口網站的*iotsitewise:DescribePortal*許可。

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2
[Invite administrators](#)

Step 3
Assign users

Assign users

Select the users you want to be able to access and view this portal. Portal administrators will send invitations to these users at a later date. [Learn more](#)

Users Roles

IAM users (1) [Manage users in IAM console](#)

Find user name

☒	Name	Date created
☒	raspberrypi-testing	11-08-2019

► Portal users (1) [Remove](#)

Cancel Previous **Assign users**

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2
Invite administrators

Step 3
Assign users

Assign users

Select the users you want to be able to access and view this portal. Portal administrators will send invitations to these users at a later date. [Learn more](#)

Users **Roles**

IAM roles (66) [Manage roles in IAM console](#)

Find role name

	Name	Date created
☐	[REDACTED]	
☐	[REDACTED]	
☒	AWSIoTSiteWiseMonitorServiceRole_4wZigNpA1	03-16-2021
☐	AWSIoTSiteWiseMonitorServiceRole_ECKt-2Oar	03-11-2021
☐	AWSIoTSiteWiseMonitorServiceRole_GTnd0O4Wr	03-16-2021
☐	AWSIoTSiteWiseMonitorServiceRole_rHINLNCS-	03-11-2021
☐	AWSIoTSiteWiseMonitorServiceRole_XB330QUIO	03-10-2021
☐	[REDACTED]	
☐	[REDACTED]	
☐	[REDACTED]	

► Portal users (2) [Remove](#)

Cancel Previous **Assign users**

恭喜您！您已成功建立入口網站、指派的入口網站管理員，以及指派可在受邀時使用該入口網站的使用者。您的入口網站管理員現在可以建立專案，並將資產新增到那些專案。然後，您的專案擁有者可以建立儀表板，以將每個專案資產的資料視覺化。

您可以稍後變更入口網站使用者的清單。如需詳細資訊，請參閱[在中新增或移除入口網站使用者 AWS IoT SiteWise](#)。

如果您需要對入口網站進行變更，請參閱[管理您的 SiteWise Monitor 入口網站](#)。

若要在入口網站中開始使用，請參閱 SiteWise Monitor 應用程式指南中的[入門](#)。

建立 AWS IoT SiteWise 儀表板 (AWS CLI)

當您使用 在儀表板中定義視覺效果（或小工具）時 AWS CLI，您必須在 `dashboardDefinition` JSON 文件中指定下列資訊。此定義是 [CreateDashboard](#) 和 [UpdateDashboard](#) 操作的參數。

widgets

Widget 定義結構的清單，每個結構都包含下列資訊：

type

widget. 的類型 AWS IoT SiteWise 提供下列小工具類型：

- `sc-line-chart` – 折線圖。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[折線圖](#)。
- `sc-scatter-chart` – 散佈圖。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[散佈圖](#)。
- `sc-bar-chart` – 長條圖。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[長條圖](#)。
- `sc-status-grid` – 狀態小工具，可將資產屬性的最新值顯示為網格。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[狀態小工具](#)。
- `sc-status-timeline` – 狀態小工具，可將資產屬性的歷史值顯示為時間軸。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[狀態小工具](#)。
- `sc-kpi` – 關鍵效能指標 (KPI) 視覺化。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[KPI 小工具](#)。
- `sc-table` – 資料表小工具。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[資料表小工具](#)。

title

Widget 的標題。

x

Widget 的水平位置，從格線的左側開始。此值是指 Widget 在儀表板格線中的位置。

y

Widget 的垂直位置，從格線的頂部開始。此值是指 Widget 在儀表板格線中的位置。

width

Widget 的寬度，以儀表板格線上的空格數表示。

height

Widget 的高度，以儀表板格線上的空格數表示。

metrics

指標結構的清單，每個結構都會定義此 Widget 的資料串流。清單中的每個結構都必須包含下列資訊：

label

要為此指標顯示的標籤。

type

此 metric. 的資料來源類型 AWS IoT SiteWise 提供下列指標類型：

- `iotsitewise` – 儀表板會擷取資產屬性的資料 AWS IoT SiteWise。如果您選擇此選項，則必須為此指標定義 `assetId` 和 `propertyId`。

assetId

(選用) AWS IoT SiteWise 中的資產 ID。

如果您在此指標中為 `type` 選擇 `iotsitewise`，則此欄位為必要。

propertyId

(選用) AWS IoT SiteWise 中的資產屬性 ID。

如果您在此指標中為 `type` 選擇 `iotsitewise`，則此欄位為必要。

analysis

(選用) 定義分析的結構，例如趨勢線，以針對小工具顯示。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[設定趨勢行](#)。您可以在 Widget 中為每個屬性新增每種趨勢線的其中一個類型。分析結構包含下列資訊：

trends

(選用) 趨勢結構清單，每個結構都會定義此小工具的趨勢分析。清單中的每個結構都包含下列資訊：

type

趨勢線的類型。選擇下列選項：

- `linear-regression` – 顯示線性迴歸線。SiteWise Monitor 使用[最小平方](#)方法來計算線性迴歸。

`annotations`

(選用) 定義小工具閾值的註釋結構。如需詳細資訊，請參閱《AWS IoT SiteWise Monitor 應用程式指南》中的[設定閾值](#)。每個小工具最多可以新增六個註釋。註釋結構包含下列資訊：

`y`

(選用) 註釋結構的清單，每個結構都會定義此小工具的水平閾值。清單中的每個結構都包含下列資訊：

`comparisonOperator`

閾值的比較運算子。選擇下列其中一項：

- `LT` – 反白顯示至少有一個資料點小於 的屬性`value`。
- `GT` – 反白顯示至少有一個資料點大於 的屬性`value`。
- `LTE` – 反白顯示至少有一個資料點小於或等於 的屬性`value`。
- `GTE` – 反白顯示至少有一個資料點大於或等於 的屬性`value`。
- `EQ` – 反白顯示至少有一個資料點等於 的屬性`value`。

`value`

將資料點與 進行比較的閾值`comparisonOperator`。

`color`

(選用) 閾值顏色的 6 位數十六進位代碼。對於至少有一個資料點符合閾值規則的屬性，視覺效果會顯示此顏色的屬性圖例。預設為黑色 (`#000000`)。

`showValue`

(選用) 是否在小工具的邊界中顯示閾值的值。預設為 `true`。

`properties`

(選用) 小工具屬性的平面字典。此結構的成員與內容有關。AWS IoT SiteWise 提供使用的下列小工具`properties`：

- [折線圖](#)、[散佈圖](#)和[長條圖](#)具有下列屬性：

`colorDataAcrossThresholds`

(選用) 是否要變更此小工具中超過閾值的資料顏色。當您啟用此選項時，超過閾值的資料會以您選擇的顏色顯示。預設為 `true`。

- [狀態網格](#)具有下列屬性：

labels

(選用) 定義要在狀態網格上顯示的標籤的結構。標籤結構包含下列資訊：

showValue

(選用) 是否要顯示此小工具中每個資產屬性的單位和值。預設為 true。

Example 範例儀表板定義

下列範例會從存放在 JSON 檔案中的承載來定義儀表板。

```
aws iotsitewise create-dashboard \  
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeEXAMPLE \  
  --dashboard-name "Wind Farm Dashboard" \  
  --dashboard-definition file://dashboard-definition.json
```

dashboard-definition.json 的下列 JSON 範例，會使用下列視覺效果 Widget 來定義儀表板：

- 在儀表板左上角將風力發電廠總電力視覺化的折線圖。此折線圖包含閾值，指出風力發電廠輸出的功率低於其預期輸出下限。此折線圖也包含線性迴歸趨勢線。
- 在儀表板右上角將四個渦輪機風速視覺化的長條圖。

Note

此範例代表儀表板上的折線圖視覺化效果。此儀表板類似於[範例風力發電廠儀表板](#)。

```
{  
  "widgets": [  
    {  
      "type": "sc-line-chart",  
      "title": "Total Average Power",  
      "x": 0,  
      "y": 0,  
      "height": 3,  
      "width": 3,  
      "metrics": [  
        {
```

```

        "label": "Power",
        "type": "iotsitewise",
        "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
        "propertyId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
        "analysis": {
            "trends": [
                {
                    "type": "linear-regression"
                }
            ]
        }
    },
    "annotations": {
        "y": [
            {
                "comparisonOperator": "LT",
                "value": 20000,
                "color": "#D13212",
                "showValue": true
            }
        ]
    }
},
{
    "type": "sc-bar-chart",
    "title": "Wind Speed",
    "x": 3,
    "y": 3,
    "height": 3,
    "width": 3,
    "metrics": [
        {
            "label": "Turbine 1",
            "type": "iotsitewise",
            "assetId": "a1b2c3d4-5678-90ab-cdef-2a2a2EXAMPLE",
            "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE"
        },
        {
            "label": "Turbine 2",
            "type": "iotsitewise",
            "assetId": "a1b2c3d4-5678-90ab-cdef-2b2b2EXAMPLE",
            "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE"
        }
    ],

```

```
{
  "label": "Turbine 3",
  "type": "iotsitewise",
  "assetId": "a1b2c3d4-5678-90ab-cdef-2c2c2EXAMPLE",
  "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE"
},
{
  "label": "Turbine 4",
  "type": "iotsitewise",
  "assetId": "a1b2c3d4-5678-90ab-cdef-2d2d2EXAMPLE",
  "propertyId": "a1b2c3d4-5678-90ab-cdef-55555EXAMPLE"
}
]
}
]
```

在 中開啟入口網站的警示 AWS IoT SiteWise

您可以 AWS IoT Events 為入口網站啟用 支援的警示功能，讓入口網站管理員可以在 SiteWise Monitor 入口網站中建立、編輯和刪除 AWS IoT Events 警示模型。專案擁有者可以設定警示。專案檢視器可以檢視警示詳細資訊。本節說明如何使用 AWS IoT SiteWise 主控台來啟用入口網站的警示功能。

Important

- 您無法在入口網站中建立外部警示。
- 如果您想要傳送警示通知，您必須為使用者身分驗證服務選擇 IAM Identity Center。
- 警示通知功能不適用於中國（北京）AWS 區域。

當您設定和建立入口網站時，您可以在步驟 2 其他功能中啟用警示和警示通知。根據使用者身分驗證服務，選擇下列其中一個選項：

IAM Identity Center

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2 - optional
Additional features

Step 3
Invite administrators

Step 4
Assign users

Additional features - *optional*

Alarms

Your portal users can create alarms in the portal to monitor equipment or processes. They can also get notified when the equipment or processes perform outside specified range.

☒ **Enable alarms**
If enabled, your portal users can define AWS IoT Events alarms in SiteWise Monitor.

AWS IoT SiteWise access role
Choose an IAM role that allows AWS IoT Events to send data to AWS IoT SiteWise. To edit the role, go to the [IAM console](#).

☒ Create a role from an AWS managed template

☐ Use an existing role

☒ **Enable alarm notifications**
If enabled, alarms can send email or SMS notifications.

Sender
Specify the email address that sends alarm notifications. To edit or add a sender, go to the [Amazon SES console](#).

AWS Lambda role
Choose an IAM role that allows AWS Lambda to send data to Amazon SES and Amazon SNS. To edit the role, go to the [IAM console](#).

☒ Create a role from an AWS managed template

☐ Use an existing role

AWS Lambda function
Choose an AWS Lambda function to manage alarm notifications. To edit the function, go to the [AWS Lambda console](#).

☒ Create a lambda from an AWS managed template

☐ Use an existing lambda

Previous **Create**

啟用入口網站的警示

- （選用）選擇啟用警示。
 - 對於AWS IoT SiteWise 存取角色，請使用現有角色或建立具有所需許可的角色。此角色需要*iotevents:BatchPutMessage*許可和信任關係，允許 *iot.amazonaws.com*和 *iotevents.amazonaws.com* 擔任該角色。
- （選用）選擇啟用警示通知。
 - 針對寄件者，選擇寄件者。

⚠ Important

您必須在 Amazon SES 中驗證寄件者電子郵件地址。如需詳細資訊，請參閱 [《Amazon Simple Email Service 開發人員指南》](#) 中的在 Amazon SES 中驗證電子郵件地址。

- b. 對於AWS Lambda 角色，請使用現有角色或建立具有所需許可的角色。此角色需要 `lambda:InvokeFunction` 和 `sso-directory:DescribeUser` 許可，以及允許 `iotevents.amazonaws.com` 和 `lambda.amazonaws.com` 擔任角色的信任關係。
- c. 針對AWS Lambda 函數，選擇現有的 Lambda 函數或建立管理警示通知的函數。如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的 [管理警示通知](#)。

IAM

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2- optional
Additional features

Step 3
Invite administrators

Step 4
Assign users

Additional features - optional

Alarms

Your portal users can create alarms in the portal to monitor equipment or processes. They can also get notified when the equipment or processes perform outside specified range.

☒ **Enable alarms**
If enabled, your portal users can define AWS IoT Events alarms in SiteWise Monitor.

AWS IoT SiteWise access role
Choose an IAM role that allows AWS IoT Events to send data to AWS IoT SiteWise. To edit the role, go to the [IAM console](#).

☒ **Create a role from an AWS managed template**
☐ Use an existing role

ⓘ Alarms created in the portal can't send notifications. If you want to send alarm notifications, choose **Previous**. Then, on the **Portal configuration** page, choose **AWS SSO** for **User authentication**.

Previous **Create**

啟用入口網站的警示

- (選用) 選擇啟用警示。
 - 對於AWS IoT SiteWise 存取角色，請使用現有角色或建立具有所需許可的角色。此角色需要 `iotevents:BatchPutMessage` 許可和信任關係，允許 `iot.amazonaws.com` 和 `iotevents.amazonaws.com` 擔任該角色。

如需 SiteWise Monitor 中警示的詳細資訊，請參閱《AWS IoT SiteWise 應用程式指南》中的[使用警示進行監控](#)。

在邊緣啟用您的 AWS IoT SiteWise 入口網站

在邊緣啟用入口網站後，此入口網站可在所有 SiteWise Edge 閘道上使用，並在您的帳戶中啟用資料處理套件。

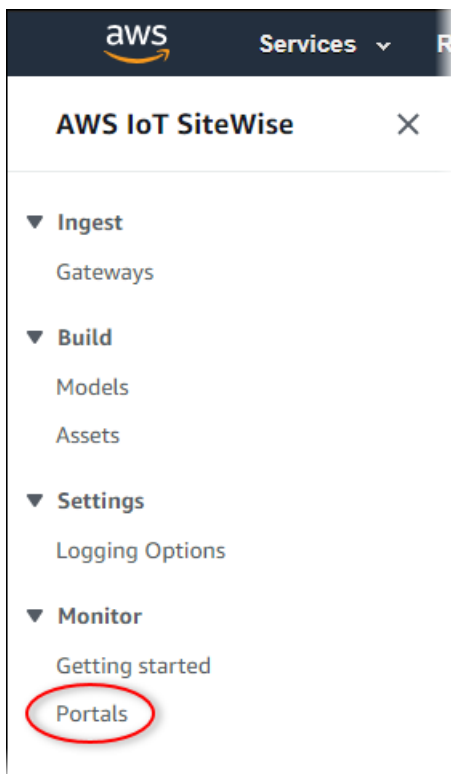
在邊緣啟用入口網站

1. 在 Edge 組態區段中，開啟在邊緣啟用此入口網站。
2. 選擇 Create (建立)。

管理您的 SiteWise Monitor 入口網站

您可以管理和設定入口網站的各個層面。這包括新增和移除使用者或管理員、設定使用者許可和角色、自訂入口網站的 URL、名稱、設定支援聯絡資訊，以及傳送電子郵件邀請給入口網站管理員。

1. 登入 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Monitor (監視器)、Portals (入口網站)。



3. 選擇入口網站，然後選擇 View details (檢視詳細資訊) (或選擇入口網站的 Name (名稱))。

4. 您可以執行下列任一管理任務：

- [在中變更入口網站詳細資訊 AWS IoT SiteWise](#)
- [在中新增或移除入口網站管理員 AWS IoT SiteWise](#)
- [傳送電子郵件邀請給入口網站管理員](#)
- [在中新增或移除入口網站使用者 AWS IoT SiteWise](#)
- [在中刪除入口網站 AWS IoT SiteWise](#)

如需如何建立入口網站的相關資訊，請參閱[開始使用 AWS IoT SiteWise Monitor \(傳統\)](#)。

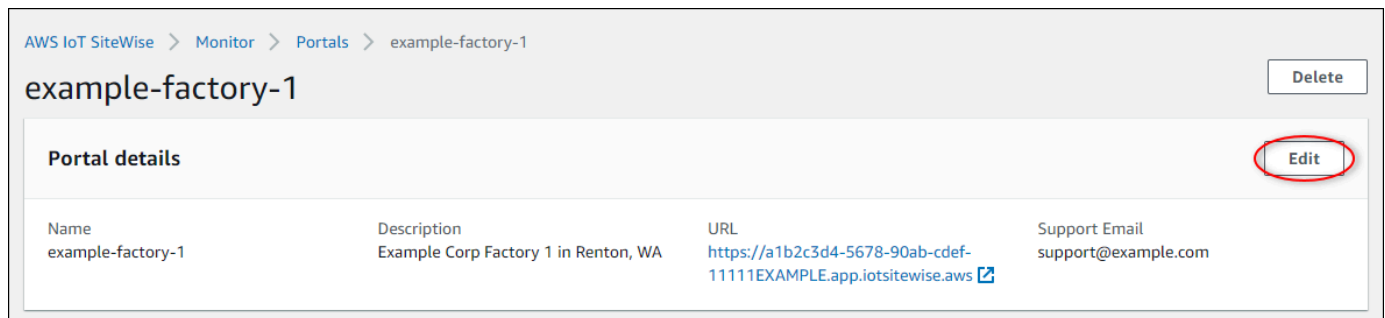
主題

- [在中變更入口網站詳細資訊 AWS IoT SiteWise](#)
- [在中新增或移除入口網站管理員 AWS IoT SiteWise](#)
- [傳送電子郵件邀請給入口網站管理員](#)
- [在中新增或移除入口網站使用者 AWS IoT SiteWise](#)
- [在中刪除入口網站 AWS IoT SiteWise](#)

在中變更入口網站詳細資訊 AWS IoT SiteWise

您可以變更入口網站的名稱、描述、品牌、支援電子郵件和許可。

1. 在入口網站詳細資訊頁面上 Portal details (入口網站詳細資訊) 區段中，選擇 Edit (編輯)。



2. 更新 Name (名稱)、Description (說明)、Portal branding (入口網站品牌)、Support contact email (支援連絡電子郵件) 或 Permissions (許可)。
3. 完成後，請選擇儲存。

在 中新增或移除入口網站管理員 AWS IoT SiteWise

只要幾個步驟，您就可以新增使用者做為入口網站的管理員或移除其入口網站管理員的身分。根據使用者身分驗證服務，選擇下列其中一個選項。

IAM Identity Center

Portal administrators (1)					Remove from portal	Send invitations	Assign administrators
☐	Display name	Type	Email address	Role			
☐	Jane Doe	SSO user	janedoe@example.com	Portal administrator			

新增入口網站管理員

1. 在入口網站詳細資訊頁面的入口網站管理員區段中，選擇指派管理員。
2. 在指派管理員頁面上，選取要以管理員身分新增至入口網站的使用者核取方塊。

Note

如果您使用 IAM Identity Center 做為身分存放區，且已登入管理 AWS Organizations 帳戶，您可以選擇建立使用者以建立 IAM Identity Center 使用者。IAM Identity Center 會傳送電子郵件給新使用者，讓他們設定密碼。然後，您可以將使用者以管理員身分指派給入口網站。如需詳細資訊，請參閱[在 IAM Identity Center 中管理身分](#)。

3. 選擇指派管理員。

AWS IoT SiteWise > Monitor > Portals > example-factory-1 > Assign administrators

Assign administrators

Choose the users that you want to be portal administrators. Portal administrators can grant users access to specific industrial equipment data. [Learn more](#)

Users (2)

Create user

< 1 > ⚙

☐	Display name	Email
☐	Jane Doe	janedoe@example.com
☒	John Doe	johndoe@example.com

Selected users (1)

Cancel Assign administrators

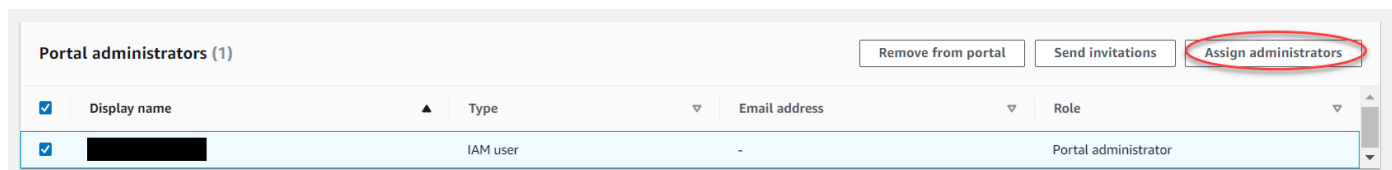
移除入口網站管理員

- 在入口網站詳細資訊頁面的 Portal administrators (入口網站管理員) 區段中，選取要移除的使用者的核取方塊，然後選擇 Remove from portal (從入口網站移除)。

Note

我們建議您至少選取一個入口網站管理員。

IAM



新增入口網站管理員

1. 在入口網站詳細資訊頁面的入口網站管理員區段中，選擇指派管理員。
2. 在指派管理員頁面上，執行下列動作：
 - 如果您想要將 IAM 使用者新增為入口網站管理員，請選擇 IAM 使用者。
 - 如果您想要將 IAM 角色新增為入口網站管理員，請選擇 IAM 角色。
3. 選取您要做為入口網站管理員的使用者或角色的核取方塊。這會將使用者或角色新增至入口網站管理員清單。
4. 選擇指派管理員。

Important

使用者或角色必須具有登入入口網站的 `iotsitewise:DescribePortal` 許可。

AWS IoT SiteWise > Monitor > Portals > example-factory-2 > Assign administrators

Assign administrators

Choose the users that you want to be portal administrators. Portal administrators can grant users access to specific industrial equipment data. [Learn more](#)

ⓘ IAM users or roles must have the `iotsitewise:DescribePortal` permission to sign in to the portal.

Users Roles

IAM users (1) [Manage users in IAM console](#)

Find user name

☒	Name	Date created
☒	raspberrypi-testing	11-08-2019

► **Portal administrators (1)** [Remove](#)

Cancel **Assign administrators**

AWS IoT SiteWise > Monitor > Portals > example-factory-2 > Assign administrators

Assign administrators

Choose the users that you want to be portal administrators. Portal administrators can grant users access to specific industrial equipment data. [Learn more](#)

ⓘ IAM users or roles must have the `iotsitewise:DescribePortal` permission to sign in to the portal.

Users **Roles**

IAM roles (66) [Manage roles in IAM console](#)

Find role name

☒	Name	Date created
☐	[REDACTED]	
☒	AWSIoTSiteWiseMonitorServiceRole_4wZigNpA1	03-16-2021
☐	AWSIoTSiteWiseMonitorServiceRole_ECKT-2Oar	03-11-2021
☐	AWSIoTSiteWiseMonitorServiceRole_GTnd004Wr	03-16-2021
☐	AWSIoTSiteWiseMonitorServiceRole_rHINLNC5-	03-11-2021
☐	AWSIoTSiteWiseMonitorServiceRole_XB330QUIO	03-10-2021
☐	[REDACTED]	
☐	[REDACTED]	
☐	[REDACTED]	

► **Portal administrators (2)** [Remove](#)

Cancel **Assign administrators**

移除入口網站管理員

- 在入口網站詳細資訊頁面的 Portal administrators (入口網站管理員) 區段中，選取要移除的使用者的核取方塊，然後選擇 Remove from portal (從入口網站移除)。



Note

不建議在沒有入口網站管理員的情況下離開入口網站。

傳送電子郵件邀請給入口網站管理員

您可以傳送電子郵件邀請給入口網站管理員。

1. 在入口網站詳細資訊頁面的 Portal administrators (入口網站管理員) 區段中，選取入口網站管理員的核取方塊。

Portal administrators (1)				Remove from portal	Send invitations	Assign users
☒	Display name	▲	Email address	▼	Role	▼
☒	John Doe		john.doe@example.com		Portal administrator	

2. 選擇 Send invitations (傳送邀請)。您的電子郵件用戶端隨即開啟，並在訊息內文中填入邀請。

您可以先自訂電子郵件，然後再將其傳送給入口網站管理員。

在 中新增或移除入口網站使用者 AWS IoT SiteWise

您可以選擇要允許哪些使用者存取您的入口網站。入口網站使用者會出現在 SiteWise Monitor 入口網站內的使用者清單中。入口網站管理員可以從此清單中新增專案擁有者，而專案擁有者可以新增專案檢視者。

Note

如果入口網站管理員和入口網站使用者需要請您新增或移除使用者，則可能會透過入口網站的支援電子郵件聯絡您。

根據使用者身分驗證服務，選擇下列其中一個選項。

IAM Identity Center

Portal users (1)

Remove from portal

Assign users

☐	Display name	Type	Email address	Role
☐	John Doe	SSO user	johndoe@example.com	Portal viewer

新增入口網站使用者

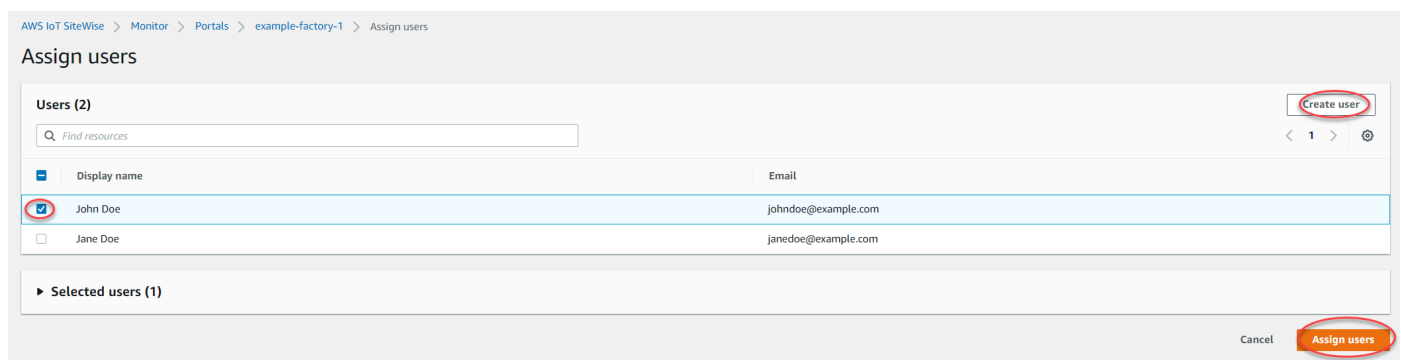
1. 在入口網站詳細資訊頁面的 Portal users (入口網站使用者) 區段中，選擇 Assign users (指派使用者)。

2. 在指派使用者頁面上，選取要新增至入口網站之使用者的核取方塊。

Note

如果您使用 IAM Identity Center 做為身分存放區，且已登入管理 AWS Organizations 帳戶，您可以選擇建立使用者來建立 IAM Identity Center 使用者。IAM Identity Center 會傳送電子郵件給新使用者，讓他們設定密碼。然後，您可以將使用者以使用者身分指派給入口網站。如需詳細資訊，請參閱[在 IAM Identity Center 中管理身分](#)。

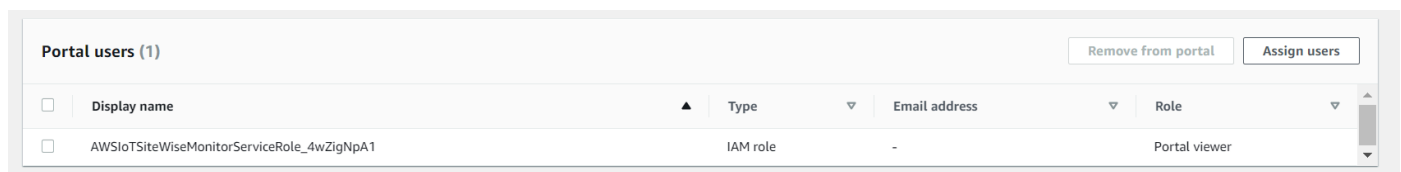
3. 選擇 Assign users (指派使用者)。



移除入口網站使用者

- 在入口網站詳細資訊頁面的入口網站使用者區段中，選取要從入口網站移除之使用者的核取方塊，然後選擇從入口網站移除。

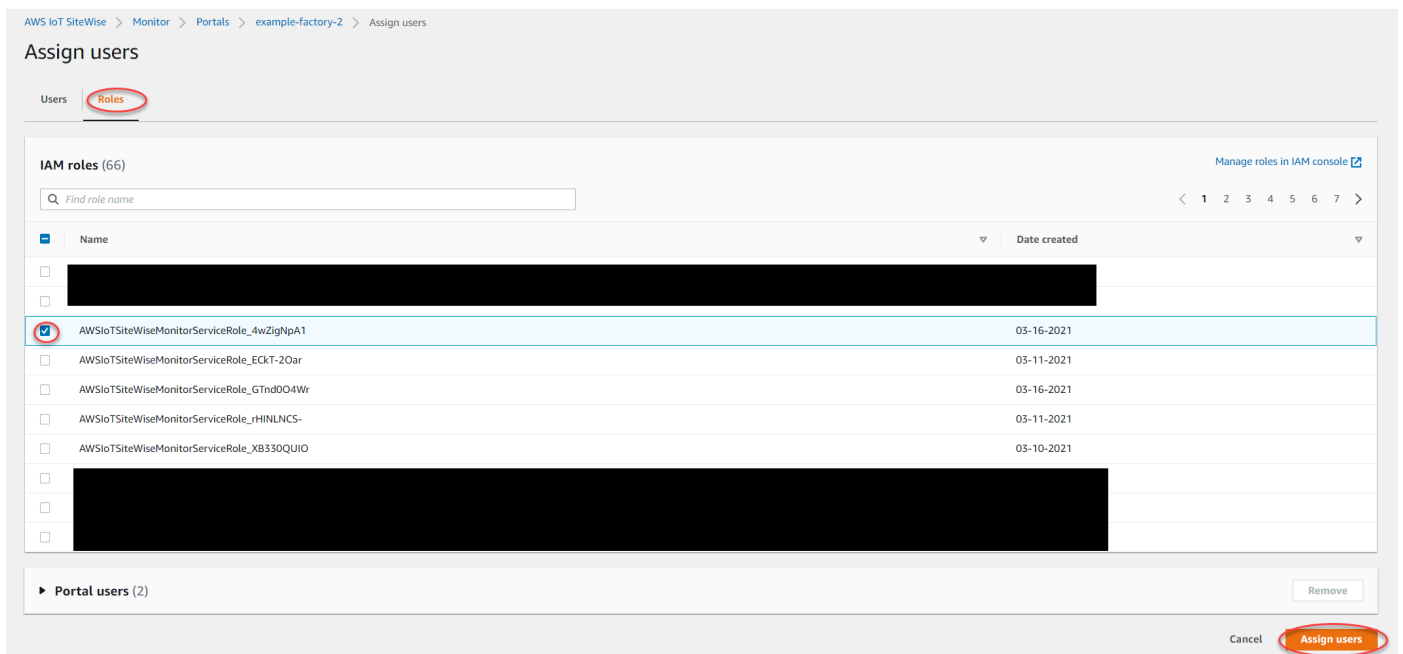
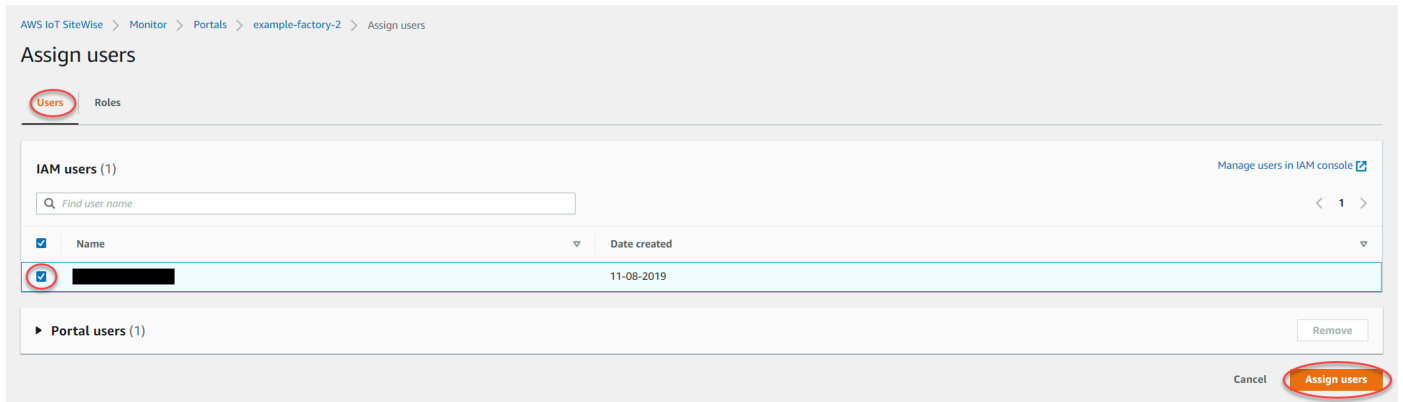
IAM



新增入口網站使用者

1. 在入口網站詳細資訊頁面的 Portal users (入口網站使用者) 區段中，選擇 Assign users (指派使用者)。
2. 在指派使用者頁面上，執行下列動作：
 - 選擇 IAM 使用者，將 IAM 使用者新增為您的入口網站使用者。

- 選擇 IAM 角色，將 IAM 角色新增為您的入口網站使用者。
3. 選取您要新增為入口網站使用者之使用者或角色的核取方塊。這會將使用者或角色新增至入口網站使用者清單。
 4. 選擇 Assign users (指派使用者)。



移除入口網站使用者

- 在入口網站詳細資訊頁面的入口網站使用者區段中，選取要從入口網站移除之使用者的核取方塊，然後選擇從入口網站移除。

⚠ Important

使用者或角色必須具有登入入口網站的*iotsitewise:DescribePortal*許可。

在 中刪除入口網站 AWS IoT SiteWise

如果您為測試目的而建立入口網站，或重複建立了已存在的入口網站，您可能會刪除入口網站。

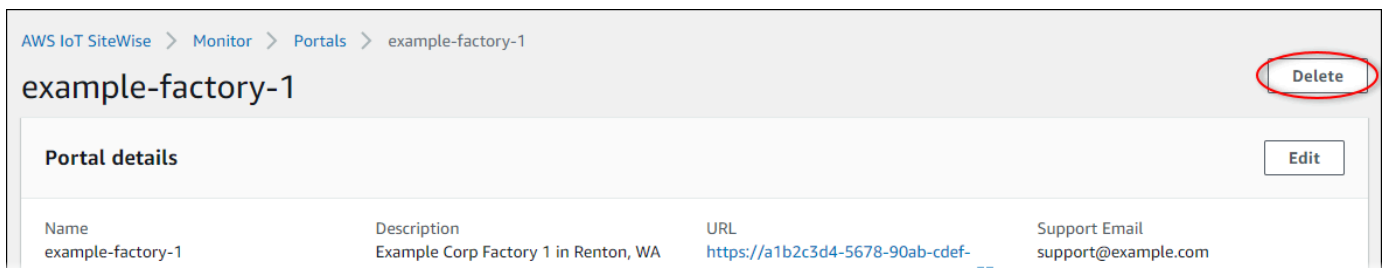
i Note

您必須先在入口網站中手動刪除所有儀表板和專案，才能刪除入口網站。如需詳細資訊，請參閱 SiteWise Monitor 應用程式指南中的[刪除專案](#)和[刪除儀表板](#)。

1. 在入口網站詳細資訊頁面上，選擇 Delete (刪除)。

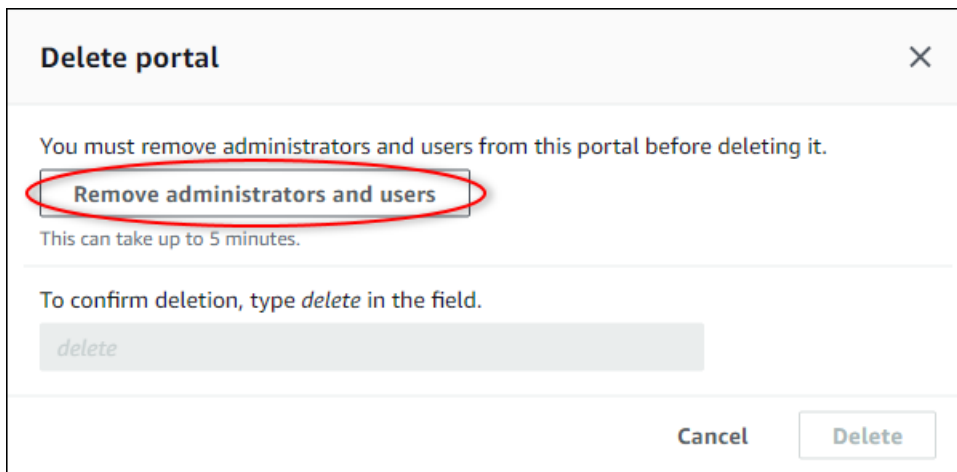
⚠ Important

當您刪除入口網站時，會遺失入口網站包含的所有專案，以及每個專案中的所有儀表板。這個動作無法復原。您的資產資料不會受到影響。

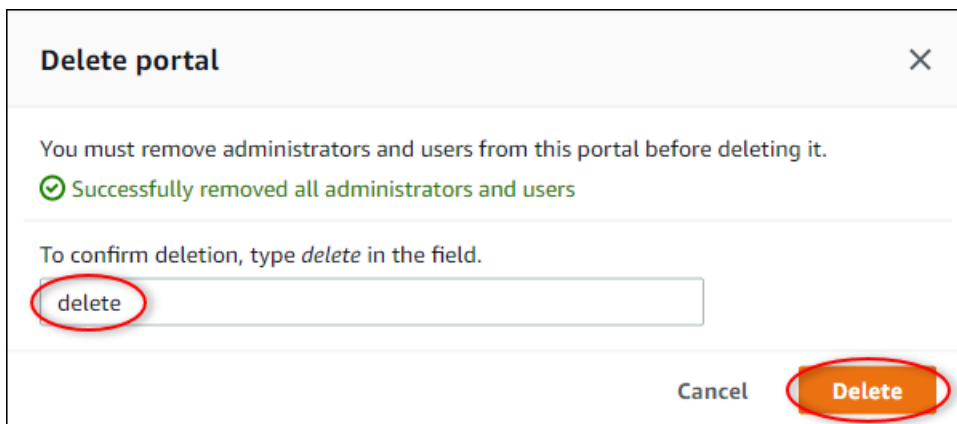


2. 在刪除入口網站對話方塊中，選擇移除管理員和使用者。

您必須先從入口網站移除管理員和使用者，才能刪除該入口網站。如果您的入口網站沒有管理員或使用者，按鈕就不會出現，而且您可以跳至下一個步驟。



3. 如果您確定要刪除整個入口網站，請輸入在欄位中 **delete** 以確認刪除。



4. 選擇 Delete (刪除)。

開始使用 AWS IoT SiteWise Monitor (AI-aware) - 預覽版

身為組織的 AWS 管理員，您可以從 AWS IoT SiteWise 主控台建立入口網站，讓您的組織成員能夠檢視 AWS IoT SiteWise 您的資料。請完成下列步驟以開始使用。

1. 設定和建立入口網站。
2. 新增入口網站管理員並傳送邀請電子郵件。
3. 新增入口網站使用者。

建立入口網站後，入口網站管理員可以建立專案並將使用者新增至專案。專案成員接著會建立儀表板，以視覺化連線的資料 AWS IoT SiteWise，讓他們能夠監控連線裝置、程序和設備的效能。

 Note

將使用者或管理員新增至入口網站時，請避免建立限制使用者許可的 AWS Identity and Access Management (IAM) 政策，例如有限的 IP。任何具有受限許可的附加政策將無法連線至 AWS IoT SiteWise 入口網站。

建立專案以與您的團隊共用。專案擁有者接著可以建立儀表板，以視覺化資產的屬性，協助專案檢視者了解裝置、程序和設備的執行方式。它也為您的團隊提供一致的操作檢視。

儀表板有助於視覺化和了解您的專案資料。它可協助企業和應用程式使用者追蹤其 AWS IoT 裝置和資料。選擇最適合您需求的視覺化類型。重新排列和調整視覺效果大小，以建立適合您團隊的配置。探索您的裝置、程序和設備資產和資料，並快速識別問題並提高營運效率。

主題

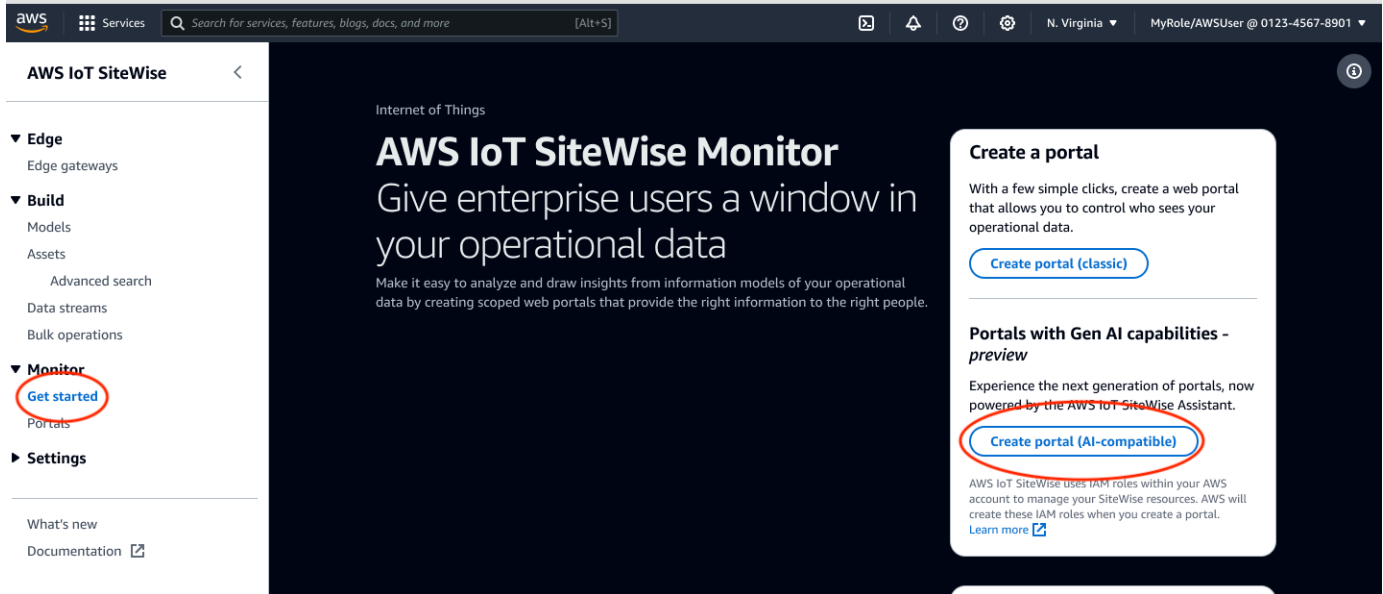
- [建立入口網站](#)
- [設定您的入口網站](#)
- [管理您的入口網站](#)
- [刪除入口網站](#)
- [使用 建立儀表板 AWS CLI](#)
- [入口網站登入](#)
- [建立專案](#)
- [更新專案](#)
- [刪除專案](#)
- [建立儀表板](#)
- [更新儀表板](#)
- [刪除儀表板](#)
- [設定儀表板](#)

建立入口網站

您可以在 AWS IoT SiteWise 主控台中建立 SiteWise Monitor 入口網站。

建立入口網站

1. 登入 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇監控、入門。
3. 選擇建立入口網站 (AI 感知)。



接下來，您必須提供一些基本資訊來設定入口網站。

設定您的入口網站

使用者會使用入口網站檢視您的資料。您可以自訂入口網站的名稱、描述、品牌、使用者身分驗證、支援聯絡人電子郵件和許可。

設定入口網站的步驟：

1. 輸入入口網站的名稱。
2. (選擇性) 輸入入口網站的描述。如果您有多個入口網站，請使用有意義的描述來協助您追蹤每個入口網站包含的內容。
3. (選用) 上傳影像以在入口網站中顯示您的品牌。選擇正方形 PNG 影像。如果您上傳非正方形影像，入口網站會將影像縮小為正方形。
4. 在支援聯絡電子郵件方塊中輸入電子郵件地址，了解支援問題。
5. 在使用者身分驗證方塊中，選擇下列選項：

- 如果您的入口網站使用者使用其公司使用者名稱和密碼登入此入口網站，請選擇 IAM Identity Center。

如果您尚未在帳戶中啟用 IAM Identity Center，請執行下列動作：

- a. 選擇 Create user (建立使用者)。
- b. 在建立使用者頁面上，若要建立第一個入口網站，請輸入使用者的電子郵件地址、名字和姓氏，然後選擇建立使用者。

 Note

IAM 憑證的支援即將推出。

6. 在服務存取區段中選擇下列其中一個選項：

- 選擇建立並使用新的服務角色。根據預設，SiteWise Monitor 會自動為每個入口網站建立服務角色。此角色可讓您的入口網站使用者存取您的 AWS IoT SiteWise 資源。如需詳細資訊，請參閱[使用的服務角色 AWS IoT SiteWise Monitor](#)。
- 選擇使用現有的服務角色，然後選擇目標角色。

7. 選擇以啟用此入口網站的 AWS IoT SiteWise 助理。AWS IoT SiteWise 助理提供快速的資料分析、即時洞見和引導式建議。

 Note

啟用 AWS IoT SiteWise 助理會產生費用。若要使用企業層級知識解決方案和指導，您必須擁有與 Amazon Kendra 索引相關聯的資料集。

8. (選用) 為您的入口網站新增標籤。如需詳細資訊，請參閱[標記您的 AWS IoT SiteWise 資源](#)。

9. 選擇建立入口網站。AWS IoT SiteWise 將建立您的入口網站。

 Note

如果您關閉主控台，您可以透過新增管理員和使用者來完成安裝程序。如需詳細資訊，請參閱[新增或移除入口網站管理員](#)。如果您不想保留此入口網站，請將其刪除，使其不使用資源。如需詳細資訊，請參閱[刪除入口網站](#)。

建立入口網站時，會出現一則訊息。

Successfully created portal "example portal".

[AWS IoT SiteWise](#) > [Monitor](#) > [Portals](#) > example portal

example portal Edit Delete Open portal

Portal details

Name example portal	Status Active	URL https://p-jfnlf2d8.gamma.iotsitewise.aws
Type AI-compatible	AWS IoT SiteWise Assistant Disabled	Portal branding -
Description -	Last updated November 1, 2024, 15:37 (UTC-07:00)	Support contact email myemail@mycompany.com
ID f5fc93a1-011c-4c5a-81a6-e001b50d2547	Date created November 1, 2024, 15:37 (UTC-07:00)	

建立入口網站後，它會列在入口網站區段中。入口網站詳細資訊區段會列出每個入口網站的名稱、描述、ID、URL、狀態、上次更新和建立的日期、入口網站品牌和支援電子郵件。

狀態欄可以是下列其中一個值。

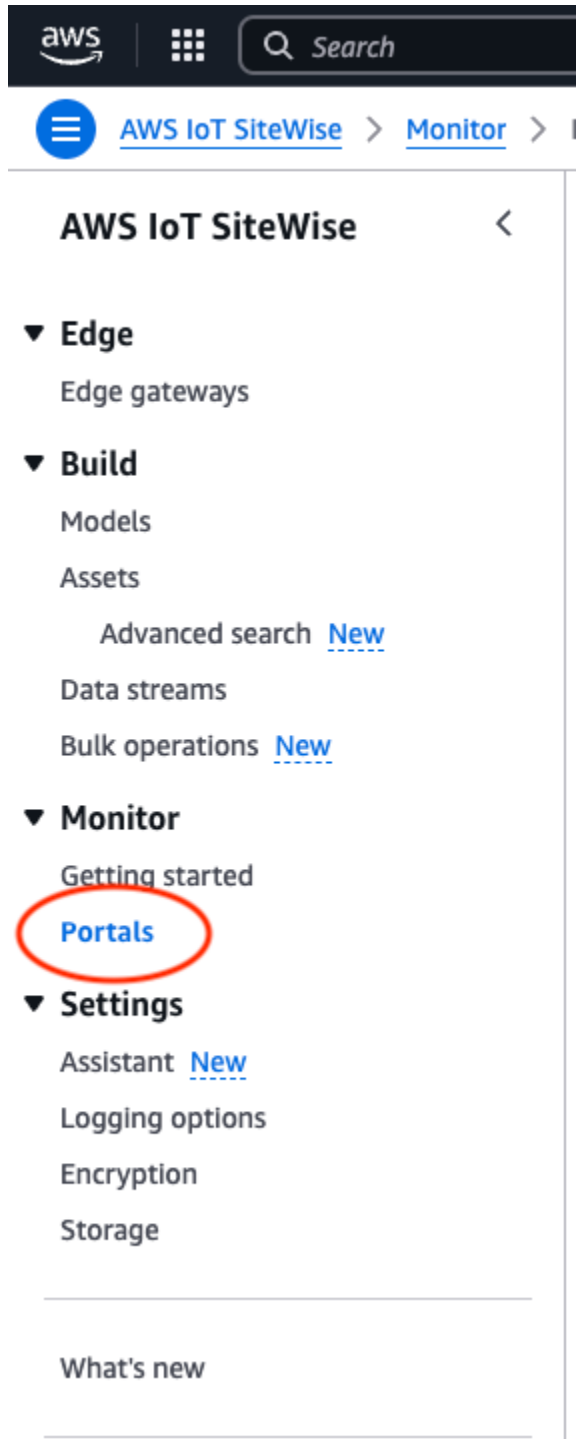
- 建立 - AWS IoT SiteWise 正在處理您的建立入口網站的請求。此程序可能需要幾分鐘的時間才能完成。
- 正在 AWS IoT SiteWise 處理您更新入口網站的請求。此程序可能需要幾分鐘的時間才能完成。
- 待定 - AWS IoT SiteWise 正在等待 DNS 記錄傳播完成。此程序可能需要幾分鐘的時間才能完成。您可以在狀態為待定时刪除入口網站。
- 刪除 - AWS IoT SiteWise 正在處理您刪除入口網站的請求。此程序可能需要幾分鐘的時間才能完成。
- ACTIVE - 當入口網站變成作用中時，您的入口網站使用者可以存取它。
- 失敗 - AWS IoT SiteWise 無法處理您的建立、更新或刪除入口網站的請求。如果您啟用 AWS IoT SiteWise 將日誌傳送至 Amazon CloudWatch Logs，您可以使用這些日誌來疑難排解問題。如需詳細資訊，請參閱[AWS IoT SiteWise 使用 CloudWatch Logs 進行監控](#)。

管理您的入口網站

您可以管理和設定入口網站的各個層面。這包括新增和移除管理員、設定許可和角色、自訂名稱、描述、設定支援電子郵件，以及邀請入口網站管理員。

1. 登入 [AWS IoT SiteWise 主控台](#)。

2. 在導覽窗格中，選擇 Monitor (監視器)、Portals (入口網站)。



3. 選擇入口網站，然後選擇開啟入口網站（或選擇入口網站的名稱）。
4. 您可以執行下列任一管理任務：
 - [編輯入口網站屬性](#)
 - [新增或移除入口網站管理員](#)

- [傳送電子郵件邀請給入口網站管理員](#)
- [在 中刪除入口網站 AWS IoT SiteWise](#)

編輯入口網站屬性

您可以變更入口網站的名稱、描述、品牌、支援電子郵件和服務存取。

1. 在入口網站詳細資訊頁面上 Portal details (入口網站詳細資訊) 區段中，選擇 Edit (編輯)。

☑ Successfully created portal "example portal".

[AWS IoT SiteWise](#) > [Monitor](#) > [Portals](#) > example portal

example portal Edit Delete Open portal

Portal details		
Name example portal	Status Active	URL https://p-jfnlf2d8.gamma.iotsitewise.aws
Type AI-compatible	AWS IoT SiteWise Assistant Disabled	Portal branding -
Description -	Last updated November 1, 2024, 15:37 (UTC-07:00)	Support contact email myemail@mycompany.com
ID f5fc93a1-011c-4c5a-81a6-e001b50d2547	Date created November 1, 2024, 15:37 (UTC-07:00)	

2. 更新名稱、描述、入口網站品牌、支援聯絡人電子郵件、AWS IoT SiteWise 助理或服務存取權。
3. 完成時，請選擇儲存變更。

新增或移除入口網站管理員

只要幾個步驟，您就可以新增使用者做為入口網站的管理員或移除其入口網站管理員的身分。根據使用者身分驗證服務，選擇下列其中一個選項。

IAM Identity Center

Administrators (2) [Info](#) Remove from portal Send invitations Assign administrators

☐	Display name	Email address
☐	Jane Doe	janedoe@amazon.com
☐	John Doe	john DOE@amazon.com

新增入口網站管理員

1. 在入口網站詳細資訊頁面的管理員區段中，選擇指派管理員。
2. 在指派管理員頁面上，選取要以管理員身分新增至入口網站的使用者。

Note

如果您使用 IAM Identity Center 做為身分存放區，且已登入管理 AWS Organizations 帳戶，您可以選擇建立使用者以建立 IAM Identity Center 使用者。IAM Identity Center 會傳送電子郵件給新使用者，讓他們設定密碼。然後，您可以將使用者以管理員身分指派給入口網站。如需詳細資訊，請參閱[在 IAM Identity Center 中管理身分](#)。

3. 選擇指派管理員。

Assign administrators [Info](#)

Choose portal administrators from the users list. Portal administrators grant access to specific industrial equipment data. [Learn more](#)

Users (11)Create user

2 matches

☐	Name	Email
☒	johndoe@amazon.com	johndoe@amazon.com
☐	janedoe@amazon.com	janedoe@amazon.com

[Cancel](#)[Assign administrators](#)

移除入口網站管理員

- 在入口網站詳細資訊頁面的 Portal administrators (入口網站管理員) 區段中，選取要移除的使用者的核取方塊，然後選擇 Remove from portal (從入口網站移除)。

Note

Administrators(#) 列出入口網站的管理員數目。您可以新增多個入口網站管理員來管理和處理專案。

傳送電子郵件邀請給入口網站管理員

您可以傳送電子郵件邀請給入口網站管理員。

1. 在入口網站詳細資訊頁面的管理員區段中，選取入口網站管理員的核取方塊。
2. 選擇 Send invitations (傳送邀請)。您的電子郵件用戶端隨即開啟，並在訊息內文中填入邀請。

您可以先自訂電子郵件，然後再將其傳送給入口網站管理員。

刪除入口網站

如果您為測試目的而建立入口網站，或重複建立了已存在的入口網站，您可能會刪除入口網站。

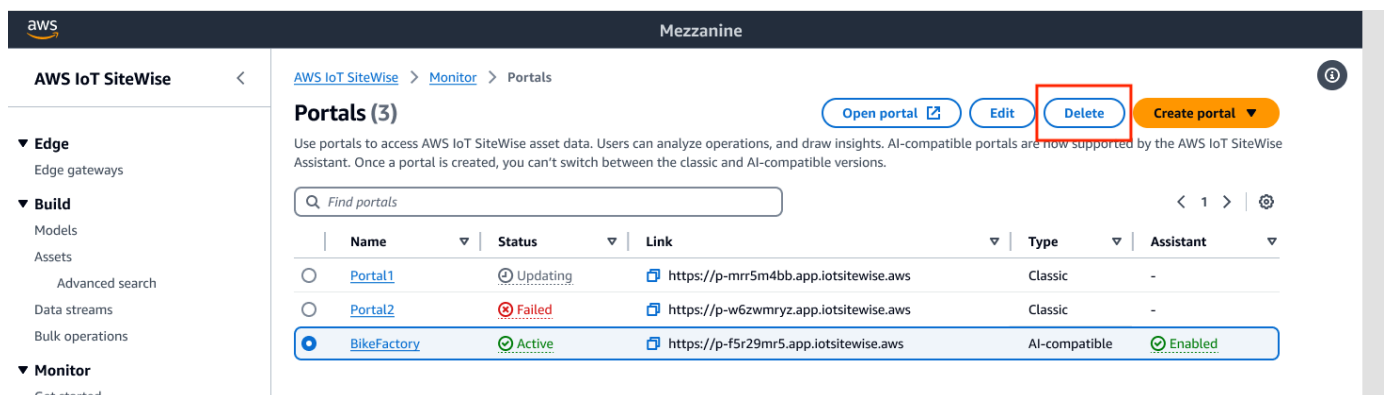
Note

您必須先在入口網站中手動刪除所有儀表板和專案，才能刪除入口網站。

1. 在入口網站詳細資訊頁面上，選擇 Delete (刪除)。

Important

當您刪除入口網站時，會遺失入口網站包含的所有專案，以及每個專案中的所有儀表板。這個動作無法復原。您的資產資料不會受到影響。



The screenshot shows the AWS IoT SiteWise console interface. On the left is a navigation menu with sections like Edge, Build, and Monitor. The main content area is titled 'Portals (3)' and includes a search bar and a table of portals. The 'Delete' button in the top right corner is highlighted with a red box. The table lists three portals: Portal1 (Updating), Portal2 (Failed), and BikeFactory (Active).

Name	Status	Link	Type	Assistant
Portal1	Updating	https://p-mrr5m4bb.app.iotsitewise.aws	Classic	-
Portal2	Failed	https://p-w6zwmryz.app.iotsitewise.aws	Classic	-
BikeFactory	Active	https://p-f5r29mr5.app.iotsitewise.aws	AI-compatible	Enabled

2. 在刪除入口網站對話方塊中，選擇移除管理員和使用者。

您必須先從入口網站移除管理員和使用者，才能刪除該入口網站。如果您的入口網站沒有管理員或使用者，按鈕就不會出現，而且您可以跳至下一個步驟。

3. 如果您確定要刪除整個入口網站，請輸入在欄位中 **confirm** 以確認刪除。
4. 選擇 刪除。

使用 建立儀表板 AWS CLI

當您使用 在儀表板中定義視覺效果（或小工具）時 AWS CLI，您必須在 `dashboardDefinition` JSON 文件中指定下列資訊。此定義是 [CreateDashboard](#) 和 [UpdateDashboard](#) 操作的參數。

`displaySettings`

具有下列參數的顯示設定：

- `numRows` – 儀表板配置中的列數。每一列皆為 `cellSize` 寬。
- `numColumns` – 儀表板配置中的欄數。每個資料欄都是 `cellSize` 寬。
- `cellSize` – （選用）配置中儲存格的大小，以像素為單位。它必須是正數。預設為 10。
- `significantDigits` – （選用）要在儀表板中顯示的簽署數字數目。預設為 4。

`querySettings`

具有下列參數的查詢資訊：

- `refreshRate` – （選用）資料重新整理的速率，以毫秒為單位。接受下列值 - 1000、5000、10000、60000、300000。

`defaultViewport`

如果未提供，則預設為最後五分鐘。包含下列參數：

- `duration` – （選用）決定從目前時間開始查詢資料的過去時間。
- `start` – （選用）其類型為 `Date`。查詢資料的開始時間範圍。需要指定 `end` 日期。
- `end` – （選用）其類型為 `Date`。查詢資料的結束時間範圍。需要指定 `start` 日期。

`widgets`

包含下列資訊的小工具定義結構清單：

`type`

`widget.` 的類型 AWS IoT SiteWise 提供下列小工具類型：

- `xy-plot` – 根據組態的折線圖或散佈圖。
- `bar-chart` – 長條圖。
-

kpi-chart – 關鍵效能指標圖表。

- status-timeline – 狀態小工具，可視覺化並導覽一或多個資料來源的時間序列資料。
- text – 文字小工具。
- table – 資料表小工具。

id

小工具的唯一識別符。

x

小工具的水平位置，從儀表板左側開始。此值是指 Widget 在儀表板格線中的位置。

y

從儀表板頂端開始，小工具的垂直位置。此值是指 Widget 在儀表板格線中的位置。

z

小工具的相對排序。較大的 Z 值小工具會顯示在較低的 Z 值小工具前面，如果重疊的話。

width

小工具的寬度，以儀表板上的儲存格數目表示。

height

小工具的高度，以儀表板上的儲存格數量表示。

properties

小工具的屬性清單。它會因小工具類型而異。如需詳細資訊，請參閱 [IoT 應用程式套件](#)。

Example 範例儀表板定義

下列範例會從存放在 JSON 檔案中的承載來定義儀表板。

```
aws iotsitewise create-dashboard \  
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE \  
  --dashboard-name "Example Dashboard" \  
  --dashboard-definition file://dashboard-definition.json
```

dashboard-definition.json 的下列 JSON 範例，會使用下列視覺效果 Widget 來定義儀表板：

```
{
  "displaySettings": {
    "numColumns": 200,
    "numRows": 1000,
    "cellSize": 20,
    "significantDigits": 4
  },
  "widgets": [{
    "id": "0t73JcxUoc6oEXAMPLE",
    "type": "xy-plot",
    "width": 33,
    "height": 20,
    "x": 0,
    "y": 0,
    "z": 0,
    "properties": {
      "aggregationType": "AVERAGE",
      "queryConfig": {
        "source": "iotsitewise",
        "query": {
          "assets": [{
            "assetId": "97c97abf-e883-47bb-a3f4-EXAMPLE",
            "properties": [{
              "propertyId": "97cc61f4-57a4-4c5f-a82c-EXAMPLE",
              "refId": "692ce941-f3d9-4074-a297-EXAMPLE",
              "aggregationType": "AVERAGE",
              "color": "#7d2105",
              "resolution": "1m"
            }]
          }]
        }
      },
      "properties": [],
      "assetModels": [],
      "alarms": [],
      "alarmModels": []
    }
  ],
  "line": {
    "connectionStyle": "linear",
    "style": "solid"
  },
  "symbol": {
```

```

        "style": "filled-circle"
    },
    "axis": {
        "yVisible": true,
        "xVisible": true
    },
    "legend": {
        "visible": true,
        "position": "right",
        "width": "30%",
        "height": "30%",
        "visibleContent": {
            "unit": true,
            "asset": true,
            "latestValue": true,
            "latestAlarmStateValue": true,
            "maxValue": false,
            "minValue": false
        }
    }
}, {
    "id": "fto7rF40Ny1EXAMPLE-G",
    "type": "bar-chart",
    "width": 33,
    "height": 20,
    "x": 0,
    "y": 20,
    "z": 0,
    "properties": {
        "aggregationType": "AVERAGE",
        "queryConfig": {
            "source": "iotsitewise",
            "query": {
                "assets": [{
                    "assetId": "97c97abf-e883-47bb-a3f4-EXAMPLE",
                    "properties": [{
                        "propertyId": "c84ca8f3-3dea-478a-afec-EXAMPLE",
                        "aggregationType": "AVERAGE",
                        "refId": "2960b958-2034-4d6e-bcc2-EXAMPLE"
                    }]
                }]
            },
            "properties": [],
            "assetModels": [],

```

```

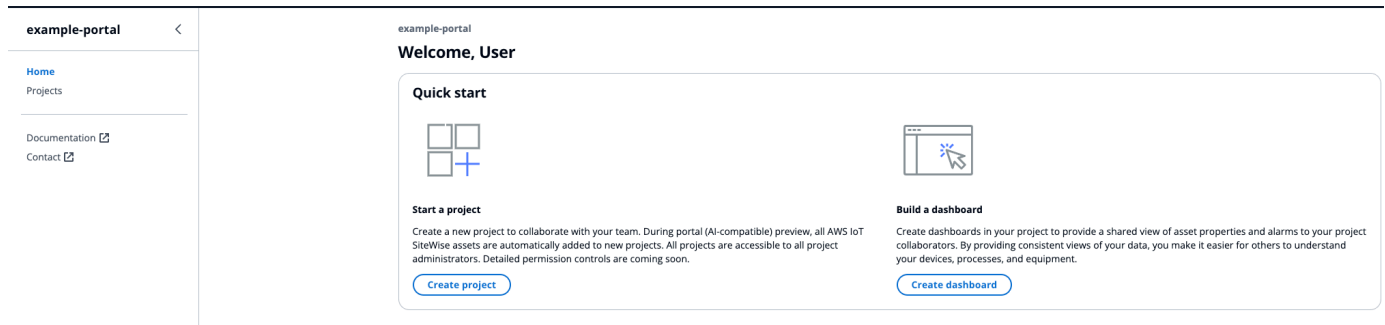
        "alarms": [],
        "alarmModels": [],
        "requestSettings": {
            "aggregation": "AVERAGE"
        }
    },
    "axis": {
        "showX": true,
        "showY": true
    },
    "styleSettings": {
        "2960b958-2034-4d6e-bcc2-360f1f02e505": {
            "color": "#7d2105"
        }
    }
}
}],
"querySettings": {
    "refreshRate": 5000
}
}

```

入口網站登入

使用者登入

1. 在瀏覽器上，輸入應用程式 URL。
2. 輸入您的使用者名稱和密碼，然後按一下登入按鈕。
3. 您現在已登入應用程式。



建立專案

建立專案

1. 建立專案的方式有兩種：
 - a. 在首頁的快速入門下方的歡迎區段中，選擇建立專案。
 - b. 從左側導覽窗格中，選擇專案。選擇右上角的建立以建立專案。
2. 在建立專案區段中，輸入專案名稱，並提供選用的描述。
3. 選擇 Create (建立)。

example-portal

example-portal <

Home
Projects
Documentation
Contact

Home > Projects > Create

Create project

Create a project to host your dashboards.

Project name
Example project
The project name must contain from 1 and 256 characters. Character count: 15/256.

Project description - optional
An example project.
The project description must contain from 1 and 2048 characters. Character count: 19/2048.

Cancel Create

更新專案

編輯專案

1. 選擇專案頁面右上角的更新按鈕，以編輯專案詳細資訊。
2. 透過編輯專案名稱來變更專案的名稱。
3. 透過編輯描述詳細資訊來變更專案的描述。
4. 選取更新以儲存您的變更。

The screenshot shows the 'example-portal' interface. On the left is a sidebar with 'example-portal', 'Home', 'Projects', 'Documentation', and 'Contact'. The main content area has a breadcrumb 'Home > Projects > Example-project'. Below this is the 'Example-project' header with a refresh icon, 'Delete', and a red-bordered 'Update' button. The 'Project details' section shows 'Description: Example project', 'Creation date: 10/25/2024, 11:17:28 AM', and 'Last update date: 10/25/2024, 11:17:28 AM'. The 'Dashboards' section has a search bar 'Filter dashboards', 'Delete', 'Update', and 'Create' buttons, and a table with one row: 'example dashboard' (dashboard, 10/25/2024, 11:17:52 AM, 10/25/2024, 11:17:52 AM).

刪除專案

刪除專案

1. 您只能在刪除專案中的所有儀表板後刪除專案。
2. 選取專案頁面右上角的刪除按鈕。
3. 再次確認您想要刪除專案。
4. 選取刪除以刪除專案。

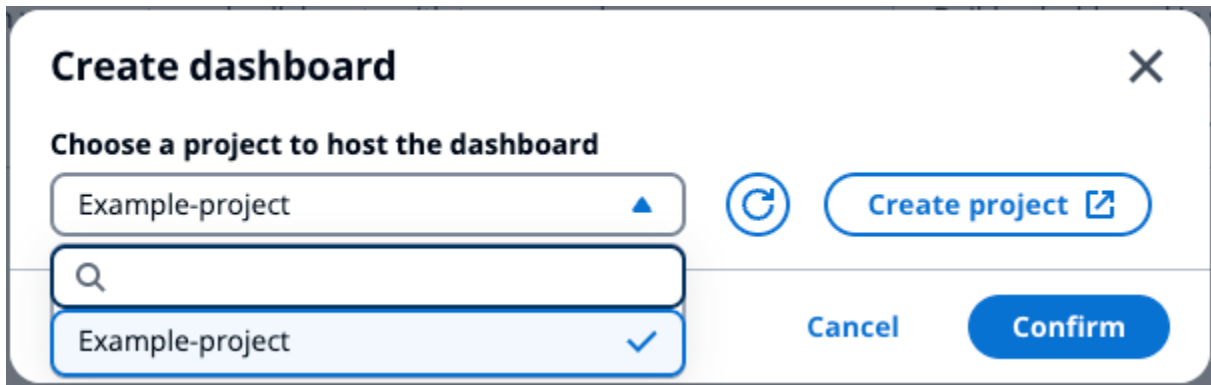
The screenshot shows the 'example-portal' interface after deleting the dashboard. The 'Delete' button in the top right is highlighted with a red box. The 'Dashboards' section now displays 'No dashboards' in a red-bordered box. The 'Project details' section remains the same.

建立儀表板

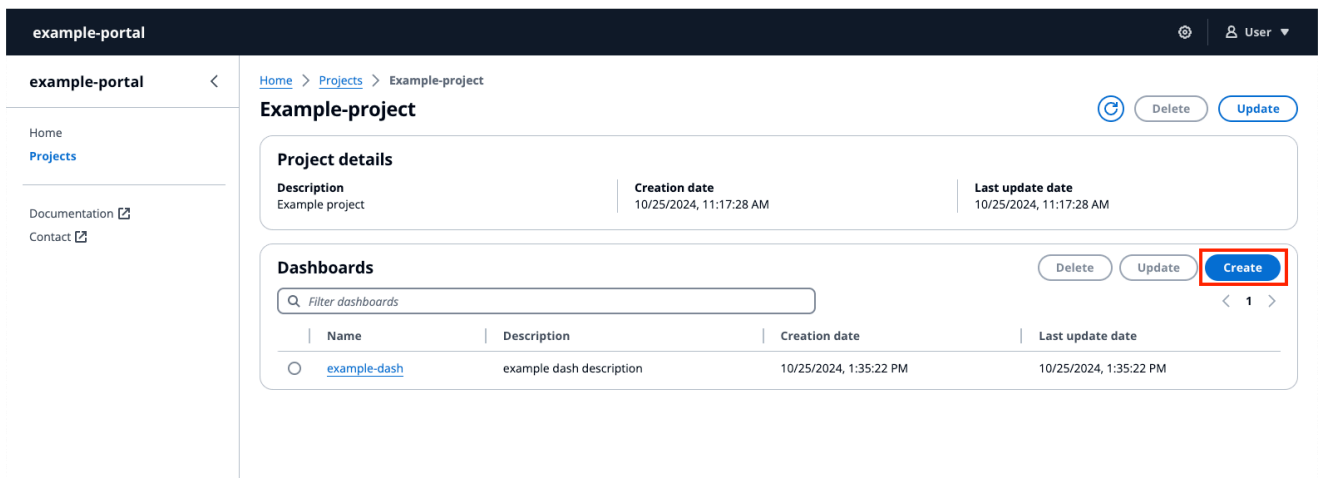
建立儀表板

1. 建立儀表板的方式有兩種：

- a. 在首頁中從建置儀表板建立儀表板。
 - i. 若要在現有專案中建立儀表板，請從選擇要託管儀表板的專案的下拉式選單中選擇專案名稱。
 - ii. 如果您沒有專案，請選擇建立專案，然後選取確認。



- b. 從專案區段中的專案建立儀表板，位於儀表板下。



2. 選取右上角的建立。
3. 輸入儀表板名稱，並提供選用的儀表板描述。
4. 選取建立。

example-portal

example-portal <

Home

Projects

Documentation

Contact

Home > Projects > Example-project > Create dashboard

Create dashboard

Create a dashboard within your project.

Dashboard name

example2-dash

The dashboard name must contain from 1 and 256 characters. Character count: 13/256.

Dashboard description - optional

dash description

The dashboard description must contain from 1 and 2048 characters. Character count: 16/2048.

Cancel Create

5. 設定新建立的儀表板。

更新儀表板

儀表板區段列出專案中的儀表板。從清單中選擇儀表板。

更新儀表板

1. 選取要更新的儀表板。

example-portal

Home > Projects > Example-project

Example-project

Project details

Description: Example project

Creation date: 10/25/2024, 11:17:28 AM

Last update date: 10/25/2024, 11:17:28 AM

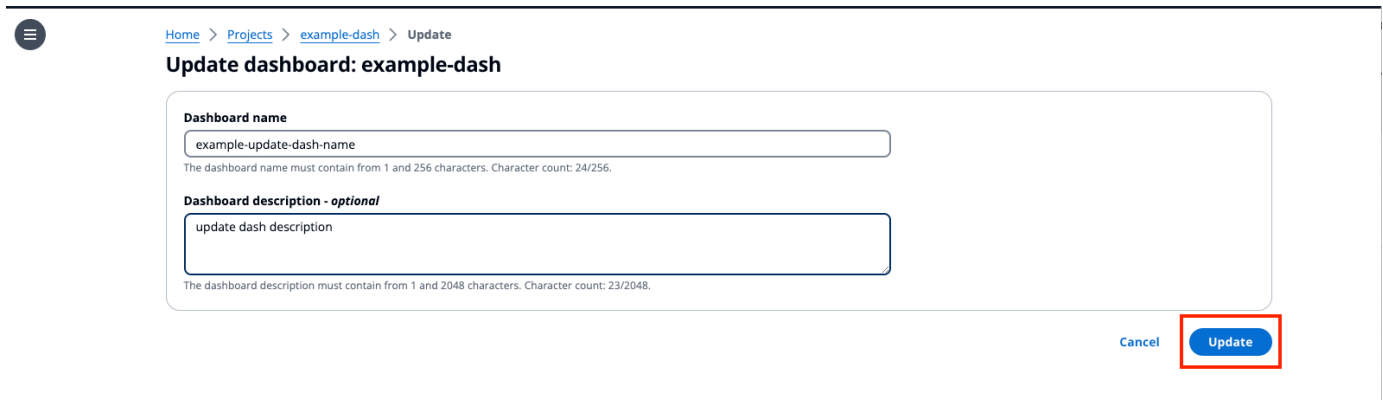
Dashboards

Filter dashboards

Name	Description	Creation date	Last update date
example-dash	example dash description	10/25/2024, 1:35:22 PM	10/25/2024, 1:35:22 PM

Delete Update Create

2. 更新儀表板名稱和選擇性的儀表板描述。選取更新以儲存變更。



Home > Projects > example-dash > Update

Update dashboard: example-dash

Dashboard name

example-update-dash-name

The dashboard name must contain from 1 and 256 characters. Character count: 24/256.

Dashboard description - optional

update dash description

The dashboard description must contain from 1 and 2048 characters. Character count: 23/2048.

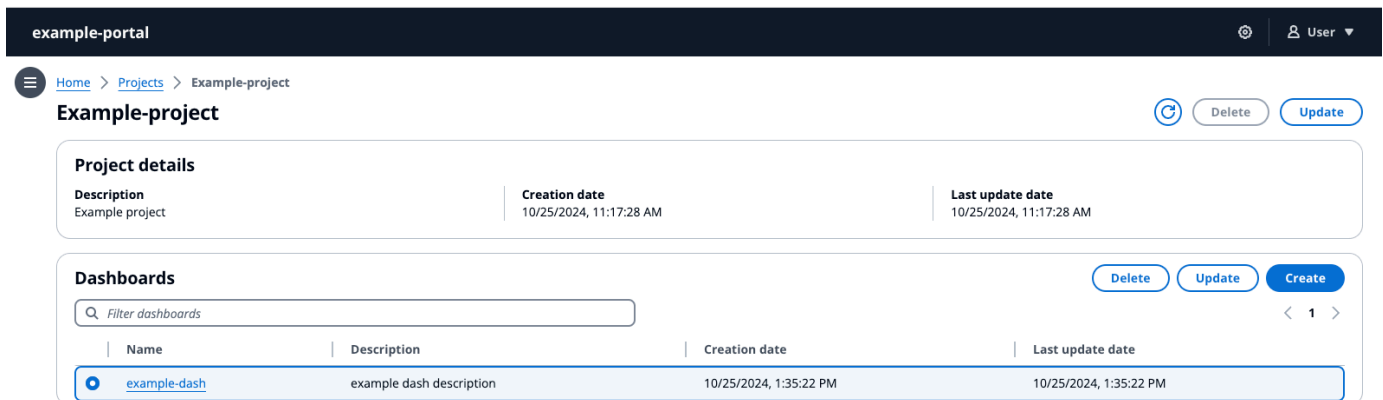
Cancel Update

刪除儀表板

儀表板區段列出專案中的儀表板。從清單中選擇儀表板。

刪除儀表板

1. 選取要刪除的儀表板。



example-portal

Home > Projects > Example-project

Example-project

Delete Update

Project details

Description
Example project

Creation date
10/25/2024, 11:17:28 AM

Last update date
10/25/2024, 11:17:28 AM

Dashboards

Filter dashboards

Name	Description	Creation date	Last update date
example-dash	example dash description	10/25/2024, 1:35:22 PM	10/25/2024, 1:35:22 PM

Delete Update Create

2. 選取刪除以刪除儀表板。這無法復原。

設定儀表板

儀表板區段列出專案中的儀表板。從清單中選擇儀表板。編輯模式可讓您透過新增小工具並進行設定來設定儀表板。預覽按鈕可讓您將變更視覺化。



設定儀表板的步驟：

- 將不同類型的資料小工具拖放到儀表板畫布，以進行資料視覺化。
- 從左側的資源瀏覽器將資料新增至所需的小工具。資源總管包含模型化、非模型化和動態資產區段。依資產名稱或屬性名稱搜尋。選取要新增的屬性，然後選擇新增。
- 透過變更小工具上的組態來微調配置和樣式。設定元件，包括標題、閾值和其他組態詳細資訊。
- 設定顯示資料的時間範圍。
 - 選擇資料的顯示時間範圍。從右上角選擇時間範圍和重新整理率，並個人化範圍。從選單中選擇要重新整理資料的速率。
 - 使用軌跡球滑鼠滾輪或按一下滑鼠右鍵，選取小工具上的時間範圍。這會移動顯示的時間範圍。
- 選擇 Save (儲存)。

主題

- [資源總管](#)
- [小工具](#)
- [設定小工具](#)
- [使用小工具](#)
- [小工具中的警示](#)
- [AWS IoT SiteWise 小工具中的助理使用](#)
- [要詢問 AWS IoT SiteWise 助理的問題範例](#)

資源總管

本節說明模型化、非模型化和動態資產。從這三個中任一個選擇資產，並將其新增至您的小工具，並將其視覺化。

主題

- [已建模](#)
- [未建模](#)
- [動態資產](#)

已建模

本節說明選取和視覺化模型化資產的程序。

資產的選擇

可依下列方式查詢資產：

- 搜尋資產名稱。使用萬用字元 *。例如，Wind*會傳回以文字 開頭的資產名稱Wind。您必須[與整合 AWS IoT TwinMaker](#)，才能使用此功能。
- 預設會列出所有資產。

從列出的資產中，依名稱、描述、ID 或資產模型 ID 進行篩選。選取一個資產以列出其屬性（資料串流）和警示。

資料串流選擇

資料串流列在資料串流功能表下方。篩選 <https://docs.aws.amazon.com/iot-sitewise/latest/APIReference/> 中依[屬性](#)中繼資料列出的資料串流。根據選取的小工具，選取一或多個資料串流。

- KPI 和量測僅支援單一資料串流。
- 其餘小工具支援具有多重選取的多個資料串流。

警示選擇

AWS IoT SiteWise 警示列在警示資料串流功能表下方。篩選依警示中繼資料列出的警示資料串流。名稱、輸入屬性和複合模型 ID 是用於篩選的一些中繼資料。根據選取的小工具，選取一或多個資料串流。

- KPI 和量測器僅支援單一警示。
- 其餘小工具支援具有多選的多個警示。

模型化資產視覺化

1. 將小工具拖曳至畫布。選取每個小工具面板的屬性以建構儀表板。
2. 篩選條件選項會篩選資產，以選擇要視覺化的資產。依文字、屬性或值進行篩選。篩選適用於載入瀏覽器的資產，而非後端篩選。
3. 搜尋以列出要新增至小工具的資產。
4. 將資產新增至畫布中的小工具。
5. 選擇重設以選取其他資產，或修改所選的資產。
6. 儲存儀表板。在預覽模式中，從下拉式選單中選擇不同的資產，以監控每個資產下的屬性，而無需重建資料面板。

Note

右側的組態設定輪會顯示使用者可選擇的偏好設定，例如頁面大小、黏性第一欄、黏性最後一欄和欄偏好設定。自訂您的偏好設定，然後選擇確認以套用變更。

Modeled

Unmodeled

Dynamic assets

Assets (1)

Browse through your asset hierarchy and select an asset to view its associated data streams.

Root

Search

Q

Search for resources

Search

Filter

Q

Filter assets by text, property, or value

< 1 > ⚙

Name

Description

Demo Wind Farm Asset

Asset properties (8)

Select a modeled datastream to add to a selected widget

Filter

Q

Filter asset properties by text, property, or value

< 1 > ⚙

☐	Name	Unit	Data ty...	Latest ...	Latest value time
☐	Total Aver...		DOUBLE	37478.2303	2024-10-01 09:50:00 p.m.
☐	Total Aver...		DOUBLE	6.0000	2022-10-26 03:42:43 p.m.
☐	Total Aver...		DOUBLE	555.0000	2022-10-13 11:59:49 p.m.
☐	Code		INTEGER	300.0000	2022-10-13 10:59:28 p.m.
☐	Reliability ...		STRING	Mary Major	2022-10-13 10:59:28 p.m.
☐	Location		STRING	Renton	2022-10-13 10:59:28 p.m.
☐	Total Over...		DOUBLE	900.0000	2024-10-01 09:50:00 p.m.
☐	recipient a...		STRING	54a88418-...	2022-10-26 03:42:43 p.m.

未建模

本節說明搜尋未建模的資料串流，並將其新增至小工具以視覺化。

非模型資料串流視覺化

1. 將小工具拖曳至畫布。選取每個小工具面板的屬性以建構儀表板。
2. 未建模的資料串流會列在時間序列區段下。它們具有可自訂的屬性。
3. 篩選條件選項會篩選要視覺化的資料串流。篩選適用於載入瀏覽器的資料串流，而非後端篩選。
4. 將資料串流新增至畫布中的小工具。
5. 選擇重設以取消選取資料串流。
6. 儲存儀表板。在預覽模式中，從下拉式選單中選擇不同的資產，以監控每個資產下的屬性，而無需重建資料面板。

Note

右側的組態設定輪會顯示使用者可選擇的偏好設定，例如頁面大小、黏性第一欄、黏性最後一欄和欄偏好設定。自訂您的偏好設定，然後選擇確認以套用變更。

Modeled | **Unmodeled** | Dynamic assets

Time series (1)
Select a unmodeled datastream to add to a selected widget

Filter < 1 > ⚙

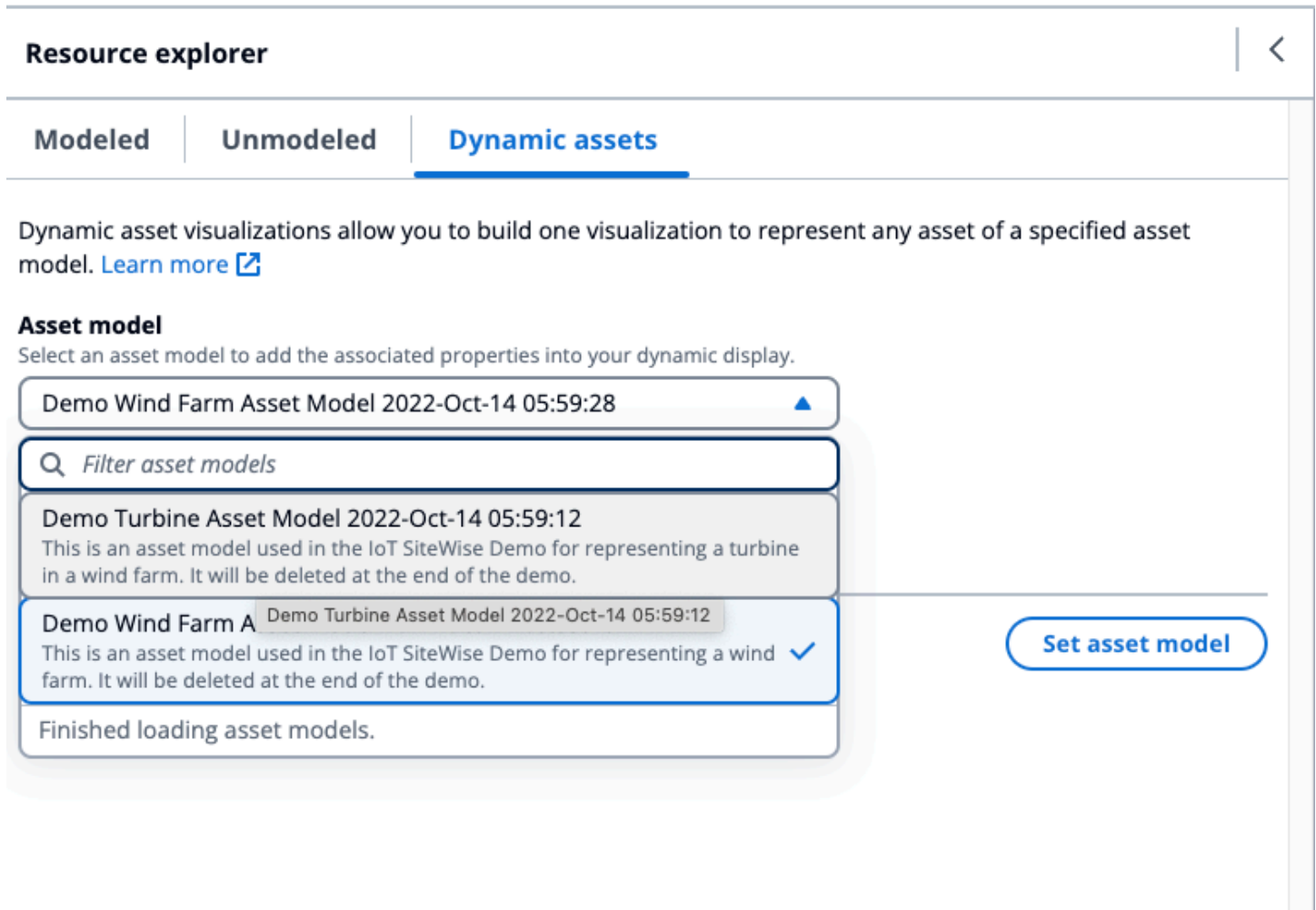
☒	Alias ▾	ID ▾	Data type ▾	Latest val... ▾	Latest valu... ▾
☒	DemoDisass...	33e78bb9-39...	DOUBLE	3.0283	2024-10-01 0...

動態資產

新的 SiteWise Monitor 可讓客戶動態切換所選資產模型的資產。您可以從下拉式選單中選取，以視覺化不同資產的屬性。

動態資產視覺化

1. 選擇資源總管上的動態資產索引標籤。
2. 從下拉式選單中選取資產模型以列出 的資產。
3. 從下拉式選單中選取預設資產。
4. 選擇設定資產模型以選取資產模型。
5. 儲存儀表板。在預覽模式中，從下拉式選單中選擇不同的資產，以監控每個資產下的屬性，而無需重建資料面板。



小工具

Widgets 支援各種功能，包括警示、高效能即時串流，以及與其他 IoT App Kit 元件的順暢同步。儀表板支援下列小工具：

- **Line – Line Widget** 是一種視覺化 Widget，可顯示隨時間變化的趨勢和變化。它由一系列資料點組成，每個點由點或標記表示，由直線區段連接以建立折線圖。它支援各種功能，包括警示、閾值、高效能即時串流，以及與其他 IoT App Kit 元件的順暢同步。此小工具可自訂，以清楚簡潔地傳達複雜的資料。
- **長條圖** – 長條圖是一種強大的視覺化工具，可顯示時間序列資料。它支援各種功能，包括警示、高效能即時串流，以及與其他 IoT App Kit 元件的順暢同步。
- **時間軸** – 時間軸小工具提供視覺化和瀏覽資料來源時間序列資料的方法。在時間軸上顯示資料串流值是唯一的。它支援一組豐富的功能，包括警示、高效能即時串流，以及跨其他 IoT App Kit 元件順暢同步。它最適合用來顯示非數字資料類型/
- **KPI – Key Performance Indicator (KPI)** 元件提供資產屬性概觀的精簡表示。它支援警示和閾值。此概觀提供裝置、設備和程序整體效能的重要洞見。KPI 僅支援單一資料串流或警示，不支援多個資料串流。
- **量測** – 量測元件提供資產屬性概觀的精簡表示。其用於視覺化裝置、設備或程序整體效能的重要洞見。它的功能與 KPI 相同，但視覺效果不同。計量會顯示資料串流值、閾值和值範圍。您可以使用 Gauge AWS IoT 與來自一或多個資料來源的資料互動。
- **資料表** – 資料表元件提供精簡形式，用於檢視一或多個時間序列資料來源中的一或多個資料串流。它以表格形式顯示具有屬性、最新值和單位的資產。支援 AWS IoT SiteWise 警示。
- **文字** – 文字小工具有助於撰寫具有各種顏色和字型的文字。您可以透過將文字與 URL 建立關聯來建立連結。此小工具未啟用屬性和閾值欄位。



設定小工具

將小工具新增至儀表板後，您可以選擇右側面板中的組態圖示來設定小工具。

- **樣式** – 在小工具標題中新增標題。不同的小工具具有不同的組態。以下列出幾個範例。
 - **長條小工具**：
 - **解析度和彙總** – 在此設定解析度和彙總的值。
 - **格式資料** – 將小數位數設定為要顯示的小數位數。
 - **顯示樣式** – 選取要顯示的值。

- 軸 – 選擇以顯示軸。
- 行小工具：
 - 解析度和彙總 – 在此設定解析度和彙總的值。
 - 格式資料 – 將小數位數設定為要顯示的小數位數。
 - Y 軸 – 新增標籤，以及最小值和最大值。
 - Widget 樣式 – 選取線條類型、線條樣式、線條厚度和資料點形狀值。
 - 圖例 – 選擇對齊和顯示。
- 量測小工具：
 - 解析度和彙總 – 在此設定解析度和彙總的值。
 - 格式資料 – 將小數位數設定為要顯示的小數位數。
 - 顯示樣式 – 選取要顯示的值。
 - Y 軸 – 新增標籤，以及最小值和最大值。
 - 字型 – 選取字型大小、單位字型大小和標籤字型大小值。

Configuration for a Line widget

Configuration

Style Properties Thresholds

Widget title
Input title

▼ Resolution and Aggregation

Resolution
Autoselect

Aggregation
Average

▼ Format data

Decimal places
Must be between 0 and 100.

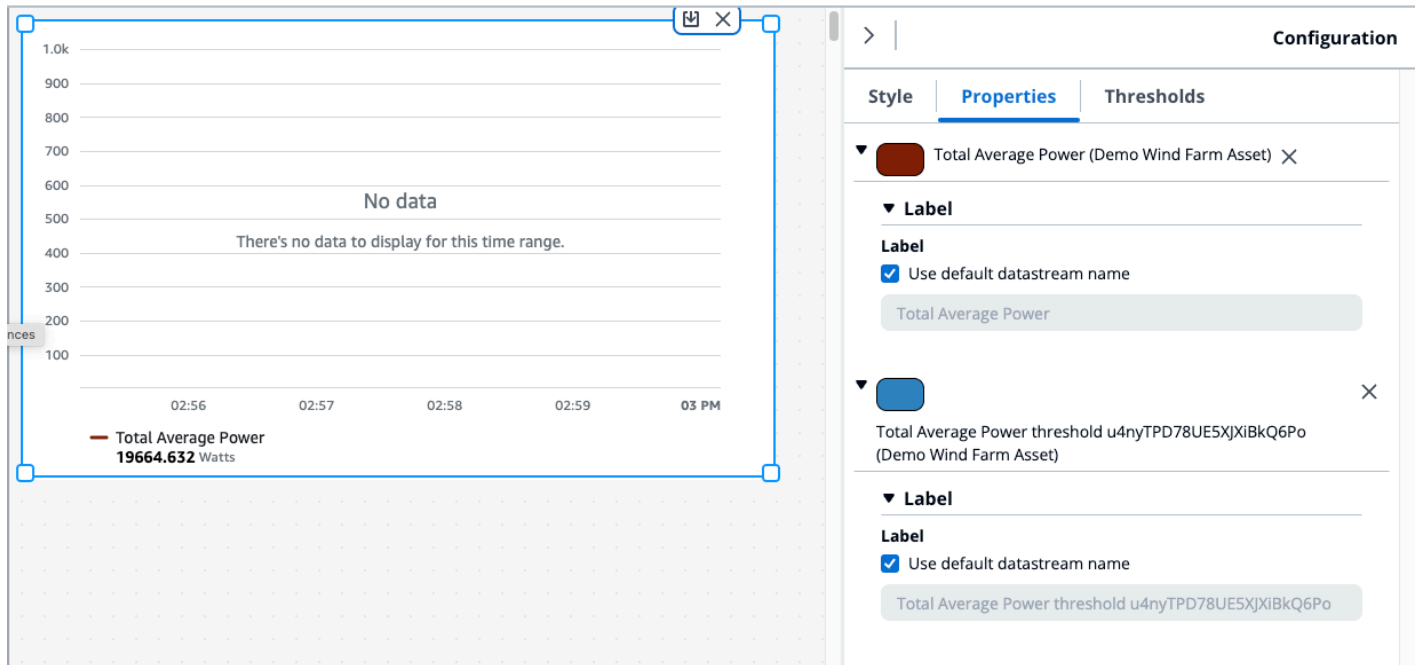
▼ Axis

☒ View X axis ☒ View Y axis

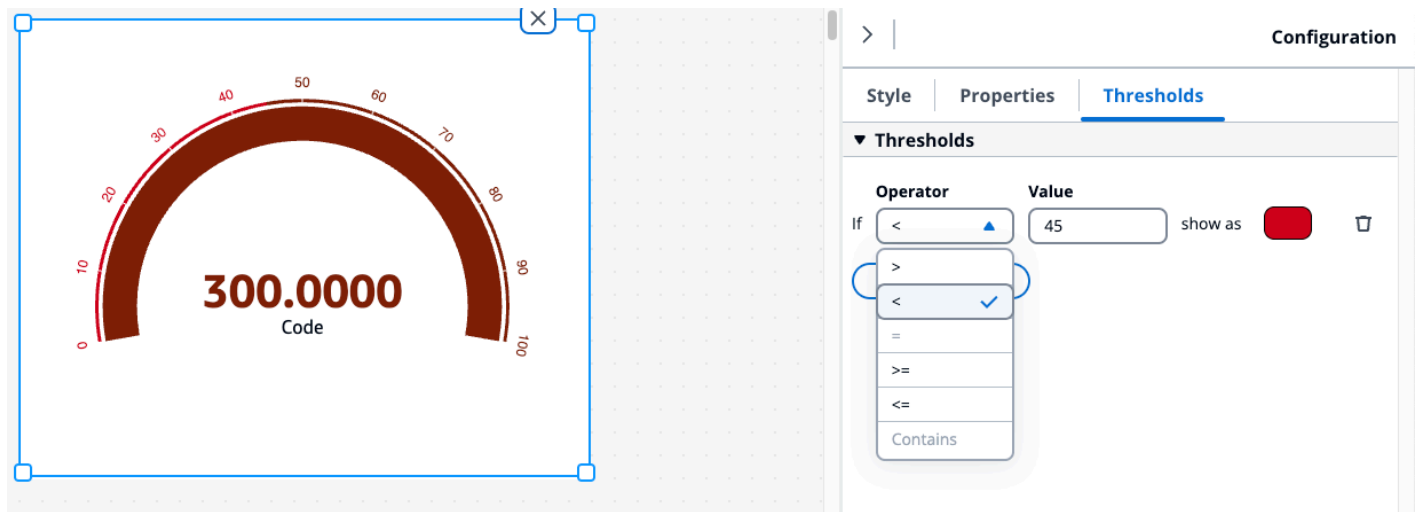
Y axis Label

- 屬性 – 本節列出小工具的所有屬性。不同的小工具具有不同的屬性。以下列出幾個範例。
- 行小工具：
 - 標籤 – 選擇使用預設資料串流名稱或提供新名稱。
 - 樣式 – 將行類型、行樣式設定為要顯示的小數位數。

- Y 軸 – 將值選取為預設樣式、顯示 Y 軸控制項，並設定最小值和最大值。
- 資料表小工具：
 - 標籤 – 選擇使用預設資料串流名稱或提供新名稱。
- 資料表小工具：
 - 標籤 – 選擇使用預設資料串流名稱或提供新名稱。



- 閾值 – 新增小工具的閾值。不同的小工具具有不同的組態。以下列出幾個範例。
- 長條圖小工具：
 - 選擇新增閾值以新增至小工具。
 - 選擇運算子，並為閾值提供值。使用調色盤中的顏色自訂閾值。
 - 您可以選擇將閾值套用至所有資料。
- 行小工具：
 - 選擇新增閾值以新增至小工具。
 - 選擇運算子，並為閾值提供值。使用調色盤中的顏色自訂閾值。
 - 從下拉式選單中選擇如何顯示閾值。
- 量測小工具：
 - 選擇新增閾值以新增至小工具。
 - 選擇運算子，並為閾值提供值。使用調色盤中的顏色自訂閾值。



使用小工具

您可以在儀表板中個別使用小工具，也可以透過多選來使用小工具。

在儀表板中編輯小工具

選擇單一小工具並進行編輯。若要編輯儀表板中的多個小工具，請按住 Shift + 滑鼠左鍵，然後選取儀表板中的所有小工具。選取後，使用者可以在樣式組態設定中新增資料串流，並修改 Widget 標題。儀表板中所有小工具的標題都會變更。

在畫布上按一下滑鼠右鍵，然後執行下列動作：

- 複製 – 將小工具的副本新增至畫布。
- 刪除 – 刪除小工具。
- 帶到前面 – 將選取的小工具帶到畫布前面。
- 傳送至背面 – 將選取的小工具傳送至畫布背面。

調整小工具大小

在儀表板中多選小工具，個別或在群組中調整小工具大小。

若要變更小工具的大小：

- 若要變更單一小工具的大小，請選取小工具，然後拖曳邊角以變更其大小。
- 若要變更多個小工具的大小，請按 Shift + 按一下滑鼠左鍵選取多個小工具，然後在邊角拖曳以變更其大小。

在儀表板中刪除小工具

在儀表板中多選小工具，個別刪除小工具，或在群組中刪除小工具。

若要刪除小工具：

- 若要刪除單一小工具，請選取小工具，然後按一下滑鼠右鍵並選擇刪除。您也可以選取，然後按一下右上角的 X 以刪除小工具。
- 若要刪除多個小工具，請依 Shift + 按一下滑鼠左鍵選取多個小工具，然後按滑鼠右鍵並選擇刪除。

小工具中的警示

當設備或程序以次最佳化方式執行時，警示會提醒您和您的團隊。機器或程序的最佳效能，表示特定指標的值應該在高低限制範圍內。當這些指標超出其操作範圍時，必須通知設備操作員，以便他們可以修正問題。警示可協助您快速識別問題並通知操作員，以最大限度地提高設備和程序的效能。

您可以在資源瀏覽器的模型化索引標籤中找到與資產相關聯的警示。

- 搜尋並選取資產。
- 向下捲動至警示資料串流區段的資料串流資料表，然後展開。
- 在警示資料表中選取警示，然後選擇新增。

主題

- [不同小工具中的警示](#)

不同小工具中的警示

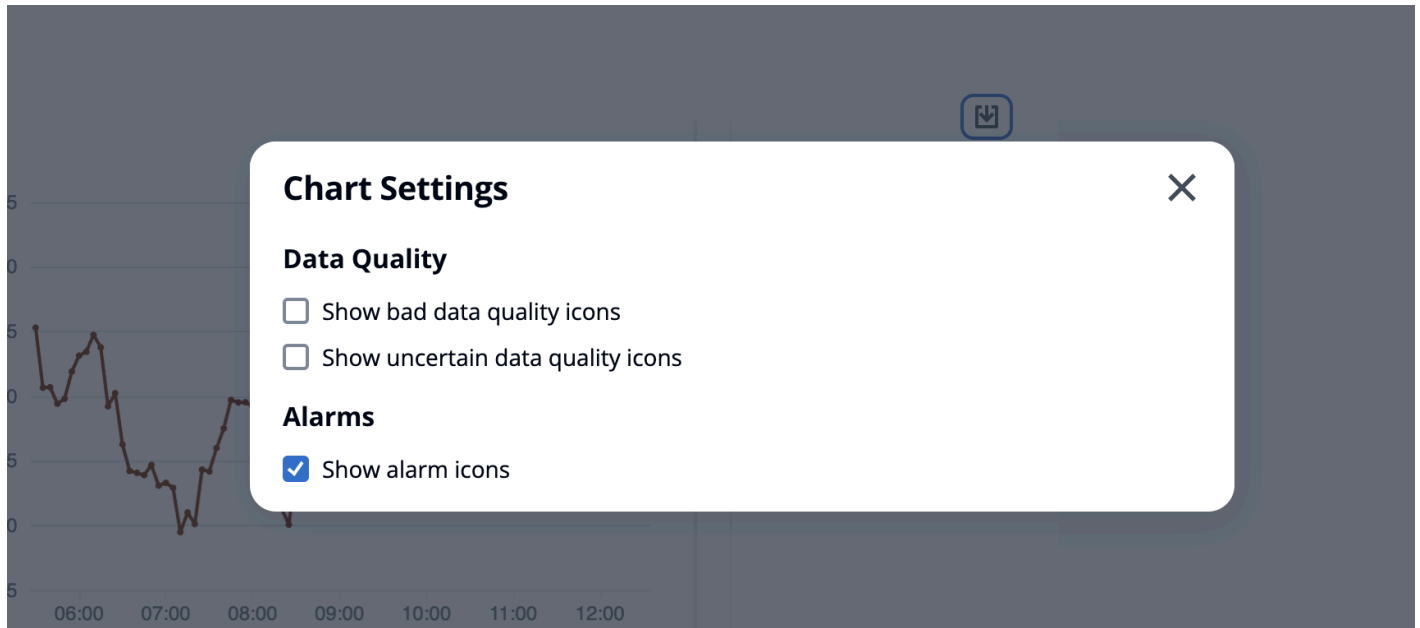
對於所有小工具：

- 資料串流屬性設定取決於要新增至小工具的屬性類型。資料串流屬性具有完整的屬性設定支援，而警示屬性目前不允許屬性設定組態。
- 如果您新增警示資料串流，其相關聯的輸入屬性資料串流也會新增至圖表。如果您移除警示資料串流，則也會移除其輸入屬性。
- 若要個別控制警示的輸入屬性資料串流，您必須分別新增它們。

以下範例說明某些小工具如何使用警示。

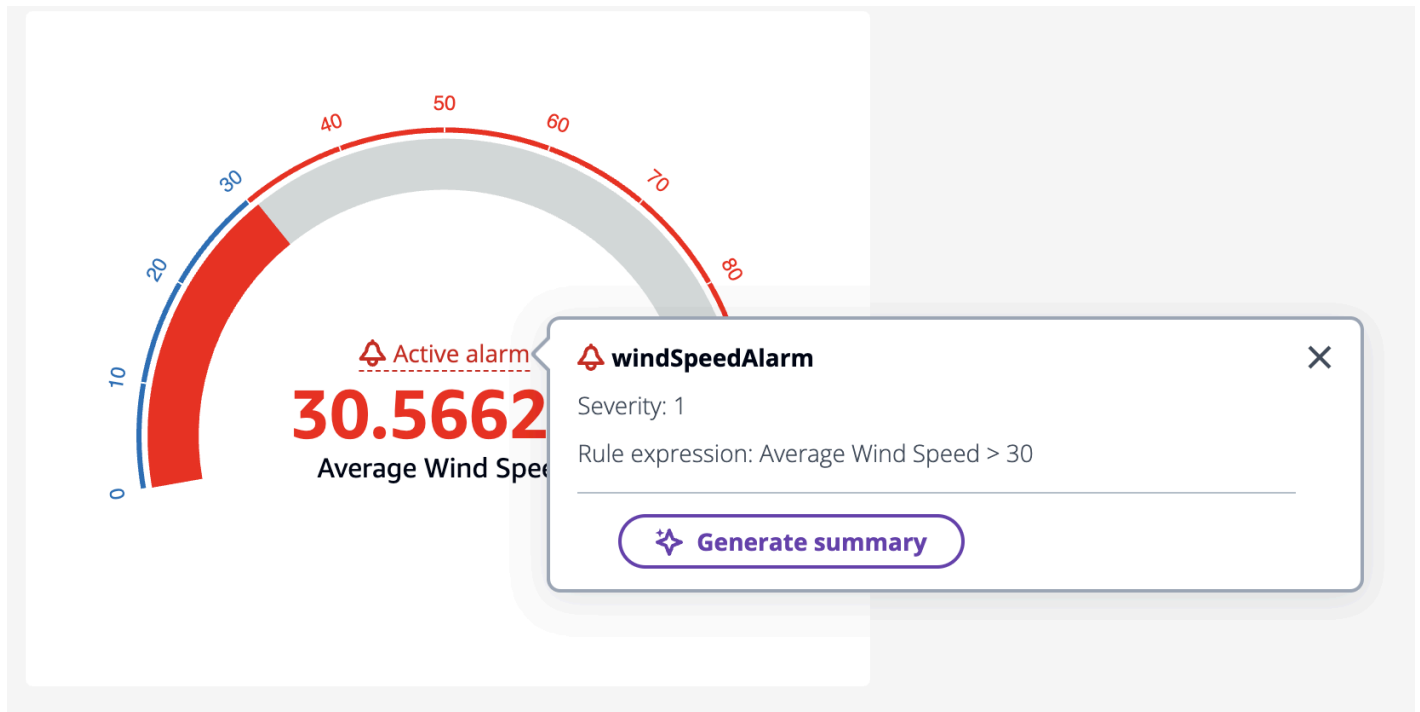
- 折線圖

- 警示及其輸入屬性資料串流會新增至圖表。
- 您可以在圖表圖例中看到警示狀態，並在警示變更狀態時將圖示懸停在資料串流上。
- 您可以從圖表設定中關閉警示圖示。

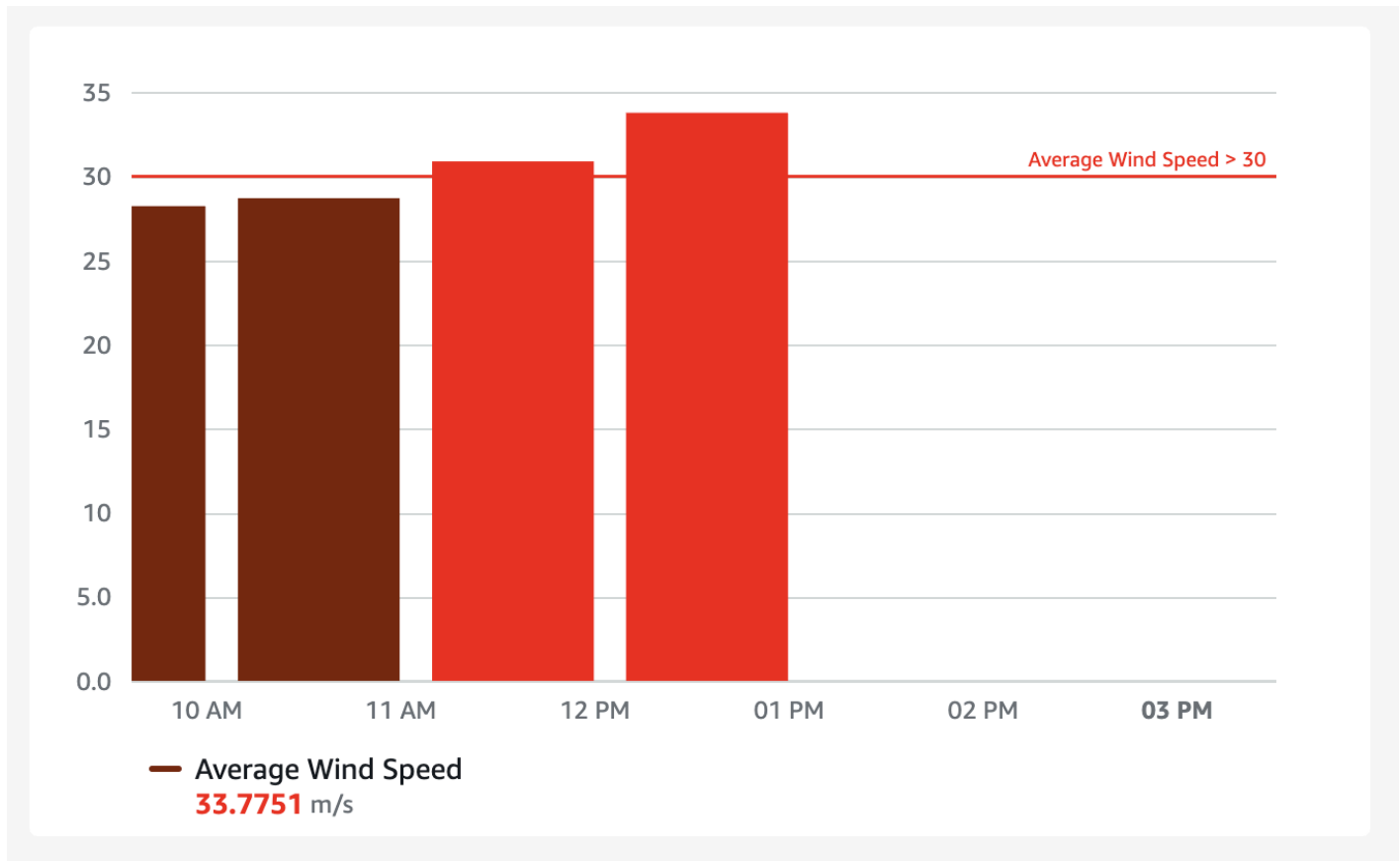


- KPI 和量測

- 警示及其輸入屬性資料串流會新增至所選的小工具。
- 警示閾值會新增至小工具，該小工具會根據其組態變更顏色。
- 您可以在小工具上選取警示狀態，請參閱警示詳細資訊，然後按一下產生摘要來呼叫 AWS IoT SiteWise 以取得警示摘要。



- 資料表
 - 警示及其輸入屬性會新增為資料表上的資料列。
- 長條圖
 - 警示會新增為圖表的閾值，這會變更超出閾值的任何資料串流的顏色。
 - 您可以個別新增任何相關聯的資料串流。
 - 您無法從小工具與 AWS IoT SiteWise 助理互動。

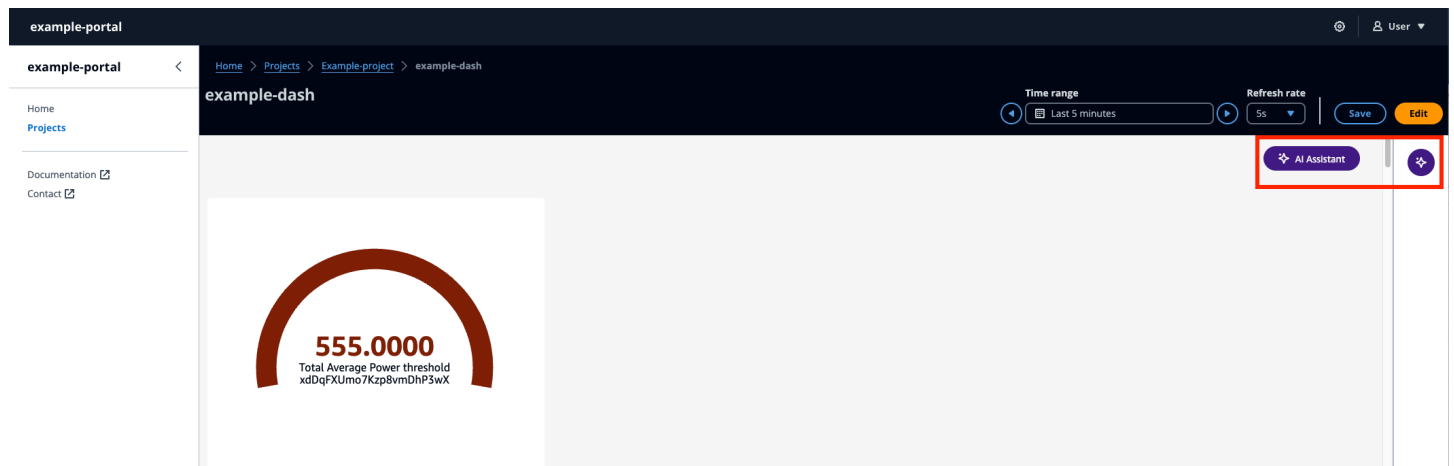


- 狀態時間軸
 - 警示會新增為時間軸的閾值。
 - 正在將警示狀態及其輸入屬性資料新增至時間軸。
 - 您無法從小工具與 AWS IoT SiteWise 助理互動。

AWS IoT SiteWise 小工具中的助理使用

AWS IoT SiteWise 助理是生成式 AI 輔助。它可讓工廠經理、品質工程師和維護技術人員等使用者直接從其營運和企業資料中取得洞見、解決問題並採取動作。AWS IoT SiteWise 助理會將 AWS IoT 資料、資產模型、手冊和文件的資訊合併為可理解的關鍵事件摘要。它還啟用互動式深入探索問答工作階段，以便於診斷、根本原因探索和引導建議。

AWS IoT SiteWise 助理按鈕位於儀表板的右上角。按一下以啟用助理。只能與儀表板的預覽模式搭配使用。



在下列案例中使用 AWS IoT SiteWise 助理：

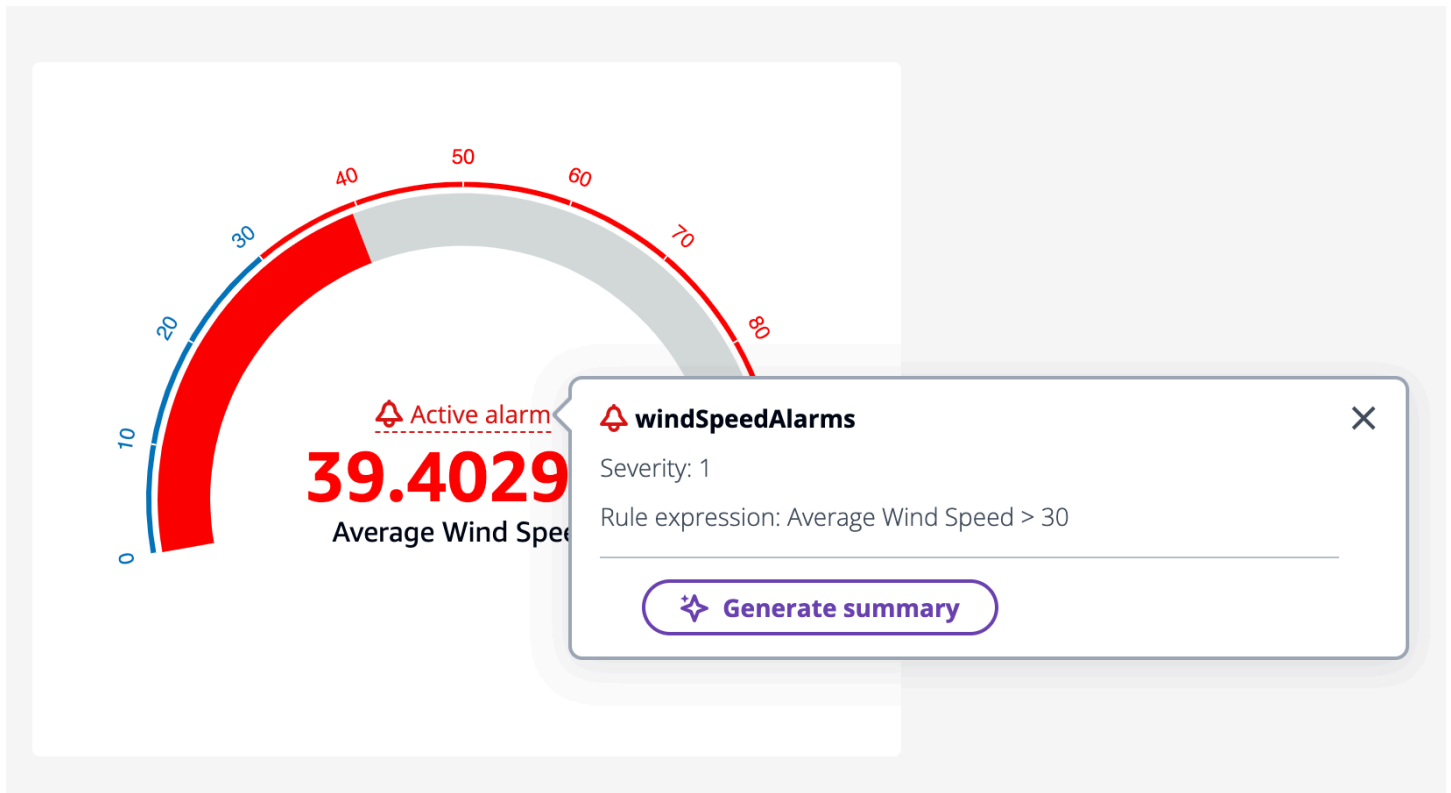
主題

- [使用案例 - 警示摘要](#)
- [使用案例 - 情境摘要](#)
- [使用案例 - 深入探討摘要](#)

使用案例 - 警示摘要

摘要儀表板上所選面板的目前警示。Line、KPI、Gauge 和 Table Widget 支援警示。選擇具有警示的小工具，並加以摘要。

- 選取小工具上的作用中警示。
- 警示會顯示嚴重性和規則表達式。
- 選擇產生摘要以產生摘要。



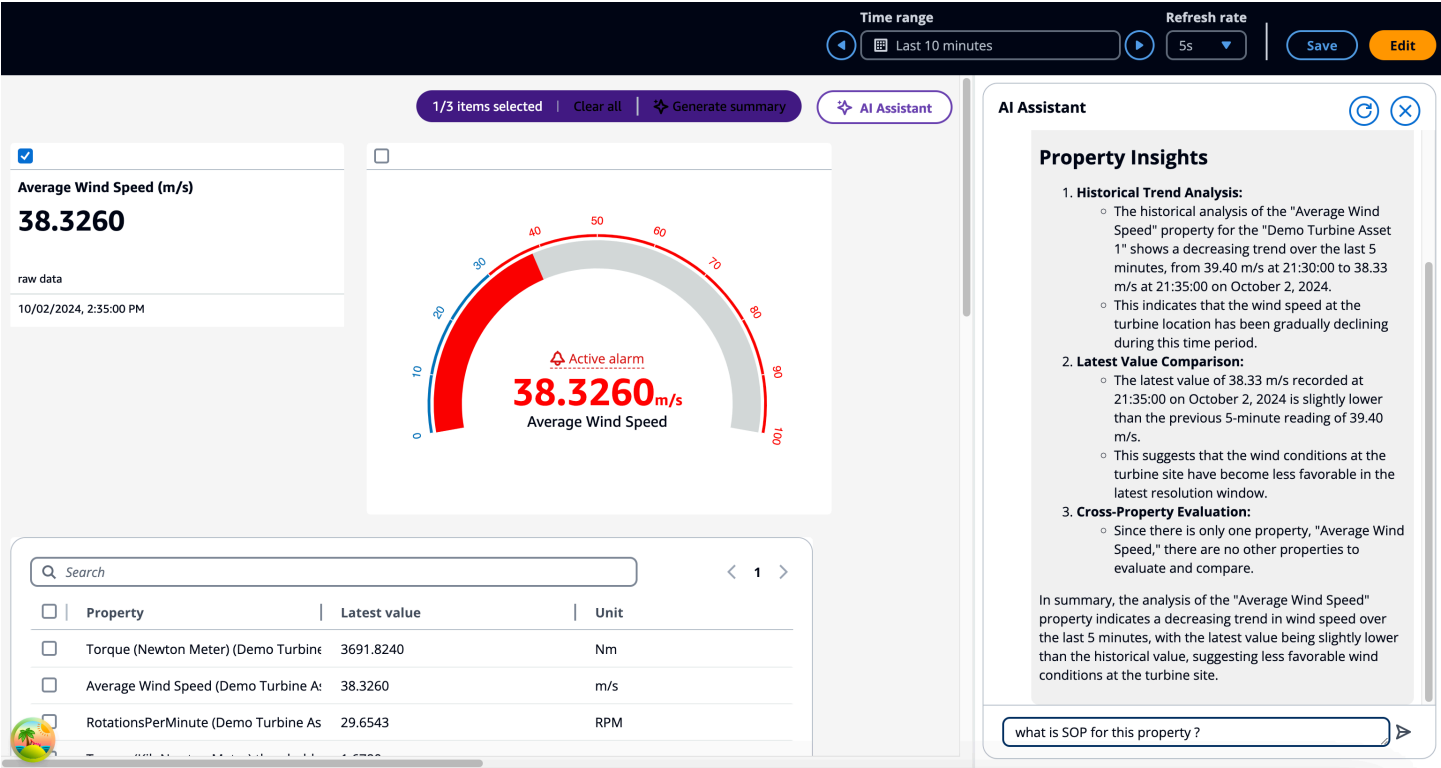
使用案例 - 情境摘要

最多選取要摘要的三個小工具。它們可以是小工具和屬性的組合。如果選取超過三個，則助理會傳回錯誤。

使用 AWS IoT SiteWise 助理產生情況摘要

- 按一下 AI 助理。它會顯示具有三個選項的選單。
 - 已選取項目 – 僅選取三個項目。您無法選取超過三個。
 - 全部清除 – 清除您的選擇。
 - 產生摘要 – 產生所選項目的摘要。
- 選擇產生摘要以產生所選項目的摘要。

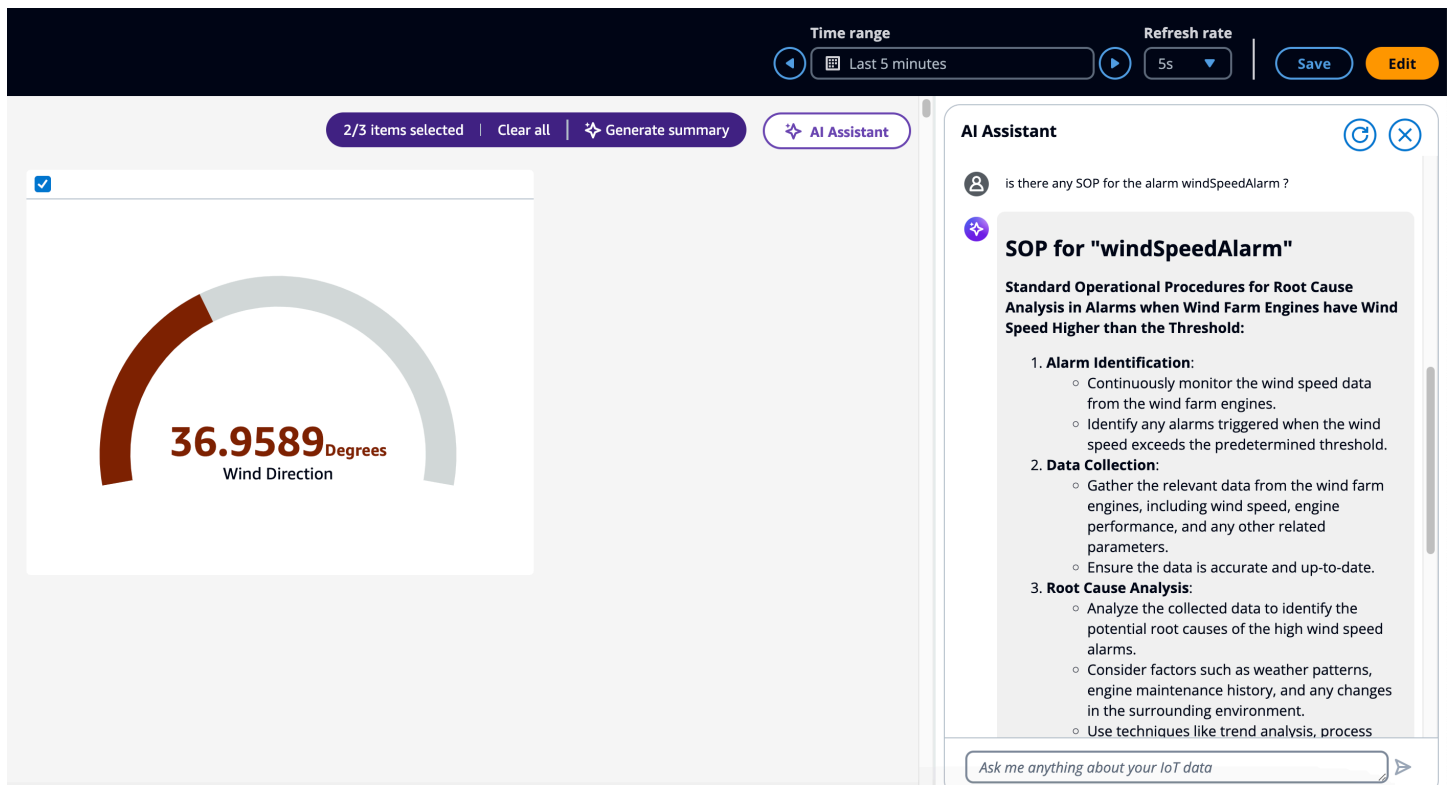
下圖已選取 Widget 和 AWS IoT SiteWise 助理的摘要。



使用案例 - 深入探討摘要

這是使用者可進行深入探索、存取 SOPs (標準操作程序)、手冊、文件，以及考慮後續步驟的使用案例。對於上一節中的範例，如果使用者選擇進一步了解此屬性的 SOP，請向助理詢問此屬性的 SOP。這會向使用者顯示有關 SOP 的深入分析資訊。

以下範例顯示「警示 windSpeedAlarm 是否有任何 SOP？」的答案



要詢問 AWS IoT SiteWise 助理的問題範例

Note

- AWS IoT SiteWise 助理必須使用具有 [Amazon Kendra](#) 索引的資料集，以取得企業層級知識和指導。如果您沒有 Amazon Kendra 索引，請參閱[建立索引](#)以建立索引。新增[資料集](#)可改善助理回應的品質。請參閱[建立資料集](#)以進一步了解。
- 有些問題需要 AWS IoT TwinMaker 整合。如需詳細資訊，請參閱[整合 AWS IoT TwinMaker 和 AWS IoT SiteWise](#)。

在儀表板中收到警示摘要後，在相同對話中要詢問助理的一些後續問題。

- 顯示上述摘要中的資產詳細資訊？
- 從根到所述資產的階層路徑是什麼？
- 上述資產的相依子系資產是什麼？
- 具有作用中警示的所述資產的相依資產是什麼？
- 尋找具有作用中警示的所有資產。

在儀表板中取得屬性摘要後，在相同對話中要詢問助理的一些後續問題。

- 在過去 24 小時內執行相同的分析。
- 尋找與上述屬性相關的文件。
- 提供資產 ID 1da67d28-14f8-4f71-a06a-386f0425a21d/asset name 示範渦輪資產 1 的詳細資訊。

從 API 叫用 AWS IoT SiteWise 助理。

- 在資產 ID 中產生警示名稱 windSpeedAlarm 的警示摘要d591e153-e5cf-4206-96bb-ce3c119d9d2d。
- 在資產 ID 中，為警示名稱 windSpeedAlarm 產生過去 12 小時/2 天/1 週的警示摘要d591e153-e5cf-4206-96bb-ce3c119d9d2d。
- 在資產 ID ab187fb7-d74b-44d9-bd9b-f2f19a9137cc中產生屬性 ID 的屬性摘要 d591e153-e5cf-4206-96bb-ce3c119d9d2d
- 針對資產 ID ab187fb7-d74b-44d9-bd9b-f2f19a9137cc中的屬性 ID 產生過去 12 小時/2 天/1 週的屬性摘要d591e153-e5cf-4206-96bb-ce3c119d9d2d。
- 尋找資產名稱為 Turbine 的資產。
- 提供5356168c-3390-456f-802c-9f6e047810d4資產 ID 中屬性 ID 的目前屬性值d591e153-e5cf-4206-96bb-ce3c119d9d2d , 3cbb084e-1ded-4b08-9f21-1b47b2fb86fd。
- 資產 ID d591e153-e5cf-4206-96bb-ce3c119d9d2d和資產 ID 之間的關係是什麼3cbb084e-1ded-4b08-9f21-1b47b2fb86fd。
- 尋找如何修正風力發電機低 RPM 問題的文件。
- 產生屬性別名 WindSpeed 的屬性摘要。
- 根據我的知識庫，什麼是操作前檢查？

從 查詢資料 AWS IoT SiteWise

您可以使用 AWS IoT SiteWise API 操作來查詢資產屬性的目前值、歷史值和特定時間間隔的彙總。

使用這些功能來深入了解您的資料。例如，使用指定的屬性值探索所有資產，或建置資料的自訂表示法。您也可以使用 API 操作來開發與儲存在 AWS IoT SiteWise 資產中的工業資料整合的軟體解決方案。您也可以即時在 AWS IoT SiteWise Monitor 中探索您的資產資料。若要了解如何設定 SiteWise Monitor，請參閱 [使用 監控資料 AWS IoT SiteWise Monitor](#)。

本節所述的操作會傳回包含時間戳記、品質、值 (TQV) 結構的屬性值物件：

- timestamp 包含目前的 Unix Epoch 時間，以秒為單位，具有奈秒偏移。
- quality 包含下列其中一個字串，指出資料點的品質：
 - GOOD – 資料不會受到任何問題的影響。
 - BAD – 資料會受到感應器故障等問題的影響。
 - UNCERTAIN – 資料受到感應器不準確等問題的影響。
- value 包含下列其中一個欄位，視 屬性的類型而定：
 - booleanValue
 - doubleValue
 - integerValue
 - stringValue
 - nullValue

主題

- [查詢 中的目前資產屬性值 AWS IoT SiteWise](#)
- [在 中查詢歷史資產屬性值 AWS IoT SiteWise](#)
- [在 中查詢資產屬性彙總 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 查詢語言](#)

查詢 中的目前資產屬性值 AWS IoT SiteWise

本教學課程顯示取得資產屬性目前值的兩種方式。您可以使用 AWS IoT SiteWise 主控台或在 AWS Command Line Interface () 中使用 API AWS CLI。

主題

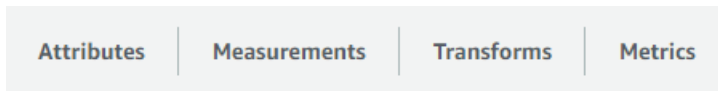
- [查詢資產屬性的目前值（主控台）](#)
- [查詢資產屬性的目前值 \(AWS CLI\)](#)

查詢資產屬性的目前值（主控台）

您可以使用 AWS IoT SiteWise 主控台來檢視資產屬性的目前值。

取得資產屬性的目前值 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇具有要查詢之屬性的資產。
4. 選擇箭頭圖示以展開資產階層來尋找您的資產。
5. 選擇屬性類型的標籤。例如，選擇 Measurements (衡量值) 以檢視衡量值屬性的目前值。



6. 尋找要檢視的屬性。目前值會顯示在 Latest value (最新值) 欄中。

查詢資產屬性的目前值 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來查詢資產屬性的目前值。

使用 [GetAssetPropertyValue](#) 操作來查詢資產屬性的目前值。

若要識別資產屬性，請指定下列其中一項：

- 資料傳送到的資產屬性propertyId的 assetId和。
- propertyAlias，這是資料串流別名（例如 /company/windfarm/3/turbine/7/temperature）。若要使用這個選項，您必須先設定您的資產屬性別名。若要設定屬性別名，請參閱 [管理的資料串流 AWS IoT SiteWise](#)。

若要取得資產屬性的目前值 (AWS CLI)

- 執行下列命令以取得資產屬性的目前值。將 *asset-id* 取代為資產的 ID，並將 *property-id* 取代為屬性的 ID。


```
aws iotsitewise get-asset-property-value \  
  --asset-id asset-id \  
  --property-id property-id
```

該操作會傳回回應，其中包含以下格式的屬性目前 TQV。

```
{  
  "propertyValue": {  
    "value": {  
      "booleanValue": Boolean,  
      "doubleValue": Number,  
      "integerValue": Number,  
      "stringValue": "String",  
      "nullValue": {  
        "valueType": "String"  
      }  
    },  
    "timestamp": {  
      "timeInSeconds": Number,  
      "offsetInNanos": Number  
    },  
    "quality": "String"  
  }  
}
```

在 中查詢歷史資產屬性值 AWS IoT SiteWise

您可以使用 AWS IoT SiteWise API [GetAssetPropertyValueHistory](#) 操作來查詢資產屬性的歷史值。

若要識別資產屬性，請指定下列其中一項：

- 資料傳送到的資產屬性propertyId的 assetId和。
- propertyAlias，這是資料串流別名（例如 /company/windfarm/3/turbine/7/temperature）。若要使用這個選項，您必須先設定您的資產屬性別名。若要設定屬性別名，請參閱 [管理 的資料串流 AWS IoT SiteWise](#)。

傳遞下列參數以精簡您的結果：

- startDate – 查詢歷史資料範圍的專屬起點，以秒為單位，以 Unix epoch 時間表示。

- `endDate` – 查詢歷史資料範圍的包含結尾，以秒為單位，以 Unix epoch 時間表示。
- `maxResults` – 在單一請求中傳回的結果數目上限。預設為20結果。
- `nextToken` – 從此操作的上一個呼叫傳回的分頁字符。
- `timeOrdering` – 要套用至傳回值的排序：ASCENDING或 DESCENDING。
- `qualities` – 篩選結果的品質：GOOD、BAD或 UNCERTAIN。

查詢資產屬性的值歷史記錄 (AWS CLI)

1. 執行下列命令以取得資產屬性的值歷程記錄。此命令會在特定的 10 分鐘間隔內查詢屬性的歷程記錄。將 *asset-id* 取代為資產的 ID，並將 *property-id* 取代為屬性的 ID。將日期參數取代為要查詢的間隔。

```
aws iotsitewise get-asset-property-value-history \
  --asset-id asset-id \
  --property-id property-id \
  --start-date 1575216000 \
  --end-date 1575216600
```

操作會傳回回應，其中包含屬性的歷史 TQVs格式如下：

```
{
  "assetPropertyValueHistory": [
    {
      "value": {
        "booleanValue": Boolean,
        "doubleValue": Number,
        "integerValue": Number,
        "stringValue": "String",
        "nullValue": {
          "valueType": "String"
        }
      },
      "timestamp": {
        "timeInSeconds": Number,
        "offsetInNanos": Number
      },
      "quality": "String"
    }
  ],
  "nextToken": "String"
}
```

```
}
```

2. 如果存在更多值項目，您可以將分頁字符從 `nextToken` 欄位傳遞至後續呼叫 [GetAssetPropertyValueHistory](#) 操作。

在中查詢資產屬性彙總 AWS IoT SiteWise

AWS IoT SiteWise 會自動計算彙總的資產屬性值，這是在多個時間間隔內計算的一組基本指標。會為您的資產屬性 AWS IoT SiteWise 計算下列每分鐘、每小時和每天彙總：

- 平均值 – 屬性值在時間間隔內的平均值（平均值）。
- count – 屬性在時間間隔內的資料點數目。
- maximum – 時間間隔內屬性值的最大值。
- minimum – 時間間隔內屬性值的最小值。
- 標準差 – 屬性值在時間間隔內的標準差。
- sum – 屬性在時間間隔內值的總和。

對於非數值屬性，例如字串和布林值，只會 AWS IoT SiteWise 計算計數彙總。

您也可以為您的資產資料計算自訂指標。使用指標屬性，您可以定義操作特有的彙總。指標屬性提供未針對 AWS IoT SiteWise API 預先計算的額外彙總函數和時間間隔。如需詳細資訊，請參閱[從屬性和其他資產彙總資料（指標）](#)。

主題

- [彙總資產屬性 \(API\)](#)
- [彙總資產屬性 \(AWS CLI\)](#)

彙總資產屬性 (API)

使用 AWS IoT SiteWise API 取得資產屬性的彙總。

使用 [GetAssetPropertyAggregates](#) 操作來查詢資產屬性的彙總。

若要識別資產屬性，請指定下列其中一項：

- 資料傳送到的資產屬性 `propertyId` 的 `assetId` 和 。


```
--property-id property-id \  
--start-date 1575216000 \  
--end-date 1575219600 \  
--aggregate-types AVERAGE SUM \  
--resolution 1h
```

此操作會傳回回應，其中包含以下格式的屬性歷史 TQV。回應只包括請求的彙總。

```
{  
  "aggregatedValues": [  
    {  
      "timestamp": Number,  
      "quality": "String",  
      "value": {  
        "average": Number,  
        "count": Number,  
        "maximum": Number,  
        "minimum": Number,  
        "standardDeviation": Number,  
        "sum": Number  
      }  
    }  
  ],  
  "nextToken": "String"  
}
```

2. 如果存在更多值項目，您可以將分頁字符從 nextToken 欄位傳遞至後續呼叫 [GetAssetPropertyAggregates](#) 操作。

Note

如果您的查詢範圍包含null值 TQVs，請參閱 [AssetPropertyValue](#) API。計數以外的所有統計資料都會產生null回應，類似於字串 TQVs統計資料。如果您的查詢範圍包含Double.NaN雙重類型 TQVs則計數以外的所有計算都會產生 Double.NaN。

AWS IoT SiteWise 查詢語言

透過 AWS IoT SiteWise 資料擷取 [ExecuteQuery](#) API 操作，您可以從下列項目擷取宣告性結構定義的相關資訊，以及與其相關聯的時間序列資料：

- 模型
- 資產
- 測量
- 指標
- 轉換
- 彙總

這可以在單一 API 請求中使用 SQL，例如查詢陳述式。

Note

除了 AWS GovCloud（美國西部）之外，此功能適用於所有同時 AWS IoT TwinMaker 提供 AWS IoT SiteWise 和 的 區域。

主題

- [先決條件](#)
- [的查詢語言參考 AWS IoT SiteWise](#)

先決條件

AWS IoT SiteWise 需要與 整合的許可，AWS IoT TwinMaker 才能組織和建模工業資料。

在您可以擷取模型、資產、測量、指標、轉換和彙總的相關資訊之前，請確定符合下列先決條件：

- 您 AWS 帳戶中 AWS IoT SiteWise 和 AWS IoT TwinMaker 設定的服務連結角色。如需服務連結角色的詳細資訊，請參閱《IAM 使用者指南》中的[建立服務連結角色](#)。
- IAM 角色的已啟用 AWS IoT SiteWise 整合。如需詳細資訊，請參閱[整合 AWS IoT SiteWise 和 AWS IoT TwinMaker](#)。
- IoTSiteWiseDefaultWorkspace 區域中您帳戶中 ID 為 的 AWS IoT TwinMaker 工作區。如需詳細資訊，請參閱AWS IoT TwinMaker 《使用者指南》中的[使用 IoTSiteWiseDefaultWorkspace](#)。
- AWS IoT TwinMaker 已啟用的標準或分層套件定價模式。如需詳細資訊，請參閱AWS IoT TwinMaker 《使用者指南》中的[切換 AWS IoT TwinMaker 定價模式](#)。

的查詢語言參考 AWS IoT SiteWise

AWS IoT SiteWise 支援豐富的查詢語言來使用您的資料。下列主題說明可用的資料類型、運算子、函數和建構。

請參閱 [查詢範例](#) 以使用查詢語言撰寫 AWS IoT SiteWise 查詢。

主題

- [了解查詢參考檢視](#)
- [支援的資料類型](#)
- [使用 SELECT 陳述式擷取資料](#)
- [邏輯運算子](#)
- [比較運算子](#)
- [查詢範例](#)

了解查詢參考檢視

本節提供的資訊可協助您了解 中的檢視 AWS IoT SiteWise，例如程序中繼資料和遙測資料。

下表提供檢視的檢視名稱和說明。

資料模型

檢視名稱	檢視說明
資產	包含資產和模型衍生的相關資訊。
asset_property	包含資產屬性結構的相關資訊。
raw_time_series	包含時間序列的歷史資料。
latest_value_time_series	包含時間序列的最新值。
precomputed_aggregates	包含自動計算的彙總資產屬性值。它們是一組在多個時間間隔內計算的基本指標。

下列檢視會列出查詢的資料欄名稱以及範例資料。

檢視：資產

asset_id	asset_name	asset_description	asset_model_id
88898498-0b8b-42b5-bf57-16180bc3d3a0	WindTurbine A	WindTurbine 資產 A	17847250-5bf0-4f74-b775-cc03f05e7cb8
17847250-5bf0-4f74-b775-cc03f05e7cb8	風力渦輪資產模型	代表風力發電廠中的渦輪機。	

檢視：asset_property

property_id	asset_id	property_name	property_alias	asset_composite_model_id	
b29be434-b000-4d74-b809-75287d83bcd6	88898498-0b8b-42b5-bf57-16180bc3d3a0	馬達溫度	Rochester2/44///Line-5/Bus-2/Machine-5/Temperature		
3b458f00-24e7-458a-b4e8-c6026eff654a	88898498-0b8b-42b5-bf57-16180bc3d3a0	風向	/company/windfarm/3/turbine/7/winddirection	2f458n00-56e7-458h-b4e8-c6026eff985g	

檢視：raw_time_series

asset_id	property_id	property_alias	event_timestamp	品質	布林值	int_value	double_value	string_value
88898498-0b8b-42b5-bf57-161	b29be434-b000-4d74-b809-752	Rochester2/44///Line-5/	157521960	良好			115.0	

asset_id	property_id	property_alias	event_timestamp	品質	布林值	int_value	double_value	string_value
80bc3d3a	87d83bcd	Bus-2/ Machine-5/ Temperature						
888984980b8b-42b1-bf57-16180bc3d3a	3b458f00-24e7-458a-b4e8-c6026eff654a	/ company, windfarm turbine /7/ winddirection	157521937	良好			348.75	

Note

您必須在資料event_timestamp欄上包含篩選條件子句，才能查詢raw_time_series檢視。這是必要的篩選條件，如果沒有它，查詢將會失敗。

Example query

```
SELECT event_timestamp, double_value FROM raw_time_series WHERE event_timestamp > 1234567890
```

檢視：latest_value_time_series

asset_id	property_id	property_alias	event_timestamp	品質	布林值	int_value	double_value	string_value
888984980b8b-42b1-bf57-16180bc3d3a	3b458f00-24e7-458a-b4e8-c6026eff654a	/ company, windfarm turbine /7/ winddirection	157521960	良好			355.39	

asset_id	property_id	property_alias	event_timestamp	品質	布林值	int_value	double_value	string_value
-bf57-16180bc3d3a	-b4e8-c6026eff654a	windfarm3/turbine/7/winddirection						

檢視：precomputed_aggregates

asset_id	property_id	property_alias	event_timestamp	品質	解析度	sum_value	count_value	average_value	maximum_value	minimum_value	stdev_value
8889840b8b-4	b29be4b000-4	Roche2/44/, 0	157521	良好	15m	1105.4	15	73.4	80.6	68	3.64
-bf57-16180bc3d3a	-b4e8-c6026eff654a	Li ne-5/Bus-2/Machine-5/Temperature									

支援的資料類型

AWS IoT SiteWise 查詢語言支援下列資料類型。

純量值

資料類型	Description
STRING	長度上限為 1024 位元組的字串。

資料類型	Description
INTEGER	帶正負號的 32 位元整數，範圍為 -2,147,483,648 to 2,147,483,647 。
DOUBLE	範圍為 -10^{100} to 10^{100} 或IEEE 754雙精度Nan的浮點數。
BOOLEAN	true 或 false

Null 值：表示true缺少定義資料的布林值。

Note

雙精確度資料並不精確。某些值不會完全轉換，而且由於精確度有限，不會代表所有實際數字。查詢中的浮點資料可能與內部表示的值不同。如果輸入號碼的精確度太高，則值會四捨五入。

使用 SELECT 陳述式擷取資料

SELECT 陳述式用於從一或多個檢視擷取資料。AWS IoT SiteWise 支援隱含JOIN檢視。您可以使用逗號來列出要聯結的檢視（在 SELECT陳述式的 FROM子句中）。

Example

使用下列SELECT陳述式：

```
SELECT select_expr [, ...]
[ FROM from_item [, ...] ]
[ WHERE [LIKE condition ESCAPE condition] ]
```

在先前的範例中，LIKE子句會使用萬用字元指定搜尋和篩選條件。AWS IoT SiteWise 支援 percentage (%)做為萬用字元。

Example 在 條件%中使用：

```
Prefix search: String%
Infix search: %String%
```

```
Suffix search: %String
```

Example 搜尋資產：

```
SELECT asset_name, asset_description FROM asset WHERE asset_name LIKE 'Wind%'
```

Example 使用 ESCAPE 條件搜尋資產：

```
SELECT asset_name, asset_description FROM asset WHERE asset_name LIKE 'room\%' ESCAPE
'\'
```

邏輯運算子

AWS IoT SiteWise 支援下列邏輯運算子。

邏輯運算子

運算子	Description	範例
AND	TRUE 如果兩個值都是 true	a AND b

如果 a 或 b 是 FALSE，則先前的表達式會評估為 false。若要讓 AND 運算子評估為 true，a 和 b 都必須為 true。

Example

```
SELECT a.asset_name
FROM asset as a, latest_value_time_series as t
WHERE t.int_value > 30 AND t.event_timestamp > 1234567890
```

比較運算子

AWS IoT SiteWise 支援下列比較運算子。所有比較操作都可用於內建資料類型，並評估為布林值。

邏輯運算子

運算子	Description
<	小於

運算子	Description
>	大於
<=	小於或等於
>=	大於或等於
=	等於
!=	不等於

非數值的比較操作事實資料表

類型	類型 >= x	類型 <= x	類型 > x	類型 < x	類型 = x	輸入 != x
NaN	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

有些述詞的行為類似於運算子，但具有特殊語法。如需範例，請參閱以下。

比較述詞

運算子	Description
IS NULL	測試值是否為 NULL。
IS NOT NULL	測試值是否不是 NULL。
IS NaN	測試值是否為 NaN。
IS NOT NaN	測試值是否不是 NaN。

查詢範例

中繼資料篩選

下列範例適用於使用 AWS IoT SiteWise 查詢語言的 SELECT 陳述式進行中繼資料篩選：

```
SELECT a.asset_name, p.property_name
FROM asset a, asset_property p
WHERE a.asset_id = p.asset_id AND a.asset_name LIKE '%windmill%'
```

值篩選

以下是使用具有 AWS IoT SiteWise 查詢語言的SELECT陳述式篩選值的範例：

```
SELECT a.asset_name FROM asset a, raw_time_series r
WHERE a.asset_id = r.asset_id AND r.int_value > 30 AND r.event_timestamp > 1234567890
AND r.event_timestamp < 1234567891
```

與其他 AWS 服務互動

AWS IoT SiteWise 可以將資產資料發佈至 AWS IoT MQTT 發佈訂閱訊息代理程式，以便您可以從其他 AWS 服務與資產資料互動。AWS IoT SiteWise 會為每個資產屬性指派唯一的 MQTT 主題，您可以使用 AWS IoT 核心規則將資產資料路由至其他 AWS 服務。例如，您可以設定 AWS IoT 核心規則來執行下列任務：

- 識別設備故障，並透過將資料傳送至 [AWS IoT Events](#) 來通知適當的人員。
- 將資料傳送至 [Amazon DynamoDB](#)，將要用於外部軟體解決方案的選定資產資料組織化。
- 透過觸發 [AWS Lambda](#) 函數來產生每週報告。

您可以遵循教學課程，逐步完成設定規則所需的步驟，將屬性值存放在 DynamoDB 中。如需詳細資訊，請參閱[將屬性值更新發佈至 Amazon DynamoDB](#)。

如需如何設定規則的詳細資訊，請參閱《AWS IoT 開發人員指南》中的[規則](#)。

您也可以將其他 AWS 服務的資料耗用回 AWS IoT SiteWise。若要透過 AWS IoT SiteWise 規則動作擷取資料，請參閱[AWS IoT SiteWise 使用 AWS IoT Core 規則將資料擷取至](#)。

主題

- [了解 MQTT 主題中的資產屬性](#)
- [在 中開啟資產屬性通知 AWS IoT SiteWise](#)
- [在 中查詢資產屬性通知 AWS IoT SiteWise](#)
- [使用資產屬性通知將資料匯出至 Amazon S3](#)
- [AWS IoT SiteWise 與 Grafana 整合](#)
- [整合 AWS IoT SiteWise 和 AWS IoT TwinMaker](#)
- [使用 Lookout for Equipment 偵測異常](#)

了解 MQTT 主題中的資產屬性

每個資產屬性都有以下格式的唯一 MQTT 主題路徑。

```
$aws/sitewise/asset-models/assetModelId/assets/assetId/properties/propertyId
```

 Note

AWS IoT SiteWise 不支援 AWS IoT 核心規則引擎中的 # (多層級) 主題篩選條件萬用字元。您可以使用 + (單層級) 萬用字元。例如，您可以使用下列主題篩選條件來比對特定資產模型的所有更新。

```
$aws/sitewise/asset-models/assetModelId/assets/+/properties/+
```

若要進一步了解主題篩選條件萬用字元，請參閱 AWS IoT 核心開發人員指南中的[主題](#)。

在中開啟資產屬性通知 AWS IoT SiteWise

您可以啟用屬性通知，將資產資料更新發佈至 AWS IoT Core，然後對資料執行查詢。透過資產屬性通知，AWS IoT SiteWise 提供範本 AWS CloudFormation，可讓您用來將 AWS IoT SiteWise 資料匯出至 Amazon S3。

 Note

無論值是否變更 AWS IoT SiteWise，資產資料 AWS IoT Core 都會在每次收到時傳送至。

主題

- [開啟資產屬性通知 \(主控台\)](#)
- [開啟資產屬性通知 \(AWS CLI\)](#)

開啟資產屬性通知 (主控台)

根據預設，AWS IoT SiteWise 不會發佈屬性值更新。您可以使用 AWS IoT SiteWise 主控台來啟用資產屬性的通知。

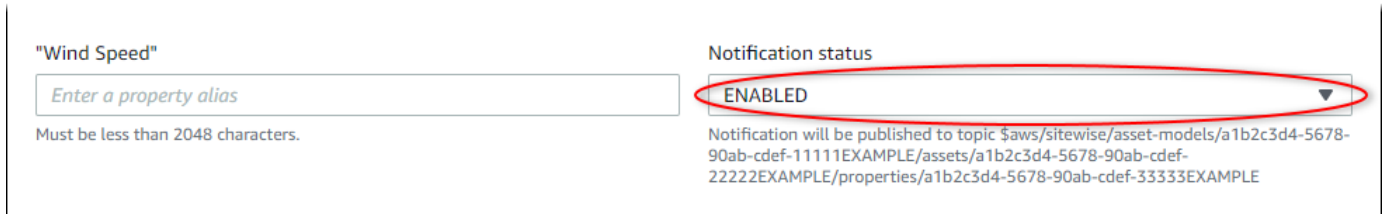
啟用或停用資產屬性的通知 (主控台)

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Assets (資產)。
3. 選擇要啟用屬性通知的資產。

i Tip

您可以選擇箭頭圖示來展開資產階層，以尋找您的資產。

4. 選擇編輯。
5. 針對資產屬性的 Notification status (通知狀態)，選擇 ENABLED (已啟用)。



The screenshot shows a form for editing an asset property. On the left, there is a text input field labeled '"Wind Speed"' with a placeholder 'Enter a property alias' and a note 'Must be less than 2048 characters.' On the right, there is a 'Notification status' dropdown menu. The dropdown is open, showing 'ENABLED' as the selected option. Below the dropdown, there is a text block indicating the notification topic: 'Notification will be published to topic \$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'.

您也可以選擇 DISABLED (已停用) 以停用資產屬性的通知。

6. 選擇 Save (儲存)。

開啟資產屬性通知 (AWS CLI)

根據預設，AWS IoT SiteWise 不會發佈屬性值更新。您可以使用 AWS Command Line Interface (AWS CLI) 來啟用或停用資產屬性的通知。

您必須知道資產的 `assetId` 和財產的 `propertyId`，才能完成此程序。您也可以使用外部 ID。如果您建立資產但不知道其 `assetId`，請使用 [ListAssets](#) API 列出特定模型的所有資產。使用 [DescribeAsset](#) 操作來檢視資產的屬性，包括屬性 IDs。

使用 [UpdateAssetProperty](#) 操作來啟用或停用資產屬性的通知。指定下列參數：

- `assetId` – 資產的 ID。
- `propertyId` – 資產屬性的 ID。
- `propertyNotificationState` – 屬性值通知狀態：ENABLED 或 DISABLED。
- `propertyAlias` – 屬性的別名。當您更新通知狀態時，指定屬性的現有別名。如果您省略此參數，則會移除屬性的現有別名。

啟用或停用資產屬性的通知 (CLI)

1. 執行下列命令來擷取資產屬性的別名。將 *asset-id* 取代為資產的 ID，並將 *property-id* 取代為屬性的 ID。

```
aws iotsitewise describe-asset-property \  
  --asset-id asset-id \  
  --property-id property-id
```

此操作會以下列格式傳回包含資產屬性詳細資料的回應。屬性別名位於 JSON 物件的 `assetProperty.alias` 中。

```
{  
  "assetId": "a1b2c3d4-5678-90ab-cdef-2222EXAMPLE",  
  "assetName": "Wind Turbine 7",  
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-1111EXAMPLE",  
  "assetProperty": {  
    "id": "a1b2c3d4-5678-90ab-cdef-3333EXAMPLE",  
    "name": "Wind Speed",  
    "alias": "/company/windfarm/3/turbine/7/windspeed",  
    "notification": {  
      "topic": "$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-1111EXAMPLE/  
assets/a1b2c3d4-5678-90ab-cdef-2222EXAMPLE/properties/a1b2c3d4-5678-90ab-  
cdef-3333EXAMPLE",  
      "state": "DISABLED"  
    },  
    "dataType": "DOUBLE",  
    "unit": "m/s",  
    "type": {  
      "measurement": {}  
    }  
  }  
}
```

2. 執行下列命令來啟用資產屬性的通知。將 *property-alias* 取代為先前命令回應中的屬性別名，或省略 `--property-alias` 以更新屬性但不帶別名。

```
aws iotsitewise update-asset-property \  
  --asset-id asset-id \  
  --property-id property-id \  
  --property-notification-state ENABLED \  
  --property-alias property-alias
```

您也可以傳遞 `--property-notification-state DISABLED` 以停用資產屬性的通知。

在 中查詢資產屬性通知 AWS IoT SiteWise

若要查詢資產屬性通知，請建立由 SQL 陳述式組成的 AWS IoT Core 規則。

AWS IoT SiteWise 會以下列格式將資產屬性資料更新發佈至 AWS IoT Core。

```
{
  "type": "PropertyValueUpdate",
  "payload": {
    "assetId": "String",
    "propertyId": "String",
    "values": [
      {
        "timestamp": {
          "timeInSeconds": Number,
          "offsetInNanos": Number
        },
        "quality": "String",
        "value": {
          "booleanValue": Boolean,
          "doubleValue": Number,
          "integerValue": Number,
          "stringValue": "String",
          "nullValue": {
            "valueType": "String"
          }
        }
      }
    ]
  }
}
```

values 清單中的每個結構都是時間戳記品質值 (TQV) 結構。

- timestamp 包含目前的 Unix epoch 時間，以秒為單位，具有奈秒偏移。
- quality 包含下列其中一個字串，指出資料點的品質：
 - GOOD – 資料不會受到任何問題的影響。
 - BAD – 資料會受到感應器故障等問題的影響。
 - UNCERTAIN – 資料受到感應器不準確等問題的影響。
- 根據 屬性的類型，value 包含下列其中一個欄位：

- booleanValue
- doubleValue
- integerValue
- stringValue
- nullValue

nullValue – 具有下列欄位的結構，表示值為 Null 且品質為 BAD 或 的屬性值類型 UNCERTAIN。

- valueType – {"B"、"D"、"S"、"I"} 的列舉

若要解析 values 陣列中的值，您需要使用規則 SQL 陳述式中複雜的巢狀物件查詢。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[巢狀物件查詢](#)，或參閱 [將屬性值更新發佈至 Amazon DynamoDB](#) 教學課程以取得剖析資產屬性通知訊息的特定範例。

Example 擷取值陣列的查詢範例

下列陳述式示範如何在有該屬性的所有資產上針對特定雙類型屬性查詢更新屬性值的陣列。

```
SELECT
  (SELECT VALUE (value.doubleValue) FROM payload.values) AS windspeed
FROM
  '$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/+/
properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'
WHERE
  type = 'PropertyValueUpdate'
```

先前的規則查詢陳述式會以下列格式輸出資料。

```
{
  "windspeed": [
    26.32020195042838,
    26.282584572975477,
    26.352566977372508,
    26.283084346171442,
    26.571883739599322,
    26.60684140743005,
    26.628738636715045,
    26.273486932802125,
    26.436379105473964,
```

```
    26.600590095377303
  ]
}
```

Example 擷取單一值的查詢範例

下列陳述式示範如何在有該屬性的所有資產上針對特定雙類型屬性，從屬性值的陣列查詢第一個值。

```
SELECT
  get((SELECT VALUE (value.doubleValue) FROM payload.values), 0) AS windspeed
FROM
  '$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/+/
properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'
WHERE
  type = 'PropertyValueUpdate'
```

先前的規則查詢陳述式會以下列格式輸出資料。

```
{
  "windspeed": 26.32020195042838
}
```

Important

這個規則查詢陳述式會忽略每個批次中第一個項目之外的其他值更新。每個批次最多可以包含 10 個值。如果您需要包含其餘的值，您必須設定更複雜的解決方案，將資產屬性值輸出到其他服務。例如，您可以設定具有 AWS Lambda 動作的規則，將陣列中的每個值重新發佈至另一個主題，並設定另一個規則來查詢該主題，並將每個值發佈至所需的規則動作。

使用資產屬性通知將資料匯出至 Amazon S3

您可以將傳入的資料從 匯出 AWS IoT SiteWise 到您帳戶中的 Amazon S3 儲存貯體。您可以備份資料的格式，以用來建立歷史報告或使用複雜的方法分析資料。

若要從 匯出時間序列資料 AWS IoT SiteWise，請啟用冷層功能，將資料存放在 Amazon S3 儲存貯體中。如需詳細資訊，請參閱在 [中管理資料儲存 AWS IoT SiteWise](#)。

若要從 匯出資產模型和資產中繼資料 AWS IoT SiteWise，請使用大量操作功能將中繼資料匯出至 Amazon S3 儲存貯體。如需詳細資訊，請參閱 [使用資產和模型進行大量操作](#)。

AWS IoT SiteWise 與 Grafana 整合

Grafana 是一種資料視覺化平台，用於視覺化和監控儀表板中的資料。在 Grafana 10.4.0 版及更新版本中，使用 AWS IoT SiteWise 外掛程式在 Grafana 儀表板中視覺化您的 AWS IoT SiteWise 資產資料。使用者可以使用單一 Grafana 儀表板視覺化來自多個 AWS 來源（例如 AWS IoT SiteWise Amazon Timestream 和 Amazon CloudWatch）和其他資料來源的資料。

您有兩個選項可以使用 AWS IoT SiteWise 外掛程式：

- 本機 Grafana 伺服器

您可以在您管理的 Grafana 伺服器上設定 AWS IoT SiteWise 外掛程式。如需如何新增和使用外掛程式的詳細資訊，請參閱 GitHub 網站上的[AWS IoT SiteWise 資料來源 README](#) 檔案。

- AWS Managed Service for Grafana

您可以在 AWS Managed Service for Grafana (AMG) 中使用 AWS IoT SiteWise 外掛程式。AMG 會為您管理 Grafana 伺服器，因此您可以視覺化資料，而無需建置、封裝或部署任何硬體或任何其他 Grafana 基礎設施。如需詳細資訊，請參閱《AWS Managed Service for Grafana 使用者指南》中的下列主題：

- [什麼是 Amazon Managed Service for Grafana \(AMG\)？](#)
- [使用 AWS IoT SiteWise 資料來源](#)

Example Grafana 儀表板範例

下列 Grafana 儀表板可視覺化[示範風力發電廠](#)。您可以在 [Grafana Play](#) 網站上存取此示範儀表板。



整合 AWS IoT SiteWise 和 AWS IoT TwinMaker

與整合可 AWS IoT TwinMaker 授予中強大功能的存取權 AWS IoT SiteWise，例如 AWS IoT SiteWise 主控台內的 AWS IoT SiteWise 資料擷取 ExecuteQuery API 和進階資產搜尋。若要整合服務並使用這些功能，您必須先啟用整合。

主題

- [啟用整合](#)
- [整合 AWS IoT SiteWise 和 AWS IoT TwinMaker](#)

啟用整合

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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。如需 AWS IoT SiteWise 支援動作的詳細資訊，請參閱《服務授權參考》中的 [定義的動作 AWS IoT SiteWise](#)。

如需 AWS IoT TwinMaker 服務連結角色的詳細資訊，請參閱AWS IoT TwinMaker 《使用者指南》中的 [的服務連結角色 AWS IoT TwinMaker](#)。

您必須先 AWS IoT TwinMaker 授予下列許可 AWS IoT SiteWise，允許與 AWS IoT TwinMaker 連結的工作區整合，才能整合 AWS IoT SiteWise 和：

- `iotsitewise:EnableSiteWiseIntegration` – 允許與連結的 AWS IoT TwinMaker 工作區 AWS IoT SiteWise 整合。此整合允許 AWS IoT SiteWise 透過 AWS IoT TwinMaker 服務連結角色 AWS IoT TwinMaker 讀取中的所有模型資訊。若要啟用此許可，請將下列政策新增至您的 IAM 角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:EnableSiteWiseIntegration"
      ],
      "Resource": "*"
    }
  ]
}
```

整合 AWS IoT SiteWise 和 AWS IoT TwinMaker

若要整合 AWS IoT SiteWise 和 AWS IoT TwinMaker，您必須具有下列項目：

- AWS IoT SiteWise 帳戶中設定的服務連結角色
- AWS IoT TwinMaker 帳戶中設定的服務連結角色
- AWS IoT TwinMaker `IoTSiteWiseDefaultWorkspace` 區域中您帳戶中 ID 為 的工作區。

使用 AWS IoT SiteWise 主控台整合

當您在主控台中看到與橫幅整合 AWS IoT TwinMaker時，請選擇授予許可。先決條件會在您的帳戶中建立。

使用 整合 AWS CLI

若要使用 AWS IoT TwinMaker 整合 AWS IoT SiteWise 和 AWS CLI，請輸入下列命令：

1. `CreateServiceLinkedRole` 使用 `AWSServiceName` 的 呼叫 `iotsitewise.amazonaws.com`。

```
aws iam create-service-linked-role --aws-service-name iotsitewise.amazonaws.com
```

2. `CreateServiceLinkedRole` 使用 `AWSServiceName` 的 呼叫 `iottwinmaker.amazonaws.com`。

```
aws iam create-service-linked-role --aws-service-name iottwinmaker.amazonaws.com
```

3. `CreateWorkspace` 使用 ID 的 呼叫 `IoTSiteWiseDefaultWorkspace`。

```
aws iottwinmaker create-workspace --workspace-id IoTSiteWiseDefaultWorkspace
```

使用 Lookout for Equipment 偵測異常

Note

異常偵測僅適用於可使用 Amazon Lookout for Equipment 的區域。

您可以將 AWS IoT SiteWise 與 Amazon Lookout for Equipment 整合，透過異常偵測和工業設備的預測性維護來深入了解工業設備。Lookout for Equipment 是一種機器學習 (ML) 服務，用於監控偵測異常設備行為並識別潛在故障的工業設備。使用 Lookout for Equipment，您可以實作預測性維護計畫，並識別最佳的設備程序。如需 Lookout for Equipment 的詳細資訊，請參閱 [《Amazon Lookout for Equipment 使用者指南》](#) 中的什麼是 Amazon Lookout for Equipment？。

當您建立預測來訓練 ML 模型以偵測異常設備行為時，AWS IoT SiteWise 會將資產屬性值傳送至 Lookout for Equipment 以訓練 ML 模型以偵測異常設備行為。若要在資產模型上定義預測定義，請指

定 Lookout for Equipment 存取資料所需的 IAM 角色，以及傳送至 Lookout for Equipment 並將處理的資料傳送至 Amazon S3 的屬性。如需詳細資訊，請參閱[在中建立資產模型 AWS IoT SiteWise](#)。

若要整合 AWS IoT SiteWise 和 Lookout for Equipment，您將執行下列高階步驟：

- 在資產模型上新增預測定義，概述您要追蹤哪些屬性。預測定義是可重複使用的測量、轉換和指標集合，用於根據該資產模型在資產上建立預測。
- 根據您提供的歷史資料來訓練預測。
- 排程推論，告知執行特定預測 AWS IoT SiteWise 的頻率。

排定推論後，Lookout for Equipment 模型會監控從設備接收到的資料，並尋找設備行為的異常。您可以使用 AWS IoT SiteWise GET API 操作或 Lookout for Equipment 主控台，在 SiteWise Monitor 中檢視和分析結果。您也可以使用資產模型中的警示偵測器來建立警示，以提醒您異常的設備行為。

主題

- [新增預測定義（主控台）](#)
- [訓練預測（主控台）](#)
- [在預測上開始或停止推論（主控台）](#)
- [新增預測定義 \(CLI\)](#)
- [訓練預測並開始推論 \(CLI\)](#)
- [訓練預測 \(CLI\)](#)
- [在預測上啟動或停止推論 \(CLI\)](#)

新增預測定義（主控台）

若要開始 AWS IoT SiteWise 將收集的資料傳送至 Lookout for Equipment，您必須將 AWS IoT SiteWise 預測定義新增至資產模型。

將預測定義新增至 AWS IoT SiteWise 資產模型

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇模型，然後選取您要新增預測定義的資產模型。
3. 選擇預測。
4. 選擇新增預測定義。
5. 定義預測定義的詳細資訊。

- a. 為您的預測定義輸入唯一的名稱和描述。仔細選擇名稱，因為在建立預測定義之後，您無法變更其名稱。
- b. 建立或選取允許與 Amazon Lookout for Equipment AWS IoT SiteWise 共用資產資料的 IAM 許可角色。該角色應具有下列 IAM 和信任政策。如需建立角色的說明，請參閱[使用自訂信任政策建立角色（主控台）](#)。

IAM 政策

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "L4EPermissions",
    "Effect": "Allow",
    "Action": [
      "lookoutequipment:CreateDataset",
      "lookoutequipment:CreateModel",
      "lookoutequipment:CreateInferenceScheduler",
      "lookoutequipment:DescribeDataset",
      "lookoutequipment:DescribeModel",
      "lookoutequipment:DescribeInferenceScheduler",
      "lookoutequipment:ListInferenceExecutions",
      "lookoutequipment:StartDataIngestionJob",
      "lookoutequipment:StartInferenceScheduler",
      "lookoutequipment:UpdateInferenceScheduler",
      "lookoutequipment:StopInferenceScheduler"
    ],
    "Resource": [
      "arn:aws:lookoutequipment:Region:Account_ID:inference-scheduler/IoTSiteWise_*",
      "arn:aws:lookoutequipment:Region:Account_ID:model/IoTSiteWise_*",
      "arn:aws:lookoutequipment:Region:Account_ID:dataset/IoTSiteWise_*"
    ]
  },
  {
    "Sid": "L4EPermissions2",
    "Effect": "Allow",
    "Action": [
      "lookoutequipment:DescribeDataIngestionJob"
    ],
    "Resource": "*"
  }
]}
```

```

    },
    {
      "Sid": "S3Permissions",
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:PutObject",
        "s3:GetObject"
      ],
      "Resource": ["arn:aws:s3:::iotsitewise-*"]
    },
    {
      "Sid": "IAMPermissions",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::Account_ID:role/Role_name"
    }
  ]
}

```

信任政策

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "iotsitewise.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "Account_ID"
      },
      "ArnEquals": {
        "aws:SourceArn":
"arn:aws:iotsitewise:Region:Account_ID:asset/*"
      }
    }
  ]
}

```

```

    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lookoutequipment.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "Account_ID"
        },
        "ArnEquals": {
          "aws:SourceArn":
            "arn:aws:lookoutequipment:Region:Account_ID:*"
        }
      }
    }
  ]
}

```

- c. 選擇 Next (下一步)。
6. 選取您要傳送至 Lookout for Equipment 的資料屬性 (度量、轉換和指標)。
 - a. (選用) 選取度量。
 - b. (選用) 選取轉換。
 - c. (選用) 選取指標。
 - d. 選擇 Next (下一步)。
7. 檢閱您的選擇。若要將預測定義新增至資產模型，請在摘要頁面上選擇新增預測定義。

您也可以編輯或刪除已連接作用中預測的現有預測定義。

訓練預測 (主控台)

將預測定義新增至資產模型後，您可以訓練資產上的預測。

在 中訓練預測 AWS IoT SiteWise

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇資產，然後選取您要監控的資產。
3. 選擇預測。

4. 選取您要訓練的預測。
5. 在動作下，選擇開始訓練，然後執行下列動作：
 - a. 在預測詳細資訊下，選取允許與 Lookout for Equipment AWS IoT SiteWise 共用資產資料的 IAM 許可角色。如果您需要建立新角色，請選擇建立新角色。
 - b. 針對訓練資料設定，輸入訓練資料時間範圍，以選取要用來訓練預測的資料。
 - c. （選用）在後置處理後選取資料的取樣率。
 - d. （選用）對於資料標籤，請提供 Amazon S3 儲存貯體和前綴，以保留標籤資料。如需標記資料的詳細資訊，請參閱《Amazon Lookout for Equipment 使用者指南》中的[標記您的資料](#)。
 - e. 選擇 Next (下一步)。
6. （選用）如果您希望預測在訓練完成後立即啟用，請在進階設定下，選取訓練後自動啟用預測，然後執行下列動作：
 - a. 在輸入資料下，對於資料上傳頻率，定義資料上傳的頻率，對於位移延遲時間，定義要使用的緩衝區數量。
 - b. 選擇 Next (下一步)。
7. 檢閱預測的詳細資訊，然後選擇儲存並開始。

在預測上開始或停止推論（主控台）

Note

Lookout for Equipment 費用適用於排程推論，其中包含在 AWS IoT SiteWise 和 Lookout for Equipment 之間傳輸的資料。如需詳細資訊，請參閱 [Amazon Lookout for Equipment 定價](#)。

如果您新增了預測 `lookoutequipment:CreateDataset`，但未在訓練後選擇啟用它，則必須啟用它以開始監控您的資產。

開始預測的推論

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇資產，然後選取要新增預測的資產。
3. 選擇預測。
4. 選取您要啟用的預測。

5. 在動作下，選擇開始推論，然後執行下列動作：
 - a. 在輸入資料下，對於資料上傳頻率，定義資料上傳的頻率，對於位移延遲時間，定義要使用的緩衝區數量。
 - b. 選擇儲存並開始。

停止預測的推論

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇資產，然後選取要新增預測的資產。
3. 選擇預測。
4. 選取您要停止的預測。
5. 在動作下，選擇停止推論。

新增預測定義 (CLI)

若要在新的或現有的資產模型上定義預測定義，您可以使用 AWS Command Line Interface (AWS CLI)。在資產模型上定義預測定義之後，您可以在 [中訓練和排程資產的預測](#)，AWS IoT SiteWise 以使用 Lookout for Equipment 進行異常偵測。

先決條件

若要完成這些步驟，您必須建立資產模型和至少一個資產。如需詳細資訊，請參閱 [建立資產模型 \(AWS CLI\)](#) 和 [建立資產 \(AWS CLI\)](#)。

如果您是初次使用 AWS IoT SiteWise，您必須呼叫 CreateBulkImportJob API 操作，將資產屬性值匯入到 AWS IoT SiteWise，以用於訓練模型。如需詳細資訊，請參閱 [建立 AWS IoT SiteWise 大量匯入任務 \(AWS CLI\)](#)。

新增預測定義

1. 建立稱為 asset-model-payload.json 的檔案。請依照這些其他章節中的步驟，將資產模型的詳細資訊新增至 檔案，但不要提交建立或更新資產模型的請求。
 - 如需如何建立資產模型的詳細資訊，請參閱 [建立資產模型 \(AWS CLI\)](#)
 - 如需如何更新現有資產模型的詳細資訊，請參閱 [更新資產或元件模型 \(AWS CLI\)](#)
2. 透過新增下列程式碼，將 Lookout for Equipment 複合模型 (assetModelCompositeModels) 新增至資產模型。

- *Property* 以您要包含的屬性 ID 取代。若要取得這些 IDs，請呼叫 [DescribeAssetModel](#)。
- *RoleARN* 以允許 Lookout for Equipment 存取 AWS IoT SiteWise 資料的 IAM 角色 ARN 取代。

```
{
  ...
  "assetModelCompositeModels": [
    {
      "name": "L4Epredictiondefinition",
      "type": "AWS/L4E_ANOMALY",
      "properties": [
        {
          "name": "AWS/L4E_ANOMALY_RESULT",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/L4E_ANOMALY_RESULT",
          "unit": "none",
          "type": {
            "measurement": {}
          }
        },
        {
          "name": "AWS/L4E_ANOMALY_INPUT",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/L4E_ANOMALY_INPUT",
          "type": {
            "attribute": {
              "defaultValue": "{\"properties\": [\"Property1\", \"Property2\"]}"
            }
          }
        },
        {
          "name": "AWS/L4E_ANOMALY_PERMISSIONS",
          "dataType": "STRUCT",
          "dataTypeSpec": "AWS/L4E_ANOMALY_PERMISSIONS",
          "type": {
            "attribute": {
              "defaultValue": "{\"roleArn\": \"RoleARN\"}"
            }
          }
        }
      ]
    }
  ]
}
```



```

        "name": "AWS/L4E_ANOMALY_DATASET",
        "dataType": "STRUCT",
        "dataTypeSpec": "AWS/L4E_ANOMALY_DATASET",
        "type": {
            "attribute": {}
        }
    },
    {
        "name": "AWS/L4E_ANOMALY_MODEL",
        "dataType": "STRUCT",
        "dataTypeSpec": "AWS/L4E_ANOMALY_MODEL",
        "type": {
            "attribute": {}
        }
    },
    {
        "name": "AWS/L4E_ANOMALY_INFERENCE",
        "dataType": "STRUCT",
        "dataTypeSpec": "AWS/L4E_ANOMALY_INFERENCE",
        "type": {
            "attribute": {}
        }
    },
    {
        "name": "AWS/L4E_ANOMALY_TRAINING_STATUS",
        "dataType": "STRUCT",
        "dataTypeSpec": "AWS/L4E_ANOMALY_TRAINING_STATUS",
        "type": {
            "attribute": {
                "defaultValue": "{}"
            }
        }
    },
    {
        "name": "AWS/L4E_ANOMALY_INFERENCE_STATUS",
        "dataType": "STRUCT",
        "dataTypeSpec": "AWS/L4E_ANOMALY_INFERENCE_STATUS",
        "type": {
            "attribute": {
                "defaultValue": "{}"
            }
        }
    }
]

```

```
}
```

3. 建立資產模型或更新現有的資產模型。執行以下任意一項：

- 若要建立資產模型，請執行下列命令：

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-payload.json
```

- 若要更新現有的資產模型，請執行下列命令。*asset-model-id* 以您要更新的資產模型 ID 取代。

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://asset-model-payload.json
```

執行命令之後，請在回應 `assetModelId` 中記下。

訓練預測並開始推論 (CLI)

現在已定義預測定義，您可以根據它來訓練資產並開始推論。如果您想要訓練預測，但未開始推論，請跳至 [訓練預測 \(CLI\)](#)。若要在資產上訓練預測並開始推論，您需要目標資源 `assetId` 的。

訓練並開始推論預測

- 執行下列命令，在 `assetModelCompositeModelId` 下尋找 `assetModelCompositeModelSummaries`。*asset-model-id* 以您在 中建立的資產模型 ID 取代 [更新資產或元件模型 \(AWS CLI\)](#)。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

- 執行下列命令來尋找 `TrainingWithInference動作actionDefinitionId` 的。*asset-model-id* 將取代為上一個步驟中使用的 ID，並將取代 *asset-model-composite-model-id* 為上一個步驟中傳回的 ID。

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. 建立名為 `train-start-inference-prediction.json` 並新增下列程式碼，取代下列程式碼：

- *asset-id* 目標資產的 ID
- *action-definition-id* 搭配 `TrainingWithInference` 動作的 ID
- *StartTime* 訓練資料的開頭，以 epoch 秒提供
- *EndTime* 訓練資料的結尾，以 epoch 秒提供
- *TargetSamplingRate* Lookout for Equipment 處理後的資料取樣率。允許的值為：`PT1S` | `PT5S` | `PT10S` | `PT15S` | `PT30S` | `PT1M` | `PT5M` | `PT10M` | `PT15M` | `PT30M` | `PT1H`。

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-id",
  "actionPayload": {
    "stringValue": "{\"l4ETrainingWithInference\":{\"trainingWithInferenceMode\": \"START\", \"trainingPayload\": {\"exportDataStartTime\": StartTime, \"exportDataEndTime\": EndTime}, \"targetSamplingRate\": \"TargetSamplingRate\"}, \"inferencePayload\": {\"dataDelayOffsetInMinutes\": 0, \"dataUploadFrequency\": \"PT5M\"}}}"
  }
}
```

4. 執行下列命令以開始訓練和推論：

```
aws iotsitewise execute-action --cli-input-json file://train-start-inference-prediction.json
```

訓練預測 (CLI)

現在已定義預測定義，您可以根據它來訓練資產。若要訓練資產的預測，您需要目標資產 `assetId` 的。

訓練預測

1. 執行下列命令，在 `assetModelCompositeModelId` 下尋找 `assetModelCompositeModelSummaries`。*asset-model-id* 以您在 中建立的資產模型 ID 取代 [更新資產或元件模型 \(AWS CLI\)](#)。

```
aws iotsitewise describe-asset-model \  
  --asset-model-id asset-model-id \  
  --asset-model-composite-model-id asset-model-composite-model-id \  
  --output format \  
  --query query
```

2. 執行下列命令來尋找 Training動作 `ActionDefinitionId` 的。*asset-model-id* 將 取代為上一個步驟中使用的 ID，並將 取代 *asset-model-composite-model-id* 為上一個步驟中傳回的 ID。

```
aws iotsitewise describe-asset-model-composite-model \  
  --asset-model-id asset-model-id \  
  --asset-model-composite-model-id asset-model-composite-model-id \  
  --output format \  
  --query query
```

3. 建立名為 的檔案 `train-prediction.json` 並新增下列程式碼，取代下列程式碼：

- *asset-id* 目標資產的 ID
- *action-definition-id* 訓練動作的 ID
- *StartTime* 訓練資料的開頭，以 epoch 秒提供
- *EndTime* 訓練資料的結尾，以 epoch 秒提供
- (選用) *BucketName* 具有存放標籤資料的 Amazon S3 儲存貯體名稱
- (選用) *Prefix* 加上與 Amazon S3 儲存貯體相關聯的字首。
- *TargetSamplingRate* Lookout for Equipment 處理後的資料取樣率。允許的值為：PT1S | PT5S | PT10S | PT15S | PT30S | PT1M | PT5M | PT10M | PT15M | PT30M | PT1H。

Note

同時包含儲存貯體名稱和字首或兩者。

```
{  
  "targetResource": {  
    "assetId": "asset-id"  
  },  
  "trainingData": {  
    "actionDefinitionId": "action-definition-id"  
  },  
  "startTime": epoch-time,  
  "endTime": epoch-time,  
  "bucketName": bucket-name,  
  "prefix": prefix,  
  "targetSamplingRate": target-sampling-rate  
}
```

```
"actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\\"l4ETraining\\": {\\"trainingMode\\":
\\"START\\",\\"exportDataStartTime\\": StartTime, \\"exportDataEndTime\\": EndTime,
\\"targetSamplingRate\\":\\"TargetSamplingRate\\", \\"labelInputConfiguration\\":
{\\"bucketName\\": \\"BucketName\\", \\"prefix\\": \\"Prefix\\"}}}}"
}
```

4. 執行下列命令以開始訓練：

```
aws iotsitewise execute-action --cli-input-json file://train-prediction.json
```

您必須先完成訓練，才能開始推論。若要檢查訓練的狀態，請執行下列其中一項操作：

- 從主控台導覽至預測所在的資產。
- 從中 AWS CLI，BatchGetAssetPropertyValue 使用 trainingStatus 屬性 propertyId 的呼叫。

在預測上啟動或停止推論 (CLI)

訓練預測後，您就可以開始推論，告知 Lookout for Equipment 開始監控您的資產。若要開始或停止推論，您需要目標資源 assetId 的。

開始推論

1. 執行下列命令，在 assetModelCompositeModelId 下尋找 assetModelCompositeModelSummaries。*asset-model-id* 以您在 中建立的資產模型 ID 取代 [更新資產或元件模型 \(AWS CLI\)](#)。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 執行下列命令來尋找 Inference 動作 actionDefinitionId 的。*asset-model-id* 將取代為上一個步驟中使用的 ID，並將 取代 *asset-model-composite-model-id* 為上一個步驟中傳回的 ID。

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. 建立名為 `start-inference.json` 的檔案並新增下列程式碼，取代下列程式碼：

- *asset-id* 目標資產的 ID
- *action-definition-id* 使用啟動推論動作的 ID
- *Offset* 搭配要使用的緩衝區數量
- *Frequency* 資料上傳的頻率

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-id",
  "actionPayload": { "stringValue": "{ \"l4EInference\": { \"inferenceMode\": \"START\", \"dataDelayOffsetInMinutes\": Offset, \"dataUploadFrequency\": \"Frequency\" } }" }
}
```

4. 執行下列命令以開始推論：

```
aws iotsitewise execute-action --cli-input-json file://start-inference.json
```

停止推論

1. 執行下列命令，在 `assetModelCompositeModelId` 下尋找 `assetModelCompositeModelSummaries`。*asset-model-id* 以您在 中建立的資產模型 ID 取代 [更新資產或元件模型 \(AWS CLI\)](#)。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 執行下列命令來尋找 Inference 動作 `actionDefinitionId` 的。*asset-model-id* 將取代為上一個步驟中使用的 ID，並將 *asset-model-composite-model-id* 為上一個步驟中傳回的 ID。

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. 建立名為 `stop-inference.json` 的檔案並新增下列程式碼，取代下列程式碼：

- *asset-id* 目標資產的 ID
- *action-definition-id* 使用啟動推論動作的 ID

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-id",
  "actionPayload": { "stringValue": "{\"l4EIInference\":{\"inferenceMode\":\"STOP\"}}"
  }
}
```

4. 執行下列命令來停止推論：

```
aws iotsitewise execute-action --cli-input-json file://stop-inference.json
```

在 中管理資料儲存 AWS IoT SiteWise

您可以設定 AWS IoT SiteWise 將資料儲存在下列儲存層：

熱層

熱儲存層是 AWS IoT SiteWise 受管時間序列儲存。熱層對經常存取的資料最有效，具有低write-to-read延遲。存放在熱層中的資料會供需要快速存取設備中最新測量值的工業應用程式使用。這包括使用互動式儀表板視覺化即時指標的應用程式，或監控操作和啟動警示以識別效能問題的應用程式。

根據預設，擷取至 的資料 AWS IoT SiteWise 會存放在熱層中。您可以定義熱層的保留期間，之後會根據您的組態，將熱層中的資料 AWS IoT SiteWise 移至暖層或冷層儲存。為了獲得最佳效能和成本效益，請將熱層保留期設定為比經常擷取資料所花費的時間更長。這用於即時指標、警示和監控案例。如果未設定保留期間，您的資料會無限期儲存在熱層中。

暖層

暖儲存層是 AWS IoT SiteWise 受管層，可有效以經濟實惠的方式儲存歷史資料。它最適合用於擷取具有中等write-to-read延遲特性的大量資料。使用暖層來存放大型工作負載所需的歷史資料。例如，它用於分析、商業智慧應用程式 (BI)、報告工具和機器學習 (ML) 模型訓練的資料擷取。如果您啟用冷儲存層，您可以定義暖層保留期。保留期間結束後，會從暖層 AWS IoT SiteWise 刪除資料。

冷層

冷儲存層使用 Amazon S3 儲存貯體來存放很少使用的資料。啟用冷層後，AWS IoT SiteWise 會每 6 小時複寫一次時間序列，包括測量、指標、轉換和彙總，以及資產模型定義。冷層用於儲存資料，以容忍歷史報告和備份的高讀取延遲。

主題

- [在 中設定儲存體設定 AWS IoT SiteWise](#)
- [對 的儲存設定進行故障診斷 AWS IoT SiteWise](#)
- [儲存在冷層中的資料檔案路徑和結構描述](#)

在 中設定儲存體設定 AWS IoT SiteWise

您可以設定儲存設定來選擇加入服務受管暖層儲存，也可以將資料複寫到冷層。若要進一步了解暖層和熱層的保留期間，請參閱 [資料保留影響](#)。設定儲存設定時，請執行下列動作：

- **熱層保留** — 設定熱層中資料在刪除之前存放多久的保留期，並根據儲存設定移至服務受管暖層儲存或冷層儲存。AWS IoT SiteWise 會刪除保留期結束前存在於熱層中的任何資料。如果您未設定保留期間，您的資料將無限期儲存在熱層中。
- **暖層保留** — 設定保留期間，用於將資料從 AWS IoT SiteWise 儲存中刪除並移至客戶受管的冷層儲存之前，在暖層中存放多久。會 AWS IoT SiteWise 刪除保留期間結束前已存在的暖層中的任何資料。如果未設定保留期，您的資料將無限期儲存在暖層中。

Note

若要改善查詢效能，請使用暖層儲存設定熱層保留期間。

熱層和暖層儲存中資料保留的影響

- 當您減少熱層儲存的保留期間時，資料會從熱層永久移至暖層或冷層。當您減少暖層的保留期間時，資料會移至冷層，並從暖層永久刪除。
- 當您增加熱或暖層儲存的保留期間時，變更會影響 AWS IoT SiteWise 從 傳送至 的資料。AWS IoT SiteWise 不會從暖或冷儲存擷取資料以填入熱層。例如，如果熱層儲存的保留期間最初設定為 30 天，然後增加到 60 天，則熱層儲存需要 30 天才能包含 60 天的資料。

主題

- [設定暖層的儲存設定（主控台）](#)
- [設定暖層的儲存設定 \(AWS CLI\)](#)
- [設定冷層的儲存設定（主控台）](#)
- [設定冷層的儲存設定 \(AWS CLI\)](#)

設定暖層的儲存設定（主控台）

下列程序說明如何在 AWS IoT SiteWise 主控台中設定儲存設定，將資料複寫至暖層。

在主控台中設定儲存設定

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中的設定下，選擇儲存。
3. 在右上角，選擇 Edit (編輯)。
4. 在編輯儲存頁面上，執行下列動作：
5. 對於熱層設定，請執行下列動作：
 - 如果您想要設定資料在刪除之前儲存在熱層中的保留期間，並移至服務受管暖層儲存，請選擇啟用保留期間。
 - 若要設定保留期間，請輸入整數並選擇單位。保留期間必須大於或等於 30 天。

AWS IoT SiteWise 會刪除熱層中早於保留期間的任何資料。如果您未設定保留期間，您的資料會無限期儲存。

6. (建議) 針對暖層設定，請執行下列動作：
 - 若要選擇加入暖層儲存，請選取我確認選擇加入暖層儲存，以選擇加入暖層儲存。
 - (選用) 若要設定保留期間，請輸入整數並選擇單位。保留期間必須大於或等於 365 天。

AWS IoT SiteWise 會刪除暖層中比保留期間更早存在的資料。如果您未設定保留期間，您的資料會無限期儲存。

Note

- 當您選擇加入暖層時，組態只會顯示一次。
- 若要設定熱層保留，您必須擁有暖層或冷層儲存。如需成本效益和歷史資料擷取，AWS IoT SiteWise 建議您將長期資料存放在暖層。
- 若要設定暖層保留，您必須擁有冷層儲存。

7. 選擇儲存以儲存儲存設定。

在AWS IoT SiteWise 儲存區段中，暖層儲存處於下列其中一種狀態：

- 已啟用 – 如果您的資料在熱層保留期之前存在，AWS IoT SiteWise 會將資料移至暖層。」
- 停用 – 停用暖層儲存。

設定暖層的儲存設定 (AWS CLI)

您可以使用 AWS CLI 和下列命令，設定儲存設定將資料移至暖層。

若要防止覆寫現有的組態，請執行下列命令來擷取目前的儲存組態資訊：

```
aws iotsitewise describe-storage-configuration
```

Example 沒有現有冷層組態的回應

```
{
  "storageType": "SITEWISE_DEFAULT_STORAGE",
  "disassociatedDataStorage": "ENABLED",
  "configurationStatus": {
    "state": "ACTIVE"
  },
  "lastUpdateDate": "2021-10-14T15:53:35-07:00",
  "warmTier": "DISABLED"
}
```

Example 使用現有冷層組態的 回應

```
{
  "storageType": "MULTI_LAYER_STORAGE",
  "multiLayerStorage": {
    "customerManagedS3Storage": {
      "s3ResourceArn": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/",
      "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
    }
  },
  "disassociatedDataStorage": "ENABLED",
  "retentionPeriod": {
    "numberOfDays": retention-in-days
  },
  "configurationStatus": {
    "state": "ACTIVE"
  },
  "lastUpdateDate": "2023-10-25T15:59:46-07:00",
  "warmTier": "DISABLED"
}
```

使用 設定暖層的儲存設定 AWS CLI

執行下列命令來設定儲存體設定。file-name 將 取代為包含 AWS IoT SiteWise 儲存組態的檔案名稱。

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise 具有熱層和暖層的 組態

```
{
    "storageType": "SITEWISE_DEFAULT_STORAGE",
    "disassociatedDataStorage": "ENABLED",
    "warmTier": "ENABLED",
    "retentionPeriod": {
        "numberOfDays": hot-tier-retention-in-days
    }
}
```

hot-tier-retention-in-days 必須是大於或等於 30 天的整數。

Example response

```
{
    "storageType": "SITEWISE_DEFAULT_STORAGE",
    "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
    }
}
```

如果您已啟用冷層儲存，請參閱 [使用 AWS CLI 和現有的冷方案設定儲存設定](#)。

使用 AWS CLI 和現有的冷方案設定儲存設定

使用 AWS CLI 搭配現有的冷層儲存來設定儲存設定

- 執行下列命令來設定儲存體設定。將 *file-name* 取代為包含 AWS IoT SiteWise 儲存組態的檔案名稱。

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise 儲存體組態

- 將 *amzn-s3-demo-bucket* 取代為您的 Amazon S3 儲存貯體名稱。
- 將 *##* 取代為您的 Amazon S3 字首。
- 將 *aws-account-id* 取代為 AWS 您的帳戶 ID。
- 將 *role-name* 取代為允許 AWS IoT SiteWise 將資料傳送至 Amazon S3 的 Amazon S3 存取角色名稱。
- 以大於或等於 30 天的整數取代 *hot-tier-retention-in-days*-day。
- 以大於或等於 365 天的整數取代 *warm-tier-retention-in-days*-day。

Note

AWS IoT SiteWise 會刪除暖層中早於冷層保留期的任何資料。如果您未設定保留期間，您的資料會無限期儲存。

```
{
  "storageType": "MULTI_LAYER_STORAGE",
  "multiLayerStorage": {
    "customerManagedS3Storage": {
      "s3ResourceArn": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/",
      "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
    }
  },
  "disassociatedDataStorage": "ENABLED",
  "retentionPeriod": {
    "numberOfDays": hot-tier-retention-in-days
  },
  "warmTier": "ENABLED",
  "warmTierRetentionPeriod": {
    "numberOfDays": warm-tier-retention-in-days
  }
}
```

Example response

```
{
```

```
"storageType": "MULTI_LAYER_STORAGE",
"configurationStatus": {
  "state": "UPDATE_IN_PROGRESS"
}
}
```

設定冷層的儲存設定（主控台）

下列程序說明如何在 AWS IoT SiteWise 主控台中設定儲存設定，將資料複寫至冷層。

在主控台中設定儲存設定

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中的設定下，選擇儲存。
3. 在右上角，選擇 Edit (編輯)。
4. 在編輯儲存頁面上，執行下列動作：
 - a. 針對儲存設定，選擇啟用冷層儲存。冷層儲存預設為停用。
 - b. 針對 S3 儲存貯體位置，輸入現有 Amazon S3 儲存貯體的名稱和字首。

Note

- Amazon S3 使用 字首做為 Amazon S3 儲存貯體中的資料夾名稱。字首必須有 1-255 個字元，並以正斜線 (/) 結尾。AWS IoT SiteWise 您的資料會儲存在此資料夾中。
- 如果您沒有 Amazon S3 儲存貯體，請選擇檢視，然後在 Amazon S3 主控台中建立一個。如需詳細資訊，請參閱《Amazon [S3 使用者指南](#)》中的 [建立您的第一個 S3 儲存貯體](#)。Amazon S3

- c. 對於 S3 存取角色，請執行下列其中一項操作：
 - 選擇從 AWS 受管範本建立角色，AWS 自動建立允許 將資料傳送至 Amazon S3 AWS IoT SiteWise 的 IAM 角色。
 - 選擇使用現有角色，然後從清單中選擇您建立的角色。

Note

- 您必須對上一個步驟和 IAM 政策中使用的 S3 儲存貯體位置使用相同的 Amazon S3 儲存貯體名稱。S3
- 請確定您的角色具有下列範例所示的許可。

Example 許可政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:GetBucketLocation",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

將 amzn-s3-demo-bucket 取代為您的 Amazon S3 儲存貯體名稱。

- 如果 Amazon S3 儲存貯體使用客戶受管 KMS 金鑰加密，則 KMS 金鑰必須具有 kms:Decrypt 和 kms:GenerateDataKey 操作的 IAM 角色的存取政策。

- d. 若要設定熱層，請參閱 中的步驟 5 [設定暖層的儲存設定（主控台）](#)。
- e. （選用）對於 AWS IoT Analytics 整合，請執行下列動作。

 Note

終止支援通知：將於 2025 年 12 月 15 日結束對 AWS IoT Analytics 的支援。2025 年 12 月 15 日之後，您將無法再存取 AWS IoT Analytics 主控台或 AWS IoT Analytics 資源。如需詳細資訊，請參閱[AWS IoT Analytics 終止支援](#)。

- i. 如果您想要使用 AWS IoT Analytics 查詢資料，請選擇已啟用 AWS IoT Analytics 的資料存放區。
- ii. AWS IoT SiteWise 會為您的資料存放區產生名稱，或者您可以輸入不同的名稱。

AWS IoT SiteWise 會自動在 中建立資料存放區 AWS IoT Analytics，以儲存您的資料。若要查詢資料，您可以使用 AWS IoT Analytics 來建立資料集。如需詳細資訊，請參閱AWS IoT Analytics 《使用者指南》中的[使用 AWS IoT SiteWise 資料](#)。

- f. 選擇儲存。

在AWS IoT SiteWise 儲存區段中，冷層儲存可以是下列其中一個值：

- 已啟用 – 將您的資料 AWS IoT SiteWise 複寫到指定的 Amazon S3 儲存貯體。
- 啟用 – AWS IoT SiteWise 正在處理您的請求，以啟用冷層儲存。此程序可能需要幾分鐘的時間才能完成。
- Enable_Failed – AWS IoT SiteWise 無法處理您的請求以啟用冷層儲存。如果您啟用 AWS IoT SiteWise 將日誌傳送至 Amazon CloudWatch Logs，您可以使用這些日誌來疑難排解問題。如需詳細資訊，請參閱[使用 Amazon CloudWatch Logs 監控](#)。
- 已停用 – 冷層儲存已停用。

設定冷層的儲存設定 (AWS CLI)

下列程序說明如何設定儲存設定，以使用 將資料複寫至冷層 AWS CLI。

使用 設定儲存體設定 AWS CLI

1. 若要將資料匯出至您帳戶中的 Amazon S3 儲存貯體，請執行下列命令來設定儲存體設定。將 *file-name* 取代為包含 AWS IoT SiteWise 儲存組態的檔案名稱。


```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise 儲存體組態

- 將 *amzn-s3-demo-bucket* 取代為您的 Amazon S3 儲存貯體名稱。
- 將 *##* 取代為您的 Amazon S3 字首。
- 將 *aws-account-id* 取代為 AWS 您的帳戶 ID。
- 將 *role-name* 取代為允許 AWS IoT SiteWise 將資料傳送至 Amazon S3 的 Amazon S3 存取角色名稱。
- 將 *retention-in-days* 替換為整數大於或等於 30 天。

```
{
  "storageType": "MULTI_LAYER_STORAGE",
  "multiLayerStorage": {
    "customerManagedS3Storage": {
      "s3ResourceArn": "arn:aws:s3::amzn-s3-demo-bucket/prefix",
      "roleArn": "arn:aws:iam::aws-account-id:role/role-name"
    }
  },
  "retentionPeriod": {
    "numberOfDays": retention-in-days,
    "unlimited": false
  }
}
```

Note

- 您必須在 AWS IoT SiteWise 儲存組態和 IAM 政策中使用相同的 Amazon S3 儲存貯體名稱。
- 請確定您的角色具有下列範例所示的許可。

Example 許可政策：

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

```

        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:DeleteObject",
            "s3:GetBucketLocation",
            "s3:ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ]
    }
]
}

```

將 `amzn-s3-demo-bucket` 取代為您的 Amazon S3 儲存貯體名稱。

- 如果 Amazon S3 儲存貯體使用客戶受管 KMS 金鑰加密，則 KMS 金鑰必須具有 `kms:Decrypt` 和 `kms:GenerateDataKey` 操作的 IAM 角色的存取政策。

Example response

```

{
  "storageType": "MULTI_LAYER_STORAGE",
  "retentionPeriod": {
    "numberOfDays": 100,
    "unlimited": false
  },
  "configurationStatus": {
    "state": "UPDATE_IN_PROGRESS"
  }
}

```

Note

AWS IoT SiteWise 更新儲存組態可能需要幾分鐘的時間。

2. 若要擷取儲存組態資訊，請執行下列命令。

```
aws iotsitewise describe-storage-configuration
```

Example response

```
{
  "storageType": "MULTI_LAYER_STORAGE",
  "multiLayerStorage": {
    "customerManagedS3Storage": {
      "s3ResourceArn": "arn:aws:s3:::amzn-s3-demo-bucket/torque/",
      "roleArn": "arn:aws:iam::123456789012:role/SWAccessS3Role"
    }
  },
  "retentionPeriod": {
    "numberOfDays": 100,
    "unlimited": false
  },
  "configurationStatus": {
    "state": "ACTIVE"
  },
  "lastUpdateDate": "2021-03-30T15:54:14-07:00"
}
```

- 若要停止將資料匯出至 Amazon S3 儲存貯體，請執行下列命令來設定儲存體設定。

```
aws iotsitewise put-storage-configuration --storage-type SITEWISE_DEFAULT_STORAGE
```

Note

根據預設，您的資料只會儲存在 的熱層中 AWS IoT SiteWise。

Example response

```
{
  "storageType": "SITEWISE_DEFAULT_STORAGE",
  "configurationStatus": {
    "state": "UPDATE_IN_PROGRESS"
  }
}
```

4. 若要擷取儲存組態資訊，請執行下列命令。

```
aws iotsitewise describe-storage-configuration
```

Example response

```
{
  "storageType": "SITEWISE_DEFAULT_STORAGE",
  "configurationStatus": {
    "state": "ACTIVE"
  },
  "lastUpdateDate": "2021-03-30T15:57:14-07:00"
}
```

(選用) 建立 AWS IoT Analytics 資料存放區 (AWS CLI)

Note

終止支援通知：將於 2025 年 12 月 15 日結束對的支援 AWS IoT Analytics。2025 年 12 月 15 日之後，您將無法再存取 AWS IoT Analytics 主控台或 AWS IoT Analytics 資源。如需詳細資訊，請參閱[AWS IoT Analytics 終止支援](#)。

AWS IoT Analytics 資料存放區是可擴展且可查詢的儲存庫，可接收和存放資料。您可以使用 AWS IoT SiteWise 主控台或 AWS IoT Analytics APIs 來建立 AWS IoT Analytics 資料存放區，以儲存 AWS IoT SiteWise 您的資料。若要查詢資料，您可以使用 建立資料集 AWS IoT Analytics。如需詳細資訊，請參閱 AWS IoT Analytics 《使用者指南》中的[使用 AWS IoT SiteWise 資料](#)。

下列步驟使用 在其中 AWS CLI 建立資料存放區 AWS IoT Analytics。

若要建立資料存放區，請執行下列命令。將 *file-name* 取代為包含資料存放區組態的檔案名稱。

```
aws iotanalytics create-datastore --cli-input-json file://file-name.json
```

 Note

- 您必須指定現有 Amazon S3 儲存貯體的名稱。如果您沒有 Amazon S3 儲存貯體，請先建立一個。如需詳細資訊，請參閱《Amazon [S3 使用者指南](#)》中的[建立您的第一個 S3 儲存貯體](#)。Amazon S3
- 您必須在 AWS IoT SiteWise 儲存組態、IAM 政策和 AWS IoT Analytics 資料存放區組態中使用相同的 Amazon S3 儲存貯體名稱。

Example AWS IoT Analytics 資料存放區組態

將 *data-store-name* 和 *amzn-s3-demo-bucket* 取代為您的 AWS IoT Analytics 資料存放區名稱和 Amazon S3 儲存貯體名稱。

```
{
  "datastoreName": "data-store-name",
  "datastoreStorage": {
    "iotSiteWiseMultiLayerStorage": {
      "customerManagedS3Storage": {
        "bucket": "amzn-s3-demo-bucket"
      }
    }
  },
  "retentionPeriod": {
    "numberOfDays": 90
  }
}
```

Example response

```
{
  "datastoreName": "datastore_IoTSiteWise_demo",
  "datastoreArn": "arn:aws:iotanalytics:us-west-2:123456789012:datastore/datastore_IoTSiteWise_demo",
  "retentionPeriod": {
    "numberOfDays": 90,
    "unlimited": false
  }
}
```

對的儲存設定進行故障診斷 AWS IoT SiteWise

使用下列資訊來疑難排解和解決儲存組態的問題。

問題

- [錯誤：儲存貯體不存在](#)
- [錯誤：拒絕存取 Amazon S3 路徑](#)
- [錯誤：無法擔任角色 ARN](#)
- [錯誤：無法存取跨區域 Amazon S3 儲存貯體](#)

錯誤：儲存貯體不存在

Solution： AWS IoT SiteWise 找不到您的 Amazon S3 儲存貯體。請確定您在目前區域中輸入現有 Amazon S3 儲存貯體的名稱。

錯誤：拒絕存取 Amazon S3 路徑

解決方案： AWS IoT SiteWise 無法存取您的 Amazon S3 儲存貯體。請執行下列操作：

- 請務必使用您在 IAM 政策中指定的相同 Amazon S3 儲存貯體。
- 請確定您的角色具有下列範例所示的許可。

Example 許可政策

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject",
        "s3:GetBucketLocation",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
```

```
        "arn:aws:s3:::amzn-s3-demo-bucket/*"  
      ]  
    }  
  ]  
}
```

將 amzn-s3-demo-bucket 取代為您的 Amazon S3 儲存貯體名稱。

錯誤：無法擔任角色 ARN

Solution： AWS IoT SiteWise 無法代表您擔任 IAM 角色。請確定您的角色信任下列服務：`iotsitewise.amazonaws.com`。如需詳細資訊，請參閱 IAM 使用者指南中的 [我無法擔任角色](#)。

錯誤：無法存取跨區域 Amazon S3 儲存貯體

解決方案： 您指定的 Amazon S3 儲存貯體位於不同的 AWS 區域。請確定您的 Amazon S3 儲存貯體和 AWS IoT SiteWise 資產位於相同的區域。

儲存在冷層中的資料檔案路徑和結構描述

AWS IoT SiteWise 透過複寫時間序列，包括測量、指標、轉換和彙總，以及資產和資產模型定義，將您的資料存放在冷層中。以下說明傳送到冷層的資料檔案路徑和結構描述。

主題

- [設備資料（度量）](#)
- [指標、轉換和彙總](#)
- [資產中繼資料](#)
- [資產階層中繼資料](#)
- [儲存資料索引檔案](#)

設備資料（度量）

AWS IoT SiteWise 每六小時將設備資料（度量）匯出至冷層一次。原始資料會以 [Apache AVRO](#) (.avro) 格式儲存在冷層中。

檔案路徑

AWS IoT SiteWise 使用以下範本，將設備資料（度量）存放在冷層中。

```
{keyPrefix}/raw/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/raw_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Amazon S3 中原始資料的每個檔案路徑都包含下列元件。

路徑元件	描述
keyPrefix	您在 AWS IoT SiteWise 儲存組態中指定的 Amazon S3 字首。Amazon S3 使用字首做為儲存貯體中的資料夾名稱。
raw	存放來自設備的時間序列資料的資料夾（度量）。raw 資料夾會儲存在字首資料夾中。
seriesBucket	介於 00 和 ff 之間的十六進位數字。此數字衍生自 timeSeriesId。此分割區用於在 AWS IoT SiteWise 寫入冷層時提高輸送量。當您使用 Amazon Athena 執行查詢時，您可以使用分割區進行精細分割，以改善查詢效能。 seriesBucket 資產中繼資料 timeSeriesBucket 中的 和 是相同的數字。
startYear	與時間序列資料相關聯的專屬開始時間年份。
startMonth	與時間序列資料相關聯的專屬開始時間月份。
startDay	與時間序列資料相關聯的專屬開始時間當月日期。
fileName	檔案名稱使用底線 (_) 字元做為分隔符號來分隔下列項目： - raw 字首。 - timeSeriesId 值。

路徑元件	描述
	- 與時間序列資料相關聯的獨佔開始時間的 epoch 時間戳記。 - 資料的品質。有效值：GOOD、BAD和 UNCERTAIN 。如需詳細資訊，請參閱 AWS IoT SiteWise API 參考中的 AssetPropertyValue。 使用 Snappy 壓縮將檔案儲存為 .avro 格式。

Example 冷層中原始資料的檔案路徑

```
keyPrefix/raw/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
raw_7020c8e2-e6db-40fa-9845-ed0ddddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

欄位

匯出至冷層的原始資料結構描述包含下列欄位。

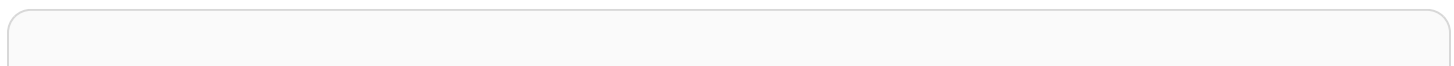
AWS IoT SiteWise 建議客戶在從冷層讀取原始資料的系統上實作結構描述演變的支援，因為未來可能會引入其他欄位。

Null 資料表示為所有值欄位為 Null。不過，使用 AWS IoT SiteWise APIs 查詢時，客戶仍會收到正確的資料類型。

欄位名稱	支援的 類型	預設 類型	描述
seriesId	string	N/A	識別來自設備的時間序列資料的 ID（度量）。您可以使用此欄位，在查詢中加入原始資料和資產中繼資料。
timeInSeconds	long	N/A	時間戳記日期，以秒為單位，以 Unix

欄位名稱	支援的 類型	預設 類型	描述
			epoch 格式顯示。 分數奈秒資料由 提供offsetInNanos。
offsetInNanos	long	N/A	與 的奈秒位 移timeInSeconds。
quality	string	N/A	時間序列值的品質。
doubleValue	double 或 null	null	雙類型的時間序列資料（浮點數）。
stringValue	string 或 null	null	類型字串的時間序列資料（字元序列）。
integerValue	int 或 null	null	整數類型的時間序列資料（整數）。
booleanValue	boolean 或 null	null	布林值 (true 或 false) 類型的時間序列資料。
jsonValue	string 或 null	null	JSON 類型的時間序列資料（儲存為字串的複雜資料類型）。
recordVersion	long 或 null	null	記錄的版本號碼。您可以使用版本號碼來選取最新的記錄。較新的記錄具有較大的版本號碼。

Example 冷層中的原始資料



```
{
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675887,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": {
    "double": 0.75,
    "stringValue": null,
    "integerValue": null,
    "booleanValue": null,
    "jsonValue": null,
    "re": null
  }
},
{
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675889,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": {
    "double": 0.69,
    "stringValue": null,
    "integerValue": null,
    "booleanValue": null,
    "jsonValue": null,
    "re": null
  }
},
{
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675890,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": {
    "double": 0.66,
    "stringValue": null,
    "integerValue": null,
    "booleanValue": null,
    "jsonValue": null,
    "re": null
  }
},
{
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675891,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": {
    "double": 0.92,
    "stringValue": null,
    "integerValue": null,
    "booleanValue": null,
    "jsonValue": null,
    "re": null
  }
},
{
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675892,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": {
    "double": 0.73,
    "stringValue": null,
    "integerValue": null,
    "booleanValue": null,
    "jsonValue": null,
    "re": null
  }
}
```

指標、轉換和彙總

AWS IoT SiteWise 每六小時匯出一次指標、轉換和彙總至冷層。指標、轉換和彙總會以 [Apache AVRO](#) (.avro) 格式儲存在冷層中。

檔案路徑

AWS IoT SiteWise 使用以下範本，將指標、轉換和彙總存放在冷層中。

```
{keyPrefix}/agg/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/agg_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Amazon S3 中指標、轉換和彙總的每個檔案路徑都包含下列元件。

路徑元件	描述
keyPrefix	您在 AWS IoT SiteWise 儲存組態中指定的 Amazon S3 字首。Amazon S3 使用 字首做為儲存貯體中的資料夾名稱。
agg	存放來自 指標的時間序列資料的資料夾。agg 資料夾會儲存在字首資料夾中。
seriesBucket	介於 00 和 ff 之間的十六進位數字。此數字衍生自 timeSeriesId 。此分割區用於在 AWS

路徑元件	描述
	IoT SiteWise 寫入冷層時提高輸送量。當您使用 Amazon Athena 執行查詢時，您可以使用分割區進行精細分割，以改善查詢效能。 <code>seriesBucket</code> 資產中繼資料 <code>timeSeriesBucket</code> 中的 和 是相同的數字。
<code>startYear</code>	與時間序列資料相關聯的專屬開始時間年份。
<code>startMonth</code>	與時間序列資料相關聯的專屬開始時間月份。
<code>startDay</code>	與時間序列資料相關聯的專屬開始時間當月日期。
<code>fileName</code>	檔案名稱使用底線 (_) 字元做為分隔符號來分隔下列項目： - raw 字首。 <code>timeSeriesId</code> 值。 - 與時間序列資料相關聯的獨佔開始時間的 epoch 時間戳記。 - 資料的品質。有效值：GOOD、BAD 和 UNCERTAIN。如需詳細資訊，請參閱《AWS IoT SiteWise API 參考》中的 AssetPropertyValue。 使用 Snappy 壓縮將檔案儲存為 .avro 格式。

Example 冷層中指標的檔案路徑

```
keyPrefix/agg/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/agg_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-bc6f-1b490154b07a_1609577700_G00D.avro
```

欄位

匯出至冷層的指標、轉換和彙總結構描述包含下列欄位。

欄位名稱	支援的 類型	預設 類型	描述
seriesId	string	N/A	識別來自設備、指標或轉換的時間序列資料的 ID。您可以使用此欄位，在查詢中加入原始資料和資產中繼資料。
timeInSeconds	long	N/A	時間戳記日期，以秒為單位，以 Unix epoch 格式顯示。分數奈秒資料由提供offsetInNanos。
offsetInNanos	long	N/A	與 的奈秒位移timeInSeconds。
quality	string	N/A	篩選資產資料的品質。
resolution	string	N/A	要彙總資料的時間間隔。
count	double 或 null	null	目前時間間隔內指定變數的資料點總數。
average	double 或 null	null	目前時間間隔內指定變數值的平均值。
min	double 或 null	null	目前時間間隔內指定變數值的最小值。

欄位名稱	支援的 類型	預設 類型	描述
max	boolean 或 null	null	目前時間間隔內指定變數值的最大值。
sum	string 或 null	null	在目前時間間隔內指定變數值的總和。
recordVersion	long 或 null	null	記錄的版本號碼。您可以使用版本號碼來選取最新的記錄。較新的記錄具有較大的版本號碼。

Example 冷層中的指標資料

```
{
  "seriesId": "f74c2828-5317-4df3-ba16-6d41b5bcb531",
  "timeInSeconds": 1637334060,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "resolution": "1s",
  "double": 16.0,
  "min": {
    "double": 1.0
  },
  "max": {
    "double": 31.0
  },
  "sum": {
    "double": 496.0
  },
  "recordVersion": null
},
{
  "seriesId": "f74c2828-5317-4df3-ba16-6d41b5bcb531",
  "timeInSeconds": 1637334120,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "resolution": "1s",
  "double": 46.0,
  "min": {
    "double": 32.0
  },
  "max": {
    "double": 60.0
  },
  "sum": {
    "double": 1334.0
  },
  "recordVersion": null
},
{
  "seriesId": "f74c2828-5317-4df3-ba16-6d41b5bcb531",
  "timeInSeconds": 1637334540,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "resolution": "1s",
  "double": 16.0,
  "min": {
    "double": 1.0
  },
  "max": {
    "double": 31.0
  },
  "sum": {
    "double": 496.0
  },
  "recordVersion": null
},
{
  "seriesId": "f74c2828-5317-4df3-ba16-6d41b5bcb531",
  "timeInSeconds": 1637334600,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "resolution": "1s",
  "double": 46.0,
  "min": {
    "double": 32.0
  },
  "max": {
    "double": 60.0
  },
  "sum": {
    "double": 1334.0
  },
  "recordVersion": null
},
{
  "seriesId": "f74c2828-5317-4df3-ba16-6d41b5bcb531",
  "timeInSeconds": 1637335020,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "resolution": "1s",
  "double": 16.0,
  "min": {
    "double": 1.0
  },
  "max": {
    "double": 31.0
  },
  "sum": {
    "double": 496.0
  },
  "recordVersion": null
}
```

資產中繼資料

當您第一次啟用 AWS IoT SiteWise 將資料匯出至冷層時，資產中繼資料會匯出至冷層。在初始組態之後，只有在您變更資產模型定義或資產定義時，才會將資產中繼資料 AWS IoT SiteWise 匯出至層。資產中繼資料會以換行分隔 JSON (.ndjson) 格式儲存在冷層中。

檔案路徑

AWS IoT SiteWise 使用以下範本，將資產中繼資料存放在冷層。

```
{keyPrefix}/asset_metadata/asset_{assetId}.ndjson
```

冷層中資產中繼資料的每個檔案路徑都包含下列元件。

路徑元件	描述
keyPrefix	您在 儲存組態中指定的 Amazon S3 AWS IoT SiteWise 字首。Amazon S3 使用 字首做為儲存貯體中的資料夾名稱。
asset_metadata	存放資產中繼資料的資料夾。asset_metadata 資料夾會儲存在字首資料夾中。
fileName	檔案名稱使用底線 (_) 字元做為分隔符號來分隔下列項目： - asset 字首。 - assetId 值。 檔案會以 .ndjson 格式儲存。

Example 較冷層中資產中繼資料的檔案路徑

```
keyPrefix/asset_metadata/asset_35901915-d476-4dca-8637-d9ed4df939ed.ndjson
```

欄位

匯出至冷層的資產中繼資料結構描述包含下列欄位。

欄位名稱	描述
assetId	資產的 ID。
assetName	資產的名稱。
assetExternalId	資產的外部 ID。
assetModelId	用來建立此資產的資產模型 ID。
assetModelName	資產模型的名稱。
assetModelExternalId	資產模型的外部 ID。
assetPropertyId	資產屬性的 ID。
assetPropertyName	資產屬性的名稱。
assetPropertyExternalId	資產屬性的外部 ID。
assetPropertyDataType	資產屬性的資料類型。
assetPropertyUnit	資產屬性的單位 (例如 Newtons 和 RPM)。
assetPropertyAlias	識別資產屬性的別名，例如 OPC UA 伺服器資料串流路徑 (例如 /company/windfarm/3/turbine/7/temperature)。
timeSeriesId	識別來自設備、指標或轉換的時間序列資料的 ID。您可以使用此欄位，在查詢中加入原始資料和資產中繼資料。
timeSeriesBucket	介於 00 和 ff 之間的十六進位數字。此數字衍生自 timeSeriesId 。此分割區用於在 AWS IoT SiteWise 寫入冷層時提高輸送量。當您使用 Amazon Athena 執行查詢時，您可以使用分割區進行精細分割，以改善查詢效能。 timeSeriesBucket 原始資料的檔案路徑 seriesBucket 中的 和 是相同的數字。

欄位名稱	描述
assetCompositeModelId	複合模型的 ID。
assetCompositeModelExternalId	複合模型的外部 ID。
assetCompositeModelDescription	複合模型的描述。
assetCompositeModelName	複合模型的名稱。
assetCompositeModelType	複合模型的類型。對於警示複合模型，類型是 AWS/ALARM 。
assetCreationDate	資產建立的日期，以 Unix epoch 時間表示。
assetLastUpdateDate	資產上次更新的日期，以 Unix epoch 時間表示。
assetStatusErrorCode	錯誤代碼。
assetStatusErrorMessage	錯誤訊息。
assetStatusState	資產的目前狀態。

Example 冷層中的資產中繼資料

```
{
  "assetId": "7020c8e2-e6db-40fa-9845-ed0ddddd4c77d",
  "assetExternalId": null,
  "assetName": "Wind Turbine Asset 2",
  "assetModelId": "ec1d924f-f07d-444f-b072-e2994c165d35",
  "assetModelExternalId": null,
  "assetModelName": "Wind Turbine Asset Model",
  "assetPropertyId": "95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "assetPropertyExternalId": null,
  "assetPropertyName": "Temperature",
  "assetPropertyDescription": "Temperature in Washington/Seattle/WT2/temp",
  "timeSeriesId": "7020c8e2-e6db-40fa-9845-ed0ddddd4c77d_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeSeriesBucket": "f6",
  "assetArn": null,
  "assetCompositeModelDescription": null
}
```

```
{
  "assetId": "7020c8e2-e6db-40fa-9845-ed0ddddd4c77d",
  "assetExternalId": null,
  "assetName": "Wind Turbine Asset 2",
  "assetModelId": "ec1d924f-f07d-444f-b072-e2994c165d35",
  "assetModelExternalId": null,
  "assetModelName": "Wind Turbine Asset Model",
  "assetPropertyId": "c706d54d-4c11-42dc-9a01-63662fc697b4",
  "assetPropertyExternalId": null
}
```

```
Washington/Seattle/WT2/pressure", "timeSeriesId": "7020c8e2-e6db-40fa-9845-  
ed0dddd4c77d_c706d54d-4c11-42dc-9a01-63662fc697b4", "timeSeriesBucket": "1e", "assetArn": null, "asset"  
  {"assetId": "7020c8e2-e6db-40fa-9845-  
ed0dddd4c77d", "assetExternalId": null, "assetName": "Wind Turbine Asset  
  2", "assetModelId": "ec1d924f-f07d-444f-b072-  
e2994c165d35", "assetModelExternalId": null, "assetModelName": "Wind  
  Turbine Asset Model", "assetPropertyId": "8cf1162f-dead-4fbe-b468-  
c8e24cde9f50", "assetPropertyExternalId": null, "assetPropertyName": "Max  
  Temperature", "assetPropertyDataType": "DOUBLE", "assetPropertyUnit": null, "assetPropertyAlias": null,  
e6db-40fa-9845-ed0dddd4c77d_8cf1162f-dead-4fbe-b468-  
c8e24cde9f50", "timeSeriesBucket": "d7", "assetArn": null, "assetCompositeModelDescription": null, "asset"  
  
  {"assetId": "3a5f2a22-3b37-4332-9c1c-404ea1d73fab", "assetExternalId": null, "assetName": "BatchAss  
ebc75e75e827", "assetModelExternalId": null, "assetModelName": "FlashTestAssetModelDouble", "assetPr  
b410-  
ab401a9176ed", "assetPropertyExternalId": null, "assetPropertyName": "measurementProperty", "assetPr  
ae89-  
ff316f5ff8aa", "timeSeriesBucket": "af", "assetArn": null, "assetCompositeModelDescription": null, "asset"
```

資產階層中繼資料

當您啟用 AWS IoT SiteWise 以將資料儲存在冷層時，資產階層中繼資料會匯出至冷層。在初始組態之後，只有在您變更資產模型或資產定義時，才會將資產階層中繼資料 AWS IoT SiteWise 匯出至冷層。資產階層中繼資料會以換行分隔 JSON (.ndjson) 格式儲存在冷層中。

階層、目標資產或來源資產的外部識別符是透過呼叫 [DescribeAsset](#) API 來擷取。

檔案路徑

AWS IoT SiteWise 使用以下範本，將資產階層中繼資料存放在冷層中。

```
{keyPrefix}/asset_hierarchy_metadata/{parentAssetId}_{hierarchyId}.ndjson
```

冷層中資產階層中繼資料的每個檔案路徑都包含下列元件。

路徑元件	描述
keyPrefix	您在 AWS IoT SiteWise 儲存組態中指定的 Amazon S3 字首。Amazon S3 使用 字首做為儲存貯體中的資料夾名稱。
asset_hierarchy_metadata	存放資產階層中繼資料的資料夾。asset_hierarchy_metadata 資料夾會儲存在字首資料夾中。
fileName	檔案名稱使用底線 (_) 字元做為分隔符號來分隔下列項目： - parentAssetId 值。 - hierarchyId 值。 檔案會以 .ndjson 格式儲存。

Example 冷層中資產階層中繼資料的檔案路徑

keyPrefix/asset_hierarchy_metadata/35901915-d476-4dca-8637-d9ed4df939ed_c5b3ced8-589a-48c7-9998-cdcccfc9747a0.ndjson

欄位

匯出至冷層的資產階層中繼資料結構描述包含下列欄位。

欄位名稱	描述
sourceAssetId	此資產關係中來源資產的 ID。
targetAssetId	此資產關係中目標資產的 ID。
hierarchyId	階層的 ID。
associationType	此資產關係的關聯類型。

欄位名稱	描述
	值必須為 CHILD。目標資產是來源資產的子資產。

Example 冷層中的資產階層中繼資料

```
{
  "sourceAssetId": "80388e72-2284-44fb-9c89-bfbaf0dfedd2",
  "targetAssetId": "2b866c25-0c74-4750-bdf5-b73683c8a2a2",
  "hierarchyId": "bbed9f59-0412-4585-a61d-6044db526aee",
  "associationType": "CHILD"
}
{
  "sourceAssetId": "80388e72-2284-44fb-9c89-bfbaf0dfedd2",
  "targetAssetId": "6b51246e-984d-460d-bc0b-470ea47d1e31",
  "hierarchyId": "bbed9f59-0412-4585-a61d-6044db526aee",
  "associationType": "CHILD"
}
```

在冷方案中檢視您的資料

1. 導覽至 [Amazon S3 主控台](#)。
2. 在導覽窗格中，選擇儲存貯體，然後選擇您的 Amazon S3 儲存貯體。
3. 導覽至包含原始資料、資產中繼資料或資產階層中繼資料的資料夾。
4. 選取檔案，然後從動作中選擇下載。

儲存資料索引檔案

AWS IoT SiteWise 使用這些檔案來最佳化資料查詢效能。它們會出現在 Amazon S3 儲存貯體中，但您不需要使用它們。

檔案路徑

AWS IoT SiteWise 使用以下範本將資料索引檔案存放在冷層。

```
keyPrefix/index/series=timeseriesId/startYear=startYear/startMonth=startMonth/
startDay=startDay/index_timeseriesId_startTimestamp_quality
```

Example 資料儲存索引檔案的檔案路徑

```
keyPrefix/index/series=7020c8e2-e6db-40fa-9845-ed0ddddd4c77d_95e63da7-
d34e-43e1-bc6f-1b490154b07a/startYear=2022/startMonth=02/startDay=03/
```

index_7020c8e2-e6db-40fa-9845-ed0ddddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1643846400_G00D

AWS IoT SiteWise 使用 AWS SDKs 程式碼範例

下列程式碼範例示範如何 AWS IoT SiteWise 搭配 AWS 軟體開發套件 (SDK) 使用。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

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中查看內容中的動作。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。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

您好 AWS IoT SiteWise

下列程式碼範例示範如何開始使用 AWS IoT SiteWise。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public class HelloSitewise {
    private static final Logger logger =
        LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
        fetchAssetModels();
    }

    /**
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
            IoTSiteWiseAsyncClient.create();
```

```

        ListAssetModelsRequest assetModelsRequest =
ListAssetModelsRequest.builder()
    .assetModelTypes(AssetModelType.ASSET_MODEL)
    .build();

    // Asynchronous paginator - process paginated results.
    ListAssetModelsPublisher listModelsPaginator =
siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
    CompletableFuture<Void> future = listModelsPaginator.subscribe(response -
> {
        response.assetModelSummaries().forEach(assetSummary ->
            logger.info("Asset Model Name: {} ", assetSummary.name())
        );
    });

    // Wait for the asynchronous operation to complete
    future.join();
}
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [ListAssetModels](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
    paginateListAssetModels,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";

// Call ListDocuments and display the result.
export const main = async () => {
    const client = new IoTSiteWiseClient();
    const listAssetModelsPaginated = [];

```

```
console.log(
  "Hello, AWS Systems Manager! Let's list some of your documents:\n",
);
try {
  // The paginate function is a wrapper around the base command.
  const paginator = paginateListAssetModels({ client }, { maxResults: 5 });
  for await (const page of paginator) {
    listAssetModelsPaginated.push(...page.assetModelSummaries);
  }
} catch (caught) {
  console.error(`There was a problem saying hello: ${caught.message}`);
  throw caught;
}
for (const { name, creationDate } of listAssetModelsPaginated) {
  console.log(`${name} - ${creationDate}`);
}
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [ListAssetModels](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import boto3
```



```
def hello_iot_sitewise(iot_sitewise_client):
    """
    Use the AWS SDK for Python (Boto3) to create an AWS IoT SiteWise
    client and list the asset models in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param iot_sitewise_client: A Boto3 AWS IoT SiteWise Client object. This
    object wraps
                                the low-level AWS IoT SiteWise service API.
    """
    print("Hello, AWS IoT SiteWise! Let's list some of your asset models:\n")
    paginator = iot_sitewise_client.get_paginator("list_asset_models")
    page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

    asset_model_names: [str] = []
    for page in page_iterator:
        for asset_model in page["assetModelSummaries"]:
            asset_model_names.append(asset_model["name"])

    print(f"{len(asset_model_names)} asset model(s) retrieved.")
    for asset_model_name in asset_model_names:
        print(f"\t{asset_model_name}")

if __name__ == "__main__":
    hello_iot_sitewise(boto3.client("iotsitewise"))
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [ListAssetModels](#)。

程式碼範例

- [AWS IoT SiteWise 使用 AWS SDKs 的基本範例](#)
 - [您好 AWS IoT SiteWise](#)
 - [AWS IoT SiteWise 使用 AWS SDK 了解的基本概念](#)
 - [AWS IoT SiteWise 使用 AWS SDKs 的動作](#)
 - [BatchPutAssetPropertyValue 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateAsset 搭配 AWS SDK 或 CLI 使用](#)

- [CreateAssetModel 搭配 AWS SDK 或 CLI 使用](#)
- [CreateGateway 搭配 AWS SDK 或 CLI 使用](#)
- [CreatePortal 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteAsset 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteAssetModel 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteGateway 搭配 AWS SDK 或 CLI 使用](#)
- [DeletePortal 搭配 AWS SDK 或 CLI 使用](#)
- [DescribeAssetModel 搭配 AWS SDK 或 CLI 使用](#)
- [DescribeGateway 搭配 AWS SDK 或 CLI 使用](#)
- [DescribePortal 搭配 AWS SDK 或 CLI 使用](#)
- [GetAssetPropertyValue 搭配 AWS SDK 或 CLI 使用](#)
- [ListAssetModels 搭配 AWS SDK 或 CLI 使用](#)

AWS IoT SiteWise 使用 AWS SDKs 的基本範例

下列程式碼範例示範如何 AWS IoT SiteWise 搭配 AWS SDKs 使用 的基本概念。

範例

- [您好 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 使用 AWS SDK 了解 的基本概念](#)
- [AWS IoT SiteWise 使用 AWS SDKs 的動作](#)
 - [BatchPutAssetPropertyValue 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateAsset 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateAssetModel 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateGateway 搭配 AWS SDK 或 CLI 使用](#)
 - [CreatePortal 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteAsset 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteAssetModel 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteGateway 搭配 AWS SDK 或 CLI 使用](#)
 - [DeletePortal 搭配 AWS SDK 或 CLI 使用](#)
 - [DescribeAssetModel 搭配 AWS SDK 或 CLI 使用](#)

- [DescribeGateway 搭配 AWS SDK 或 CLI 使用](#)
- [DescribePortal 搭配 AWS SDK 或 CLI 使用](#)
- [GetAssetPropertyValue 搭配 AWS SDK 或 CLI 使用](#)
- [ListAssetModels 搭配 AWS SDK 或 CLI 使用](#)

您好 AWS IoT SiteWise

下列程式碼範例示範如何開始使用 AWS IoT SiteWise。

Java

適用於 Java 2.x 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
public class HelloSitewise {
    private static final Logger logger =
        LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
        fetchAssetModels();
    }

    /**
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
            IoTSiteWiseAsyncClient.create();
        ListAssetModelsRequest assetModelsRequest =
            ListAssetModelsRequest.builder()
                .assetModelTypes(AssetModelType.ASSET_MODEL)
                .build();

        // Asynchronous paginator - process paginated results.
        ListAssetModelsPublisher listModelsPaginator =
            siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
    }
}
```

```
CompletableFuture<Void> future = listModelsPaginator.subscribe(response -> {
    response.assetModelSummaries().forEach(assetSummary ->
        logger.info("Asset Model Name: {} ", assetSummary.name())
    );
});

// Wait for the asynchronous operation to complete
future.join();
}
```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [ListAssetModels](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import {
    paginateListAssetModels,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";

// Call ListDocuments and display the result.
export const main = async () => {
    const client = new IoTSiteWiseClient();
    const listAssetModelsPaginated = [];
    console.log(
        "Hello, AWS Systems Manager! Let's list some of your documents:\n",
    );
    try {
        // The paginate function is a wrapper around the base command.
        const paginator = paginateListAssetModels({ client }, { maxResults: 5 });
        for await (const page of paginator) {
            listAssetModelsPaginated.push(...page.assetModelSummaries);
        }
    }
}
```

```
    }  
  } catch (caught) {  
    console.error(`There was a problem saying hello: ${caught.message}`);  
    throw caught;  
  }  
  for (const { name, creationDate } of listAssetModelsPaginated) {  
    console.log(`${name} - ${creationDate}`);  
  }  
};  
  
// Call function if run directly.  
import { fileURLToPath } from "node:url";  
if (process.argv[1] === fileURLToPath(import.meta.url)) {  
  main();  
}
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [ListAssetModels](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import boto3  
  
def hello_iot_sitewise(iot_sitewise_client):  
    """  
    Use the AWS SDK for Python (Boto3) to create an AWS IoT SiteWise  
    client and list the asset models in your account.  
    This example uses the default settings specified in your shared credentials  
    and config files.  
    """
```

```
:param iot_sitewise_client: A Boto3 AWS IoT SiteWise Client object. This
object wraps
                                the low-level AWS IoT SiteWise service API.
"""
print("Hello, AWS IoT SiteWise! Let's list some of your asset models:\n")
paginator = iot_sitewise_client.get_paginator("list_asset_models")
page_iterator = paginator.paginate(PaginationConfig={"MaxItems": 10})

asset_model_names: [str] = []
for page in page_iterator:
    for asset_model in page["assetModelSummaries"]:
        asset_model_names.append(asset_model["name"])

print(f"{len(asset_model_names)} asset model(s) retrieved.")
for asset_model_name in asset_model_names:
    print(f"\t{asset_model_name}")

if __name__ == "__main__":
    hello_iot_sitewise(boto3.client("iotsitewise"))
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [ListAssetModels](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

AWS IoT SiteWise 使用 AWS SDK 了解的基本概念

下列程式碼範例示範如何：

- 建立 AWS IoT SiteWise 資產模型。
- 建立 AWS IoT SiteWise 資產。
- 擷取屬性 ID 值。
- 將資料傳送至 AWS IoT SiteWise 資產。
- 擷取 AWS IoT SiteWise 資產屬性的值。
- 建立 AWS IoT SiteWise 入口網站。
- 建立 AWS IoT SiteWise 閘道。

- 描述 AWS IoT SiteWise 閘道。
- 刪除 AWS IoT SiteWise 資產。

Java

適用於 Java 2.x 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

執行示範 AWS IoT SiteWise 功能的互動式案例。

```
public class SitewiseScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");

    private static final Logger logger =
    LoggerFactory.getLogger(SitewiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static final String ROLES_STACK = "RoleSitewise";

    static SitewiseActions sitewiseActions = new SitewiseActions();

    public static void main(String[] args) throws Throwable {
        Scanner scanner = new Scanner(System.in);
        String contactEmail = "user@mydomain.com"; // Change email address.
        String assetModelName = "MyAssetModel1";
        String assetName = "MyAsset1" ;
        String portalName = "MyPortal1" ;
        String gatewayName = "MyGateway1" ;
        String myThing = "MyThing1" ;

        logger.info("""
            AWS IoT SiteWise is a fully managed software-as-a-service (SaaS)
            that
```

makes it easy to collect, store, organize, and monitor data from industrial equipment and processes.

It is designed to help industrial and manufacturing organizations collect data from their equipment and processes, and use that data to make informed decisions about their operations.

One of the key features of AWS IoT SiteWise is its ability to connect to a wide range of industrial

equipment and systems, including programmable logic controllers (PLCs), sensors, and other

industrial devices. It can collect data from these devices and organize it into a unified data model,

making it easier to analyze and gain insights from the data. AWS IoT SiteWise also provides tools for

visualizing the data, setting up alarms and alerts, and generating reports.

Another key feature of AWS IoT SiteWise is its ability to scale to handle large volumes of data.

It can collect and store data from thousands of devices and process millions of data points per second,

making it suitable for large-scale industrial operations.

Additionally, AWS IoT SiteWise is designed

to be secure and compliant, with features like role-based access controls, data encryption,

and integration with other AWS services for additional security and compliance features.

Let's get started...

```
""");
```

```
waitForInputToContinue(scanner);
```

```
logger.info(DASHES);
```

```
try {
```

```
    runScenario(assetModelName, assetName, portalName, contactEmail, gatewayName, myThing);
```

```
    } catch (RuntimeException e) {
```

```
        logger.info(e.getMessage());
```

```
    }
```

```
}
```



```

    public static void runScenario(String assetModelName, String assetName,
String portalName, String contactEmail, String gatewayName, String myThing)
throws Throwable {
    logger.info("Use AWS CloudFormation to create an IAM role that is
required for this scenario.");
    CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
    Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
    String iamRole = stackOutputs.get("SitewiseRoleArn");
    logger.info("The ARN of the IAM role is {}", iamRole);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("1. Create an AWS SiteWise Asset Model");
    logger.info("""
        An AWS IoT SiteWise Asset Model is a way to represent the physical
assets, such as equipment,
        processes, and systems, that exist in an industrial environment.
This model provides a structured and
        hierarchical representation of these assets, allowing users to
define the relationships and properties
        of each asset.

        This scenario creates two asset model properties: temperature and
humidity.
        """);
    waitForInputToContinue(scanner);
    String assetModelId = null;
    try {
        CreateAssetModelResponse response =
sitewiseActions.createAssetModelAsync(assetModelName).join();
        assetModelId = response.assetModelId();
        logger.info("Asset Model successfully created. Asset Model ID: {}. ",
assetModelId);
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof ResourceAlreadyExistsException) {
            try {
                assetModelId =
sitewiseActions.getAssetModelIdAsync(assetModelName).join();
                logger.info("The Asset Model {} already exists. The id of the
existing model is {}. Moving on...", assetModelName, assetModelId);
            } catch (CompletionException cex) {

```

```

        logger.error("Exception thrown acquiring the asset model id:
{}", cex.getCause().getCause(), cex);
        return;
    }
    } else {
        logger.info("An unexpected error occurred: " +
cause.getMessage(), cause);
        return;
    }
}
waitForInputToContinue(scanner);

logger.info(DASHES);
logger.info("2. Create an AWS IoT SiteWise Asset");
logger.info("""
    The IoT SiteWise model that we just created defines the structure
and metadata for your physical assets.
    Now we create an asset from the asset model.

    """);
logger.info("Let's wait 30 seconds for the asset to be ready.");
countdown(30);
waitForInputToContinue(scanner);
String assetId;
try {
    CreateAssetResponse response =
sitewiseActions.createAssetAsync(assetName, assetModelId).join();
    assetId = response.assetId();
    logger.info("Asset created with ID: {}", assetId);
} catch (CompletionException ce) {
    Throwable cause = ce.getCause();
    if (cause instanceof ResourceNotFoundException) {
        logger.info("The asset model id was not found: {}",
cause.getMessage(), cause);
    } else {
        logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
    }
    return;
}
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);

```

```

        logger.info("3. Retrieve the property ID values");
        logger.info("""
            To send data to an asset, we need to get the property ID values. In
this scenario, we access the
            temperature and humidity property ID values.
            """);
        waitForInputToContinue(scanner);
        Map<String, String> propertyIds = null;
        try {
            propertyIds = sitewiseActions.getPropertyIds(assetModelId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException) {
                logger.error("IoTSiteWiseException occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }
        String humPropId = propertyIds.get("Humidity");
        logger.info("The Humidity property Id is {}", humPropId);
        String tempPropId = propertyIds.get("Temperature");
        logger.info("The Temperature property Id is {}", tempPropId);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Send data to an AWS IoT SiteWise Asset");
        logger.info("""
            By sending data to an IoT SiteWise Asset, you can aggregate data
from
            multiple sources, normalize the data into a standard format, and
store it in a
            centralized location. This makes it easier to analyze and gain
insights from the data.

            In this example, we generate sample temperature and humidity data and
send it to the AWS IoT SiteWise asset.

            """);
        waitForInputToContinue(scanner);

```

```

        try {
            sitewiseActions.sendDataToSiteWiseAsync(assetId, tempPropId,
humPropId).join();
            logger.info("Data sent successfully.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The AWS resource was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Retrieve the value of the IoT SiteWise Asset property");
        logger.info("""
            IoT SiteWise is an AWS service that allows you to collect, process,
and analyze industrial data
            from connected equipment and sensors. One of the key benefits of
reading an IoT SiteWise property
            is the ability to gain valuable insights from your industrial data.

            """);
        waitForInputToContinue(scanner);
        try {
            Double assetVal = sitewiseActions.getAssetPropValueAsync(tempPropId,
assetId).join();
            logger.info("The property name is: {}", "Temperature");
            logger.info("The value of this property is: {}", assetVal);

            waitForInputToContinue(scanner);

            assetVal = sitewiseActions.getAssetPropValueAsync(humPropId,
assetId).join();
            logger.info("The property name is: {}", "Humidity");
            logger.info("The value of this property is: {}", assetVal);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {

```

```

        logger.info("The AWS resource was not found: {}",
cause.getMessage(), cause);
    } else {
        logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
    }
    return;
}
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);
logger.info("6. Create an IoT SiteWise Portal");
logger.info("""
    An IoT SiteWise Portal allows you to aggregate data from multiple
industrial sources,
    such as sensors, equipment, and control systems, into a centralized
platform.
    """);
waitForInputToContinue(scanner);
String portalId;
try {
    portalId = sitewiseActions.createPortalAsync(portalName, iamRole,
contactEmail).join();
    logger.info("Portal created successfully. Portal ID {}", portalId);
} catch (CompletionException ce) {
    Throwable cause = ce.getCause();
    if (cause instanceof IoTSiteWiseException siteWiseEx) {
        logger.error("IoT SiteWise error occurred: Error message: {},
Error code {}",
            siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
    } else {
        logger.error("An unexpected error occurred: {}",
cause.getMessage());
    }
    return;
}
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);
logger.info("7. Describe the Portal");
logger.info("""

```

In this step, we get a description of the portal and display the portal URL.

```
        """);
        waitForInputToContinue(scanner);
        try {
            String portalUrl =
sitewiseActions.describePortalAsync(portalId).join();
            logger.info("Portal URL: {}", portalUrl);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                    notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an IoT SiteWise Gateway");
        logger.info(
            """
```

IoT SiteWise Gateway serves as the bridge between industrial equipment, sensors, and the cloud-based IoT SiteWise service. It is responsible for securely collecting, processing, and transmitting data from various industrial assets to the IoT SiteWise platform, enabling real-time monitoring, analysis, and optimization of industrial operations.

```
        """);
        waitForInputToContinue(scanner);
        String gatewayId = "";
        try {
            gatewayId = sitewiseActions.createGatewayAsync(gatewayName,
myThing).join();
```

```

        logger.info("Gateway creation completed successfully. id is {}",
gatewayId );
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IoTSiteWiseException siteWiseEx) {
            logger.error("IoT SiteWise error occurred: Error message: {},
Error code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage());
        }
        return;
    }
    logger.info(DASHES);
    logger.info(DASHES);

    logger.info("9. Describe the IoT SiteWise Gateway");
    waitForInputToContinue(scanner);
    try {
        sitewiseActions.describeGatewayAsync(gatewayId)
            .thenAccept(response -> {
                logger.info("Gateway Name: {}", response.gatewayName());
                logger.info("Gateway ARN: {}", response.gatewayArn());
                logger.info("Gateway Platform: {}",
response.gatewayPlatform());
                logger.info("Gateway Creation Date: {}",
response.creationDate());
            }).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof ResourceNotFoundException notFoundException) {
            logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), cause);
        }
        return;
    }
    logger.info(DASHES);

```

```

logger.info(DASHES);
logger.info("10. Delete the AWS IoT SiteWise Assets");
logger.info(
    ""
    Before you can delete the Asset Model, you must delete the assets.

    "");
logger.info("Would you like to delete the IoT SiteWise Assets? (y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    logger.info("You selected to delete the SiteWise assets.");
    waitForInputToContinue(scanner);
    try {
        sitewiseActions.deletePortalAsync(portalId).join();
        logger.info("Portal {} was deleted successfully.", portalId);

    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof ResourceNotFoundException notFoundException)
        {
            logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage());
        }
    }

    try {
        sitewiseActions.deleteGatewayAsync(gatewayId).join();
        logger.info("Gateway {} was deleted successfully.", gatewayId);
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof ResourceNotFoundException notFoundException)
        {
            logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
        } else {

```



```

        logger.error("An unexpected error occurred: {}",
cause.getMessage());
    }
}

try {
    sitewiseActions.deleteAssetAsync(assetId).join();
    logger.info("Request to delete asset {} sent successfully",
assetId);
} catch (CompletionException ce) {
    Throwable cause = ce.getCause();
    if (cause instanceof ResourceNotFoundException notFoundException)
{
        logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
    } else {
        logger.error("An unexpected error occurred: {}",
cause.getMessage());
    }
}
logger.info("Let's wait 1 minute for the asset to be deleted.");
countdown(60);
waitForInputToContinue(scanner);
logger.info("Delete the AWS IoT SiteWise Asset Model");
try {
    sitewiseActions.deleteAssetModelAsync(assetModelId).join();
    logger.info("Asset model deleted successfully.");
} catch (CompletionException ce) {
    Throwable cause = ce.getCause();
    if (cause instanceof ResourceNotFoundException notFoundException)
{
        logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
    } else {
        logger.error("An unexpected error occurred: {}",
cause.getMessage());
    }
}
waitForInputToContinue(scanner);

```

```

    } else {
        logger.info("The resources will not be deleted.");
    }
    logger.info(DASHES);

    logger.info(DASHES);
    CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
    logger.info("This concludes the AWS IoT SiteWise Scenario");
    logger.info(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            logger.info("Invalid input. Please try again.");
        }
    }
}

public static void countdown(int totalSeconds) throws InterruptedException {
    for (int i = totalSeconds; i >= 0; i--) {
        int displayMinutes = i / 60;
        int displaySeconds = i % 60;
        System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
        Thread.sleep(1000); // Wait for 1 second
    }
    System.out.println(); // Move to the next line after countdown
    logger.info("Countdown complete!");
}
}

```

SDK AWS IoT SiteWise 方法的包裝函式類別。

```
public class SitewiseActions {
```

```

    private static final Logger logger =
LoggerFactory.getLogger(SitewiseActions.class);

    private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

    private static IoTSiteWiseAsyncClient getAsyncClient() {
        if (ioTSiteWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ioTSiteWiseAsyncClient;
    }

    /**
     * Creates an asset model.
     *
     * @param name the name of the asset model to create.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *
     * <p>
     *
     * If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it

```

```

        *          available to the calling code as a {@link CompletionException}. By
calling
        *          {@link CompletionException#getCause()}, the calling code can
access the original exception.
        */
    public CompletableFuture<CreateAssetModelResponse>
createAssetModelAsync(String name) {
        PropertyType humidity = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        PropertyType temperaturePropertyType = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
            .name("Temperature")
            .dataType(PropertyDataType.DOUBLE)
            .type(temperaturePropertyType)
            .build();

        AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
            .name("Humidity")
            .dataType(PropertyDataType.DOUBLE)
            .type(humidity)
            .build();

        CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
            .assetModelName(name)
            .assetModelDescription("This is my asset model")
            .assetModelProperties(temperatureProperty, humidityProperty)
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }
}

```

```

/**
 * Creates an asset with the specified name and asset model Id.
 *
 * @param assetName    the name of the asset to create.
 * @param assetModelId the Id of the asset model to associate with the asset.
 * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
 *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
 *         {@link CompletableFuture#get()}.
 *
 * <p>
 *
 *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
 *         available to the calling code as a {@link CompletionException}. By
calling
 *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<CreateAssetResponse> createAssetAsync(String
assetName, String assetModelId) {
    CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
        .assetModelId(assetModelId)
        .assetDescription("Created using the AWS SDK for Java")
        .assetName(assetName)
        .build();

    return getAsyncClient().createAsset(createAssetRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
            }
        });
}

/**
 * Sends data to the SiteWise service.
 *
 * @param assetId        the ID of the asset to which the data will be sent.
 * @param tempPropertyId the ID of the temperature property.
 * @param humidityPropId the ID of the humidity property.

```

```

    * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
    *      calling code can attach callbacks, then handle the result or
exception by calling
    *      {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *      <p>
    *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *      available to the calling code as a {@link CompletionException}. By
calling
    *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
    Map<String, Double> sampleData = generateSampleData();
    long timestamp = Instant.now().toEpochMilli();

    TimeInNanos time = TimeInNanos.builder()
        .timeInSeconds(timestamp / 1000)
        .offsetInNanos((int) ((timestamp % 1000) * 1000000))
        .build();

    BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
        .entries(Arrays.asList(
            PutAssetPropertyValueEntry.builder()
                .entryId("entry-3")
                .assetId(assetId)
                .propertyId(tempPropertyId)
                .propertyValues(Arrays.asList(
                    AssetPropertyValue.builder()
                        .value(Variant.builder()
                            .doubleValue(sampleData.get("Temperature"))
                            .build())
                        .timestamp(time)
                        .build()
                ))
                .build(),
            PutAssetPropertyValueEntry.builder()
                .entryId("entry-4")
                .assetId(assetId)

```

```

        .propertyId(humidityPropId)
        .propertyValues(Arrays.asList(
            AssetPropertyValue.builder()
                .value(Variant.builder()
                    .doubleValue(sampleData.get("Humidity"))
                    .build())
                .timestamp(time)
                .build()
        ))
        .build()
    ))
    .build();

    return getAsyncClient().batchPutAssetPropertyValue(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("An exception occurred: {}",
exception.getCause().getMessage());
            }
        });
}

/**
 * Fetches the value of an asset property.
 *
 * @param propId the ID of the asset property to fetch.
 * @param assetId the ID of the asset to fetch the property value for.
 * @return a {@link CompletableFuture} that represents a {@link Double}
result. The calling code can attach
 *      callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
 *      {@link CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
By calling
 *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
    GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()

```

```

        .propertyId(propId)
        .assetId(assetId)
        .build();

return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
    .handle((response, exception) -> {
        if (exception != null) {
            logger.error("Error occurred while fetching property
value: {}.", exception.getCause().getMessage());
            throw (CompletionException) exception;
        }
        return response.propertyValue().value().doubleValue();
    });
}

/**
 * Retrieves the property IDs associated with a specific asset model.
 *
 * @param assetModelId the ID of the asset model that defines the properties.
 * @return a {@link CompletableFuture} that represents a {@link Map} result
that associates the property name to the
 *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
 *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
 *
 * <p>
 *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *         it available to the calling code as a {@link CompletionException}.
By calling
 *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
    ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
    return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
        .handle((response, throwable) -> {
            if (response != null) {
                return response.assetModelPropertySummaries().stream()
                    .collect(Collectors
                        .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
            }
        });
}

```



```

        } else {
            logger.error("Error occurred while fetching property IDs:
{ }.", throwable.getCause().getMessage());
            throw (CompletionException) throwable;
        }
    });
}

/**
 * Deletes an asset.
 *
 * @param assetId the ID of the asset to be deleted.
 * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
 *      attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
 *      {@link CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
By calling
 *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String
assetId) {
    DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
        .assetId(assetId)
        .build();

    return getAsyncClient().deleteAsset(deleteAssetRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("An error occurred deleting asset with id: { }",
assetId);
            }
        });
}

/**
 * Deletes an Asset Model with the specified ID.
 *
 * @param assetModelId the ID of the Asset Model to delete.

```

```

    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<DeleteAssetModelResponse>
deleteAssetModelAsync(String assetModelId) {
        DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
            .assetModelId(assetModelId)
            .build();

        return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
                }
            });
    }

    /**
    * Creates a new IoT SiteWise portal.
    *
    * @param portalName the name of the portal to create.
    * @param iamRole the IAM role ARN to use for the portal.
    * @param contactEmail the email address of the portal contact.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the portal ID. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps

```

```

        *          it available to the calling code as a {@link CompletionException}.
By calling
        *          {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<String> createPortalAsync(String portalName, String
iamRole, String contactEmail) {
        CreatePortalRequest createPortalRequest = CreatePortalRequest.builder()
            .portalName(portalName)
            .portalDescription("This is my custom IoT SiteWise portal.")
            .portalContactEmail(contactEmail)
            .roleArn(iamRole)
            .build();

        return getAsyncClient().createPortal(createPortalRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create portal: {} ",
exception.getCause().getMessage());
                    throw (CompletionException) exception;
                }
                return response.portalId();
            });
    }

    /**
     * Deletes a portal.
     *
     * @param portalId the ID of the portal to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeletePortalResponse}. The calling code can attach
     *          callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *          {@link CompletableFuture#get()}.
     *          <p>
     *          If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *          it available to the calling code as a {@link CompletionException}.
By calling
     *          {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<DeletePortalResponse> deletePortalAsync(String
portalId) {

```

```

        DeletePortalRequest deletePortalRequest = DeletePortalRequest.builder()
            .portalId(portalId)
            .build();

        return getAsyncClient().deletePortal(deletePortalRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete portal with ID: {}. Error:
{}\"", portalId, exception.getCause().getMessage());
                }
            });
    }

    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
     * @return a {@link CompletableFuture} that represents a {@link String}
     result of the asset model ID or null if the
     *     asset model cannot be found. The calling code can attach
     callbacks, then handle the result or exception
     *     by calling {@link CompletableFuture#join()} or {@link
     CompletableFuture#get()}.
     *
     <p>
     *     If any completion stage in this method throws an exception, the
     method logs the exception cause and keeps
     *     it available to the calling code as a {@link CompletionException}.
     By calling
     *     {@link CompletionException#getCause()}, the calling code can
     access the original exception.
     */
    public CompletableFuture<String> getAssetModelIdAsync(String assetModelName)
    {
        ListAssetModelsRequest listAssetModelsRequest =
        ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
            .handle((listAssetModelsResponse, exception) -> {
                if (exception != null) {
                    logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                    throw (CompletionException) exception;
                }
                for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {

```

```

        if (assetModelSummary.name().equals(assetModelName)) {
            return assetModelSummary.id();
        }
    }
    return null;
});
}

/**
 * Retrieves a portal's description.
 *
 * @param portalId the ID of the portal to describe.
 * @return a {@link CompletableFuture} that represents a {@link String}
result of the portal's start URL
 *      (see: {@link DescribePortalResponse#portalStartUrl()}). The
calling code can attach callbacks, then handle the
 *      result or exception by calling {@link CompletableFuture#join()} or
{@link CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
By calling
 *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<String> describePortalAsync(String portalId) {
    DescribePortalRequest request = DescribePortalRequest.builder()
        .portalId(portalId)
        .build();

    return getAsyncClient().describePortal(request)
        .handle((response, exception) -> {
            if (exception != null) {
                logger.error("An exception occurred retrieving the portal
description: {}", exception.getCause().getMessage());
                throw (CompletionException) exception;
            }
            return response.portalStartUrl();
        });
}

/**

```

```

    * Creates a new IoT SiteWise gateway.
    *
    * @param gatewayName The name of the gateway to create.
    * @param myThing      The name of the core device thing to associate with the
gateway.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the gateways ID. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<String> createGatewayAsync(String gatewayName,
String myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest =
CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
            })
    }

```

```

        logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
        return response.gatewayId();
    });
}

/**
 * Deletes the specified gateway.
 *
 * @param gatewayId the ID of the gateway to delete.
 * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
 *      can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
 *      {@link CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
By calling
 *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
    DeleteGatewayRequest deleteGatewayRequest =
DeleteGatewayRequest.builder()
        .gatewayId(gatewayId)
        .build();

    return getAsyncClient().deleteGateway(deleteGatewayRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
            }
        });
}

/**
 * Describes the specified gateway.
 *
 * @param gatewayId the ID of the gateway to describe.

```

```

    * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
                }
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}

```


JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
  //} from "@aws-doc-sdk-examples/lib/scenario/index.js";
} from "../../libs/scenario/index.js";
import {
  IoTSiteWiseClient,
  CreateAssetModelCommand,
  CreateAssetCommand,
  ListAssetModelPropertiesCommand,
  BatchPutAssetPropertyValueCommand,
  GetAssetPropertyValueCommand,
  CreatePortalCommand,
  DescribePortalCommand,
  CreateGatewayCommand,
  DescribeGatewayCommand,
  DeletePortalCommand,
  DeleteGatewayCommand,
  DeleteAssetCommand,
  DeleteAssetModelCommand,
  DescribeAssetModelCommand,
} from "@aws-sdk/client-iotsitewise";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
  waitUntilStackDeleteComplete,
```

```

} from "@aws-sdk/client-cloudformation";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = "SiteWiseBasicsStack";

/**
 * @typedef {{
 *   iotSiteWiseClient: import('@aws-sdk/client-iotsitewise').IotSiteWiseClient,
 *   cloudFormationClient: import('@aws-sdk/client-
cloudformation').CloudFormationClient,
 *   stackName,
 *   stack,
 *   askToDeleteResources: true,
 *   asset: {assetName: "MyAsset1"},
 *   assetModel: {assetModelName: "MyAssetModel1"},
 *   portal: {portalName: "MyPortal1"},
 *   gateway: {gatewayName: "MyGateway1"},
 *   propertyIds: [],
 *   contactEmail: "user@mydomain.com",
 *   thing: "MyThing1",
 *   sampleData: { temperature: 23.5, humidity: 65.0}
 * }} State
 */

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `AWS IoT SiteWise is a fully managed industrial software-as-a-service (SaaS)
that makes it easy to collect, store, organize, and monitor data from industrial
equipment and processes. It is designed to help industrial and manufacturing

```

organizations collect data from their equipment and processes, and use that data to make informed decisions about their operations.

One of the key features of AWS IoT SiteWise is its ability to connect to a wide range of industrial equipment and systems, including programmable logic controllers (PLCs), sensors, and other industrial devices. It can collect data from these devices and organize it into a unified data model, making it easier to analyze and gain insights from the data. AWS IoT SiteWise also provides tools for visualizing the data, setting up alarms and alerts, and generating reports. Another key feature of AWS IoT SiteWise is its ability to scale to handle large volumes of data. It can collect and store data from thousands of devices and process millions of data points per second, making it suitable for large-scale industrial operations. Additionally, AWS IoT SiteWise is designed to be secure and compliant, with features like role-based access controls, data encryption, and integration with other AWS services for additional security and compliance features.

Let's get started...`,

```
    { header: true },
  );

const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "This scenario uses AWS CloudFormation to create an IAM role that is required for this scenario. The stack will now be deployed.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../../resources/cfn/iotsitewise_basics/SitewiseRoles-template.yml`,
        "utf8",
      );
      await state.cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: state.cloudFormationClient },
```

```

        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await state.cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );
      state.stack = stack.Stacks[0].Outputs[0];
      console.log(`The ARN of the IAM role is ${state.stack.OutputValue}`);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateAWSSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSSiteWiseAssetModel",
  `1. Create an AWS SiteWise Asset Model
  An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such
  as equipment, processes, and systems, that exist in an industrial environment.
  This model provides a structured and hierarchical representation of these
  assets, allowing users to define the relationships and properties of each asset.

  This scenario creates two asset model properties: temperature and humidity.`
);

const sdkCreateAWSSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    let assetModelResponse;
    try {
      assetModelResponse = await state.iotSiteWiseClient.send(
        new CreateAssetModelCommand({
          assetModelName: state.assetModel.assetModelName,
          assetModelProperties: [
            {
              name: "Temperature",
              dataType: "DOUBLE",
              type: {

```

```

        measurement: {},
      },
    },
    {
      name: "Humidity",
      dataType: "DOUBLE",
      type: {
        measurement: {},
      },
    },
  ],
}),
);
state.assetModel.assetModelId = assetModelResponse.assetModelId;
console.log(
  `Asset Model successfully created. Asset Model ID:
${state.assetModel.assetModelId}`,
);
} catch (caught) {
  if (caught.name === "ResourceAlreadyExistsException") {
    console.log(
      `The Asset Model ${state.assetModel.assetModelName} already exists.`,
    );
    throw caught;
  }
  console.error(`${caught.message}`);
  throw caught;
}
},
);

const displayCreateAWSIoTSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSIoTSiteWiseAssetModel",
  `2. Create an AWS IoT SiteWise Asset
The IoT SiteWise model that we just created defines the structure and metadata
for your physical assets. Now we create an asset from the asset model.

Let's wait 30 seconds for the asset to be ready.`,
);

const waitThirtySeconds = new ScenarioAction("waitThirtySeconds", async () => {
  await wait(30); // wait 30 seconds
  console.log("Time's up! Let's check the asset's status.");
});

```

```

const sdkCreateAWSIoTSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSIoTSiteWiseAssetModel",
  async (** @type {State} */ state) => {
    try {
      const assetResponse = await state.iotSiteWiseClient.send(
        new CreateAssetCommand({
          assetModelId: state.assetModel.assetModelId,
          assetName: state.asset.assetName,
        })),
      );
      state.asset.assetId = assetResponse.assetId;
      console.log(`Asset created with ID: ${state.asset.assetId}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset ${state.assetModel.assetModelName} was not found.`
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

```

```

const displayRetrievePropertyId = new ScenarioOutput(
  "displayRetrievePropertyId",
  `3. Retrieve the property ID values

```

To send data to an asset, we need to get the property ID values. In this scenario, we access the temperature and humidity property ID values.`

```

);

```

```

const sdkRetrievePropertyId = new ScenarioAction(
  "sdkRetrievePropertyId",
  async (state) => {
    try {
      const retrieveResponse = await state.iotSiteWiseClient.send(
        new ListAssetModelPropertiesCommand({
          assetModelId: state.assetModel.assetModelId,
        })),
      );
    }
  }
);

```

```

        for (const retrieveResponseKey in
retrieveResponse.assetModelPropertySummaries) {
            if (
                retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
                    .name === "Humidity"
            ) {
                state.propertyIds.Humidity =
                    retrieveResponse.assetModelPropertySummaries[
                        retrieveResponseKey
                    ].id;
            }
            if (
                retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
                    .name === "Temperature"
            ) {
                state.propertyIds.Temperature =
                    retrieveResponse.assetModelPropertySummaries[
                        retrieveResponseKey
                    ].id;
            }
        }
        console.log(`The Humidity propertyId is ${state.propertyIds.Humidity}`);
        console.log(
            `The Temperature propertyId is ${state.propertyIds.Temperature}`,
        );
    } catch (caught) {
        if (caught.name === "IoTSiteWiseException") {
            console.log(
                `There was a problem retrieving the properties: ${caught.message}`,
            );
            throw caught;
        }
        console.error(`${caught.message}`);
        throw caught;
    }
},
);

const displaySendDataToIoTSiteWiseAsset = new ScenarioOutput(
    "displaySendDataToIoTSiteWiseAsset",
    `4. Send data to an AWS IoT SiteWise Asset

```

By sending data to an IoT SiteWise Asset, you can aggregate data from multiple sources, normalize the data into a standard format, and store it in a

centralized location. This makes it easier to analyze and gain insights from the data.

In this example, we generate sample temperature and humidity data and send it to the AWS IoT SiteWise asset.`

);

```
const sdkSendDataToIoTSiteWiseAsset = new ScenarioAction(  
  "sdkSendDataToIoTSiteWiseAsset",  
  async (state) => {  
    try {  
      const sendResponse = await state.iotSiteWiseClient.send(  
        new BatchPutAssetPropertyValueCommand({  
          entries: [  
            {  
              entryId: "entry-3",  
              assetId: state.asset.assetId,  
              propertyId: state.propertyIds.Humidity,  
              propertyValues: [  
                {  
                  value: {  
                    doubleValue: state.sampleData.humidity,  
                  },  
                  timestamp: {  
                    timeInSeconds: Math.floor(Date.now() / 1000),  
                  },  
                },  
              ],  
            },  
            {  
              entryId: "entry-4",  
              assetId: state.asset.assetId,  
              propertyId: state.propertyIds.Temperature,  
              propertyValues: [  
                {  
                  value: {  
                    doubleValue: state.sampleData.temperature,  
                  },  
                  timestamp: {  
                    timeInSeconds: Math.floor(Date.now() / 1000),  
                  },  
                },  
              ],  
            },  
          ],  
        })  
      );  
    }  
  }  
);
```



```

    ],
  )),
);
console.log("The data was sent successfully.");
} catch (caught) {
  if (caught.name === "ResourceNotFoundException") {
    console.log(`The Asset ${state.asset.assetName} was not found.`);
    throw caught;
  }
  console.error(`${caught.message}`);
  throw caught;
}
},
);

const displayRetrieveValueOfIoTSiteWiseAsset = new ScenarioOutput(
  "displayRetrieveValueOfIoTSiteWiseAsset",
  `5. Retrieve the value of the IoT SiteWise Asset property

IoT SiteWise is an AWS service that allows you to collect, process, and analyze
industrial data from connected equipment and sensors. One of the key benefits of
reading an IoT SiteWise property is the ability to gain valuable insights from
your industrial data.`
);

const sdkRetrieveValueOfIoTSiteWiseAsset = new ScenarioAction(
  "sdkRetrieveValueOfIoTSiteWiseAsset",
  async (** @type {State} */ state) => {
    try {
      const temperatureResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Temperature,
        }),
      );
      const humidityResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Humidity,
        }),
      );
      console.log(
        `The property value for Temperature is
        ${temperatureResponse.propertyValue.value.doubleValue}`,
      );
    }
  }
);

```

```

    );
    console.log(
      `The property value for Humidity is
    ${humidityResponse.propertyValue.value.doubleValue}`,
    );
  } catch (caught) {
    if (caught.name === "ResourceNotFoundException") {
      console.log(`The Asset ${state.asset.assetName} was not found.`);
      throw caught;
    }
    console.error(`${caught.message}`);
    throw caught;
  }
},
);

```

```

const displayCreateIoTSiteWisePortal = new ScenarioOutput(
  "displayCreateIoTSiteWisePortal",
  `6. Create an IoT SiteWise Portal

```

An IoT SiteWise Portal allows you to aggregate data from multiple industrial sources, such as sensors, equipment, and control systems, into a centralized platform.`,

```

);

```

```

const sdkCreateIoTSiteWisePortal = new ScenarioAction(
  "sdkCreateIoTSiteWisePortal",
  async (** @type {State} */ state) => {
    try {
      const createPortalResponse = await state.iotSiteWiseClient.send(
        new CreatePortalCommand({
          portalName: state.portal.portalName,
          portalContactEmail: state.contactEmail,
          roleArn: state.stack.OutputValue,
        }),
      );
      state.portal = { ...state.portal, ...createPortalResponse };
      await wait(5); // Allow the portal to properly propagate.
      console.log(
        `Portal created successfully. Portal ID
    ${createPortalResponse.portalId}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {

```

```

        console.log(
            `There was a problem creating the Portal: ${caught.message}.`,
        );
        throw caught;
    }
    console.error(`${caught.message}`);
    throw caught;
}
},
);

const displayDescribePortal = new ScenarioOutput(
    "displayDescribePortal",
    `7. Describe the Portal

In this step, we get a description of the portal and display the portal URL.`
);

const sdkDescribePortal = new ScenarioAction(
    "sdkDescribePortal",
    async (** @type {State} */ state) => {
        try {
            const describePortalResponse = await state.iotSiteWiseClient.send(
                new DescribePortalCommand({
                    portalId: state.portal.portalId,
                }),
            );
            console.log(`Portal URL: ${describePortalResponse.portalStartUrl}`);
        } catch (caught) {
            if (caught.name === "ResourceNotFoundException") {
                console.log(`The Portal ${state.portal.portalName} was not found.`);
                throw caught;
            }
            console.error(`${caught.message}`);
            throw caught;
        }
    },
);

const displayCreateIoTSiteWiseGateway = new ScenarioOutput(
    "displayCreateIoTSiteWiseGateway",
    `8. Create an IoT SiteWise Gateway

```

```

IoT SiteWise Gateway serves as the bridge between industrial equipment, sensors,
and the cloud-based IoT SiteWise service. It is responsible for securely
collecting, processing, and transmitting data from various industrial assets
to the IoT SiteWise platform, enabling real-time monitoring, analysis, and
optimization of industrial operations.`
);

const sdkCreateIoTSiteWiseGateway = new ScenarioAction(
  "sdkCreateIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const createGatewayResponse = await state.iotSiteWiseClient.send(
        new CreateGatewayCommand({
          gatewayName: state.gateway.gatewayName,
          gatewayPlatform: {
            greengrassV2: {
              coreDeviceThingName: state.thing,
            },
          },
        })),
      );
      console.log(
        `Gateway creation completed successfully. ID is
        ${createGatewayResponse.gatewayId}`,
      );
      state.gateway.gatewayId = createGatewayResponse.gatewayId;
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the gateway: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribeIoTSiteWiseGateway = new ScenarioOutput(
  "displayDescribeIoTSiteWiseGateway",
  "9. Describe the IoT SiteWise Gateway",
);

```

```

const sdkDescribeIoTSiteWiseGateway = new ScenarioAction(
  "sdkDescribeIoTSiteWiseGateway",
  async (** @type {State} */ state) => {
    try {
      const describeGatewayResponse = await state.iotSiteWiseClient.send(
        new DescribeGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        })),
      );
      console.log("Gateway creation completed successfully.");
      console.log(`Gateway Name: ${describeGatewayResponse.gatewayName}`);
      console.log(`Gateway ARN: ${describeGatewayResponse.gatewayArn}`);
      console.log(
        `Gateway Platform:
${Object.keys(describeGatewayResponse.gatewayPlatform)}`,
      );
      console.log(
        `Gateway Creation Date: ${describeGatewayResponse.creationDate}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
  "askToDeleteResources",
  `10. Delete the AWS IoT SiteWise Assets

Before you can delete the Asset Model, you must delete the assets.`,
  { type: "confirm" },
);

const displayConfirmDeleteResources = new ScenarioAction(
  "displayConfirmDeleteResources",
  async (** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You selected to delete the SiteWise assets.";
    }
  }
);

```

```
        return "The resources will not be deleted. Please delete them manually to  
        avoid charges.";  
    },  
);  
  
const sdkDeleteResources = new ScenarioAction(  
    "sdkDeleteResources",  
    async (** @type {State} */ state) => {  
        await wait(10); // Give the portal status time to catch up.  
        try {  
            await state.iotSiteWiseClient.send(  
                new DeletePortalCommand({  
                    portalId: state.portal.portalId,  
                })),  
            );  
            console.log(  
                `Portal ${state.portal.portalName} was deleted successfully.`,  
            );  
        } catch (caught) {  
            if (caught.name === "ResourceNotFoundException") {  
                console.log(`The Portal ${state.portal.portalName} was not found.`);  
            } else {  
                console.log(`When trying to delete the portal: ${caught.message}`);  
            }  
        }  
    }  
  
    try {  
        await state.iotSiteWiseClient.send(  
            new DeleteGatewayCommand({  
                gatewayId: state.gateway.gatewayId,  
            })),  
        );  
        console.log(  
            `Gateway ${state.gateway.gatewayName} was deleted successfully.`,  
        );  
    } catch (caught) {  
        if (caught.name === "ResourceNotFoundException") {  
            console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);  
        } else {  
            console.log(`When trying to delete the gateway: ${caught.message}`);  
        }  
    }  
}  
  
try {
```

```
    await state.iotSiteWiseClient.send(
      new DeleteAssetCommand({
        assetId: state.asset.assetId,
      }),
    );
    await wait(5); // Allow the delete to finish.
    console.log(`Asset ${state.asset.assetName} was deleted successfully.`);
  } catch (caught) {
    if (caught.name === "ResourceNotFoundException") {
      console.log(`The Asset ${state.asset.assetName} was not found.`);
    } else {
      console.log(`When deleting the asset: ${caught.message}`);
    }
  }
}

await wait(30); // Allow asset deletion to finish.
try {
  await state.iotSiteWiseClient.send(
    new DeleteAssetModelCommand({
      assetModelId: state.assetModel.assetModelId,
    }),
  );
  console.log(
    `Asset Model ${state.assetModel.assetModelName} was deleted
successfully.`,
  );
} catch (caught) {
  if (caught.name === "ResourceNotFoundException") {
    console.log(
      `The Asset Model ${state.assetModel.assetModelName} was not found.`,
    );
  } else {
    console.log(`When deleting the asset model: ${caught.message}`);
  }
}

try {
  await state.cloudFormationClient.send(
    new DeleteStackCommand({
      StackName: stackName,
    }),
  );
  await waitUntilStackDeleteComplete(
    { client: state.cloudFormationClient },
```

```
        { StackName: stackName },
      );
      console.log("The stack was deleted successfully.");
    } catch (caught) {
      console.log(
        `${caught.message}. The stack was NOT deleted. Please clean up the
resources manually.`
      );
    }
  },
  { skipWhen: (/** @type {} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the IoT SiteWise Basics scenario for the AWS Javascript SDK v3.
Thank you!",
);

const myScenario = new Scenario(
  "IoTSiteWise Basics",
  [
    greet,
    pressEnter,
    displayBuildCloudFormationStack,
    sdkBuildCloudFormationStack,
    pressEnter,
    displayCreateAWSSiteWiseAssetModel,
    sdkCreateAWSSiteWiseAssetModel,
    displayCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    waitThirtySeconds,
    sdkCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    displayRetrievePropertyId,
    sdkRetrievePropertyId,
    pressEnter,
    displaySendDataToIoTSiteWiseAsset,
    sdkSendDataToIoTSiteWiseAsset,
    pressEnter,
    displayRetrieveValueOfIoTSiteWiseAsset,
    sdkRetrieveValueOfIoTSiteWiseAsset,
    pressEnter,
    displayCreateIoTSiteWisePortal,
```



```

    sdkCreateIoTSiteWisePortal,
    pressEnter,
    displayDescribePortal,
    sdkDescribePortal,
    pressEnter,
    displayCreateIoTSiteWiseGateway,
    sdkCreateIoTSiteWiseGateway,
    pressEnter,
    displayDescribeIoTSiteWiseGateway,
    sdkDescribeIoTSiteWiseGateway,
    pressEnter,
    askToDeleteResources,
    displayConfirmDeleteResources,
    sdkDeleteResources,
    goodbye,
  ],
  {
    iotSiteWiseClient: new IoTSiteWiseClient({}),
    cloudFormationClient: new CloudFormationClient({}),
    asset: { assetName: "MyAsset1" },
    assetModel: { assetModelName: "MyAssetModel1" },
    portal: { portalName: "MyPortal1" },
    gateway: { gatewayName: "MyGateway1" },
    propertyIds: [],
    contactEmail: "user@mydomain.com",
    thing: "MyThing1",
    sampleData: { temperature: 23.5, humidity: 65.0 },
  },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  },
);

```

```
});  
main({ confirmAll: values.yes });  
}
```

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
class IoTSiteWiseGettingStarted:  
    """  
    A scenario that demonstrates how to use Boto3 to manage IoT physical assets  
    using  
    the AWS IoT SiteWise.  
    """  
  
    def __init__(  
        self,  
        iot_siteWise_wrapper: IoTSiteWiseWrapper,  
        cloud_formation_resource: ServiceResource,  
    ):  
        self.iot_siteWise_wrapper = iot_siteWise_wrapper  
        self.cloud_formation_resource = cloud_formation_resource  
        self.stack = None  
        self.asset_model_id = None  
        self.asset_id = None  
        self.portal_id = None  
        self.gateway_id = None  
  
    def run(self) -> None:  
        """  
        Runs the scenario.  
        """
```

```
print(
    """
AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
makes it easy to collect, store, organize, and monitor data from industrial
equipment and processes.
It is designed to help industrial and manufacturing organizations collect data
from their equipment and
processes, and use that data to make informed decisions about their operations.

One of the key features of AWS IoT SiteWise is its ability to connect to a wide
range of industrial
equipment and systems, including programmable logic controllers (PLCs), sensors,
and other
industrial devices. It can collect data from these devices and organize it into a
unified data model,
making it easier to analyze and gain insights from the data. AWS IoT SiteWise
also provides tools for
visualizing the data, setting up alarms and alerts, and generating reports.

Another key feature of AWS IoT SiteWise is its ability to scale to handle large
volumes of data.
It can collect and store data from thousands of devices and process millions of
data points per second,
making it suitable for large-scale industrial operations. Additionally, AWS IoT
SiteWise is designed
to be secure and compliant, with features like role-based access controls, data
encryption,
and integration with other AWS services for additional security and compliance
features.

Let's get started...
    """
)
press_enter_to_continue()
print_dashes()
print(f"")
print(
    f"Use AWS CloudFormation to create an IAM role that is required for
this scenario."
)
template_file = IoTSiteWiseGettingStarted.get_template_as_string()

self.stack = self.deploy_cloudformation_stack(
    "python-iot-sitewise-basics", template_file
```

```

    )
    outputs = self.stack.outputs
    iam_role = None

    for output in outputs:
        if output.get("OutputKey") == "SitewiseRoleArn":
            iam_role = output.get("OutputValue")

    if iam_role is None:
        error_string = f"Failed to retrieve iam_role from CloudFormation
stack."
        logger.error(error_string)
        raise ValueError(error_string)

    print(f"The ARN of the IAM role is {iam_role}")
    print_dashes()
    print_dashes()
    print(f"1. Create an AWS SiteWise Asset Model")
    print(
        """

```

An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such as equipment, processes, and systems, that exist in an industrial environment. This model provides a structured and hierarchical representation of these assets, allowing users to define the relationships and values of each asset.

This scenario creates two asset model values: temperature and humidity.

```

    """
    )
    press_enter_to_continue()
    asset_model_name = "MyAssetModel1"
    temperature_property_name = "temperature"
    humidity_property_name = "humidity"
    try:
        properties = [
            {
                "name": temperature_property_name,
                "dataType": "DOUBLE",
                "type": {
                    "measurement": {},
                },
            },
        ],
    },

```

```

        {
            "name": humidity_property_name,
            "dataType": "DOUBLE",
            "type": {
                "measurement": {},
            },
        },
    ]
    self.asset_model_id = self.iot_sitewise_wrapper.create_asset_model(
        asset_model_name, properties
    )
    print(
        f"Asset Model successfully created. Asset Model ID:
{self.asset_model_id}. "
    )
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
            self.asset_model_id =
self.get_model_id_for_model_name(asset_model_name)
            print(
                f"Asset Model {asset_model_name} already exists. Asset Model
ID: {self.asset_model_id}. "
            )
        else:
            raise

    press_enter_to_continue()
    print_dashes()
    print(f"2. Create an AWS IoT SiteWise Asset")
    print(
        """

```

The IoT SiteWise model that we just created defines the structure and metadata for your physical assets.

Now we create an asset from the asset model.

```

        """
    )
    press_enter_to_continue()

    self.asset_id = self.iot_sitewise_wrapper.create_asset(
        "MyAsset1", self.asset_model_id
    )

    print(f"Asset created with ID: {self.asset_id}")

```

```

    press_enter_to_continue()
    print_dashes()
    print_dashes()
    print(f"3. Retrieve the property ID values")
    print(
        """

```

To send data to an asset, we need to get the property ID values. In this scenario, we access the temperature and humidity property ID values.

```

        """
    )
    press_enter_to_continue()
    property_ids = self.iot_sitewise_wrapper.list_asset_model_properties(
        self.asset_model_id
    )
    humidity_property_id = None
    temperature_property_id = None
    for property_id in property_ids:
        if property_id.get("name") == humidity_property_name:
            humidity_property_id = property_id.get("id")
        elif property_id.get("name") == temperature_property_name:
            temperature_property_id = property_id.get("id")
    if humidity_property_id is None or temperature_property_id is None:
        error_string = f"Failed to retrieve property IDs from Asset Model."
        logger.error(error_string)
        raise ValueError(error_string)

    print(f"The Humidity property Id is {humidity_property_id}")
    print(f"The Temperature property Id is {temperature_property_id}")
    press_enter_to_continue()
    print_dashes()
    print_dashes()

    print(f"4. Send data to an AWS IoT SiteWise Asset")
    print(
        """

```

By sending data to an IoT SiteWise Asset, you can aggregate data from multiple sources, normalize the data into a standard format, and store it in a centralized location. This makes it easier to analyze and gain insights from the data.

In this example, we generate sample temperature and humidity data and send it to the AWS IoT SiteWise asset.

```

        """
    )
    press_enter_to_continue()

    values = [
        {
            "propertyId": humidity_property_id,
            "valueType": "doubleValue",
            "value": 65.0,
        },
        {
            "propertyId": temperature_property_id,
            "valueType": "doubleValue",
            "value": 23.5,
        },
    ]
    self.iot_sitewise_wrapper.batch_put_asset_property_value(self.asset_id,
values)
    print(f>Data sent successfully.")

    press_enter_to_continue()
    print_dashes()
    print_dashes()

    print(f"5. Retrieve the value of the IoT SiteWise Asset property")
    print(
        """
IoT SiteWise is an AWS service that allows you to collect, process, and analyze
industrial data
from connected equipment and sensors. One of the key benefits of reading an IoT
SiteWise property
is the ability to gain valuable insights from your industrial data.

        """
    )
    press_enter_to_continue()

    property_value = self.iot_sitewise_wrapper.get_asset_property_value(
        self.asset_id, temperature_property_id
    )
    print(f"The property name is '{temperature_property_name}'.")

    print(

```

```

        f"The value of this property is: {property_value['value']}
['doubleValue']}"
    )
    press_enter_to_continue()

    property_value = self.iot_sitewise_wrapper.get_asset_property_value(
        self.asset_id, humidity_property_id
    )
    print(f"The property name is '{humidity_property_name}'.")
    print(
        f"The value of this property is: {property_value['value']}
['doubleValue']}"
    )
    press_enter_to_continue()
    print_dashes()
    print_dashes()

    print(f"6. Create an IoT SiteWise Portal")
    print(
        """
An IoT SiteWise Portal allows you to aggregate data from multiple industrial
sources,
such as sensors, equipment, and control systems, into a centralized platform.
        """
    )

    press_enter_to_continue()
    contact_email = q.ask("Enter a contact email for the portal:",
q.non_empty)
    print("Creating the portal. The portal may take a while to become
active.")
    self.portal_id = self.iot_sitewise_wrapper.create_portal(
        "MyPortal1", iam_role, contact_email
    )
    print(f"Portal created successfully. Portal ID {self.portal_id}")
    press_enter_to_continue()
    print_dashes()
    print_dashes()

    print(f"7. Describe the Portal")
    print(
        """
In this step, we get a description of the portal and display the portal URL.
        """

```



```

    )
    press_enter_to_continue()
    portal_description =
self.iot_sitewise_wrapper.describe_portal(self.portal_id)
    print(f"Portal URL: {portal_description['portalStartUrl']}")
    press_enter_to_continue()
    print_dashes()
    print_dashes()

    print(f"8. Create an IoT SiteWise Gateway")
    press_enter_to_continue()
    self.gateway_id = self.iot_sitewise_wrapper.create_gateway(
        "MyGateway1", "MyThing1"
    )
    print(f"Gateway creation completed successfully. id is
{self.gateway_id}")
    print_dashes()
    print_dashes()
    print(f"9. Describe the IoT SiteWise Gateway")
    press_enter_to_continue()

    gateway_description = self.iot_sitewise_wrapper.describe_gateway(
        self.gateway_id
    )
    print(f"Gateway Name: {gateway_description['gatewayName']}")
    print(f"Gateway ARN: {gateway_description['gatewayArn']}")
    print(f"Gateway Platform:\n{gateway_description['gatewayPlatform']}")
    print(f"Gateway Creation Date: {gateway_description['gatewayArn']}")
    print_dashes()
    print_dashes()

    print(f"10. Delete the AWS IoT SiteWise Assets")
    if q.ask("Would you like to delete the IoT SiteWise Assets? (y/n)",
q.is_yesno):
        self.cleanup()
    else:
        print(f"The resources will not be deleted.")
    print_dashes()
    print_dashes()
    print(f"This concludes the AWS IoT SiteWise Scenario")

def cleanup(self) -> None:
    """
    Deletes the CloudFormation stack and the resources created for the demo.

```

```

"""

if self.gateway_id is not None:
    self.iot_sitewise_wrapper.delete_gateway(self.gateway_id)
    print(f"Deleted gateway with id {self.gateway_id}.")
    self.gateway_id = None
if self.portal_id is not None:
    self.iot_sitewise_wrapper.delete_portal(self.portal_id)
    print(f"Deleted portal with id {self.portal_id}.")
    self.portal_id = None
if self.asset_id is not None:
    self.iot_sitewise_wrapper.delete_asset(self.asset_id)
    print(f"Deleted asset with id {self.asset_id}.")
    self.iot_sitewise_wrapper.wait_asset_deleted(self.asset_id)
    self.asset_id = None
if self.asset_model_id is not None:
    self.iot_sitewise_wrapper.delete_asset_model(self.asset_model_id)
    print(f"Deleted asset model with id {self.asset_model_id}.")
    self.asset_model_id = None
if self.stack is not None:
    stack = self.stack
    self.stack = None
    self.destroy_cloudformation_stack(stack)

def deploy_cloudformation_stack(
    self, stack_name: str, cfn_template: str
) -> ServiceResource:
    """
    Deploys prerequisite resources used by the scenario. The resources are
    defined in the associated `SitewiseRoles-template.yaml` AWS
    CloudFormation script and are deployed
    as a CloudFormation stack, so they can be easily managed and destroyed.

    :param stack_name: The name of the CloudFormation stack.
    :param cfn_template: The CloudFormation template as a string.
    :return: The CloudFormation stack resource.
    """
    print(f"Deploying CloudFormation stack: {stack_name}.")
    stack = self.cloud_formation_resource.create_stack(
        StackName=stack_name,
        TemplateBody=cfn_template,
        Capabilities=["CAPABILITY_NAMED_IAM"],
    )
    print(f"CloudFormation stack creation started: {stack_name}")

```

```

        print("Waiting for CloudFormation stack creation to complete...")
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_create_complete"
        )
        waiter.wait(StackName=stack.name)
        stack.load()
        print("CloudFormation stack creation complete.")

    return stack

def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
    """
    Destroys the resources managed by the CloudFormation stack, and the
    CloudFormation
    stack itself.

    :param stack: The CloudFormation stack that manages the example
    resources.
    """
    print(
        f"CloudFormation stack '{stack.name}' is being deleted. This may take
    a few minutes."
    )
    stack.delete()
    waiter = self.cloud_formation_resource.meta.client.get_waiter(
        "stack_delete_complete"
    )
    waiter.wait(StackName=stack.name)
    print(f"CloudFormation stack '{stack.name}' has been deleted.")

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        template_file_path = os.path.join(script_dir, "SitewiseRoles-
    template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def get_model_id_for_model_name(self, model_name: str) -> str:
        """
        Returns the model ID for the given model name.

```

```

:param model_name: The name of the model.
:return: The model ID.
"""
model_id = None
asset_models = self.iot_sitewise_wrapper.list_asset_models()
for asset_model in asset_models:
    if asset_model["name"] == model_name:
        model_id = asset_model["id"]
        break
return model_id

```

包裝 AWS IoT SiteWise 動作的 IoTSitewiseWrapper 類別。

```

class IoTSitewiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                   access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)

```

```

def create_asset_model(
    self, asset_model_name: str, properties: List[Dict[str, Any]]
) -> str:
    """
    Creates an AWS IoT SiteWise Asset Model.

    :param asset_model_name: The name of the asset model to create.
    :param properties: The property definitions of the asset model.
    :return: The ID of the created asset model.
    """
    try:
        response = self.iotsitewise_client.create_asset_model(
            assetModelName=asset_model_name,
            assetModelDescription="This is a sample asset model
description.",
            assetModelProperties=properties,
        )
        asset_model_id = response["assetModelId"]
        waiter = self.iotsitewise_client.get_waiter("asset_model_active")
        waiter.wait(assetModelId=asset_model_id)
        return asset_model_id
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
            logger.error("Asset model %s already exists.", asset_model_name)
        else:
            logger.error(
                "Error creating asset model %s. Here's why %s",
                asset_model_name,
                err.response["Error"]["Message"],
            )
        raise

def create_asset(self, asset_name: str, asset_model_id: str) -> str:
    """
    Creates an AWS IoT SiteWise Asset.

    :param asset_name: The name of the asset to create.
    :param asset_model_id: The ID of the asset model to associate with the
asset.
    :return: The ID of the created asset.
    """
    try:

```

```

        response = self.iotsitewise_client.create_asset(
            assetName=asset_name, assetModelId=asset_model_id
        )
        asset_id = response["assetId"]
        waiter = self.iotsitewise_client.get_waiter("asset_active")
        waiter.wait(assetId=asset_id)
        return asset_id
    except ClientError as err:
        if err.response["Error"] == "ResourceNotFoundException":
            logger.error("Asset model %s does not exist.", asset_model_id)
        else:
            logger.error(
                "Error creating asset %s. Here's why %s",
                asset_name,
                err.response["Error"]["Message"],
            )
            raise

def list_asset_models(self) -> List[Dict[str, Any]]:
    """
    Lists all AWS IoT SiteWise Asset Models.

    :return: A list of dictionaries containing information about each asset
    model.

    """
    try:
        asset_models = []
        paginator =
self.iotsitewise_client.get_paginator("list_asset_models")
        pages = paginator.paginate()
        for page in pages:
            asset_models.extend(page["assetModelSummaries"])
        return asset_models
    except ClientError as err:
        logger.error(
            "Error listing asset models. Here's why %s",
            err.response["Error"]["Message"],
        )
        raise

```

```

def list_asset_model_properties(self, asset_model_id: str) -> List[Dict[str,
Any]]:
    """
    Lists all AWS IoT SiteWise Asset Model Properties.

    :param asset_model_id: The ID of the asset model to list values for.
    :return: A list of dictionaries containing information about each asset
model property.
    """
    try:
        asset_model_properties = []
        paginator = self.iotsitewise_client.get_paginator(
            "list_asset_model_properties"
        )
        pages = paginator.paginate(assetModelId=asset_model_id)
        for page in pages:
            asset_model_properties.extend(page["assetModelPropertySummaries"])
        return asset_model_properties
    except ClientError as err:
        logger.error(
            "Error listing asset model values. Here's why %s",
            err.response["Error"]["Message"],
        )
        raise

def batch_put_asset_property_value(
    self, asset_id: str, values: List[Dict[str, str]]
) -> None:
    """
    Sends data to an AWS IoT SiteWise Asset.

    :param asset_id: The asset ID.
    :param values: A list of dictionaries containing the values in the form
        {propertyId : property_id,
         valueType : [stringValue|integerValue|doubleValue|
booleanValue],
         value : the_value}.
    """
    try:
        entries = self.properties_to_values(asset_id, values)

        self.iotsitewise_client.batch_put_asset_property_value(entries=entries)

```

```

except ClientError as err:
    if err.response["Error"]["Code"] == "ResourceNotFoundException":
        logger.error("Asset %s does not exist.", asset_id)
    else:
        logger.error(
            "Error sending data to asset. Here's why %s",
            err.response["Error"]["Message"],
        )
    raise

def properties_to_values(
    self, asset_id: str, values: list[dict[str, Any]]
) -> list[dict[str, Any]]:
    """
    Utility function to convert a values list to the entries parameter for
    batch_put_asset_property_value.
    :param asset_id : The asset ID.
    :param values : A list of dictionaries containing the values in the form
        {propertyId : property_id,
         valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                 value : the_value}.
    :return: An entries list to pass as the 'entries' parameter to
    batch_put_asset_property_value.
    """
    entries = []
    for value in values:
        epoch_ns = time.time_ns()
        self.entry_id += 1
        if value["valueType"] == "stringValue":
            property_value = {"stringValue": value["value"]}
        elif value["valueType"] == "integerValue":
            property_value = {"integerValue": value["value"]}
        elif value["valueType"] == "booleanValue":
            property_value = {"booleanValue": value["value"]}
        elif value["valueType"] == "doubleValue":
            property_value = {"doubleValue": value["value"]}
        else:
            raise ValueError("Invalid valueType: %s", value["valueType"])
        entry = {
            "entryId": f"{self.entry_id}",
            "assetId": asset_id,
            "propertyId": value["propertyId"],

```



```

        "propertyValues": [
            {
                "value": property_value,
                "timestamp": {
                    "timeInSeconds": int(epoch_ns / 1000000000),
                    "offsetInNanos": epoch_ns % 1000000000,
                },
            }
        ],
    }
    entries.append(entry)
return entries

def get_asset_property_value(
    self, asset_id: str, property_id: str
) -> Dict[str, Any]:
    """
    Gets the value of an AWS IoT SiteWise Asset Property.

    :param asset_id: The ID of the asset.
    :param property_id: The ID of the property.
    :return: A dictionary containing the value of the property.
    """
    try:
        response = self.iotsitewise_client.get_asset_property_value(
            assetId=asset_id, propertyId=property_id
        )
        return response["propertyValue"]
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error(
                "Asset %s or property %s does not exist.", asset_id,
property_id
            )
        else:
            logger.error(
                "Error getting asset property value. Here's why %s",
                err.response["Error"]["Message"],
            )
        raise

def create_portal(

```

```

        self, portal_name: str, iam_role_arn: str, portal_contact_email: str
    ) -> str:
        """
        Creates an AWS IoT SiteWise Portal.

        :param portal_name: The name of the portal to create.
        :param iam_role_arn: The ARN of an IAM role.
        :param portal_contact_email: The contact email of the portal.
        :return: The ID of the created portal.
        """
        try:
            response = self.iotsitewise_client.create_portal(
                portalName=portal_name,
                roleArn=iam_role_arn,
                portalContactEmail=portal_contact_email,
            )
            portal_id = response["portalId"]
            waiter = self.iotsitewise_client.get_waiter("portal_active")
            waiter.wait(portalId=portal_id, WaiterConfig={"MaxAttempts": 40})
            return portal_id
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
                logger.error("Portal %s already exists.", portal_name)
            else:
                logger.error(
                    "Error creating portal %s. Here's why %s",
                    portal_name,
                    err.response["Error"]["Message"],
                )
            raise

def describe_portal(self, portal_id: str) -> Dict[str, Any]:
    """
    Describes an AWS IoT SiteWise Portal.

    :param portal_id: The ID of the portal to describe.
    :return: A dictionary containing information about the portal.
    """
    try:
        response =
self.iotsitewise_client.describe_portal(portalId=portal_id)
        return response
    except ClientError as err:

```

```

        logger.error(
            "Error describing portal %s. Here's why %s",
            portal_id,
            err.response["Error"]["Message"],
        )
        raise

def create_gateway(self, gateway_name: str, my_thing: str) -> str:
    """
    Creates an AWS IoT SiteWise Gateway.

    :param gateway_name: The name of the gateway to create.
    :param my_thing: The core device thing name.
    :return: The ID of the created gateway.
    """
    try:
        response = self.iotsitewise_client.create_gateway(
            gatewayName=gateway_name,
            gatewayPlatform={
                "greengrassV2": {"coreDeviceThingName": my_thing},
            },
            tags={"Environment": "Production"},
        )
        gateway_id = response["gatewayId"]
        return gateway_id
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
            logger.error("Gateway %s already exists.", gateway_name)
        else:
            logger.error(
                "Error creating gateway %s. Here's why %s",
                gateway_name,
                err.response["Error"]["Message"],
            )
            raise

def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
    """
    Describes an AWS IoT SiteWise Gateway.

    :param gateway_id: The ID of the gateway to describe.
    :return: A dictionary containing information about the gateway.
    """

```

```

    """
    try:
        response =
self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
        return response
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Gateway %s does not exist.", gateway_id)
        else:
            logger.error(
                "Error describing gateway %s. Here's why %s",
                gateway_id,
                err.response["Error"]["Message"],
            )
        raise

def delete_gateway(self, gateway_id: str) -> None:
    """
    Deletes an AWS IoT SiteWise Gateway.

    :param gateway_id: The ID of the gateway to delete.
    """
    try:
        self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Gateway %s does not exist.", gateway_id)
        else:
            logger.error(
                "Error deleting gateway %s. Here's why %s",
                gateway_id,
                err.response["Error"]["Message"],
            )
        raise

def delete_portal(self, portal_id: str) -> None:
    """
    Deletes an AWS IoT SiteWise Portal.

    :param portal_id: The ID of the portal to delete.
    """
    try:

```

```

        self.iotsitewise_client.delete_portal(portalId=portal_id)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Portal %s does not exist.", portal_id)
        else:
            logger.error(
                "Error deleting portal %s. Here's why %s",
                portal_id,
                err.response["Error"]["Message"],
            )
        raise

def delete_asset(self, asset_id: str) -> None:
    """
    Deletes an AWS IoT SiteWise Asset.

    :param asset_id: The ID of the asset to delete.
    """
    try:
        self.iotsitewise_client.delete_asset(assetId=asset_id)
    except ClientError as err:
        logger.error(
            "Error deleting asset %s. Here's why %s",
            asset_id,
            err.response["Error"]["Message"],
        )
        raise

def delete_asset_model(self, asset_model_id: str) -> None:
    """
    Deletes an AWS IoT SiteWise Asset Model.

    :param asset_model_id: The ID of the asset model to delete.
    """
    try:
        self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
    except ClientError as err:
        logger.error(
            "Error deleting asset model %s. Here's why %s",
            asset_model_id,
            err.response["Error"]["Message"],
        )

```

```
    )
    raise

def wait_asset_deleted(self, asset_id: str) -> None:
    """
    Waits for an AWS IoT SiteWise Asset to be deleted.

    :param asset_id: The ID of the asset to wait for.
    """
    try:
        waiter = self.iotsitewise_client.get_waiter("asset_not_exists")
        waiter.wait(assetId=asset_id)
    except ClientError as err:
        logger.error(
            "Error waiting for asset %s to be deleted. Here's why %s",
            asset_id,
            err.response["Error"]["Message"],
        )
    raise
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

AWS IoT SiteWise 使用 AWS SDKs 的動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 AWS IoT SiteWise 動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

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 SiteWise API 參考](#)》。

範例

- [BatchPutAssetPropertyValue 搭配 AWS SDK 或 CLI 使用](#)
- [CreateAsset 搭配 AWS SDK 或 CLI 使用](#)
- [CreateAssetModel 搭配 AWS SDK 或 CLI 使用](#)
- [CreateGateway 搭配 AWS SDK 或 CLI 使用](#)
- [CreatePortal 搭配 AWS SDK 或 CLI 使用](#)

- [DeleteAsset 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteAssetModel 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteGateway 搭配 AWS SDK 或 CLI 使用](#)
- [DeletePortal 搭配 AWS SDK 或 CLI 使用](#)
- [DescribeAssetModel 搭配 AWS SDK 或 CLI 使用](#)
- [DescribeGateway 搭配 AWS SDK 或 CLI 使用](#)
- [DescribePortal 搭配 AWS SDK 或 CLI 使用](#)
- [GetAssetPropertyValue 搭配 AWS SDK 或 CLI 使用](#)
- [ListAssetModels 搭配 AWS SDK 或 CLI 使用](#)

BatchPutAssetPropertyValue 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 BatchPutAssetPropertyValue。

CLI

AWS CLI

將資料傳送至資產屬性

下列batch-put-asset-property-value範例會將電源和溫度資料傳送至屬性別名識別的資產屬性。

```
aws iotsitewise batch-put-asset-property-value \  
  --cli-input-json file://batch-put-asset-property-value.json
```

batch-put-asset-property-value.json 的內容：

```
{  
  "entries": [  
    {  
      "entryId": "1575691200-company-windfarm-3-turbine-7-power",  
      "propertyAlias": "company-windfarm-3-turbine-7-power",  
      "propertyValues": [  
        {  
          "value": {  
            "doubleValue": 4.92
```

```

        },
        "timestamp": {
            "timeInSeconds": 1575691200
        },
        "quality": "GOOD"
    }
]
},
{
    "entryId": "1575691200-company-windfarm-3-turbine-7-temperature",
    "propertyAlias": "company-windfarm-3-turbine-7-temperature",
    "propertyValues": [
        {
            "value": {
                "integerValue": 38
            },
            "timestamp": {
                "timeInSeconds": 1575691200
            }
        }
    ]
}
]
}

```

輸出：

```

{
    "errorEntries": []
}

```

如需詳細資訊，請參閱 IoT [SiteWise 使用者指南中的使用 AWS IoT SiteWise API 擷取資料](#)。
AWS IoT SiteWise

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [BatchPutAssetPropertyValue](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Sends data to the SiteWise service.
 *
 * @param assetId      the ID of the asset to which the data will be sent.
 * @param tempPropertyId the ID of the temperature property.
 * @param humidityPropId the ID of the humidity property.
 * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
 *      calling code can attach callbacks, then handle the result or
exception by calling
 *      {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
 *      available to the calling code as a {@link CompletionException}. By
calling
 *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
    Map<String, Double> sampleData = generateSampleData();
    long timestamp = Instant.now().toEpochMilli();

    TimeInNanos time = TimeInNanos.builder()
        .timeInSeconds(timestamp / 1000)
        .offsetInNanos((int) ((timestamp % 1000) * 1000000))
        .build();
```

```
BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
    .entries(Arrays.asList(
        PutAssetPropertyValueEntry.builder()
            .entryId("entry-3")
            .assetId(assetId)
            .propertyId(tempPropertyId)
            .propertyValues(Arrays.asList(
                AssetPropertyValue.builder()
                    .value(Variant.builder()
                        .doubleValue(sampleData.get("Temperature"))
                        .build())
                    .timestamp(time)
                    .build()
            ))
            .build(),
        PutAssetPropertyValueEntry.builder()
            .entryId("entry-4")
            .assetId(assetId)
            .propertyId(humidityPropId)
            .propertyValues(Arrays.asList(
                AssetPropertyValue.builder()
                    .value(Variant.builder()
                        .doubleValue(sampleData.get("Humidity"))
                        .build())
                    .timestamp(time)
                    .build()
            ))
            .build()
    ))
    .build();

return getAsyncClient().batchPutAssetPropertyValue(request)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            logger.error("An exception occurred: {}",
exception.getCause().getMessage());
        }
    });
}
```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [BatchPutAssetPropertyValue](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import {
  BatchPutAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Batch put asset property values.
 * @param {{ entries : array }}
 */
export const main = async ({ entries }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new BatchPutAssetPropertyValueCommand({
        entries: entries,
      }),
    );
    console.log("Asset properties batch put successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(`${caught.message}. A resource could not be found.`);
    } else {
      throw caught;
    }
  }
};
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [BatchPutAssetPropertyValue](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSiteWiseWrapper":
        """
        Creates an IoTSiteWiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSiteWiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)
```

```

def batch_put_asset_property_value(
    self, asset_id: str, values: List[Dict[str, str]]
) -> None:
    """
    Sends data to an AWS IoT SiteWise Asset.

    :param asset_id: The asset ID.
    :param values: A list of dictionaries containing the values in the form
        {propertyId : property_id,
         valueType : [stringValue|integerValue|doubleValue|
booleanValue],
         value : the_value}.
    """
    try:
        entries = self.properties_to_values(asset_id, values)

    self.iotsitewise_client.batch_put_asset_property_value(entries=entries)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Asset %s does not exist.", asset_id)
        else:
            logger.error(
                "Error sending data to asset. Here's why %s",
                err.response["Error"]["Message"],
            )
        raise

```

從值清單產生項目參數的協助程式函數。

```

def properties_to_values(
    self, asset_id: str, values: list[dict[str, Any]]
) -> list[dict[str, Any]]:
    """
    Utility function to convert a values list to the entries parameter for
    batch_put_asset_property_value.
    :param asset_id : The asset ID.
    :param values : A list of dictionaries containing the values in the form
        {propertyId : property_id,

```

```

        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
        value : the_value}.
    :return: An entries list to pass as the 'entries' parameter to
batch_put_asset_property_value.
"""
entries = []
for value in values:
    epoch_ns = time.time_ns()
    self.entry_id += 1
    if value["valueType"] == "stringValue":
        property_value = {"stringValue": value["value"]}
    elif value["valueType"] == "integerValue":
        property_value = {"integerValue": value["value"]}
    elif value["valueType"] == "booleanValue":
        property_value = {"booleanValue": value["value"]}
    elif value["valueType"] == "doubleValue":
        property_value = {"doubleValue": value["value"]}
    else:
        raise ValueError("Invalid valueType: %s", value["valueType"])
    entry = {
        "entryId": f"{self.entry_id}",
        "assetId": asset_id,
        "propertyId": value["propertyId"],
        "propertyValues": [
            {
                "value": property_value,
                "timestamp": {
                    "timeInSeconds": int(epoch_ns / 1000000000),
                    "offsetInNanos": epoch_ns % 1000000000,
                },
            },
        ],
    }
    entries.append(entry)
return entries

```

以下是要傳遞給協助程式函數的值清單範例。

```

values = [
    {

```

```

        "propertyId": humidity_property_id,
        "valueType": "doubleValue",
        "value": 65.0,
    },
    {
        "propertyId": temperature_property_id,
        "valueType": "doubleValue",
        "value": 23.5,
    },
]

```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [BatchPutAssetPropertyValue](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

CreateAsset 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateAsset。

CLI

AWS CLI

建立資產

下列create-asset範例會從風力渦輪機資產模型建立風力渦輪機資產。

```

aws iotsitewise create-asset \
  --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
  --asset-name "Wind Turbine 1"

```

輸出：

```

{
  "assetId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
  "assetArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
  "assetStatus": {
    "state": "CREATING"
  }
}

```

```
}
}
```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[建立資產](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[CreateAsset](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```
/**
 * Creates an asset with the specified name and asset model Id.
 *
 * @param assetName    the name of the asset to create.
 * @param assetModelId the Id of the asset model to associate with the asset.
 * @return a {@link CompletableFuture} that represents a {@link
 * CreateAssetResponse} result. The calling code can
 *      * attach callbacks, then handle the result or exception by calling
 *      * {@link CompletableFuture#join()} or
 *      * {@link CompletableFuture#get()}.
 *      * <p>
 *      * If any completion stage in this method throws an exception, the
 *      * method logs the exception cause and keeps it
 *      * available to the calling code as a {@link CompletionException}. By
 *      * calling
 *      * {@link CompletionException#getCause()}, the calling code can
 *      * access the original exception.
 */
public CompletableFuture<CreateAssetResponse> createAssetAsync(String
assetName, String assetModelId) {
    CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
        .assetModelId(assetModelId)
        .assetDescription("Created using the AWS SDK for Java")
        .assetName(assetName)
```



```

        .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [CreateAsset](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
    CreateAssetCommand,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetName, assetModelId }) => {
    const client = new IoTSiteWiseClient({});
    try {
        const result = await client.send(
            new CreateAssetCommand({
                assetName: assetName, // The name to give the Asset.
                assetModelId: assetModelId, // The ID of the asset model from which to
                create the asset.
            })
        );
    } catch (error) {
        console.error(error);
    }
};

```

```

    })),
    );
    console.log("Asset created successfully.");
    return result;
} catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
        console.warn(
            `${caught.message}. The asset model could not be found. Please check the
            asset model id.`);
    };
    } else {
        throw caught;
    }
}
};

```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [CreateAsset](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """

```

```

        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)

    def create_asset(self, asset_name: str, asset_model_id: str) -> str:
        """
        Creates an AWS IoT SiteWise Asset.

        :param asset_name: The name of the asset to create.
        :param asset_model_id: The ID of the asset model to associate with the
asset.

        :return: The ID of the created asset.
        """
        try:
            response = self.iotsitewise_client.create_asset(
                assetName=asset_name, assetModelId=asset_model_id
            )
            asset_id = response["assetId"]
            waiter = self.iotsitewise_client.get_waiter("asset_active")
            waiter.wait(assetId=asset_id)
            return asset_id
        except ClientError as err:
            if err.response["Error"] == "ResourceNotFoundException":
                logger.error("Asset model %s does not exist.", asset_model_id)
            else:
                logger.error(
                    "Error creating asset %s. Here's why %s",
                    asset_name,
                    err.response["Error"]["Message"],
                )
            raise

```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [CreateAsset](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

CreateAssetModel 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateAssetModel。

CLI

AWS CLI

建立資產模型

下列 create-asset-model 範例會建立資產模型，以定義具有下列屬性的風力渦輪機：

序號 - 風 turbineGenerated 功率 - 從風 turbineTemperature C - 從風力發電機以 CelsiusTemperature F 產生的溫度資料串流 - 從攝氏到華氏的映射溫度資料點

```
aws iotsitewise create-asset-model \  
  --cli-input-json file://create-wind-turbine-model.json
```

create-wind-turbine-model.json 的內容：

```
{  
  "assetModelName": "Wind Turbine Model",  
  "assetModelDescription": "Represents a wind turbine",  
  "assetModelProperties": [  
    {  
      "name": "Serial Number",  
      "dataType": "STRING",  
      "type": {  
        "attribute": {}  
      }  
    },  
    {  
      "name": "Generated Power",
```

```

        "dataType": "DOUBLE",
        "unit": "kW",
        "type": {
            "measurement": {}
        }
    },
    {
        "name": "Temperature C",
        "dataType": "DOUBLE",
        "unit": "Celsius",
        "type": {
            "measurement": {}
        }
    },
    {
        "name": "Temperature F",
        "dataType": "DOUBLE",
        "unit": "Fahrenheit",
        "type": {
            "transform": {
                "expression": "temp_c * 9 / 5 + 32",
                "variables": [
                    {
                        "name": "temp_c",
                        "value": {
                            "propertyId": "Temperature C"
                        }
                    }
                ]
            }
        }
    },
    {
        "name": "Total Generated Power",
        "dataType": "DOUBLE",
        "unit": "kW",
        "type": {
            "metric": {
                "expression": "sum(power)",
                "variables": [
                    {
                        "name": "power",
                        "value": {
                            "propertyId": "Generated Power"
                        }
                    }
                ]
            }
        }
    }
]

```

```

    }
  },
  "window": {
    "tumbling": {
      "interval": "1h"
    }
  }
}
]
}

```

輸出：

```

{
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetModelStatus": {
    "state": "CREATING"
  }
}

```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[定義資產模型](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[CreateAssetModel](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```

/**
 * Creates an asset model.

```

```

*
* @param name the name of the asset model to create.
* @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
*       can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
*       {@link CompletableFuture#get()}.
*       <p>
*       If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
*       available to the calling code as a {@link CompletionException}. By
calling
*       {@link CompletionException#getCause()}, the calling code can
access the original exception.
*/
public CompletableFuture<CreateAssetModelResponse>
createAssetModelAsync(String name) {
    PropertyType humidity = PropertyType.builder()
        .measurement(Measurement.builder().build())
        .build();

    PropertyType temperaturePropertyType = PropertyType.builder()
        .measurement(Measurement.builder().build())
        .build();

    AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
        .name("Temperature")
        .dataType(PropertyDataType.DOUBLE)
        .type(temperaturePropertyType)
        .build();

    AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
        .name("Humidity")
        .dataType(PropertyDataType.DOUBLE)
        .type(humidity)
        .build();

    CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
        .assetModelName(name)
        .assetModelDescription("This is my asset model")
        .assetModelProperties(temperatureProperty, humidityProperty)

```

```

        .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
                        exception.getCause().getMessage());
                }
            });
    }
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [CreateAssetModel](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
    CreateAssetModelCommand,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an Asset Model.
 * @param {{ assetName : string, assetModelId: string }}
 */
export const main = async ({ assetModelName, assetModelId }) => {
    const client = new IoTSiteWiseClient({});
    try {
        const result = await client.send(
            new CreateAssetModelCommand({
                assetModelName: assetModelName, // The name to give the Asset Model.
            }),
        );
    }
};

```



```

    console.log("Asset model created successfully.");
    return result;
} catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
        console.warn(
            `${caught.message}. There was a problem creating the asset model.`
        );
    } else {
        throw caught;
    }
}
};

```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [CreateAssetModel](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

```

```

@classmethod
def from_client(cls) -> "IoTSitewiseWrapper":
    """
    Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
    client.

    :return: An instance of IoTSitewiseWrapper initialized with the default
    AWS IoT SiteWise client.
    """
    iotsitewise_client = boto3.client("iotsitewise")
    return cls(iotsitewise_client)

def create_asset_model(
    self, asset_model_name: str, properties: List[Dict[str, Any]]
) -> str:
    """
    Creates an AWS IoT SiteWise Asset Model.

    :param asset_model_name: The name of the asset model to create.
    :param properties: The property definitions of the asset model.
    :return: The ID of the created asset model.
    """
    try:
        response = self.iotsitewise_client.create_asset_model(
            assetModelName=asset_model_name,
            assetModelDescription="This is a sample asset model
description.",
            assetModelProperties=properties,
        )
        asset_model_id = response["assetModelId"]
        waiter = self.iotsitewise_client.get_waiter("asset_model_active")
        waiter.wait(assetModelId=asset_model_id)
        return asset_model_id
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
            logger.error("Asset model %s already exists.", asset_model_name)
        else:
            logger.error(
                "Error creating asset model %s. Here's why %s",
                asset_model_name,
                err.response["Error"]["Message"],
            )

```

```
raise
```

以下是要傳遞至 函數的屬性清單範例。

```
properties = [
    {
        "name": temperature_property_name,
        "dataType": "DOUBLE",
        "type": {
            "measurement": {},
        },
    },
    {
        "name": humidity_property_name,
        "dataType": "DOUBLE",
        "type": {
            "measurement": {},
        },
    },
]
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [CreateAssetModel](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

CreateGateway 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateGateway。

CLI

AWS CLI

建立閘道

下列create-gateway範例會建立在 AWS IoT Greengrass 上執行的閘道。

```
aws iotsitewise create-gateway \
  --gateway-name ExampleCorpGateway \
  --gateway-platform greengrass={groupArn=arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/a1b2c3d4-5678-90ab-cdef-1b1b1EXAMPLE}
```

輸出：

```
{
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
}
```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[設定閘道](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CreateGateway](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Creates a new IoT Sitewise gateway.
 *
 * @param gatewayName The name of the gateway to create.
 * @param myThing      The name of the core device thing to associate with the
gateway.
 * @return a {@link CompletableFuture} that represents a {@link String}
result of the gateways ID. The calling code
 *          can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
 *          {@link CompletableFuture#get()}.
 *
 <p>
```

```

    *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *      it available to the calling code as a {@link CompletionException}.
By calling
    *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<String> createGatewayAsync(String gatewayName,
String myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest =
CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [CreateGateway](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import {
  CreateGatewayCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create a Gateway.
 * @param {{ }}
 */
export const main = async ({ gatewayName }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new CreateGatewayCommand({
        gatewayName: gatewayName, // The name to give the created Gateway.
      }),
    );
    console.log("Gateway created successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem creating the Gateway.`
      );
    } else {
      throw caught;
    }
  }
};
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [CreateGateway](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSiteWiseWrapper":
        """
        Creates an IoTSiteWiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSiteWiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)
```

```
def create_gateway(self, gateway_name: str, my_thing: str) -> str:
    """
    Creates an AWS IoT SiteWise Gateway.

    :param gateway_name: The name of the gateway to create.
    :param my_thing: The core device thing name.
    :return: The ID of the created gateway.
    """
    try:
        response = self.iotsitewise_client.create_gateway(
            gatewayName=gateway_name,
            gatewayPlatform={
                "greengrassV2": {"coreDeviceThingName": my_thing},
            },
            tags={"Environment": "Production"},
        )
        gateway_id = response["gatewayId"]
        return gateway_id
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
            logger.error("Gateway %s already exists.", gateway_name)
        else:
            logger.error(
                "Error creating gateway %s. Here's why %s",
                gateway_name,
                err.response["Error"]["Message"],
            )
        raise
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [CreateGateway](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

CreatePortal 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreatePortal。

CLI

AWS CLI

建立入口網站

下列create-portal範例會為風力發電廠公司建立 Web 入口網站。您只能在啟用 AWS 單一登入的相同區域中建立入口網站。

```
aws iotsitewise create-portal \
  --portal-name WindFarmPortal \
  --portal-description "A portal that contains wind farm projects for Example Corp." \
  --portal-contact-email support@example.com \
  --role-arn arn:aws:iam::123456789012:role/service-role/MySiteWiseMonitorServiceRole
```

輸出：

```
{
  "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalArn": "arn:aws:iotsitewise:us-west-2:123456789012:portal/a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalStartUrl": "https://a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE.app.iotsitewise.aws",
  "portalStatus": {
    "state": "CREATING"
  },
  "ssoApplicationId": "ins-a1b2c3d4-EXAMPLE"
}
```

如需詳細資訊，請參閱 IoT [SiteWise 使用者指南中的 AWS IoT SiteWise Monitor 入門](#)和 AWS IoT SiteWise 使用者指南中的[啟用 AWS SSO](#)。AWS IoT SiteWise

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CreatePortal](#)。

Java

SDK for Java 2.x

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Creates a new IoT SiteWise portal.
 *
 * @param portalName the name of the portal to create.
 * @param iamRole the IAM role ARN to use for the portal.
 * @param contactEmail the email address of the portal contact.
 * @return a {@link CompletableFuture} that represents a {@link String}
result of the portal ID. The calling code
 * can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
 * {@link CompletableFuture#get()}.
 *
 * <p>
 * If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 * it available to the calling code as a {@link CompletionException}.
By calling
 * {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<String> createPortalAsync(String portalName, String
iamRole, String contactEmail) {
    CreatePortalRequest createPortalRequest = CreatePortalRequest.builder()
        .portalName(portalName)
        .portalDescription("This is my custom IoT SiteWise portal.")
        .portalContactEmail(contactEmail)
        .roleArn(iamRole)
        .build();

    return getAsyncClient().createPortal(createPortalRequest)
        .handle((response, exception) -> {
            if (exception != null) {
```

```

        logger.error("Failed to create portal: {} ",
exception.getCause().getMessage());
        throw (CompletionException) exception;
    }
    return response.portalId();
});
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [CreatePortal](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
    CreatePortalCommand,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create a Portal.
 * @param {{ portalName: string, portalContactEmail: string, roleArn: string }}
 */
export const main = async ({ portalName, portalContactEmail, roleArn }) => {
    const client = new IoTSiteWiseClient({});
    try {
        const result = await client.send(
            new CreatePortalCommand({
                portalName: portalName, // The name to give the created Portal.
                portalContactEmail: portalContactEmail, // A valid contact email.
                roleArn: roleArn, // The ARN of a service role that allows the portal's
                users to access the portal's resources.
            }),
        );
    }
};

```

```

    console.log("Portal created successfully.");
    return result;
} catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
        console.warn(
            `${caught.message}. There was a problem creating the Portal.`
        );
    } else {
        throw caught;
    }
}
};

```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [CreatePortal](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

```

```

@classmethod
def from_client(cls) -> "IoTSitewiseWrapper":
    """
    Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
    client.

    :return: An instance of IoTSitewiseWrapper initialized with the default
    AWS IoT SiteWise client.
    """
    iotsitewise_client = boto3.client("iotsitewise")
    return cls(iotsitewise_client)

def create_portal(
    self, portal_name: str, iam_role_arn: str, portal_contact_email: str
) -> str:
    """
    Creates an AWS IoT SiteWise Portal.

    :param portal_name: The name of the portal to create.
    :param iam_role_arn: The ARN of an IAM role.
    :param portal_contact_email: The contact email of the portal.
    :return: The ID of the created portal.
    """
    try:
        response = self.iotsitewise_client.create_portal(
            portalName=portal_name,
            roleArn=iam_role_arn,
            portalContactEmail=portal_contact_email,
        )
        portal_id = response["portalId"]
        waiter = self.iotsitewise_client.get_waiter("portal_active")
        waiter.wait(portalId=portal_id, WaiterConfig={"MaxAttempts": 40})
        return portal_id
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
            logger.error("Portal %s already exists.", portal_name)
        else:
            logger.error(
                "Error creating portal %s. Here's why %s",
                portal_name,
                err.response["Error"]["Message"],
            )

```

```
raise
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [CreatePortal](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DeleteAsset 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DeleteAsset。

CLI

AWS CLI

刪除資產

下列delete-asset範例會刪除風力發電機資產。

```
aws iotsitewise delete-asset \  
  --asset-id a1b2c3d4-5678-90ab-cdef-3333EXAMPLE
```

輸出：

```
{  
  "assetStatus": {  
    "state": "DELETING"  
  }  
}
```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的 [刪除資產](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteAsset](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Deletes an asset.
 *
 * @param assetId the ID of the asset to be deleted.
 * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
 *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
 *         {@link CompletableFuture#get()}.
 *         <p>
 *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *         it available to the calling code as a {@link CompletionException}.
By calling
 *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String
assetId) {
    DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
        .assetId(assetId)
        .build();

    return getAsyncClient().deleteAsset(deleteAssetRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("An error occurred deleting asset with id: {}",
assetId);
            }
        });
}
```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [DeleteAsset](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import {
  DeleteAssetCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset.
 * @param {{ assetId : string }}
 */
export const main = async ({ assetId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteAssetCommand({
        assetId: assetId, // The model id to delete.
      }),
    );
    console.log("Asset deleted successfully.");
    return { assetDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the asset.`
      );
    } else {
      throw caught;
    }
  }
};
```


- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [DeleteAsset](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSiteWiseWrapper":
        """
        Creates an IoTSiteWiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSiteWiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)
```

```
def delete_asset(self, asset_id: str) -> None:
    """
    Deletes an AWS IoT SiteWise Asset.

    :param asset_id: The ID of the asset to delete.
    """
    try:
        self.iotsitewise_client.delete_asset(assetId=asset_id)
    except ClientError as err:
        logger.error(
            "Error deleting asset %s. Here's why %s",
            asset_id,
            err.response["Error"]["Message"],
        )
        raise
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [DeleteAsset](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DeleteAssetModel 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DeleteAssetModel。

CLI

AWS CLI

刪除資產模型

下列delete-asset-model範例會刪除風力渦輪機資產模型。

```
aws iotsitewise delete-asset-model \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

輸出：

```
{
  "assetModelStatus": {
    "state": "DELETING"
  }
}
```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[刪除資產模型](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[DeleteAssetModel](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```
/**
 * Deletes an Asset Model with the specified ID.
 *
 * @param assetModelId the ID of the Asset Model to delete.
 * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
 *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
 *         {@link CompletableFuture#get()}.
 *         <p>
 *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *         it available to the calling code as a {@link CompletionException}.
By calling
 *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<DeleteAssetModelResponse>
deleteAssetModelAsync(String assetModelId) {
    DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
```

```

        .assetModelId(assetModelId)
        .build();

    return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to delete asset model with ID:{}.\"",
exception.getMessage());
            }
        });
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [DeleteAssetModel](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
    DeleteAssetModelCommand,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Delete an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
    const client = new IoTSiteWiseClient({});
    try {
        await client.send(
            new DeleteAssetModelCommand({
                assetModelId: assetModelId, // The model id to delete.
            }),

```

```

    );
    console.log("Asset model deleted successfully.");
    return { assetModelDeleted: true };
} catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
        console.warn(
            `${caught.message}. There was a problem deleting the asset model.`
        );
    } else {
        throw caught;
    }
}
};

```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [DeleteAssetModel](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client

```

```

        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)

    def delete_asset_model(self, asset_model_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Asset Model.

        :param asset_model_id: The ID of the asset model to delete.
        """
        try:
            self.iotsitewise_client.delete_asset_model(assetModelId=asset_model_id)
        except ClientError as err:
            logger.error(
                "Error deleting asset model %s. Here's why %s",
                asset_model_id,
                err.response["Error"]["Message"],
            )
            raise

```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [DeleteAssetModel](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DeleteGateway 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DeleteGateway。

CLI

AWS CLI

刪除閘道

下列delete-gateway範例會刪除閘道。

```
aws iotsitewise delete-gateway \  
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[使用閘道擷取資料](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteGateway](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**  
 * Deletes the specified gateway.  
 *  
 * @param gatewayId the ID of the gateway to delete.  
 * @return a {@link CompletableFuture} that represents a {@link  
DeleteGatewayResponse} result.. The calling code  
 *         can attach callbacks, then handle the result or exception by  
calling {@link CompletableFuture#join()} or  
 *         {@link CompletableFuture#get()}.  
 *  
 * <p>
```

```

    *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *      it available to the calling code as a {@link CompletionException}.
By calling
    *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
        DeleteGatewayRequest deleteGatewayRequest =
DeleteGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().deleteGateway(deleteGatewayRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [DeleteGateway](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
    DeleteGatewayCommand,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

```



```

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeleteGatewayCommand({
        gatewayId: gatewayId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Gateway deleted successfully.");
    return { gatewayDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Gateway could not be found. Please check the Gateway Id.`
      );
    } else {
      throw caught;
    }
  }
};

```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [DeleteGateway](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class IoTSiteWiseWrapper:
```

```

"""Encapsulates AWS IoT SiteWise actions using the client interface."""

def __init__(self, iotsitewise_client: client) -> None:
    """
    Initializes the IoTSitewiseWrapper with an AWS IoT SiteWise client.

    :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
    provides low-level
                                access to AWS IoT SiteWise services.
    """
    self.iotsitewise_client = iotsitewise_client
    self.entry_id = 0 # Incremented to generate unique entry IDs for
    batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)

def delete_gateway(self, gateway_id: str) -> None:
    """
    Deletes an AWS IoT SiteWise Gateway.

    :param gateway_id: The ID of the gateway to delete.
    """
    try:
        self.iotsitewise_client.delete_gateway(gatewayId=gateway_id)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Gateway %s does not exist.", gateway_id)
        else:
            logger.error(
                "Error deleting gateway %s. Here's why %s",
                gateway_id,
                err.response["Error"]["Message"],
            )

```

```
raise
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [DeleteGateway](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DeletePortal 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DeletePortal。

CLI

AWS CLI

刪除入口網站

下列delete-portal範例會刪除風力發電廠公司的 Web 入口網站。

```
aws iotsitewise delete-portal \  
  --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

輸出：

```
{  
  "portalStatus": {  
    "state": "DELETING"  
  }  
}
```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的 [刪除入口網站](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeletePortal](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Deletes a portal.
 *
 * @param portalId the ID of the portal to be deleted.
 * @return a {@link CompletableFuture} that represents a {@link
DeletePortalResponse}. The calling code can attach
 *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
 *         {@link CompletableFuture#get()}.
 * <p>
 *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *         it available to the calling code as a {@link CompletionException}.
By calling
 *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<DeletePortalResponse> deletePortalAsync(String
portalId) {
    DeletePortalRequest deletePortalRequest = DeletePortalRequest.builder()
        .portalId(portalId)
        .build();

    return getAsyncClient().deletePortal(deletePortalRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to delete portal with ID: {}. Error:
{}", portalId, exception.getCause().getMessage());
            }
        });
}
```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [DeletePortal](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import {
  DeletePortalCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * List asset models.
 * @param {{ portalId : string }}
 */
export const main = async ({ portalId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    await client.send(
      new DeletePortalCommand({
        portalId: portalId, // The id of the portal.
      }),
    );
    console.log("Portal deleted successfully.");
    return { portalDeleted: true };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. There was a problem deleting the portal. Please check the portal id.`
      );
    } else {
      throw caught;
    }
  }
}
```

```
};
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [DeletePortal](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSiteWiseWrapper":
        """
        Creates an IoTSiteWiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSiteWiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
```

```
        return cls(iotsitewise_client)

    def delete_portal(self, portal_id: str) -> None:
        """
        Deletes an AWS IoT SiteWise Portal.

        :param portal_id: The ID of the portal to delete.
        """
        try:
            self.iotsitewise_client.delete_portal(portalId=portal_id)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error("Portal %s does not exist.", portal_id)
            else:
                logger.error(
                    "Error deleting portal %s. Here's why %s",
                    portal_id,
                    err.response["Error"]["Message"],
                )
            raise
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [DeletePortal](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DescribeAssetModel 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DescribeAssetModel。

CLI

AWS CLI

描述資產模型

下列describe-asset-model範例說明風力發電廠資產模型。

```
aws iotsitewise describe-asset-model \
```

```
--asset-model-id a1b2c3d4-5678-90ab-cdef-2222EXAMPLE
```

輸出：

```
{
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-2222EXAMPLE",
  "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/a1b2c3d4-5678-90ab-cdef-2222EXAMPLE",
  "assetModelName": "Wind Farm Model",
  "assetModelDescription": "Represents a wind farm that comprises many wind turbines",
  "assetModelProperties": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",
      "name": "Total Generated Power",
      "dataType": "DOUBLE",
      "unit": "kW",
      "type": {
        "metric": {
          "expression": "sum(power)",
          "variables": [
            {
              "name": "power",
              "value": {
                "propertyId": "a1b2c3d4-5678-90ab-cdef-66666EXAMPLE",
                "hierarchyId": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE"
              }
            }
          ]
        },
        "window": {
          "tumbling": {
            "interval": "1h"
          }
        }
      }
    }
  ],
  {
    "id": "a1b2c3d4-5678-90ab-cdef-88888EXAMPLE",
    "name": "Region",
    "dataType": "STRING",
```



```

        "type": {
            "attribute": {
                "defaultValue": " "
            }
        }
    },
    "assetModelHierarchies": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",
            "name": "Wind Turbines",
            "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
        }
    ],
    "assetModelCreationDate": 1575671284.0,
    "assetModelLastUpdateDate": 1575671988.0,
    "assetModelStatus": {
        "state": "ACTIVE"
    }
}

```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[描述特定資產模型](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[DescribeAssetModel](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```

/**
 * Retrieves the property IDs associated with a specific asset model.
 *
 * @param assetModelId the ID of the asset model that defines the properties.
 * @return a {@link CompletableFuture} that represents a {@link Map} result
 *         that associates the property name to the

```

```

    *      property ID. The calling code can attach callbacks, then handle the
    result or exception by calling
    *      {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *      <p>
    *      If any completion stage in this method throws an exception, the
    method logs the exception cause and keeps
    *      it available to the calling code as a {@link CompletionException}.
    By calling
    *      {@link CompletionException#getCause()}, the calling code can
    access the original exception.
    */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs:
{}. ", throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [DescribeAssetModel](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import {
  DescribeAssetModelCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe an asset model.
 * @param {{ assetModelId : string }}
 */
export const main = async ({ assetModelId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const { assetModelDescription } = await client.send(
      new DescribeAssetModelCommand({
        assetModelId: assetModelId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset model information retrieved successfully.");
    return { assetModelDescription: assetModelDescription };
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The asset model could not be found. Please check the asset model id.`,
      );
    } else {
      throw caught;
    }
  }
};
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [DescribeAssetModel](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DescribeGateway 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DescribeGateway。

CLI

AWS CLI

描述閘道

下列describe-gateway範例說明閘道。

```
aws iotsitewise describe-gateway \
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

輸出：

```
{
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayName": "ExampleCorpGateway",
  "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayPlatform": {
    "greengrass": {
      "groupArn": "arn:aws:greengrass:us-west-2:123456789012:/greengrass/groups/a1b2c3d4-5678-90ab-cdef-1b1b1EXAMPLE"
    }
  },
  "gatewayCapabilitySummaries": [
    {
      "capabilityNamespace": "iotsitewise:opcuacollector:1",
      "capabilitySyncStatus": "IN_SYNC"
    }
  ],
  "creationDate": 1588369971.457,
  "lastUpdateDate": 1588369971.457
}
```

```
}
```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[使用閘道擷取資料](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[DescribeGateway](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```
/**
 * Describes the specified gateway.
 *
 * @param gatewayId the ID of the gateway to describe.
 * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
 *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
 *         {@link CompletableFuture#get()}.
 *
 * <p>
 *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *         it available to the calling code as a {@link CompletionException}.
By calling
 *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
    DescribeGatewayRequest request = DescribeGatewayRequest.builder()
        .gatewayId(gatewayId)
        .build();

    return getAsyncClient().describeGateway(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
```

```

        logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
    }
});
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [DescribeGateway](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
    DescribeGatewayCommand,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Create an SSM document.
 * @param {{ content: string, name: string, documentType?: DocumentType }}
 */
export const main = async ({ gatewayId }) => {
    const client = new IoTSiteWiseClient({});
    try {
        const { gatewayDescription } = await client.send(
            new DescribeGatewayCommand({
                gatewayId: gatewayId, // The ID of the Gateway to describe.
            }),
        );
        console.log("Gateway information retrieved successfully.");
        return { gatewayDescription: gatewayDescription };
    } catch (caught) {
        if (caught instanceof Error && caught.name === "ResourceNotFound") {
            console.warn(

```

```

        `${caught.message}. The Gateway could not be found. Please check the
        Gateway Id.` ,
    );
    } else {
        throw caught;
    }
}
};

```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [DescribeGateway](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSiteWiseWrapper":
        """

```

```

        Creates an IoTSiteWiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSiteWiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)

def describe_gateway(self, gateway_id: str) -> Dict[str, Any]:
    """
    Describes an AWS IoT SiteWise Gateway.

    :param gateway_id: The ID of the gateway to describe.
    :return: A dictionary containing information about the gateway.
    """
    try:
        response =
self.iotsitewise_client.describe_gateway(gatewayId=gateway_id)
        return response
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.error("Gateway %s does not exist.", gateway_id)
        else:
            logger.error(
                "Error describing gateway %s. Here's why %s",
                gateway_id,
                err.response["Error"]["Message"],
            )
        raise

```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [DescribeGateway](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DescribePortal 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DescribePortal。

CLI

AWS CLI

描述入口網站

下列describe-portal範例說明風力發電廠公司的 Web 入口網站。

```
aws iotsitewise describe-portal \  
  --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

輸出：

```
{  
  "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",  
  "portalArn": "arn:aws:iotsitewise:us-west-2:123456789012:portal/  
a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",  
  "portalName": "WindFarmPortal",  
  "portalDescription": "A portal that contains wind farm projects for Example  
Corp.",  
  "portalClientId": "E-a1b2c3d4e5f6_a1b2c3d4e5f6EXAMPLE",  
  "portalStartUrl": "https://a1b2c3d4-5678-90ab-cdef-  
aaaaaEXAMPLE.app.iotsitewise.aws",  
  "portalContactEmail": "support@example.com",  
  "portalStatus": {  
    "state": "ACTIVE"  
  },  
  "portalCreationDate": "2020-02-04T23:01:52.90248068Z",  
  "portalLastUpdateDate": "2020-02-04T23:01:52.90248078Z",  
  "roleArn": "arn:aws:iam::123456789012:role/MySiteWiseMonitorServiceRole"  
}
```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[管理入口網站](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DescribePortal](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Retrieves a portal's description.
 *
 * @param portalId the ID of the portal to describe.
 * @return a {@link CompletableFuture} that represents a {@link String}
 * result of the portal's start URL
 *      (see: {@link DescribePortalResponse#portalStartUrl()}). The
 * calling code can attach callbacks, then handle the
 *      result or exception by calling {@link CompletableFuture#join()} or
 * {@link CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
 * method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
 * By calling
 *      {@link CompletionException#getCause()}, the calling code can
 * access the original exception.
 */
public CompletableFuture<String> describePortalAsync(String portalId) {
    DescribePortalRequest request = DescribePortalRequest.builder()
        .portalId(portalId)
        .build();

    return getAsyncClient().describePortal(request)
        .handle((response, exception) -> {
            if (exception != null) {
                logger.error("An exception occurred retrieving the portal
description: {}", exception.getCause().getMessage());
                throw (CompletionException) exception;
            }
            return response.portalStartUrl();
        });
}
```

```
}
```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [DescribePortal](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import {
  DescribePortalCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe a portal.
 * @param {{ portalId: string }}
 */
export const main = async ({ portalId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new DescribePortalCommand({
        portalId: portalId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Portal information retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {
      console.warn(
        `${caught.message}. The Portal could not be found. Please check the Portal Id.`
      );
    } else {

```

```
        throw caught;
    }
}
};
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [DescribePortal](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSiteWiseWrapper":
        """
        Creates an IoTSiteWiseWrapper instance with a default AWS IoT SiteWise
        client.
```

```
        :return: An instance of IoTSiteWiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)

def create_gateway(self, gateway_name: str, my_thing: str) -> str:
    """
    Creates an AWS IoT SiteWise Gateway.

    :param gateway_name: The name of the gateway to create.
    :param my_thing: The core device thing name.
    :return: The ID of the created gateway.
    """
    try:
        response = self.iotsitewise_client.create_gateway(
            gatewayName=gateway_name,
            gatewayPlatform={
                "greengrassV2": {"coreDeviceThingName": my_thing},
            },
            tags={"Environment": "Production"},
        )
        gateway_id = response["gatewayId"]
        return gateway_id
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExistsException":
            logger.error("Gateway %s already exists.", gateway_name)
        else:
            logger.error(
                "Error creating gateway %s. Here's why %s",
                gateway_name,
                err.response["Error"]["Message"],
            )
        raise
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [DescribePortal](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

GetAssetPropertyValue 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 GetAssetPropertyValue。

CLI

AWS CLI

擷取資產屬性的目前值

下列get-asset-property-value範例會擷取風力渦輪機資產目前的總功率。

```
aws iotsitewise get-asset-property-value \
  --asset-id a1b2c3d4-5678-90ab-cdef-3333EXAMPLE \
  --property-id a1b2c3d4-5678-90ab-cdef-6666EXAMPLE
```

輸出：

```
{
  "propertyValue": {
    "value": {
      "doubleValue": 6890.8677520453875
    },
    "timestamp": {
      "timeInSeconds": 1580853000,
      "offsetInNanos": 0
    },
    "quality": "GOOD"
  }
}
```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的 [查詢目前的資產屬性值](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [GetAssetPropertyValue](#)。

Java

SDK for Java 2.x

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * Fetches the value of an asset property.
 *
 * @param propId the ID of the asset property to fetch.
 * @param assetId the ID of the asset to fetch the property value for.
 * @return a {@link CompletableFuture} that represents a {@link Double}
 * result. The calling code can attach
 *     callbacks, then handle the result or exception by calling {@link
 * CompletableFuture#join()} or
 *     {@link CompletableFuture#get()}.
 *     <p>
 *     If any completion stage in this method throws an exception, the
 * method logs the exception cause and keeps
 *     it available to the calling code as a {@link CompletionException}.
 * By calling
 *     {@link CompletionException#getCause()}, the calling code can
 * access the original exception.
 */
public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
    GetAssetPropertyValueRequest assetPropertyValueRequest =
    GetAssetPropertyValueRequest.builder()
        .propertyId(propId)
        .assetId(assetId)
        .build();

    return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
        .handle((response, exception) -> {
            if (exception != null) {
                logger.error("Error occurred while fetching property
value: {}.", exception.getCause().getMessage());
                throw (CompletionException) exception;
            }
        });
}
```

```

        }
        return response.propertyValue().value().doubleValue();
    });
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [GetAssetPropertyValue](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
  GetAssetPropertyValueCommand,
  IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * Describe an asset property value.
 * @param {{ entryId : string }}
 */
export const main = async ({ entryId }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new GetAssetPropertyValueCommand({
        entryId: entryId, // The ID of the Gateway to describe.
      }),
    );
    console.log("Asset property information retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceNotFound") {

```



```

        console.warn(
            `${caught.message}. The asset property entry could not be found. Please
            check the entry id.` ,
        );
    } else {
        throw caught;
    }
}
};

```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [GetAssetPropertyValue](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSiteWiseWrapper":

```

```

        """
        Creates an IoTSiteWiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSiteWiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)

    def get_asset_property_value(
        self, asset_id: str, property_id: str
    ) -> Dict[str, Any]:
        """
        Gets the value of an AWS IoT SiteWise Asset Property.

        :param asset_id: The ID of the asset.
        :param property_id: The ID of the property.
        :return: A dictionary containing the value of the property.
        """
        try:
            response = self.iotsitewise_client.get_asset_property_value(
                assetId=asset_id, propertyId=property_id
            )
            return response["propertyValue"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.error(
                    "Asset %s or property %s does not exist.", asset_id,
property_id
                )
            else:
                logger.error(
                    "Error getting asset property value. Here's why %s",
err.response["Error"]["Message"],
                )
            raise

```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [GetAssetPropertyValue](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

ListAssetModels 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ListAssetModels。

CLI

AWS CLI

列出所有資產模型

下列list-asset-models範例列出目前區域中 AWS 您帳戶中定義的所有資產模型。

```
aws iotsitewise list-asset-models
```

輸出：

```
{
  "assetModelSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
      "name": "Wind Farm Model",
      "description": "Represents a wind farm that comprises many wind turbines",
      "creationDate": 1575671284.0,
      "lastUpdateDate": 1575671988.0,
      "status": {
        "state": "ACTIVE"
      }
    },
    {
      "id": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "arn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
      "name": "Wind Turbine Model",
      "description": "Represents a wind turbine manufactured by Example Corp",
      "creationDate": 1575671207.0,
```

```

        "lastUpdateDate": 1575686273.0,
        "status": {
            "state": "ACTIVE"
        }
    }
]
}

```

如需詳細資訊，請參閱 AWS IoT SiteWise 使用者指南中的[列出所有資產模型](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[ListAssetModels](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

```

/**
 * Retrieves the asset model ID for the given asset model name.
 *
 * @param assetModelName the name of the asset model for the ID.
 * @return a {@link CompletableFuture} that represents a {@link String}
 * result of the asset model ID or null if the
 *      asset model cannot be found. The calling code can attach
 * callbacks, then handle the result or exception
 *      by calling {@link CompletableFuture#join()} or {@link
 * CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
 * method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
 * By calling
 *      {@link CompletionException#getCause()}, the calling code can
 * access the original exception.
 */
public CompletableFuture<String> getAssetModelIdAsync(String assetModelName)
{

```

```

        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
            .handle((listAssetModelsResponse, exception) -> {
                if (exception != null) {
                    logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                    throw (CompletionException) exception;
                }
                for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                    if (assetModelSummary.name().equals(assetModelName)) {
                        return assetModelSummary.id();
                    }
                }
                return null;
            });
    }
}

```

- 如需 API 詳細資訊，請參閱AWS SDK for Java 2.x 《API 參考》中的 [ListAssetModels](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

import {
    ListAssetModelsCommand,
    IoTSiteWiseClient,
} from "@aws-sdk/client-iotsitewise";
import { parseArgs } from "node:util";

/**
 * List asset models.
 * @param {{ assetModelTypes : array }}
 */

```

```
export const main = async ({ assetModelTypes = [] }) => {
  const client = new IoTSiteWiseClient({});
  try {
    const result = await client.send(
      new ListAssetModelsCommand({
        assetModelTypes: assetModelTypes, // The model types to list
      }),
    );
    console.log("Asset model types retrieved successfully.");
    return result;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "IoTSiteWiseError") {
      console.warn(
        `${caught.message}. There was a problem listing the asset model types.`
      );
    } else {
      throw caught;
    }
  }
};
```

- 如需 API 詳細資訊，請參閱適用於 JavaScript 的 AWS SDK 《API 參考》中的 [ListAssetModels](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
class IoTSiteWiseWrapper:
    """Encapsulates AWS IoT SiteWise actions using the client interface."""

    def __init__(self, iotsitewise_client: client) -> None:
        """
        Initializes the IoTSiteWiseWrapper with an AWS IoT SiteWise client.
        """
```

```

        :param iotsitewise_client: A Boto3 AWS IoT SiteWise client. This client
        provides low-level
                                access to AWS IoT SiteWise services.
        """
        self.iotsitewise_client = iotsitewise_client
        self.entry_id = 0 # Incremented to generate unique entry IDs for
        batch_put_asset_property_value.

    @classmethod
    def from_client(cls) -> "IoTSitewiseWrapper":
        """
        Creates an IoTSitewiseWrapper instance with a default AWS IoT SiteWise
        client.

        :return: An instance of IoTSitewiseWrapper initialized with the default
        AWS IoT SiteWise client.
        """
        iotsitewise_client = boto3.client("iotsitewise")
        return cls(iotsitewise_client)

    def list_asset_models(self) -> List[Dict[str, Any]]:
        """
        Lists all AWS IoT SiteWise Asset Models.

        :return: A list of dictionaries containing information about each asset
        model.

        """
        try:
            asset_models = []
            paginator =
self.iotsitewise_client.get_paginator("list_asset_models")
            pages = paginator.paginate()
            for page in pages:
                asset_models.extend(page["assetModelSummaries"])
            return asset_models
        except ClientError as err:
            logger.error(
                "Error listing asset models. Here's why %s",
                err.response["Error"]["Message"],
            )
            raise

```

- 如需 API 詳細資訊，請參閱《適用於 AWS Python (Boto3) 的 SDK API 參考》中的 [ListAssetModels](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

中的安全性 AWS IoT SiteWise

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，該架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[共同責任模型](#)將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。AWS 也為您提供可安全使用的服務。在[AWS 合規計劃](#)中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用的合規計劃 AWS IoT SiteWise，請參閱[AWS 合規計劃範圍內的服務](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您了解如何在使用時套用共同責任模型 AWS IoT SiteWise。下列主題說明如何設定 AWS IoT SiteWise 以符合您的安全與合規目標。您也會了解如何使用其他 AWS 服務來協助您監控和保護 AWS IoT SiteWise 資源。

主題

- [中的資料保護 AWS IoT SiteWise](#)
- [中的資料加密 AWS IoT SiteWise](#)
- [的身分和存取管理 AWS IoT SiteWise](#)
- [的合規驗證 AWS IoT SiteWise](#)
- [中的彈性 AWS IoT SiteWise](#)
- [中的基礎設施安全 AWS IoT SiteWise](#)
- [中的組態和漏洞分析 AWS IoT SiteWise](#)
- [的 VPC 端點 AWS IoT SiteWise](#)
- [的安全最佳實務 AWS IoT SiteWise](#)

中的資料保護 AWS IoT SiteWise

AWS [共同責任模型](#)適用於 中的資料保護 AWS IoT SiteWise。如此模型所述，AWS 負責保護執行所有的全球基礎設施 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，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 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 和使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱 AWS CloudTrail 《使用者指南》中的 [使用 CloudTrail 追蹤](#)。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 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 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，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 \(FIPS\) 140-3](#)。

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 AWS IoT SiteWise 或使用主控台、API AWS CLI 或其他 AWS 服務 AWS SDKs 時。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。

主題

- [的網際網路流量隱私權 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 商業服務改進助理](#)

的網際網路流量隱私權 AWS IoT SiteWise

SiteWise Edge 閘道等 AWS IoT SiteWise 內部部署應用程式之間的連線會透過 Transport Layer Security (TLS) 連線進行保護。如需詳細資訊，請參閱 [的傳輸中資料加密 AWS IoT SiteWise](#)。

AWS IoT SiteWise 不支援 AWS 區域內可用區域之間的連線，或 AWS 帳戶之間的連線。

您一次只能在一個區域中設定 IAM Identity Center。SiteWise Monitor 會連線至您為 IAM Identity Center 設定的區域。這表示您使用一個區域進行 IAM Identity Center 存取，但您可以在任何區域中建立入口網站。

AWS IoT SiteWise 商業服務改進助理

AWS IoT SiteWise 助理不會將客戶資料用於改善服務或改善基礎 LLMs。

中的資料加密 AWS IoT SiteWise

資料加密是指在傳輸中（往返 AWS IoT SiteWise 以及 SiteWise Edge 閘道和伺服器之間）和靜態（存放在本機裝置或服務中）時保護資料 AWS。您可以使用 Transport Layer Security (TLS) 保護傳輸中的資料，或使用用戶端加密保護靜態資料。

Note

AWS IoT SiteWise 邊緣處理會公開在 SiteWise Edge 閘道中託管，並透過本機網路存取的 APIs。這些 APIs 透過由 AWS IoT SiteWise Edge 連接器擁有的伺服器憑證支援的 TLS 連線公開。對於用戶端身分驗證，這些 APIs 使用存取控制密碼。伺服器憑證私有金鑰和存取控制密碼都存放在 disk。AWS IoT SiteWise edge 處理上，依賴檔案系統加密來確保這些靜態憑證的安全性。

如需伺服器端加密和用戶端加密的相關資訊，請檢閱下列主題。

主題

- [中的靜態加密 AWS IoT SiteWise](#)
- [的傳輸中資料加密 AWS IoT SiteWise](#)
- [中的金鑰管理 AWS IoT SiteWise](#)

中的靜態加密 AWS IoT SiteWise

AWS IoT SiteWise 會將您的資料存放在 AWS 雲端和 AWS IoT SiteWise Edge 閘道。

AWS 雲端中的靜態資料

AWS IoT SiteWise 會將資料存放在依預設加密靜態資料的 AWS 其他服務中。靜態加密與 AWS Key Management Service (AWS KMS) 整合，用於管理用來加密資產屬性值和彙總值的加密金鑰 AWS IoT SiteWise。您可以選擇使用客戶受管金鑰來加密資產屬性值和彙總值 AWS IoT SiteWise。您可以透過建立、管理和檢視加密金鑰 AWS KMS。

您可以選擇 AWS 擁有的金鑰 來加密您的資料，或選擇客戶受管金鑰來加密您的資產屬性值和彙總值：

運作方式

靜態加密與 整合 AWS KMS，用於管理用來加密資料的加密金鑰。

- AWS 擁有的金鑰 – 預設加密金鑰。AWS IoT SiteWise 擁有此金鑰。您無法在 AWS 帳戶中檢視此金鑰。您也無法在 AWS CloudTrail 日誌中查看金鑰的操作。您可以使用此金鑰，無需額外費用。
- 客戶受管金鑰 – 金鑰存放在您的帳戶中，由您建立、擁有和管理。您可以完全控制 KMS 金鑰。AWS KMS 需支付額外費用。

AWS 擁有的金鑰

AWS 擁有的金鑰 不會存放在您的帳戶中。它們是 KMS 金鑰集合的一部分，這些金鑰 AWS 擁有和管理以在多個 AWS 帳戶中使用。AWS services 可以使用 AWS 擁有的金鑰 來保護您的資料。

您無法檢視、管理 AWS 擁有的金鑰、使用或稽核其使用方式。不過，您不需要進行任何工作或變更任何程式，即可保護加密資料的金鑰。

如果您使用，則無需支付月費或使用費 AWS 擁有的金鑰，也不會計入您帳戶的 AWS KMS 配額。

客戶受管金鑰

客戶受管金鑰為您在 帳戶中建立、擁有和管理的 KMS 金鑰。您可以完全控制這些 KMS 金鑰，如下所示：

- 建立和維護其金鑰政策、IAM 政策和授權
- 啟用和停用它們
- 輪換其密碼編譯資料
- 新增標籤
- 建立參考它們的別名
- 排程刪除它們

您也可以使用 CloudTrail 和 Amazon CloudWatch Logs 來追蹤 AWS KMS 代表您 AWS IoT SiteWise 傳送至 的請求。

如果您使用客戶受管金鑰，則需要授予您帳戶中存放之 KMS 金鑰的 AWS IoT SiteWise 存取權。AWS IoT SiteWise 會使用信封加密和金鑰階層來加密資料。您的 AWS KMS 加密金鑰用於加密此金鑰階層的根金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[封套加密](#)。

下列範例政策會代表您授予建立客戶受管金鑰的 AWS IoT SiteWise 許可。建立金鑰時，您需要允許 `kms:CreateGrant` 和 `kms:DescribeKey` 動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1603902045292",
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

您建立之授權的加密內容會使用您的 `aws:iotsitewise:subscriberId` 和 帳戶 ID。

SiteWise Edge 閘道上的靜態資料

AWS IoT SiteWise 閘道會將下列資料存放在本機檔案系統上：

- OPC UA 來源組態資訊
- 來自連線 OPC UA 來源的一組 OPC UA 資料串流路徑
- SiteWise Edge 閘道失去網際網路連線時快取的工業資料

SiteWise Edge 閘道會在 Unix 檔案許可和全磁碟加密（如果啟用）上執行 on AWS IoT Greengrass. AWS IoT Greengrass relies，以保護核心上的靜態資料。保護檔案系統和裝置是您的責任。

不過，AWS IoT Greengrass 會加密從 Secrets Manager 擷取的 OPC UA 伺服器秘密的本機副本。如需詳細資訊，請參閱《AWS IoT Greengrass Version 1 開發人員指南》中的[秘密加密](#)。

如需 AWS IoT Greengrass 核心靜態加密的詳細資訊，請參閱《AWS IoT Greengrass Version 1 開發人員指南》中的[靜態加密](#)。

的傳輸中資料加密 AWS IoT SiteWise

AWS IoT SiteWise 使用傳輸中的加密來保護裝置、閘道和 AWS 雲端之間傳輸的資料。與的通訊 AWS IoT SiteWise 會使用 HTTPS 和 TLS 1.2 加密，確保您的資料保持機密並受到保護，免於未經授權的存取或攔截。

資料正在傳輸時有三種通訊模式：

- [透過網際網路](#) – 本機裝置（包括 SiteWise Edge 閘道）與之間的通訊 AWS IoT SiteWise 已加密。
- [透過本機網路](#) – OpsHub for SiteWise 應用程式與 SiteWise Edge 閘道之間的通訊一律會加密。在瀏覽器中執行的 SiteWise 監控應用程式與 SiteWise Edge 閘道之間的通訊一律會加密。SiteWise Edge 閘道和 OPC UA 來源之間的通訊可以加密。
- [SiteWise Edge 閘道上的元件之間](#) – SiteWise Edge 閘道上的 AWS IoT Greengrass 元件之間的通訊不會加密。

主題

- [透過網際網路傳輸資料](#)
- [透過區域網路傳輸資料](#)
- [SiteWise Edge 上的本機元件之間傳輸中的資料](#)

透過網際網路傳輸資料

AWS IoT SiteWise 使用 Transport Layer Security (TLS) 來加密透過網際網路的所有通訊。傳送到 AWS 雲端的所有資料都會使用 MQTT 或 HTTPS 通訊協定透過 TLS 連線傳送，因此預設為安全。在上執行的 SiteWise Edge 閘道 AWS IoT Greengrass 和屬性值通知會使用 AWS IoT 傳輸安全模型。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[傳輸安全性](#)。

透過區域網路傳輸資料

SiteWise Edge 閘道遵循 OPC UA 規格，以與本機 OPC UA 來源進行通訊。您有責任將來源設為使用會加密傳輸中資料的訊息安全模式。

如果您選擇簽署訊息安全模式，則會簽署 SiteWise Edge 閘道與來源之間的傳輸中資料，但不會加密。如果您選擇簽署和加密訊息安全模式，則會簽署和加密 SiteWise Edge 閘道與來源之間的傳輸中資料。如需設定來源的詳細資訊，請參閱[將資料來源新增至 AWS IoT SiteWise Edge 閘道](#)。

邊緣主控台應用程式與 SiteWise Edge 閘道之間的通訊一律由 TLS 加密。SiteWise Edge 閘道上的 SiteWise Edge 連接器會產生並儲存自我簽署憑證，以便能夠與邊緣主控台建立適用於 AWS IoT

SiteWise 應用程式的 TLS 連線。您必須先將此憑證從 SiteWise Edge 閘道複製到 AWS IoT SiteWise 應用程式的邊緣主控台，再將應用程式連線至 SiteWise Edge 閘道。這可確保 AWS IoT SiteWise 應用程式的邊緣主控台能夠驗證是否已連線至您信任的 SiteWise Edge 閘道。

除了用於私密性和伺服器真實性的 TLS 之外，SiteWise Edge 還使用 SigV4 通訊協定來建立邊緣主控台應用程式的真實性。SiteWise Edge 閘道上的 SiteWise Edge 連接器接受並存放密碼，以驗證來自節點主控台應用程式的傳入連線、在瀏覽器中執行的 SiteWise Monitor 應用程式，以及根據 AWS IoT SiteWise SDK 的其他用戶端。

如需產生密碼和伺服器憑證的詳細資訊，請參閱[the section called “管理閘道”](#)。

SiteWise Edge 上的本機元件之間傳輸中的資料

SiteWise Edge 閘道會在 上執行 AWS IoT Greengrass，不會加密在核心本機 AWS IoT Greengrass 交換的資料，因為資料不會離開裝置。這包括 AWS IoT SiteWise 連接器等元件之間的 AWS IoT Greengrass 通訊。如需詳細資訊，請參閱《AWS IoT Greengrass Version 1 開發人員指南》中的[核心裝置上的資料](#)。

中的金鑰管理 AWS IoT SiteWise

AWS IoT SiteWise 雲端金鑰管理

根據預設，AWS IoT SiteWise 會使用 AWS 受管金鑰 來保護您在 AWS 雲端中的資料。您可以更新您的設定，以使用客戶受管金鑰來加密部分資料 AWS IoT SiteWise。您可以透過 () 建立、管理和檢視加密金鑰 AWS Key Management Service AWS KMS。

AWS IoT SiteWise 支援使用存放在 中的客戶受管金鑰進行伺服器端加密 AWS KMS，以加密下列資料：

- 資產屬性值
- 彙總值

Note

其他資料和資源會使用預設加密搭配 管理的金鑰進行加密 AWS IoT SiteWise。此金鑰會存放在 AWS IoT SiteWise 帳戶中。

如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[什麼是 AWS Key Management Service ?](#)。

使用客戶受管金鑰啟用加密

若要搭配使用客戶受管金鑰 AWS IoT SiteWise，您需要更新您的 AWS IoT SiteWise 設定。

使用 KMS 金鑰啟用加密

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 選擇帳戶設定，然後選擇編輯以開啟編輯帳戶設定頁面。
3. 針對加密金鑰類型，選擇選擇不同的 AWS KMS 金鑰。這可使用存放在 中的客戶受管金鑰進行加密 AWS KMS。

Note

目前，您只能將客戶受管金鑰加密用於資產屬性值和彙總值。

4. 使用下列其中一個選項選擇您的 KMS 金鑰：
 - 若要使用現有的 KMS 金鑰 – 從清單中選擇您的 KMS 金鑰別名。
 - 若要建立新的 KMS 金鑰 – 選擇建立 AWS KMS 金鑰。

Note

如此即會開啟 AWS KMS 儀表板。如需建立 KMS 金鑰的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[建立金鑰](#)。

5. 選擇儲存以更新您的設定。

SiteWise Edge 閘道金鑰管理

SiteWise Edge 閘道會在上執行 AWS IoT Greengrass，而 AWS IoT Greengrass 核心裝置會使用公有和私有金鑰向 AWS 雲端進行身分驗證，並加密本機秘密，例如 OPC UA 身分驗證秘密。如需詳細資訊，請參閱《AWS IoT Greengrass Version 1 開發人員指南》中的[金鑰管理](#)。

的身分和存取管理 AWS IoT SiteWise

AWS Identity and Access Management (IAM) 是 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行驗證（登入）和授權（具有許可）來使用 AWS IoT SiteWise 資源。IAM 是您可以免費使用 AWS 服務的。

主題

- [AWS IoT SiteWise 安全對象](#)
- [使用 中的身分進行身分驗證 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 如何使用 IAM](#)
- [AWS 的 受管政策 AWS IoT SiteWise](#)
- [使用 的服務連結角色 AWS IoT SiteWise](#)
- [在 中設定事件警示的許可 AWS IoT SiteWise](#)
- [中的跨服務混淆代理人預防 AWS IoT SiteWise](#)
- [對 AWS IoT SiteWise 身分和存取進行故障診斷](#)

AWS IoT SiteWise 安全對象

使用方式 AWS Identity and Access Management (IAM) 會有所不同，取決於您在其中執行的工作 AWS IoT SiteWise。

服務使用者 – 如果您使用 AWS IoT SiteWise 服務來執行任務，您的管理員會為您提供所需的登入資料和許可。當您使用更多 AWS IoT SiteWise 功能來執行工作時，您可能需要額外的許可。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 SiteWise 中的某項功能，請參閱 [對 AWS IoT SiteWise 身分和存取進行故障診斷](#)。

服務管理員 – 如果您負責公司 AWS IoT SiteWise 的資源，您可能擁有 的完整存取權 AWS IoT SiteWise。您的任務是判斷服務使用者應該存取哪些 AWS IoT SiteWise 功能和資源。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。若要進一步了解貴公司如何搭配 使用 IAM AWS IoT SiteWise，請參閱 [AWS IoT SiteWise 如何使用 IAM](#)。

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 SiteWise 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的以 AWS IoT SiteWise 身分為基礎的政策範例，請參閱 [AWS IoT SiteWise 身分型政策範例](#)。

使用 中的身分進行身分驗證 AWS IoT SiteWise

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者身分、IAM 使用者身分或擔任 IAM 角色來驗證 (登入 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分身分登入。AWS IAM Identity Center (IAM Identity Center) 使用者、您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。當您使用聯合 AWS 身分存取時，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。如需登入的詳細資訊 AWS，請參閱AWS 登入 《使用者指南》中的[如何登入您的 AWS 帳戶](#)。

如果您以 AWS 程式設計方式存取，AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。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[適用於 API 請求的AWS Signature 第 4 版](#)。

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 建議您使用多重驗證 (MFA) 來提高帳戶的安全性。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[IAM 中的AWS 多重要素驗證](#)。

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，該身分可完整存取帳戶中的所有 AWS 服務和資源。此身分稱為 AWS 帳戶 Theroot 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 中的身分 AWS 帳戶，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，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 使用者集合的身分。您無法以群組身分簽署。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，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 [IAM 使用者的使用案例](#)。

IAM 角色

[IAM 角色](#)是 中具有特定許可 AWS 帳戶 的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在 中擔任 IAM 角色 AWS Management Console，您可以從[使用者切換至 IAM 角色（主控台）](#)。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 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[擔任角色的方法](#)。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，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](#)》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分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。不過，對於某些 AWS 服務，您可以將政策直接連接到資源（而不是使用角色做為代理）。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 [IAM 中的跨帳戶資源存取](#)。
- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。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。
- 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您會被視為委託人。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 使用呼叫的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 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權給 AWS 服務](#)。

- **服務連結角色** – 服務連結角色是一種連結至的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶，並由服務擁有。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 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。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 執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。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臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式](#)。

AWS IoT SiteWise 如何使用 IAM

在您使用 AWS Identity and Access Management (IAM) 來管理 的存取權之前 AWS IoT SiteWise，您應該了解可使用哪些 IAM 功能 AWS IoT SiteWise。

IAM 功能	支援 AWS IoT SiteWise
具有資源層級許可的身分型政策	是
政策動作	是
政策資源	是
政策條件索引鍵	是
資源型政策	否
存取控制清單 (ACL)	否
標籤型授權 (ABAC)	是
臨時憑證	是
轉送存取工作階段 (FAS)	是

IAM 功能	支援 AWS IoT SiteWise
服務連結角色	是
服務角色	是

若要全面了解 AWS IoT SiteWise 和其他 AWS 服務如何與 IAM 搭配使用，請參閱《[AWS IAM 使用者指南](#)》中的與 IAM 搭配使用的 服務。

內容

- [AWS IoT SiteWise IAM 角色](#)
 - [搭配 使用臨時登入資料 AWS IoT SiteWise](#)
 - [轉送 的存取工作階段 \(FAS\) AWS IoT SiteWise](#)
 - [服務連結角色](#)
 - [服務角色](#)
 - [在 中選擇 IAM 角色 AWS IoT SiteWise](#)
- [以 AWS IoT SiteWise 標籤為基礎的授權](#)
- [AWS IoT SiteWise 身分型政策](#)
 - [政策動作](#)
 - [BatchPutAssetPropertyValue 授權](#)
 - [政策資源](#)
 - [政策條件索引鍵](#)
 - [範例](#)
- [AWS IoT SiteWise 身分型政策範例](#)
 - [政策最佳實務](#)
 - [使用 AWS IoT SiteWise 主控台](#)
 - [允許使用者檢視他們自己的許可](#)
 - [允許使用者在一個階層中將資料擷取至資產](#)
 - [根據標籤檢視 AWS IoT SiteWise 資產](#)

- [使用 中的政策管理存取權 AWS IoT SiteWise](#)
 - [身分型政策](#)
 - [資源型政策](#)
 - [存取控制清單 \(ACL\)](#)
 - [其他政策類型](#)
 - [多種政策類型](#)

AWS IoT SiteWise IAM 角色

[IAM 角色](#)是您 AWS 帳戶中具備特定許可的實體。

搭配 使用臨時登入資料 AWS IoT SiteWise

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。您取得暫時安全憑證的方式是透過呼叫 AWS STS API 作業 (例如，[AssumeRole](#) 或 [GetFederationToken](#))。

AWS IoT SiteWise 支援使用臨時登入資料。

SiteWise Monitor 支援聯合身分使用者存取入口網站。入口網站使用者使用其 IAM Identity Center 或 IAM 登入資料進行身分驗證。

Important

使用者或角色必須具有登入入口網站的*iotsitewise:DescribePortal*許可。

當使用者登入入口網站時，SiteWise Monitor 會產生工作階段政策，提供下列許可：

- AWS IoT SiteWise 帳戶中資產和資產資料的唯讀存取權，該入口網站的角色提供存取權。
- 存取該入口網站中使用者具有管理員 (專案擁有者) 或唯讀 (專案檢視者) 存取權的專案。

如需聯合入口網站使用者許可的詳細資訊，請參閱[使用的服務角色 AWS IoT SiteWise Monitor](#)。

轉送 的存取工作階段 (FAS) AWS IoT SiteWise

支援轉寄存取工作階段 (FAS)：是

當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為委託人。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 使用呼叫 的委託人許可 AWS 服務，結合

AWS 服務 請求向下游服務提出請求。只有當服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會提出 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帳戶中 AWS ，且由 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 SiteWise 支援服務連結角色。如需建立或管理 AWS IoT SiteWise 服務連結角色的詳細資訊，請參閱[使用的服務連結角色 AWS IoT SiteWise](#)。

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帳戶中 AWS ，並由該帳戶擁有。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 SiteWise 使用服務角色來允許 SiteWise Monitor 入口網站使用者代表您存取您的 AWS IoT SiteWise 部分資源。如需詳細資訊，請參閱[使用的服務角色 AWS IoT SiteWise Monitor](#)。

您必須先擁有必要的許可，才能在其中建立 AWS IoT Events 警示模型 AWS IoT SiteWise。如需詳細資訊，請參閱[在中設定事件警示的許可 AWS IoT SiteWise](#)。

在中選擇 IAM 角色 AWS IoT SiteWise

在中建立portal資源時 AWS IoT SiteWise，您必須選擇一個角色，以允許 SiteWise Monitor 入口網站的聯合身分使用者 AWS IoT SiteWise 代表您存取。如果您先前已建立服務角色，則 AWS IoT SiteWise 會為您提供可供選擇的角色清單。否則，您可以在建立入口網站時建立具有必要許可的角色。請務必選擇允許存取您資產和資產資料的角色。如需詳細資訊，請參閱[使用的服務角色 AWS IoT SiteWise Monitor](#)。

以 AWS IoT SiteWise 標籤為基礎的授權

您可以將標籤連接至 AWS IoT SiteWise 資源，或在請求中將標籤傳遞至其中 AWS IoT SiteWise。如需根據標籤控制存取，請使用 `aws: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。如需標記 AWS IoT SiteWise 資源的詳細資訊，請參閱 [標記您的 AWS IoT SiteWise 資源](#)。

若要檢視身分型政策範例，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 SiteWise 資產](#)。

AWS IoT SiteWise 身分型政策

IAM 政策可讓您控制誰可以執行哪些操作 AWS IoT SiteWise。您可以決定是否允許哪些動作，並為這些動作設定特定條件。例如，您可以制定有關誰可以在 中查看或變更資訊的規則 AWS IoT SiteWise。AWS IoT SiteWise 支援特定動作、資源和條件索引鍵。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 [IAM JSON 政策元素參考](#)。

政策動作

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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。政策動作通常具有與相關聯 AWS API 操作相同的名稱。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。

中的政策動作在動作之前 AWS IoT SiteWise 使用下列字首：iotsitewise:。例如，若要授予某人使用 BatchPutAssetPropertyValue API 操作將資產屬性資料上傳至 AWS IoT SiteWise 的許可，請在其政策中包含 iotsitewise:BatchPutAssetPropertyValue 動作。政策陳述式必須包含 Action 或 NotAction element。AWS IoT SiteWise 會定義自己的一組動作，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": [  
  "iotsitewise:action1",  
  "iotsitewise:action2"  
]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，如需指定開頭是 Describe 文字的所有動作，請包含以下動作：

```
"Action": "iotsitewise:Describe*"
```

若要查看 AWS IoT SiteWise 動作清單，請參閱《IAM 使用者指南》中的 [定義的動作 AWS IoT SiteWise](#)。

BatchPutAssetPropertyValue 授權

AWS IoT SiteWise 以不尋常的方式授權 [BatchPutAssetPropertyValue](#) 動作的存取。對於大多數動作，當您允許或拒絕存取時，如果未授予許可，該動作會傳回錯誤。使用 [BatchPutAssetPropertyValue](#)，您可以將多個資料項目傳送至單一 API 請求中的不同資產和資產屬性。AWS IoT SiteWise 會獨立授權每個資料項目。對於請求中授權失敗的任何個別項目，`AccessDeniedException` 會在傳回的錯誤清單中 AWS IoT SiteWise 包含。會 AWS IoT SiteWise 接收授權和成功之任何項目的資料，即使相同請求中的另一個項目失敗。

Important

在將資料擷取至資料串流之前，請執行下列動作：

- 如果您使用屬性別名來識別資料串流，請授權 `time-series` 資源。
- 如果您使用資產 ID 來識別包含相關聯資產屬性的資產，請授權 `asset` 資源。

政策資源

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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": "*" 
```

每個 IAM 政策陳述式都會套用到您使用其 ARN 指定的資源。ARN 一般的語法如下所示。

```
arn:${Partition}:${Service}:${Region}:${Account}:${ResourceType}/${ResourcePath}
```

如需 ARNs 格式的詳細資訊，請參閱 [使用 Amazon Resource Name \(ARNs\) 識別 AWS 資源](#)。

例如，若要在陳述式中指定 ID 為 `a1b2c3d4-5678-90ab-cdef-2222EXAMPLE` 的資產，請使用下列 ARN。

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/a1b2c3d4-5678-90ab-cdef-2222EXAMPLE"
```

若要指定屬於特定帳戶的所有資料串流，請使用萬用字元 (*)：

```
"Resource": "arn:aws:iotsitewise:region:123456789012:time-series/*"
```

若要指定屬於特定帳戶的所有資產，請使用萬用字元 (*)：

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/*"
```

有些 AWS IoT SiteWise 動作無法對特定資源執行，例如用於建立資源的動作。在這些情況下，您必須使用萬用字元 (*)。

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

若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

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

若要查看 AWS IoT SiteWise 資源類型及其 ARNs 的清單，請參閱《IAM 使用者指南》中的 [定義的資源類型 AWS IoT SiteWise](#)。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 SiteWise 定義的動作](#)。

政策條件索引鍵

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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，會使用邏輯 OR 操作 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的條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 [AWS 全域條件內容索引鍵](#)。

Important

許多條件鍵均為資源專屬，而且部分 API 動作使用多個資源。若您撰寫的政策陳述式具備條件鍵，請使用陳述式的 Resource 元素來指定套用該條件鍵的資源。如果您未這麼做，該政策可能會導致使用者完全無法執行該動作，因為未套用條件鍵之資源的條件檢查會失敗。若您不希望指定資源，或者若已撰寫政策的 Action 元素來納入多個 API 動作，則您必須使用 ...IfExists 條件類型，以確保未使用該條件鍵的資源，忽略該條件鍵。如需詳細資訊，請參閱 [...IAM 使用者指南中的 IfExists 條件](#)。

AWS IoT SiteWise 會定義自己的一組條件金鑰，也支援使用一些全域條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 [AWS 全域條件內容索引鍵](#)。

AWS IoT SiteWise 條件索引鍵

條件金鑰	描述	類型
<code>iotsitewise:isAssociatedWithAssetProperty</code>	資料串流是否與資產屬性相關聯。使用此條件索引鍵，根據資料串流的相關聯資產屬性是否存在來定義許可。 範例值：true	字串
<code>iotsitewise:assetHierarchyPath</code>	資產的階層路徑，也就是資產 ID 的字串，每個都以正斜線分隔。使用此條件鍵可根據帳戶中所有資產階層的子集來定義許可。 範例值：/a1b2c3d4-5678-90ab-cdef-2222EXAMPLE/a1b2c3d	字串

條件金鑰	描述	類型
	4-5678-90ab-cdef-6666EXAMPLE	
iotsitewise:propertyId	資產屬性的 ID。使用此條件鍵可根據資產模型的指定屬性來定義許可。此條件鍵適用於該模型的所有資產。 範例值：a1b2c3d4-5678-90ab-cdef-3333EXAMPLE	字串
iotsitewise:childAssetId	資產的 ID，此資產是其他資產相關聯的子系。使用此條件鍵可根據子資產定義許可。若要根據父資產來定義許可，請使用政策陳述式的資源區段。 範例值：a1b2c3d4-5678-90ab-cdef-6666EXAMPLE	字串
iotsitewise:iam	列出存取政策時的 IAM 身分 ARN。使用此條件金鑰來定義 IAM 身分的存取政策許可。 範例值：arn:aws:iam::123456789012:user/JohnDoe	字串、Null
iotsitewise:propertyAlias	識別資產屬性或資料串流的別名。使用此條件索引鍵根據別名定義許可。	字串

條件金鑰	描述	類型
<code>iotsitewise:user</code>	列出存取政策時 IAM Identity Center 使用者的 ID。使用此條件金鑰來定義 IAM Identity Center 使用者的存取政策許可。 範例值：a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE	字串、Null
<code>iotsitewise:group</code>	列出存取政策時的 IAM Identity Center 群組 ID。使用此條件金鑰來定義 IAM Identity Center 群組的存取政策許可。 範例值：a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbbEXAMPLE	字串、Null
<code>iotsitewise:portal</code>	存取政策中入口網站的 ID。使用此條件鍵，根據入口網站定義存取政策許可。 範例值：a1b2c3d4-5678-90ab-cdef-77777EXAMPLE	字串、Null
<code>iotsitewise:project</code>	存取政策中專案的 ID，或儀表板的專案 ID。使用此條件鍵，根據專案定義儀表板或存取政策許可。 範例值：a1b2c3d4-5678-90ab-cdef-88888EXAMPLE	字串、Null

若要了解您可以使用條件索引鍵的動作和資源，請參閱 [定義的動作 AWS IoT SiteWise](#)。

範例

若要檢視以 AWS IoT SiteWise 身為基礎的政策範例，請參閱 [AWS IoT SiteWise 身分型政策範例](#)。

AWS IoT SiteWise 身分型政策範例

根據預設，實體（使用者和角色）沒有建立或修改 AWS IoT SiteWise 資源的許可。他們也無法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 執行任務。若要調整許可，AWS Identity and Access Management (IAM) 管理員必須執行下列動作：

1. 建立 IAM 政策，授予使用者和角色對所需資源執行特定 API 操作的許可。
2. 將這些政策連接到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 標籤上建立政策](#)。

主題

- [政策最佳實務](#)
- [使用 AWS IoT SiteWise 主控台](#)
- [允許使用者檢視他們自己的許可](#)
- [允許使用者在一個階層中將資料擷取至資產](#)
- [根據標籤檢視 AWS IoT SiteWise 資產](#)

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 AWS IoT SiteWise 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 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，進一步減少許可。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。如果透過特定使用服務動作，您也可以使用條件來授予其存取權 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，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 IAM Access Analyzer 驗證政策](#)。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html中的透過 MFA 的安全 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 SiteWise 主控台

若要存取 AWS IoT SiteWise 主控台，您需要一組基本許可。這些許可可讓您查看和管理帳戶中 AWS IoT SiteWise 資源的詳細資訊 AWS。

如果您將政策設為太嚴格的政策，則對於具有該政策的使用者或角色（實體），主控台可能無法如預期般運作。為了確保這些實體仍然可以使用 AWS IoT SiteWise 主控台，請將 [AWSIoTSiteWiseConsoleFullAccess](#) 受管政策連接到這些實體，或定義這些實體的同等許可。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](#)。

如果實體僅使用 AWS Command Line Interface (CLI) 或 AWS IoT SiteWise API，而不是主控台，則不需要這些最低許可。在這種情況下，只要讓他們存取 API 任務所需的特定動作即可。

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。此政策包含在主控台或使用或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

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

```

        "Effect": "Allow",
        "Action": [
            "iam:GetUserPolicy",
            "iam:ListGroupsForUser",
            "iam:ListAttachedUserPolicies",
            "iam:ListUserPolicies",
            "iam:GetUser"
        ],
        "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
        "Sid": "NavigateInConsole",
        "Effect": "Allow",
        "Action": [
            "iam:GetGroupPolicy",
            "iam:GetPolicyVersion",
            "iam:GetPolicy",
            "iam:ListAttachedGroupPolicies",
            "iam:ListGroupPolicies",
            "iam:ListPolicyVersions",
            "iam:ListPolicies",
            "iam:ListUsers"
        ],
        "Resource": "*"
    }
]
}

```

允許使用者在一個階層中將資料擷取至資產

在此範例中，您想要授予 AWS 帳戶中的使用者從根資產 開始，將資料寫入特定資產階層中所有資產屬性的存取權 `a1b2c3d4-5678-90ab-cdef-2222EXAMPLE`。政策會將 `iotsitewise:BatchPutAssetPropertyValue` 許可授予使用者。此政策會使用 `iotsitewise:assetHierarchyPath` 條件鍵來限制對階層路徑符合資產或其子代之資產的存取。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PutAssetPropertyValuesForHierarchy",
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "arn:aws:iotsitewise::*:asset/*",

```



```

    "Condition": {
      "StringLike": {
        "iotsitewise:assetHierarchyPath": [
          "/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
          "/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/*"
        ]
      }
    }
  }
}
]
}

```

根據標籤檢視 AWS IoT SiteWise 資產

在身分型政策中使用條件，根據標籤控制對 AWS IoT SiteWise 資源的存取。此範例說明如何建立允許資產檢視的政策。但是，只有在資產標籤 Owner 數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 主控台上完成此動作的許可。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListAllAssets",
      "Effect": "Allow",
      "Action": [
        "iotsitewise:ListAssets",
        "iotsitewise:ListAssociatedAssets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "DescribeAssetIfOwner",
      "Effect": "Allow",
      "Action": "iotsitewise:DescribeAsset",
      "Resource": "arn:aws:iotsitewise:*:*:asset/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    }
  ]
}

```

將此政策連接至您帳戶中的使用者。如果名為的使用者richard-roe嘗試檢視 AWS IoT SiteWise 資產，資產必須加上標籤 Owner=richard-roe或 owner=richard-roe。否則，Richard 會被拒絕存取。條件標籤金鑰名稱不區分大小寫。因此，同時Owner符合 Owner和 owner。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 中的政策管理存取權 AWS IoT SiteWise

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件，AWS 當與身分或資源相關聯時，會定義其許可。當委託人（使用者、根使用者或角色工作階段）發出請求時，會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 形式存放在 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。具有該政策的使用者可以從 AWS Management Console AWS CLI、或 API AWS 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。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的 [透過客戶管理政策定義自訂 IAM 許可](#)。

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。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，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](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 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 AWS WAF 和 Amazon VPC 是支援 ACLs 的服務範例。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政策類型。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](#)。
- 服務控制政策 (SCPs) – SCPs 是 JSON 政策，可指定中組織或組織單位 (OU) 的最大許可 AWS Organizations。AWS Organizations 是一種用於分組和集中管理您企業擁有 AWS 帳戶之多個的服務。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 會限制成員帳戶中實體的許可，包括每個實體 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的 [服務控制政策](#)。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs 的詳細資訊，包括支援 RCPs AWS 服務的清單，請參閱 AWS Organizations 《使用者指南》中的 [資源控制政策 \(RCPs\)](#)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。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。若要了解如何 AWS 在涉及多個政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的 [政策評估邏輯](#)。

AWS 的 受管政策 AWS IoT SiteWise

使用 AWS 受管政策來簡化使用者、群組和角色的新增許可，而不是自行撰寫政策。建立提供團隊精確許可的 [IAM 客戶受管政策](#) 需要時間和專業知識。為了加快設定速度，請考慮將我們的 AWS 受管政策用於常見使用案例。尋找您 AWS 帳戶中的 AWS 受管政策。如需 AWS 受管政策的更多相關資訊，請參閱「IAM 使用者指南」中的 [AWS 受管政策](#)。

AWS 服務負責更新和維護 AWS 受管政策，這表示您無法修改這些政策的許可。有時，AWS IoT SiteWise 可能會新增許可以容納新功能，影響附加政策的所有身分。這類更新在引進新的服務或功能時很常見。不過，許可永遠不會移除，確保您的設定保持不變。

此外，AWS 支援跨多個 服務之任務函數的受管政策。例如，ReadOnlyAccess AWS 受管政策提供所有 AWS 服務和資源的唯讀存取權。當服務啟動新功能時，會為新操作和資源 AWS 新增唯讀許可。如需任務函數政策說明的清單，請參閱《IAM 使用者指南》中的 [AWS 任務函數的受管政策](#)。

AWS 受管政策：AWSIoTSiteWiseReadOnlyAccess

使用 AWSIoTSiteWiseReadOnlyAccess AWS 受管政策允許唯讀存取 AWS IoT SiteWise。

您可將 AWSIoTSiteWiseReadOnlyAccess 政策連接到 IAM 身分。

服務層級許可

此政策提供 的唯讀存取權 AWS IoT SiteWise。此政策中不包含任何其他服務許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:Describe*",
        "iotsitewise:List*",
        "iotsitewise:BatchGet*",
        "iotsitewise:Get*"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS 受管政策：AWSServiceRoleForIoTSiteWise

AWSServiceRoleForIoTSiteWise 角色使用具有下列許可AWSServiceRoleForIoTSiteWise的政策。此政策：

- 允許 部署 SiteWise Edge 閘道 AWS IoT SiteWise (在 上執行AWS IoT Greengrass)。
- 允許 AWS IoT SiteWise 執行記錄。
- 允許 對 AWS IoT TwinMaker 資料庫 AWS IoT SiteWise 執行中繼資料搜尋查詢。

如果您使用 AWS IoT SiteWise 搭配單一使用者帳戶，AWSServiceRoleForIoTSiteWise角色會在您的 IAM 帳戶中建立AWSServiceRoleForIoTSiteWise政策，並將其連接至AWSServiceRoleForIoTSiteWise的服務連結角色。 [AWS IoT SiteWise](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSiteWiseReadGreenGrass",
      "Effect": "Allow",
      "Action": [
        "greengrass:GetAssociatedRole",
        "greengrass:GetCoreDefinition",
        "greengrass:GetCoreDefinitionVersion",
        "greengrass:GetGroup",
        "greengrass:GetGroupVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowSiteWiseAccessLogGroup",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:DescribeLogGroups"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*"
    },
    {
      "Sid": "AllowSiteWiseAccessLog",
      "Effect": "Allow",
```

```

    "Action": [
      "logs:CreateLogStream",
      "logs:DescribeLogStreams",
      "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
  },
  {
    "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
    "Effect": "Allow",
    "Action": [
      "iottwinmaker:GetWorkspace",
      "iottwinmaker:ExecuteQuery"
    ],
    "Resource": "arn:aws:iottwinmaker:*:*:workspace/*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "iottwinmaker:linkedServices": [
          "IOTSITewise"
        ]
      }
    }
  }
]
}

```

AWS IoT SiteWiseAWS 受管政策的更新

您可以檢視 AWS 受管政策更新的詳細資訊 AWS IoT SiteWise，從此服務開始追蹤變更開始。如需此頁面變更的自動提醒，請訂閱 AWS IoT SiteWise 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
AWSServiceRoleForIoTSiteWise – 更新至現有政策	AWS IoT SiteWise 現在可對 AWS IoT TwinMaker 資料庫執行中繼資料搜尋查詢。	2023 年 11 月 6 日
AWSIoTSiteWiseReadOnlyAccess – 更新現有政策	AWS IoT SiteWise 新增了新的政策字首 BatchGet*，可讓您執行批次讀取操作。	2022 年 9 月 16 日

變更	描述	日期
AWSIoTSiteWiseReadOnlyAccess – 新政策	AWS IoT SiteWise 新增了新的政策，以授予 的唯讀存取權 AWS IoT SiteWise。	2021 年 11 月 24 日
AWS IoT SiteWise 開始追蹤變更	AWS IoT SiteWise 已開始追蹤其 AWS 受管政策的變更。	2021 年 11 月 24 日

使用 的服務連結角色 AWS IoT SiteWise

AWS IoT SiteWise use AWS Identity and Access Management (IAM) [服務連結角色](#)。服務連結角色是直接連結的唯一 IAM 角色類型 AWS IoT SiteWise。服務連結角色是由 預先定義，AWS IoT SiteWise 並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色透過自動包含所有必要的許可 AWS IoT SiteWise 來簡化 的組態。AWS IoT SiteWise 定義其服務連結角色的許可，除非另有定義，否則只能 AWS IoT SiteWise 擔任其角色。已定義的許可包括信任政策和許可政策。而且該許可政策無法連接到任何其他 IAM 實體。

您必須先刪除服務連結角色的相關資源，才能將其刪除。這可保護您的 AWS IoT SiteWise 資源，因為您不會不小心移除存取資源的許可。

如需有關支援服務連結角色的其他 服務的資訊，請參閱[AWS 使用 IAM 的服務](#)，並在服務連結角色欄中尋找具有是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

主題

- [AWS IoT SiteWise的服務連結角色許可](#)
- [為 AWS IoT SiteWise建立服務連結角色](#)
- [更新 的服務連結角色 AWS IoT SiteWise](#)
- [刪除 的服務連結角色 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 服務連結角色支援的 區域](#)
- [使用 的服務角色 AWS IoT SiteWise Monitor](#)

AWS IoT SiteWise的服務連結角色許可

AWS IoT SiteWise 使用名為 AWSServiceRoleForIoTSiteWise 的服務連結角色。AWS IoT SiteWise 會使用此服務連結角色來部署 SiteWise Edge 閘道 (在 上執行 AWS IoT Greengrass) 並執行記錄。

AWSServiceRoleForIoTSiteWise 服務連結角色使用具有下列許可AWSServiceRoleForIoTSiteWise的政策。此政策：

- 允許 部署 SiteWise Edge 閘道 AWS IoT SiteWise (在 上執行AWS IoT Greengrass)。
- 允許 AWS IoT SiteWise 執行記錄。
- 允許 對 AWS IoT TwinMaker 資料庫 AWS IoT SiteWise 執行中繼資料搜尋查詢。

如需 中允許動作的詳細資訊AWSServiceRoleForIoTSiteWise，請參閱 [AWS 的 受管政策 AWS IoT SiteWise](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSiteWiseReadGreenGrass",
      "Effect": "Allow",
      "Action": [
        "greengrass:GetAssociatedRole",
        "greengrass:GetCoreDefinition",
        "greengrass:GetCoreDefinitionVersion",
        "greengrass:GetGroup",
        "greengrass:GetGroupVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowSiteWiseAccessLogGroup",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:DescribeLogGroups"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*"
    },
    {
      "Sid": "AllowSiteWiseAccessLog",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents"
      ]
    }
  ]
}
```



```

    ],
    "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
  },
  {
    "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
    "Effect": "Allow",
    "Action": [
      "iottwinmaker:GetWorkspace",
      "iottwinmaker:ExecuteQuery"
    ],
    "Resource": "arn:aws:iottwinmaker:*:*:workspace/*",
    "Condition": {
      "ForAnyValue:StringEquals": {
        "iottwinmaker:linkedServices": [
          "IOTSITewise"
        ]
      }
    }
  }
]
}

```

您可以使用日誌來監控和疑難排解 SiteWise Edge 閘道。如需詳細資訊，請參閱[監控 SiteWise Edge 閘道日誌](#)。

若要允許 IAM 實體（例如使用者、群組或角色）建立、編輯或刪除服務連結角色，請先設定許可。如需詳細資訊，請參閱 IAM 使用者指南中的[服務連結角色許可](#)。

為 AWS IoT SiteWise 建立服務連結角色

AWS IoT SiteWise 需要服務連結角色才能執行特定動作，並代表您存取資源。服務連結角色是直接連結的唯一 AWS Identity and Access Management (IAM) 角色類型 AWS IoT SiteWise。透過建立此角色，您授予 AWS IoT SiteWise 必要的許可，以存取其操作所需的其他服務 AWS 和資源，例如 Amazon S3 用於資料儲存或 AWS IoT 裝置通訊。

您不需要手動建立一個服務連結角色。當您在 AWS IoT SiteWise 主控台中執行下列操作時，會為您 AWS IoT SiteWise 建立服務連結角色。

- 建立 Greengrass V1 閘道。
- 設定記錄選項。
- 在執行查詢橫幅中選擇加入按鈕。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您在 AWS IoT SiteWise 主控台中執行任何操作時，會再次為您 AWS IoT SiteWise 建立服務連結角色。

您也可以使用 IAM 主控台或 API 為 建立服務連結角色 AWS IoT SiteWise。

- 若要在 IAM 主控台中執行此作業，請使用 `AWSServiceRoleForIoTSiteWise` 政策建立角色，並與 建立信任關係 `iotsitewise.amazonaws.com`。
- 若要使用 AWS CLI 或 IAM API 執行此操作，請使用 `arn:aws:iam::aws:policy/aws-service-role/AWSServiceRoleForIoTSiteWise` 政策和與 的信任關係來建立角色 `iotsitewise.amazonaws.com`。

如需詳細資訊，請參閱《IAM 使用者指南》中的[建立服務連結角色](#)。

如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

更新 的服務連結角色 AWS IoT SiteWise

AWS IoT SiteWise 不允許您編輯 `AWSServiceRoleForIoTSiteWise` 服務連結角色。因為可能有各種實體會參考服務連結角色，所以您無法在建立角色之後變更其名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱 [《IAM 使用者指南》中的更新服務連結角色](#)。

刪除 的服務連結角色 AWS IoT SiteWise

如果不再使用需要服務連結角色的功能或服務，建議您刪除相關聯的角色。這是為了避免具有未受監控或維護的非作用中實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

Note

如果 AWS IoT SiteWise 服務在您嘗試刪除資源時使用角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試。

刪除 `AWSServiceRoleForIoTSiteWise` 使用 AWS IoT SiteWise 的資源

1. 停用 的記錄 AWS IoT SiteWise。如需詳細資訊，請參閱[變更您的記錄層級](#)
2. 刪除任何作用中的 SiteWise Edge 閘道。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台 AWS CLI、或 AWS API 來刪除 AWSServiceRoleForIoTSiteWise 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[刪除角色或執行個體描述](#)檔。

AWS IoT SiteWise 服務連結角色支援的 區域

AWS IoT SiteWise 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱 [AWS IoT SiteWise 端點和配額](#)。

使用 的服務角色 AWS IoT SiteWise Monitor

服務角色是服務擔任的 [IAM 角色](#)，可代您執行動作。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權給 AWS 服務](#)。

若要允許聯合身分 SiteWise Monitor 入口網站使用者存取您的 AWS IoT SiteWise 和 AWS IAM Identity Center 資源，您必須將服務角色連接至您建立的每個入口網站。服務角色必須將 SiteWise Monitor 指定為信任的實體，並包含 [AWSIoTSiteWiseMonitorPortalAccess](#) 受管政策或定義[同等許可](#)。此政策由維護 AWS，並定義 SiteWise Monitor 用來存取 AWS IoT SiteWise 和 IAM Identity Center 資源的一組許可。

建立 SiteWise Monitor 入口網站時，您必須選擇允許該入口網站使用者存取 AWS IoT SiteWise 和 IAM Identity Center 資源的角色。AWS IoT SiteWise 主控台可以為您建立和設定角色。您可以稍後在 IAM 中編輯角色。如果您從角色移除必要的許可或刪除角色，您的入口網站使用者將遇到使用 SiteWise Monitor 入口網站的問題。

Note

在 2020 年 4 月 29 日前建立的入口網站不需要服務角色。如果您在此日期前建立了入口網站，則必須連接服務角色才能繼續使用。若要這樣做，請導覽至 [AWS IoT SiteWise 主控台](#) 中的入口網站頁面，然後選擇遷移所有入口網站以使用 IAM 角色。

下列各節說明如何在 AWS Management Console 或 中建立和管理 SiteWise Monitor 服務角色 AWS Command Line Interface。

內容

- [SiteWise Monitor 的服務角色許可 \(傳統\)](#)
- [SiteWise Monitor 的服務角色許可 \(AI 感知\)](#)
- [管理 SiteWise Monitor 服務角色 \(主控台\)](#)
 - [尋找入口網站的服務角色 \(主控台\)](#)

- [建立 SiteWise Monitor 服務角色 \(AWS IoT SiteWise 主控台\)](#)
- [建立 SiteWise Monitor 服務角色 \(IAM 主控台\)](#)
- [變更入口網站的服務角色 \(主控台\)](#)
- [管理 SiteWise Monitor 服務角色 \(CLI\)](#)
 - [尋找入口網站的服務角色 \(CLI\)](#)
 - [建立 SiteWise Monitor 服務角色 \(CLI\)](#)
- [SiteWise Monitor 更新至 AWSIoTSiteWiseMonitorServiceRole](#)

SiteWise Monitor 的服務角色許可 (傳統)

當您建立入口網站時，會 AWS IoT SiteWise 讓您建立名稱開頭為 `AWSIoTSiteWiseMonitorServiceRole` 的角色。此角色允許聯合 SiteWise Monitor 使用者存取您的入口網站組態、資產、資產資料，以及 IAM Identity Center 組態。

該角色會信任下列服務來擔任此角色：

- `monitor.iotsitewise.amazonaws.com`

此角色使用下列許可政策，其開頭為 `AWSIoTSiteWiseMonitorServicePortalPolicy`，以允許 SiteWise Monitor 使用者對您帳戶中的資源完成動作。[AWSIoTSiteWiseMonitorPortalAccess](#) 受管政策定義對等許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:DescribePortal",
        "iotsitewise:CreateProject",
        "iotsitewise:DescribeProject",
        "iotsitewise:UpdateProject",
        "iotsitewise>DeleteProject",
        "iotsitewise:ListProjects",
        "iotsitewise:BatchAssociateProjectAssets",
        "iotsitewise:BatchDisassociateProjectAssets",
        "iotsitewise:ListProjectAssets",
        "iotsitewise:CreateDashboard",
        "iotsitewise:DescribeDashboard",

```

```

        "iotsitewise:UpdateDashboard",
        "iotsitewise:DeleteDashboard",
        "iotsitewise:ListDashboards",
        "iotsitewise:CreateAccessPolicy",
        "iotsitewise:DescribeAccessPolicy",
        "iotsitewise:UpdateAccessPolicy",
        "iotsitewise:DeleteAccessPolicy",
        "iotsitewise:ListAccessPolicies",
        "iotsitewise:DescribeAsset",
        "iotsitewise:ListAssets",
        "iotsitewise:ListAssociatedAssets",
        "iotsitewise:DescribeAssetProperty",
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetAssetPropertyValueHistory",
        "iotsitewise:GetAssetPropertyAggregates",
        "iotsitewise:BatchPutAssetPropertyValue",
        "iotsitewise:ListAssetRelationships",
        "iotsitewise:DescribeAssetModel",
        "iotsitewise:ListAssetModels",
        "iotsitewise:UpdateAssetModel",
        "iotsitewise:UpdateAssetModelPropertyRouting",
        "sso-directory:DescribeUsers",
        "sso-directory:DescribeUser",
        "iotevents:DescribeAlarmModel",
        "iotevents:ListTagsForResource"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "iotevents:BatchAcknowledgeAlarm",
        "iotevents:BatchSnoozeAlarm",
        "iotevents:BatchEnableAlarm",
        "iotevents:BatchDisableAlarm"
    ],
    "Resource": "*",
    "Condition": {
        "Null": {
            "iotevents:keyValue": "false"
        }
    }
},
{

```

```

        "Effect": "Allow",
        "Action": [
            "iotevents:CreateAlarmModel",
            "iotevents:TagResource"
        ],
        "Resource": "*",
        "Condition": {
            "Null": {
                "aws:RequestTag/iotsitewisemonitor": "false"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "iotevents:UpdateAlarmModel",
            "iotevents:DeleteAlarmModel"
        ],
        "Resource": "*",
        "Condition": {
            "Null": {
                "aws:ResourceTag/iotsitewisemonitor": "false"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "iam:PassedToService": [
                    "iotevents.amazonaws.com"
                ]
            }
        }
    }
]
}

```

如需警示所需許可的詳細資訊，請參閱 [在中設定事件警示的許可 AWS IoT SiteWise](#)。

當入口網站使用者登入時，SiteWise Monitor 會根據服務角色的交集和該使用者的存取政策來[建立工作階段](#)政策。存取政策會定義身分對入口網站和專案的存取權層級。如需入口網站許可和存取政策的詳細資訊，請參閱[管理您的 SiteWise Monitor 入口網站](#)和[建立存取政策](#)。

SiteWise Monitor 的服務角色許可 (AI 感知)

當您建立入口網站時，會 AWS IoT SiteWise 讓您建立名稱開頭為 IoTSiteWisePortalRole 的角色。此角色允許聯合 SiteWise Monitor 使用者存取您的入口網站組態、資產、資產資料，以及 IAM Identity Center 組態。

 Warning

SiteWise Monitor (AI 感知) 不支援專案擁有者和專案檢視器角色。

該角色會信任下列服務來擔任此角色：

- `monitor.iotsitewise.amazonaws.com`

此角色使用下列許可政策，其開頭為 IoTSiteWiseAIPortalAccessPolicy，以允許 SiteWise Monitor 使用者對您帳戶中的資源完成動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:CreateProject",
        "iotsitewise:DescribePortal",
        "iotsitewise:ListProjects",
        "iotsitewise:DescribeProject",
        "iotsitewise:UpdateProject",
        "iotsitewise>DeleteProject",
        "iotsitewise:CreateDashboard",
        "iotsitewise:DescribeDashboard",
        "iotsitewise:UpdateDashboard",
        "iotsitewise>DeleteDashboard",
        "iotsitewise:ListDashboards",
        "iotsitewise:ListAssets",
        "iotsitewise:DescribeAsset",

```

```

        "iotsitewise:ListAssociatedAssets",
        "iotsitewise:ListAssetProperties",
        "iotsitewise:DescribeAssetProperty",
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetAssetPropertyValueHistory",
        "iotsitewise:GetAssetPropertyAggregates",
        "iotsitewise:GetInterpolatedAssetPropertyValues",
        "iotsitewise:BatchGetAssetPropertyAggregates",
        "iotsitewise:BatchGetAssetPropertyValue",
        "iotsitewise:BatchGetAssetPropertyValueHistory",
        "iotsitewise:ListAssetRelationships",
        "iotsitewise:DescribeAssetModel",
        "iotsitewise:ListAssetModels",
        "iotsitewise:DescribeAssetCompositeModel",
        "iotsitewise:DescribeAssetModelCompositeModel",
        "iotsitewise:ListAssetModelProperties",
        "iotsitewise:ExecuteQuery",
        "iotsitewise:ListTimeSeries",
        "iotsitewise:DescribeTimeSeries",
        "iotsitewise:InvokeAssistant",
        "iotsitewise:DescribeDataset",
        "iotsitewise:ListDatasets",
        "iotevents:DescribeAlarmModel",
        "iotevents:ListTagsForResource",
        "iottwinmaker:ListWorkspaces",
        "iottwinmaker:ExecuteQuery",
        "iottwinmaker:GetWorkspace",
        "identitystore:DescribeUser"
    ],
    "Resource": "*"
}
]
}

```

當入口網站使用者登入時，SiteWise Monitor 會根據服務角色的交集和該使用者的存取政策來[建立工作階段](#)政策。

管理 SiteWise Monitor 服務角色（主控台）

AWS IoT SiteWise 主控台 有助於管理入口網站的 SiteWise Monitor 服務角色。建立入口網站時，主控台會檢查是否適合連接的現有角色。如果沒有可用，主控台可以為您建立和設定服務角色。如需詳細資訊，請參閱[在 SiteWise Monitor 中建立入口網站](#)。

主題

- [尋找入口網站的服務角色 \(主控台\)](#)
- [建立 SiteWise Monitor 服務角色 \(AWS IoT SiteWise 主控台\)](#)
- [建立 SiteWise Monitor 服務角色 \(IAM 主控台\)](#)
- [變更入口網站的服務角色 \(主控台\)](#)

尋找入口網站的服務角色 (主控台)

使用下列步驟來尋找連接至 SiteWise Monitor 入口網站的服務角色。

尋找入口網站的服務角色

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中選擇 Portals (入口網站)。
3. 選擇您要尋找服務角色的入口網站。

連接至入口網站的角色會顯示在 Permissions (許可)、[Service role (服務角色) 下方。

建立 SiteWise Monitor 服務角色 (AWS IoT SiteWise 主控台)

當您建立 SiteWise Monitor 入口網站時，您可以為入口網站建立服務角色。如需詳細資訊，請參閱[在 SiteWise Monitor 中建立入口網站](#)。

您也可以在此 AWS IoT SiteWise 主控台中為現有入口網站建立服務角色。這會取代入口網站的現有服務角色。

建立現有入口網站的服務角色

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Portals (入口網站)。
3. 選擇您要建立新服務角色的入口網站。
4. 在 Portal details (入口網站詳細資訊) 底下，選擇 Edit (編輯)。
5. 在 Permissions (許可) 下方，從清單中選擇 Create and use a new service role (建立並使用新的服務角色)。
6. 輸入您的新角色名稱。
7. 選擇儲存。

建立 SiteWise Monitor 服務角色 (IAM 主控台)

您可以從 IAM 主控台內的服務角色範本建立服務角色。此角色範本包含 [AWSIoTSiteWiseMonitorPortalAccess](#) 受管政策，並將 SiteWise Monitor 指定為信任的實體。

從入口網站服務角色範本建立服務角色

1. 導覽至 [IAM 主控台](#)。
2. 在導覽窗格中，選擇 Roles (角色)。
3. 選擇 Create Role (建立角色)。
4. 在選擇使用案例中，選擇 IoT SiteWise。
5. 在選取您的使用案例中，選擇 IoT SiteWise Monitor - Portal。
6. 選擇 Next: Permissions (下一步：許可)。
7. 選擇 Next: Tags (下一步：標籤)。
8. 選擇下一步：檢閱。
9. 輸入新服務角色的角色名稱。
10. 選擇建立角色。

變更入口網站的服務角色 (主控台)

使用下列程序為入口網站選擇不同的 SiteWise Monitor 服務角色。

變更入口網站的服務角色

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Portals (入口網站)。
3. 選擇您要變更服務角色的入口網站。
4. 在 Portal details (入口網站詳細資訊) 底下，選擇 Edit (編輯)。
5. 在 Permissions (許可) 下方，選擇 Use an existing role (使用現有的角色)。
6. 選擇要連接至此入口網站的現有角色。
7. 選擇儲存。

管理 SiteWise Monitor 服務角色 (CLI)

您可以使用 AWS CLI 執行下列入口網站服務角色管理任務：

主題

- [尋找入口網站的服務角色 \(CLI\)](#)
- [建立 SiteWise Monitor 服務角色 \(CLI\)](#)

尋找入口網站的服務角色 (CLI)

若要尋找連接至 SiteWise Monitor 入口網站的服務角色，請執行下列命令來列出目前區域中的所有入口網站。

```
aws iotsitewise list-portals
```

該操作會以下列格式傳回包含您入口網站摘要的回應。

```
{
  "portalSummaries": [
    {
      "id": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
      "name": "WindFarmPortal",
      "description": "A portal that contains wind farm projects for Example Corp.",
      "roleArn": "arn:aws:iam::123456789012:role/service-role/role-name",
      "startUrl": "https://a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE.app.iotsitewise.aws",
      "creationDate": "2020-02-04T23:01:52.90248068Z",
      "lastUpdateDate": "2020-02-04T23:01:52.90248078Z"
    }
  ]
}
```

如果您知道入口網站的 ID，也可以使用[描述入口網站](#)操作來尋找入口網站的角色。

建立 SiteWise Monitor 服務角色 (CLI)

使用下列步驟建立新的 SiteWise Monitor 服務角色。

建立 SiteWise Monitor 服務角色

1. 建立具有信任政策的角色，允許 SiteWise Monitor 擔任該角色。此範例會根據儲存在 JSON 字串中的信任政策，建立名為 **MySiteWiseMonitorPortalRole** 的角色。

Linux, macOS, or Unix

```
aws iam create-role --role-name MySiteWiseMonitorPortalRole --assume-role-
policy-document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "monitor.iotsitewise.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'
```

Windows command prompt

```
aws iam create-role --role-name MySiteWiseMonitorPortalRole --assume-role-
policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
,\"Principal\":{\"Service\":\"monitor.iotsitewise.amazonaws.com\"},\"Action\":
\"sts:AssumeRole\"}]}"
```

2. 從輸出中的角色中繼資料，複製角色 ARN。建立入口網站時，您可使用此 ARN 將角色與入口網站建立關聯。如需建立入口網站的詳細資訊，請參閱 AWS IoT SiteWise API 參考中的 [CreatePortal](#)。
3. a. 對於 SiteWise Monitor (Classic) – 將AWSIoTSiteWiseMonitorPortalAccess政策連接至角色，或連接定義同等許可的政策。

```
aws iam attach-role-policy --role-name MySiteWiseMonitorPortalRole --policy-arn
arn:aws:iam::aws:policy/service-role/AWSIoTSiteWiseMonitorPortalAccess
```

- b. 對於 SiteWise Monitor (AI 感知) – 將IoTSiteWiseAIPortalAccessPolicy政策連接到角色，或連接定義同等許可的政策。例如，建立具有入口網站存取許可的政策。下列範例會建立名為的政策MySiteWiseMonitorPortalAccess。

```
aws iam create-policy \
  --policy-name MySiteWiseMonitorPortalAccess \
  --policy-document '{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "iotsitewise:CreateProject",
      "iotsitewise:DescribePortal",
      "iotsitewise:ListProjects",
      "iotsitewise:DescribeProject",
      "iotsitewise:UpdateProject",
      "iotsitewise>DeleteProject",
      "iotsitewise:CreateDashboard",
      "iotsitewise:DescribeDashboard",
      "iotsitewise:UpdateDashboard",
      "iotsitewise>DeleteDashboard",
      "iotsitewise:ListDashboards",
      "iotsitewise:ListAssets",
      "iotsitewise:DescribeAsset",
      "iotsitewise:ListAssociatedAssets",
      "iotsitewise:ListAssetProperties",
      "iotsitewise:DescribeAssetProperty",
      "iotsitewise:GetAssetPropertyValue",
      "iotsitewise:GetAssetPropertyValueHistory",
      "iotsitewise:GetAssetPropertyAggregates",
      "iotsitewise:GetInterpolatedAssetPropertyValues",
      "iotsitewise:BatchGetAssetPropertyAggregates",
      "iotsitewise:BatchGetAssetPropertyValue",
      "iotsitewise:BatchGetAssetPropertyValueHistory",
      "iotsitewise:ListAssetRelationships",
      "iotsitewise:DescribeAssetModel",
      "iotsitewise:ListAssetModels",
      "iotsitewise:DescribeAssetCompositeModel",
      "iotsitewise:DescribeAssetModelCompositeModel",
      "iotsitewise:ListAssetModelProperties",
      "iotsitewise:ExecuteQuery",
      "iotsitewise:ListTimeSeries",
      "iotsitewise:DescribeTimeSeries",
      "iotsitewise:InvokeAssistant",
      "iotsitewise:DescribeDataset",
      "iotsitewise:ListDatasets",
      "iotevents:DescribeAlarmModel",
      "iotevents:ListTagsForResource",
      "iottwinmaker:ListWorkspaces",
      "iottwinmaker:ExecuteQuery",

```

```

        "iottwinmaker:GetWorkspace",
        "identitystore:DescribeUser"
    ],
    "Resource": "*"
  }
]
}'

```

將服務角色連接至現有入口網站

- 若要擷取入口網站的現有詳細資訊，請執行下列命令。將 *portal-id* 取代為入口網站的 ID。

```
aws iotsitewise describe-portal --portal-id portal-id
```

此操作會以下列格式傳回包含入口網站詳細資訊的回應。

```

{
  "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalArn": "arn:aws:iotsitewise:region:account-id:portal/a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalName": "WindFarmPortal",
  "portalDescription": "A portal that contains wind farm projects for Example Corp.",
  "portalClientId": "E-1a2b3c4d5e6f_sn6tbqHVzLWVEXAMPLE",
  "portalStartUrl": "https://a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE.app.iotsitewise.aws",
  "portalContactEmail": "support@example.com",
  "portalStatus": {
    "state": "ACTIVE"
  },
  "portalCreationDate": "2020-04-29T23:01:52.90248068Z",
  "portalLastUpdateDate": "2020-04-29T00:28:26.103548287Z",
  "roleArn": "arn:aws:iam::123456789012:role/service-role/AWSIoTSiteWiseMonitorServiceRole_1aEXAMPLE"
}

```

- 若要將服務角色連接至入口網站，請執行下列命令。將 *role-arn* 取代為服務角色 ARN，並以入口網站的現有值取代其餘參數。

```
aws iotsitewise update-portal \
```

```
--portal-id portal-id \
--role-arn role-arn \
--portal-name portal-name \
--portal-description portal-description \
--portal-contact-email portal-contact-email
```

SiteWise Monitor 更新至 AWSIoTSiteWiseMonitorServiceRole

您可以檢視 SiteWise Monitor 的 AWSIoTSiteWiseMonitorServiceRole 更新的詳細資訊，從此服務開始追蹤變更開始。如需此頁面變更的自動提醒，請訂閱 AWS IoT SiteWise 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
AWSIoTSiteWiseMonitorPortalAccess – 更新的策略	AWS IoT SiteWise 已更新警示功能的 AWSIoTSiteWiseMonitorPortalAccess 受管政策。	2021 年 5 月 27 日
AWS IoT SiteWise 開始追蹤變更	AWS IoT SiteWise 已開始追蹤其服務角色的變更。	2020 年 12 月 15 日

在 中設定事件警示的許可 AWS IoT SiteWise

當您使用 AWS IoT Events 警示模型來監控 AWS IoT SiteWise 資產屬性時，您必須擁有下列 IAM 許可：

- 允許 將資料 AWS IoT Events 傳送至其中 AWS IoT Events 的服務角色 AWS IoT SiteWise。如需詳細資訊，請參閱《AWS IoT Events 開發人員指南》中的 [Identity and Access Management AWS IoT Events](#)。
- 您必須具有下列 AWS IoT SiteWise 動作許可：iotsitewise:DescribeAssetModel 和 iotsitewise:UpdateAssetModelPropertyRouting。這些許可允許 AWS IoT SiteWise 將資產屬性值傳送至 AWS IoT Events 警示模型。

如需詳細資訊，請參閱《IAM 使用者指南》中的 [資源型政策](#)。

必要的動作許可

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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。

定義 AWS IoT Events 警示模型之前，您必須授予下列許可，AWS IoT SiteWise 允許將資產屬性值傳送至警示模型。

- `iotsitewise:DescribeAssetModel`、`iotsitewise:ListAssetModels` – 允許 AWS IoT Events 檢查資產屬性是否存在。
- `iotsitewise:UpdateAssetModelPropertyRouting` – 允許 AWS IoT SiteWise 自動建立訂閱，讓 AWS IoT SiteWise 將資料傳送到其中 AWS IoT Events。

如需 AWS IoT SiteWise 支援動作的詳細資訊，請參閱《服務授權參考》中的 [定義的動作 AWS IoT SiteWise](#)。

Example 範例許可政策 1

下列政策允許 AWS IoT SiteWise 將資產屬性值傳送至任何 AWS IoT Events 警示模型。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotevents:CreateAlarmModel",
        "iotevents:UpdateAlarmModel"
      ],
      "Resource": "arn:aws:iotevents:us-east-1:123456789012:alarmModel/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:DescribeAssetModel",
        "iotsitewise:ListAssetModels",
        "iotsitewise:UpdateAssetModelPropertyRouting"
      ],
      "Resource": "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/*"
    }
  ]
}
```



```
}
```

Example 範例許可政策 2

下列政策允許 AWS IoT SiteWise 將指定資產屬性的值傳送至指定的 AWS IoT Events 警示模型。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotevents:CreateAlarmModel",
        "iotevents:UpdateAlarmModel"
      ],
      "Resource": "arn:aws:iotevents:us-east-1:123456789012:alarmModel/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:DescribeAssetModel",
        "iotsitewise:ListAssetModels"
      ],
      "Resource": "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:UpdateAssetModelPropertyRouting"
      ],
      "Resource": [
        "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/12345678-90ab-cdef-1234-567890abcdef"
      ],
      "Condition": {
        "StringLike": {
          "iotsitewise:propertyId": "abcdef12-3456-7890-abcd-ef1234567890",
          "iotevents:alarmModelArn": "arn:aws:iotevents:us-east-1:123456789012:alarmModel/MyAlarmModel"
        }
      }
    }
  ]
}
```

```
}
```

(選用) ListInputRoutings 許可

當您更新或刪除資產模型時，AWS IoT SiteWise 可以檢查 中的 AWS IoT Events 警示模型是否正在監控與此資產模型相關聯的資產屬性。這可防止您刪除 AWS IoT Events 警示目前正在使用的資產屬性。若要在 中啟用此功能 AWS IoT SiteWise，您必須擁有 `iotevents:ListInputRoutings` 許可。此許可允許 AWS IoT SiteWise 呼叫 支援的 [ListInputRoutings](#) API 操作 AWS IoT Events。

Note

我們強烈建議您新增 ListInputRoutings 許可。

Example 許可政策範例

下列政策可讓您更新和刪除資產模型，並在其中使用 ListInputRoutings API AWS IoT SiteWise。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:UpdateAssetModel",
        "iotsitewise:DeleteAssetModel",
        "iotevents:ListInputRoutings"
      ],
      "Resource": "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/*"
    }
  ]
}
```

SiteWise Monitor 的必要許可

如果您想要在 SiteWise Monitor 入口網站中使用警示功能，您必須使用下列政策更新 [SiteWise Monitor 服務角色](#)：

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

```

    "Effect": "Allow",
    "Action": [
        "iotsitewise:DescribePortal",
        "iotsitewise:CreateProject",
        "iotsitewise:DescribeProject",
        "iotsitewise:UpdateProject",
        "iotsitewise>DeleteProject",
        "iotsitewise:ListProjects",
        "iotsitewise:BatchAssociateProjectAssets",
        "iotsitewise:BatchDisassociateProjectAssets",
        "iotsitewise:ListProjectAssets",
        "iotsitewise:CreateDashboard",
        "iotsitewise:DescribeDashboard",
        "iotsitewise:UpdateDashboard",
        "iotsitewise>DeleteDashboard",
        "iotsitewise:ListDashboards",
        "iotsitewise:CreateAccessPolicy",
        "iotsitewise:DescribeAccessPolicy",
        "iotsitewise:UpdateAccessPolicy",
        "iotsitewise>DeleteAccessPolicy",
        "iotsitewise:ListAccessPolicies",
        "iotsitewise:DescribeAsset",
        "iotsitewise:ListAssets",
        "iotsitewise:ListAssociatedAssets",
        "iotsitewise:DescribeAssetProperty",
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetAssetPropertyValueHistory",
        "iotsitewise:GetAssetPropertyAggregates",
        "iotsitewise:BatchPutAssetPropertyValue",
        "iotsitewise:ListAssetRelationships",
        "iotsitewise:DescribeAssetModel",
        "iotsitewise:ListAssetModels",
        "iotsitewise:UpdateAssetModel",
        "iotsitewise:UpdateAssetModelPropertyRouting",
        "sso-directory:DescribeUsers",
        "sso-directory:DescribeUser",
        "iotevents:DescribeAlarmModel",
        "iotevents:ListTagsForResource"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [

```

```

        "iotevents:BatchAcknowledgeAlarm",
        "iotevents:BatchSnoozeAlarm",
        "iotevents:BatchEnableAlarm",
        "iotevents:BatchDisableAlarm"
    ],
    "Resource": "*",
    "Condition": {
        "Null": {
            "iotevents:keyValue": "false"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "iotevents:CreateAlarmModel",
        "iotevents:TagResource"
    ],
    "Resource": "*",
    "Condition": {
        "Null": {
            "aws:RequestTag/iotsitewisemonitor": "false"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "iotevents:UpdateAlarmModel",
        "iotevents>DeleteAlarmModel"
    ],
    "Resource": "*",
    "Condition": {
        "Null": {
            "aws:ResourceTag/iotsitewisemonitor": "false"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
    ],
    "Resource": "*",

```

```

    "Condition": {
      "StringEquals": {
        "iam:PassedToService": [
          "iotevents.amazonaws.com"
        ]
      }
    }
  }
]
}

```

中的跨服務混淆代理人預防 AWS IoT SiteWise

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許可的實體執行該動作。在 AWS 中，跨服務模擬可能會導致混淆代理人問題。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。

我們建議在資源政策中使用 [aws:SourceArn](#) 和 [aws:SourceAccount](#) 全域條件內容索引鍵，以限制將另一個服務 AWS IoT SiteWise 提供給資源的許可。如果 `aws:SourceArn` 值不包含帳戶 ID (例如 Amazon S3 儲存貯體 Amazon Resource Name (ARN))，則您必須使用這兩個全域條件內容索引鍵來限制許可。如果同時使用這兩個全域條件內容索引鍵，且 `aws:SourceArn` 值包含帳戶 ID，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。

- 如果您想要僅允許一個資源與跨服務存取相關聯，則請使用 `aws:SourceArn`。
- 如果您想要允許該帳戶中的任何資源與跨服務使用相關聯，請使用 `aws:SourceAccount`。

的值 `aws:SourceArn` 必須是與 `sts:AssumeRole` 請求相關聯的 AWS IoT SiteWise 客戶資源。

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，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` 全域條件內容索引鍵，同時使用萬用字元 (*) 表示 ARN 的未知部分。例如 `arn:aws:servicename:*:123456789012:*`。

Example – 預防混淆代理人

下列範例示範如何在 中使用 `aws:SourceArn` 和 `aws:SourceAccount` 全域條件內容索引鍵 AWS IoT SiteWise，以防止混淆代理人問題。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "iotsitewise.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Resource": [
      "arn:aws:iotsitewise:::ResourceName/*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:iotsitewise:*:123456789012:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

對 AWS IoT SiteWise 身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 AWS IoT SiteWise 和 AWS Identity and Access Management (IAM) 時可能遇到的常見問題。

主題

- [我無權在 中執行動作 AWS IoT SiteWise](#)
- [我未獲得執行 iam:PassRole 的授權](#)
- [我想要允許 AWS 帳戶外的人員存取我的 AWS IoT SiteWise 資源](#)

我無權在 中執行動作 AWS IoT SiteWise

如果 AWS Management Console 告知您無權執行 動作，則必須聯絡您的管理員尋求協助。您的管理員是提供您使用者名稱和密碼的人員。

當 IAM mateojackson 使用者嘗試使用主控台檢視資產的詳細資訊，但沒有 iotsitewise:DescribeAsset 許可時，會發生下列範例錯誤。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
iotsitewise:DescribeAsset on resource: a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

在此情況下，Mateo 會請求管理員更新政策，允許其使用 `iotsitewise:DescribeAsset` 動作存取 ID 為 `a1b2c3d4-5678-90ab-cdef-22222EXAMPLE` 的資產資源。

我未獲得執行 `iam:PassRole` 的授權

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 SiteWise。

有些 AWS 服務可讓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 SiteWise 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 管理員。您的管理員提供您的簽署憑證。

我想要允許 AWS 帳戶外的人員存取我的 AWS IoT SiteWise 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：

- 若要了解是否 AWS IoT SiteWise 支援這些功能，請參閱 [AWS IoT SiteWise 如何使用 IAM](#)。
- 若要了解如何提供您擁有 AWS 帳戶的資源存取權，請參閱 [《IAM 使用者指南》中的在您的 AWS 帳戶的另一個中為 IAM 使用者提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱 [《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶擁有](#)。
- 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 \(聯合身分\)](#)。

- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 [IAM 中的跨帳戶資源存取](#)。

的合規驗證 AWS IoT SiteWise

AWS IoT SiteWise 不在任何 AWS 合規計劃的範圍內。

如需特定合規計劃範圍內 AWS 的服務清單，請參閱[AWS 合規計劃範圍內的服務](#)。如需一般資訊，請參閱 [AWS Compliance Programs](#)。

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[下載報告 in AWS Artifact](#)

您使用時的合規責任 AWS IoT SiteWise 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。AWS 提供下列資源來協助合規：

- [安全與合規快速入門指南](#) – 這些部署指南討論架構考量，並提供部署以安全與合規為中心之基準環境的步驟 AWS。
- [HIPAA 安全與合規架構白皮書](#) – 此白皮書說明公司如何使用 AWS 來建立符合 HIPAA 規範的應用程式。
- [AWS 合規資源](#) – 此工作手冊和指南的集合可能適用於您的產業和位置。
- AWS Config 開發人員指南中的[使用規則評估資源](#) – AWS Config 服務會評估資源組態符合內部實務、產業準則和法規的程度。
- [AWS Security Hub](#) – AWS 此服務提供 內安全狀態的全方位檢視 AWS，可協助您檢查是否符合安全產業標準和最佳實務。
- [工業 IoT 解決方案的十項安全黃金規則](#) – 此部落格文章介紹十項黃金規則，可協助保護您的工業控制系統 (ICS)、工業物聯網 (IIoT) 和雲端環境。
- [製造 OT 的安全最佳實務](#) – 此白皮書說明針對 AWS 雲端設計、部署和架構這些內部部署混合式製造工作負載的安全最佳實務。

中的彈性 AWS IoT SiteWise

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置。AWS 區域提供多個實體分隔和隔離的可用區域，這些可用區域以低延遲、高輸送量和高備援聯網連接。透過可用區域，您可以設計與操作的應用程式和資料庫，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、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

AWS IoT SiteWise 完全受管，並使用高可用性且耐用 AWS 的服務，例如 Amazon S3 和 Amazon EC2。為了確保在可用區域中斷的情況下的可用性，會跨多個可用區域 AWS IoT SiteWise 運作。

如需 AWS 區域和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

除了 AWS 全球基礎設施之外，AWS IoT SiteWise 還提供數種功能，以協助支援您的資料彈性和備份需求：

- 您可以透過 AWS IoT Core MQTT 訊息將屬性值更新發佈至 `aws-iot-site`，然後設定規則以對該資料採取行動。使用此功能，您可以備份 Amazon S3 和 Amazon DynamoDB 等 AWS 其他服務中的資料。如需詳細資訊，請參閱[與其他 AWS 服務互動](#)及[使用資產屬性通知將資料匯出至 Amazon S3](#)。
- 您可以使用 AWS IoT SiteWise Get* APIs來擷取和備份歷史資產屬性資料。如需詳細資訊，請參閱[在 中查詢歷史資產屬性值 AWS IoT SiteWise](#)。
- 您可以使用 AWS IoT SiteWise Describe* APIs 來擷取 資源的定義，例如資產和模型。您可以備份這些定義，稍後再用來重新建立您的資源。如需詳細資訊，請參閱 [AWS IoT SiteWise API 參考](#)。

中的基礎設施安全 AWS IoT SiteWise

作為受管服務，AWS IoT SiteWise 受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全的最佳實務來設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 發佈的 API 呼叫，AWS IoT SiteWise 透過網路存取。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 [AWS Security Token Service](#) (AWS STS) 來產生暫時安全憑證來簽署請求。

SiteWise Edge 閘道在 上執行 AWS IoT Greengrass，使用 X.509 憑證和密碼編譯金鑰來連接和驗證 AWS 雲端。如需詳細資訊，請參閱《AWS IoT Greengrass Version 1 開發人員指南》中的 [裝置身分驗證和授權 AWS IoT Greengrass](#)。

中的組態和漏洞分析 AWS IoT SiteWise

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。這些特點使叢集設定複雜且極易出錯。由於裝置通常具有有限的處理能力、記憶體和儲存體，因此不一定支援加密和其他安全措施。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 透過提供識別安全問題和偏離最佳實務的工具來解決這些挑戰。使用 AWS IoT Device Defender 來分析、稽核和監控連線裝置，以偵測異常行為並降低安全風險。AWS IoT Device Defender 可以稽核裝置機群，以確保他們遵守安全最佳實務，並偵測裝置上的異常行為。這可讓您在 AWS IoT 裝置機群中強制執行一致的安全政策，並在裝置遭到入侵時快速回應。如需詳細資訊，請參閱 AWS IoT Device Defender 開發人員指南中的[什麼是 AWS IoT Device Defender](#)。

如果您使用 SiteWise Edge 閘道將資料擷取至服務，則您有責任設定和維護 SiteWise Edge 閘道的環境。此責任包括升級至 SiteWise Edge 閘道系統軟體、AWS IoT Greengrass 軟體和 AWS IoT SiteWise 連接器的最新版本。如需詳細資訊，請參閱《AWS IoT Greengrass Version 1 開發人員指南》和《[設定 AWS IoT Greengrass 核心管理 SiteWise Edge 閘道](#)》。

的 VPC 端點 AWS IoT SiteWise

介面 VPC 端點會在您的虛擬私有雲端 (VPC) 與 之間建立私有連線 AWS IoT SiteWise。[AWS PrivateLink](#) 支援介面端點，讓私有存取 API AWS IoT SiteWise 操作。您可以略過網際網路閘道、NAT 裝置、VPN 連接或 的需求 AWS Direct Connect。VPC 中的執行個體不需要公有 IP 地址，即可與 AWS IoT SiteWise API 操作通訊。VPC 與 之間的流量 AWS IoT SiteWise 不會離開 AWS 網路。

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。

在您設定介面 VPC 端點之前 AWS IoT SiteWise，請檢閱《AWS PrivateLink 指南》中的[使用介面 VPC 端點存取 AWS 服務](#)。

中的 VPC 端點 API 操作 AWS IoT SiteWise

AWS IoT SiteWise 支援從您的 VPC 呼叫下列 AWS IoT SiteWise API 操作：

- 對於所有資料平面 API 操作，請使用下列端點：將 取代 *region* 為 AWS 區域

```
data.iotsitewise.region.amazonaws.com
```

資料平面 API 操作包括下列項目：

- [BatchGetAssetPropertyValue](#)
 - [BatchGetAssetPropertyValueHistory](#)
 - [BatchPutAssetPropertyValue](#)
 - [GetAssetPropertyAggregates](#)
 - [GetAssetPropertyValue](#)
 - [GetAssetPropertyValueHistory](#)
 - [GetInterpolatedAssetPropertyValues](#)
- 對於您用來管理資產模型、資產、SiteWise Edge 閘道、標籤和帳戶組態的控制平面 API 操作，請使用下列端點。*region*將 取代為您的 AWS 區域。

```
api.iotsitewise.region.amazonaws.com
```

支援的控制平面 API 操作包括下列項目：

- [AssociateAssets](#)
- [CreateAsset](#)
- [CreateAssetModel](#)
- [DeleteAsset](#)
- [DeleteAssetModel](#)
- [DeleteDashboard](#)
- [DescribeAsset](#)
- [DescribeAssetModel](#)
- [DescribeAssetProperty](#)
- [DescribeDashboard](#)
- [DescribeLoggingOptions](#)
- [DisassociateAssets](#)
- [ListAssetModels](#)
- [ListAssetRelationships](#)
- [ListAssets](#)
- [ListAssociatedAssets](#)
- [PutLoggingOptions](#)
- [UpdateAsset](#)

- [UpdateAssetModel](#)
- [UpdateAssetProperty](#)
- [CreateGateway](#)
- [DeleteGateway](#)
- [DescribeDefaultEncryptionConfiguration](#)
- [DescribeGateway](#)
- [DescribeGatewayCapabilityConfiguration](#)
- [DescribeStorageConfiguration](#)
- [ListGateways](#)
- [ListTagsForResource](#)
- [UpdateGateway](#)
- [UpdateGatewayCapabilityConfiguration](#)
- [PutDefaultEncryptionConfiguration](#)
- [PutStorageConfiguration](#)
- [TagResource](#)
- [UntagResource](#)

 Note

控制平面 API 操作的介面 VPC 端點目前不支援呼叫下列 SiteWise Monitor API 操作：

- [BatchAssociateProjectAssets](#)
- [BatchDisassociateProjectAssets](#)
- [CreateAccessPolicy](#)
- [CreateDashboard](#)
- [CreatePortal](#)
- [CreateProject](#)
- [DeleteAccessPolicy](#)
- [DeletePortal](#)
- [DeleteProject](#)
- [DescribeAccessPolicy](#)
- [DescribePortal](#)

- [DescribeProject](#)
- [ListAccessPolicies](#)
- [ListDashboards](#)
- [ListPortals](#)
- [ListProjects](#)
- [ListProjectAssets](#)
- [UpdateAccessPolicy](#)
- [UpdateDashboard](#)
- [UpdatePortal](#)
- [UpdateProject](#)

為 AWS IoT SiteWise 建立介面 VPC 端點

若要建立 AWS IoT SiteWise 服務的 VPC 端點，請使用 Amazon VPC 主控台或 AWS Command Line Interface (AWS CLI)。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的[使用介面 VPC 端點存取 AWS 服務](#)。

AWS IoT SiteWise 使用下列其中一個服務名稱建立的 VPC 端點：

- 對於資料平面 API 操作，請使用下列服務名稱：

```
com.amazonaws.region.iotsitewise.data
```

- 對於控制平面 API 操作，請使用下列服務名稱：

```
com.amazonaws.region.iotsitewise.api
```

AWS IoT SiteWise 透過介面 VPC 端點存取

當您建立介面端點時，我們會產生端點特定的 DNS 主機名稱，供您用來與之通訊 AWS IoT SiteWise。預設會啟用私有 DNS 選項。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[使用私有託管區域](#)。

如果您為端點啟用私有 DNS，您可以透過下列其中一個 VPC 端點向提出 API AWS IoT SiteWise 請求。

- 對於資料平面 API 操作，請使用下列端點：將##取代為您的 AWS 區域。

```
data.iotsitewise.region.amazonaws.com
```

- 對於控制平面 API 操作，請使用下列端點：將##取代為 AWS 區域。

```
api.iotsitewise.region.amazonaws.com
```

如果您停用端點的私有 DNS，您必須執行下列動作，才能 AWS IoT SiteWise 透過端點存取：

1. 在 API 請求中指定 VPC 端點 URL。

- 對於資料平面 API 操作，請使用下列端點 URL。將 *vpc-endpoint-id* 和##取代為您的 VPC 端點 ID 和區域。

```
vpc-endpoint-id.data.iotsitewise.region.vpce.amazonaws.com
```

- 對於控制平面 API 操作，請使用下列端點 URL。將 *vpc-endpoint-id* 和##取代為您的 VPC 端點 ID 和區域。

```
vpc-endpoint-id.api.iotsitewise.region.vpce.amazonaws.com
```

2. 停用主機字首注入。當您呼叫每個 API 操作時，AWS CLI and AWS SDKs會在服務端點前面加上各種主機字首。當您指定 VPC 端點 AWS IoT SiteWise 時，此功能會導致 AWS CLI AWS SDKs 產生不適用於 URLs。

 Important

您無法在 AWS CLI 或 中停用主機字首注入 AWS Tools for PowerShell。這表示如果您停用私有 DNS，則無法使用這些工具 AWS IoT SiteWise 透過 VPC 端點存取。啟用私有 DNS 以使用 AWS CLI 或 AWS IoT SiteWise 透過端點 AWS Tools for PowerShell 存取。

如需如何在 AWS SDKs 中停用主機字首注入的詳細資訊，請參閱每個 SDK 的下列文件章節：

- [適用於 C++ 的 AWS SDK](#)
- [適用於 Go 的 AWS SDK](#)
- [適用於 Go 的 AWS SDK v2](#)
- [適用於 Java 的 AWS SDK](#)

- [AWS SDK for Java 2.x](#)
- [適用於 JavaScript 的 AWS SDK](#)
- [適用於 .NET 的 AWS SDK](#)
- [適用於 PHP 的 AWS SDK](#)
- [適用於 Python \(Boto3\) 的 AWS SDK](#)
- [適用於 Ruby 的 AWS SDK](#)

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的[使用介面 VPC 端點存取 AWS 服務](#)。

建立的 VPC 端點政策 AWS IoT SiteWise

您可以將端點政策連接至控制 AWS IoT SiteWise 存取權限的 VPC 端點。此政策會指定下列資訊：

- 可執行操作的委託人。
- 可執行的操作。
- 可對其執行操作的資源。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的[使用端點政策控制 VPC 端點的存取](#)。

範例：AWS IoT SiteWise 動作的 VPC 端點政策

以下是端點政策的範例 AWS IoT SiteWise。連接到端點時，此政策會授予指定資產上 AWS 帳戶 **123456789012** *iotsitewiseadmin* 使用者所列出 AWS IoT SiteWise 動作的存取權。

```
{
  "Statement": [
    {
      "Action": [
        "iotsitewise:CreateAsset",
        "iotsitewise:ListGateways",
        "iotsitewise:ListTagsForResource"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:iotsitewise:us-west-2:123456789012:asset/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
      "Principal": {
        "AWS": [
          "123456789012:user/iotsitewiseadmin"
        ]
      }
    }
  ]
}
```

```
}  
  }  
}  
}
```

的安全最佳實務 AWS IoT SiteWise

本主題包含 的安全最佳實務 AWS IoT SiteWise。

在 OPC UA 伺服器上使用身分驗證憑證

需要身分驗證憑證才能連線至您的 OPC UA 伺服器。請參閱伺服器的文件以執行這項操作。然後，若要允許 SiteWise Edge 閘道連線至 OPC UA 伺服器，請將伺服器身分驗證秘密新增至 SiteWise Edge 閘道。如需詳細資訊，請參閱[設定 SiteWise Edge 的資料來源身分驗證](#)。

為您的 OPC UA 伺服器使用加密的通訊模式

當您為 SiteWise Edge 閘道設定 OPC UA 來源時，請選擇未取代的加密訊息安全模式。這有助於保護從 OPC UA 伺服器移至 SiteWise Edge 閘道的工業資料。如需詳細資訊，請參閱[透過區域網路傳輸資料](#)及在 [SiteWise Edge 中設定 OPC UA 來源](#)。

讓您的元件保持在最新狀態

如果您使用 SiteWise Edge 閘道將資料擷取至服務，則您有責任設定和維護 SiteWise Edge 閘道的環境。此責任包括升級至最新版本的閘道系統軟體、AWS IoT Greengrass 軟體和連接器。

Note

AWS IoT SiteWise Edge 連接器會將秘密存放在您的檔案系統上。這些秘密會控制誰可以檢視 SiteWise Edge 閘道中快取的資料。強烈建議您為執行 SiteWise Edge 閘道的系統開啟磁碟或檔案系統加密。

如需如何在 AWS IoT SiteWise 主控台中升級元件的詳細資訊，請參閱 [變更 SiteWise Edge 閘道元件套件的版本](#)。

加密 SiteWise Edge 閘道的檔案系統

加密並保護 SiteWise Edge 閘道，讓您的工業資料在通過 SiteWise Edge 閘道時安全無虞。如果您的 SiteWise Edge 閘道具有硬體安全模組，您可以設定 AWS IoT Greengrass 來保護 SiteWise Edge 閘

道。如需詳細資訊，請參閱《AWS IoT Greengrass Version 1 開發人員指南》中的[硬體安全整合](#)。您也可以參閱作業系統的文件，以了解如何加密和保護您的檔案系統。

安全存取您的邊緣組態

請勿共用您的邊緣主控台應用程式密碼或 SiteWise Monitor 應用程式密碼。請勿將此密碼放在任何人都可看見的地方。為您的密碼設定適當的過期時間，以實作運作狀態良好的密碼輪換政策。

保護上的資料 Siemens Industrial Edge Management

您選擇與 AWS IoT SiteWise Edge 共用的裝置資料取決於您的 Siemens IEM Databus 組態主題。透過選擇要與 SiteWise Edge 共用的主題，您可以共用主題層級的資料 AWS IoT SiteWise。Siemens Industrial Edge Marketplace 是一個獨立的市場，與分開 AWS。為了保護您的共用資料，除非您使用，否則 SiteWise Edge 應用程式將不會執行 Siemens Secured Storage。如需詳細資訊，請參閱 Siemens 文件中的[安全儲存](#)。

授予 SiteWise Monitor 使用者最低可能許可

使用入口網站使用者的最低存取政策許可集，以遵循最低權限原則。

- 建立入口網站時，請定義可允許該入口網站所需最低資產集的角色。如需詳細資訊，請參閱[使用的服務角色 AWS IoT SiteWise Monitor](#)。
- 當您和您的入口網站管理員建立並共用專案時，請使用該專案所需的最低資產集。
- 當身分不再需要存取入口網站或專案時，請從該資源中移除它們。如果該身分不再適用於您的組織，請從您的身分存放區刪除該身分。

最低原則最佳實務也適用於 IAM 角色。如需詳細資訊，請參閱[政策最佳實務](#)。

請勿公開敏感資訊

您應該避免記錄登入資料和其他敏感資訊，例如個人識別資訊 (PII)。即使存取 SiteWise Edge 閘道上的本機日誌需要根權限，而且存取 CloudWatch Logs 需要 IAM 許可，仍建議您實作下列保護措施。

- 請勿在資產或模型的名稱、說明或屬性中使用敏感資訊。
- 請勿在 SiteWise Edge 閘道或來源名稱中使用敏感資訊。
- 請勿在入口網站、專案或儀表板的名稱或描述中使用敏感資訊。

遵循 AWS IoT Greengrass 安全最佳實務

遵循 SiteWise Edge 闡道 AWS IoT Greengrass 的安全最佳實務。如需詳細資訊，請參閱《AWS IoT Greengrass Version 1 開發人員指南》中的[安全最佳實務](#)。

另請參閱

- 《AWS IoT 開發人員指南》中的[安全最佳實務](#)
- [工業 IoT 解決方案的十項安全黃金規則](#)

在 中記錄和監控 AWS IoT SiteWise

監控是維護 AWS IoT SiteWise 及其他 AWS 解決方案可靠性、可用性和效能的重要部分。AWS IoT SiteWise 支援下列監控工具來監看服務、在發生錯誤時回報，以及適時採取自動動作：

- Amazon CloudWatch AWS 會即時監控您的 AWS 資源和您在 上執行的應用程式。收集和追蹤指標、建立自訂儀表板，以及設定警示，在指定的指標達到特定閾值時通知您或採取動作。例如，您可以讓 CloudWatch 追蹤 CPU 使用量或其他 Amazon EC2 執行個體指標，並在需要時自動啟動新的執行個體。如需詳細資訊，請參閱 [Amazon CloudWatch 使用者指南](#)。
- Amazon CloudWatch Logs 會監控、存放和存取 SiteWise Edge 閘道、CloudTrail 和其他來源的日誌檔案。CloudWatch Logs 可監控日誌檔案中的資訊，並在達到特定閾值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 [Amazon CloudWatch Logs 使用者指南](#)。
- AWS CloudTrail 會擷取由您的帳戶發出或代表 AWS 您的帳戶發出的 API 呼叫和相關事件。然後 CloudTrail 會將日誌檔案交付至您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、進行呼叫的來源 IP 地址，以及呼叫的時間。如需詳細資訊，請參閱《[AWS CloudTrail 使用者指南](#)》。

主題

- [使用 Amazon CloudWatch Logs 監控](#)
- [監控 SiteWise Edge 閘道日誌](#)
- [AWS IoT SiteWise 使用 Amazon CloudWatch 指標進行監控](#)
- [使用 記錄 AWS IoT SiteWise API 呼叫 AWS CloudTrail](#)

使用 Amazon CloudWatch Logs 監控

設定 AWS IoT SiteWise 將資訊記錄到 CloudWatch Logs 以監控和疑難排解服務。

當您使用 AWS IoT SiteWise 主控台時，會 AWS IoT SiteWise 建立服務連結角色，允許服務代表您記錄資訊。如果您不使用 AWS IoT SiteWise 主控台，則必須手動建立服務連結角色才能接收日誌。如需詳細資訊，請參閱 [為 AWS IoT SiteWise 建立服務連結角色](#)。

您必須擁有允許 AWS IoT SiteWise 將日誌事件放入 CloudWatch 串流的資源政策。若要建立和更新 CloudWatch Logs 的資源政策，請執行下列命令。將 *logging-policy-name* 取代為要建立的 policy 名稱。

```
aws logs put-resource-policy --policy-name logging-policy-name --policy-
document "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Sid\":
  \"IoTSiteWiseToCloudWatchLogs\", \"Effect\": \"Allow\", \"Principal\": { \"Service\":
  [ \"iotsitewise.amazonaws.com\" ] }, \"Action\": \"logs:PutLogEvents\", \"Resource\":
  \"*\" } ] }"
```

CloudWatch Logs 也支援 [aws : SourceArn](#) 和 [aws : SourceAccount](#) 條件內容索引鍵。這些條件內容索引鍵是選用的。

若要建立或更新資源政策，允許 AWS IoT SiteWise 僅將與指定 AWS IoT SiteWise 資源相關聯的日誌放入 CloudWatch 串流，請執行 命令並執行下列動作：

- 將 *logging-policy-name* 取代為要建立的策略名稱。
- 將 *source-ARN* 取代為您 AWS IoT SiteWise 資源的 ARN，例如資產模型或資產。若要尋找每個 AWS IoT SiteWise 資源類型的 ARN，請參閱《服務授權參考》中的 [定義的資源類型 AWS IoT SiteWise](#)。
- 將 *account-ID* 取代為與指定 AWS IoT SiteWise 資源相關聯的 AWS 帳戶 ID。

```
aws logs put-resource-policy --policy-name logging-policy-name --policy-
document "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Sid\":
  \"IoTSiteWiseToCloudWatchLogs\", \"Effect\": \"Allow\", \"Principal\": { \"Service
  \": [ \"iotsitewise.amazonaws.com\" ] }, \"Action\": \"logs:PutLogEvents\", \"Resource
  \": \"*\", \"Condition\": { \"StringLike\": { \"aws:SourceArn\": [\"source-ARN\"],
  \"aws:SourceAccount\": [\"account-ID\"] } } } ] }"
```

根據預設，AWS IoT SiteWise 不會將資訊記錄到 CloudWatch Logs。若要啟用記錄，請選擇已停用 () 以外的記錄層級OFF。AWS IoT SiteWise 支援下列記錄層級：

- OFF – 記錄已關閉。
- ERROR – 記錄錯誤。
- INFO – 記錄錯誤和資訊訊息。

您可以設定 SiteWise Edge 閘道，透過 將資訊記錄到 CloudWatch Logs AWS IoT Greengrass。如需詳細資訊，請參閱[監控 SiteWise Edge 閘道日誌](#)。

如果您正在對 AWS IoT SiteWise 規則動作進行故障診斷，也可以設定 AWS IoT Core 將資訊記錄到 CloudWatch Logs。如需詳細資訊，請參閱[對 AWS IoT SiteWise 規則動作進行故障診斷](#)。

內容

- [管理登入 AWS IoT SiteWise](#)
 - [尋找您的記錄層級](#)
 - [變更您的記錄層級](#)
- [範例：AWS IoT SiteWise 日誌檔案項目](#)

管理登入 AWS IoT SiteWise

使用 AWS IoT SiteWise 主控台或 AWS CLI 執行下列記錄組態任務。

尋找您的記錄層級

Console

使用下列程序在 AWS IoT SiteWise 主控台中尋找目前的記錄層級。

尋找您目前的 AWS IoT SiteWise 記錄層級

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中，選擇 Logging options (記錄選項)。

目前的記錄狀態會顯示在 Logging status (記錄狀態) 下方。如果啟用記錄，目前的記錄層級會顯示在詳細程度下。

AWS CLI

執行下列命令，使用 尋找您目前的 AWS IoT SiteWise 記錄層級 AWS CLI。

```
aws iotsitewise describe-logging-options
```

此操作會以下列格式傳回包含您記錄層級的回應。

```
{
  "loggingOptions": {
    "level": "String"
  }
}
```

變更您的記錄層級

使用下列程序，在 AWS IoT SiteWise 主控台或使用 變更您的記錄層級 AWS CLI。

Console

變更您的 AWS IoT SiteWise 記錄層級

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在左側導覽窗格中，選擇 Logging options (記錄選項)。
3. 選擇編輯。
4. 選擇要啟用的詳細程度。
5. 選擇 Save (儲存)。

AWS CLI

執行下列 AWS CLI 命令來變更您的 AWS IoT SiteWise 記錄層級。將 *logging-level* 取代為您想要的記錄層級。

```
aws iotsitewise put-logging-options --logging-options level=logging-level
```

範例：AWS IoT SiteWise 日誌檔案項目

每個 AWS IoT SiteWise 日誌項目都包含該事件的事件資訊和相關資源，因此您可以了解和分析日誌資料。

下列範例顯示 CloudWatch Logs 項目，在您成功建立資產模型時 AWS IoT SiteWise 記錄該項目。

```
{
  "eventTime": "2020-05-05T00:10:22.902Z",
  "logLevel": "INFO",
  "eventType": "AssetModelCreationSuccess",
  "message": "Successfully created asset model.",
  "resources": {
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
  }
}
```

監控 SiteWise Edge 閘道日誌

您可以設定 AWS IoT SiteWise Edge 閘道，將資訊記錄到 Amazon CloudWatch Logs 或本機檔案系統。

主題

- [使用 Amazon CloudWatch Logs](#)
- [在 中使用服務日誌 AWS IoT SiteWise](#)

使用 Amazon CloudWatch Logs

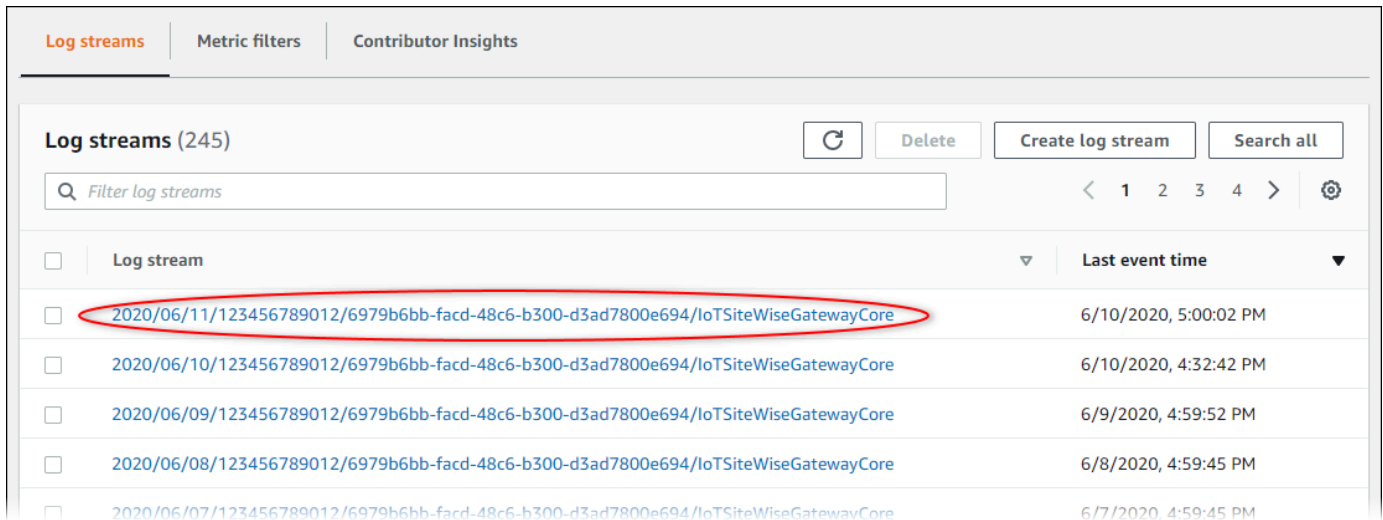
您可以設定 SiteWise Edge 閘道，將日誌傳送至 CloudWatch Logs。如需詳細資訊，請參閱《AWS IoT Greengrass Version 2 開發人員指南》中的[啟用 CloudWatch Logs 的記錄](#)。

設定和存取 CloudWatch Logs（主控台）

1. 導覽至 [CloudWatch 主控台](#)。
2. 在導覽窗格中，選擇 Log groups（日誌群組）。
3. 您可以在下列日誌群組中找到 AWS IoT SiteWise 元件日誌：
 - `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgeCollectorOpcua` – SiteWise Edge 閘道元件的日誌，從 SiteWise Edge 閘道的 OPC UA 來源收集資料。
 - `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgePublisher` – 發佈 OPC UA 資料串流的 SiteWise Edge 閘道元件日誌 AWS IoT SiteWise。

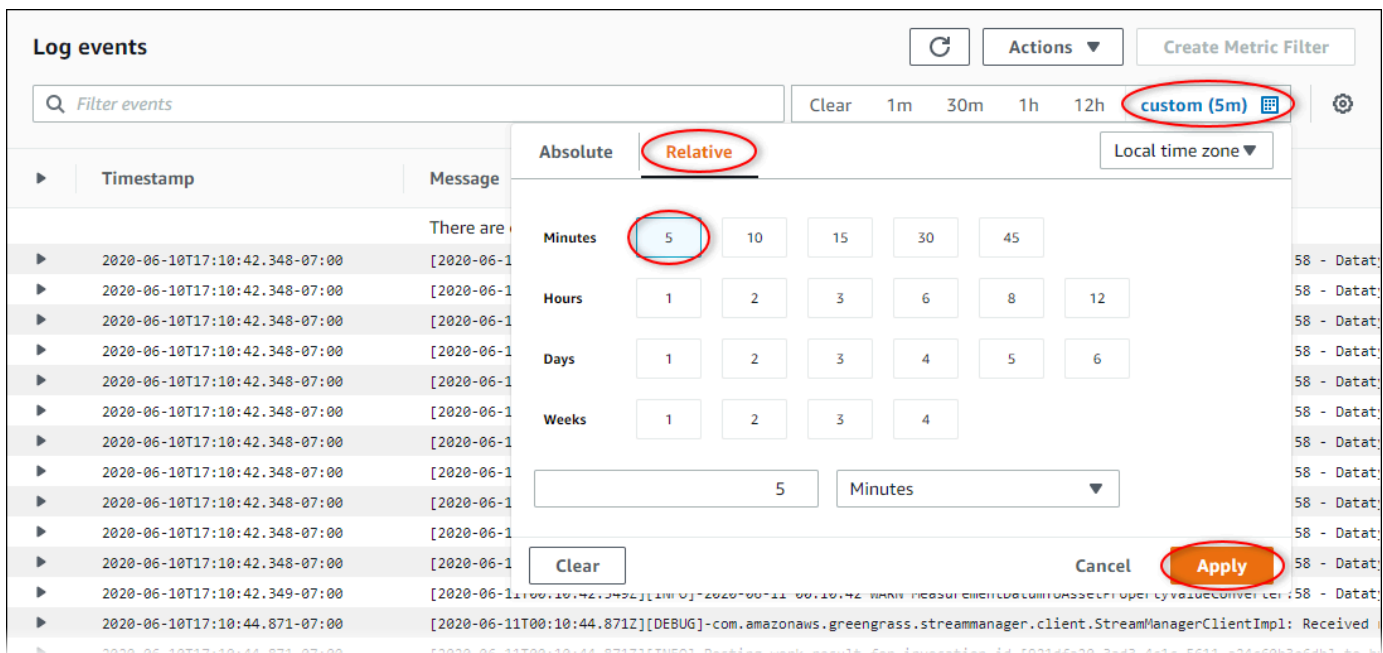
選擇函數要除錯的日誌群組。

4. 選擇名稱結尾為 AWS IoT Greengrass 群組名稱的日誌串流。根據預設，CloudWatch 會先顯示最新的日誌串流。



5. 若要顯示過去 5 分鐘的日誌，請執行下列動作：

- 選擇右上角的 custom (自訂)。
- 選擇 Relative (相對)。
- 選擇 5 分鐘。
- 選擇套用。



6. (選用) 若要查看較少的日誌，您可以從右上角選擇 1m (1 分鐘)。

7. 捲動至日誌項目底部以顯示最近的日誌。

在 中使用服務日誌 AWS IoT SiteWise

SiteWise Edge 閘道裝置包含服務日誌檔案，可協助偵錯問題。以下各節將協助您尋找和使用 AWS IoT SiteWise OPC UA Collector 和 AWS IoT SiteWise Publisher 元件的服務日誌檔案。

AWS IoT SiteWise OPC UA Collector 服務日誌檔案

AWS IoT SiteWise OPC UA 收集器元件使用以下日誌檔案。

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Windows

```
C:\igreengrassgreengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log
```

檢視此元件的日誌

- 在核心裝置上執行下列命令，以即時檢視此元件的日誌檔案。將 */greengrass/v2* 或 *C#\greengrass\v2* 取代為 AWS IoT Greengrass 根資料夾的路徑。

Linux

```
sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Windows (PowerShell)

```
Get-Content C:\igreengrassgreengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log -Tail  
10 -Wait
```

AWS IoT SiteWise 發佈者服務日誌檔案

AWS IoT SiteWise Publisher 元件使用以下日誌檔案。

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log
```

檢視此元件的日誌

- 在核心裝置上執行下列命令，以即時檢視此元件的日誌檔案。將 `/greengrass/v2` 或 `C:\greengrass\v2` 取代為 AWS IoT Greengrass 根資料夾的路徑。

Linux

```
sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log -Tail 10 -Wait
```

AWS IoT SiteWise 使用 Amazon CloudWatch 指標進行監控

您可以使用 AWS IoT SiteWise CloudWatch 監控，這會收集原始資料並將其處理為可讀且近乎即時的指標。這些統計資料會保留 15 個月，以便您存取歷史資訊，並更清楚 Web 應用程式或服務的執行效能。您也可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 [Amazon CloudWatch 使用者指南](#)。

AWS IoT SiteWise 會將下列各節中列出的指標和維度發佈至 AWS/IoTSiteWise 命名空間。

Tip

AWS IoT SiteWise 每隔一分鐘發佈指標。當您在 CloudWatch 主控台的圖表中檢視這些指標時，我們建議您選擇 1 分鐘的期間。這可讓您查看指標資料的最高可用解析度。

主題

- [AWS IoT Greengrass Version 2 閘道指標](#)

AWS IoT Greengrass Version 2 閘道指標

AWS IoT SiteWise 發佈 Classic 串流、V2 閘道和啟用 MQTT 的 V3 閘道的閘道指標。除非另有說明，否則每個指標都適用於兩個自我託管閘道版本。所有 SiteWise Edge 閘道指標都會以一分鐘的間隔發佈。

SiteWise Edge 閘道指標

指標	描述
Gateway.AvailableMemory	SiteWise Edge 閘道的可用記憶體。 單位：位元組 維度：無
Gateway.AvailableDiskSpace	SiteWise Edge 閘道的可用磁碟空間。 單位：位元組 維度：無
Gateway.CloudConnectivity	SiteWise Edge 閘道的雲端連線狀態。 單位：無 維度：GatewayId
Gateway.CpuUsage	SiteWise Edge 閘道的 CPU 用量。 單位：百分比 維度：無
Gateway.TotalDiskSpace	SiteWise Edge 閘道的總磁碟空間。 單位：位元組 維度：無
Gateway.TotalMemory	SiteWise Edge 閘道的總記憶體。 單位：位元組

指標	描述
	維度：無
Gateway.UsedDiskSpace	SiteWise Edge 閘道的已使用磁碟空間。 單位：位元組 維度：無
Gateway.UsedMemory	SiteWise Edge 閘道的已使用記憶體。 單位：位元組 維度：無
Gateway.UsedPercentageDiskSpace	SiteWise Edge 閘道已使用的磁碟空間百分比。 單位：位元組 維度：無
Gateway.UsedPercentageMemory	SiteWise Edge 閘道的已使用百分比記憶體。 單位：位元組 維度：無

AWS IoT SiteWise 發佈者指標

指標	描述
IoTSiteWisePublisher.ComponentBuildVersion	此指標表示在閘道上執行的 IoT SiteWise 發佈者元件建置版本。值 1 表示閘道正在執行與 ComponentBuildVersion 維度對應的發佈者版本。 單位：1 維度：GatewayId、ComponentBuildVersion

指標	描述
<code>IoTSiteWisePublisher.DroppedCount</code>	SiteWise Edge 閘道 (GatewayId) 捨棄且未發佈至雲端的資料點數量，每分鐘產生。 單位：計數 維度：GatewayId
<code>IoTSiteWisePublisher.Heartbeat</code>	由 SiteWise Edge 閘道中的發佈者每分鐘產生。 單位：1 (1 代表發佈者正在執行，而缺少代表發佈者的資料點未執行。) 維度：GatewayId
<code>IoTSiteWisePublisher.IsConnectedToMqttBroker</code>	由 SiteWise Edge 閘道中的發佈者每分鐘產生。 單位：1 (1 代表發佈者連接到 MQTT 代理程式。) 維度：GatewayId
<code>IoTSiteWisePublisher.MessageCheckpointPersistenceErrorCount</code>	指標表示閘道偵測到用於追蹤發佈者所處理資料的檢查點檔案有問題。值 1 表示發生故障。 單位：無 維度：AccountId、GatewayId
<code>IoTSiteWisePublisher.MqttMessageReceivedSuccessCount</code>	發佈者成功從 MQTT 代理程式接收的訊息數量，每分鐘產生。 單位：計數 維度：GatewayId

指標	描述
<code>IoTSiteWisePublisher.MqttReceivedSuccessBytes</code>	發佈者從 MQTT 代理程式成功接收的訊息資料的位元組數，每分鐘產生。 單位：計數 維度：GatewayId
<code>IoTSiteWisePublisher.NumberOfSubscriptionsToMqttBroker</code>	發佈者訂閱 MQTT 代理程式的主題數量，每分鐘產生。多層級萬用字元主題計為 1。 單位：計數 維度：GatewayId
<code>IoTSiteWisePublisher.NumberOfUniqueMqttTopicsReceived</code>	發佈者從 MQTT 代理程式接收的唯一主題數目，每分鐘產生。 單位：計數 維度：GatewayId
<code>IoTSiteWisePublisher.PublisherFailureCount</code>	SiteWise Edge 閘道 (GatewayId) 無法發佈的資料點數量，每分鐘產生一次。 單位：計數 維度：GatewayId
<code>IoTSiteWisePublisher.PublisherRejectedCount</code>	SiteWise Edge 閘道 (GatewayId) 從雲端拒絕的資料點數目，每分鐘產生。 單位：計數 維度：GatewayId

指標	描述
<code>IoTSiteWisePublisher.PublisherSuccessCount</code>	SiteWise Edge 閘道 (GatewayId) 每分鐘成功發佈到雲端的資料點數量。 單位：計數 維度：GatewayId
<code>IoTSiteWisePublisher.PublisherToS3FailureCount</code>	閘道 (GatewayId) 無法發佈至 Amazon S3 儲存貯體的資料點數目。 單位：計數 維度：GatewayId
<code>IoTSiteWisePublisher.PublisherToS3SuccessCount</code>	閘道 (GatewayId) 成功發佈至 Amazon S3 儲存貯體的資料點數目。 單位：計數 維度：GatewayId

OPC UA 收集器指標

指標	描述
<code>OpcUaCollector.ActiveDataStreamCount</code>	SiteWise Edge 閘道 (gatewayId) 為 OPC UA 來源 () 訂閱的資料串流數量sourceName 。 單位：計數 維度：GatewayId、SourceName、PropertyGroup
<code>OpcUaCollector.ComponentBuildVersion</code> (不適用於 Classic 串流、V2 閘道)	此指標表示在閘道上執行的 IoT SiteWise OPC UA 收集器元件建置版本。值 1 表示閘道正在執行與ComponentBuildVersion 維度對應的收集器版本。

指標	描述
	單位：1 維度：GatewayId、 ComponentBuildVersion
OpcUaCollector.ConversionErrors	SiteWise Edge 閘道 (gatewayId) 針對 OPC UA 來源 (sourceName) 接收的資料點數量，這會導致傳送資料時的轉換錯誤 AWS IoT SiteWise。OPC UA Collector 不會擷取這些資料點。 單位：計數 維度：GatewayId、 SourceName
OpcUaCollector.Heartbeat	每分鐘為連接到 SiteWise Edge 閘道 (sourceName) 的每個 OPC UA 來源 () 產生一次 gatewayId 。 單位：計數 (1 表示來源已連線，0 表示來源已中斷連線。) 維度：GatewayId、 SourceName
OpcUaCollector.IncomingValuesCount	SiteWise Edge 閘道 (gatewayId) 每分鐘針對 OPC UA 來源 (sourceName) 收到的資料點數量。 單位：計數 維度：GatewayId、 SourceName、 Pro PropertyGroup

指標	描述
<code>OpcUaCollector.IncomingValueErrors</code>	SiteWise Edge 閘道 (gatewayId) 從非有效值的 OPC UA 來源 (sourceName) 接收的資料點數目。OPC UA 收集器不會擷取這些資料點，每分鐘產生一次。 單位：計數 維度：GatewayId、SourceName、PropertyGroup
<code>OpcUaCollector.IsConnectedToMqttBroker</code> (不適用於 Classic 串流、V2 閘道)	每分鐘由 SiteWise Edge 閘道中的 IoT SiteWise OPC UA 收集器元件產生。 單位：1 (1 代表 IoT SiteWise OPC UA 收集器元件連接到 MQTT 代理程式) 維度：GatewayId
<code>OpcUaCollector.MqttMessagesDroppedCount</code> (不適用於 Classic 串流、V2 閘道)	IoT SiteWise OPC UA 收集器元件捨棄的 MQTT 訊息數量。 單位：計數 維度：GatewayId、SourceName
<code>OpcUaCollector.MqttMessagesPublishedBytes</code> (不適用於 Classic 串流、V2 閘道)	IoT SiteWise OPC UA 收集器元件成功發佈至 MQTT 代理程式的 MQTT 訊息資料位元組數。 單位：計數 維度：GatewayId、SourceName
<code>OpcUaCollector.MqttMessagesPublishedCount</code> (不適用於 Classic 串流、V2 閘道)	IoT SiteWise OPC UA 收集器元件成功發佈至 MQTT 代理程式的 MQTT 訊息數量。 單位：計數 維度：GatewayId、SourceName

指標	描述
<code>OpcUaCollector.NullValueCount</code> (不適用於 Classic 串流、V2 閘道)	IoT SiteWise OPC UA 收集器元件從 OPC UA 伺服器收到的 null 值數目。 單位：計數 維度：GatewayId、SourceName、PropertyGroup
<code>OpcUaCollector.NumberOfUniqueMqttTopicsPublished</code> (不適用於 Classic 串流、V2 閘道)	IoT SiteWise OPC UA 收集器發佈給 MQTT 代理程式的唯一 MQTT 主題數量。 單位：計數 維度：GatewayId、SourceName

AWS IoT SiteWise 處理器指標

指標	描述
<code>Gateway.DataProcessor.IngestionThrottled</code> (不適用於啟用 MQTT 的 V3 閘道)	已調節、每分鐘產生的資料點數量。 單位：計數 維度：ThrottledAt
<code>Gateway.DataProcessor.MeasurementRejected</code> (不適用於啟用 MQTT 的 V3 閘道)	每分鐘產生的已拒絕測量數。 單位：計數 維度：原因
<code>Gateway.DataProcessor.MessagesRemaining</code> (不適用於啟用 MQTT 的 V3 閘道)	串流中剩餘的訊息數量，每分鐘產生。 單位：計數 維度：StreamName

指標	描述
Gateway.DataProcessor.ProcessingError (不適用於啟用 MQTT 的 V3 閘道)	每分鐘產生的處理錯誤數量。 單位：計數 維度：原因

使用 記錄 AWS IoT SiteWise API 呼叫 AWS CloudTrail

AWS IoT SiteWise 已與 服務整合 AWS CloudTrail，此服務提供使用者、角色或 AWS 服務在其中採取之動作的記錄 AWS IoT SiteWise。CloudTrail 會擷取 AWS IoT SiteWise 的 API 呼叫當作事件。擷取的呼叫包括從 AWS IoT SiteWise 主控台的呼叫，以及對 AWS IoT SiteWise API 操作的程式碼呼叫。如果您建立線索，則可以啟用 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括的事件 AWS IoT SiteWise。即使您未設定追蹤，依然可以透過 CloudTrail 主控台的事件歷史記錄檢視最新事件。使用 CloudTrail 收集的資訊，您可以判斷提出的請求 AWS IoT SiteWise、提出請求的 IP 地址、提出請求的人員、提出請求的時間，以及其他詳細資訊。

如需有關 CloudTrail 的相關資訊，請參閱 [AWS CloudTrail 使用者指南](#)。

AWS IoT SiteWise CloudTrail 中的資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。在 中發生支援的事件活動時 AWS IoT SiteWise，該活動會記錄在 CloudTrail 事件中，以及事件歷史記錄中的其他服務 AWS 事件。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱[使用 CloudTrail 事件歷史記錄檢視事件](#)。

若要不持續記錄您 AWS 帳戶中的事件，包括的事件 AWS IoT SiteWise，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- [建立追蹤的概觀](#)
- [CloudTrail 支援的服務和整合](#)
- [設定 CloudTrail 的 Amazon SNS 通知](#)
- [接收多個區域的 CloudTrail 日誌檔案及接收多個帳戶的 CloudTrail 日誌檔案](#)

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：

- 請求是否使用根還是 AWS Identity and Access Management (IAM) 使用者登入資料提出。
- 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 userIdentity 元素](#)。

AWS IoT SiteWise CloudTrail 中的資料事件

[資料事件](#)提供在資源上或在資源中執行的資源操作的相關資訊 (例如，讀取或寫入 Amazon S3 物件)。這些也稱為資料平面操作。資料事件通常是大量資料的活動。根據預設，CloudTrail 不會記錄資料事件。CloudTrail 事件歷史記錄不會記錄資料事件。

資料事件需支付額外的費用。如需 CloudTrail 定價的詳細資訊，請參閱 [AWS CloudTrail 定價](#)。

您可以使用 CloudTrail 主控台 AWS CLI 或 CloudTrail API 操作來記錄 AWS IoT SiteWise 資源類型的資料事件。本節中的 [表格](#)顯示可用的資源類型 AWS IoT SiteWise。

- 若要使用 CloudTrail 主控台記錄資料事件，請建立 [追蹤](#)或[事件資料存放區](#)以記錄資料事件，或[更新現有的追蹤或事件資料存放區](#)以記錄資料事件。
 1. 選擇資料事件以記錄資料事件。
 2. 從資料事件類型清單中，選擇您要記錄資料事件的資源類型。
 3. 選擇您要使用的日誌選取器範本。您可以記錄資源類型的所有資料事件、記錄所有readOnly事件、記錄所有writeOnly事件，或建立自訂日誌選取器範本，以篩選 readOnly、eventName和 resources.ARN 欄位。
- 若要使用 記錄資料事件 AWS CLI，請設定 --advanced-event-selectors 參數以將 eventCategory 欄位設定為 Data並將 resources.type 欄位設定為 資源類型值（請參閱 [資料表](#)）。您可以新增條件來篩選 readOnly、eventName和 resources.ARN 欄位的值。
 - 若要設定線索記錄資料事件，請執行 [AWS CloudTrail put-event-selectors](#) 命令。如需詳細資訊，請參閱[使用 記錄線索的資料事件 AWS CLI](#)。
 - 若要設定事件資料存放區來記錄資料事件，請執行 [AWS CloudTrail create-event-data-store](#) 命令來建立新的事件資料存放區以記錄資料事件，或執行 [AWS CloudTrail update-event-data-store](#) 命令來更新現有的事件資料存放區。如需詳細資訊，請參閱[使用 記錄事件資料存放區的資料事件 AWS CLI](#)。

下表列出 AWS IoT SiteWise 資源類型。資料事件類型 (主控台) 資料行會顯示從 CloudTrail 主控台上的資料事件類型清單中選擇的值。resources.type 值欄會顯示值，您會在使用 AWS CLI 或 CloudTrail APIs 設定進階事件選取器時指定該resources.type值。記錄到 CloudTrail 的資料 API 資料行會針對資源類型顯示記錄到 CloudTrail 的 API 呼叫。

資料事件類型 (主控台)	resources.type 值	記錄到 CloudTrail 的資料 APIs*
AWS IoT SiteWise 資產	AWS::IoTSiteWise::Asset	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory • BatchGetAssetPropertyAggregates
AWS IoT SiteWise 時間序列	AWS::IoTSiteWise::TimeSeries	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue

資料事件類型 (主控台)	resources.type 值	記錄到 CloudTrail 的資料 APIs*
		BatchGetAssetPropertyHistory BatchGetAssetPropertyAggregates
AWS IoT SiteWise 助理	AWS::SitewiseAssistant::Conversation	InvokeAssistant

Note

Cloudtrail 事件中記錄的 resources.type 取決於 API 請求中使用的識別符。如果在請求中指定資產 ID，則會記錄 Asset resources.type，否則會記錄 TimeSeries resources.type。

*您可以設定進階事件選取器來篩選 eventName、和 resources.ARN 欄位readOnly，以僅記錄對您重要的事件。如需有關這些欄位的詳細資訊，請參閱 [AdvancedFieldSelector](#)。

AWS IoT SiteWise CloudTrail 中的 管理事件

[記錄管理事件](#)提供對 AWS 帳戶中資源執行的管理操作的相關資訊。這些也稱為控制平面操作。根據預設，CloudTrail 記錄管理事件。

AWS IoT SiteWise 會將所有 AWS IoT SiteWise 控制平面操作記錄為管理事件。如需 AWS IoT SiteWise 記錄到 CloudTrail 的 AWS IoT SiteWise 控制平面操作清單，請參閱 [AWS IoT SiteWise API 參考](#)。

範例：AWS IoT SiteWise 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌專案。一個事件代表任何來源提出的單一請求，並包含所請求之操作的相關資訊、操作的日期和時間、請求參數等等。CloudTrail 日誌檔案並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的 CloudTrail 日誌項目會示範 CreateAsset 操作：

```
{
```

```

"eventVersion": "1.05",
"userIdentity": {
  "type": "IAMUser",
  "principalId": "AIDACKCEVSQ6C2EXAMPLE",
  "arn": "arn:aws:iam::123456789012:user/Administrator",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "userName": "Administrator",
  "sessionContext": {
    "sessionIssuer": {},
    "webIdFederationData": {},
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2020-03-11T17:26:40Z"
    }
  },
  "invokedBy": "signin.amazonaws.com"
},
"eventTime": "2020-03-11T18:01:22Z",
"eventSource": "iotsitewise.amazonaws.com",
"eventName": "CreateAsset",
"awsRegion": "us-east-1",
"sourceIPAddress": "203.0.113.0",
"userAgent": "signin.amazonaws.com",
"requestParameters": {
  "assetName": "Wind Turbine 1",
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "clientToken": "a1b2c3d4-5678-90ab-cdef-00000EXAMPLE"
},
"responseElements": {
  "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetArn": "arn:aws:iotsitewise:us-east-1:123456789012:asset/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetStatus": {
    "state": "CREATING"
  }
},
"requestID": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
"eventID": "a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}

```

標記您的 AWS IoT SiteWise 資源

標記您的 AWS IoT SiteWise 資源提供了一種強大的方法來有效率地分類、管理和擷取組織資產。透過指派由鍵/值對組成的標籤，您可以將描述性中繼資料連接到資源。標籤的中繼資料可用來簡化操作。例如，在風力發電廠案例中，標籤可讓您使用位置、容量和操作狀態等特定屬性來標記渦輪機，以便在其中快速識別和管理 AWS IoT SiteWise。

將標籤與 AWS Identity and Access Management (IAM) 政策整合，可透過定義條件式存取規則來增強安全性和操作控制。這表示您可以指定只有具有特定標籤的使用者。例如，只有標記特定角色或部門的那些才能存取或修改特定資源。

在 中使用標籤 AWS IoT SiteWise

使用標籤，依用途、擁有者、環境或您的使用案例的任何其他分類來分類您的 AWS IoT SiteWise 資源。當您有許多相同類型的資源時，您可以依據先前指派的標籤來快速識別特定資源。

每個標籤都由您指定的索引鍵和選用值組成。例如，您可以為資產模型建立一系列標籤，以根據其支援的工業程序來追蹤它們。建議您為您管理的每種資源類型開發一組量身打造的標籤金鑰。使用一組一致的標籤索引鍵可以讓您更輕鬆地管理資源。

使用 標記 AWS Management Console

中的標籤編輯器 AWS Management Console 提供集中、統一的方式，讓您建立和管理來自所有 AWS 服務之資源的標籤。如需詳細資訊，請參閱《標記 AWS 資源和[標籤編輯器使用者指南](#)》中的[標籤編輯器入門](#)。

使用 AWS IoT SiteWise API 標記

AWS IoT SiteWise API 也會使用標籤。在建立標籤之前，請注意下列標籤限制。如需詳細資訊，請參閱《》中的[標籤命名和使用慣例](#)AWS 一般參考。

- 若要在建立資源時新增標籤，請在資源的 tags 屬性中定義標籤。
- 若要將標籤新增至現有資源或更新標籤值，請使用 [TagResource](#) 操作。
- 若要從資源移除標籤，請使用 [UntagResource](#) 操作。
- 若要擷取與資源建立關聯的標籤，請使用 [ListTagsForResource](#) 操作，或描述資源並檢查其 tags 屬性。

下表列出您可以使用 AWS IoT SiteWise API 及其對應 Create 和 Describe 操作來標記的資源。

可標記 AWS IoT SiteWise 的資源

資源	建立操作	描述操作
資產模型或元件模型	CreateAssetModel	DescribeAssetModel
資產	CreateAsset	DescribeAsset
SiteWise Edge 閘道	CreateGateway	DescribeGateway
Portal	CreatePortal	DescribePortal
專案	CreateProject	DescribeProject
儀表板	CreateDashboard	DescribeDashboard
存取政策	CreateAccessPolicy	DescribeAccessPolicy
時間序列	BatchPutAssetPropertyValue	DescribeTimeSeries

對於 [BatchPutAssetPropertyValue](#)，您可以設定資料來源，在建立資產模型和資產 AWS IoT SiteWise 之前將工業資料傳送至。AWS IoT SiteWise 會自動建立資料串流，以接收來自設備的原始資料串流。如需詳細資訊，請參閱[管理資料擷取](#)。

使用下列操作檢視並管理支援標記之資源的標籤：

- [TagResource](#) – 將標籤新增至資源，或更新現有標籤的值。
- [ListTagsForResource](#) – 列出資源的標籤。
- [UntagResource](#) – 從資源移除標籤。

隨時從資源新增或移除標籤。若要更新現有標籤索引鍵的值，請將具有相同索引鍵和所需新值的新標籤新增至資源。此動作會將舊值取代為新的值。雖然可以將空字串指派為標籤值，但您無法指派 null 值。

刪除資源也會移除與其連結的任何標籤。

搭配 IAM 政策使用標籤

在 IAM 政策中使用資源標籤來控制使用者存取和許可。例如，政策可讓使用者只建立已連接特定標籤的資源。政策也可以限制使用者建立或修改具有特定標籤的資源。

Note

如果您使用標籤來允許或拒絕使用者存取資源，您應該拒絕使用者為相同資源新增或移除這些標籤的能力。否則，使用者可以透過修改其標籤來繞過您的限制並存取資源。

您可以在政策陳述式的 Condition 元素 (也稱為 Condition 區塊) 中使用下列條件內容索引鍵和值。

`aws:ResourceTag/tag-key: tag-value`

允許或拒絕資源對具有特定標籤之資源的動作。

`aws:RequestTag/tag-key: tag-value`

在建立或修改可標記資源時，請求使用 (或不使用) 特定標籤。

`aws:TagKeys: [tag-key, ...]`

在建立或修改可標記資源時，請求使用 (或不使用) 一組特定標籤鍵。

Note

IAM 政策中的條件內容索引鍵和值僅適用於具有可標記資源作為必要參數的動作。例如，您可以為 [ListAssets](#) 設定標籤型條件式存取。您無法在 [PutLoggingOptions](#) 上設定標籤型條件式存取，因為在請求中沒有參考可標記資源。

如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 AWS 資源標籤和 IAM JSON 政策參考控制對資源的存取](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies.html)。 https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies.html

使用標籤的範例 IAM 政策

- [根據標籤檢視 AWS IoT SiteWise 資產](#)

故障診斷 AWS IoT SiteWise

使用下列資訊對 的問題進行疑難排解 AWS IoT SiteWise。

主題

- [SiteWise Edge 閘道故障診斷](#)
- [入口網站故障診斷 AWS IoT SiteWise](#)
- [對 AWS IoT SiteWise 規則動作進行故障診斷](#)
- [對大量匯入和匯出操作進行故障診斷](#)

SiteWise Edge 閘道故障診斷

探索相關主題，針對常見的 AWS IoT SiteWise Edge 閘道問題進行故障診斷。

您也可以檢視 SiteWise Edge 閘道報告的 CloudWatch 指標，以疑難排解連線或資料串流的問題。如需詳細資訊，請參閱[AWS IoT SiteWise 使用 Amazon CloudWatch 指標進行監控](#)。

主題

- [設定和存取 SiteWise Edge 閘道日誌](#)
- [故障診斷 SiteWise Edge 閘道問題](#)
- [對 上的 AWS IoT SiteWise Edge 應用程式進行故障診斷 Siemens Industrial Edge](#)
- [對 Edge 的開放原始碼整合進行故障診斷](#)
- [故障診斷 AWS IoT Greengrass 問題](#)

設定和存取 SiteWise Edge 閘道日誌

您必須先設定 SiteWise Edge 閘道將日誌傳送至 Amazon CloudWatch Logs 或在本機檔案系統上存放日誌，才能檢視 SiteWise Edge 閘道日誌。SiteWise

- 如果您想要使用 檢視 SiteWise Edge 閘道的日誌檔案 AWS Management Console ，請使用 CloudWatch Logs。SiteWise 如需詳細資訊，請參閱[使用 Amazon CloudWatch Logs](#)。
- 如果您想要使用命令列或本機軟體來檢視 SiteWise Edge 閘道的日誌檔案，請使用本機檔案系統日誌。如需詳細資訊，請參閱[在 中使用服務日誌 AWS IoT SiteWise](#)。

故障診斷 SiteWise Edge 閘道問題

使用以下資訊對 SiteWise Edge 閘道問題進行故障診斷。

問題

- [無法將套件部署至 SiteWise Edge 閘道](#)
- [AWS IoT SiteWise 不會從 OPC UA 伺服器接收資料](#)
- [儀表板中未顯示任何資料](#)
- [顯示在 /greengrass/v2/logs 的 aws.iot.SiteWiseEdgePublisher 日誌中的「找不到或載入主類別」錯誤](#)
- [我看到 'SESSION_TAKEN_OVER' 或 'com.aws.greengrass.mqttclient.MqttClient：無法透過 Spooler 發佈訊息，將會重試。' 日誌中的](#)
- [我看到 'com.aws.greengrass.deployment.IotJobsHelper：找不到部署任務。' 或 '部署結果已回報。' 日誌中的](#)
- [嘗試在 OPC UA 資料來源的屬性群組中設定時間戳記設定時，我看到「SYNC_FAILED」狀態](#)
- [不包含轉換後的資料類型](#)
- [信任存放區問題](#)
- [啟用代理的安裝問題](#)

無法將套件部署至 SiteWise Edge 閘道

如果核心元件 AWS IoT Greengrass (aws.greengrass.Nucleus) 已過時，您可能無法將套件部署至 SiteWise Edge 閘道。您可以使用 AWS IoT Greengrass V2 主控台來升級 nucleus AWS IoT Greengrass 元件。

升級 nucleus AWS IoT Greengrass 元件（主控台）

1. 導覽至 [AWS IoT Greengrass 主控台](#)。
2. 在導覽窗格中的下AWS IoT Greengrass，選擇部署。
3. 在部署清單中，選取您要修改的部署。
4. 選擇修訂。
5. 在指定目標頁面上，選擇下一步。
6. 在選取元件頁面的公有元件下，於搜尋方塊中輸入 **aws.greengrass.Nucleus**，然後選取 aws.greengrass.Nucleus。

7. 選擇下一步。
8. 在設定元件頁面上，選擇下一步。
9. 在設定進階設定頁面上，選擇下一步。
10. 在 Review (檢閱) 頁面，選擇 Deploy (部署)。

AWS IoT SiteWise 不會從 OPC UA 伺服器接收資料

如果您的 AWS IoT SiteWise 資產未接收 OPC UA 伺服器傳送的資料，您可以搜尋 SiteWise Edge 閘道的日誌來排除問題。尋找包含下列訊息的資訊層級swPublisher日誌。

```
Emitting diagnostic name=PublishError.SomeException
```

根據日誌中的 *SomeException* 類型，請使用下列例外狀況類型和對應的問題來疑難排解 SiteWise Edge 閘道：

- ResourceNotFoundException – OPC UA 伺服器正在傳送不符合任何資產屬性別名的資料。此例外狀況可能會發生在兩種情況下：
 - 您的屬性別名與您的 OPC UA 變數不完全相符，包括您定義的任何來源字首。檢查您的屬性別名和來源前綴是否正確。
 - 您尚未將 OPC UA 變數映射至資產屬性。如需詳細資訊，請參閱[管理 的資料串流 AWS IoT SiteWise](#)。

如果您已映射所有想要的 OPC UA 變數 AWS IoT SiteWise，您可以篩選 SiteWise Edge 閘道傳送的 OPC UA 變數。如需詳細資訊，請參閱[在 SiteWise Edge 中使用 OPC UA 節點篩選條件](#)。

- InvalidRequestException – 您的 OPC UA 變數資料類型不符合您的資產屬性資料類型。例如，如果 OPC UA 變數具有整數資料類型，則對應的資產屬性必須是整數資料類型。雙類型資產屬性無法接收 OPC UA 整數值。若要修正此問題，請使用正確的資料類型定義新的屬性。
- TimestampOutOfRangeException – SiteWise Edge 閘道正在傳送超出 AWS IoT SiteWise 接受範圍的資料。AWS IoT SiteWise 拒絕過去 7 天內或未來 5 分鐘內具有時間戳記的任何資料點。如果您的 SiteWise Edge 閘道失去電源或與 AWS 雲端的連線，您可能需要清除 SiteWise Edge 閘道的快取。
- ThrottlingException 或 LimitExceededException – 您的請求超過 AWS IoT SiteWise 服務配額，例如擷取資料點的速率或資產屬性資料 API 操作的請求速率。核對您的組態沒有超過 [AWS IoT SiteWise 配額](#)。

儀表板中未顯示任何資料

如果您的儀表板中沒有顯示任何資料，則 SiteWise Edge 閘道的發佈者組態和資料來源可能會不同步。如果不同步，更新資料來源的名稱可能會加速從雲端到邊緣的同步，修正不同步錯誤。

更新資料來源的名稱

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取連線至儀表板的 SiteWise Edge 閘道。
4. 在資料來源下，選取編輯。
5. 選取新的來源名稱，然後選取儲存以確認您的變更。
6. 確認資料來源資料表中的資料來源名稱已更新，以驗證您的變更。

顯示在 /greengrass/v2/logs 的 aws.iot.SiteWiseEdgePublisher 日誌中的「找不到或載入主類別」錯誤

如果您看到此錯誤，您可能需要更新 SiteWise Edge 閘道的 Java 版本。

- 從終端機執行下列命令：

```
java -version
```

SiteWise Edge 閘道執行的 Java 版本會顯示在下 OpenJDK Runtime Environment。您將看到如下所示的回應：

```
openjdk version "11.0.20" 2023-07-18 LTS
OpenJDK Runtime Environment Corretto011.0.20.8.1 (build 11.0.20+8-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.20.8.1 (build 11.0.20+8-LTS, mixed mode)
```

如果您執行的是 Java 11.0.20.8.1 版，則必須將 IoT SiteWise Publisher 套件更新至 2.4.1 版或更新版本。只有 Java 11.0.20.8.1 版受到影響，具有其他 Java 版本的環境可以繼續使用舊版 IoT SiteWise Publisher 元件。如需更新元件套件的詳細資訊，請參閱 [變更 SiteWise Edge 閘道元件套件的版本](#)。

我看到 'SESSION_TAKEN_OVER' 或 'com.aws.greengrass.mqttclient.MqttClient : 無法透過 Spooler 發佈訊息，將會重試。' 日誌中的

如果您在 的日誌 `com.aws.greengrass.mqttclient.MqttClient: Failed to publish the message via Spooler and will retry.` 中看到包含 的警告 `SESSION_TAKEN_OVER` 或包含 的錯誤 `/greengrass/v2/logs/greengrass.log`，您可能會嘗試在多個裝置上對多個 SiteWise Edge 閘道使用相同的組態檔案。每個 SiteWise Edge 閘道都需要唯一的組態檔案才能連線到 AWS 您的帳戶。

我看到 'com.aws.greengrass.deployment.IotJobsHelper : 找不到部署任務。' 或 '部署結果已回報。' 日誌中的

如果您在 的日誌 `Deployment result already reported.` 中看到 `com.aws.greengrass.deployment.IotJobsHelper: No deployment job found.` 或 `/greengrass/v2/logs/greengrass.log`，您可能會嘗試重複使用相同的組態檔案。

有多個解決方案：

- 如果您想要重複使用組態檔案，請執行下列動作：

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選擇您要重複使用的 SiteWise Edge 閘道。
4. 選擇更新索引標籤。
5. 選取不同的發佈者版本，然後選擇部署。

請依照中的步驟[建立的閘道 Siemens Industrial Edge](#)建立新的組態檔案。

嘗試在 OPC UA 資料來源的屬性群組中設定時間戳記設定時，我看到「SYNC_FAILED」狀態

在 2.5.0 版 AWS IoT Greengrass 中 AWS IoT SiteWise 更新的 OPC UA 收集器元件時，我們推出了新的時間戳記組態選項。您可以使用來自裝置的時間戳記，或來自伺服器的時間戳記。OPC UA 收集器元件的較舊版本不支援此選項，且無法同步。

有兩種方法可以解決失敗的資料來源同步狀態。建議的方式是將 IoT SiteWise OPC UA 收集器元件升級至 2.5.0 版或更新版本。或者，如果您將時間戳記設定為 `SERVER`，則可以繼續使用較舊的 OPC UA 收集

器元件版本Source。若要了解如何升級 IoT SiteWise OPC UA 收集器元件，請參閱 [更新 AWS IoT SiteWise 元件的版本](#)。建議使用所有元件的最新版本。

Note

當資料來源同步狀態失敗時，不會中斷資料。來源資料會繼續流入 AWS IoT SiteWise。組態不會與您 AWS IoT Greengrass V2 部署上的 IoT SiteWise OPC UA 收集器元件同步。

若要變更屬性群組的時間戳記組態

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取要編輯的閘道。
4. 在資料來源區段中，選取具有失敗同步狀態的資料來源，然後選擇編輯。
5. 展開進階組態，然後展開群組設定。
6. 在時間戳記中，選取來源。選取來源會從組態中移除 timestampToReturn 屬性。此設定預設可從您的裝置收集資料來源時間戳記，允許資料來源與 IoT SiteWise OPC UA 收集器元件同步。
7. 選擇儲存。

不包含轉換後的資料類型

如果您在將不支援的 OPC UA 資料類型轉換為 字串時看到錯誤 AWS IoT SiteWise，有幾個可能的原因：

- 您嘗試轉換的資料類型是複雜的資料類型。不支援複雜的資料類型。
- 使用目的地做為AWS IoT SiteWise 緩衝使用 Amazon S3 時，完整字串值會保留在推送至 Amazon S3 儲存貯體的檔案中。當您稍後將資料擷取到 時 AWS IoT SiteWise，超過 1024 個位元組的完整字串值會遭到拒絕。

信任存放區問題

如果您在 SiteWise Edge 中遇到與信任存放區相關的問題，請考慮下列疑難排解步驟：

- 驗證 AWS IoT Greengrass 根 CA 憑證是否存在，並在適當的信任存放區中正確格式化
- 確定 Java KeyStore 密碼已正確設定且可供 SiteWise Edge 元件存取

- 檢查任何自訂憑證（例如 HTTPS 代理）的格式是否正確（通常是 PEM），並正確匯入信任存放區
- 確認信任存放區具有正確的檔案許可，並且可供 SiteWise Edge 程序存取
- 檢閱 SiteWise Edge 日誌是否有任何 SSL/TLS 相關錯誤，這可能表示信任存放區問題
- 使用 `openssl` 等工具獨立測試 SSL/TLS 連線，以驗證信任存放區功能

啟用代理的安裝問題

如果您在代理組態程序期間遇到問題，請考慮下列疑難排解步驟：

- 確認代理 URL 已正確格式化，並包含適當的結構描述 (`http://` 或 `https://`)
- 如果代理登入資料包含特殊字元，請確保任何代理登入資料都以 URL 編碼
- 確認無代理清單包含所有必要的本機地址 AWS 和服務端點
- 對於 HTTPS 代理，請確認提供的 CA 憑證為 PEM 格式
- 檢閱安裝日誌是否有可能指出問題來源的特定錯誤訊息
- 獨立測試代理連線，以確保其正常運作

對上的 AWS IoT SiteWise Edge 應用程式進行故障診斷 Siemens Industrial Edge

若要對 Siemens Industrial Edge 裝置上的 AWS IoT SiteWise Edge 應用程式進行故障診斷，您可以透過 Siemens Industrial Edge Management 或 Siemens Industrial Edge Device (IED) 入口網站存取應用程式的日誌。如需詳細資訊，請參閱 Siemens 文件中的 [下載日誌](#)。

我的資料不會顯示在 中 AWS IoT SiteWise

- 確保您的 Databus 使用者沒有問題，且 Databus_Configuration 的核取記號圖示為綠色而非灰色。
- 您可能未在包含的版本 Siemens Industrial Edge Management 上執行 Secure Storage。升級您的 Siemens 作業系統版本。如需詳細資訊，請參閱 [Siemens Secure Storage](#) 和 [AWS IoT SiteWise Edge 應用程式](#)。

我在日誌中看到「Config 檔案缺少 AWS_REGION」

如果您在 Siemens 日誌 Config file missing AWS_REGION 中看到，組態檔案的 JSON 已損毀。您需要建立新的組態檔案。請依照中的步驟 [建立的開道 Siemens Industrial Edge](#) 建立新的組態檔案。

我在 Edge 閘道組態上看到「不同步」錯誤訊息

如果您在部署完成後在Siemens Industrial Edge閘道上看到Out of sync錯誤訊息，表示IoT SiteWise 發佈者元件與您的閘道不同步。IoT SiteWise 發佈者元件可在Siemens Industrial Edge閘道的背景運作，以提供 MQTT 主題功能。我們升級了Siemens Industrial Edge閘道以使用功能命名空間*iotsitewise:publisher:3*，而不是 *iotsitewise:publisher:2*。您可以更新至發佈者的最新版本，以解決此問題。

升級至最新版本的 IoT SiteWise 發佈者

1. 導覽至 [AWS IoT SiteWise 主控台](#)。
2. 在導覽窗格中，選擇 Edge 閘道。
3. 選取要編輯的Siemens Industrial Edge閘道。
4. 在 Edge 功能區段中，選取檢視軟體版本。
5. 在發佈者下拉式功能表下，選取最新版本的 IoT SiteWise 發佈者。
6. 選擇完成。

對 Edge 的開放原始碼整合進行故障診斷

本節提供將開放原始碼工具與 SiteWise Edge 整合時可能遇到的常見問題的解決方案。

Note

Node-RED®、InfluxDB® 和 Grafana® 不是 SiteWise Edge 的廠商或供應商。

連線問題

Node-RED 無法連線至 MQTT 代理程式

確認 MQTT 代理程式正在執行，並在指定的連接埠上存取。檢查您的網路組態，並確保代理程式地址正確無誤。

若要驗證 MQTT 代理程式狀態，請執行：

```
docker ps | grep emqx
```

InfluxDB 連線錯誤

請確定您的身分驗證字符有效，且您已指定正確的組織和儲存貯體名稱。檢查 InfluxDB 是否正在執行並可存取。

若要驗證 InfluxDB 狀態，請執行：

```
curl -I http://localhost:8086
```

Grafana 無法連線至 InfluxDB

驗證 Grafana 中的 InfluxDB 資料來源組態是否正確，包括 URL、身分驗證字符、組織和儲存貯體。

資料流程問題

中沒有資料出現 AWS IoT SiteWise

檢查 Node-RED 流程中的屬性別名是否符合預期的格式。確認 MQTT 主題結構正確，且 SiteWise Edge 閘道已正確設定為從 MQTT 代理程式接收資料。

沒有存放在 InfluxDB 中的 SiteWise Edge 資料

確認 Node-RED 保留流程已正確設定，且 InfluxDB 寫入器節點具有適當的儲存貯體和測量設定。檢查 Node-RED 偵錯輸出是否有任何錯誤。

資料格式錯誤

確保您的資料轉換功能可在格式之間正確轉換資料。使用 Node-RED 偵錯節點來檢查流程每個階段的資料。

效能問題

CPU 或記憶體用量高

監控資源用量，並視需要調整元件的組態。考慮降低資料收集頻率或實作資料篩選，以減少處理負載。

若要監控資源用量，請執行：

```
docker stats
```

慢速載入 Grafana 儀表板

最佳化您的 InfluxDB 查詢，並考慮將時間範圍限制新增至儀表板面板。使用適當的彙總函數減少顯示的資料點數量。

記錄和診斷

若要疑難排解問題，請檢查每個元件的日誌：

Node-RED 日誌

在 Node-RED 主控台中檢視日誌或執行：

```
docker logs node-red
```

InfluxDB 日誌

透過執行 存取日誌：

```
docker logs influxdb
```

Grafana 日誌

執行下列動作來檢視日誌：

```
docker logs grafana
```

SiteWise Edge 日誌

檢查 SiteWise Edge 閘道日誌是否有 MQTT 連線和資料處理問題。如需詳細資訊，請參閱[SiteWise Edge 閘道故障診斷](#)。

故障診斷 AWS IoT Greengrass 問題

若要尋找設定或部署 SiteWise Edge 閘道的許多問題的解決方案 AWS IoT Greengrass，請參閱《AWS IoT Greengrass 開發人員指南》中的[故障診斷 AWS IoT Greengrass](#)。

入口網站故障診斷 AWS IoT SiteWise

對 AWS IoT SiteWise 入口網站的常見問題進行故障診斷。

使用者和管理員無法存取 AWS IoT SiteWise 入口網站

如果使用者或管理員無法存取您的 AWS IoT SiteWise 入口網站，您可能在連接 AWS Identity and Access Management (IAM) 政策中具有限制許可，以防止成功登入。

請參閱下列會導致登入失敗的 IAM 政策範例：

Note

任何包含 "Condition" 元素的連接 IAM 政策都會導致登入失敗。

範例 1：此處的條件是有限的 IP，這會導致登入失敗。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:DescribePortal"
      ],
      "Resource": "*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "REPLACESAMPLEIP"
          ]
        }
      }
    }
  ]
}
```

範例 2：此處的條件是包含的標籤，這會導致登入失敗。

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

```
    "Action": [
      "iotsitewise:DescribePortal"
    ],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "aws:ResourceTag/project": "*"
      }
    }
  }
]
```

將使用者或管理員新增至入口網站時，請避免建立限制使用者許可的 IAM 政策，例如有限的 IP。任何具有受限許可的連接政策將無法連線至 AWS IoT SiteWise 入口網站。

對 AWS IoT SiteWise 規則動作進行故障診斷

若要對 中的 AWS IoT SiteWise 規則動作進行故障診斷 AWS IoT Core，您可以執行下列其中一個程序：

- 設定 Amazon CloudWatch Logs
- 為規則設定重新發佈錯誤動作

接下來，將錯誤訊息與本主題所述的錯誤相比較，對您的問題進行疑難排解。

主題

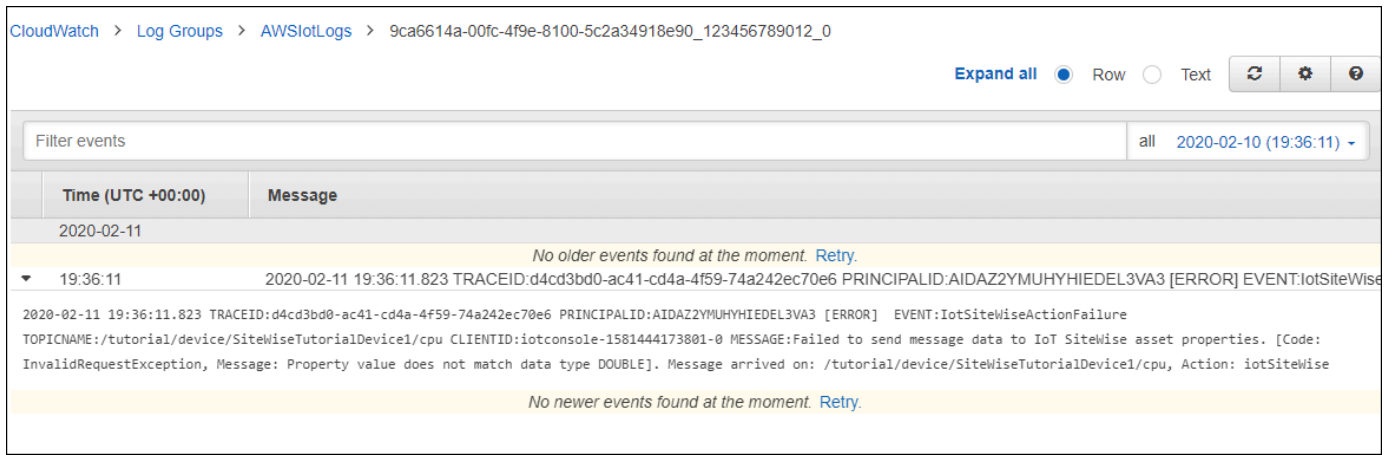
- [設定 AWS IoT Core 日誌](#)
- [設定重新發佈錯誤動作](#)
- [對規則問題進行故障診斷](#)
- [對規則進行故障診斷 \(AWS IoT SiteWise\)](#)
- [對規則進行故障診斷 \(DynamoDB\)](#)

設定 AWS IoT Core 日誌

您可以設定 AWS IoT 將各種層級的資訊記錄到 CloudWatch Logs。

設定和存取 CloudWatch Logs

1. 若要設定的記錄 AWS IoT Core，請參閱《AWS IoT 開發人員指南》中的[使用 CloudWatch Logs 進行監控](#)。
2. 導覽至 [CloudWatch 主控台](#)。
3. 在導覽窗格中，選擇 Log groups (日誌群組)。
4. 選擇 AWSIoTLogs 群組。
5. 選擇最近的日誌串流。根據預設，CloudWatch 會先顯示最新的日誌串流。
6. 選擇日誌項目以展開日誌訊息。日誌項目看起來可能與下列螢幕擷取畫面類似。



7. 比較錯誤訊息與本主題所述的錯誤，針對您的問題進行故障診斷。

設定重新發佈錯誤動作

您可以設定規則上的錯誤動作以處理錯誤訊息。在此程序中，您會設定重新發佈規則動作做為錯誤動作，以檢視 MQTT 測試用戶端中的錯誤訊息。

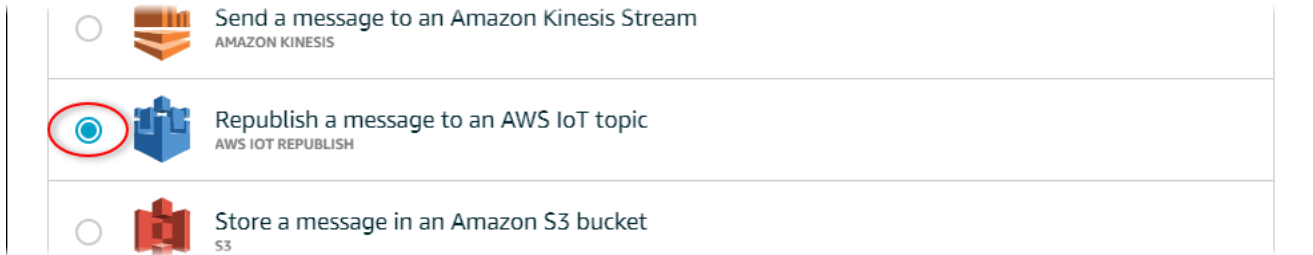
Note

重新發佈錯誤動作只會輸出相當於 ERROR 層級日誌的內容。如果您想要更多詳細日誌，您必須[設定 CloudWatch Logs](#)。

將重新發佈錯誤動作新增到規則

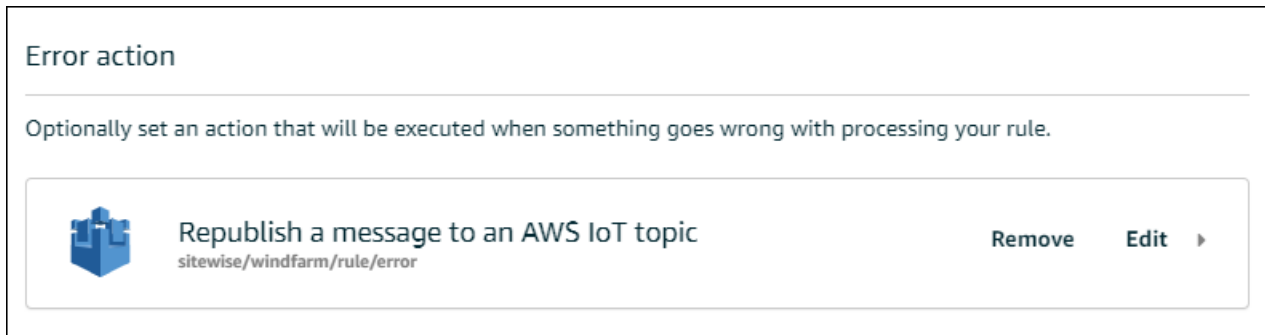
1. 導覽至 [AWS IoT 主控台](#)。
2. 在左側導覽窗格中，選擇 Act (動作)，然後選擇 Rules (規則)。

3. 選擇您的規則。
4. 在 Error action (錯誤動作) 下方，選擇 Add action (新增動作)。
5. 選擇重新發佈訊息至 AWS IoT 主題。



6. 選擇頁面底部的 Configure action (設定動作)。
7. 在主題中，輸入唯一主題（例如 **sitewise/windfarm/rule/error**）。AWS IoT Core 將重新發佈錯誤訊息至此主題。
8. 選擇選取以授予執行錯誤動作的 AWS IoT Core 存取權。
9. 選擇您已針對規則建立之角色旁的 Select (選取)。
10. 選擇 Update Role (更新角色) 以將其他許可新增到角色。
11. 選擇新增動作。

您規則的錯誤動作看起來應與下列螢幕擷取畫面類似。



12. 選擇主控台左上角的向後箭頭，以返回 AWS IoT 主控台首頁。

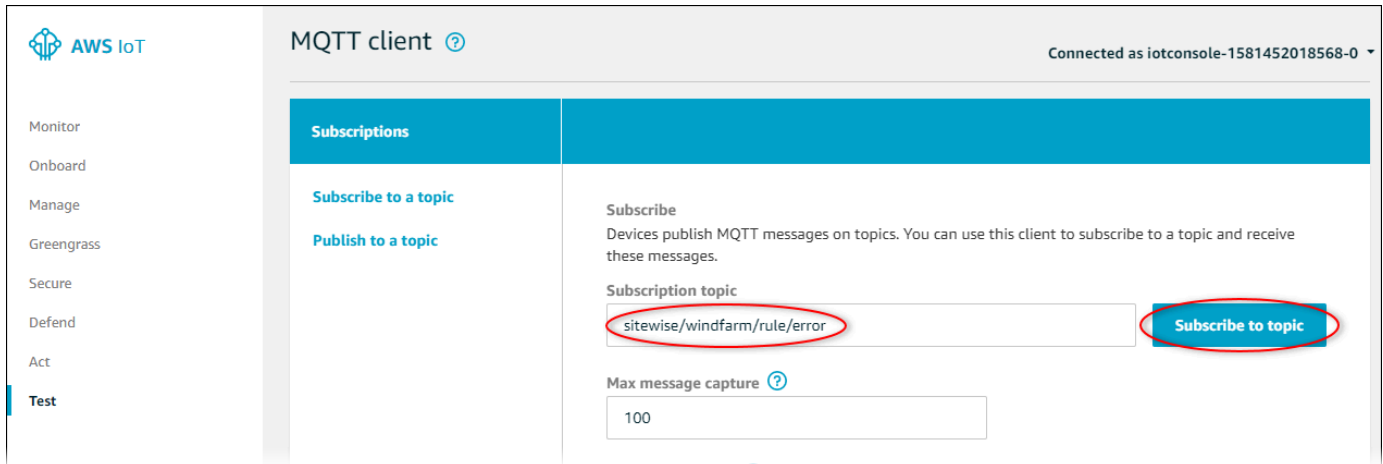
在您設定重新發佈錯誤動作後，您可以在 AWS IoT Core 中檢視 MQTT 測試用戶端內的錯誤訊息。

在以下程序中，您會在 MQTT 測試用戶端中訂閱錯誤主題。在 MQTT 測試用戶端中，您可以接收規則的錯誤訊息，以針對問題進行故障診斷。

訂閱錯誤動作主題

1. 導覽至 [AWS IoT 主控台](#)。

2. 在左側導覽頁面中，選擇 Test (測試) 以開啟 MQTT 測試用戶端。
3. 在 Subscription topic (訂閱主題) 欄位中，輸入您之前設定的錯誤主題 (例如 **sitewise/windfarm/rule/error**)，然後選擇 Subscribe to topic (訂閱主題)。



4. 注意出現的錯誤訊息，然後展開錯誤訊息中的 failures 陣列。

接下來，請比較錯誤訊息與本主題所述的錯誤，針對您的問題進行故障診斷。

對規則問題進行故障診斷

使用下列資訊來疑難排解規則問題。

問題

- [錯誤：成員必須在目前時間戳記之前 604800 秒內，以及目前時間戳記之後 300 秒內](#)
- [錯誤：屬性值與 <type> 資料類型不相符](#)
- [錯誤：使用者 <role-arn> 未獲得在資源上執行 iotsitewise:BatchPutAssetPropertyValue 的授權](#)
- [錯誤：iot.amazonaws.com 無法在資源: <role-arn> 上執行 sts:AssumeRole](#)
- [資訊：未傳送任何要求。在執行替代範本後，PutAssetPropertyValueEntries 是空的。](#)

錯誤：成員必須在目前時間戳記之前 604800 秒內，以及目前時間戳記之後 300 秒內

相較於目前的 Unix epoch 時間，您的時間戳記超過 7 天或超過 5 分鐘。請嘗試以下做法：

- 檢查您的時間戳記是否採用 Unix Epoch (UTC) 時間格式。如果您提供不同時區的時間戳記，則會發生此錯誤。

- 檢查您的時間戳記以秒為單位。AWS IoT SiteWise 預期時間戳記會以秒為單位分割為時間（以 Unix epoch 時間為單位），並以奈秒為單位偏移。
- 檢查您上傳的資料時間戳記是否不超過過去 7 天。

錯誤：屬性值與 <type> 資料類型不相符

規則動作中的項目具有與目標資產屬性不同的資料類型。例如，您的目標資產屬性為 DOUBLE，而您選取的資料類型為 Integer (整數)，或您已傳遞 integerValue 的值。請嘗試以下做法：

- 如果您從 AWS IoT 主控台設定規則，請檢查您是否為每個項目選擇正確的資料類型。
- 如果您從 API 或 AWS Command Line Interface (AWS CLI) 設定規則，請檢查value物件是否使用正確的類型欄位（例如DOUBLE，doubleValue 屬性）。

錯誤：使用者 <role-arn> 未獲得在資源上執行 iotsitewise:BatchPutAssetPropertyValue 的授權

您的規則未獲得存取目標資產屬性的授權，或目標資產屬性不存在。請嘗試以下做法：

- 檢查您的屬性別名是否正確，並確認您擁有具有指定屬性別名的資產屬性。如需詳細資訊，請參閱[管理的資料串流 AWS IoT SiteWise](#)。
- 檢查您的規則是否具有角色，且該角色允許目標資產屬性的 iotsitewise:BatchPutAssetPropertyValue 許可，例如透過目標資產的階層。如需詳細資訊，請參閱[授予 AWS IoT 必要的存取權](#)。

錯誤：iot.amazonaws.com 無法在資源: <role-arn> 上執行 sts:AssumeRole

您的使用者無權在 AWS Identity and Access Management (IAM) 中擔任規則上的角色。

檢查您的使用者是否獲得規則上角色的iam:PassRole許可。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[傳遞角色許可](#)。

資訊：未傳送任何要求。在執行替代範本後，PutAssetPropertyValueEntries 是空的。

Note

此訊息是 INFO 層級日誌。

您的請求必須至少具有一個含有所有必要參數的項目。

檢查您規則的參數 (包括替代範本) 是否導致非空白的值。替代範本無法存取規則查詢陳述式的 AS 子句中定義的值。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[替代範本](#)。

對規則進行故障診斷 (AWS IoT SiteWise)

如果 CPU 和記憶體使用量資料未 AWS IoT SiteWise 如預期出現在 中，請依照此程序中的步驟來疑難排解您的規則。在此程序中，您會設定重新發佈規則動作做為錯誤動作，以檢視 MQTT 測試用戶端中的錯誤訊息。您也可以設定 CloudWatch Logs 的記錄以進行疑難排解。如需詳細資訊，請參閱[對 AWS IoT SiteWise 規則動作進行故障診斷](#)。

將重新發佈錯誤動作新增到規則

1. 導覽至 [AWS IoT 主控台](#)。
2. 在左側導覽窗格中，選擇訊息路由，然後選擇規則。
3. 選擇您先前建立的規則，然後選擇編輯。
4. 在錯誤動作 - 選用下，選擇新增錯誤動作。
5. 選擇重新發佈訊息至 AWS IoT 主題。
6. 在主題中，輸入錯誤路徑 (例如 `sitewise/rule/tutorial/error`)。AWS IoT Core 將重新發佈錯誤訊息至此主題。
7. 選擇您先前建立的角色 (例如 `SiteWiseTutorialDeviceRuleRole`)。
8. 選擇更新。

在您設定重新發佈錯誤動作後，您可以在 AWS IoT Core 中檢視 MQTT 測試用戶端內的錯誤訊息。

在以下程序中，您會在 MQTT 測試用戶端中訂閱錯誤主題。

訂閱錯誤動作主題

1. 導覽至 [AWS IoT 主控台](#)。
2. 在左側導覽頁面中，選擇 MQTT 測試用戶端以開啟 MQTT 測試用戶端。
3. 在主題篩選條件欄位中，輸入 `sitewise/rule/tutorial/error` 並選擇訂閱。

出現錯誤訊息時，請檢視任何錯誤訊息中的 `failures` 陣列以診斷問題。如需可能問題和解決方式的詳細資訊，請參閱[對 AWS IoT SiteWise 規則動作進行故障診斷](#)。

如果沒有出現錯誤，請檢查您的規則已啟用，以及您訂閱的主題和您在重新發佈錯誤動作中設定的主題相同。如果錯誤在您這麼做之後仍未出現，請檢查裝置指令碼是否正在執行，並成功更新裝置的陰影。

Note

您也可以訂閱裝置的影子更新主題，以檢視 AWS IoT SiteWise 動作剖析的承載。若要這麼做，請訂閱下列主題。

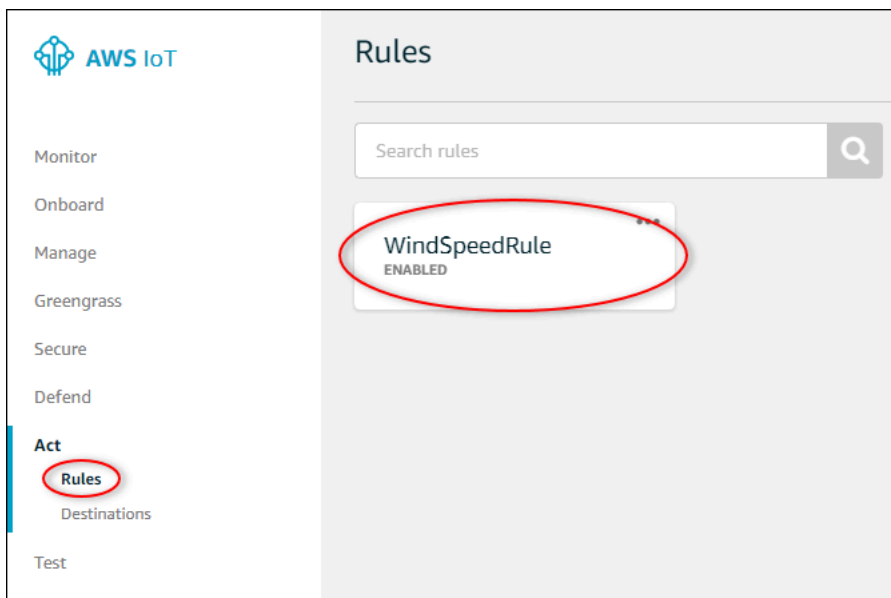
```
$aws/things/+/shadow/update/accepted
```

對規則進行故障診斷 (DynamoDB)

如果示範資產資料未如預期出現在 DynamoDB 資料表中，請依照此程序中的步驟對規則進行疑難排解。在此程序中，您會設定重新發佈規則動作做為錯誤動作，以檢視 MQTT 測試用戶端中的錯誤訊息。您也可以設定 CloudWatch Logs 的記錄以進行疑難排解。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的[使用 CloudWatch Logs 進行監控](#)。

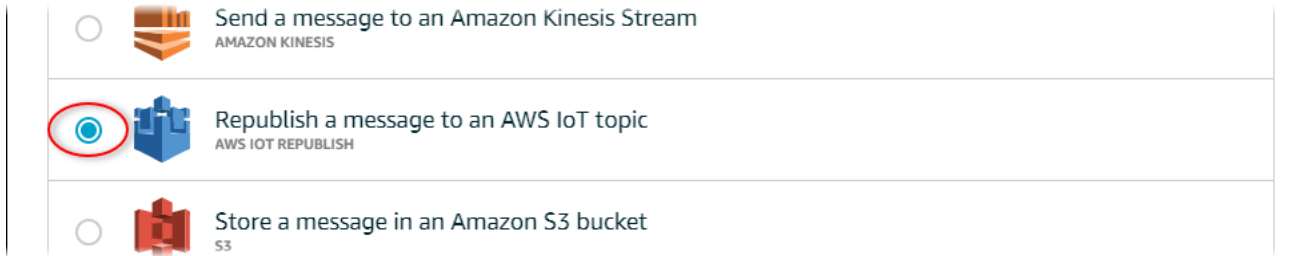
將重新發佈錯誤動作新增到規則

1. 導覽至 [AWS IoT 主控台](#)。
2. 在左側導覽窗格中，選擇 Act (動作)，然後選擇 Rules (規則)。
3. 選擇您稍早建立的規則。



4. 在 Error action (錯誤動作) 下方，選擇 Add action (新增動作)。

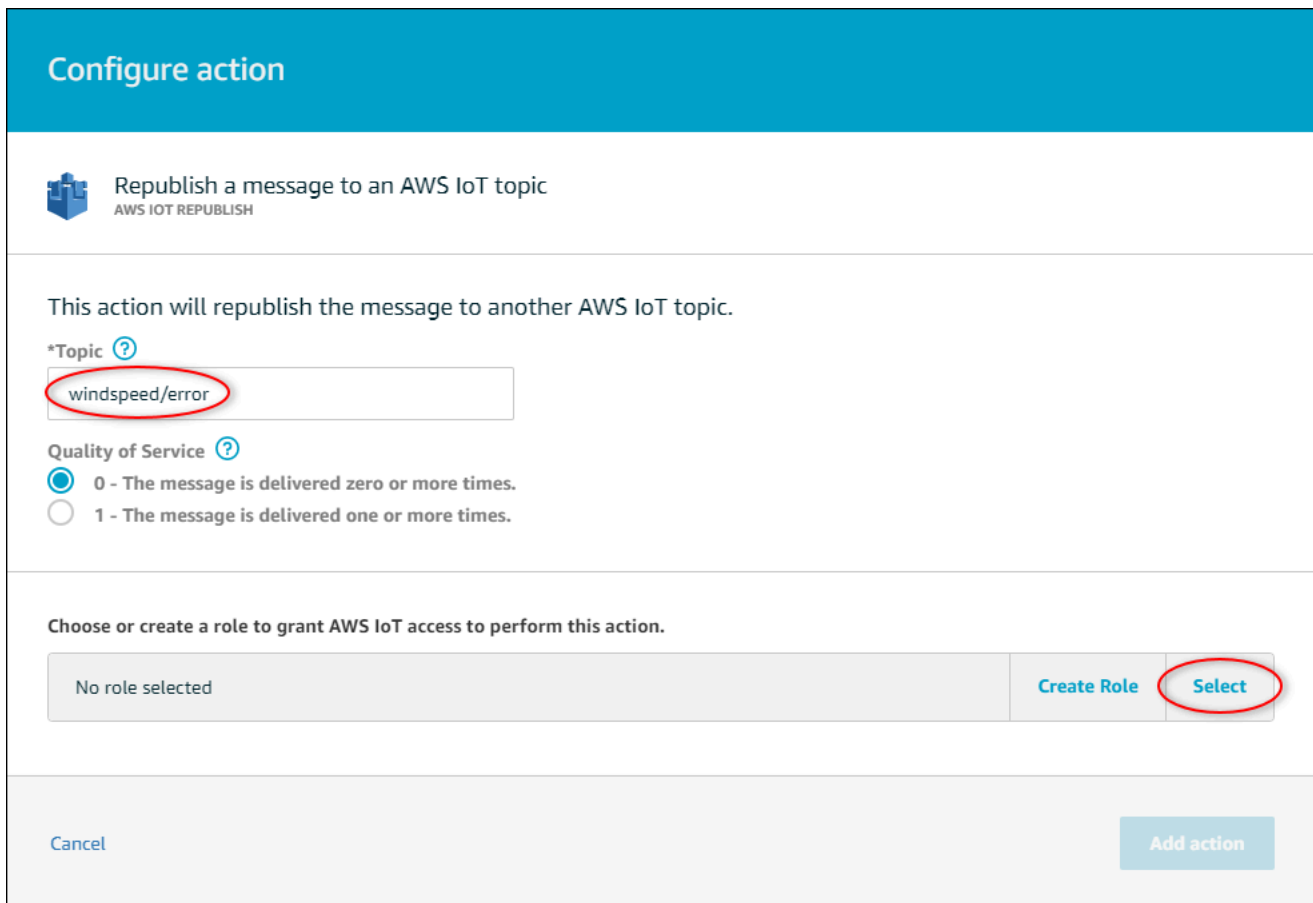
5. 選擇重新發佈訊息至 AWS IoT 主題。



The screenshot shows a configuration menu with three options, each with a radio button and an icon. The second option, 'Republish a message to an AWS IoT topic' with the AWS IoT icon, is selected and circled in red. The first option is 'Send a message to an Amazon Kinesis Stream' with the Amazon Kinesis icon. The third option is 'Store a message in an Amazon S3 bucket' with the Amazon S3 icon.

6. 選擇頁面底部的 Configure action (設定動作)。

7. 在 Topic (主題) 中，輸入 **windspeed/error**。AWS IoT Core 會將錯誤訊息重新發佈至此主題。



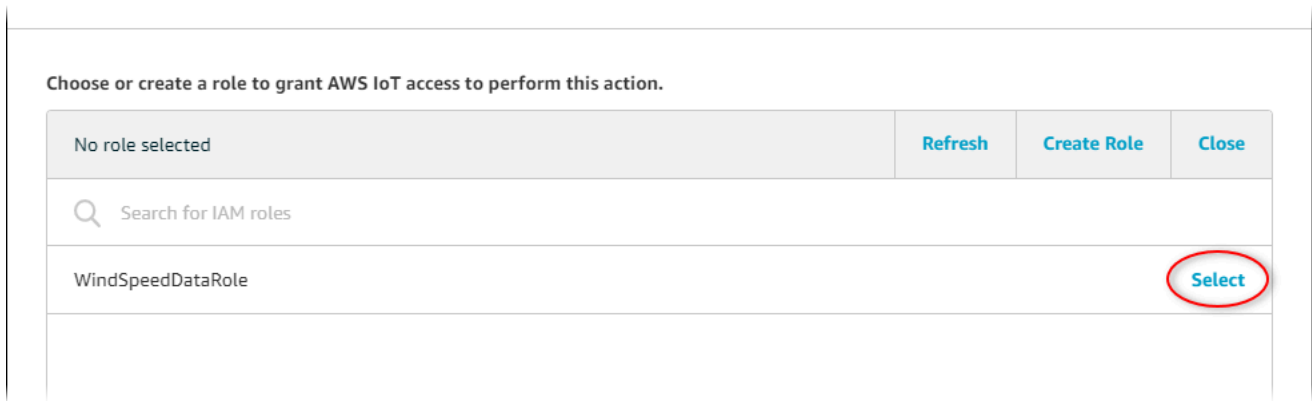
The screenshot shows the 'Configure action' dialog box for the 'Republish a message to an AWS IoT topic' action. The dialog has a blue header with the title 'Configure action'. Below the header, the action name and icon are displayed. The main content area contains the following fields and options:

- A text field labeled '*Topic' with a question mark icon, containing the text 'windspeed/error'. This field is circled in red.
- A section labeled 'Quality of Service' with a question mark icon, containing two radio button options:
 - ☒ 0 - The message is delivered zero or more times.
 - ☐ 1 - The message is delivered one or more times.
- A section labeled 'Choose or create a role to grant AWS IoT access to perform this action.' containing a dropdown menu with the text 'No role selected', a 'Create Role' button, and a 'Select' button. The 'Select' button is circled in red.

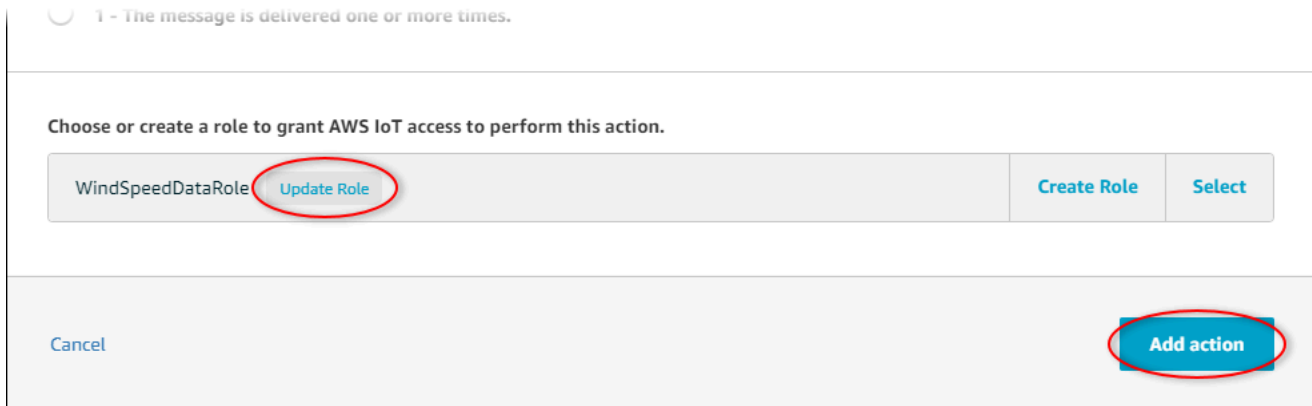
At the bottom of the dialog, there are two buttons: 'Cancel' on the left and 'Add action' on the right.

8. 選擇選取以授予 AWS IoT 核心存取權，以使用您先前建立的角色執行錯誤動作。

9. 選擇您角色旁邊的 Select (選取)。



10. 選擇 Update Role (更新角色) 以將其他許可新增到角色。



11. 選擇 Add action (新增動作) 來完成新增錯誤動作。

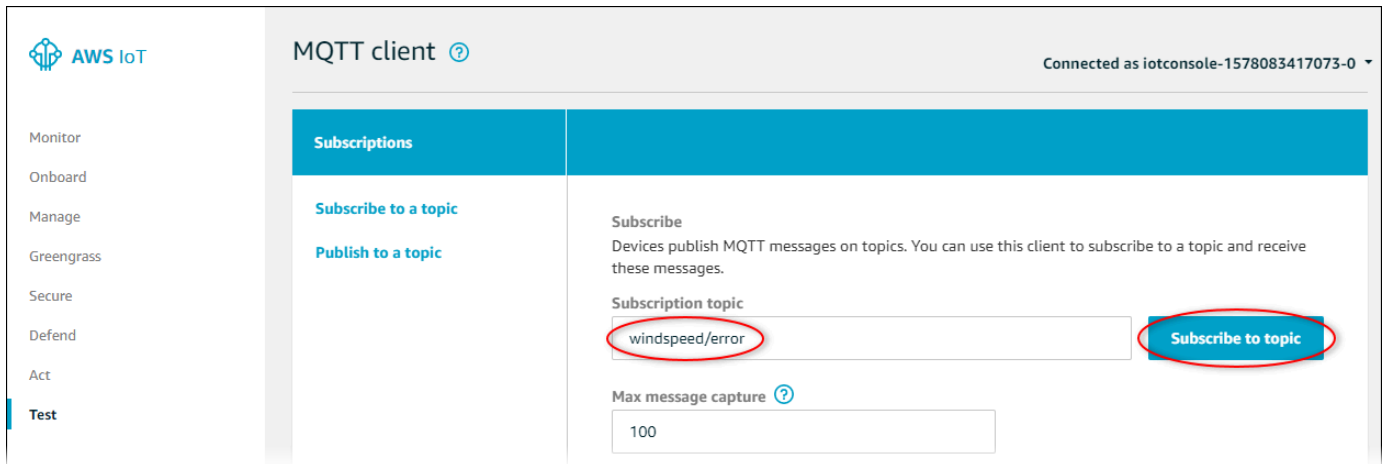
12. 選擇主控台左上角的向後箭頭，以返回 AWS IoT 核心主控台首頁。

設定重新發佈錯誤動作後，您可以在 AWS IoT Core 的 MQTT 測試用戶端中檢視錯誤訊息。

在以下程序中，您會在 MQTT 測試用戶端中訂閱錯誤主題。

訂閱錯誤動作主題

1. 在 AWS IoT 核心主控台的左側導覽頁面中，選擇測試。
2. 在 Subscription topic (訂閱主題) 欄位中，輸入 **windspeed/error**，然後選擇 Subscribe to topic (訂閱主題)。



3. 注意出現的錯誤訊息，並探索錯誤訊息中的 failures 陣列，以診斷下列常見問題：

- 規則查詢陳述式中的拼字錯誤
- 角色許可不足

如果沒有出現錯誤，請檢查您的規則已啟用，以及您訂閱的主題和您在重新發佈錯誤動作中設定的主題相同。如果仍然沒有出現錯誤，請檢查您的示範風力發電廠資產仍然存在，並且您已在風速屬性上啟用通知。如果您的示範資產已過期且從中消失 AWS IoT SiteWise，您可以建立新的示範，並更新規則查詢陳述式以反映更新的資產模型和屬性 IDs。

對大量匯入和匯出操作進行故障診斷

若要處理和診斷傳輸任務期間產生的錯誤，請參閱 AWS IoT TwinMaker GetMetadataTransferJob API：

1. 建立並執行傳輸任務之後，請呼叫 GetMetadataTransferJob API：

```
aws iottwinmaker get-metadata-transfer-job \  
--metadata-transfer-job-id your_metadata_transfer_job_id \  
--region us-east-1
```

2. 任務的狀態會變更為下列其中一個狀態：

- COMPLETED (已完成)
- CANCELLED (已取消)
- ERROR

3. GetMetadataTransferJob API 會傳回 [MetadataTransferJobProgress](#) 物件。
4. MetadataTransferJobProgress 物件包含下列參數：
 - failedCount : 表示轉移程序期間失敗的資產計數。
 - skippedCount : 表示轉移過程中略過的資產計數。
 - succeededCount : 表示轉移程序期間成功的資產計數。
 - totalCount : 表示轉移程序中涉及的資產總數。
5. 此外，API 呼叫會傳回 reportUrl 元素，其中包含預先簽章的 URL。如果您的傳輸任務發生錯誤，需要調查，您可以在此 URL 下載完整的錯誤報告。

AWS IoT SiteWise 端點和配額

下列各節說明 的端點和配額 AWS IoT SiteWise。

主題

- [AWS IoT SiteWise 端點](#)
- [AWS IoT SiteWise 配額](#)

AWS IoT SiteWise 端點

AWS 一般參考 指南列出的 AWS IoT SiteWise 端點 AWS 帳戶。如需詳細資訊，請參閱《AWS 一般參考 指南》中的[AWS IoT SiteWise 端點和配額](#)。

AWS IoT SiteWise 配額

下表說明 中的配額 AWS IoT SiteWise。如需配額以及如何請求提高配額的詳細資訊，請參閱 中的[AWS 服務配額](#)AWS 一般參考。如需 AWS IoT SiteWise 配額的詳細資訊，請參閱 中的[AWS IoT SiteWise 服務配額](#)AWS 一般參考。

AWS IoT SiteWise 資產和資產模型的配額

資源	描述	配額	可調整
每個 AWS 區域 中的資產模型數量 AWS 帳戶	您可以在 中 AWS 區域 為 建立的資產模型數目上限 AWS 帳戶。	1000	是
每個資產模型中的資產數量	您可以為每個資產模型建立的資產數量上限。	10,000	是
每個父資產中的子資產數量	您可以與父資產建立關聯的子資產數量上限。	2000	是
資產模型階層樹狀結構的深度	資產模型的資產階層樹狀結構深度上限。	30	是

資源	描述	配額	可調整
每個資產模型中的階層定義數目	您可以在資產模型中擁有的階層定義數目上限。	30	是
每個資產模型中根層級中的屬性數量	每個資產模型 <code>assetModelProperties</code> 的數目上限。此計數不包含 <code>compositeModelProperties</code> 。此配額也適用於從此資產模型建立的任何唯一資產。	500	是
資產模型中的屬性數目	類型 <code>ASSET_MODEL</code> 或 <code>COMPONENT_MODEL</code> 之資產模型的屬性數目上限。此數字是透過結合根資產模型的屬性和任何包含 <code>component-model-based</code> 或內嵌複合模型來決定。此配額也適用於從此資產模型建立的任何唯一資產。	5000	是
每個複合模型中的屬性數量	複合模型允許的屬性數目上限。此外，類型為 <code>COMPONENT_MODEL</code> 的資產模型允許的屬性數目上限。	100	是

資源	描述	配額	可調整
資產模型中屬性樹狀結構的深度	例如，具有轉換屬性 C 的模型深度為 3，其轉換屬性 C 會消耗轉換屬性 B，而轉換屬性 B 會消耗衡量值屬性 A。	10	否
每個階層樹狀結構中的資產模型數量	您可以在單一階層樹狀結構中包含的資產模型數量上限。	100	是
資產模型的直接相依屬性數目	此配額會如屬性公式表達式中定義的方式，限制可以直接依存於單一屬性的屬性數目。相依屬性的數量必須大於資產模型的直接相依屬性數量。如果資產模型有比相依屬性更直接的相依屬性，則請求增加這兩個配額。	20	是
資產模型中的相依屬性數目	此配額會如屬性公式表達式中定義的方式，限制可以直接或間接依存於單一屬性的屬性數目。	30	否
資產模型中的複合模型數量	您可以在單一資產模型上擁有的複合模型數量上限。	50	是

資源	描述	配額	可調整
複合模型深度	每個資產模型中複合模型樹的最大深度，包括內嵌和component-model-based複合模型。	2	是
使用相同元件模型的唯一資產模型數量	至少有一個component-model-based複合模型，直接參考類型 COMPONENT_MODEL 的特定資產模型的唯一資產模型數目上限。	20	是
屬性公式表達式中的屬性變數數目	例如，運算式 temp 中有兩個屬性變數 power 和 avg(power) + max(temp)。這也適用於轉換運算結果。	10	否
屬性公式表達式中的函數數量	例如，在表達式 max 中有兩個函數 avg 和 avg(power) + max(temp)。	10	否

AWS IoT SiteWise 資產屬性資料的配額

資源	描述	配額	可調整
資產屬性資料 API 操作的請求率	您可以在 AWS 區域 每個 中每秒執行的資產屬性資料 API 請求數目上限 AWS 帳戶。此配額	1000	是

資源	描述	配額	可調整
	適用於 GetAssetProperty 和 BatchPutAssetProperty 之類的 API 操作。		
每個資產屬性的每個資料品質每秒的資料點數	此配額適用於每個資產屬性每個資料品質具有相同時間戳記的 timestamp-quality-value (TQV) 資料點數目上限，以秒為單位。您可以為每個資產屬性儲存最多此數量的良好品質、不確定品質和不良品質資料點。	10	否
在 的每個資產屬性中每秒擷取 BatchPutAssetProperty 的項目數 AWS 區域 AWS 帳戶。	BatchPutAssetProperty 來自所有來源的每個資產屬性中的項目數量上限，包括 SiteWise Edge 閘道、AWS IoT Core 規則和 API 呼叫。	10	否
資料點擷取速率	每個 每秒擷取的 timestamp-quality-value (TQV) AWS 區域 資料點數目上限 AWS 帳戶。	5000	是

資源	描述	配額	可調整
的請求率 BatchGetAssetPropertyAggregates	您可以在每個 中每秒執行的最大BatchGetAssetPropertyAggregates 請求數 AWS 區域 AWS 帳戶。	200	是
的請求率 BatchGetAssetPropertyValue	您可以在每個 中每秒執行的最大BatchGetAssetPropertyValue 請求數 AWS 區域 AWS 帳戶。	500	是
的請求率 BatchGetAssetPropertyValueHistory	您可以每秒執行的BatchGetAssetPropertyValueHistory 請求數目上限。	200	是
AWS 區域 中每個資產屬性每秒擷取BatchPutAssetPropertyValue 的項目數 AWS 帳戶。	此配額適用於BatchPutAssetPropertyValue 來自所有來源之 的每個資產屬性中的項目，包括 SiteWise Edge 閘道、AWS IoT Core 規則和 API 呼叫。	10	否

資源	描述	配額	可調整
每個資產屬性的GetAssetPropertyAggregates 請求和BatchGetAssetPropertyAggregates 項目查詢速率	每個資產屬性中每秒每個資產屬性的GetAssetPropertyAggregates 請求和BatchGetAssetPropertyAggregates 項目總數上限 AWS 區域 AWS 帳戶。	50	否
每個資產屬性的GetAssetPropertyValue 請求和BatchGetAssetPropertyValue 項目查詢速率	每個資產屬性中每秒的GetAssetPropertyValue 請求和BatchGetAssetPropertyValue 項目總數上限 AWS 區域 AWS 帳戶。	500	否
每個資產屬性的GetAssetPropertyValueHistory 請求和BatchGetAssetPropertyValueHistory 項目查詢速率	每個資產屬性中每秒的GetAssetPropertyValueHistory 請求和BatchGetAssetPropertyValueHistory 項目總數上限 AWS 區域 AWS 帳戶。	30	否

資源	描述	配額	可調整
GetInterp olatedAss etPropert yValues 請求率	您可以在每個 中 每秒執行的最 大GetInterp olatedAss etPropert yValues 請求數 AWS 區域 AWS 帳 戶。	500	是
每個GetInterp olatedAss etPropert yValues 請求中結果 的數量	分頁GetInterp olatedAss etPropert yValues 請求傳回的 結果數目上限。	10	是

資源	描述	配額	可調整
從 GetAssetPropertyValueHistory 和 BatchGetAssetPropertyHistory 擷取的資料點速率	AWS 帳戶 跨 GetAssetPropertyValueHistory 和 BatchGetAssetPropertyHistory 在 AWS 區域 中每個 每秒擷取的資料點最大位元組速率 (MB/秒) BatchGetAssetPropertyHistory 。此配額評估的回應承載會針對每個資料點使用 Timestamp-Quality-Value (TQV) 欄位，並將每個 API 請求的位元組大小四捨五入至下一個 4KB 增量。 每秒擷取的Timestamp-quality-value(TQV) 資料點會因每種資料類型而異： • 整數 – 每秒高達 500 萬 TQV • 雙重 – 每秒高達 400 萬 TQV • 布林值 – 每秒最多 600 萬 TQV • 字串 – 因每個字串值大小而異。	100	是

SiteWise Edge 閘道的配額

資源	描述	配額	可調整
的每個 中的 SiteWise Edge AWS 區域 閘道數量 AWS 帳戶	您可以在 中 AWS 區域 為 建立的 SiteWise Edge 閘道數量上限 AWS 帳戶。	100	是
SiteWise Edge 閘道中的 OPC UA 來源數量	您可以在 SiteWise Edge 閘道中設定的 OPC UA 來源數量上限。	100	否
SiteWise Edge 閘道中的目的地總數	您可以在 SiteWise Edge 閘道中設定的目的地數量上限。	100	否

的配額 AWS IoT SiteWise Monitor

資源	描述	配額	可調整
的每個 中的入口網站數量 AWS 區域 AWS 帳戶	您可以在 中 AWS 區域 為 建立的 SiteWise Monitor 入口網站數目上限 AWS 帳戶。	100	是
入口網站中的專案數量	您可以在 SiteWise Monitor 入口網站中建立的專案數量上限。	100	是
專案中的儀表板數量	您可以在 SiteWise Monitor 中的專案中建立的儀表板數量上限。	100	是
專案中的根資產數目	您可以在 SiteWise Monitor 中新增至專案	1	否

資源	描述	配額	可調整
	的最上層資產數目上限。		
儀表板中的視覺化數量	您可以在 SiteWise Monitor 中新增至儀表板的視覺化元素（例如圖表、圖形或資料表）數目上限。	10	是
每個儀表板視覺化中的指標數量	您可以在 SiteWise Monitor 儀表板上的單一視覺化中顯示的指標或資料點數量上限。	5	是
每個儀表板視覺化的閾值數量	您可以在 SiteWise Monitor 中為儀表板上的每個視覺效果設定的閾值層級數量上限。	12	否

AWS IoT SiteWise 大量匯入和匯出中繼資料的配額

資源	描述	配額	可調整
佇列中的中繼資料傳輸任務數量	佇列中PENDING中繼資料傳輸任務的數量上限。	10	是
中繼資料傳輸任務匯入檔案的大小	匯入檔案的大小上限（以 MB 為單位）。	100	是
任務中的 AWS IoT SiteWise 匯入資源數量	單一任務中的 AWS IoT SiteWise 匯入資源數目上限。資源包含資產和資產模型。	5000	是

資源	描述	配額	可調整
任務中的 AWS IoT SiteWise 匯出資源數量	單一任務中的 AWS IoT SiteWise 匯出資源數目上限。資源包含資產和資產模型。	5000	是

AWS IoT SiteWise 大量匯入資料的配額

資源	描述	配額	可調整
執行中的大量匯入任務數量	可同時執行的大量匯入任務數量上限。	100	否
CSV 檔案的大小	大量匯入任務中的 CSV 檔案大小上限 (以 GB 為單位)。	10	否
未壓縮 parquet 檔案的大小	大量匯入任務中未壓縮 parquet 檔案的檔案大小上限 (MB)。	256 MB	否
緩衝擷取的 CSV 檔案大小	在大量匯入任務上使用緩衝擷取時的最大 CSV 檔案大小 (以 MB 為單位)。	256 MB	否
未壓縮的 parquet 資料列群組大小	未壓縮 parquet 資料列群組的大小上限。	64 MB	否
parquet 檔案中唯一度量的數量	parquet 檔案中唯一測量的最大數量。	10000	否
緩衝擷取過去和今天時間戳記之間的天數	使用緩衝擷取時，過去時間戳記與今天日期之間的天數上限。	30	是
每個 CreateBulkImportJobs 中		10	是

資源	描述	配額	可調整
每個 AWS 區域 的請求率 AWS 帳戶			
每個 AWS 區域 中 每個 ListBulkImportJobs 的請求率 AWS 帳戶		50	是
每個 AWS 區域 中 每個 DescribeBulkImportJobs 的請求率 AWS 帳戶		50	是

AWS IoT SiteWise 助理 API 限流配額

AWS IoT SiteWise 助理 API 限流限制的配額

資源	描述	配額	可調整
InvokeAssistant 操作的請求率	每分鐘可對 中的 AWS IoT SiteWise InvokeAssistant API 進行的最大交易數 (TPM) AWS 帳戶。TPM 限制適用於所有支援的區域，而且在某些區域中是可調整的。	10	否

異常偵測的配額

異常偵測的配額會在 AWS IoT SiteWise 和 Amazon Lookout for Equipment 之間共用。如需詳細資訊，請參閱[使用 Lookout for Equipment 的配額](#)。

AWS IoT SiteWise 使用者指南的文件歷史記錄

下表說明此版本 的文件 AWS IoT SiteWise。

- API 版本：2019-12-02

變更	描述	日期
使用使用者名稱和密碼將外部應用程式連接至 EMQX 代理程式	新增支援使用使用者名稱和密碼將外部應用程式連線至預設部署的 AWS IoT Greengrass EMQX 代理程式，以建立新的啟用 MQTT 的 V3 閘道。	2025 年 5 月 1 日
在 SiteWise Edge 上支援啟用 MQTT 的 V3 閘道	新增新功能並移除已棄用的內容 • 新增對啟用 MQTT 的 V3 閘道的支援。使用路徑篩選條件來訂閱 MQTT 主題的增強型目的地組態，包括直接擷取即時資料或使用 Amazon S3 AWS IoT SiteWise 緩衝資料擷取。 • 已發行 IoT SiteWise OPC UA 收集器的 3.0.0 版，以及 IoT SiteWise 發佈者元件的 4.0.0 版 AWS IoT Greengrass V2。 • 將舊版自我託管 SiteWise Edge 閘道重新命名為 Classic 串流 V2 閘道。 • 已移除 SiteWise Edge 文件 AWS IoT Greengrass V1 中的 參考，因為它不再	2025 年 2 月 26 日

支援與 搭配使用 AWS IoT SiteWise。

支援AWS IoT SiteWise Assistant	新增對 AWS IoT SiteWise Assistant 的支援 - 生成式 AI 輔助。	2024 年 11 月 18 日
新增 SiteWise Edge APIs 的可設定工作階段逾時	新增可設定的工作階段逾時設定，以管理和 SiteWise Edge APIs 的 AWS OpsHub 閒置期間。	2024 年 10 月 31 日
新增 SiteWise Edge APIs 的可設定代理設定	新增管理信任存放區資訊，以啟用 SiteWise Edge 閘道的 HTTPS 代理支援。	2024 年 10 月 31 日
在上為 SiteWise Edge APIs 啟用 CORS	新增 SiteWise Edge APIs 的 CORS 支援，以啟用安全的跨網域 Web 應用程式存取。	2024 年 9 月 30 日
支援 CloudRail 和 Litmus Edge 合作夥伴資料來源	新增對 CloudRail 和 Litmus Edge 做為合作夥伴資料來源的支援。	2024 年 9 月 5 日
在 Siemens Industrial Edge 上執行 SiteWise Edge 的一般可用性	AWS IoT SiteWise 現在支援在 Siemens Industrial Edge 裝置上執行 SiteWise Edge 的一般可用性。	2024 年 7 月 24 日
新增對 OPC UA 資料來源時間戳記組態的支援	AWS IoT SiteWise 現在支援 OPC UA 資料來源的時間戳記組態。	2024 年 7 月 24 日
新增對 OPC UA 資料來源之資料類型轉換的支援	AWS IoT SiteWise 現在支援對不支援的 OPC UA 資料類型進行資料類型轉換。	2024 年 7 月 24 日

新增在 Siemens Industrial Edge 上執行 SiteWise Edge 預覽的支援	AWS IoT SiteWise 現在支援在 Siemens Industrial Edge 裝置上執行 SiteWise Edge 預覽。	2023 年 11 月 26 日
新增對暖層儲存的支援	AWS IoT SiteWise 現在支援暖儲存，這是全受管儲存層，可讓客戶輕鬆安全地存放和存取工業資料。	2023 年 11 月 15 日
新增對使用者定義的唯一識別符的支援	AWS IoT SiteWise 現在支援將使用者定義的唯一識別符用於資產、資產模型、屬性和階層。	2023 年 11 月 15 日
新增對工業資產多變體異常偵測的支援	AWS IoT SiteWise 現在透過將歷史和即時設備資料與 Amazon Lookout for Equipment 整合，支援工業資產的多變體異常偵測。	2023 年 11 月 15 日
新增對 中時間序列資料的成本效益和可擴展擷取的支援 AWS IoT SiteWise	AWS IoT SiteWise 現在支援分析使用案例所需的經濟實惠且可擴展的時間序列資料擷取。	2023 年 11 月 15 日
新增對大量匯入、匯出和更新的支援	AWS IoT SiteWise 現在支援大量匯入、匯出和更新工業設備中繼資料。	2023 年 11 月 15 日
新增對資產模型元件的支援	AWS IoT SiteWise 現在支援資產模型元件，協助工業客戶建立可重複使用的元件。	2023 年 11 月 15 日
新增對 IoT 儀表板應用程式的支援	AWS IoT SiteWise 現在支援開放原始碼儀表板應用程式，您可以在其中視覺化操作資料並與之互動。	2023 年 11 月 15 日

更新的服務連結角色 AWS IoT SiteWise	AWS IoT SiteWise 有新的服務連結角色，可以針對 AWS IoT TwinMaker 資料庫執行中繼資料搜尋查詢。	2023 年 11 月 6 日
更新 AWS IoT SiteWise 資料串流資源的標記	新增了標記資料串流資源的支援。	2022 年 8 月 18 日
更新的 SiteWise Edge 閘道	您現在可以設定發佈者，以控制從邊緣傳送至雲端的資料，以及傳送至雲端的順序。	2022 年 1 月 12 日
已更新 AWS IoT SiteWise 示範	您現在可以使用示範來建立 SiteWise Monitor 入口網站。	2022 年 1 月 10 日
更新儲存管理	您現在可以定義保留期間，以控制您的資料在熱層中保留多久。	2021 年 11 月 29 日
新增對資料串流管理的支援	您現在可以在建立資產模型和資產 AWS IoT SiteWise 之前，將資料擷取至。	2021 年 11 月 24 日
更新資產模型階層	子資產模型現在可以與多個父資產模型建立關聯。	2021 年 10 月 28 日
區域啟動	已啟動 AWS IoT SiteWise in AWS GovCloud (美國西部)。	2021 年 9 月 29 日
已更新 函數	新增下列功能 - 在指標中，您可以在彙總函數和暫時函數中使用巢狀運算式。 - 在轉換中，您可以使用 pretrigger() 函數，在觸發目前轉換計算的屬性更新之前擷取變數的值。	2021 年 8 月 10 日

自訂指標時間間隔	新增對指標中自訂時間間隔和偏移的支援。	2021 年 8 月 3 日
AWS IoT SiteWise 在邊緣使用	邊緣處理功能現已正式推出。	2021 年 7 月 29 日
將資料匯出至 Amazon S3	AWS IoT SiteWise 現在可將資料匯出至 Amazon S3。	2021 年 7 月 27 日
VPC 端點 (AWS PrivateLink)	控制平面 API 操作的介面 VPC 端點現已正式推出。	2021 年 7 月 15 日
轉換	轉換現在可以輸入多個資產屬性變數。	2021 年 7 月 8 日
更新 timestamp() 函數	在轉換中，您現在可以提供變數做為timestamp() 函數的引數。	2021 年 6 月 16 日
警示一般可用性	警示功能現已正式推出。	2021 年 5 月 27 日
已發行 Modbus-TCP 通訊協定轉接器第 2 版	Modbus-TCP 通訊協定轉接器連接器 第 2 版已推出。此版本新增了對 ASCII、UTF8 和 ISO8859 編碼來源字串的支援。	2021 年 5 月 24 日
已更新服務配額	新增了 GetInterpolatedAssetPropertyValues API 的下列配額：GetInterpolatedAssetPropertyValues 請求率、每個GetInterpolatedAssetPropertyValues 請求的結果數量，以及過去和今天開始日期之間的天數GetInterpolatedAssetPropertyValues 。	2021 年 4 月 29 日

[更新的公式表達式](#)

新增下列運算子和函數：

2021 年 4 月 22 日

- 新增下列[運算子](#)：<、>、<=、>=、==、!=、and、or、和 not。
- 新增下列[比較函數](#)：neq(x, y)。
- 新增下列[字串函數](#)：join()、format()和 f' '。

[VPC 端點 \(AWS PrivateLink\)](#)

新增有關如何透過建立介面 VPC 端點，在虛擬私有雲端 (VPC) 與 AWS IoT SiteWise 控制平面 APIs 之間建立私有連線的資訊。

2021 年 3 月 16 日

[IAM 聯合](#)

您的 SiteWise Monitor 入口網站管理員和使用者現在可以使用其 IAM 登入資料登入其指派的入口網站。

2021 年 3 月 16 日

[區域啟動](#)

AWS IoT SiteWise 在中國（北京）推出。

2021 年 2 月 3 日

[IoT SiteWise 連接器版本 10 已發行](#)

IoT SiteWise 連接器第 10 版已推出。此版本設定 StreamManager 以改善來源連線中斷並重新建立時的處理。當沒有可用的 ServerTimestamp 時，此版本也接受具有的 OPC UA SourceTimestamp 值。

2021 年 1 月 22 日

[日期和時間函數](#)

AWS IoT SiteWise 現在支援日期和時間函數。

2021 年 1 月 21 日

[函數語法](#)

您現在可以將統一函數呼叫語法 (UFCS) 用於 AWS IoT SiteWise 函數。

2021 年 1 月 11 日

[與 Grafana 整合](#)

新增如何在 Grafana 儀表板中視覺化 AWS IoT SiteWise 資料的相關資訊。

2020 年 12 月 15 日

[AWS IoT SiteWise 功能版本](#)

您現在可以使用警示監控資料、在邊緣處理工業資料、對 SiteWise Edge 閘道使用 Modbus TCP 和乙太網路/IP 來源、使用無效頻帶篩選傳入資料等。

2020 年 12 月 15 日

- 已新增[使用警示監控資料](#)區段，可用於在 中定義、設定和回應警示 AWS IoT SiteWise。
- 新增 [Edge 處理](#)區段，您可以用來設定在邊緣裝置上處理工業資料。
- 將 [Modbus TCP 和乙太網路/IP](#) 區段新增至 SiteWise Edge 閘道來源文件。
- 已新增[來源目的地](#)區段，可用來自訂您傳送傳入工業資料的位置。
- 新增 [OPC UA 篩選](#)區段，可用來控制從工業本機伺服器傳送至 SiteWise Edge 閘道的頻率和資料類型。

[AWS IoT SiteWise 現在支援客戶受管 CMKs。](#)

AWS IoT SiteWise 現在支援使用客戶受管 CMKs。

2020 年 11 月 24 日

IoT SiteWise 連接器版本 8 已發行	IoT SiteWise 連接器第 8 版已推出。當連接器遇到間歇性網路連線時，此版本可改善穩定性。	2020 年 11 月 19 日
在公式表達式中使用字串和條件	新增如何在轉換和指標的公式表達式中字串和條件函數的相關資訊。	2020 年 11 月 16 日
使用 AWS IoT Greengrass 串流管理員擷取資料	新增如何使用 AWS IoT Greengrass 邊緣裝置從本機資料來源擷取大量 IoT 資料的相關資訊。	2020 年 9 月 16 日
VPC 端點 (AWS PrivateLink)	新增如何透過建立介面 VPC 端點，在虛擬私有雲端 (VPC) 與 AWS IoT SiteWise 資料 APIs 之間建立私有連線的相關資訊。	2020 年 9 月 4 日
IoT SiteWise 連接器版本 7 已發佈	IoT SiteWise 連接器第 7 版已推出。此版本修正 SiteWise Edge 閘道指標的問題。	2020 年 8 月 14 日
從 AWS IoT SiteWise 主控台建立 IAM Identity Center 使用者	新增如何在 AWS IoT SiteWise 主控台中建立 IAM Identity Center 使用者的相關資訊。您現在可以在將使用者指派給新的或現有的入口網站時建立 IAM Identity Center 使用者。已更新 視覺化和共用風力發電廠資料 教學課程，以使用此功能。此變更可減少教學課程中的步驟數量。	2020 年 8 月 4 日

[改善 SiteWise Edge 閘道疑難排解](#)

新增有關如何對 SiteWise Edge 閘道進行故障診斷，以及如何[匯出來源的 OPC UA 用戶端憑證](#)的其他資訊。

2020 年 6 月 18 日

[主控台任務文件](#)

新增了[為產業資產建立模型](#)、[查詢資產屬性資料](#)，以及[與其他服務互動](#)的主控台任務文件。您可以遵循這些指示來完成 AWS IoT SiteWise 主控台

2020 年 6 月 11 日

[分析匯出的資料教學課程](#)

新增教學課程，讓您了解如何使用 Amazon Athena 來分析使用匯出[功能 AWS CloudFormation 範本](#)匯出至 Amazon S3 的資產資料。

2020 年 5 月 27 日

[使用公式表達式改善](#)

新增有關 AWS IoT SiteWise 公式屬性行為的詳細資訊，並新增如何計算篩選資料點的範例。

2020 年 5 月 18 日

[IoT SiteWise 連接器第 6 版已發行](#)

IoT SiteWise 連接器第 6 版已推出。此版本新增對 CloudWatch 指標的支援，並自動探索新的 OPC UA 標籤。這表示當 OPC UA 來源的標籤變更時，您不需要重新啟動 SiteWise Edge 閘道。此版本的連接器需要串流管理員和 AWS IoT Greengrass Core 軟體 v1.10.0 或更高版本。

2020 年 4 月 29 日

[AWS IoT SiteWise 功能版本](#)

2020 年 4 月 29 日

AWS IoT SiteWise 功能版本。您現在可以使用 API 管理 SiteWise Edge 閘道、將標誌新增至入口網站、檢視 SiteWise Edge 閘道指標等。

- 已新增[將資料匯出至 Amazon S3](#) 區段，其中包含可用來將新資料值匯出至 Amazon S3 儲存貯體的 AWS CloudFormation 範本。
- 新增[設定資料來源](#)區段，以改善 SiteWise Edge 閘道來源文件，並包含新的 SiteWise Edge 閘道 APIs。
- 新增 SiteWise Edge 閘道指標區段，說明 SiteWise Edge 閘道發佈的 CloudWatch 指標。
- 已新增在 Amazon EC2 上設定 SiteWise Edge 閘道一節，其中包含可用來在 Amazon EC2 執行個體上快速設定 SiteWise Edge 閘道相依性的 AWS CloudFormation 範本。
- 新增[入口網站服務角色](#)章節，說明 SiteWise Monitor 入口網站的新許可功能。
- 已更新入口網站服務角色和入口網站標誌的[入口網站文件](#)。
- 已新增[標記您的 AWS IoT SiteWise 資源](#)區段。

- 已更新儀表板定義結構的[建立儀表板 \(CLI\)](#)章節。
- 已新增[安全](#)章節。

[從 擷取資料 AWS IoT Events](#)

新增如何在事件發生 AWS IoT Events 時從 擷取資料的相關資訊。

2020 年 4 月 20 日

[在 SiteWise Monitor 教學課程中視覺化和共用風力發電廠資料](#)

新增您可以遵循的教學課程，了解如何使用 AWS IoT SiteWise Monitor 視覺化和共用資產資料。

2020 年 3 月 12 日

[AWS IoT SiteWise 概念](#)

新增了 AWS IoT SiteWise 概念詞彙表，您可以用來了解服務及其常用術語。

2020 年 3 月 5 日

[已移除 AWS IoT Greengrass 安裝指示](#)

已從 AWS IoT SiteWise 使用者指南中移除 AWS IoT Greengrass 核心軟體安裝說明。[AWS IoT Greengrass 開發人員指南](#)提供裝置設定指令碼，以及 AWS IoT Greengrass 在其他平台上設定的指示，例如 Amazon EC2 和 Docker。

2020 年 2 月 14 日

[改善使用 AWS IoT Core 規則的擷取資料](#)

新增[有關如何使用](#)以及[如何對規則動作進行故障診斷](#)的詳細資訊，您可以使用它從 MQTT 訊息中擷取資料 AWS IoT Core。AWS IoT SiteWise

2020 年 2 月 14 日

[IoT SiteWise 連接器版本 5 已發行](#)

IoT SiteWise 連接器第 5 版已推出。此版本修正 AWS IoT Greengrass 了 Core 軟體 v1.9.4 的相容性問題。

2020 年 2 月 12 日

IoT SiteWise 連接器版本 4 已發佈	IoT SiteWise 連接器第 4 版已推出。此版本修正了 OPC UA 伺服器重新連線的問題。	2020 年 2 月 7 日
重組建模工業資產	將〈更新資產和模型〉一節重新編排為〈建立工業資產模型〉內的多個主題。 • 資產和模型狀態 • 管理的資料串流 AWS IoT SiteWise • 更新屬性值 • 關聯和取消關聯資產 • 更新資產和模型 • 在中刪除資產和模型 AWS IoT SiteWise	2020 年 2 月 4 日
從 AWS IoT 實物教學課程擷取資料	新增教學課程，讓您了解如何設定 AWS IoT SiteWise 規則動作，從新的或現有的 AWS IoT 實物機群擷取資料。	2020 年 2 月 4 日
從 擷取資料的重組 AWS IoT SiteWise	將擷取資料區段重組為兩個最上層區段： 查詢資產屬性值 ，以及 彙總和與其他 AWS 服務互動 。	2020 年 1 月 21 日
發佈屬性值更新至 Amazon DynamoDB 教學課程	新增教學課程，讓您了解如何使用屬性值通知將資產資料儲存在 DynamoDB 中。	2020 年 1 月 8 日
使用公式表達式	已新增公式表示式參考，以組織可在轉換和指標屬性中使用的常數和函數。將 資產屬性 重新編排為每種屬性類型的個別主題。	2020 年 1 月 7 日

使用 OPC UA 節點篩選條件	新增如何在新增 SiteWise Edge 閘道來源時，使用 OPC UA 節點篩選條件來改善 SiteWise Edge 閘道效能的相關資訊。	2020 年 1 月 3 日
升級連接器	新增如何在新連接器版本發佈時升級 SiteWise Edge 閘道的相關資訊。	2019 年 12 月 30 日
IoT SiteWise 連接器第 3 版已發行	IoT SiteWise 連接器第 3 版已推出。此版本移除 iot:* 許可要求。	2019 年 12 月 17 日
IoT SiteWise 連接器版本 2 已發佈	IoT SiteWise 連接器第 2 版已推出。此版本新增了對多個 OPC UA 秘密資源的支援。	2019 年 12 月 10 日
建立儀表板 (AWS CLI)	新增如何使用在 AWS IoT SiteWise Monitor 中建立儀表板的相關資訊 AWS CLI。	2019 年 12 月 6 日

[AWS IoT SiteWise 第 2 版已發行](#)

已發行第 2 版的預覽 AWS IoT SiteWise 版。您現在可以透過 OPC UA、MQTT 和 HTTP 擷取資料、在資產階層中建立資料模型，並使用 SiteWise Monitor 視覺化資料。

2019 年 12 月 2 日

- 已針對資產、資產模型和資產階層的變更重寫[資產模型](#)區段。
- 更新[資料擷取](#)區段，以包含 AWS IoT Greengrass 連接器步驟和非閘道資料擷取區段。
- 新增 [AWS IoT SiteWise Monitor](#) 章節和[單獨的應用程式指南](#)，說明如何使用 SiteWise Monitor Web 應用程式。
- 已新增 [從查詢資料 AWS IoT SiteWise](#) 和 [與其他 AWS 服務互動](#) 區段。
- 已重寫[入門](#)區段，以匹配已更新的示範體驗。

[AWS IoT SiteWise 第 1 版已發行](#)

發行第 1 版的初始預覽 AWS IoT SiteWise 版。

2019 年 2 月 25 日

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。