亚马逊云科技

用户指南

Amazon IoT SiteWise

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon IoT SiteWise: 用户指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

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

Table of Contents

什么是 Amazon IoT SiteWise？ 1
- 如何 Amazon IoT SiteWise 运作 2
  - 摄取工业数据 2
  - 对资产进行建模，为收集的数据提供上下文 3
  - 使用查询、警报和预测进行分析 3
  - 可视化操作 4
  - 存储数据 4
  - 与其他服务集成 4
- 的用例 Amazon IoT SiteWise 4
  - 制造业 5
  - 食品和饮料 5
  - 能源与公用事业 5
- 与 Amazon SDKs 5
- 概念 6

开始使用 11
- 要求 11
- 设置一个 Amazon 账户 12
  - 注册获取 Amazon Web Services 账户 12
  - 保护 IAM 用户 12

使用快速入门演示 13
- 创建演 Amazon IoT SiteWise 示 13
- 删除演 Amazon IoT SiteWise 示 15

教程 16
- 计算 OEE 16
  - 先决条件 16
  - 如何计算 OEE 17
- 摄取数据 19
  - 先决条件 20
  - 步骤 1：创建 Amazon IoT 策略 20
  - 第 2 步：创建 Amazon IoT 事物 22
  - 步骤 3：创建设备资产模型 24
  - 步骤 4：创建设备队列资产模型 25
  - 步骤 5：创建和配置设备资产 27
  - 步骤 6：创建和配置设备队列资产 27

步骤 7：在 Amazon IoT Core 中创建向设备资产发送数据的规则 28
步骤 8：运行设备客户端脚本 31
步骤 9：在完成教程后清理资源 37
在 Monito SiteWise r 中可视化和共享数据 39
先决条件 40
步骤 1：在 Monito SiteWise r 中创建入口 40
步骤 2：登录门户 44
步骤 3：创建风电场项目 46
步骤 4：创建控制面板以可视化风电场数据 49
步骤 5：浏览门户 55
步骤 6：在完成教程后清理资源 56
发布到 Amazon DynamoDB 59
先决条件 59
步骤 1：配置 Amazon IoT SiteWise 以发布属性值更新 60
步骤 2：在 Amazon IoT Core 中创建规则 62
步骤 3：创建 DynamoDB 表 64
步骤 4：配置 DynamoDB 规则操作 66
步骤 5：在 DynamoDB 中浏览数据 67
步骤 6：在完成教程后清理资源 68
将数据摄取到 Amazon IoT SiteWise 73
管理数据流 73
配置权限和设置 74
将数据流与资产属性关联 76
取消数据流与资产属性的关联 77
删除数据流 77
更新资产属性的别名 78
常见场景 79
使用摄取数据 Amazon IoT SiteWise APIs 81
BatchPutAssetPropertyValue API 81
CreateBulkImportJob API 84
使用 Amazon IoT Core 规则 91
为所需的访问权限 92
配置规则操作 93
通过基本摄取降低成本 100
使用 Amazon IoT Events 动作 101
使用 Amazon IoT Greengrass 直播管理器 102

使用 SiteWise 边缘网关 103
网关关键概念 103
实施 SiteWise Edge 的好处 104
自托管网关 104
要求 105
创建网关 110
安装网关软件 112
支持 MQTT 的 V3 网关 115
经典直播、V2 网关 143
添加数据源 154
SiteWise Edge 组件 183
筛选 资产 184
代理支持和信任存储 185
使用 APIs 191
在 Siemens Industrial Edge 上托管网关 207
安全性 208
Siemens Secure Storage和 Amazon IoT SiteWise Edge 应用程序 208
Siemens Industrial Edge设备的目的地 209
从预览版应用程序迁移 210
故障排除 211
Amazon IoT SiteWise Edge 应用程序更新日志 211
要求 212
创建网关 213
创建 Siemens Databus user 214
访问应用程序 215
安装应用程序 215
更新已安装的应用程序配置 217
目的地和路径过滤器 218
了解目的地 218
了解路径过滤器 221
添加实时目的地 224
使用 Amazon S3 添加缓冲目的地 232
添加路径过滤器 238
管理目的地 241
管理网关 244
使用控制 Amazon IoT SiteWise 台管理您的 SiteWise Edge 网关 244

使用 Amazon OpsHub 管理 SiteWise 边缘网关 Amazon IoT SiteWise 245
使用本地操作系统凭据访问您的 SiteWise Edge 网关 246
管理 SiteWise Edge 网关证书 249
更改 SiteWise Edge 网关组件包的版本 249
更新 Amazon IoT SiteWise 组件的版本 249
删除 SiteWise Edge 网关 250
备份和恢复网关 250
指标数据的每日备份 251
恢复 SiteWise Edge 网关 251
恢复 Amazon IoT SiteWise 数据 252
验证备份和恢复是否成功 253
传统网关（Amazon IoT Greengrass Version 1） 254
为工业资产建模 255
资产和模型状态 256
检查资产的状态 257
检查资产或组件模型的状态 259
资产模型版本 260
检索资产模型或组件模型的活动版本（控制台） 261
检索资产模型或组件模型的活动版本（Amazon CLI） 262
自定义复合模型（组件） 263
内联自定义复合模型 263
Component-model-based 自定义复合模型 265
使用路径来引用自定义复合模型属性 266
设置对象 IDs 268
处理对象 UUIDs 268
使用外部 IDs 269
创建 模型 270
在中创建资产模型 Amazon IoT SiteWise 271
创建组件模型 285
定义数据属性 288
创建自定义复合模型（组件） 361
创建资产 365
创建资产（控制台） 366
创建资产（Amazon CLI） 366
配置新资产 368
搜索 资产 368

先决条件 368
高级搜索开启 Amazon IoT SiteWise 控制台 368
更新属性值 371
关联资产和取消关联资产 374
关联和取消关联资产（控制台） 374
关联资产和取消关联资产（Amazon CLI） 376
更新资产和模型 377
更新中的资产 Amazon IoT SiteWise 378
更新资产模型和组件模型 379
更新自定义复合模型（组件） 384
对资产模型写入进行乐观锁定 387
删除中的资源和模型 Amazon IoT SiteWise 390
删除资产 391
删除资产模型 393
使用资产和模型进行批量操作 395
关键概念和术语 395
支持的功能 396
批量操作先决条件 396
运行批量导入作业 399
运行批量导出作业 401
作业进度跟踪和错误处理 404
导入元数据示例 409
导出元数据示例 424
元数据传输作业架构 426
通过警报监控数据 446
警报类型 446
警报状态 447
警报状态属性 448
为资产模型定义警报 451
定义 Amazon IoT Events 警报 453
定义外部警报 474
为资产配置警报 476
配置阈值（控制台） 477
配置阈值（Amazon CLI） 477
响应警报 479
响应警报（控制台） 480

响应警报（API） 483
摄取外部警报状态 483
映射外部警报状态流 484
摄取警报状态数据 485
Amazon IoT SiteWise 助手 487
配置 Amazon IoT SiteWise 助手 487
创建数据集 488
编辑数据集 492
删除数据集 494
Amazon IoT SiteWise 助手问题 495
使用监控数据 Amazon IoT SiteWise Monitor 496
SiteWise 监控角色 497
SAML 联合身份验证 498
SiteWise 显示器概念 499
开始使用 Amazon IoT SiteWise Monitor （经典） 500
创建门户 501
配置门户 502
邀请管理员 505
添加门户用户 506
创建控制面板（CLI） 509
为门户打开警报 514
在边缘启用您的门户 517
管理门户 517
开始使用 Amazon IoT SiteWise Monitor （AI 感知） 525
创建门户 526
配置门户 527
管理门户 529
删除门户 533
使用创建仪表板 Amazon CLI 534
门户网站登录 538
创建项目 539
更新项目 539
删除项目 540
创建控制面板 540
更新控制面板 542
删除控制面板 543

配置控制面板 543
从中查询数据 Amazon IoT SiteWise 563
查询当前资产值 563
查询资产属性的当前值（控制台） 564
查询资产属性的当前值（Amazon CLI） 564
查询历史资产属性值 565
查询资产属性聚合 567
资产属性的聚合（API） 567
资产属性的聚合（Amazon CLI） 568
Amazon IoT SiteWise 查询语言 570
先决条件 570
查询语言参考 571
与其他服务交互 579
了解 MQTT 主题中的资产属性 579
使用通知 580
启用资产属性通知（控制台） 580
启用资产属性通知（Amazon CLI） 581
查询通知 583
将数据导出到 Amazon S3 585
集成 Grafana 586
与集成 Amazon IoT TwinMaker 587
启用集成 588
整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker 588
检测设备异常 589
添加预测定义（控制台） 590
训练预测（控制台） 593
开始或停止对预测的推理（控制台） 594
添加预测定义（CLI） 595
训练预测并开始推理（CLI） 598
训练预测（CLI） 600
开始或停止对预测的推理（CLI） 601
管理数据存储 604
配置存储设置 605
数据留存的影响 605
为温层进行配置（控制台） 605
配置温层（Amazon CLI） 607

配置冷层（控制台） 610
配置冷层（Amazon CLI） 612
对存储设置进行故障排除 617
错误：存储桶不存在 617
错误：访问 Amazon S3 路径被拒 617
错误：无法代入角色 ARN 618
错误：无法访问跨区域 Amazon S3 存储桶 618
冷层中保存的数据的文件路径和架构 618
设备数据（测量） 619
指标、转换和聚合 622
资产元数据 626
资产层次结构元数据 629
存储数据索引文件 631
代码示例 633
基本功能 637
你好 Amazon IoT SiteWise 638
了解基础知识 641
操作 705
安全性 780
数据保护 780
互联网络流量隐私 781
Amazon IoT SiteWise 助理商务服务改进 781
数据加密 782
静态加密 782
传输中加密 785
密钥管理 786
身份和访问管理 787
受众 788
使用身份进行身份验证 788
如何 Amazon IoT SiteWise 与 IAM 配合使用 790
托管策略 806
服务相关角色 810
为警报设置权限 823
跨服务混淆了副手预防 Amazon IoT SiteWise 829
对 身份和访问进行故障排除 830
合规性验证 832

恢复能力 833
基础结构安全性 833
配置和漏洞分析 834
VPC 端点 834
支持的 API 操作 835
创建接口 VPC 终端节点 837
Amazon IoT SiteWise 通过接口 VPC 终端节点进行访问 838
创建 VPC 端点策略 839
安全最佳实践 840
保持组件更新 840
加密您的 SiteWise Edge 网关的文件系统 841
安全访问您的边缘配置 841
在 Siemens Industrial Edge Management 上保护数据安全 841
向 SiteWise 监控器用户授予可能的最低权限 841
不要暴露敏感信息 841
遵循 Amazon IoT Greengrass 安全最佳实践 842
另请参阅 842
日志记录和监控 843
监控服务日志 843
管理登录 Amazon IoT SiteWise 845
示例： Amazon IoT SiteWise 日志文件条目 846
监控 SiteWise 边缘网关日志 847
使用 Amazon CloudWatch 日志 847
使用服务日志 849
使用事件日志 850
使用 Amazon CloudWatch 指标进行监控 852
Amazon IoT Greengrass Version 2 网关指标 853
使用记录 API 调用 Amazon CloudTrail 861
Amazon IoT SiteWise 信息在 CloudTrail 861
Amazon IoT SiteWise 中的数据事件 CloudTrail 862
Amazon IoT SiteWise 中的管理事件 CloudTrail 864
示例： Amazon IoT SiteWise 日志文件条目 864
标记资源 866
在中使用标签 Amazon IoT SiteWise 866
用 tag Amazon Web Services Management Console 866
使用 Amazon IoT SiteWise API 进行标记 866

将标签与 IAM 策略结合使用 867
故障排除 Amazon IoT SiteWise 869
批量导入和导出故障排除 869
对门户进行故障排除 870
用户和管理员无法访问 Amazon IoT SiteWise 门户 870
网关故障排除 871
配置和访问 SiteWise Edge 网关日志 872
排除 SiteWise Edge 网关问题 872
对 Amazon IoT SiteWise Edge 应用程序进行故障排除 Siemens Industrial Edge 877
疑难解答 Amazon IoT Greengrass 问题 878
对 Amazon IoT SiteWise 规则操作进行故障排除 878
配置 Amazon IoT Core 日志 879
配置重新发布错误操作 880
排查规则问题 882
排查规则问题（Amazon IoT SiteWise） 883
对规则进行故障排除（DynamoDB） 884
端点和限额 889
了解如何查看、监控和管理 SageMaker 端点。 889
限额 889
Amazon IoT SiteWise 助手 API 限流限制 901
异常检测的配额 902
文档历史记录 903
..... cmxvi

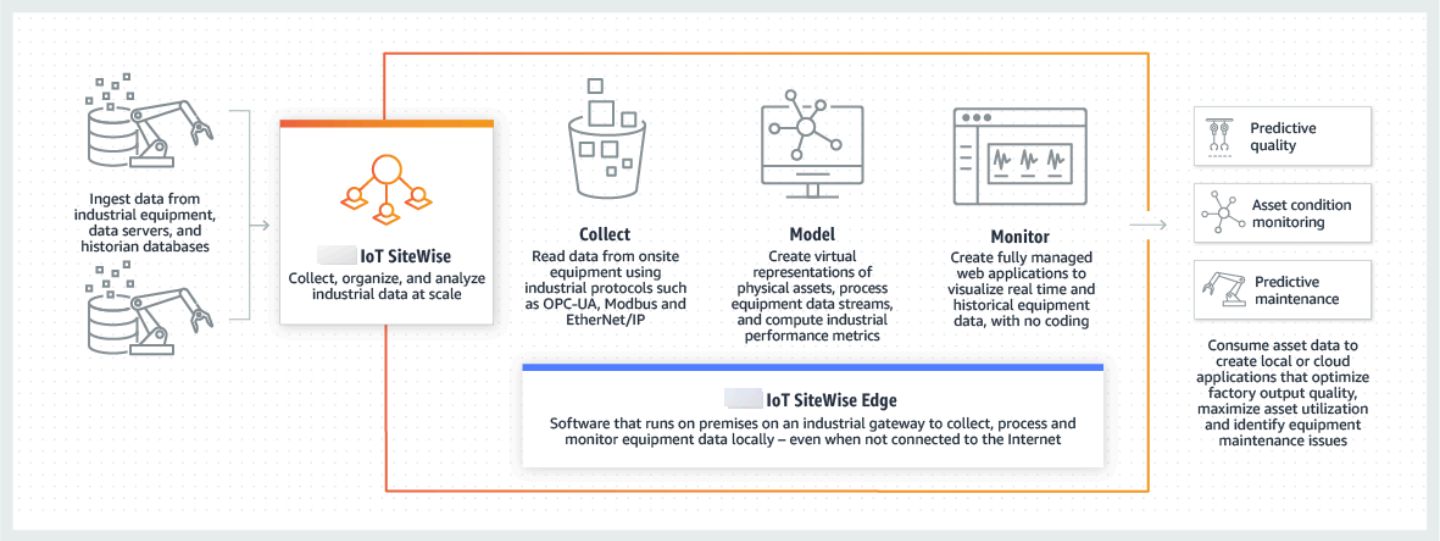
什么是 Amazon IoT SiteWise？

Amazon IoT SiteWise 是一项托管服务，您可以使用它大规模收集、存储、组织和监控来自工业设备的数据，以帮助您做出更好的数据驱动型决策。您可以使用 Amazon IoT SiteWise 监控各设施的操作，快速计算常见的工业性能指标，并创建分析工业设备数据的应用程序，以防止出现昂贵的设备问题并缩小生产缺口。

借 Amazon IoT SiteWise Monitor助，您的操作用户可以创建 Web 应用程序来实时查看和分析您的工业数据。您可以通过配置和监控指标（如两次故障之间的平均时间和整体设备效率 (OEE)）了解您的工业运营情况。

Amazon IoT SiteWise Edge 是其中的一个组件 Amazon IoT SiteWise ，它允许在本地设备上收集、存储和处理数据。如果您的互联网访问权限有限或需要对数据保密，则此功能非常有用。

下图显示了的基本架构 Amazon IoT SiteWise：



主题

- 如何 Amazon IoT SiteWise 运作
- 的用例 Amazon IoT SiteWise
- 将此服务与 Amazon SDK 配合使用
- Amazon IoT SiteWise 概念

如何 Amazon IoT SiteWise 运作

Amazon IoT SiteWise 提供了一个资源建模框架，可用于创建工业设备、过程和设施的表示形式。您的设备和流程的表示形式在中称为资产模型 Amazon IoT SiteWise。使用资产模型，您可以定义要使用的原始数据以及如何将其处理为有用的指标。在 Amazon IoT SiteWise 控制台中为工业运营构建并可视化资产和模型。您还可以配置资产模型，以便在边缘或 Amazon 云端收集和处理数据。

主题

- 摄取工业数据
- 对资产进行建模，为收集的数据提供上下文
- 使用查询、警报和预测进行分析
- 可视化操作
- 存储数据
- 与其他服务集成

摄取工业数据

Amazon IoT SiteWise 通过摄取工业数据开始使用。摄取数据可通过以下几种方法完成：

- 直接从现场服务器摄取：利用 OPC UA 等协议直接从现场设备读取数据。在各种平台（例如常见的工业网关或虚拟服务器）上部署与 Amazon IoT Greengrass V2之兼容的 SiteWise Edge 网关软件。您最多可以将 100 台 OPC UA 服务器连接到单个 Amazon IoT SiteWise 网关。有关更多信息，请参阅 Amazon IoT SiteWise Edge 自托管网关要求。

  请注意，我们与之合作，支持 Modbus TCP 和以太网/IP (EIP) 等协议 Domatica 在... 的背景下 Amazon IoT Greengrass V2。
- 使用包进行边缘数据处理：通过添加包来增强您的 SiteWise 边缘网关，以启用全面的边缘功能。启用 SiteWise Edge 后 Amazon IoT Greengrass V2，数据处理直接在现场执行，然后使用数据 Amazon IoT Greengrass 流安全地传输到 Amazon 云端。有关更多信息，请参阅 在 Edge 中 SiteWise 设置 OPC UA 来源。
- 通过 Amazon S3 利用批量操作进行自适应摄取：处理大量资产或资产模型时，使用批量操作从 Amazon S3 存储桶中批量导入和导出资源。有关更多信息，请参阅 使用资产和模型进行批量操作。
- 带有 Amazon IoT 核心规则的 MQTT 消息：对于连接到 Amazon IoT Core 发送 MQTT 消息的设备，请使用 Amazon IoT 核心规则引擎将这些消息定向到 Amazon IoT SiteWise。如果您有设备连接到 Amazon IoT Core 发送 MQTT 消息，请使用 Amazon IoT 核心规则引擎将这些消息路由到。

Amazon IoT SiteWise有关更多信息，请参阅 Amazon IoT SiteWise 使用 Amazon IoT Core 规则提取数据。

- 事件触发的数据提取：使用Amazon IoT Events 操作在中配置物联 SiteWise 网操作， Amazon IoT Events 以便在事件发生时向其发送数据。 Amazon IoT SiteWise 有关更多信息，请参阅 从中提取数据 Amazon IoT SiteWiseAmazon IoT Events。
- Amazon IoT SiteWise API：您在边缘或云端的应用程序可以直接将数据发送到 Amazon IoT SiteWise。有关更多信息，请参阅 使用摄取数据 Amazon IoT SiteWise APIs。

对资产进行建模，为收集的数据提供上下文

摄取数据后，您可以使用这些数据通过构建物理操作模型来创建资产、流程和设施的虚拟表示形式。代表设备或进程的资产将数据流传输到 Amazon 云端。资产也可以表示逻辑设备分组。层次结构是通过关联资产以镜像复杂操作而形成的。通过这些层次结构，资产可以访问关联的子资产中的数据。从资产模型创建资产。资产模型是标准化资产格式的声明式结构。重复使用资产组件，以实现模型的组织性和可维护性。有关更多信息，请参阅 为工业资产建模。

借 Amazon IoT SiteWise助，您可以配置资产，将传入的数据转换为上下文指标并进行转换。

- 接收设备数据时，转换会起作用。
- 指标按您定义的时间间隔进行计算。

指标和转换既适用于单个资产，也适用于多个资产。Amazon IoT SiteWise 自动计算与您的设备数据、指标和转换相关的不同时间范围内的常用统计聚合，例如平均值、总和与计数。

可以使用同步资产 Amazon IoT TwinMaker。有关更多信息，请参阅 整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker。

使用查询、警报和预测进行分析

Amazon IoT SiteWise 通过运行查询和设置警报来分析收集的日期。您还可以使用 Amazon Lookout 自动检测指标中的异常情况并确定其根本原因。

- 设置特定警报，以便在设备或流程运行不佳时提醒您的团队，确保快速识别和解决问题。有关更多信息，请参阅 使用警报监控数据 Amazon IoT SiteWise。
- 使用 Amazon IoT SiteWise API 操作查询资产属性的当前值、历史值以及特定时间间隔内的聚合。有关更多信息，请参阅 从中查询数据 Amazon IoT SiteWise。

- 在 Amazon Lookout for Equipment 中使用异常检测来识别和可视化设备或操作条件的变化。通过异常检测，您可以为运营确定预防性维护措施。这种集成支持客户在 Amazon IoT SiteWise 和 Amazon Lookout for Equipment 之间同步数据。有关更多信息，请参阅 使用 Lookout for Equipment 检测异常。

可视化操作

设置 M SiteWise onitor，为您的运营员工创建 Web 应用程序。Web 应用程序可帮助员工将操作可视化。使用 IAM Identity Center 或 IAM 为员工处理不同级别的访问权限。为每位员工配置唯一的登录名和权限，以查看整个工业运营的特定子集。 Amazon IoT SiteWise 为这些员工提供了学习如何使用 Monito SiteWise r 的应用指南。

有关可视化操作的更多信息，请参阅使用监控数据 Amazon IoT SiteWise Monitor。

存储数据

您可以将时序存储与工业数据湖集成。 Amazon IoT SiteWise 有三个用于存储工业数据的存储层：

- 专为实时应用程序优化的热存储层。
- 针对分析工作负载进行优化的温存储层。
- 基于 Amazon S3 的客户托管冷存储层，用于具有高延迟容忍度的操作数据应用程序。

Amazon IoT SiteWise 通过将最新数据保存在热存储层中来帮助您管理存储成本。然后，您可以定义数据留存策略，将历史数据移动到温层或冷层存储。有关更多信息，请参阅 管理中的数据存储 Amazon IoT SiteWise。

您还可以导入和导出资产元数据。有关更多信息，请参阅 资产元数据。

与其他服务集成

Amazon IoT SiteWise 与多项 Amazon 服务集成，可在 Amazon 云端开发完整的 Amazon IoT 解决方案。有关更多信息，请参阅 与其他 Amazon 服务互动。

的用例 Amazon IoT SiteWise

Amazon IoT SiteWise 用于各行各业，用于许多工业数据收集和分析应用程序。

始终如一地从所有来源收集数据，以帮助快速解决问题。 Amazon IoT SiteWise 提供远程监控，可直接在现场收集数据，或从多个设施的多个来源收集数据。 Amazon IoT SiteWise 为工业物联网数据解决方案提供了必要的灵活性。

制造业

Amazon IoT SiteWise 可以简化从设备中收集和利用数据的过程，从而精确定位并最大限度地减少效率低下现象，从而增强工业运营。 Amazon IoT SiteWise 帮助您从生产线和设备收集数据。借 Amazon IoT SiteWise助，您可以将数据传输到 Amazon 云端，并为您的特定设备和流程建立性能指标。您可以使用生成的指标来了解运营的总体效果，并确定创新和改进的机会。您还可以查看您的制造过程，并识别设备和工艺缺陷、生产缺口或产品缺陷。

食品和饮料

食品和饮料工业设施可处理各个种类的食品加工，包括将谷物磨成面粉、屠宰和包装肉类，以及组装、烹饪和冷冻适于微波炉烧煮的餐食。食品加工厂通常跨越多个地点，由工厂和设备操作员在一个集中的位置监控流程与设备。例如，制冷装置可评估原料的处理方式和过期情况。他们监控各设施的废物产生，以确保运营效率。借 Amazon IoT SiteWise助，您可以按生产线和设施对来自多个位置的传感器数据流进行分组，以便您的过程工程师可以更好地了解各个设施并进行改进。

能源与公用事业

借 Amazon IoT SiteWise助，您可以更轻松、更高效地解决设备问题。您可以远程实时监控资产性能。从任何地方访问历史设备数据，以查明潜在问题，调度准确的资源，并更快地预防和修复问题。

将此服务与 Amazon SDK 配合使用

Amazon 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档
Amazon CLI
适用于 Java 的 Amazon SDK
适用于 JavaScript 的 Amazon SDK
适用于 .NET 的 Amazon SDK

SDK 文档
适用于 PHP 的 Amazon SDK
Amazon Tools for PowerShell
适用于 Python (Boto3) 的 Amazon SDK
适用于 Ruby 的 Amazon SDK
适用于 SAP ABAP 的 Amazon SDK

Amazon IoT SiteWise 概念

以下是以下的核心概念 Amazon IoT SiteWise：

聚合

聚合是 Amazon IoT SiteWise 自动计算所有时间序列数据的基本指标或度量值。有关更多信息，请参阅 在中查询资产属性聚合 Amazon IoT SiteWise。

资产

当你 Amazon IoT SiteWise 从工业设备中输入或摄取数据时，你的设备、设备和流程都显示为资产。每个资产都有关联的数据。例如，一套设备可能具有序列号、位置、品牌和型号以及安装日期。它还可能具有可用性、性能、质量、温度、压力等时间序列值。将资产分组到层次结构，支持资产访问存储在其子资产中的数据。有关更多信息，请参阅 为工业资产建模。

资产层次结构

设置资产层次结构来创建工业运营的逻辑表示形式。为此，请在资产模型中定义一个层次结构，并将根据该模型创建的资产与指定的层次结构相关联。父资产中的指标可以组合来自子资产属性的数据，便于您计算指标，深入了解整体运营情况或其中的特定部分。有关更多信息，请参阅 定义资产模型层次结构。

资产模型

每个资产都是使用资产模型创建的。资产模型是定义和标准化资产格式的结构。他们确保在同一类型的多个资产之间提供一致信息，以便您可以处理表示设备组的资产中的数据。在每个资产模型中，您可以定义属性、时间序列输入（测量值）、时间序列转换（转换）、时间序列聚合（指标）和资产层次结构。有关更多信息，请参阅 为工业资产建模。

通过为边缘配置资产模型来决定资产模型属性的处理位置。使用此特征可处理和监控本地设备上的资产数据。

资产属性

资产属性是每个资产中保存工业数据的结构。每个属性都有一个数据类型，也可以有一个单位。特性可以是属性、测量值、转换或指标。有关更多信息，请参阅 定义数据属性。

属性

属性是资产的属性，通常保持不变，例如设备制造商或设备位置。属性可以具有预设值。从资产模型创建的每个资产都包含该模型中定义的属性默认值。有关更多信息，请参阅 定义静态数据（属性）。

控制面板

每个项目都包含一组控制面板。控制面板为一组资产的值提供一组可视化效果。项目拥有者创建控制面板及其包含的可视化效果。当项目拥有者准备好共享控制面板集时，拥有者可以邀请查看者加入项目，从而使他们能够访问项目中的所有控制面板。如果希望为不同的控制面板使用一组不同的查看器，则必须在项目之间划分控制面板。在查看者查看控制面板时，他们可以自定义用于查看特定数据的时间范围。

数据集

数据集是表示时间序列数据、数据和非设备 non-time-series数据（例如轮班计划、维护记录和员工数据库）的数据集合。它们支持外部数据并使用 Amazon IoT SiteWise 分析功能。它包括数据集源、数据集架构和数据集参数。 Amazon IoT SiteWise 助手使用消耗 Amazon Kendra 索引的数据集。

数据流

Amazon IoT SiteWise 甚至在创建资产模型和资产之前，就输入或摄取工业数据。 Amazon IoT SiteWise 自动生成数据流以从您的设备收集原始数据流。

数据流别名

数据流别名可帮助您轻松识别数据流。例如，别名 `server1-windfarm/3/turbine/7/temperature` 表示来自风电场 #3 中涡轮机 #7 的温度值。术语 `server1` 是帮助识别 OPC UA 服务器的数据来源名称，`server1-` 是附加到从此 OPC UA 服务器报告的所有数据流的前缀。

数据流关联

创建资产模型和资产后，将数据流与资产中定义的资产属性相关联以构建数据。 Amazon IoT SiteWise 然后可以使用资产模型和资产来处理来自数据流的传入数据。您还可以取消数据流与资产属性的关联。有关更多信息，请参阅 管理数据流 Amazon IoT SiteWise。

目的地

SiteWise Edge 中的目的地表示您要将遥测数据或处理过的数据发送到的终端节点。 SiteWise Edge 支持将 Amazon IoT SiteWise 热层、缓冲摄取或 Amazon S3 存储桶作为目标。您可以使用路径过滤器将目的地配置为订阅特定的 MQTT 主题。有关更多信息，请参阅 了解 Amazon IoT SiteWise 边缘目的地。

公式

每个转换和指标属性都附带一个公式，概述了该属性如何转换或聚合数据。这些公式包含属性输入、运算符和 Amazon IoT SiteWise提供的函数。有关更多信息，请参阅 使用公式表达式。

测量值

测量值是描述设备或装置的原始传感器时间序列数据流的资产属性。有关更多信息，请参阅 定义来自设备的数据流（测量值）。

指标

指标是表示聚合时间序列数据的资产属性。每个指标都附有一个数学表达式（公式），概述了如何聚合数据点，以及计算该聚合的时间间隔。指标在每个指定的时间间隔内生成一个数据点。有关更多信息，请参阅 聚合来自属性和其他资产（指标）的数据。

MQTT

MQTT（消息队列遥测传输）是一种用于传感器和设备的轻量级消息协议。

OPC UA

OPC UA（开放平台通信统一架构）是一种用于工业自动化的通信协议。

路径过滤器

在网关中使用路径过滤器订阅 MQTT 主题并发布到 Amazon IoT SiteWise 支持的目的地。基于 MQTT 的源、数据处理管道和目标都在支持 MQTT 的自托管的 V3 网关上使用 MQTT 主题交换数据。您可以定义主题筛选器来指定要采集或路由到不同目的地的数据。

门户

Amazon IoT SiteWise Monitor 门户是一个 Web 应用程序，可用于可视化和共享 Amazon IoT SiteWise 数据。门户具有一个或多个管理员，并且包含零个或多个项目。

门户管理员

每个 M SiteWise onitor 门户都有一个或多个门户管理员。门户管理员使用门户创建包含资产和控制面板集合的项目。门户管理员随后会将资产和拥有者分配给每个项目。通过控制对项目的访问，门户管理员可以指定项目拥有者和查看者可以查看哪些资产。

Project

每个 M SiteWise onitor 门户都包含一组项目。每个项目都有一个与之关联的 Amazon IoT SiteWise 资产子集。项目拥有者创建一个或多个控制面板，以提供一致的方式来查看与这些资产关联的数据。项目拥有者可以邀请查看者加入项目，以允许他们查看项目中的资产和控制面板。该项目是 Monito SiteWise r 内部共享的基本单元。项目所有者可以邀请 Amazon 管理员授予门户访问权限的用户。用户必须具有门户访问权限，然后才能与该用户共享该门户中的项目。

项目拥有者

每个 M SiteWise onitor 项目都有所有者。项目所有者以控制面板形式创建可视化效果，来一致地呈现您的操作数据。当控制面板准备好共享时，项目所有者可以邀请查看者加入项目。项目所有者还可以将其他所有者分配给项目。项目所有者可以为警报配置阈值和通知设置。

项目查看者

每个 M SiteWise onitor 项目都有查看器。项目查看者可以连接到门户以查看项目拥有者创建的控制面板。在每个控制面板中，项目查看者可以调整时间范围以更好地了解运营数据。项目查看者只能查看其具有访问权限的项目中的控制面板。项目查看者可以确认和暂停警报。

属性别名

您可以选择在资产属性上创建别名，例如 OPC UA 服务器数据流路径（例如，/company/windfarm/3/turbine/7/temperature），从而简化资产数据摄取或检索期间资产属性的识别。当您使用 SiteWise Edge 网关从服务器提取数据时，您的属性别名必须与原始数据流的路径相匹配。有关更多信息，请参阅 管理数据流 Amazon IoT SiteWise。

属性通知

当您为资产属性启用属性通知时， Amazon IoT Core 每当该资产收到新值时，都会向其 Amazon IoT SiteWise 发布一条 MQTT 消息。消息有效载荷包含有关该属性值更新的详细信息。使用财产价值通知来创建将您的工业数据 Amazon IoT SiteWise 与其他 Amazon 服务连接起来的解决方案。有关更多信息，请参阅 与其他 Amazon 服务互动。

SiteWise 边缘网关

E SiteWise dge 网关安装在客户所在地，用于收集、处理和指挥数据。 SiteWise 边缘网关通过各种协议连接到您的工业数据源，以收集和处理数据，然后将其发送到 Amazon 云端。 SiteWise 边缘网关还可以连接到合作伙伴的数据源。有关更多信息，请参阅 使用 Amazon IoT SiteWise 边缘网关。

转换

转换是表示转换后的时间序列数据的资产属性。每个转换都附有一个数学表达式（公式），用于指定如何将数据点从一种形式转换为另一种形式。变换后的数据点与输入数据点有关 one-to-one系。有关更多信息，请参阅 转换数据（转换）。

可视化

在每个控制面板中，项目拥有者决定如何显示与项目关联的资产的属性和警报。可用性可以用折线图表示，而其他值可能显示为条形图或关键绩效指标 (KPIs)。警报最好以状态网格和状态时间轴的形式显示。项目所有者可以自定义每个可视化效果，以便更好地了解该资产的数据。

开始使用 Amazon IoT SiteWise

借 Amazon IoT SiteWise助，您可以收集、组织、分析和可视化您的数据。

Amazon IoT SiteWise 提供了一个演示，您无需配置真实数据源即可使用该演示来探索服务。有关更多信息，请参阅 使用演 Amazon IoT SiteWise 示。

您可以完成以下教程来探索以下各项的某些功能 Amazon IoT SiteWise：

- 从事物中摄 Amazon IoT SiteWise 取 Amazon IoT 数据
- 在 Monito SiteWise r 中可视化和共享风电场数据
- 将属性值更新发布到 Amazon DynamoDB

要了解更多信息，请参阅以下主题 Amazon IoT SiteWise：

- 将数据摄取到 Amazon IoT SiteWise
- 为工业资产建模
- 使用监控数据 Amazon IoT SiteWise Monitor
- 从中查询数据 Amazon IoT SiteWise
- 与其他 Amazon 服务互动

主题

- 要求
- 设置一个 Amazon 账户

要求

你必须有一个 Amazon 账户才能开始使用 Amazon IoT SiteWise。如果您没有相应账户，请参阅以下部分，获取有关如何设置账户的说明。

使用可 Amazon IoT SiteWise 用的区域。有关更多信息，请参阅 Amazon IoT SiteWise 终端节点和限额。您可以使用中的区域选择器切换 Amazon Web Services Management Console 到其中一个区域。

设置一个 Amazon 账户

主题

- 注册获取 Amazon Web Services 账户
- 保护 IAM 用户

注册获取 Amazon Web Services 账户

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

要注册 Amazon Web Services 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话，要求使用电话键盘输入一个验证码。

   当您注册时 Amazon Web Services 账户，就会创建Amazon Web Services 账户根用户一个。根用户有权访问该账户中的所有 Amazon Web Services 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

Amazon 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择 “我的账户”，查看您当前的账户活动并管理您的账户。

保护 IAM 用户

注册后 Amazon Web Services 账户，开启多重身份验证 (MFA)，保护您的管理用户。有关说明，请参阅《IAM 用户指南》中的 为 IAM 用户启用虚拟 MFA 设备（控制台）。

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

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

- 在你的 IAM 用户中创建 Amazon Web Services 账户
- 适用于 Amazon 资源的访问权限管理
- 基于 IAM 身份的策略示例

使用演 Amazon IoT SiteWise 示

您可以使用 Amazon IoT SiteWise 演示 Amazon IoT SiteWise 轻松探索。 Amazon IoT SiteWise 将演示作为 Amazon CloudFormation 模板提供，您可以部署该模板来创建资产模型、资产和 M SiteWise onitor 门户，并生成长达一周的示例数据。

> **Important**
>
> 创建演示后，此演示创建和使用的资源即会开始计费。

主题

- 创建演 Amazon IoT SiteWise 示
- 删除演 Amazon IoT SiteWise 示

创建演 Amazon IoT SiteWise 示

您可以从 Amazon IoT SiteWise 控制台创建 Amazon IoT SiteWise 演示。

> **Note**
>
> 该演示创建了 Lambda 函数、一个 CloudWatch 事件规则和演示所需的 Amazon Identity and Access Management (IAM) 角色。您可能会在自己的 Amazon 账户中看到这些资源。我们建议您保留这些资源，直到您完成演示。如果您删除这些资源，演示可能会停止正常工作。

在 Amazon IoT SiteWise 控制台中创建演示

1. 导航到Amazon IoT SiteWise 控制台，在页面的右上角找到SiteWise 演示。
2. （可选）在 SiteWise demo 下，更改保留演示资产的天数字段，以指定在删除演示之前保留演示的天数。
3. （可选）要创建用于 SiteWise 监控示例数据的监控器入口，请执行以下操作。

> **Note**
>
> 您需要为本演示创建和使用的 SiteWise 监控器资源付费。有关更多信息，请参阅《Amazon IoT SiteWise 定价》中的 “SiteWise 监控”。

a. 请选择监控资源。

b. 请选择权限。

c. 请选择一个现有 IAM 角色来授予您的联合 IAM 用户访问门户的权限。

> **Important**
>
> 您的 IAM 角色必须包括以下权限。

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

有关如何使用 SiteWise 监控器的更多信息，请参阅什么是 Amazon IoT SiteWise Monitor？ 在《Amazon IoT SiteWise Monitor 应用指南》中。

4. 选择 创建演示。

   创建演示约需 3 分钟时间。如果创建演示失败，说明您的账户可能没有足够的权限。切换到具有管理权限的账户，或使用以下步骤删除演示并重试：

   a. 选择 删除演示。

      删除演示约需 15 分钟时间。

   b. 如果演示未删除，请打开Amazon CloudFormation 控制台，选择名为 Io 的堆栈 TSiteWiseDemoAssets，然后在右上角选择删除。

   c. 如果演示无法再次删除，请按照 Amazon CloudFormation 控制台中的步骤跳过删除失败的资源，然后重试。

5. 演示创建成功后，您可以浏览 Amazon IoT SiteWise 控制台中的演示资产和数据。

删除演 Amazon IoT SiteWise 示

Amazon IoT SiteWise 演示将在一周后自行删除，或者如果您从 Amazon CloudFormation 控制台创建演示堆栈，则该演示将在您选择的天数后自行删除。如果您已完成演示资源的使用，则可以提前删除演示。如果演示创建失败，您也可以删除演示。使用以下步骤手动删除演示。

要删除演 Amazon IoT SiteWise 示

1. 导航至 Amazon CloudFormation 控制台。
2. 选择 IoTSiteWiseDemoAssets来自堆栈列表。
3. 选择删除。

   当您删除堆栈时，为演示创建的所有资源都将被删除。

4. 在确认对话框中，选择 删除堆栈。

   删除堆栈约需 15 分钟时间。如果演示无法删除，请再次选择右上角的 删除。如果演示无法再次删除，请按照 Amazon CloudFormation 控制台中的步骤跳过删除失败的资源，然后重试。

Amazon IoT SiteWise 教程

欢迎来到教 Amazon IoT SiteWise 程页面。教程库在持续扩充，帮助您深入了解驾驭 Amazon IoT SiteWise复杂功能所需的知识和技能。这些教程提供了各种各样的基本主题，可以满足您的需求。在深入研究教程时，发现有关各个方面的宝贵见解。 Amazon IoT SiteWise

每个教程都使用特定的设备示例。这些教程适用于测试环境，使用的是虚构的公司名称、模型、资产和属性等等。其目的在于提供一般性指导。在未进行仔细审核并进行改进以满足组织独特需求的情况下，这些教程不适合在您的生产环境中直接使用。

主题

- 计算 OEE Amazon IoT SiteWise
- 从事物中摄 Amazon IoT SiteWise 取 Amazon IoT 数据
- 在 Monito SiteWise r 中可视化和共享风电场数据
- 将属性值更新发布到 Amazon DynamoDB

计算 OEE Amazon IoT SiteWise

本教程提供了一个示例，讲解如何计算制造过程的整体设备效率 (OEE)。因此，您的 OEE 计算或公式可能与此处显示的不同。通常，OEE 被定义为 `Availability * Quality * Performance`。要了解有关计算 OEE 的更多信息，请参阅维基百科上的整体设备效率。

先决条件

要完成本教程，您必须为具有以下三个数据流的设备配置数据引入：

- `Equipment_State` – 代表机器状态的数字代码，如闲置、故障、计划停止或正常运行。
- `Good_Count` – 一个数据流，其中每个数据点都包含自上一个数据点以来的成功操作数。
- `Bad_Count` – 一个数据流，其中每个数据点都包含自上一个数据点以来的不成功操作数。

要配置数据引入，请参阅将数据摄取到 Amazon IoT SiteWise。如果您没有可用的工业运营，您可以编写一个脚本，通过 Amazon IoT SiteWise API 生成并上传示例数据。

如何计算 OEE

在本教程中，您将创建一个资产模型，该模型通过三个数据输入流计算 OEE：`Equipment_State`、`Good_Count` 和 `Bad_Count`。在此示例中，请考虑通用包装机，如用于包装糖、薯片或油漆的包装机。在Amazon IoT SiteWise 控制台中，使用以下度量、转换和指标创建 Amazon IoT SiteWise 资产模型。然后，您可以创建一个资产来表示包装机器，并观察如何 Amazon IoT SiteWise 计算 OEE。

定义以下测量值来表示来自包装机的原始数据流。

测量值

- `Equipment_State` – 以数字代码提供包装机当前状态的数据流（或测量值）：
  - `1024` – 机器处于闲置状态。
  - `1020` – 故障，如错误或延迟。
  - `1000` – 计划的停止。
  - `1111` – 正常运行。
- `Good_Count` – 一个数据流，其中每个数据点都包含自上一个数据点以来的成功操作数。
- `Bad_Count` – 一个数据流，其中每个数据点都包含自上一个数据点以来的不成功操作数。

使用 `Equipment_State` 测量数据流及其包含的代码，定义以下转换（或派生的测量值）。变换 one-to-one与原始测量值有关。

转换

- `Idle = eq(Equipment_State, 1024)` – 包含机器闲置状态的转换后的数据流。
- `Fault = eq(Equipment_State, 1020)` – 包含机器故障状态的转换后的数据流。
- `Stop = eq(Equipment_State, 1000)` – 包含机器计划停止状态的转换后的数据流。
- `Running = eq(Equipment_State, 1111)` – 包含机器正常运行状态的转换后的数据流。

使用原始测量值和转换的测量值，定义在指定时间间隔内聚合机器数据的以下指标。定义本节介绍的指标时，为每个指标选择相同的时间间隔。

Metrics

- `Successes = sum(Good_Count)` – 在指定时间间隔内成功填充的包数。

- `Failures = sum(Bad_Count)` – 在指定时间间隔内未成功填充的包数。
- `Idle_Time = statetime(Idle)` – 机器在每个指定时间间隔内的总闲置时间（以秒为单位）。
- `Fault_Time = statetime(Fault)` – 机器在每个指定时间间隔内的总故障时间（以秒为单位）。
- `Stop_Time = statetime(Stop)` – 机器在每个指定时间间隔内的总计划停止时间（以秒为单位）。
- `Run_Time = statetime(Running)` – 机器在每个指定时间间隔内的总正常运行时间（以秒为单位）。
- `Down_Time = Idle_Time + Fault_Time + Stop_Time` – 机器在指定时间间隔内的总停机时间（以秒为单位），计算为除 `Run_Time` 以外的机器状态的时间总计。
- `Availability = Run_Time / (Run_Time + Down_Time)` – 机器在指定时间间隔内可以运行的正常运行时间，或计划运行时间的百分比。
- `Quality = Successes / (Successes + Failures)` – 机器在指定时间间隔内成功填充包的百分比。
- `Performance = ((Successes + Failures) / Run_Time) /` *`Ideal_Run_Rate`* – 在指定时间间隔内，您的进程以超出理想运行速率（以秒为单位）的百分比表示的机器性能。

  例如，您的 `Ideal_Run_Rate` 可能是每分钟 60 个包（每秒 1 个包）。如果您的 `Ideal_Run_Rate` 以分钟或小时为单位，则需要除以适当的单位转换系数，因为 `Run_Time` 以秒为单位。
- `OEE = Availability * Quality * Performance` – 机器在指定时间间隔内的整体设备效率。此公式将 OEE 计算为 1 的比例。

> **Note**
>
> 如果将 OEE 定义为转换，则系统会针对每个输入值计算输出值。由于转换评估会使用公式中所有相关属性的最新可用值，因此可能会生成意想不到的值。对于具有相同时间戳的属性更新，输出值可能会被来自其他传入属性的更新所覆盖。例如，在计算可用性、质量和性能时，OEE 是基于其他两个属性的最新可用数据点计算的。这些用于计算的值共享时间戳，这就导致 OEE 的输出值出现错误。转换计算的顺序无法得到保证。

从事物中摄 Amazon IoT SiteWise 取 Amazon IoT 数据

在本教程中，学习如何使用设备影子 Amazon IoT SiteWise 从一组设备中提取数据。 Amazon IoT 设备影子是存储 Amazon IoT 设备当前状态信息的 JSON 对象。有关更多信息，请参阅 Amazon IoT 开发人员指南中的设备影子服务。

完成本教程后，可以在中 Amazon IoT SiteWise 根据 Amazon IoT 内容设置操作。通过使用 Amazon IoT 东西，您可以将您的操作与的其他有用功能集成 Amazon IoT。例如，您可以配置 Amazon IoT 功能以执行以下任务：

- 配置其他规则以将数据流式传输到 Amazon IoT Events、Amazon DynamoDB 和其他 Amazon 服务。有关更多信息，请参阅 Amazon IoT 开发人员指南 中的规则。
- 使用 Amazon IoT 队列索引服务对您的设备数据进行索引、搜索和汇总。有关更多信息，请参阅 Amazon IoT 开发人员指南中的实例集索引服务。
- 使用来审核和保护您的设备 Amazon IoT Device Defender。有关更多信息，请参阅 Amazon IoT 开发人员指南中的 Amazon IoT Device Defender。

在本教程中，您将学习如何将数据从设备影子提取 Amazon IoT 到中的资产。 Amazon IoT SiteWise 为此，您需要创建一件或多件 Amazon IoT 事，然后运行一个脚本，该脚本使用 CPU 和内存使用数据更新每件事物的设备影子。在本教程中，您可以使用 CPU 利用率和内存使用量数据来模拟真实传感器数据。然后，您创建一条规则，其中包含一个 Amazon IoT SiteWise 操作，该操作会在 Amazon IoT SiteWise 每次事物的设备影子更新时将这些数据发送到资产。有关更多信息，请参阅 Amazon IoT SiteWise 使用 Amazon IoT Core 规则提取数据。

主题

- 先决条件
- 步骤 1：创建 Amazon IoT 策略
- 步骤 2：创建和配置 Amazon IoT 事物
- 步骤 3：创建设备资产模型
- 步骤 4：创建设备队列资产模型
- 步骤 5：创建和配置设备资产
- 步骤 6：创建和配置设备队列资产
- 步骤 7：在 Amazon IoT Core 中创建向设备资产发送数据的规则
- 步骤 8：运行设备客户端脚本
- 步骤 9：在完成教程后清理资源

先决条件

要完成本教程，您需要：

- 一个 Amazon 账户。如果没有，请参阅设置一个 Amazon 账户。
- 一台正在运行的开发计算机 Windows, macOS, Linux，或 Unix 访问 Amazon Web Services Management Console。有关更多信息，请参阅 Amazon Web Services Management Console入门。
- 具有管理员权限的 Amazon Identity and Access Management (IAM) 用户。
- Python 3 安装在您的开发计算机上或安装在要注册为 Amazon IoT 事物的设备上。

步骤 1：创建 Amazon IoT 策略

在此过程中，创建一个允许您的设备访问 Amazon IoT 本教程中使用的资源的 Amazon IoT 策略。

创建 Amazon IoT 策略

1. 登录到 Amazon Web Services Management Console。
2. 查看支持AmazonAmazon IoT SiteWise 的地区。如有必要，切换到其中一个受支持的区域。
3. 导航至 Amazon IoT 控制台。如果出现连接设备按钮，请选择该按钮。
4. 在左侧导航窗格中，选择安全，然后选择策略。
5. 选择创建。
6. 输入 Amazon IoT 策略的名称（例如，**SiteWiseTutorialDevicePolicy**）。
7. 在政策文档下，选择 JSON，以 JSON 形式输入以下政策。*account-id*用您的地区*region*和账户 ID 替换和，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*"
    }
  ]
}
```

此策略使您的 Amazon IoT 设备能够使用 MQTT 消息建立连接并与设备影子通信。有关 MQTT 消息的更多信息，请参阅什么是 MQTT？。为了与设备影子进行交互，你的 Amazon IoT 东西会发布和接收有关以开头的主题的 MQTT 消息。`$aws/things/thing-name/shadow/`此策略包含一个称为 `${iot:Connection.Thing.ThingName}` 的事物策略变量。此变量用于替换每个主题中连接事物的名称。`iot:Connect` 语句限制了可以建立连接的设备，确保此事物策略变量只能替换以 `SiteWiseTutorialDevice` 开头的名称。

有关更多信息，请参阅Amazon IoT 开发人员指南中的事物策略变量。

> **Note**
>
> 此策略适用于名称以 `SiteWiseTutorialDevice` 开头的事物。要为您的事物使用不同的名称，您必须相应地更新策略。

8. 选择创建。

步骤 2：创建和配置 Amazon IoT 事物

在此过程中，您将创建和配置一个 Amazon IoT 事物。你可以将你的开发计算机指定为一个 Amazon IoT 东西。随着学习的深入，请记住，您在本教程中掌握的原则可以应用到实际项目中。你可以灵活地在任何能够运行 Amazon IoT SDK 的设备上制作和设置 Amazon IoT 东西，包括 Amazon IoT Greengrass 和 FreeRTOS。有关更多信息，请参阅 Amazon IoT 开发人员指南中的 Amazon IoT SDKs。

创建和配置 Amazon IoT 事物

1. 打开命令行并运行以下命令，以便为本教程创建目录。

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

2. 运行以下命令为您的事物证书创建一个目录。

```
mkdir device1
```

   如果您正在创建其他事物，请相应地增加目录名称中的数字，以跟踪哪些证书属于哪个事物。

3. 导航至 Amazon IoT 控制台。
4. 在左侧导航窗格的管理部分中，选择所有设备。然后选择 Things (事物)。
5. 如果显示 您还没有任何事物 对话框，请选择 创建事物。否则，请选择创建事物。
6. 在创建事物页面上，选择创建单个事物，然后选择下一步。
7. 在指定事物属性页面上，为 Amazon IoT 事物输入名称（例如 **`SiteWiseTutorialDevice1`**），然后选择下一步。如果您正在创建其他事物，请相应地增加事物名称中的数字。

   > **Important**
   > 事物名称必须与您在步骤 1：创建策略中创建的 Amazon IoT 策略中使用的名称相匹配。否则，您的设备将无法连接到 Amazon IoT。

8. 在配置证书 - 可选页面上，选择自动生成新证书（推荐），然后选择下一步。证书可以安全 Amazon IoT 地识别您的设备。
9. 在将策略附加到证书-可选页面上，选择您在步骤 1：创建策略中创建的 Amazon IoT 策略，然后选择创建事物。
10. 在下载证书与密钥对话框中，执行以下操作：

    a. 选择 下载 链接以下载事物的证书、公钥和私钥。将所有三个文件保存到您为事物证书创建的目录（例如，`iot-sitewise-rule-tutorial/device1`）。

    > **Important**
    > 这是您唯一一次可以下载在设备成功连接 Amazon IoT时所需的事物证书和密钥。

    b. 选择下载链接，以下载根 CA 证书。将根 CA 证书保存到 `iot-sitewise-rule-tutorial`。我们建议下载 Amazon 根 CA 1。

11. 选择完成。

现在，你已经在电脑上注册了一个 Amazon IoT 东西。现在，执行以下后续步骤之一：

- 继续步骤 3：在不创建其他 Amazon IoT 内容的情况下创建设备资产模型。您可以只使用一个事物完成本教程。
- 在另一台计算机或设备上重复本部分中的步骤以创建更多 Amazon IoT 事物。在本教程中，我们建议您按照此选项操作，以便从多个设备提取唯一的 CPU 利用率和内存使用量数据。
- 在同一台设备（您的计算机）上重复此部分中的步骤以创建更多 Amazon IoT 事物。每个 Amazon IoT 设备都会从您的计算机接收相似的 CPU 和内存使用数据，因此请使用这种方法来演示从多个设备摄取非唯一数据。

步骤 3：创建设备资产模型

在此过程中，您将在中创建一个资产模型 Amazon IoT SiteWise 来表示流式传输 CPU 和内存使用情况数据的设备。为处理表示设备组的资产中的数据，资产模型会在同一类型的多个资产之间强制提供一致信息。有关更多信息，请参阅 为工业资产建模。

创建表示设备的资产模型

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择 模型。
3. 选择 创建模型。
4. 在模型详细信息下，输入您的模型的名称。例如，**SiteWise Tutorial Device Model**。
5. 在 测量定义 下，执行以下操作：

   a. 在名称 中，输入 **CPU Usage**。

   b. 在 单位 中，输入 **%**。

   c. 将 数据类型 保留为 双精度。

   测量属性表示设备的原始数据流。有关更多信息，请参阅 定义来自设备的数据流（测量值）。
6. 选择添加测量以添加第二个测量属性。
7. 在 测量定义 下的第二行中，执行以下操作：

   a. 在名称 中，输入 **Memory Usage**。

   b. 在 单位 中，输入 **%**。

   c. 将 数据类型 保留为 双精度。
8. 在 指标定义 下，执行以下操作：

a. 在名称中，输入 **Average CPU Usage**。

b. 在 公式 中，输入 **avg(CPU Usage)**。选择 CPU Usage出现时从自动完成列表中显示。

c. 在 时间间隔 中，输入 **5 minutes**。

指标属性可定义聚合计算，用于处理一个时间间隔内的所有输入数据点，并针对每个时间间隔输出一个数据点。此指标属性每 5 分钟计算一次每台设备的平均 CPU 利用率。有关更多信息，请参阅聚合来自属性和其他资产（指标）的数据。

9. 选择添加指标以添加第二个指标属性。

10. 在 指标定义 下的第二行中，执行以下操作：

a. 在名称中，输入 **Average Memory Usage**。

b. 在 公式 中，输入 **avg(Memory Usage)**。选择 Memory Usage出现时从自动完成列表中显示。

c. 在 时间间隔 中，输入 **5 minutes**。

此指标属性每 5 分钟计算一次每台设备的平均内存使用量。

11. （可选）添加您对每个设备进行计算时感兴趣的其他指标。一些有趣的函数包括 `min` 和 `max`。有关更多信息，请参阅 使用公式表达式。在步骤 4：创建一个设备实例集资产模型，您将创建一个父资产，该资产可以使用来自整个设备队列的数据来计算指标。

12. 选择创建模型。

步骤 4：创建设备队列资产模型

在此过程中，您将制作一个资产模型 Amazon IoT SiteWise 来符号化您的设备集合。在此资产模型中，您构建一个结构，来支持将大量设备资产关联到一个总体队列资产。接下来，您在队列资产模型中设定指标，以整合来自所有已连接设备资产的数据。通过这种方法，您可以全面了解整个队列的整体表现。

创建表示设备队列的资产模型

1. 导航至 Amazon IoT SiteWise 控制台。

2. 在左侧导航窗格中，选择 模型。

3. 选择 创建模型。

4. 在模型详细信息下，输入您的模型的名称。例如，**SiteWise Tutorial Device Fleet Model**。
5. 在 层次结构定义 下，执行以下操作：
   a. 在 层次结构名称 中，输入 **Device**。
   b. 在 层次结构模型 中，选择您的设备资产模型 (**SiteWise Tutorial Device Model**)。

   层次结构定义父（队列）资产模型和子（设备）资产模型之间的关系。父资产可以访问子资产的属性数据。以后创建资产时，需要根据父资产模型中的层次结构定义将子资产与父资产关联。有关更多信息，请参阅 定义资产模型层次结构。
6. 在 指标定义 下，执行以下操作：
   a. 在名称中，输入 **Average CPU Usage**。
   b. 在 公式 中，输入 **avg(Device | Average CPU Usage)**。当自动完成列表出现时，选择 Device选择层次结构，然后选择 Average CPU Usage从您之前创建的设备资产中选择指标。
   c. 在 时间间隔 中，输入 **5 minutes**。

   此指标属性通过 **Device** 层次结构计算与队列资产关联的所有设备资产的平均 CPU 利用率。
7. 选择添加指标以添加第二个指标属性。
8. 在 指标定义 下的第二行中，执行以下操作：
   a. 在名称中，输入 **Average Memory Usage**。
   b. 在 公式 中，输入 **avg(Device | Average Memory Usage)**。当自动完成列表出现时，选择 Device选择层次结构，然后选择 Average Memory Usage从您之前创建的设备资产中选择指标。
   c. 在 时间间隔 中，输入 **5 minutes**。

   此指标属性通过 **Device** 层次结构计算与队列资产关联的所有设备资产的平均内存使用量。
9. （可选）添加您有兴趣跨设备队列计算的其他指标。
10. 选择创建模型。

步骤 5：创建和配置设备资产

在此过程中，您将根据您的设备资产模型生成设备资产。然后，为每个测量属性定义属性别名。属性别名是唯一标识资产属性的字符串。之后，您可以使用别名而不是资产 ID 和属性 ID 来识别要上传数据的属性。有关更多信息，请参阅 管理数据流 Amazon IoT SiteWise。

创建设备资产并定义属性别名

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择 资产。
3. 选择 创建资产。
4. 在模型信息下，选择您的设备资产模型，**SiteWise Tutorial Device Model**。
5. 在资产信息下，输入资产的名称。例如，**SiteWise Tutorial Device 1**。
6. 选择创建资产。
7. 对于新设备资产，请选择 编辑。
8. 在 下CPU Usage，**/tutorial/device/SiteWiseTutorialDevice1/cpu**作为属性别名输入。您可以在属性别名中包含 Amazon IoT 事物的名称，这样您就可以使用一条 Amazon IoT 规则从所有设备中提取数据。
9. 在 下Memory Usage，**/tutorial/device/SiteWiseTutorialDevice1/memory**作为属性别名输入。
10. 选择保存。

如果您之前创建了多个 Amazon IoT 内容，请对每台设备重复步骤 3 到 10，并相应地增加资产名称和属性别名中的数字。例如，第二个设备资产的名称应为 **SiteWise Tutorial Device 2**，其属性别名应为 **/tutorial/device/SiteWiseTutorialDevice2/cpu** 和 **/tutorial/device/SiteWiseTutorialDevice2/memory**。

步骤 6：创建和配置设备队列资产

在此过程中，您基于设备队列资产模型构建设备队列资产。然后，您将单个设备资产关联到队列资产。通过这种关联，队列资产的指标属性能够编译和分析来自多个设备的数据。这些数据为您提供了整个队列整体表现的综合视图。

创建设备队列资产并关联设备资产

1. 导航至 Amazon IoT SiteWise 控制台。

2. 在左侧导航窗格中，选择 资产。
3. 选择 创建资产。
4. 在模型信息中，选择您的设备实例集资产模型 **SiteWise Tutorial Device Fleet Model**。
5. 在资产信息下，输入资产的名称。例如，**SiteWise Tutorial Device Fleet 1**。
6. 选择创建资产。
7. 对于新设备队列资产，请选择 编辑。
8. 在关联到此资产的资产下，选择添加关联资产，然后进行以下操作：
   a. 在 “层次结构” 下，选择 Device。 该层次结构确定了设备和设备队列资产之间的层次关系。您在本教程前面的设备队列资产模型中定义了此层次结构。
   b. 在 “资产” 下，选择您的设备资产，SiteWise Tutorial Device 1 。
9. （可选）如果您之前创建了多个设备资产，请对您创建的每个设备资产重复步骤 8 到 10。
10. 选择保存。

    现在，您应该可以看到设备资产按层次结构进行组织。

步骤 7：在 Amazon IoT Core 中创建向设备资产发送数据的规则

在此过程中，您需要在 Amazon IoT Core中建立规则。该规则旨在解释来自设备影子的通知消息，并将数据传输到您的设备资产 Amazon IoT SiteWise。每次设备影子更新时，都会 Amazon IoT 发送一条 MQTT 消息。您可以创建一个规则，以便在设备影子基于 MQTT 消息发生变化时执行操作。在这种情况下，目标是处理更新消息，提取属性值，然后将其传输到您在 Amazon IoT SiteWise中的设备资产。

使用 Amazon IoT SiteWise 操作创建规则

1. 导航至 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择消息路由，然后选择规则。
3. 选择 创建规则。
4. 输入规则的名称和说明，然后选择下一步。
5. 输入以下 SQL 语句并选择下一步。

```
SELECT
  *
FROM
  '$aws/things/+/shadow/update/accepted'
WHERE
```

```
startsWith(topic(3), "SiteWiseTutorialDevice")
```

此规则查询语句起作用，因为 Device Shadow 服务将影子更新发布到 `$aws/things/thingName/shadow/update/accepted`。有关设备影子的更多信息，请参阅 Amazon IoT 开发人员指南中的设备影子服务。

在 `WHERE` 子句中，此规则查询语句使用 `topic(3)` 函数从主题的第三段获取事物名称。然后，该语句筛选掉名称与教程设备名称不匹配的设备。有关 Amazon IoT SQL 的更多信息，请参阅《Amazon IoT 开发人员指南》中的 Amazon IoT SQL 参考。

6. 在规则操作下，选择向 Amazon IoT SiteWise中的资产属性发送消息数据，然后执行以下操作：

   a. 选择 按属性别名。

   b. 在 属性别名 中，输入 **`/tutorial/device/${topic(3)}/cpu`**。

      `${...}`语法是一个替换模板。 Amazon IoT 评估大括号内的内容。此替换模板从主题中提取事物名称，以创建对每个事物唯一的别名。有关更多信息，请参阅 Amazon IoT 开发人员指南中的替换模板。

      > **Note**
      > 由于替换模板中的表达式与 `SELECT` 语句分开评估，因此您不能使用替换模板来引用使用 `AS` 子句创建的别名。除了支持的函数和运算符之外，您只能引用原始负载中存在的信息。

   c. 在条目 ID -可选中，输入 **`${concat(topic(3), "-cpu-", floor(state.reported.timestamp))}`**。

      条目 IDs 唯一标识每次输入值的尝试。如果条目返回错误，您可以在错误输出中找到条目 ID 以对问题进行故障排除。此条目 ID 中的替换模板将事物名称与设备报告的时间戳组合起来。例如，生成的条目 ID 可能看起来像 `SiteWiseTutorialDevice1-cpu-1579808494`。

   d. 在时间（以秒为单位）中，输入 **`${floor(state.reported.timestamp)}`**。

      此替换模板从设备报告的时间戳计算时间（以秒为单位）。在本教程中，设备将 Unix 纪元时间中的时间戳（以秒为单位）报告为浮点数。

   e. 在偏移量（以纳秒为单位）- 可选中，输入 **`${floor((state.reported.timestamp % 1) * 1E9)}`**。

      此替换模板通过转换设备报告时间戳的小数部分来计算时间（以秒为单位）的纳秒偏移量。

> **Note**
>
> Amazon IoT SiteWise 要求您的数据具有以 Unix 纪元时间为单位的当前时间戳。如果您的设备没有准确报告时间，您可以使用 timestamp() 从 Amazon IoT 规则引擎获取当前时间。此函数以毫秒为单位报告时间，因此您必须将规则操作的时间参数更新为以下值：
>
> - 在时间（以秒为单位）中，输入 **`${floor(timestamp() / 1E3)}`**。
> - 在 偏移量 (以纳秒为单位) 中，输入 **`${(timestamp() % 1E3) * 1E6}`**。

f. 在 数据类型 中，选择 双精度。

   此数据类型必须与您在资产模型中定义的资产属性的数据类型相匹配。

g. 在值中，输入 **`${state.reported.cpu}`**。在替换模板中，您可以使用 **`.`** 运算符从 JSON 结构中检索值。

h. 选择 添加条目 为内存使用量属性添加新条目，然后再次为该属性完成以下步骤：

   i. 选择 按属性别名。

   ii. 在 属性别名 中，输入 **`/tutorial/device/${topic(3)}/memory`**。

   iii. 在条目 ID -可选中，输入 **`${concat(topic(3), "-memory-", floor(state.reported.timestamp))}`**。

   iv. 在时间（以秒为单位）中，输入 **`${floor(state.reported.timestamp)}`**。

   v. 在偏移量（以纳秒为单位）- 可选中，输入 **`${floor((state.reported.timestamp % 1) * 1E9)}`**。

   vi. 在 数据类型 中，选择 双精度。

   vii. 在值中，输入 **`${state.reported.memory}`**。

i. 在 IAM 角色下，选择创建新角色，以便为此规则操作创建 IAM 角色。此角色允许将数据推送 Amazon IoT 到设备队列资产及其资产层次结构中的属性。

j. 输入角色名称，然后选择创建。

7. （可选）配置可用于对规则进行故障排除的错误操作。有关更多信息，请参阅 排查规则问题（Amazon IoT SiteWise）。

8. 选择 下一步。

9. 查看设置并选择创建以创建规则。

步骤 8：运行设备客户端脚本

在本教程中，您不会使用实际设备来报告数据，取而代之的是，你运行一个脚本来使用 CPU 和内存使用情况更新你的 Amazon IoT 设备影子，以模仿真实的传感器数据。要运行脚本，必须先进行安装 Python 包裹。在此过程中，您需要安装所需的 Python 打包，然后运行设备客户端脚本。

配置和运行设备客户端脚本

1. 导航至 Amazon IoT 控制台。
2. 在左侧导航窗格底部，选择 设置。
3. 保存您的自定义终端节点以便与设备客户端脚本一起使用。您可以使用此终端节点与事物影子进行交互。此终端节点对您在当前区域中的账户是唯一的。

   您的自定义终端节点应与以下示例类似。

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

4. 打开命令行并运行以下命令以导航到之前创建的教程目录。

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

5. 运行以下命令来安装 Amazon IoT Device SDK for Python。

   ```
   pip3 install AWSIoTPythonSDK
   ```

   有关更多信息，请参阅 Amazon IoT 开发人员指南中的 Amazon IoT Device SDK for Python。
6. 运行以下命令来安装 psutil，这是一个跨平台的进程和系统实用程序库。

   ```
   pip3 install psutil
   ```

   有关更多信息，请参阅 Python Package 索引中的 psutil。
7. 在 `iot-sitewise-rule-tutorial` 目录中创建一个名为 `thing_performance.py` 的文件，然后将以下 Python 代码复制到该文件中。

   ```
   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 {}...
\n".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. 在命令行中运行 `thing_performance.py`及以下参数：

- `-n, --thingName` - 您的事物名称，例如 **`SiteWiseTutorialDevice1`**。
- `-e`，`--endpoint`— 您在本过程前面保存的自定义 Amazon IoT 终端节点。
- `-r`，`--rootCA`— Amazon IoT 根 CA 证书的路径。
- `-c`，`--cert`— 你的 Amazon IoT 事物证书的路径。
- `-k`，`--key`— 你的 Amazon IoT 东西证书私钥的路径。
- `-d, --requestDelay` -（可选）每个设备影子更新之间等待的时间（以秒为单位）。默认值为 1 秒。
- `-v, --enableLogging` -（可选）如果存在此参数，则脚本将从 Amazon IoT Device SDK for Python打印调试消息。

您的命令应类似于以下示例。

```
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
```

如果您正在为其他 Amazon IoT 内容运行脚本，请相应地更新事物名称和证书目录。

9. 尝试打开和关闭设备上的程序，了解 CPU 利用率和内存使用量如何变化。脚本会打印每个 CPU 利用率和内存使用量读数。如果脚本成功将数据上传到 Device Shadow 服务，则脚本的输出应与以下示例类似。

```
[SiteWiseTutorialDevice1]
CPU:    24.6%
Memory: 85.2%

[SiteWiseTutorialDevice1]
Update request e6686e44-fca0-44db-aa48-3ca81726f3e3 accepted
```

10. 请按照以下步骤操作，验证脚本是否正在更新设备影子：

    a. 导航至 Amazon IoT 控制台。

    b. 在左侧导航窗格中选择所有设备，然后选择事物。

    c. 选择你的东西，SiteWiseTutorialDevice。

d. 选择设备影子选项卡，选择经典影子，然后核实影子状态是否与以下示例类似。

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

如果你的东西的影子状态为空或者看起来不像前面的示例，请检查脚本是否正在运行并成功连接 Amazon IoT。如果脚本在连接时继续超时 Amazon IoT，请检查您的事物策略是否已根据本教程进行配置。

11. 请按照以下步骤操作，验证规则操作是否正在将数据发送到 Amazon IoT SiteWise：

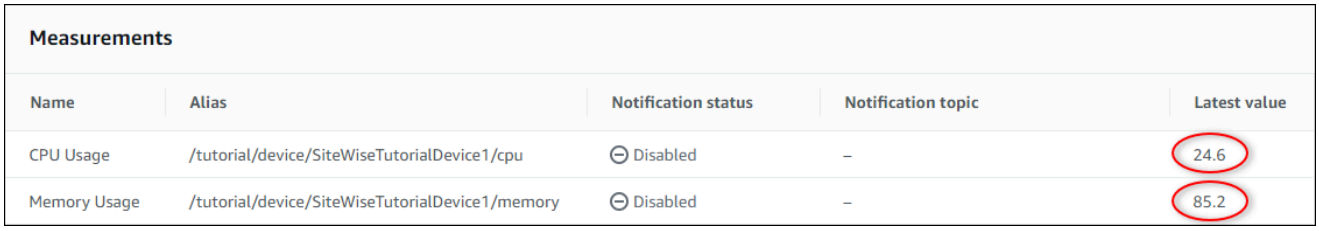
a. 导航至 Amazon IoT SiteWise 控制台。

b. 在左侧导航窗格中，选择 资产。

c. 选择您的设备队列资产旁边的箭头 (SiteWise Tutorial Device Fleet 1 1) 扩展其资产层次结构，然后选择您的设备资产 (SiteWise Tutorial Device 1)。

d. 选择 测量值。

e. 验证 “最新” 值单元格是否包含以下值 CPU Usage 和 Memory Usage属性。



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

f. 如果 CPU Usage 和 Memory Usage属性没有最新值，请刷新页面。如果在几分钟后没有显示值，请参阅排查规则问题（Amazon IoT SiteWise）。

您已完成本教程。如果要探索数据的实时可视化效果，可以在 Amazon IoT SiteWise Monitor中配置门户。有关更多信息，请参阅 使用监控数据 Amazon IoT SiteWise Monitor。否则，您可以在命令提示符下按 Ctrl+C 停止设备客户端脚本。该 Python 程序发送的消息应该不足以产生费用，但完成后最好将其停止。

步骤 9：在完成教程后清理资源

完成有关从 Amazon IoT 事物中提取数据的教程后，请清理资源以免产生额外费用。

要删除中的分层资产 Amazon IoT SiteWise

1. 导航到 Amazon IoT SiteWise 控制台
2. 在左侧导航窗格中，选择 资产。
3. 在中删除资源时 Amazon IoT SiteWise，必须先取消与它们的关联。

   请完成以下步骤以取消设备资产与设备队列资产之间的关联：

   a. 选择您的设备队列资产 (SiteWise Tutorial Device Fleet 1).
   b. 选择编辑。
   c. 在 与此资产关联的资产 下，针对与此设备队列资产关联的每个设备资产，选择 取消关联。
   d. 选择保存。

      现在，您应该可以看到设备资产不再按层次结构进行组织。
4. 选择您的设备资产 (SiteWise Tutorial Device 1)。
5. 选择删除。
6. 在确认对话框中，输入 **Delete**，然后选择 删除。
7. 对每项设备资产和设备队列资产重复步骤 4 到 6（SiteWise Tutorial Device Fleet 1).

要在中删除分层资产模型 Amazon IoT SiteWise

1. 导航至 Amazon IoT SiteWise 控制台。
2. 如果还没有删除，请删除设备和设备队列资产。有关更多信息，请参阅前一过程。如果您拥有从某个模型创建的资产，则无法删除该模型。
3. 在左侧导航窗格中，选择 模型。
4. 选择您的设备队列资产模型 (SiteWise Tutorial Device Fleet Model).

   删除分层资产模型时，首先删除父资产模型。
5. 选择删除。
6. 在确认对话框中，输入 **Delete**，然后选择 删除。
7. 对您的设备资产模型重复步骤 4 到 6 (SiteWise Tutorial Device Model).

禁用或删除中的规则 Amazon IoT Core

1. 导航至 Amazon IoT 控制台。

2. 在左侧导航窗格中，选择消息路由，然后选择规则。
3. 选择您的规则，然后选择删除。
4. 在确认对话框中，键入规则名称，然后选择删除。

在 Monito SiteWise r 中可视化和共享风电场数据

本教程介绍 Amazon IoT SiteWise Monitor 如何使用托管 Web 应用程序（称为门户）来可视化和共享工业数据。每个门户都包含项目，方便您灵活地选择可以访问每个项目中的哪些数据。然后，在组织中指定能够访问各个门户的人员。您的用户使用 Amazon IAM Identity Center 账户登录门户，这样您就可以使用现有的身份存储或 Amazon托管的存储。

> **Important**
>
> 本教程介绍如何使用 IAM Identity Center 对门户用户进行身份验证。IAM Identity Center 在中国地区不可用。相反， SiteWise 监控门户使用 Amazon Identity and Access Management (IAM) 对用户进行身份验证。您无法按照本教程中与 IAM Identity Center 交互的步骤进行操作，但可以使用 IAM 身份完成本教程。

您以及具有足够权限的用户可以在各个项目中创建控制面板，以有意义的方式可视化工业数据。然后，用户可以查看这些控制面板，以快速了解数据并监控操作。您可以为公司中的每个用户，针对每个项目配置管理权限或只读权限。有关更多信息，请参阅 使用监控数据 Amazon IoT SiteWise Monitor。

在整个教程中，您将增强 Amazon IoT SiteWise 演示，提供风力发电场的示例数据集。您可以在 Monito SiteWise r 中配置门户、创建项目和仪表板来可视化风电场数据。本教程还包括创建其他用户，以及分配拥有或查看项目及其相关控制面板的权限。

> **Note**
>
> 使用 M SiteWise onitor 时，按每位登录门户的用户付费（每月）。在本教程中，您将创建三个用户，但只需使用一个用户登录。完成本教程后，这一个用户将产生费用。有关更多信息，请参阅Amazon IoT SiteWise 定价。

主题

- 先决条件
- 步骤 1：在 Monito SiteWise r 中创建入口

- 步骤 2：登录门户
- 步骤 3：创建风电场项目
- 步骤 4：创建控制面板以可视化风电场数据
- 步骤 5：浏览门户
- 步骤 6：在完成教程后清理资源

先决条件

要完成本教程，您需要：

- 一个 Amazon Web Services 账户。如果没有，请参阅设置一个 Amazon 账户。
- 一台正在运行的开发计算机 Windows, macOS, Linux，或 Unix 访问 Amazon Web Services Management Console。有关更多信息，请参阅什么是 Amazon Web Services Management Console？。
- 具有管理员权限的 Amazon Identity and Access Management (IAM) 用户。
- 正在运行的 Amazon IoT SiteWise 风力发电场演示。当你设置演示时，它会定义模型和资产，Amazon IoT SiteWise 并将数据流式传输到它们以表示风力发电场。有关更多信息，请参阅 使用演 Amazon IoT SiteWise 示。
- 如果您在账户中启用了 IAM Identity Center，请登录您的 Amazon Organizations 管理账户。有关更多信息，请参阅 Amazon Organizations 术语和概念。如果您尚未启用 IAM Identity Center，则需要在本教程中将其启用，并将您的帐户设置为管理帐户。

  如果您无法登录 Amazon Organizations 管理账户，则只要您的组织中有 IAM Identity Center 用户，就可以部分完成本教程。在这种情况下，您可以创建门户和控制面板，但无法创建新的 IAM Identity Center 用户来分配项目。

步骤 1：在 Monito SiteWise r 中创建入口

在此过程中，您在 Amazon IoT SiteWise Monitor中创建门户。每个门户都是一个托管 Web 应用程序，您和您的用户可以使用 Amazon IAM Identity Center 账户登录该应用程序。IAM Identity Center 支持使用贵公司的现有身份存储或创建由 Amazon托管的身份存储。贵公司的员工无需创建单独的 Amazon Web Services 账户即可登录。

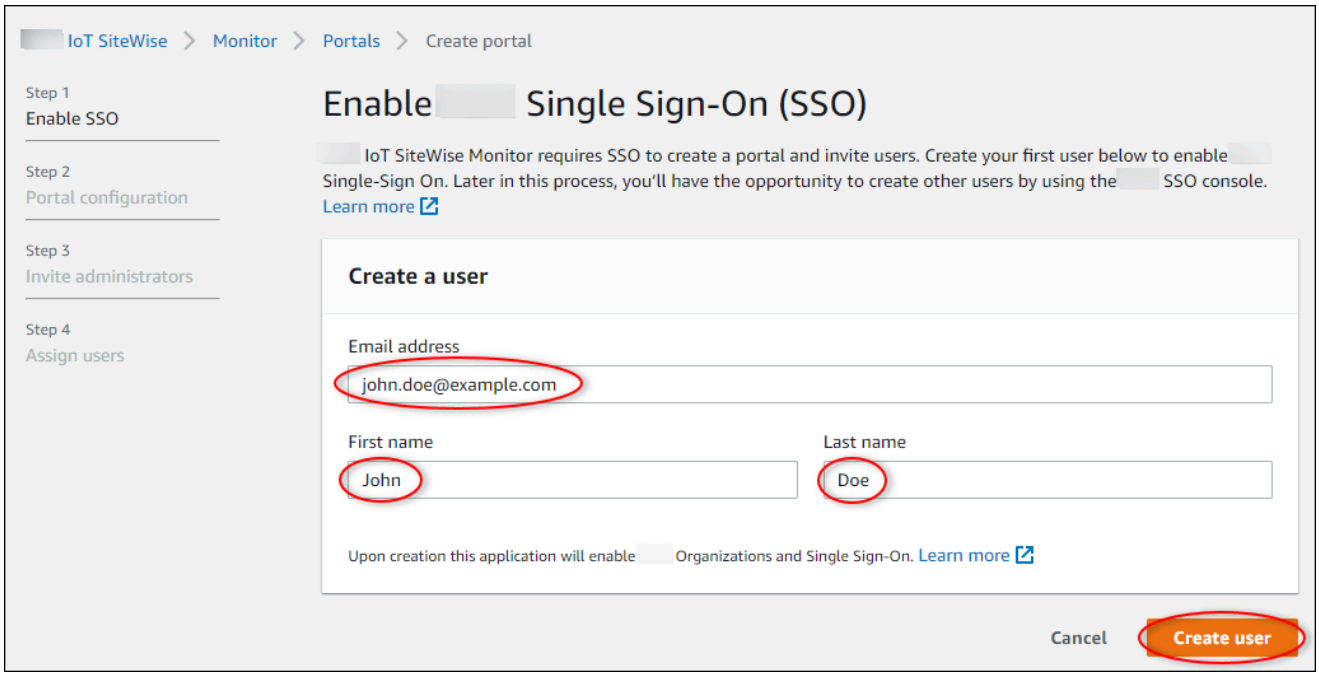
创建门户

1. 登录 Amazon IoT SiteWise 控制台。

2. 查看支持的Amazon IoT SiteWise 终端节点和配额，并在需要时切换区域。 Amazon IoT SiteWise 您必须在同一地区运行 Amazon IoT SiteWise 演示。
3. 在左侧导航窗格中，选择 门户。
4. 选择 创建门户。
5. 如果您已经启用了 IAM Identity Center ，请跳至步骤 6。否则，请完成以下步骤以启用 IAM Identity Center。

   a. 在启用 Amazon IAM Identity Center (SSO) 页面上，输入您的电子邮件地址、名字和姓氏，为自己创建一个 IAM Identity Center 用户，使其成为门户管理员。请使用您可以访问的电子邮件地址，这样您会收到一封为新 IAM Identity Center 用户设置密码的电子邮件。

      在门户中，门户管理员会创建项目并为项目分配用户。之后您可以创建更多用户。



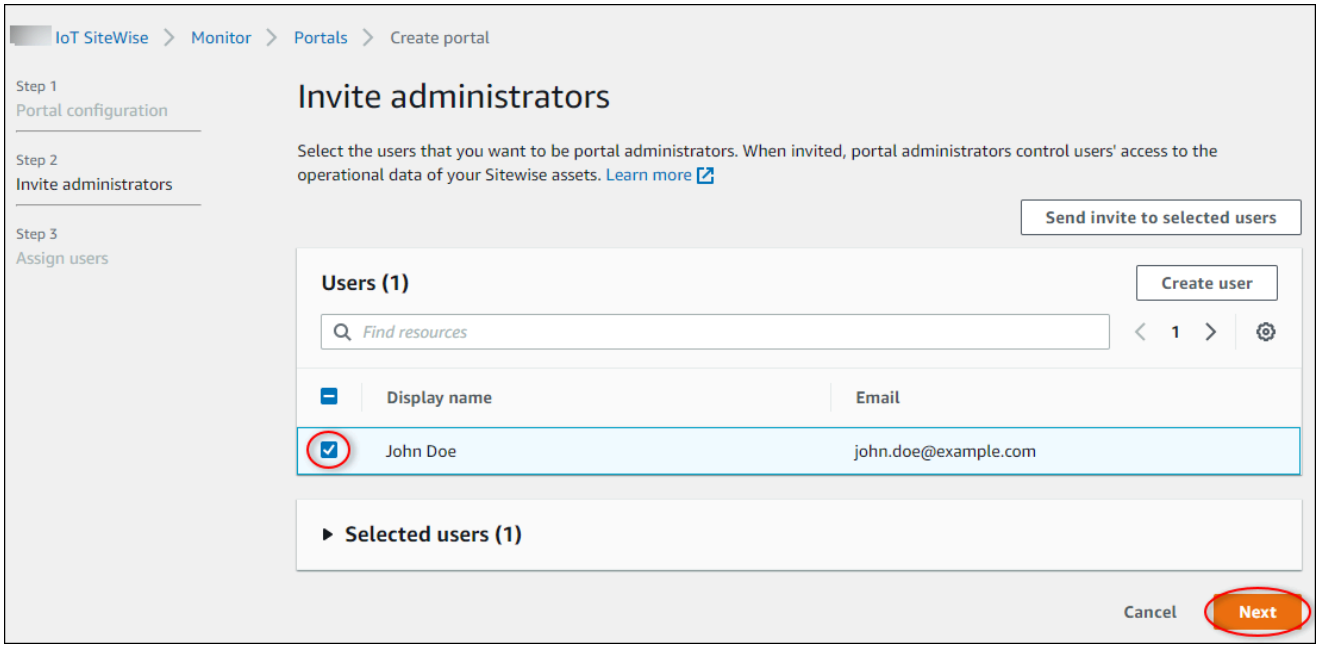
   b. 选择 创建用户。
6. 在 门户配置 页面上，完成以下步骤：

   a. 输入门户的名称，例如 **`WindFarmPortal`**。
   b. （可选）输入门户的描述。如果您有多个门户，请使用有意义的描述来跟踪每个门户包含的内容。
   c. （可选）上传图像以在门户中显示。
   d. 输入一个电子邮件地址，当门户用户遇到门户问题并需要贵公司 Amazon 管理员的帮助才能解决问题时，可以联系到该地址。

e. 选择 创建门户。

7. 在邀请管理员页面上，您可以将 IAM Identity Center 用户作为管理员分配到门户。门户管理员管理门户内的权限和项目。在此页上，执行以下操作：

   a. 选择作为门户管理员的用户。如果您在本教程前面部分启用了 IAM Identity Center，请选择您创建的用户。



   b. （可选）选择 向选定用户发送邀请。您的电子邮件客户端将打开，邮件正文中会显示一个邀请。在将电子邮件发送给门户管理员之前您可以对其进行自定义。您也可以稍后再向门户管理员发送电子邮件。如果您是首次尝试使用 M SiteWise onitor，并且将成为门户管理员，则无需给自己发送电子邮件。

   c. 选择下一步。

8. 在分配用户页面上，您可以将 IAM Identity Center 用户分配给门户。门户管理员稍后可以将这些用户分配为项目所有者或查看者。项目所有者可以在项目中创建控制面板。项目查看者对分配给他们的项目具有只读访问权限。在此页面上，您可以创建要添加到门户的 IAM Identity Center 用户。

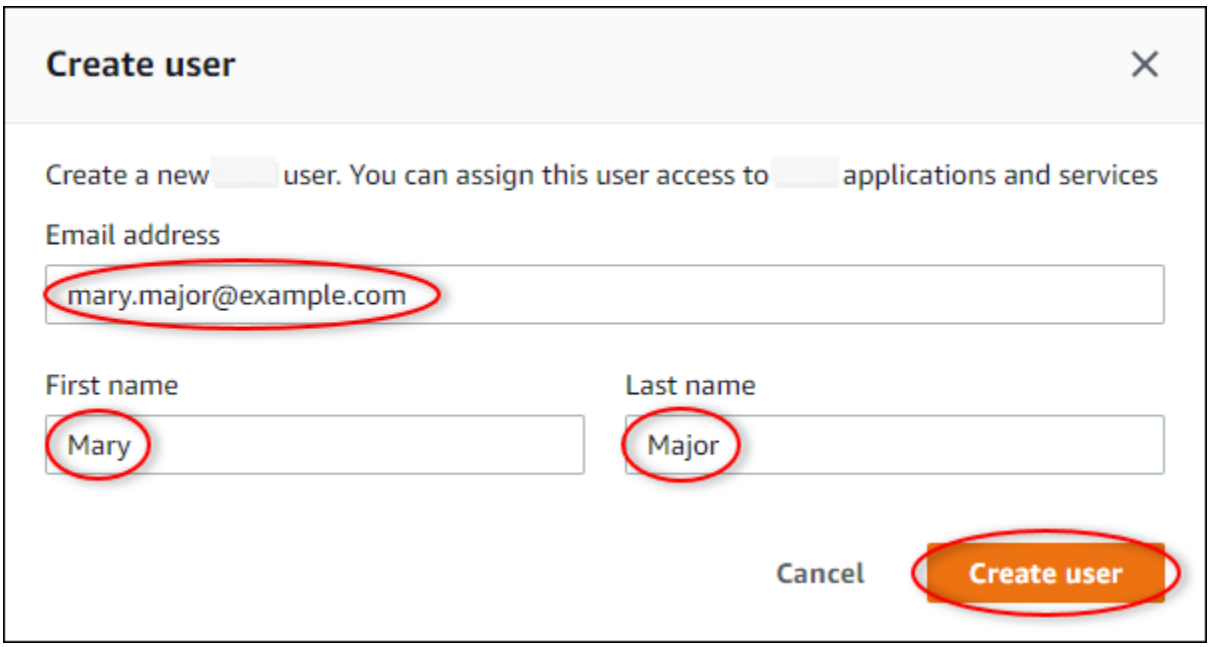
> **Note**
>
> 如果您未登录 Amazon Organizations 管理账户，则无法创建 IAM Identity Center 用户。选择分配用户以创建没有门户用户的门户，然后跳过此步骤。

在此页上，执行以下操作：

a. 完成以下步骤两次即可创建两个 IAM Identity Center 用户：

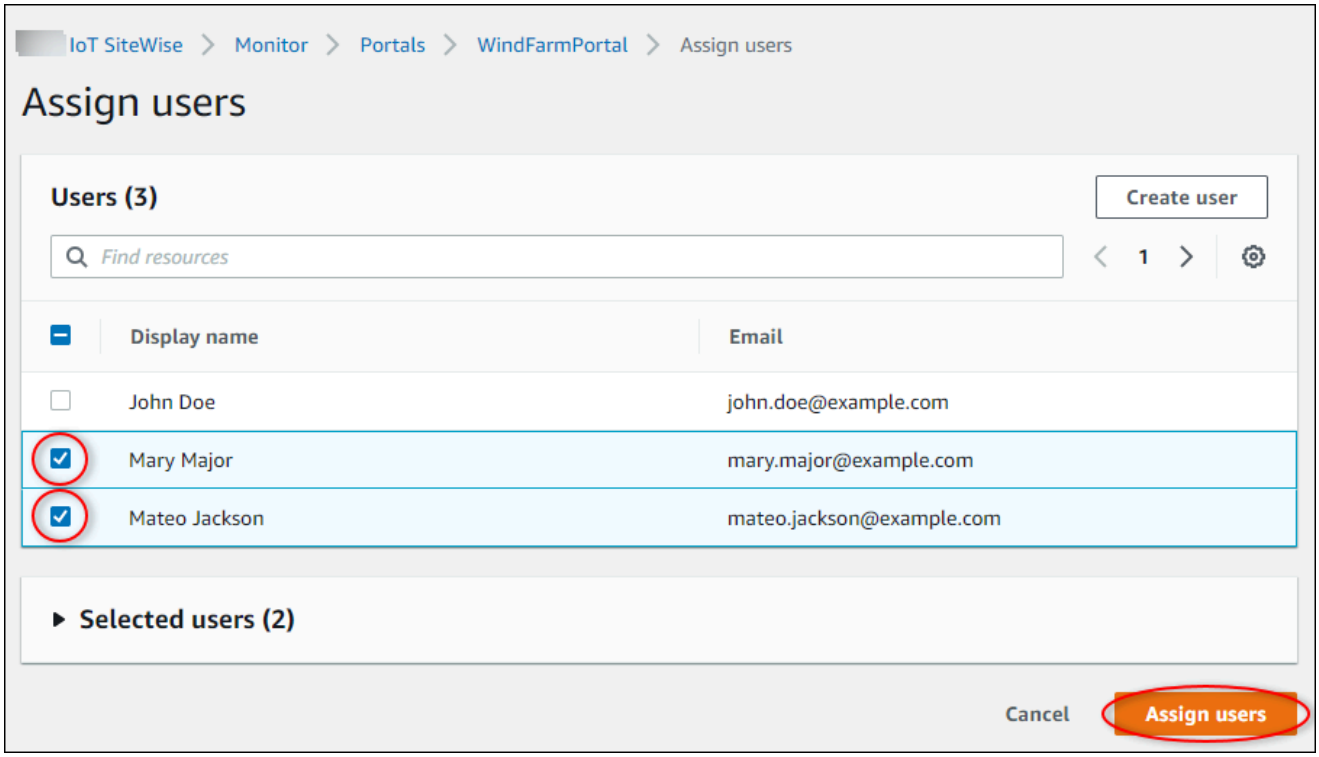
   i. 选择创建用户以打开一个对话框，您可以在其中输入新用户的详细信息。

   ii. 输入要注册的用户的名字、姓氏和电子邮件地址。IAM Identity Center 会向用户发送一封电子邮件，让他们设置密码。如果您想以这些用户的身份登录门户，请选择您可以访问的电子邮件地址。每个电子邮件地址必须唯一。您的用户使用其电子邮件地址作为用户名登录门户。



   iii. 选择创建用户。

b. 选择您在上一步中创建的两个 IAM Identity Center 用户。



c. 选择分配用户以将这些用户添加到门户。

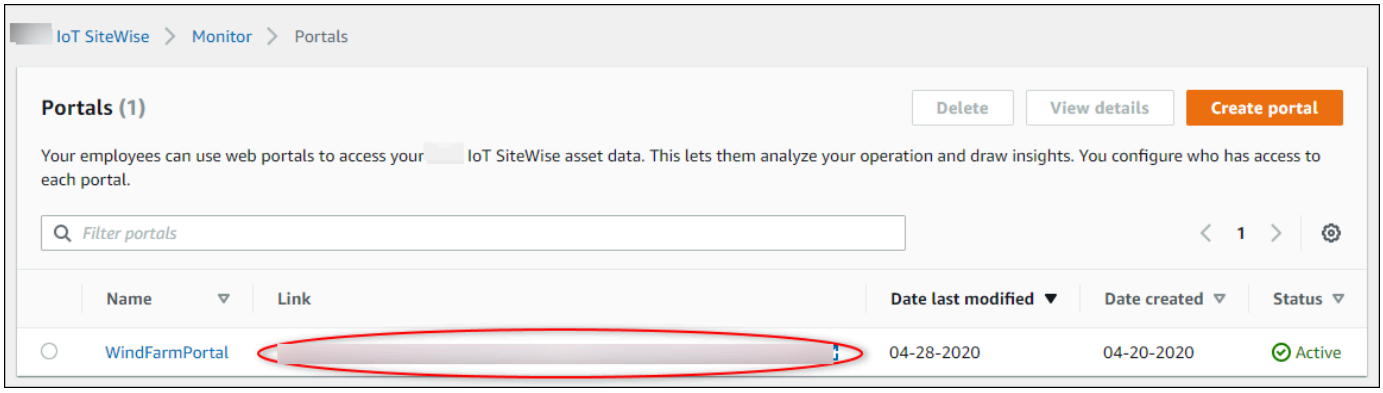
此时将打开门户页面，其中列出了您的新门户。

步骤 2：登录门户

在此过程中，您将使用添加到新门户的 Amazon IAM Identity Center 用户登录新门户。

登录门户

1. 在 门户 页面上，选择新门户的 链接 以在新选项卡中打开门户。



2. 如果您在本教程前面部分中创建了第一个 IAM Identity Center 用户，请使用以下步骤为该用户创建密码：

   a. 请查看您的电子邮件中的主题行 Invitation to join Amazon IAM Identity Center.

   b. 打开该邀请电子邮件并选择 Accept invitation.

   c. 在新窗口中，为 IAM Identity Center 用户设置密码。

   如果您稍后想以之前创建的第二和第三个 IAM Identity Center 用户身份登录门户，也可以完成这些步骤，为这些用户设置密码。

   > **Note**
   >
   > 如果没有收到电子邮件，您可以在 IAM Identity Center 控制台中为用户生成密码。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的重置最终用户的 IAM Identiy Center 用户密码。

3. 进入您的 IAM 身份中心 Username 和 Password。 如果您在本教程的前面部分创建了 IAM 身份中心用户，则您的 Username是您创建的门户管理员用户的电子邮件地址。

   所有门户用户（包括门户管理员）都必须使用其 IAM Identity Center 用户凭证登录。这些凭据通常不是用于登录 Amazon Web Services Management Console的那些凭据。

4. 选择 Sign in.

   您的门户打开。

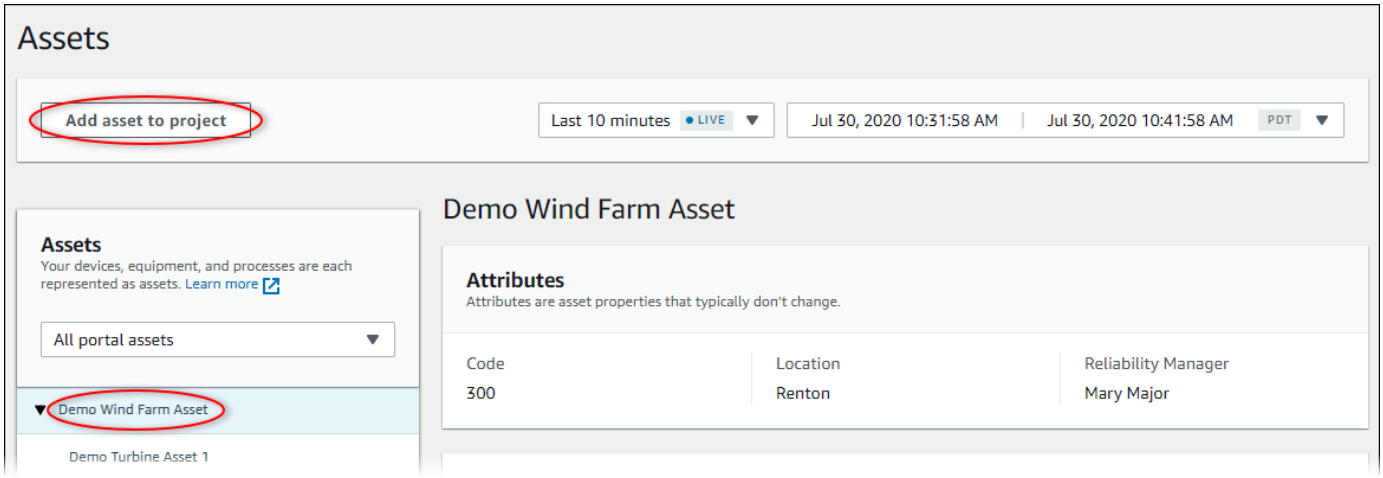
步骤 3：创建风电场项目

在此过程中，您将在门户中创建项目。项目是定义一组权限、资产和控制面板的资源，您可以对其进行配置，以在项目中可视化资产数据。通过项目，您可以定义谁有权访问操作的哪些子集，以及如何可视化这些子集的数据。您可以将门户用户分配作为各个项目的拥有者或查看者。项目所有者可以通过创建控制面板来可视化数据，并与其他用户共享项目。项目查看者可以查看控制面板，但不能进行编辑。有关 SiteWise 监控器中角色的更多信息，请参阅SiteWise 监控角色。

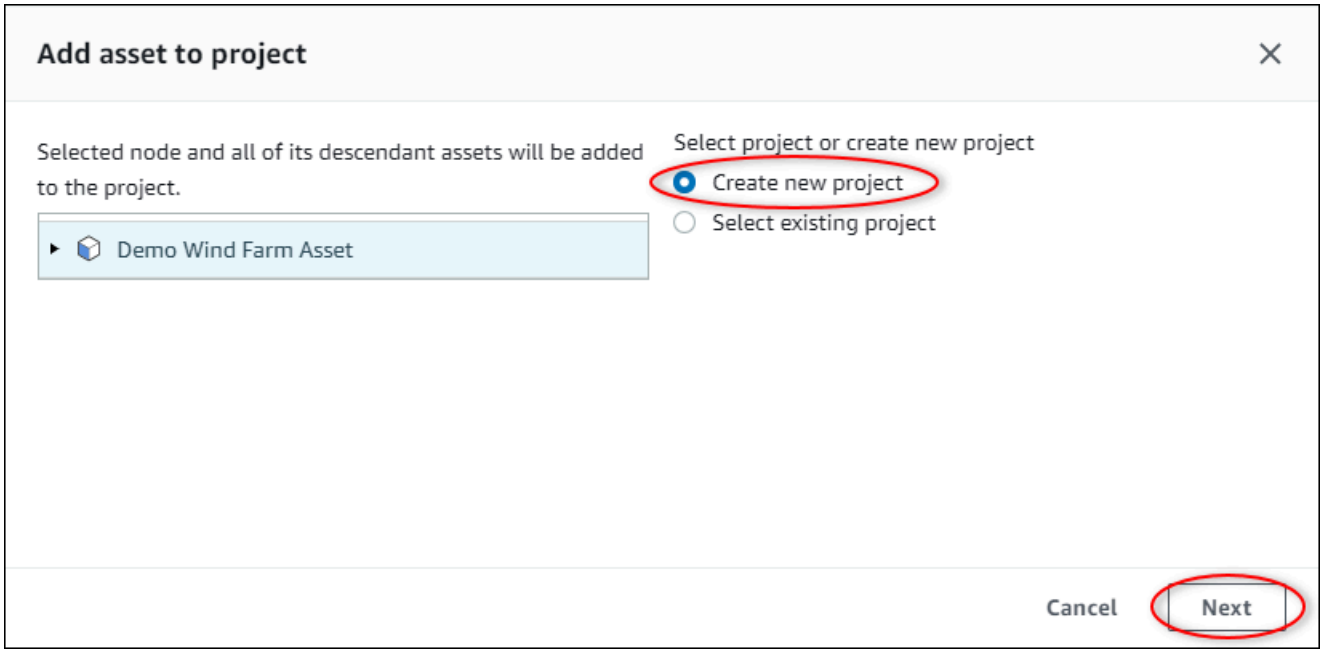
创建风电场项目

1. 在门户左侧的导航窗格中，选择资产选项卡。在资产页面，您可以浏览门户中的所有可用资产并将资产添加到项目中。
2. 在资源浏览器中，选择 Demo Wind Farm Asset。 选择资产时，您可以浏览该资产的实时数据和历史数据。您也可以按Shift下选择多个资源并比较其数据 side-by-side。

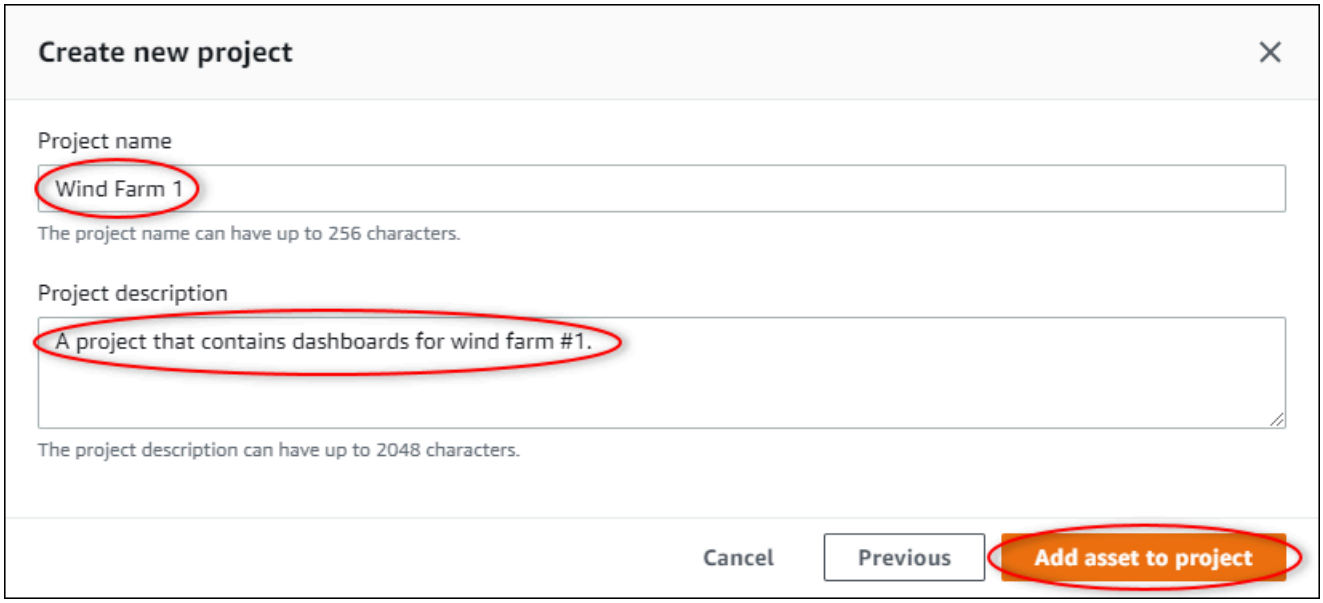
3. 选择左上角的向项目添加资产。项目包含控制面板，门户用户可以在其中进行查看来浏览数据。每个项目都可以访问您的资源子集 Amazon IoT SiteWise。将资产添加到项目时，对该项目具有访问权限的所有用户也可以访问该资产及其子集的数据。



4. 在向项目添加资产对话框中，选择创建新项目，然后选择下一步。



5. 在创建新项目对话框中，为项目输入项目名称和项目描述，然后选择向项目添加资产。



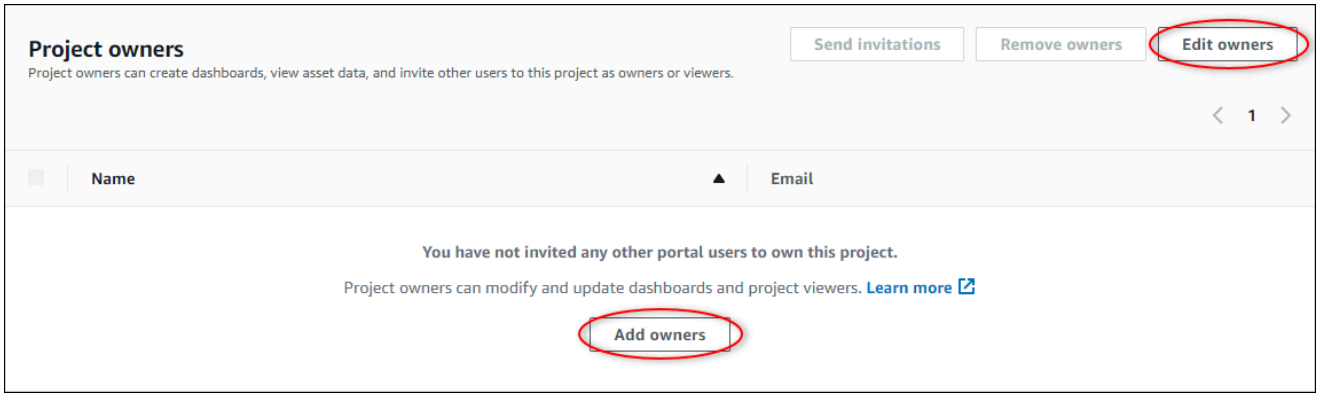
此时将打开新项目的页面。

6. 在项目页面上，您可以将门户用户添加为该项目的所有者或查看者。

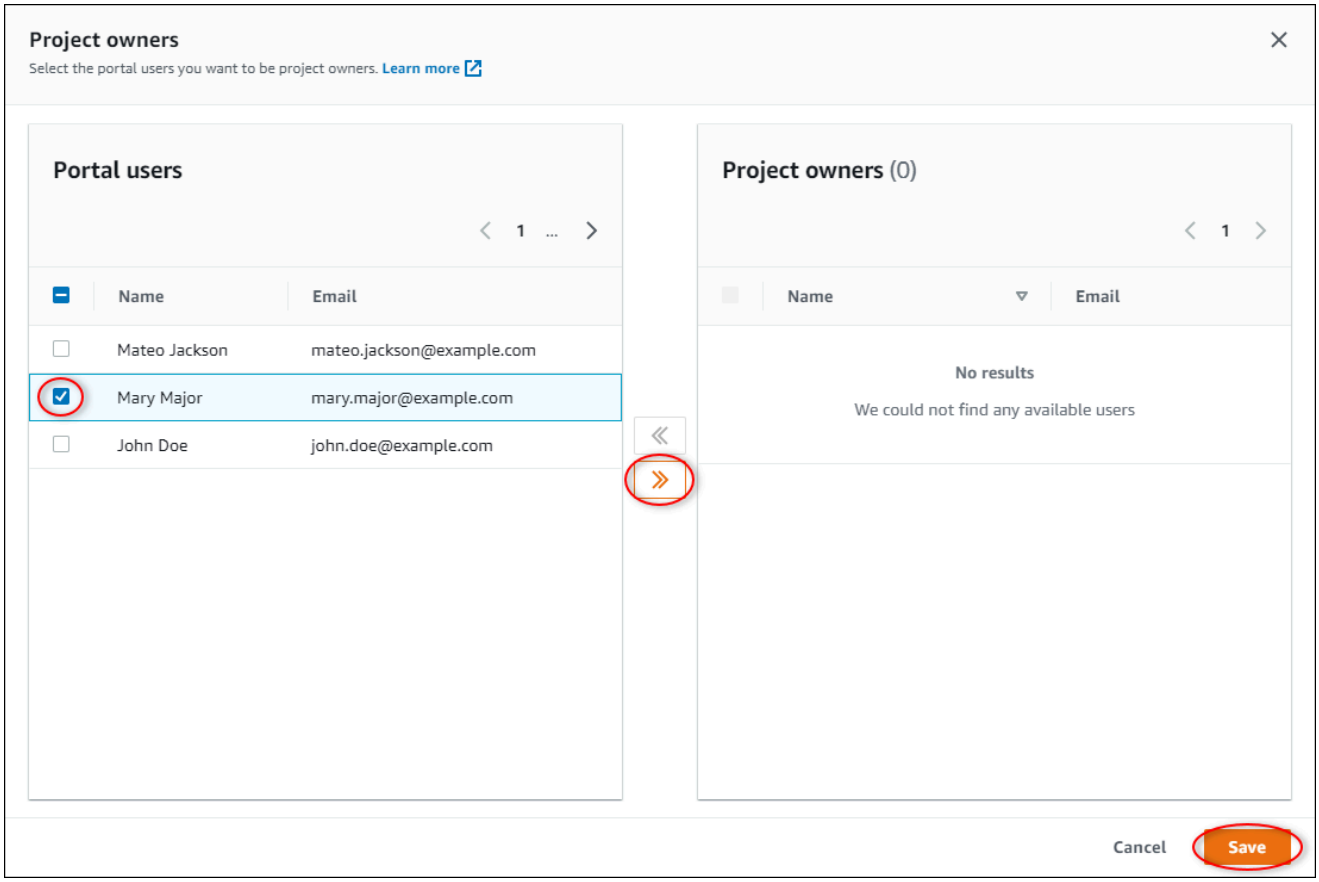
> **Note**
>
> 如果您尚未登录 Amazon Organizations 管理帐户，则可能没有门户用户可以分配给此项目，因此您可以跳过此步骤。

在此页上，执行以下操作：

a. 在项目所有者下，选择添加所有者或编辑用户。



b. 选择要添加为项目所有者的用户（例如 Mary Major），然后选择 >> 图标。



c. 选择保存。

   您的 IAM 身份中心用户 Mary Major可以登录此门户编辑此项目中的仪表板，并与该门户中的其他用户共享此项目。

d. 在项目查看者下，选择添加查看者或编辑用户。

e. 选择要添加为项目查看者的用户（例如 Mateo Jackson)，然后选择 >> 图标。

f. 选择保存。

   您的 IAM 身份中心用户 Mateo Jackson可以登录此门户网站查看但不能编辑风电场项目中的仪表板。

步骤 4：创建控制面板以可视化风电场数据

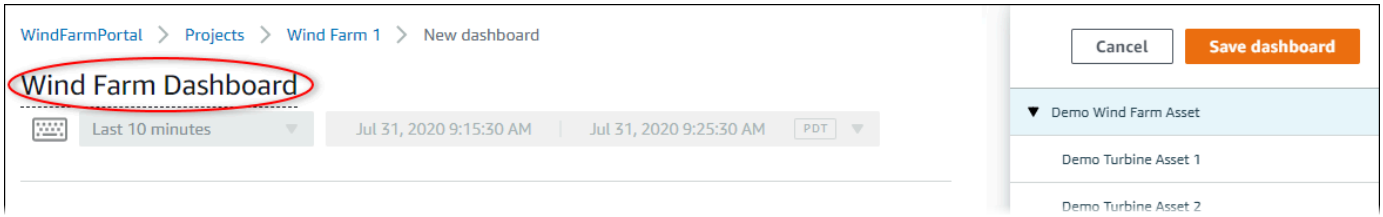
在此过程中，您将创建控制面板以可视化演示风电场数据。控制面板包含项目资产数据可自定义的可视化。每个可视化可以具有不同的类型，例如折线图、条形图或关键绩效指标（KPI）显示。您可以选择最适合您数据的可视化类型。项目拥有者可以编辑控制面板，而项目查看者只能通过查看控制面板来获得见解。

创建具有可视化的控制面板

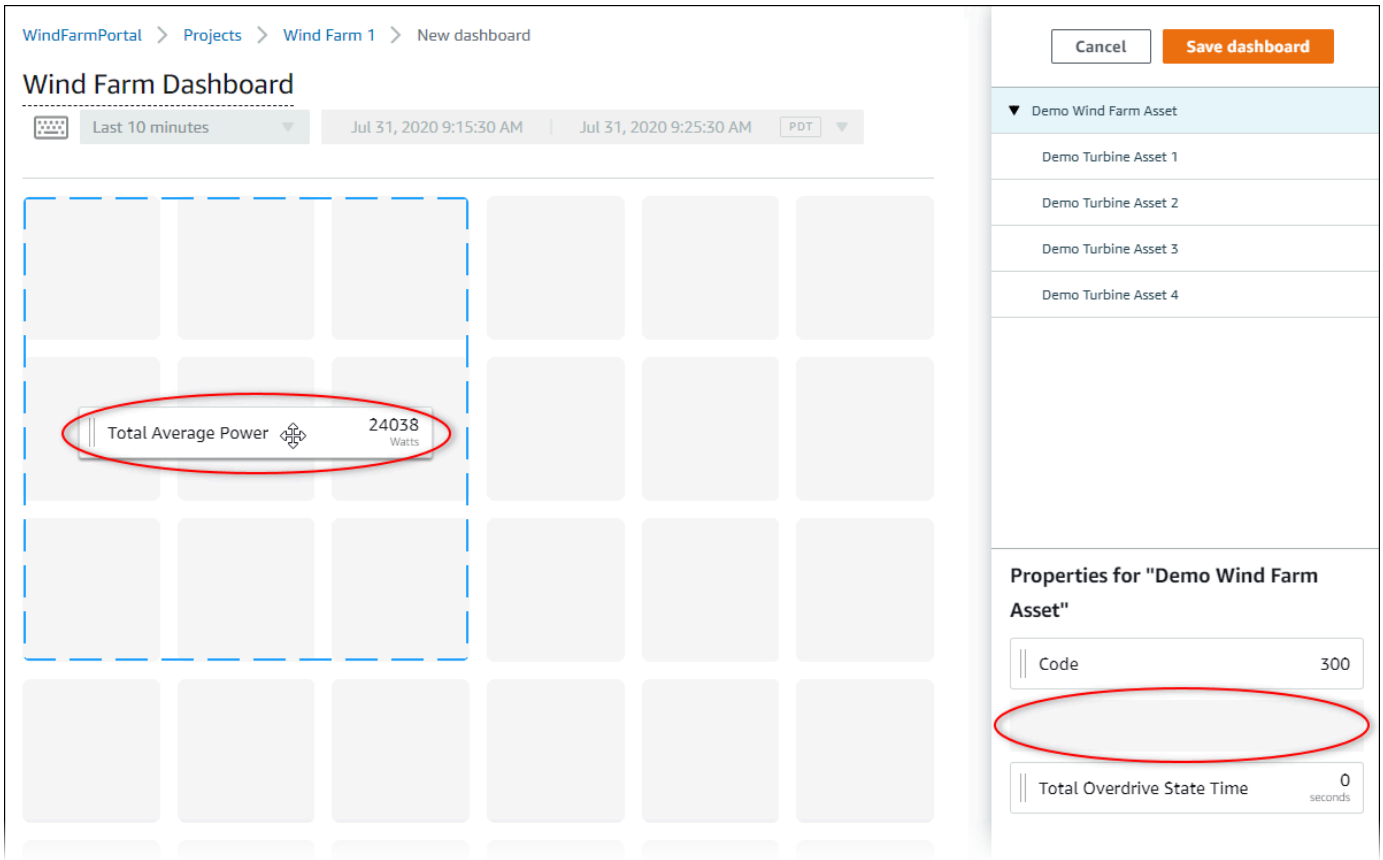
1. 在新项目的页面上，选择创建控制面板以创建控制面板并打开其编辑页面。

   在控制面板的编辑页面中，您可以将资产属性从资产层次结构拖动到控制面板以创建可视化。然后，您可以在控制面板中编辑每个可视化的标题、图例标题、类型、大小和位置。

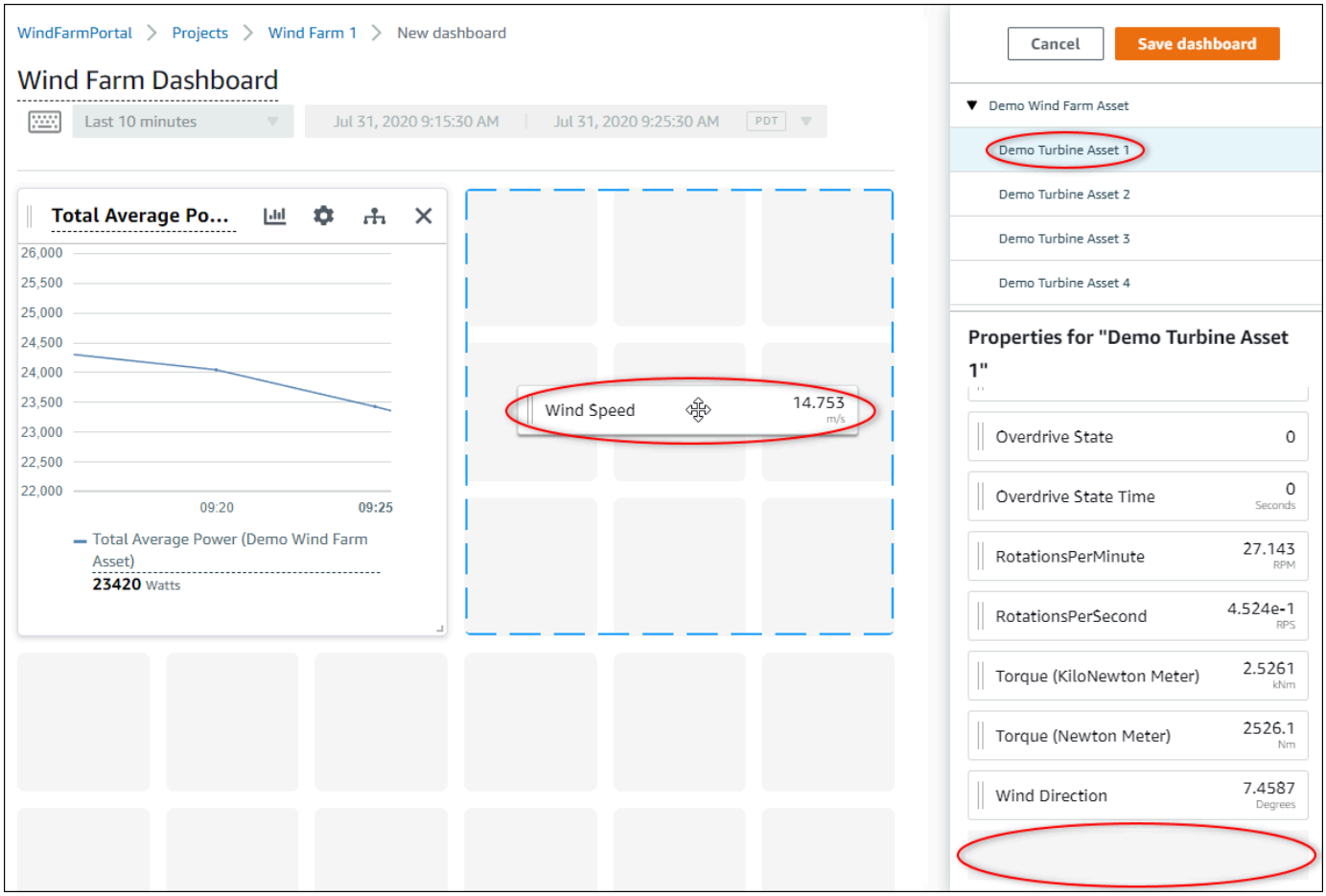
2. 输入您的控制面板的名称。



3. 拖动 Total Average Power来自 的 Demo Wind Farm Asset到仪表板以创建可视化。

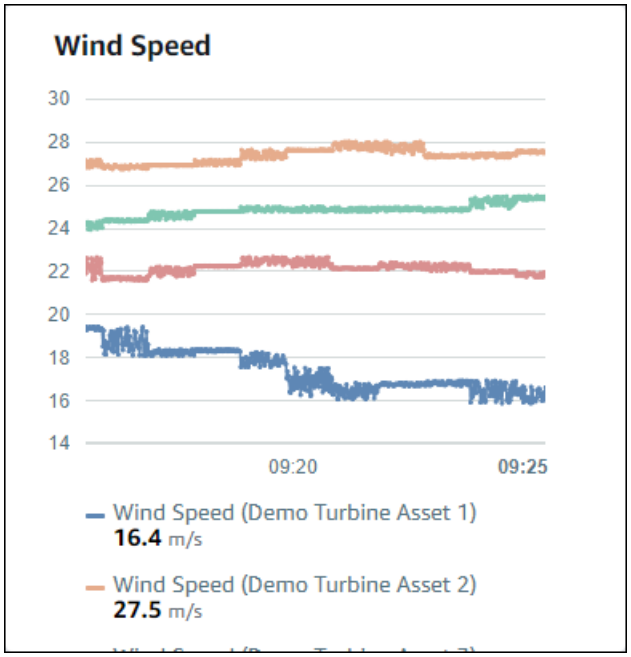


4. 选择 Demo Turbine Asset 1显示该资产的属性，然后拖动 Wind Speed进入仪表板以创建风速的可视化效果。

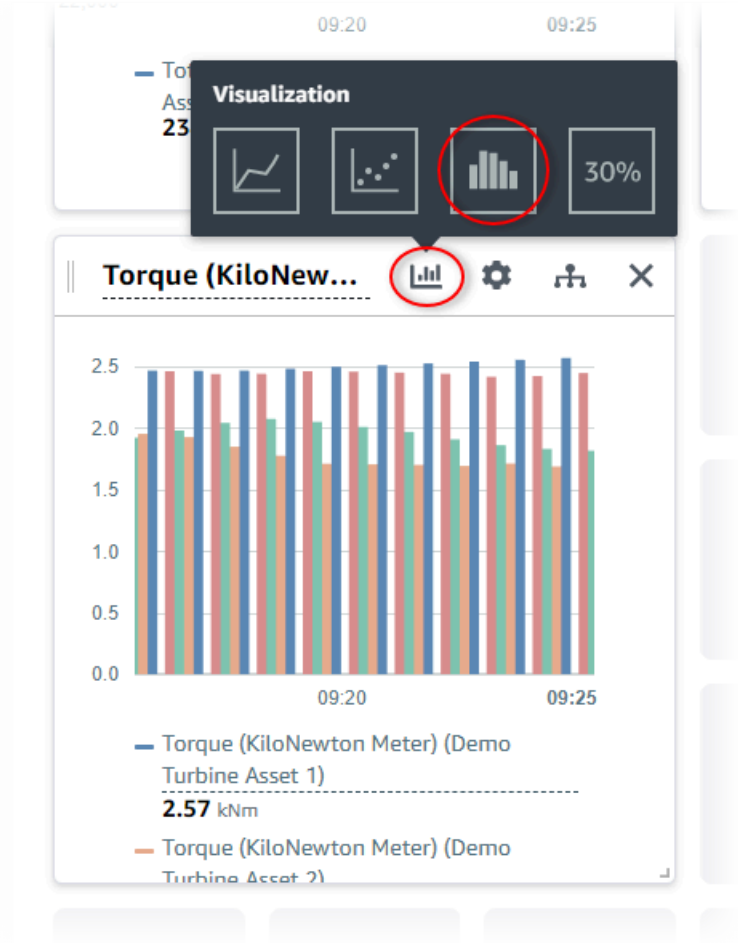


5. 添加 Wind Speed到每个风速的新可视化视图 Demo Turbine Asset 2, 3 和 4（按顺序排列）。

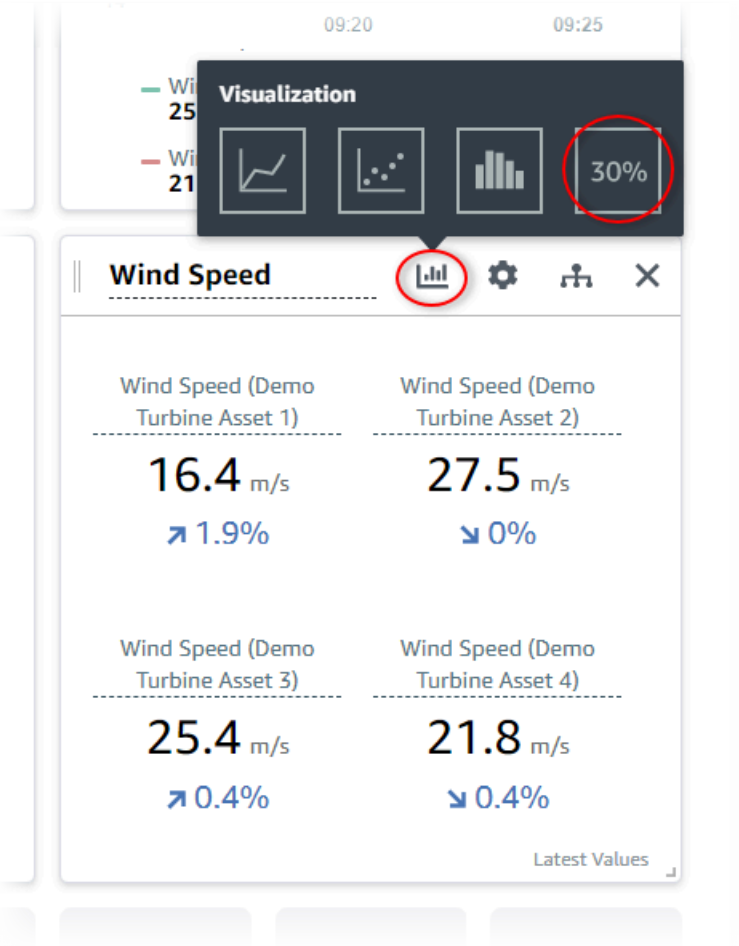
   您的 Wind Speed可视化应与以下屏幕截图类似。



6. 对风力涡轮机重复步骤 4 和 5 Torque (KiloNewton Meter)属性可创建风力涡轮机扭矩的可视化效果。
7. 选择可视化类型图标 Torque (KiloNewton Meter)可视化，然后选择条形图图标。

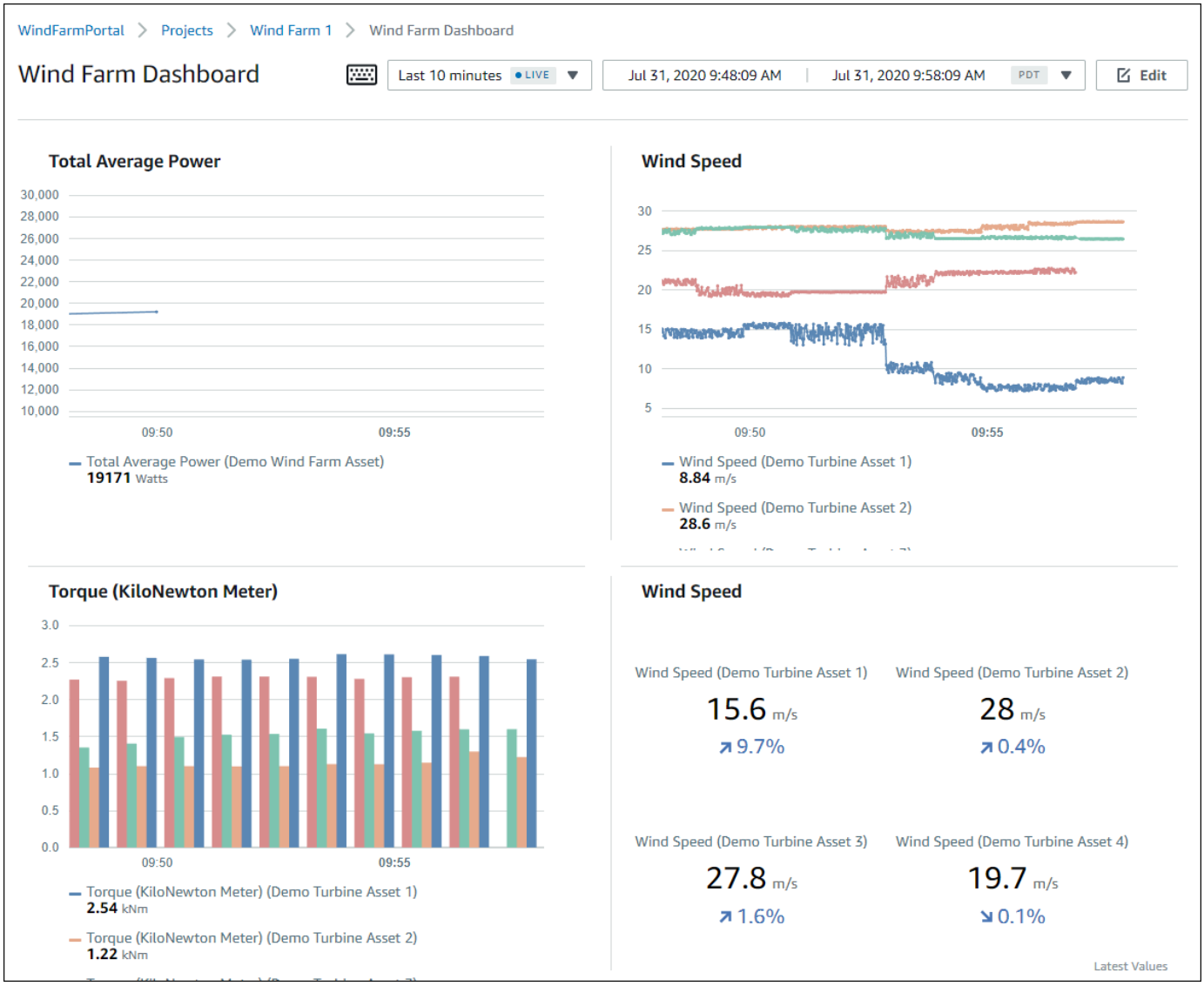


8. 对风力涡轮机重复步骤 4 和 5 Wind Direction属性来创建风向的可视化效果。
9. 选择可视化类型图标 Wind Direction可视化，然后选择 KPI 图表图标 (30%).



10. （可选）根据需要对每个可视化的标题、图例标题、类型、大小和位置进行其他更改。
11. 选择右上角的保存控制面板以保存控制面板。

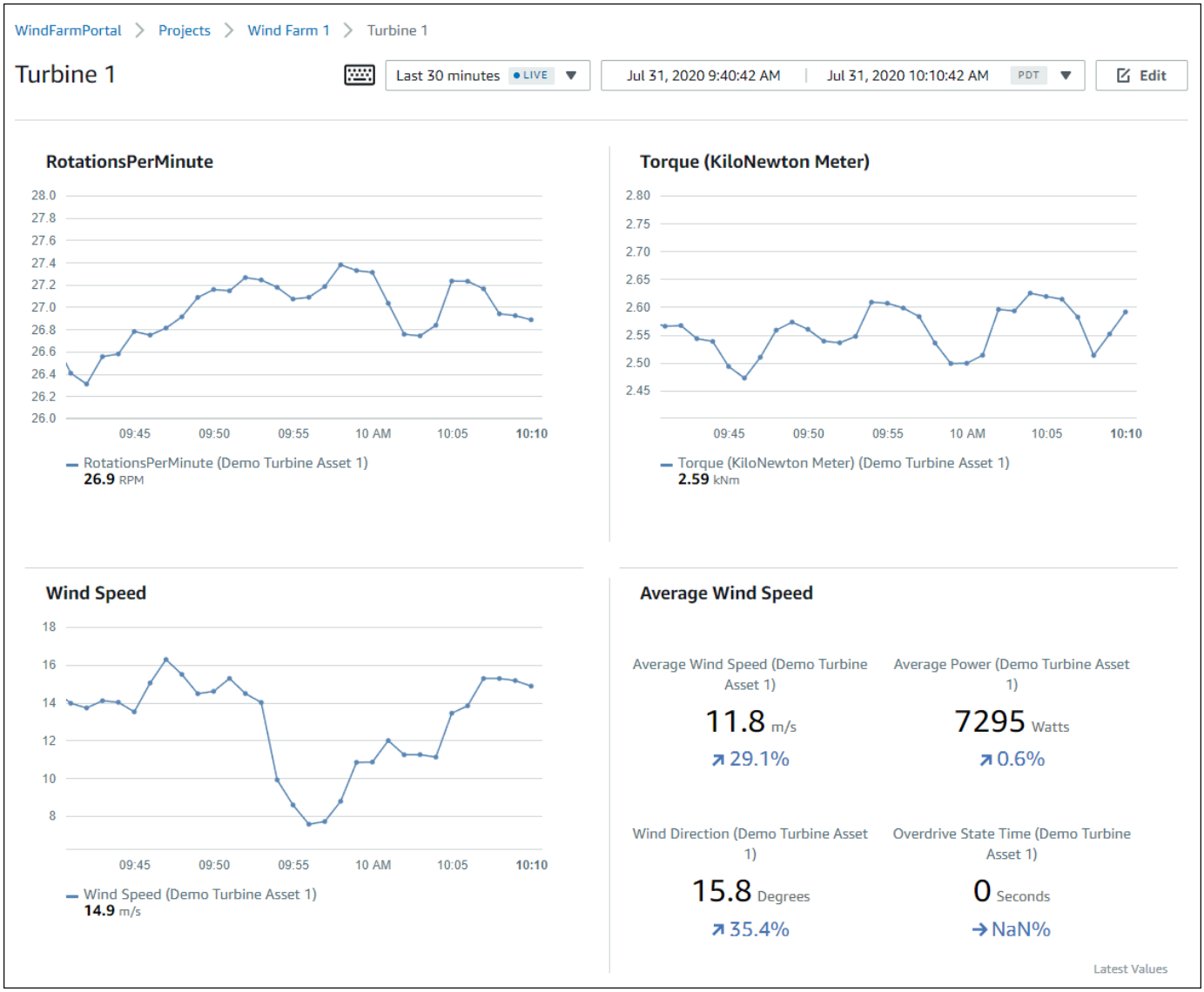
    您的控制面板应类似于以下屏幕截图。



12. （可选）为每个风力涡轮机资产创建额外的控制面板。

    作为最佳实践，我们建议您为每个资产创建控制面板，以便项目查看者可以调查每个单独资产的任何问题。您只能向每个可视化添加最多 5 个资产，因此在许多情况下必须为分层资产创建多个控制面板。

    演示风力涡轮机的控制面板类似于以下屏幕截图。



13. （可选）在可视化中更改时间轴或选择数据点，以浏览控制面板中的数据。有关更多信息，请参阅 Amazon IoT SiteWise Monitor 应用指南中的查看控制面板。

步骤 5：浏览门户

在此过程中，您能够以比 Amazon IoT SiteWise 门户管理员权限更少的用户身份浏览门户。

浏览门户网站并完成教程

- （可选）如果您将其他用户作为所有者或查看者添加到项目中，则可以以这些用户的身份登录门户。这样，您就能够以比门户管理员权限更少的用户身份浏览门户。

> **Important**
>
> 您需要为每位登录门户的用户付费。有关更多信息，请参阅Amazon IoT SiteWise 定价。

要以其他用户身份浏览门户，请执行以下操作：

a. 选择门户左下角的注销退出 Web 应用程序。

b. 在 IAM Identity Center 应用程序门户的右上角选择注销，退出您的 IAM Identity Center 用户。

c. 以您分配为项目所有者或项目查看者的 IAM Identity Center 用户身份登录门户。有关更多信息，请参阅 步骤 2：登录门户。

您已完成本教程。在 Monito SiteWise r 中完成对演示风电场的探索后，请按照下一个步骤清理资源。

步骤 6：在完成教程后清理资源

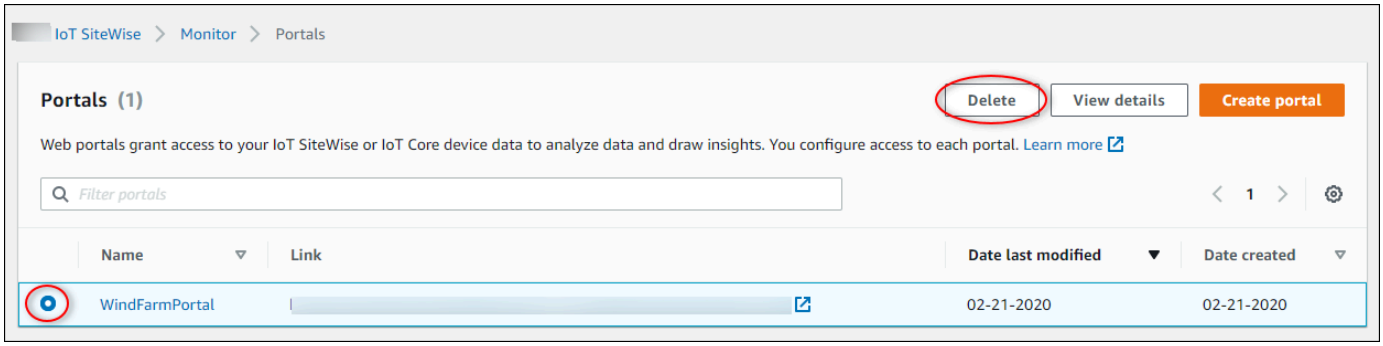
完成本教程后，您可以清除资源。如果用户未登录您的门户，则您无需为 Amazon IoT SiteWise 付费，但您可以删除门户和 Amazon IAM Identity Center 目录 用户。您的演示风电场资产将在您创建演示时选择的时段结束时间删除，或者您也可以手动删除演示。有关更多信息，请参阅 删除演 Amazon IoT SiteWise 示。

使用以下过程删除门户和 IAM Identity Center 用户。

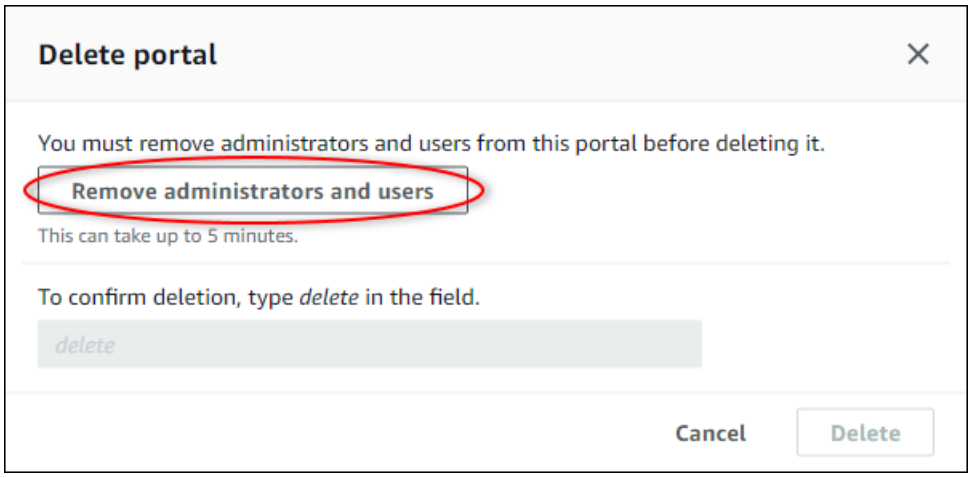
删除门户

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择 门户。
3. 选择您的门户 WindFarmPortal，然后选择删除。

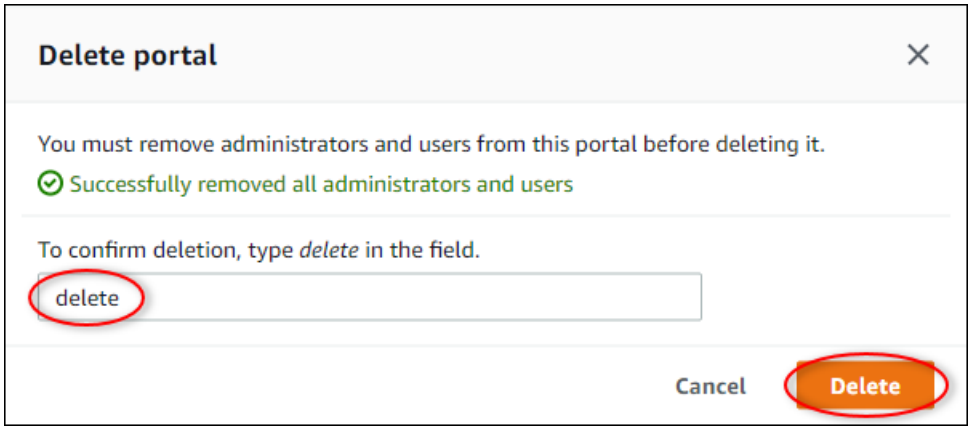
   删除门户或项目时，与已删除项目关联的资产不受影响。



4. 在删除门户对话框中，选择删除管理员和用户。

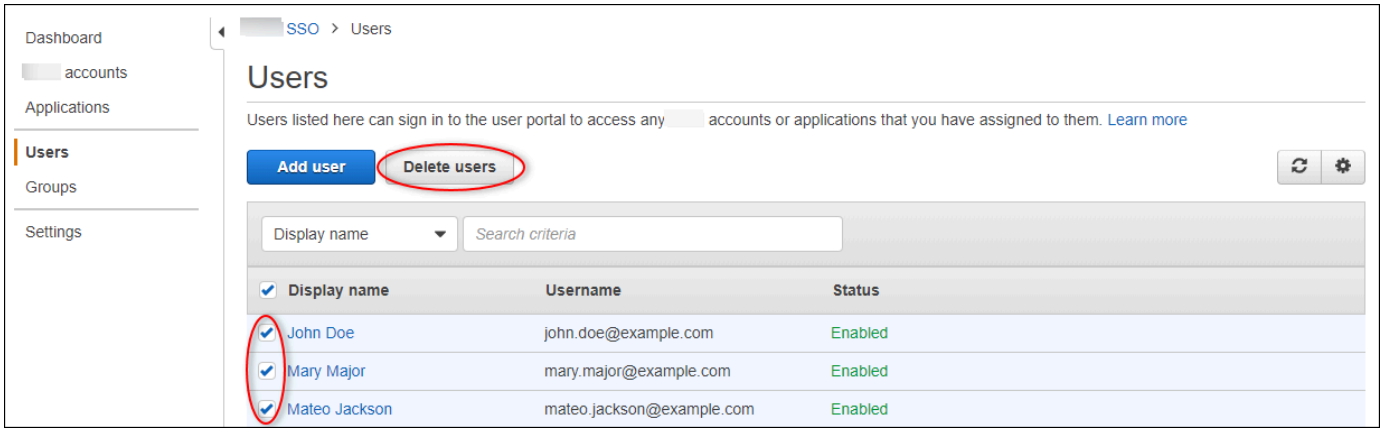


5. 输入 **`delete`** 以确认删除，然后选择删除。

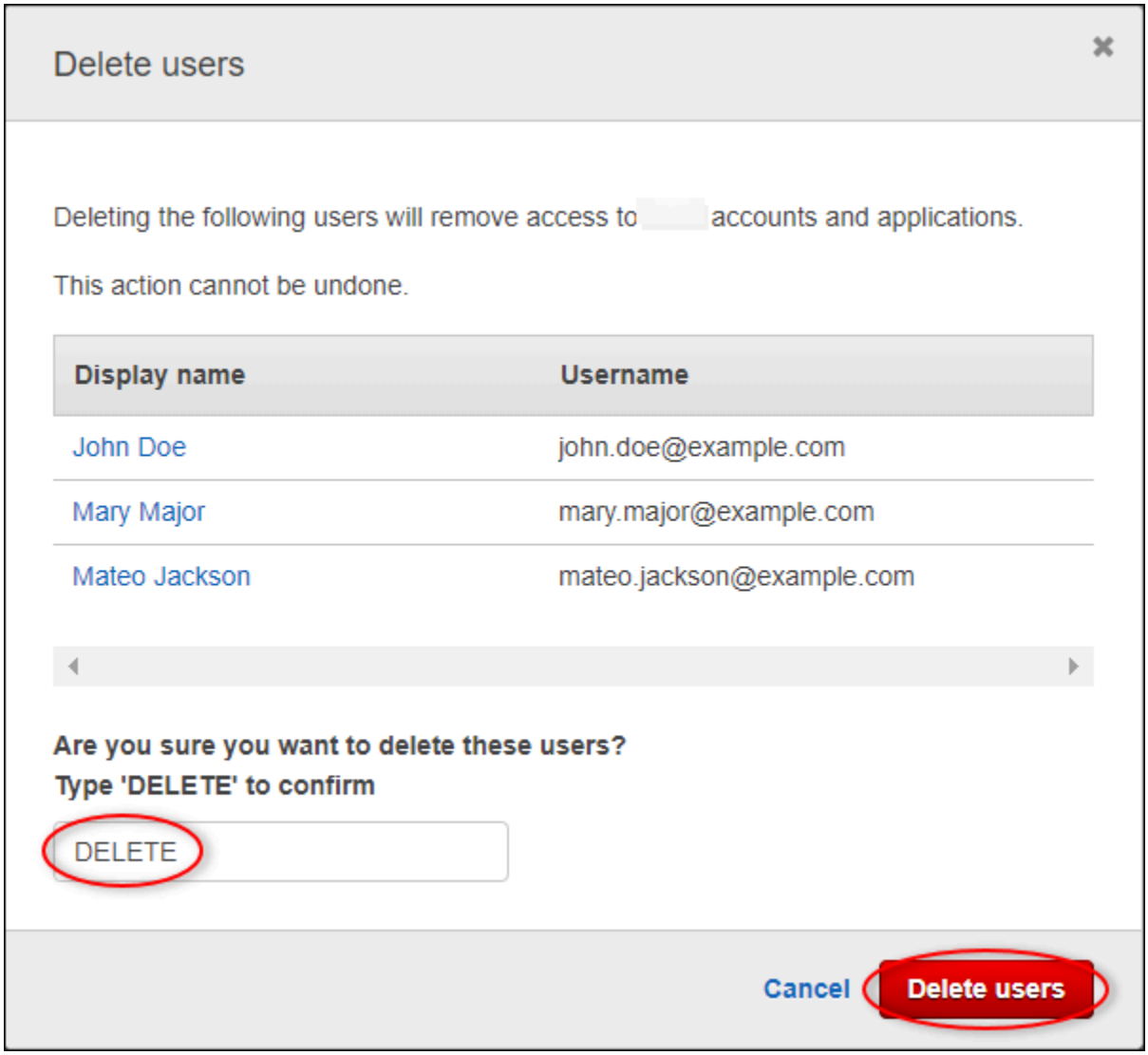


删除 IAM Identity Center 用户

1. 导航到 IAM Identity Center 控制台。
2. 在左侧导航窗格中，选择 用户。
3. 选中要删除的各个用户旁的复选框，然后选择 删除用户。



4. 在删除用户 对话框中，输入 **DELETE**，然后选择删除用户。



将属性值更新发布到 Amazon DynamoDB

本教程介绍了一种使用 Amazon Dynamo DB 存储数据的便捷方法，无需重复查询 API 即可更轻松地访问历史资产数据。 Amazon IoT SiteWise 完成本教程后，您可以创建自定义软件以使用资产数据，例如整个风电场的实时风速和风向图。如果您希望在未实施自定义软件解决方案的情况下监控和可视化数据，请参阅使用监控数据 Amazon IoT SiteWise Monitor。

在本教程中，您将在提供风力发电场示例数据集的 Amazon IoT SiteWise 演示基础上进行构建。您可以配置从风电场演示的属性值更新，通过 Amazon IoT Core 规则将数据发送到您创建的 DynamoDB 表。启用属性值更新后， Amazon IoT SiteWise 会以 MQTT 消息的 Amazon IoT Core 形式将您的数据发送到。然后，根据这些消息的内容定义执行操作（例如 DynamoDB 操作）的 Amazon IoT 核心规则。有关更多信息，请参阅 与其他 Amazon 服务互动。

主题

- 先决条件
- 步骤 1：配置 Amazon IoT SiteWise 以发布属性值更新
- 步骤 2：在 Amazon IoT Core 中创建规则
- 步骤 3：创建 DynamoDB 表
- 步骤 4：配置 DynamoDB 规则操作
- 步骤 5：在 DynamoDB 中浏览数据
- 步骤 6：在完成教程后清理资源

先决条件

要完成本教程，您需要：

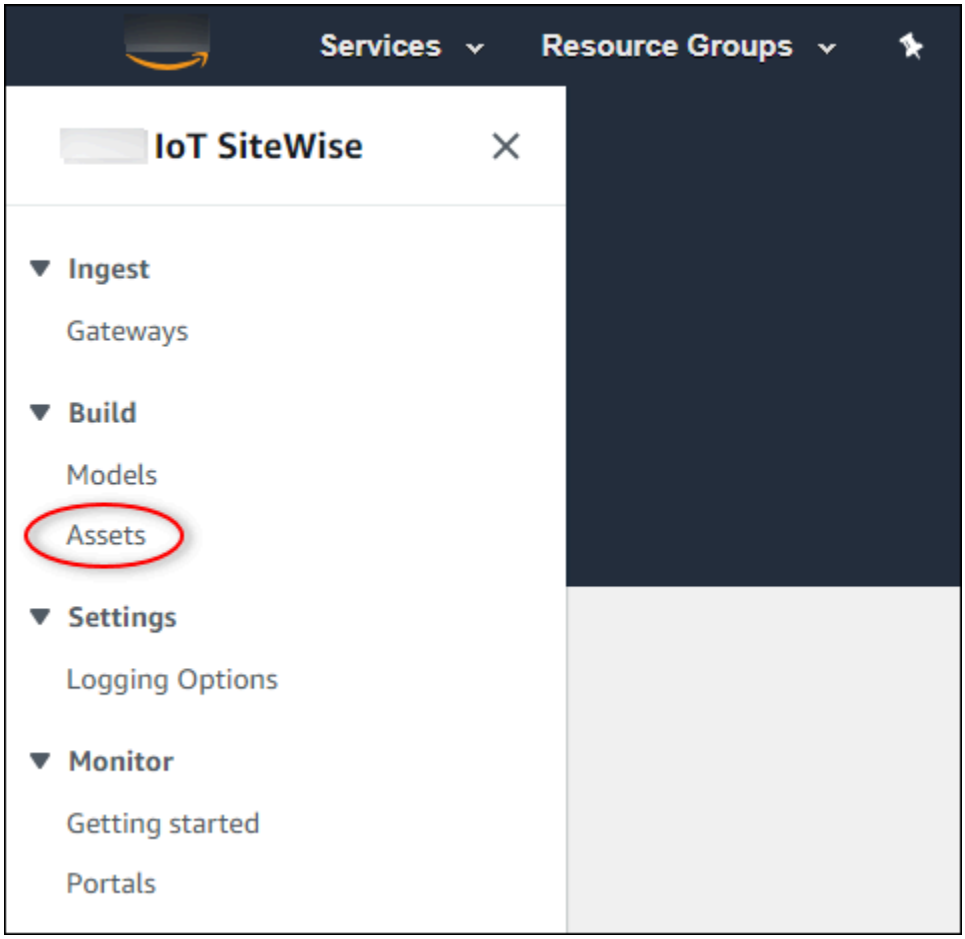
- 一个 Amazon 账户。如果没有，请参阅设置一个 Amazon 账户。
- 一台运行 Windows、macOS、Linux 或 Unix 的开发计算机，用于访问。 Amazon Web Services Management Console有关更多信息，请参阅 What is the Amazon Web Services Management Console?
- 具有管理员权限的 IAM 用户。
- 正在运行的 Amazon IoT SiteWise 风力发电场演示。当你设置演示时，它会定义模型和资产， Amazon IoT SiteWise 并将数据流式传输到它们以表示风力发电场。有关更多信息，请参阅 使用演 Amazon IoT SiteWise 示。

步骤 1：配置 Amazon IoT SiteWise 以发布属性值更新

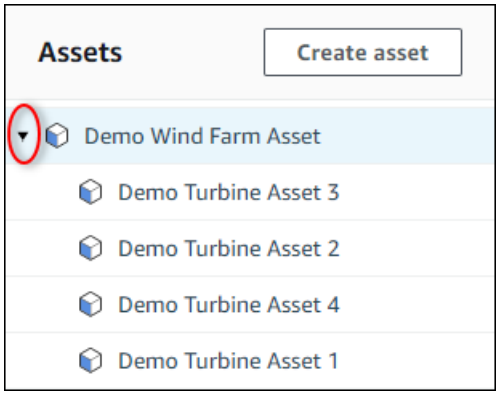
在此过程中，您可以在演示涡轮机资产上启用属性值通知 Wind Speed属性。启用属性值通知后，在 MQTT 消息中将每个值更新 Amazon IoT SiteWise 发布到 Amazon IoT Core。

在资产属性上启用属性值更新通知

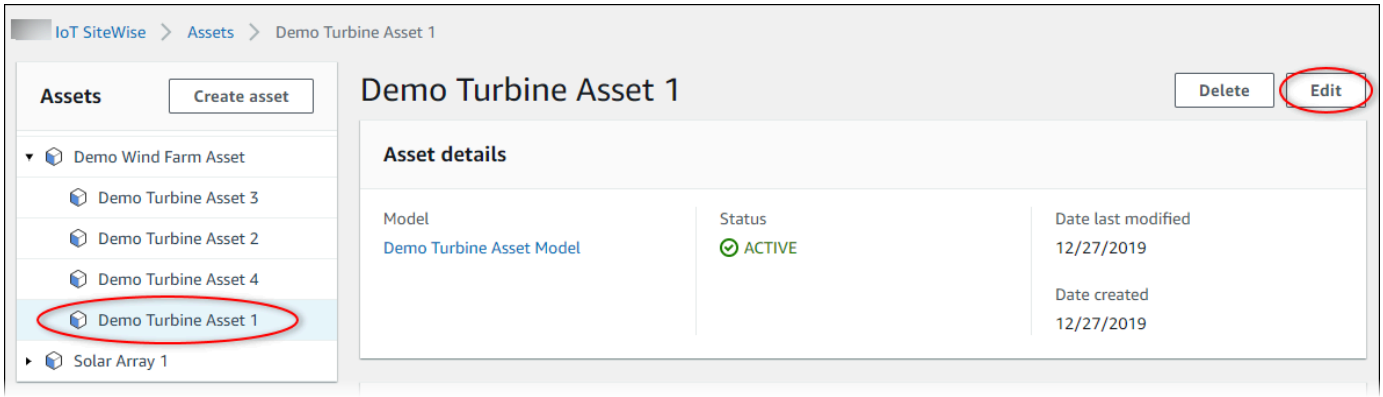
1. 登录 Amazon IoT SiteWise 控制台。
2. 查看支持的Amazon IoT SiteWise 终端节点和配额，必要时切换 Amazon 区域。 Amazon IoT SiteWise 切换到你正在运行 Amazon IoT SiteWise 演示的区域。
3. 在左侧导航窗格中，选择 资产。



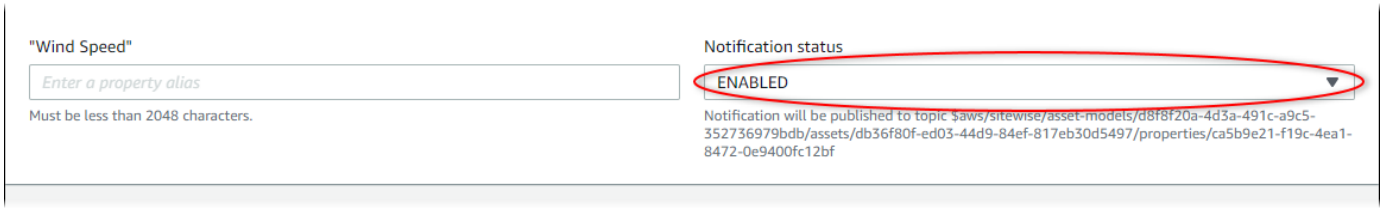
4. 选择旁边的箭头 Demo Wind Farm Asset以扩展风电场资产的层次结构。



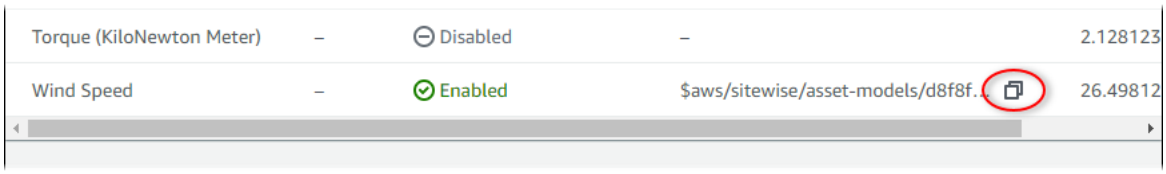
5. 选择一个演示涡轮机，然后选择 编辑。



6. 更新 Wind Speed属性的通知状态为 “已启用”。



7. 选择页面底部的 保存资产。
8. 对于每个演示涡轮机资产，重复步骤 5 至 7。
9. 选择演示涡轮机（例如，Demo Turbine Asset 1).
10. 选择 测量值。
11. 选择旁边的复制图标 Wind Speed属性可将通知主题复制到剪贴板。保存通知主题以在本教程后面使用。您只需记录一个涡轮机中的通知主题。



通知主题应类似于以下示例。

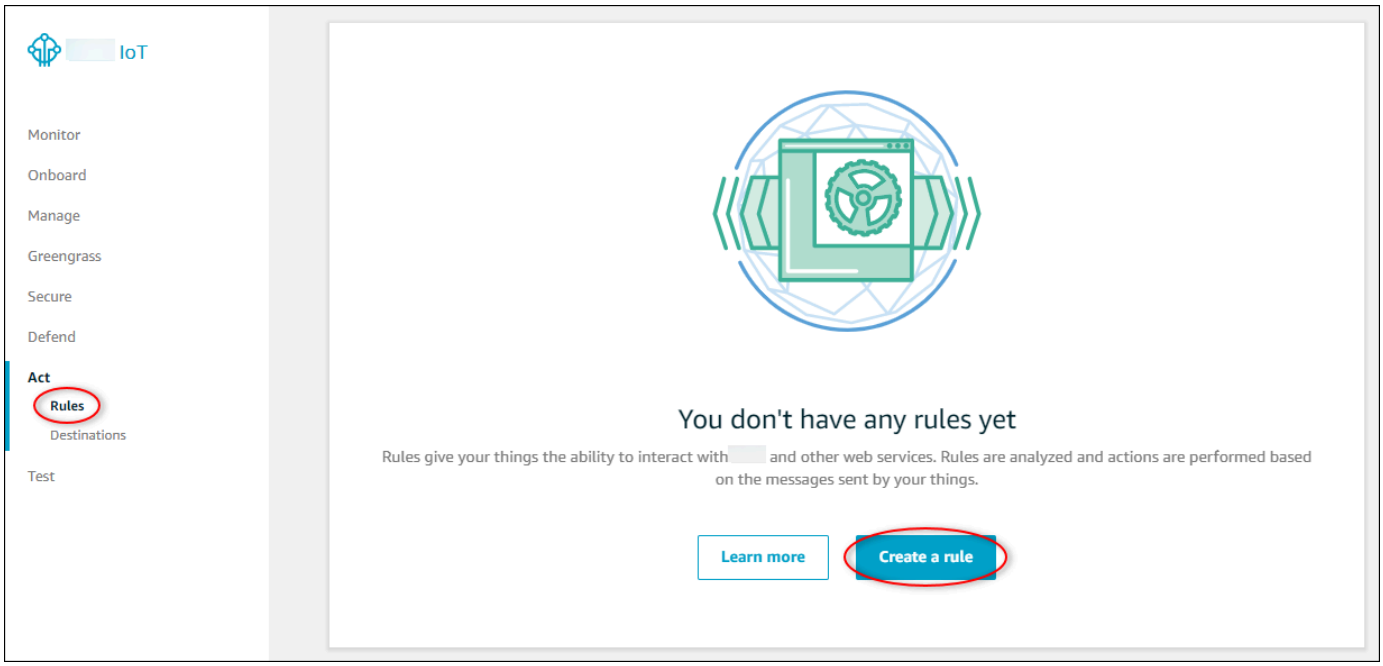
```
$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE
```

步骤 2：在 Amazon IoT Core 中创建规则

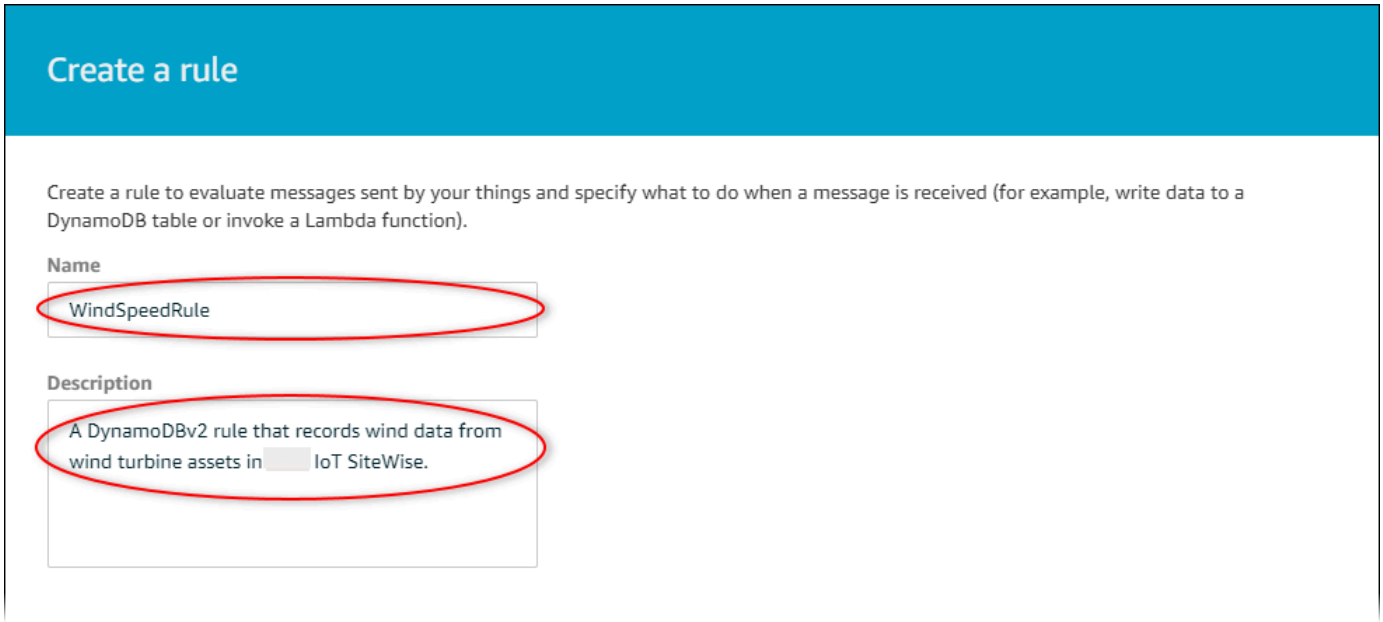
在此过程中，您将在 Amazon IoT Core 中创建一条规则，用于解析属性值通知消息并将数据插入到 Amazon DynamoDB 表中。 Amazon IoT 核心规则解析 MQTT 消息，并根据每条消息的内容和主题执行操作。之后，创建带有 DynamoDB 操作的规则，将数据插入到 DynamoDB 表（这是您在本教程中创建的表）中。

使用 DynamoDB 操作创建规则

1. 导航至 Amazon IoT 控制台。如果显示 开始 按钮，请选择该按钮。
2. 在左侧导航窗格中，选择行动，然后选择规则。



3. 如果显示您还没有任何规则对话框，请选择创建规则。否则，选择 创建。
4. 为规则输入名称和描述。



5. 查找您在本教程前面保存的通知主题。

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

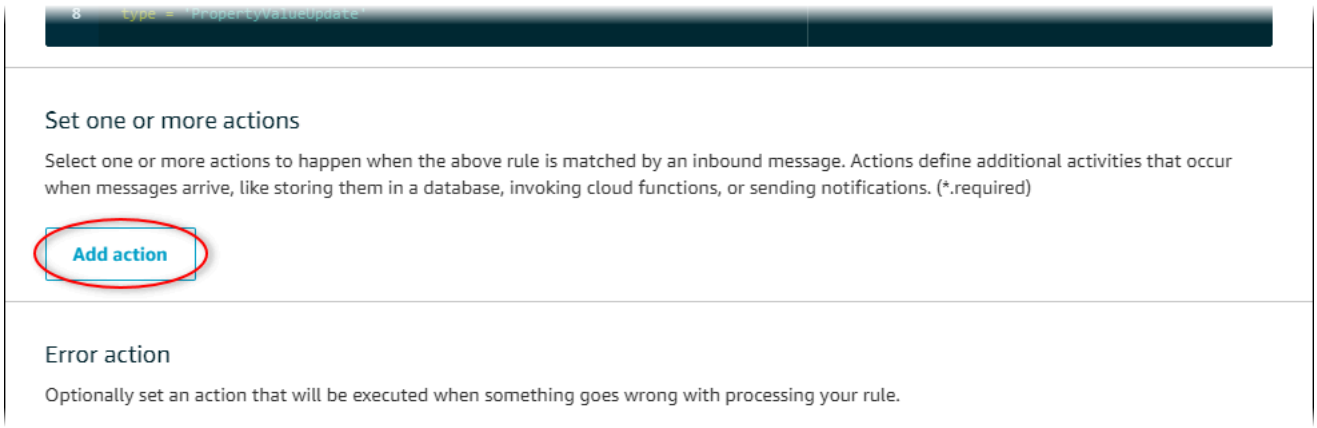
将主题中的资产 ID（`assets/` 后的 ID）替换为 `+`。这将为所有演示风力涡轮机资产选择风速属性。`+` 主题筛选器接受主题中的单个级别的所有节点。您的主题应类似于以下示例。

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

6. 输入以下规则查询语句。将 `FROM` 部分中的主题替换为您的通知主题。

```
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 设置一个或多个操作 下，选择 添加操作。



8. 在选择操作页面上，选择将消息拆分为 DynamoDB 表的多列 (Dynam o)。DBv2

9. 选择页面底部的 配置操作。
10. 在 配置操作 页面，选择 创建新资源。

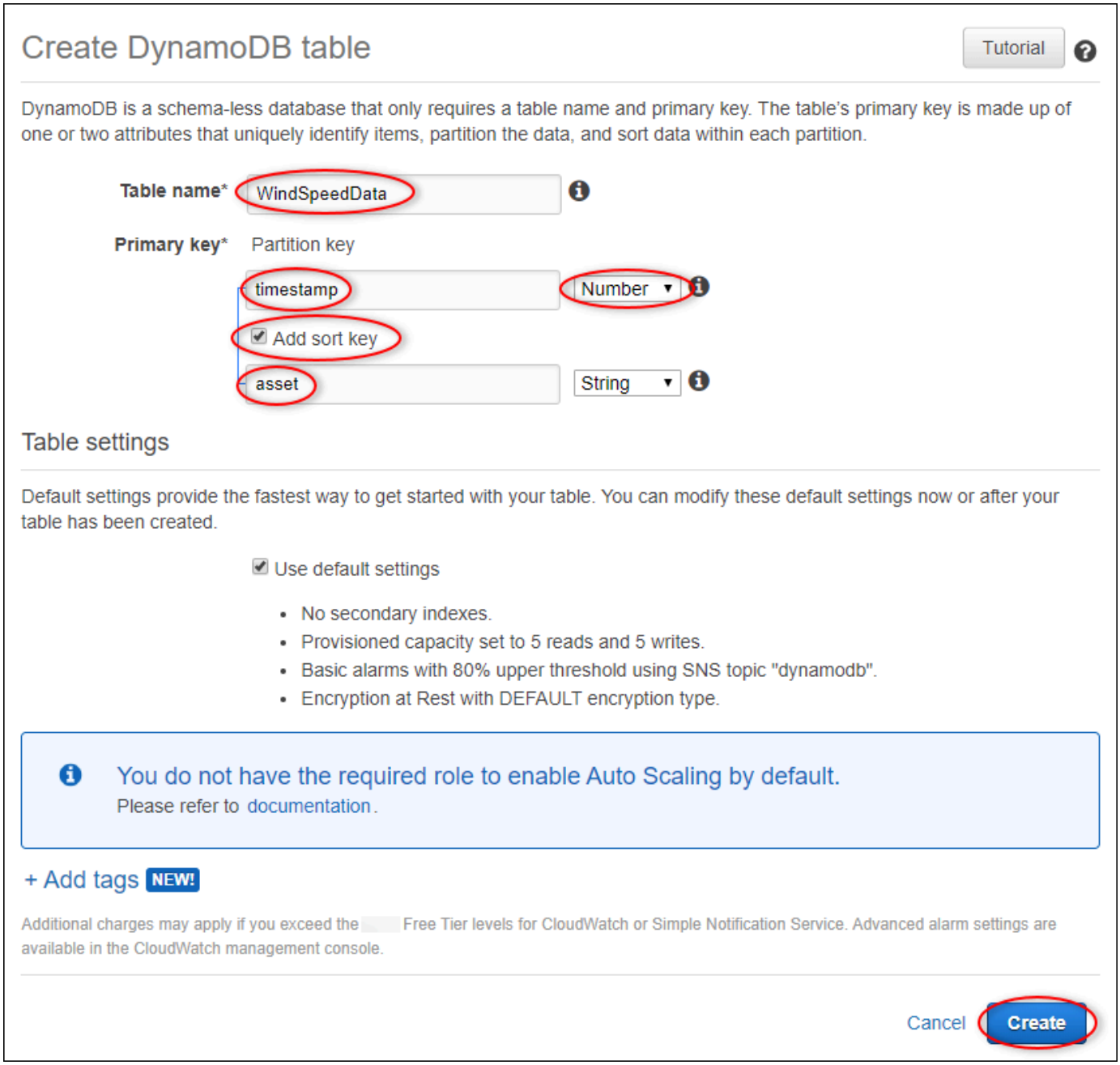
    DynamoDB 控制台将打开新的选项卡。在完成以下过程时，请将规则操作选项卡保持打开状态。

步骤 3：创建 DynamoDB 表

在该过程中，您创建一个 Amazon DynamoDB 表以从规则操作中接收风速数据。

创建 DynamoDB 表

1. 在 DynamoDB 控制台控制面板中，选择创建表。
2. 输入表的名称。



3. 对于 主键，请执行以下操作：

   a. 输入 **`timestamp`** 以作为分区键。

   b. 选择 数字 类型。

   c. 选中 添加排序键 复选框。

   d. 输入 **`asset`** 以作为排序键，并保留默认排序键类型 字符串。

4. 选择创建。

在 正在创建表 通知消失时，您的表将准备就绪。

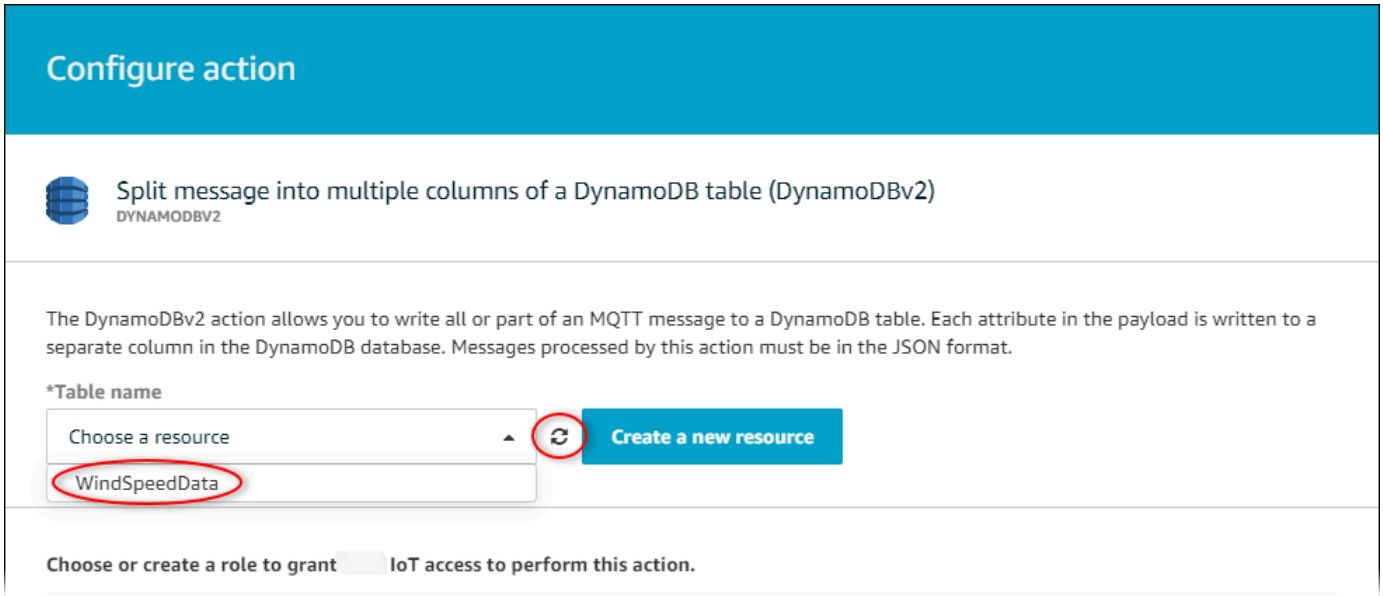
5. 返回到具有 配置操作 页面的选项卡。在完成以下过程时，请将 DynamoDB 选项卡保持打开状态。

步骤 4：配置 DynamoDB 规则操作

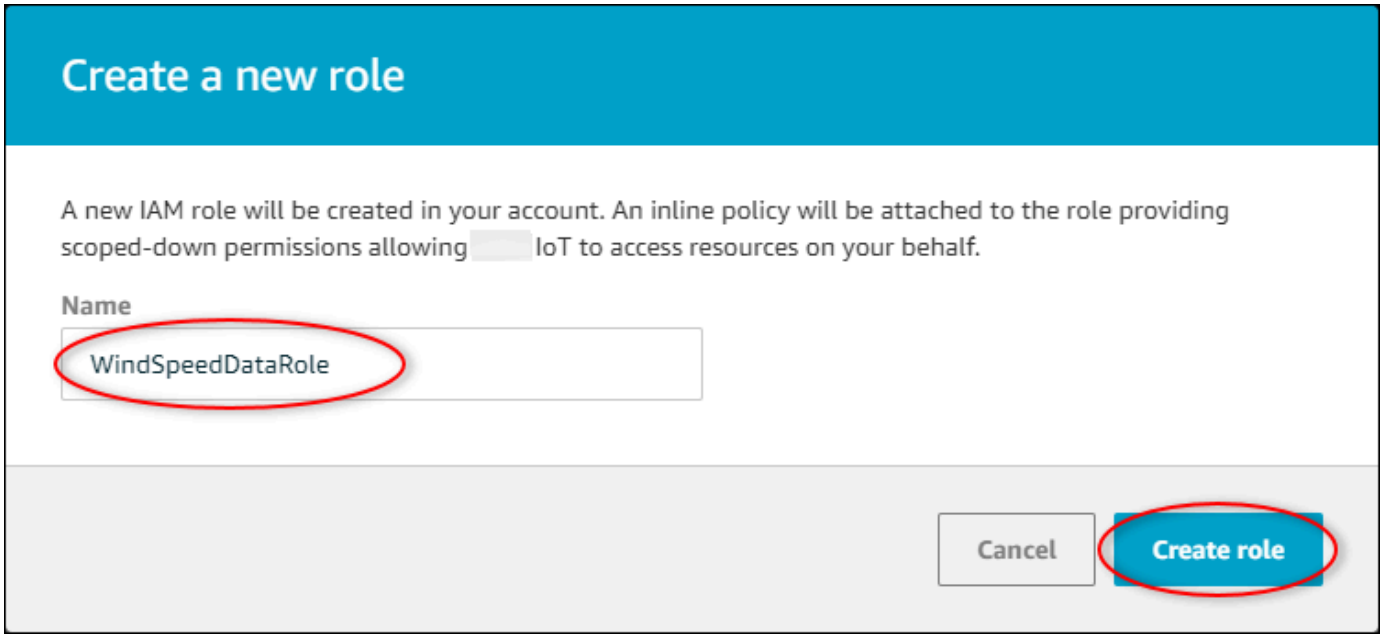
在此过程中，您将配置 Amazon DynamoDB 规则操作，将来自属性值更新的数据插入到新的 DynamoDB 表中。

配置 DynamoDB 规则操作

1. 在配置操作页面上，刷新表名称列表，然后选择新的 DynamoDB 表。



2. 选择创建角色以创建一个 IAM 角色，该角色向 Amazon IoT 核心授予执行规则操作的访问权限。
3. 输入角色名称，然后选择创建角色。



4. 选择添加操作。
5. 选择页面底部的 创建规则 以完成创建规则的过程。

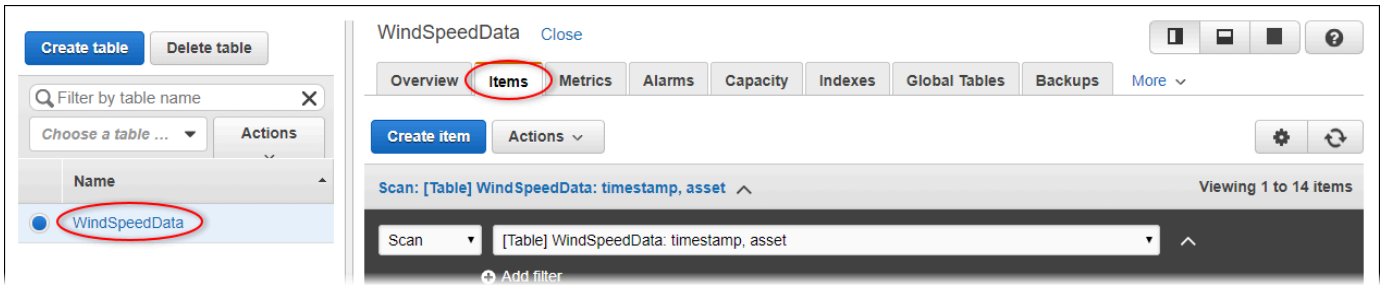
   将开始在 DynamoDB 表中显示您的演示资产数据。

步骤 5：在 DynamoDB 中浏览数据

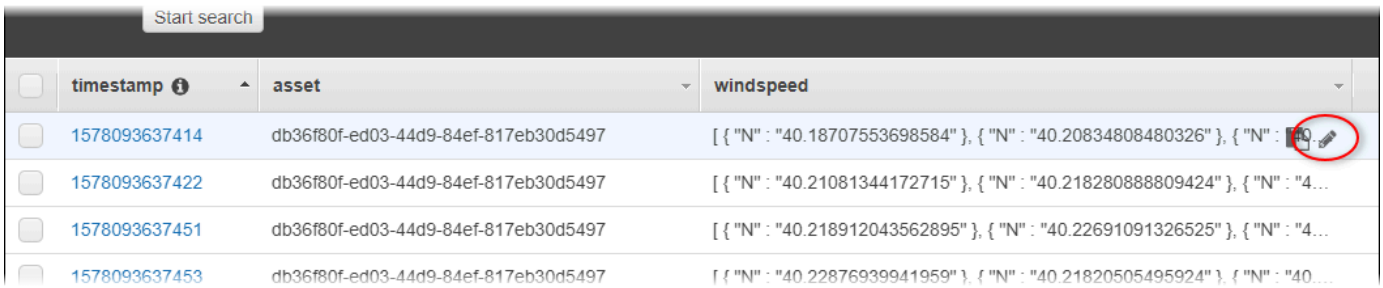
在该过程中，您在新的 Amazon DynamoDB 表中浏览演示资产的风速数据。

在 DynamoDB 中浏览资产数据

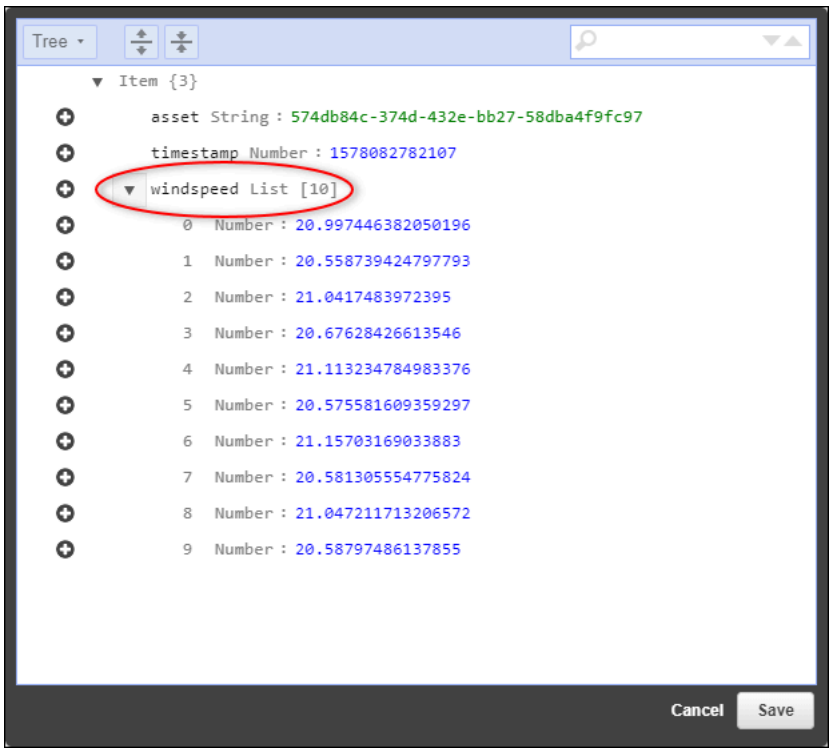
1. 返回到打开了 DynamoDB 表的选项卡。
2. 在您以前创建的表中，选择 项目 选项卡以查看表中的数据。如果您没有看到表中的行，请刷新页面。如果在几分钟后没有显示行，请参阅对规则进行故障排除（DynamoDB）。



3. 在表的一行中，选择编辑图标以展开数据。



4. 选择旁边的箭头 windspeed用于扩展风速数据点列表的结构。每个列表都反映了风电场演示发送到 Amazon IoT SiteWise 的一批风速数据点。如果您设置规则操作以供自己使用，则可能需要使用不同的数据格式。有关更多信息，请参阅 在中查询资产属性通知 Amazon IoT SiteWise。



完成本教程后，禁用或删除规则并删除 DynamoDB 表，避免产生额外费用。要清理资源，请参阅步骤 6：在完成教程后清理资源。

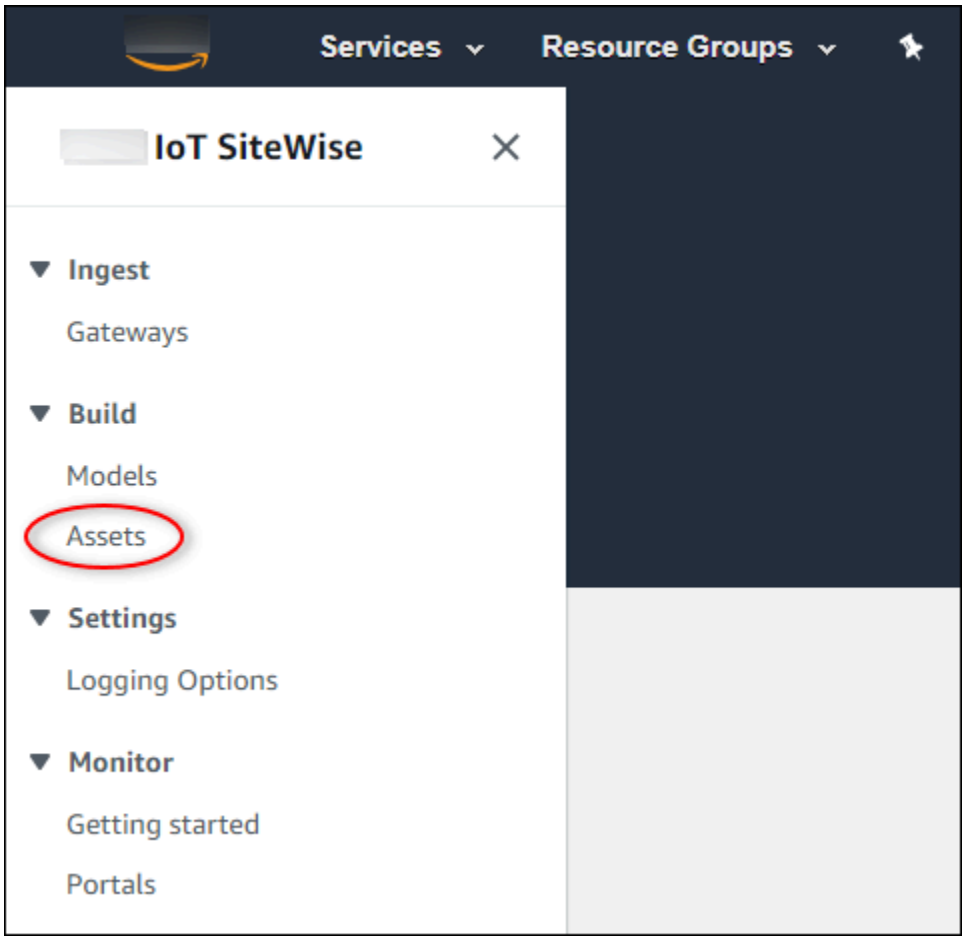
步骤 6：在完成教程后清理资源

完成本教程后，清理资源以避免产生额外费用。您的演示风电场资产将在您创建演示时选择的时段结束时间删除。您也可以手动删除演示。有关更多信息，请参阅 删除演 Amazon IoT SiteWise 示。

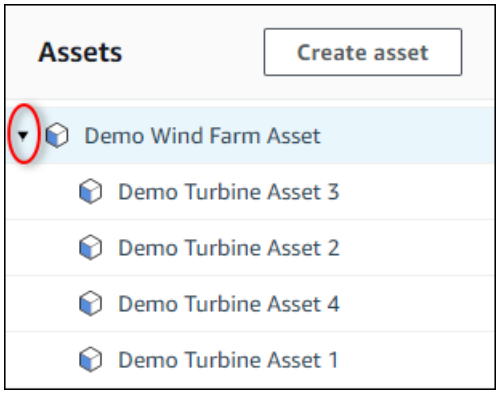
使用以下过程禁用属性值更新通知（如果您没有删除演示）、禁用或删除您的 Amazon IoT 规则，以及删除您的 DynamoDB 表。

禁用资产属性上的属性值更新通知

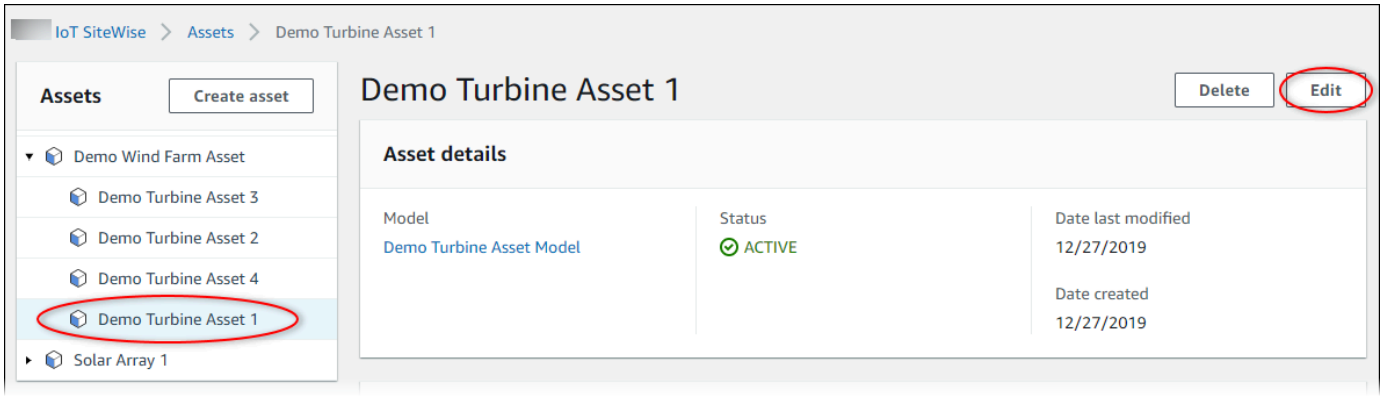
1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择 资产。



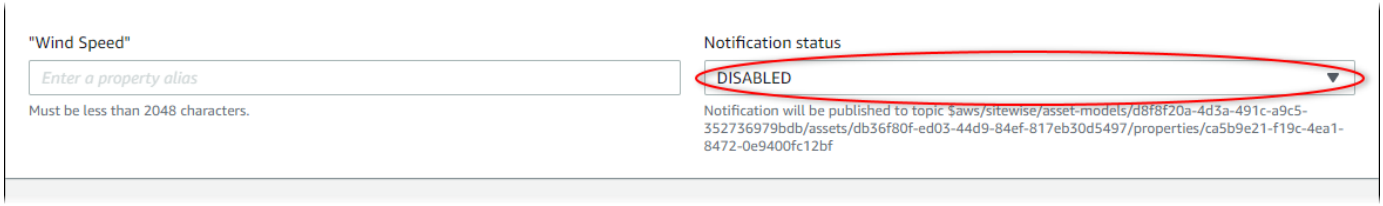
3. 选择旁边的箭头 Demo Wind Farm Asset以扩展风电场资产的层次结构。



4. 选择一个演示涡轮机，然后选择 编辑。



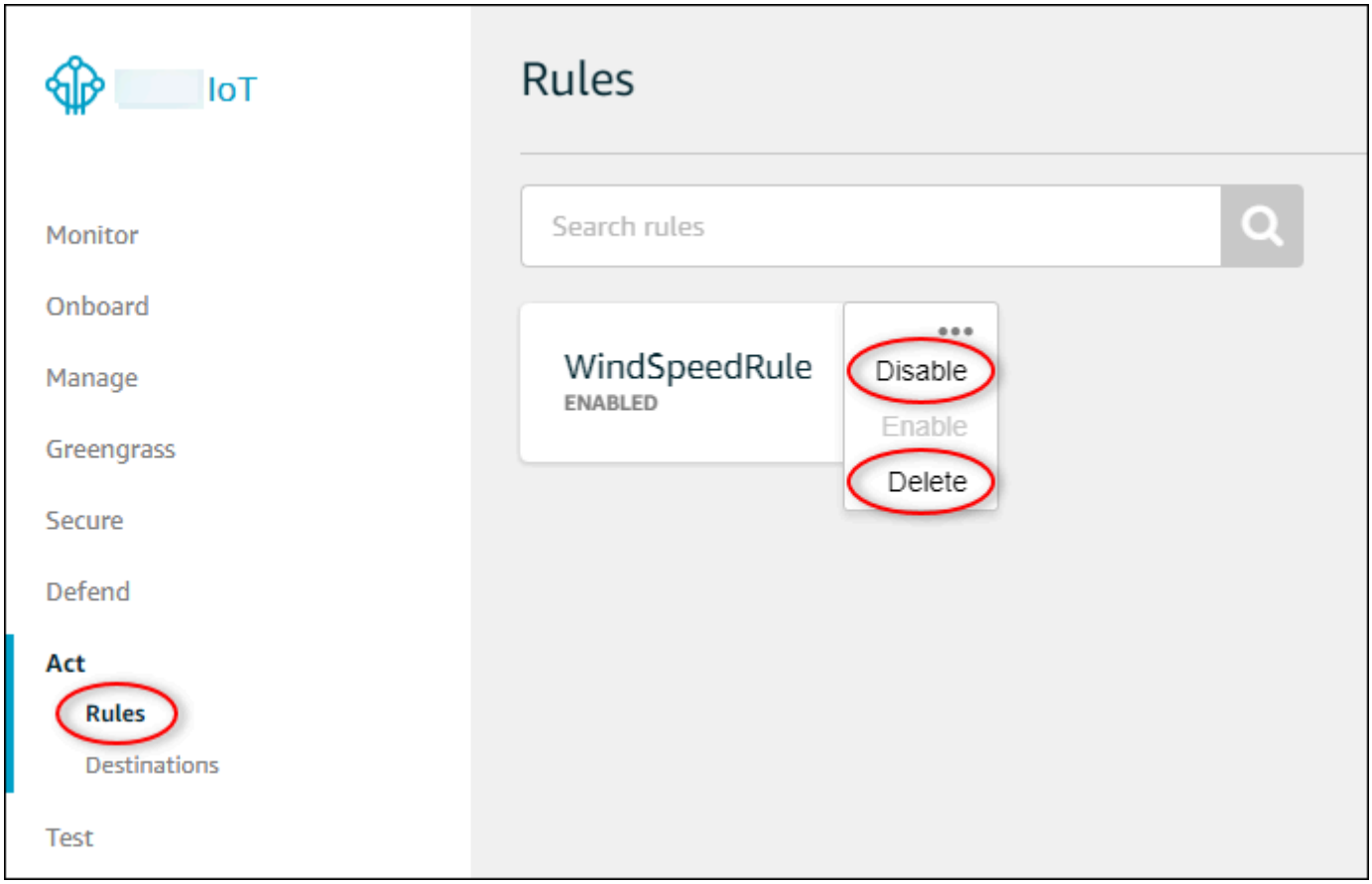
5. 更新 Wind Speed属性的通知状态为 “已禁用”。



6. 选择页面底部的 保存资产。
7. 对于每个演示涡轮机资产，重复步骤 4 至 6。

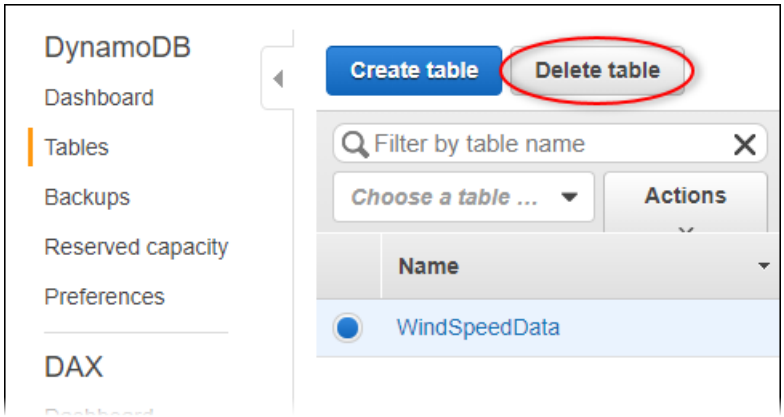
禁用或删除中的规则 Amazon IoT Core

1. 导航至 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择行动，然后选择规则。
3. 选择您的规则上的菜单，然后选择 禁用 或 删除。

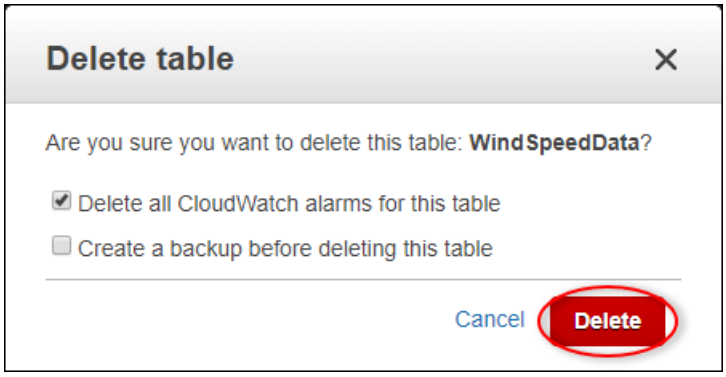


删除 DynamoDB 表

1. 导航到 DynamoDB 控制台。
2. 在左侧导航窗格中，选择 表。
3. 选择您之前创建的表，WindSpeedData.
4. 选择 删除表。



5. 在 删除表 对话框中，选择 删除。


Delete table
Are you sure you want to delete this table: WindSpeedData?
Delete all CloudWatch alarms for this table
Create a backup before deleting this table
Cancel
Delete

将数据摄取到 Amazon IoT SiteWise

Amazon IoT SiteWise 旨在有效地收集工业数据并将其与相应的资产相关联，这些资产代表了工业运营的各个方面。本文档重点介绍将数据摄入的实际方面 Amazon IoT SiteWise，提供了针对不同工业用例量身定制的多种方法。有关构建虚拟工业运营的说明，请参阅为工业资产建模。

您可以使用以下任一选项将 Amazon IoT SiteWise 工业数据发送到：

- Amazon IoT SiteWise 边缘-使用 SiteWise Edge 网关作为 Amazon IoT SiteWise 和您的数据服务器之间的中介。 Amazon IoT SiteWise 提供了可以在任何平台上部署的 Amazon IoT Greengrass 组件，这些组件可以运行 Amazon IoT Greengrass 以设置 SiteWise Edge 网关。此选项支持与 OPC UA 服务器协议的链接。
- Amazon IoT SiteWise API-使用 Amazon IoT SiteWise API 上传来自任何其他来源的数据。使用我们的流式传输 BatchPutAssetPropertyValueAPI 在几秒钟内进行摄取，或者使用面向批量的 CreateBulkImportJobAPI 来促进更具成本效益的大批量摄取。
- Amazon IoT 核心规则-使用Amazon IoT 核心规则上传 Amazon IoT 事物或其他 Amazon 服务发布的 MQTT 消息中的数据。
- Amazon IoT Events ac tions —使用中由特定事件触发的Amazon IoT EventsAmazon IoT Events操作。此方法适用于数据上传与事件发生相关的场景。
- Amazon IoT Greengrass 流管理器-使用Amazon IoT Greengrass 流管理器使用边缘设备从本地数据源上传数据。此选项适用于数据来自本地或边缘站点的情况。

这些方法为管理来自不同来源的数据提供了一系列解决方案。深入研究每个选项的详细信息，以全面了解所提供的数据摄取功能。 Amazon IoT SiteWise

管理数据流 Amazon IoT SiteWise

数据流是包含历史时间序列数据的资源。每个数据流都由唯一的别名标识，因此可以更轻松地跟踪每条数据的来源。数据流是在收到第一个时间序列数据 Amazon IoT SiteWise 时自动创建的。如果第一个时间序列数据使用别名标识，则使用该别名 Amazon IoT SiteWise 创建新的数据流，前提是尚未为资产属性分配该别名。或者，如果第一个时间序列数据使用资产 ID 和属性 ID 进行标识，则 Amazon IoT SiteWise 会创建一个新的数据流并将该数据流与资产属性相关联。

有两种方法可以为资产属性分配别名。使用的方法取决于是先将数据发送到 Amazon IoT SiteWise 第一个，还是先创建资产。

- 如果将数据发送到 f Amazon IoT SiteWise irst，则会自动使用分配的别名创建数据流。稍后创建资产时，使用 AssociateTimeSeriesToAssetPropertyAPI 将数据流及其别名关联到资产属性。
- 如果首先创建了资产，请使用 UpdateAssetPropertyAPI 为资产属性分配别名。稍后将数据发送到时 Amazon IoT SiteWise，会自动创建数据流并将其与资产属性相关联。

目前，您只能将数据流与测量值相关联。测量值是一种资产属性类型，表示设备的原始传感器数据流，如带有时间戳的温度值或带有时间戳的每分钟转数 (RPM) 值。

当这些测量值定义指标或转换时，传入的数据会触发特定的计算。值得注意的是，一个资产属性一次只能链接到一个数据流。

Amazon IoT SiteWise 使用 `TimeSeries` Amazon 资源名称 (ARN) 资源来确定您的存储费用。有关更多信息，请参阅Amazon IoT SiteWise 定价。

以下各节介绍如何使用 Amazon IoT SiteWise 控制台或 API 来管理数据流。

主题

- 配置权限和设置
- 将数据流与资产属性关联
- 取消数据流与资产属性的关联
- 删除数据流
- 更新资产属性的别名
- 常见场景

配置权限和设置

数据流是在收到第一个时间序列数据 Amazon IoT SiteWise 时自动创建的。如果提取的数据未与资产属性关联，则 Amazon IoT SiteWise 会创建一个新的已解除关联的数据流，该数据流可配置为与资产属性相关联。使用 IAM 策略配置向其发送数据的网关的访问控制 Amazon IoT SiteWise，以指定要摄取的数据类型。

以下 IAM 策略禁止从网关提取已取消关联的数据，同时仍允许向与资产属性关联的数据流中提取数据：

Example 禁止从网关提取已取消关联的数据的 IAM 用户策略

```
{
```

```
    "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 禁止从网关提取所有数据的 IAM 用户策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyPutAssetPropertyValues",
      "Effect": "Deny",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": {
```

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

将数据流与资产属性关联

使用 Amazon IoT SiteWise 控制台 或管理您的数据流 Amazon CLI。

Console

使用 Amazon IoT SiteWise 控制台管理您的数据流。

要管理数据流（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 数据流。
3. 通过筛选数据流别名或在筛选器下拉菜单中选择 “取消关联的数据流” 来选择数据流。
4. 选择要更新的数据流。您可以选择多个数据流。单击右上角的 “管理数据流”。
5. 从 “更新数据流关联” 中选择要关联的数据流，然后单击 “选择测量” 按钮。
6. 在 “选择衡量指标” 部分，找到相应的资产衡量属性。选择测量值，然后单击 “选择”。
7. 对步骤 3 中选定的其他数据流执行步骤 4 和步骤 5。为所有数据流分配资产属性。
8. 选择 “更新” 以提交更改。将显示成功的确认横幅以确认更新。

Amazon CLI

要将数据流（由其别名标识）与资产属性（由其标识 IDs）相关联，请运行以下命令：

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

取消数据流与资产属性的关联

Console

使用 Amazon IoT SiteWise 控制台取消您的数据流与资产属性的关联。

取消数据流与资产属性的关联（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 数据流。
3. 通过筛选数据流别名或在筛选器下拉菜单中选择 “关联的数据流” 来选择数据流。
4. 选择要取消关联的数据流。数据流别名列必须包含别名。资产名称和资产属性名称列必须包含与数据流关联的资产属性的值。您可以选择多个数据流。
5. 单击右上角的 “管理数据流”。
6. 在 “更新数据流关联” 部分中，单击 “测量名称” 列中的 X。`submitted`状态应显示在 “状态” 列中。
7. 选择 “更新” 以提交更改。现在，数据流已与资产属性解除关联，现在使用别名来标识数据流。

Amazon CLI

要取消数据流与资产属性（由其 `ID` s 及其别名标识）的关联，请运行以下命令：

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

现在，数据流已与资产属性解除关联，使用别名来标识数据流。别名不再与资产属性关联，因为它现在已与数据流相关联。

删除数据流

从资产模型中移除某个属性时， Amazon IoT SiteWise 会从该资产模型管理的所有资产中删除该属性及其数据流。当资产被删除时，它还会删除该资产的所有属性及其数据流。如果必须保留数据流数据，则必须先将其与资产属性断开关联，然后再将其删除。

> **Warning**
> 从资产中删除属性时，关联的数据流也会被删除。要保留数据流，请先将其与资产属性解除关联，然后再从资产模型中删除该属性或删除资产。

Console

使用 Amazon IoT SiteWise 控制台取消您的数据流与资产属性的关联。

删除数据流（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 数据流。
3. 通过筛选数据流别名来选择数据流。
4. 选择要删除的数据流。您可以选择多个数据流。
5. 选择 “删除” 按钮以删除数据流。

Amazon CLI

使用 DeleteTimeSeriesAPI 按别名删除特定数据流。

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

更新资产属性的别名

别名在一个 Amazon 区域内必须是唯一的。这包括资产属性和数据流的别名。如果其他属性或数据流正在使用该别名，则不要为该资产属性分配别名。

Console

使用 Amazon IoT SiteWise 控制台更新资产属性别名。

更新资产属性别名（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。

2. 在导航窗格中，选择资产。
3. 从表格中选择资产。
4. 单击 Edit 按钮。
5. 在 “属性” 表中选择属性类型。
6. 找到该属性，然后在属性别名文本字段中键入新的别名。
7. 单击 “保存” 按钮保存更改。

Amazon CLI

要更新资产属性的别名，请运行以下命令：

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

> **Note**
> 如果当前启用了属性通知，则必须重新提供该通知以确保其继续启用。

常见场景

移动数据流

要更改数据流与其他资产属性的关联，请先取消该数据流与当前资产属性的关联。在解除数据流与资产属性的关联时，必须为该资产属性分配一个别名。

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

现在，将数据流重新分配给新的资产属性。

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

为资产属性分配别名时出错

使用 `UpdateAssetProperty` API 为属性分配别名时，您可能会看到以下错误消息：

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

此错误消息表示该别名未分配给该属性，因为该别名当前正由其他属性或数据流使用。

如果 Amazon IoT SiteWise 使用别名提取数据，则会发生这种情况。当发送数据时使用的别名未被其他数据流或资产属性使用时，将使用该别名创建一个新的数据流。以下两个选项解决了这个问题。

- 使用 `AssociateTimeSeriesToAssetProperty` API 将数据流及其别名关联到资产属性。
- 暂时停止数据摄取并删除数据流。使用 `UpdateAssetProperty` API 为资产属性分配别名，然后重新开启数据提取功能。

将数据流与资产属性关联时出错

将数据流与资产属性关联时，会看到以下错误消息。

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

此错误消息表明 asset 属性已与包含数据的数据流相关联。在将其他数据流与该资产属性关联之前，必须取消关联或删除该数据流。

> **Note**
>
> 解除数据流与资产属性的关联时，分配给该属性的别名会被赋予数据流。要使该别名保持分配给该属性的状态，请在取消关联数据流之前为该属性分配一个新的别名。

要保留存储在 asset 属性中的数据，请执行以下操作：

- 确保没有向资产属性提取任何数据，以防止创建新的数据流。
- 使用 `UpdateAssetProperty` API 为当前分配的数据流设置一个新的别名。
- 使用 `DisassociateTimeSeriesFromAssetProperty` API 取消当前数据流与资产属性的关联。
- 使用 `AssociateTimeSeriesToAssetProperty` API 将所需的数据流关联到资产属性。

如果必须删除存储在资产属性中的数据，请执行以下操作：

- 确保没有向资产属性提取任何数据，以防止创建新的数据流。
- 使用 `DeleteTimeSeries` API 删除当前分配的数据流。
- 使用 `AssociateTimeSeriesToAssetProperty` API 将所需的数据流关联到资产属性。

使用摄取数据 Amazon IoT SiteWise APIs

用于 Amazon IoT SiteWise APIs 将带有时间戳的工业数据发送到您的资产属性和测量属性。 APIs 接受包含 timestamp-quality-value (TQV) 结构的有效载荷。

BatchPutAssetPropertyValue API

使用 BatchPutAssetPropertyValue 操作上传数据。通过此操作，您可以一次上传多个数据条目，以从多个设备收集数据并在单个请求中全部发送。

> **Important**
>
> 该BatchPutAssetPropertyValue操作受以下配额限制：
>
> - 每个请求最多 10 个条目。
> - 每个条目最多 10 个属性值（TQV 数据点）。
> - Amazon IoT SiteWise 拒绝任何时间戳超过过去 7 天或将来 10 分钟以上的数据。
>
> 有关这些配额的更多信息，请参阅 Amazon IoT SiteWise API 参考BatchPutAssetPropertyValue中的。

要标识资产属性，请指定以下项之一：

- 向其发送数据的资产属性的 `assetId` 和 `propertyId`。

- `propertyAlias`，这是一个数据流别名（例如，`/company/windfarm/3/turbine/7/temperature`）。要使用此选项，您必须首先设置资产属性的别名。要设置属性别名，请参阅管理数据流 Amazon IoT SiteWise。

以下示例演示了如何从存储在 JSON 文件中的负载发送风力涡轮机的温度和每分钟转数 (RPM) 读数。

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

中的示例负载`batch-put-payload.json`包含以下内容。

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

指定enablePartialEntryProcessing为true允许提取所有不会导致失败的值。默认行为是false。如果值无效，则整个条目将无法读取。

负载中的每个条目都包含一个 entryId，您可以将其定义为任何唯一的字符串。如果任何请求条目失败，则每个错误都将包含相应请求的 entryId，以便您知道要重试的请求。

列表中的每个结构都propertyValues是一个 timestamp-quality-value (TQV) 结构，其中包含 a value timestamp、a 和（可选）。quality

- value – 包含下列字段之一的结构，具体取决于要设置的属性的类型：
  - booleanValue
  - doubleValue
  - integerValue
  - stringValue
  - nullValue
- nullValue— 一种结构，其中的以下字段表示属性值的类型，值为 Null，质量为BAD或UNCERTAIN。
  - valueType— {“B”、“D”、“S”、“I”} 的枚举

- timestamp – 包含当前 Unix 纪元时间（以秒为单位）的结构，timeInSeconds如果您有时间精确的数据，也可以在timestamp结构中设置offsetInNanos密钥。 Amazon IoT SiteWise 拒绝任何时间戳在过去超过 7 天或将来超过 10 分钟的数据点。
- quality –（可选）以下质量字符串之一：
  - GOOD –（默认）数据不受任何问题的影响。
  - BAD – 数据受传感器故障等问题的影响。
  - UNCERTAIN – 数据受传感器不准确等问题的影响。

  有关如何在计算中 Amazon IoT SiteWise 处理数据质量的更多信息，请参阅公式表达式中的数据质量。

CreateBulkImportJob API

使用 CreateBulkImportJob API 从 Amazon S3 导入大量数据。您的数据必须以 CSV 格式保存在 Amazon S3 中。数据文件可以包含以下列。

> **Note**
>
> 不支持早于世界标准时间 1970 年 1 月 1 日 00:00:00 的数据。
> 要标识资产属性，请指定以下项之一。
>
> - 您要向其发送数据的资产属性的 ASSET_ID 和 PROPERTY_ID。
> - ALIAS，这是一个数据流别名（例如，/company/windfarm/3/turbine/7/temperature）。要使用此选项，您必须首先设置资产属性的别名。要了解如何设置属性别名，请参阅the section called “管理数据流”。

- ALIAS – 标识属性的别名，例如 OPC UA 服务器数据流路径（例如，/company/windfarm/3/turbine/7/temperature）。有关更多信息，请参阅 管理数据流 Amazon IoT SiteWise。
- ASSET_ID – 资产的 ID。
- PROPERTY_ID – 资产属性的 ID。
- DATA_TYPE – 属性的数据类型可以是以下类型之一。
  - STRING – 最多 1024 字节的字符串。
  - INTEGER – 有符号的 32 位整数，范围是 [-2,147,483,648, 2,147,483,647]。
  - DOUBLE – 具有 [-10^100, 10^100] 范围和 IEEE 754 双精度的浮点数。

  - BOOLEAN – true 或 false。
- TIMESTAMP_SECONDS – 数据点的时间戳，以 Unix 纪元时间表示。
- TIMESTAMP_NANO_OFFSET – 从 TIMESTAMP_SECONDS 转换时的纳秒偏移量。
- QUALITY –（可选）资产属性值的质量。值可以是以下值之一。
  - GOOD –（默认）数据不受任何问题的影响。
  - BAD – 数据受传感器故障等问题的影响。
  - UNCERTAIN – 数据受传感器不准确等问题的影响。

  有关如何在计算中 Amazon IoT SiteWise 处理数据质量的更多信息，请参阅公式表达式中的数据质量。
- VALUE – 资产属性的值。

Example .csv 格式的数据文件

```
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
```

Amazon IoT SiteWise 提供了以下 API 操作来创建批量导入任务和获取有关现有任务的信息。

- CreateBulkImportJob— 创建新的批量导入任务。
- DescribeBulkImportJob— 检索有关批量导入任务的信息。
- ListBulkImportJob— 检索所有批量导入任务摘要的分页列表。

创建 Amazon IoT SiteWise 批量导入任务 (Amazon CLI)

使用 CreateBulkImportJobAPI 操作将数据从 Amazon S3 传输到 Amazon IoT SiteWise。CreateBulkImportJobAPI 支持摄取大量历史数据，并支持小批量缓冲摄取分析数据流。是一种经济高效的数据摄取机制。下面的示例使用了 Amazon CLI。

> **Important**
>
> 在创建批量导入任务之前，必须启用 Amazon IoT SiteWise 热层或 Amazon IoT SiteWise 冷层。有关更多信息，请参阅 在中配置存储设置 Amazon IoT SiteWise。CreateBulkImportJobAPI 支持将历史数据提取到， Amazon IoT SiteWise 并可以选择设置参数。 adaptive-ingestion-flag
>
> - 在设置为 `false` 时，该 API 会在不触发计算或通知的情况下摄取历史数据。
> - 在设置为 `true` 时，该 API 会在七天内摄取新数据、计算指标并转换数据，以优化持续分析和通知。

运行以下命令。*`file-name`*替换为包含批量导入任务配置的文件的名称。

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

Example 批量导入作业配置

以下是配置设置示例：

- 将 *`adaptive-ingestion-flag`* 替换为 `true` 或 `false`。
  - 如果设置为 `false`，则批量导入作业会将历史数据摄取到 Amazon IoT SiteWise。
  - 如果设置为 `true`，则批量导入作业会执行以下操作：
    - 将新数据摄入。 Amazon IoT SiteWise
    - 计算指标和转换，并支持对时间戳在七天内的数据发出通知。
- *`delete-files-after-import-flag`*替换`true`为可在将数据摄入到 Amazon IoT SiteWise 温层存储后从 Amazon S3 数据存储桶中删除。
- 将 amzn-s3-demo-bucket 替换为发送与此批量导入任务相关的错误的 Amazon S3 存储桶的*`-for-errors`*名称。
- 将 amzn-s3-demo-bucket 替换为发送与此批量导入任务相关的错误的 Amazon S3 存储桶*`-for-errors-prefix`*的前缀。

Amazon S3 使用前缀作为文件夹名称来整理存储桶中的数据。每个 Amazon S3 对象都有一个键，作为对象在存储桶中的唯一标识符。桶内的每个对象都只能有一个键。前缀必须以正斜杠（/）结尾。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的使用前缀整理对象。

- 将 amzn-s3-demo-bucket 替换为从中导入数据的 Amazon S3 存储桶的*-data*名称。
- *data-bucket-key*替换为包含您的数据的 Amazon S3 对象的密钥。每个对象都有一个作为唯一标识符的键。每个对象有且只有一个键。
- *data-bucket-version-id*替换为版本 ID，以识别包含您的数据的 Amazon S3 对象的特定版本。此参数为可选的。
- *column-name*替换为.csv 文件中指定的列名。
- *job-name*替换为标识批量导入任务的唯一名称。
- *job-role-arn*替换为允许 Amazon IoT SiteWise 读取 Amazon S3 数据的 IAM 角色。

> **Note**
>
> 确保您的角色具有以下示例中显示的权限。将 amzn-s3-demo-bucket 替换为包含您的数据的 Amazon S3 存储桶的*-data*名称。此外，请*amzn-s3-demo-bucket-for-errors*替换为发送与此批量导入任务相关的错误的 Amazon S3 存储桶的名称。

```
{
    "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 响应

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

```
}
```

描述 Amazon IoT SiteWise 批量导入任务 (Amazon CLI)

使用 DescribeBulkImportJobAPI 操作检索有关中特定批量导入任务的信息 Amazon IoT SiteWise。如果作业失败，此操作会返回诸如作业状态、创建时间和错误信息之类的详细信息。您可以使用此操作来监控作业进度并排查问题。要使用 `DescribeBulkImportJob`，您需要 `CreateBulkImportJob` 操作的作业 ID。此 API 会返回以下信息：

- 正在导入的文件列表，包括它们的 Amazon S3 存储桶位置和密钥
- 错误报告位置（如适用）
- 作业配置详情，例如文件格式和 CSV 列名称
- 作业创建和上次更新时间戳
- 当前作业状态（例如，作业是处于正在进行中、已完成还是失败状态）
- 用于导入作业的 IAM 角色 ARN

对于已完成的作业，请查看结果，确认数据集成是否成功。如果作业失败，请检查错误详情以诊断和解决问题。

*job-ID*替换为要检索的批量导入任务的 ID。

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

Example 响应

```
{
   "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 Amazon IoT SiteWise 批量导入任务 (Amazon CLI)

使用 ListBulkImportJobsAPI 操作检索中批量导入任务的摘要列表 Amazon IoT SiteWise。此操作为监控和管理数据导入过程提供了一种有效方法。它会为每个作业返回以下关键信息：

- 作业 ID。每个批量导入作业的唯一标识符
- 作业名称。作业创建时所分配到的名称。
- 当前状态。作业的当前状态（例如，COMPLETED、RUNNING、FAILED）

ListBulkImportJobs 对于全面了解所有批量导入任务特别有用。这可以帮您跟踪多个数据导入，识别任何需要注意的作业，并保持工作流程井然有序。该操作支持分页，便于您高效地检索大量作业摘要。您可以将此操作 IDs返回的任务与操作配合使用，DescribeBulkImportJob以检索有关特定任务的更多详细信息。通过这个两步过程，您可以先简要了解所有作业，然后深入了解相关作业的详细信息。使用 `ListBulkImportJobs` 时，您可以应用筛选条件来缩小结果范围。例如，您可以根据状态来筛选

作业，以便仅检索已完成的作业或仅检索正在运行的作业。此特征可帮助您专注于与当前任务最相关的信息。如果有更多可用结果，该操作还会返回 `nextToken`。您可以在后续调用中使用此令牌来检索下一组作业摘要，这样一来，即使有大量批量导入作业，您也可以遍历所有作业。以下示例演示如何`ListBulkImportJobs`与一起使用 Amazon CLI 来检索已完成任务的列表。

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

Example 针对已完成作业筛选条件的响应

```
{
   "jobSummaries":[
      {
         "id":"bdbbfa52-d775-4952-b816-13ba1c7cb9da",
         "name":"myBulkImportJob",
         "status":"COMPLETED"
      },
      {
         "id":"15ffc641-dbd8-40c6-9983-5cb3b0bc3e6b",
         "name":"myBulkImportJob2",
         "status":"COMPLETED"
      }
   ]
}
```

此命令演示了如何使用 `ListBulkImportJobs` 来检索已完成但失败的作业列表。结果数量上限设置为 50 条，并使用下一页令牌来获取分页结果。

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

Amazon IoT SiteWise 使用 Amazon IoT Core 规则提取数据

使用中的规则将数据 Amazon IoT SiteWise 从 Amazon IoT 事物和其他 Amazon 服务发送到 Amazon IoT Core。规则转换 MQTT 消息并执行与 Amazon 服务交互的操作。 Amazon IoT SiteWise 规则操作将消息数据从 Amazon IoT SiteWise API 转发给BatchPutAssetPropertyValue操作。有关更多信息，请参阅 Amazon IoT 开发人员指南中的规则和 Amazon IoT SiteWise 操作。

如要按照教程逐步完成设置规则来通过设备影子摄取数据的步骤，请参阅从事物中摄 Amazon IoT SiteWise 取 Amazon IoT 数据。

您也可以将数据从发送 Amazon IoT SiteWise 到其他 Amazon 服务。有关更多信息，请参阅 与其他 Amazon 服务互动。

主题

- 授 Amazon IoT 予所需的访问权限
- 配置 Amazon IoT SiteWise 规则操作
- 使用 Basic Ingest in 降低成本 Amazon IoT SiteWise

授 Amazon IoT 予所需的访问权限

您可以使用 IAM 角色来控制每条规则可以访问的 Amazon 资源。在创建规则之前，您必须创建一个 IAM 角色，其策略允许该规则对所需 Amazon 资源执行操作。 Amazon IoT 在运行规则时担任此角色。

如果您在 Amazon IoT 控制台中创建规则操作，则可以选择根资产来创建有权访问所选资产层次结构的角色。有关如何为规则手动定义角色的更多信息，请参阅《Amazon IoT 开发人员指南 Amazon IoT 》中的授予所需访问权限和传递角色权限。

对于 Amazon IoT SiteWise 规则操作，您必须定义一个允许`iotsitewise:BatchPutAssetPropertyValue`访问规则向其发送数据的资产属性的角色。为了提高安全性，您可以在`Condition`属性中指定 Amazon IoT SiteWise 资产层次结构路径。

以下示例信任策略允许访问特定资产及其子级。

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

```
    }
  ]
}
```

从该策略中删除 `Condition`，以便于访问您的所有资产。以下示例信任策略允许访问您在当前区域中的所有资产。

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

配置 Amazon IoT SiteWise 规则操作

Amazon IoT SiteWise 规则操作将启动规则的 MQTT 消息中的数据发送到中的资产属性。 Amazon IoT SiteWise您可以同时将多个数据条目上传到不同的资产属性，以便在一条消息中发送设备的所有传感器的更新。您还可以为每个数据条目一次上传多个数据点。

> **Note**
>
> 当您 Amazon IoT SiteWise 使用规则操作向发送数据时，您的数据必须满足`BatchPutAssetPropertyValue`操作的所有要求。例如，您的数据的时间戳不能早于当前 Unix 纪元时间 7 天。有关更多信息，请参阅使用 Amazon IoT SiteWise API 提取数据。

对于规则操作中的每个数据条目，您可以标识资产属性，并为该资产属性指定每个数据点的时间戳、质量和值。规则操作要求所有参数均为字符串。

要在条目中标识资产属性，可以指定以下项之一：

- 资产 ID (`assetId`) 和 属性 ID (`propertyId`)，这是您要将数据发送到的资产属性的值。您可以使用 Amazon IoT SiteWise 控制台查找资产 ID 和属性 ID。如果您知道资产 ID，则可以使用 to call Amazon CLI DescribeAsset来查找资产 ID。

- 属性别名 (`propertyAlias`)，这是一个数据流别名（例如，`/company/windfarm/3/turbine/7/temperature`）。要使用此选项，您必须首先设置资产属性的别名。要了解如何设置属性别名，请参阅管理数据流 Amazon IoT SiteWise。

对于每个条目中的时间戳，使用设备报告的时间戳或 Amazon IoT Core提供的时间戳。时间戳有两个参数：

- 时间（以秒为单位）(`timeInSeconds`) – 传感器或设备报告数据的 Unix 纪元时间（以秒为单位）。
- 偏移量（以纳秒为单位）(`offsetInNanos`) –（可选）以秒为单位的时间的纳秒偏移量。

> **Important**
>
> 如果您的时间戳是一个字符串、包含小数部分或不是以秒为单位，则 Amazon IoT SiteWise 会拒绝该请求。您必须将时间戳转换为秒和纳秒偏移量。使用 Amazon IoT 规则引擎的功能来转换时间戳。有关更多信息，请参阅下列内容：
>
> - 获取未报告准确时间的设备的时间戳
> - 转换字符串格式的时间戳

在执行计算、调用函数以及从消息负载中提取值的操作中，您可以为多个参数使用替换模板。有关更多信息，请参阅 Amazon IoT 开发人员指南中的替换模板。

> **Note**
>
> 由于替换模板中的表达式与 `SELECT` 语句分开评估，因此您不能使用替换模板来引用使用 `AS` 子句创建的别名。除了支持的函数和运算符之外，您只能引用原始负载中存在的信息。

主题

- 获取未报告准确时间的设备的时间戳
- 转换字符串格式的时间戳
- 转换纳秒精度的时间戳字符串
- 示例规则配置
- 规则操作问题排查

获取未报告准确时间的设备的时间戳

如果您的传感器或设备未报告准确的时间数据，请使用 time stamp () 从 Amazon IoT 规则引擎获取当前 Unix 纪元时间。此函数以毫秒为单位输出时间，因此您必须将值转换为以秒为单位的时间，并将偏移量转换为以纳秒为单位。为此，请使用以下转换：

- 对于 时间 (以秒为单位) (`timeInSeconds`)，使用 **`${floor(timestamp() / 1E3)}`** 将时间从毫秒转换为秒。
- 对于 偏移量 (以纳秒为单位) (`offsetInNanos`)，使用 **`${(timestamp() % 1E3) * 1E6}`** 计算时间戳的纳秒偏移量。

转换字符串格式的时间戳

如果您的传感器或设备以字符串格式报告时间数据（例如，`2020-03-03T14:57:14.699Z`），请使用 time_to_epoch(String, String)。此函数将时间戳和格式模式作为参数输入，并以毫秒为单位输出时间。然后，您必须将时间转换为以秒为单位的时间和以纳秒为单位的偏移量。为此，请使用以下转换：

- 对于时间（以秒为单位）(`timeInSeconds`)，使用 **`${floor(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z'") / 1E3)}`** 将时间戳字符串转换为毫秒，然后转换为秒。
- 对于偏移量（以纳秒为单位）(`offsetInNanos`)，使用 **`${(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z'") % 1E3) * 1E6}`** 计算时间戳字符串的纳秒偏移量。

> **Note**
>
> `time_to_epoch` 函数支持高达毫秒精度的时间戳字符串。要转换精度为微秒或纳秒的字符串，请配置规则调 Amazon Lambda 用的函数以将时间戳转换为数值。有关更多信息，请参阅转换纳秒精度的时间戳字符串。

转换纳秒精度的时间戳字符串

如果您的设备以精度为纳秒的字符串格式发送时间戳信息（例如 `2020-03-03T14:57:14.699728491Z`），请使用以下过程配置规则操作。你可以创建一个 Amazon Lambda 函数，将时间戳从字符串转换为以秒为单位的时间 (`timeInSeconds`) 和以 nanos 为单位的

偏移量 (`offsetInNanos`)。然后，在规则操作参数中使用 aws_lambda(functionArn, inputJson)，在规则中调用该 Lambda 函数并使用输出。

> **Note**
>
> 本节包含高级说明（假设您已熟悉如何创建以下资源）：
>
> - Lambda 函数。有关更多信息，请参阅《Amazon Lambda 开发人员指南》中的创建第一个 Lambda 函数。
> - Amazon IoT 带有规则操作的 Amazon IoT SiteWise 规则。有关更多信息，请参阅 Amazon IoT SiteWise 使用 Amazon IoT Core 规则提取数据。

创建解析时间戳字符串的 Amazon IoT SiteWise 规则操作

1. 创建具有以下属性的 Lambda 函数：

   - 函数名称 – 使用描述性函数名称（例如 **`ConvertNanosecondTimestampFromString`**）。
   - 运行时 – 使用 Python 3 运行时，如 Python 3.11（`python3.11`）。
   - 权限-创建具有基本 Lambda 权限的角色 () Amazon LambdaBasicExecutionRole。
   - 图层 — 添加 Amazon SDKPandas-Python311 层以供 Lambda 函数使用。numpy
   - 函数代码 – 使用以下函数代码，该函数使用名为 `timestamp` 的字符串参数，并为该时间戳输出 `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 函数使用来自 datetime64 输入 ISO 8601 格式的时间戳字符串。 NumPy

> **Note**
>
> 如果您的时间戳字符串不是 ISO 8601 格式，则可以使用以下方法实现解决方案 pandas 它定义了时间戳格式。有关更多信息，请参阅 pandas.to_datetime。

2. 为规则配置 Amazon IoT SiteWise 操作时，请使用以下替换模板来替换时间（以秒为单位`timeInSeconds`）和以纳为单位的偏移量（`offsetInNanos`）。这些替换模板假定您的消息负载包含 `timestamp` 格式的时间戳字符串。`aws_lambda` 函数为其第二个参数使用 JSON 结构，因此您可以根据需要修改以下替换模板。

   - 对于 时间 (以秒为单位) (`timeInSeconds`)，请使用以下替换模板。

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

   - 对于以 偏移量 (以纳秒为单位) (`offsetInNanos`)，请使用以下替换模板。

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

   对于每个参数，*`account-id`*使用您的地区*`region`*和 Amazon 账户 ID 替换和。如果您为 Lambda 函数使用了不同的名称，请相应更改。

3. 授予使用该 Amazon IoT 权限调用您的函数的`lambda:InvokeFunction`权限。有关更多信息，请参阅 aws_lambda(functionArn, inputJson)。

4. 测试您的规则（例如，使用 Amazon IoT MQTT 测试客户端），并验证它 Amazon IoT SiteWise 接收了您发送的数据。

   如果您的规则未按预期工作，请参阅 对 Amazon IoT SiteWise 规则操作进行故障排除。

> **Note**
>
> 此解决方案为每个时间戳字符串调用 Lambda 函数两次。如果您的规则处理各个负载中具有相同时间戳的多个数据点，则可以创建另一个规则来减少 Lambda 函数的调用次数。
> 为此，请使用重新发布操作创建规则，该操作会调用 Lambda 并发布原始负载，其中时间戳字符串转换为 `timeInSeconds` 和 `offsetInNanos`。然后，使用规则操作创建 Amazon IoT SiteWise 规则以使用转换后的有效负载。使用这种方法，您可以减少规则调用 Lambda 的次数，但可以增加规则操作的 Amazon IoT 运行次数。如果将此解决方案应用于您的使用案例，请考虑每项服务的定价。

示例规则配置

本节包含使用 Amazon IoT SiteWise 操作创建规则的示例规则配置。

Example 使用属性别名作为消息主题的示例规则操作

以下示例创建了一个规则，其 Amazon IoT SiteWise 操作使用主题（通过 topi c ()）作为属性别名来标识资产属性。使用此示例定义一条规则，用于将双类型数据摄取到所有风电场中的所有风力涡轮机。此示例要求您为所有涡轮机资产的属性定义属性别名。您需要定义另一个类似的规则来摄取整数型数据。

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

`sitewise-rule-payload.json` 中的示例负载包含以下内容。

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

通过此规则操作，将以下消息作为主题发送到风力涡轮机属性别名（例如 `/company/windfarm/3/turbine/7/temperature`），用于摄取数据。

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

Example 使用 timestamp() 确定时间的示例规则操作

以下示例创建了一个规则，其 Amazon IoT SiteWise 操作通过标识资产属性 IDs 并使用 timestamp () 来确定当前时间。

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

`sitewise-rule-payload.json` 中的示例负载包含以下内容。

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

使用此规则操作，您可以向 `my/asset/property/topic` 发送以下消息以摄取数据。

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

规则操作问题排查

要对中的 Amazon IoT SiteWise 规则操作进行故障排除 Amazon IoT Core，请为您的规则配置 CloudWatch 日志或配置重新发布错误操作。有关更多信息，请参阅 对 Amazon IoT SiteWise 规则操作进行故障排除。

使用 Basic Ingest in 降低成本 Amazon IoT SiteWise

Amazon IoT Core 提供了一项名为 Basic Ingest 的功能，您可以使用该功能发送数据， Amazon IoT Core 而不会产生Amazon IoT 消息传递成本。基本提取功能通过从提取路径中删除发布/订阅消息代

理来优化大量数据提取工作负载的数据流。如果您知道消息应该路由到哪些规则，则可以使用“基本提取”。

要使用基本提取，您可以使用特殊主题 `$aws/rules/rule-name`，直接将消息发送到特定规则。例如，要向名为 `SiteWiseWindFarmRule` 的规则发送消息，您可以向主题 `$aws/rules/SiteWiseWindFarmRule` 发送消息。

如果您的规则操作使用包含 topic(Decimal) 的替换模板，则可以在“基本提取”特殊主题的末尾传递原始主题，例如 `$aws/rules/rule-name/original-topic`。例如，要将基本提取与之前部分中的风电场属性别名示例结合使用，您可以向以下主题发送消息。

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

> **Note**
>
> 上面的示例包含第二个斜杠 (//)，因为从规则操作可见的主题中 Amazon IoT 删除了 Basic Ingest 前缀 (`$aws/rules/rule-name/`)。在此示例中，规则接收主题 `/company/windfarm/3/turbine/7/temperature`。

有关更多信息，请参阅 Amazon IoT 开发人员指南中的借助基本摄取功能降低消息收发成本。

从中提取数据 Amazon IoT SiteWiseAmazon IoT Events

借 Amazon IoT Events助，您可以在 Amazon 云端为物联网队列构建复杂的事件监控应用程序。使用中的 IoT SiteWise 操作在事件发生 Amazon IoT SiteWise 时将数据发送 Amazon IoT Events 到资产属性。

Amazon IoT Events 旨在简化 Amazon 云端物联网设备和系统的事件监控应用程序的开发。使用 Amazon IoT Events，您可以：

- 检测 IoT 实例集中的变化、异常或特定情况并作出响应。
- 提高运营效率并实现对 IoT 生态系统的主动管理。

Amazon IoT SiteWise 通过与 Amazon IoT SiteWise 操作集成， Amazon IoT Events 扩展了其功能，允许您自动更新资产属性 Amazon IoT SiteWise 以响应特定事件。这种交互可以简化数据摄取和管理，还可以为您提供切实可行的见解。

有关更多信息，请参阅 Amazon IoT Events 开发人员指南中的以下主题：

- 什么是 Amazon IoT Events？
- Amazon IoT Events 操作
- 物联网 SiteWise 行动

在中使用 Amazon IoT Greengrass 直播管理器 Amazon IoT SiteWise

Amazon IoT Greengrass 流管理器是一项集成功能，可便于将数据流从本地源传输到 Amazon 云端。它充当管理数据流的中间层，使在边缘运行的设备能够在数据发送到之前收集和存储数据 Amazon IoT SiteWise，以进行进一步的分析和处理。

通过在 Amazon IoT SiteWise 控制台上配置本地源来添加数据目标。您还可以在自定义 Amazon IoT Greengrass 解决方案中使用流管理器将数据采集到。 Amazon IoT SiteWise

有关如何在自定义 Amazon IoT Greengrass 解决方案中使用流管理器摄取数据的更多信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的以下主题：

- 什么是 Amazon IoT Greengrass？
- 在 Amazon IoT Greengrass Core 上管理数据流
- 将数据导出到 Amazon IoT SiteWise 资产属性

使用 Amazon IoT SiteWise 边缘网关

Amazon IoT SiteWise Edge 将云功能扩展到工业边缘环境，支持本地数据处理、分析和决策。SiteWise Edge 与其他 Amazon 服务集成，提供全面的工业物联网解决方案。 Amazon IoT SiteWise 网关充当您的工业设备和之间的中介 Amazon IoT SiteWise。

SiteWise Edge 网关在两个不同的部署目标上运行：

- Amazon IoT Greengrass V2
- Siemens Industrial Edge

您可以使用 SiteWise Edge 网关在边缘收集数据并将其发布到云端。对于在上运行的网关 Amazon IoT Greengrass，您还可以使用资产模型和资产在边缘处理数据。

上的 Amazon IoT SiteWise Edge 应用程序Siemens Industrial Edge支持工业设备之间的集成，Amazon IoT SiteWise 因此您可以聚合和处理原始机器数据，并在本地运行分析，然后再将精确的数据发送到 Amazon 云端。

SiteWise 边缘网关的关键概念

SiteWise Edge 为工业环境中的边缘计算提供了多种有用的功能。

本地数据收集和处理

支持使用 OPC-UA 和 MQTT 等协议从工业资产中收集数据。网关在 Amazon IoT Greengrass 核心设备上运行或Siemens Industrial Edge.

离线操作

在互联网中断期间继续收集和处理数据，并在连接恢复后与云端同步。

带 Amazon IoT Greengrass 组件的边缘计算

使用 IoT SiteWise 发布器将数据转发到云端和 Amazon IoT SiteWise 处理器以进行本地转换和计算。发行商和处理器都是 Amazon IoT Greengrass V2 组件。有关 Amazon IoT Greengrass 组件的更多信息，请参阅Amazon提供的组件。

与集成 Amazon IoT SiteWise 以扩展云功能

使用 Amazon IoT SiteWise 云功能，将资产模型、分析和仪表板扩展到边缘。

对于启用了数据处理包的网关，您可以使用 Amazon IoT SiteWise 来集中管理您 Amazon OpsHub 的 SiteWise Edge 网关。 Amazon OpsHub 提供远程管理和监控功能。有关更多信息，请参阅 使用 Amazon OpsHub 管理 SiteWise 边缘网关 Amazon IoT SiteWise。

边缘的本地可视化

提供自定义仪表板，以便在边缘获得实时见解。

使用本地设备上的 M SiteWise onitor 门户在设施中本地监控数据。有关更多信息，请参阅 在边缘启用您的 Amazon IoT SiteWise 门户。

实施 SiteWise Edge 的好处

SiteWise Edge 提供了许多优势，可以显著改善工业运营和决策流程。

- 实时运营洞察力，避免云处理延迟
- 断开连接的环境中的操作连续性
- 通过边缘预处理降低带宽和存储成本
- 能够在本地做出以数据为依据的决策，从而提高可靠性

使用自托管 Amazon IoT SiteWise Edge 网关 Amazon IoT Greengrass V2

将 Amazon IoT SiteWise Edge 设置为在本地收集、处理和可视化来自工业设备的数据，然后再将其发送到云端。使用 Amazon IoT Greengrass Version 2自托管。

边 Amazon IoT SiteWise 缘网关充当工业设备和之间的中介 Amazon IoT SiteWise。 SiteWise Edge 网关在上 Amazon IoT Greengrass Version 2运行，支持本地数据收集和处理。启用并 Amazon OpsHub 安装数据处理包，通过本地设备上的 Monito SiteWise r 门户在设施内本地监控数据。

自托管网关有两种类型：

支持 MQTT 的 V3 网关

支持 MQTT 的 V3 网关架构提供了改进的数据摄取能力。它利用 MQTT 协议实现高效的数据通信，并提供可配置的数据目的地。其中包括使用 Amazon S3 进行缓冲数据摄取的选项，以及实时数据摄取的选项。您可以实现路径过滤器来订阅特定的 MQTT 主题，从而实现有针对性的数据收

集。请注意，支持 MQTT 的 V3 网关不支持数据处理包功能。有关更多信息，请参阅 支持 MQTT 的边缘版 V3 网关 Amazon IoT SiteWise。

经典直播，V2 网关

经典直播，V2 网关代表了传统的 Amazon IoT SiteWise Edge 网关架构。它非常适合现有的 SiteWise Edge 部署和习惯于既定工作流程的用户。当 Classic 直播时，V2 网关支持数据处理包，但请注意，数据处理包生成的数据无法通过 Amazon S3 提取。如果您需要保持与现有部署的兼容性或者需要数据处理包功能，请使用 Classic streams、V2 网关。有关更多信息，请参阅 经典直播，适用 Amazon IoT SiteWise 于 Edge 的 V2 网关。

主题

- Amazon IoT SiteWise Edge 自托管网关要求
- 创建自托管 SiteWise Edge 网关
- 在本地设备上安装 Amazon IoT SiteWise Edge 网关软件
- 支持 MQTT 的边缘版 V3 网关 Amazon IoT SiteWise
- 经典直播，适用 Amazon IoT SiteWise 于 Edge 的 V2 网关
- 将数据源添加到您的 Amazon IoT SiteWise Edge 网关
- Amazon IoT GreengrassAmazon IoT SiteWise Edge 的组件
- 筛选 SiteWise Edge 网关上的资产
- 为 E Amazon IoT SiteWise dge 配置代理支持并管理信任存储
- Amazon IoT SiteWise APIs 在边缘使用

Amazon IoT SiteWise Edge 自托管网关要求

Amazon IoT SiteWise 边缘网关 Amazon IoT Greengrass V2 作为一组 Amazon IoT Greengrass 组件运行，支持本地数据收集、处理和发布。要配置在上运行的 SiteWise Edge 网关 Amazon IoT Greengrass V2，请在中创建网关， Amazon Web Services 云 然后运行 SiteWise Edge 网关软件来设置您的本地设备。使用创建 SiteWise Edge 网关时，会提供安装脚本。 Amazon Web Services Management Console 在目标网关设备上运行此脚本以设置必要的软件和依赖关系。

本地设备要求

本地设备必须满足以下要求才能安装和运行 SiteWise Edge 网关软件。

- 支持 Amazon IoT Greengrass V2 核心软件版本 v2.3.0 或更高版本。有关更多信息，请参阅 Amazon IoT Greengrass Version 2 开发人员指南中的要求。
- 以下支持平台之一：
  - OS：Ubuntu 20.04 或更高版本

    架构：x86_64 (AMD64) 或 ARMv8 (Aarch64)
  - OS：Red Hat Enterprise Linux (RHEL) 8

    架构：x86_64 (AMD64) 或 ARMv8 (Aarch64)
  - OS：Amazon Linux 2

    架构：x86_64 (AMD64) 或 ARMv8 (Aarch64)
  - OS：Debian 11

    架构：x86_64 (AMD64) 或 ARMv8 (Aarch64)
  - OS：Windows Server 2019 及更高版本

    架构：x86_64 () AMD64

> **Note**
>
> ARM 平台仅支持带有数据收集包的 SiteWise Edge 网关。不支持数据处理包。

- 内存最低为 4 GB。
- SiteWise Edge 网关软件至少可用 10 GB 的磁盘空间。
- 配置您的本地设备以确保可以访问正确的端口。有关所需出站服务终端节点的完整列表，请参阅 Amazon IoT SiteWise Edge 网关所需的服务终端节点。
- Java 运行时环境（JRE）版本 11 或更高版本。Java 必须在设备的 PATH 环境变量中可用。要使用 Java 开发自定义组件，您必须安装 Java 开发工具包 (JDK)。我们建议您使用 Amazon Corretto 或 OpenJDK。

允许列入本地设备许可名单的 Amazon S3 存储桶

配置您的本地设备以提供对以下 Amazon S3 存储桶的防火墙访问权限。根据设备的相应区域配置访问权限。

区域	终端节点
亚太地区（东京）	https://iot-sitewise-gateway-ap-northeast-1-785558802005.s3.ap-northeast-1.amazonaws.com
亚太地区（首尔）	https://iot-sitewise-gateway-ap-northeast-2-310055672453.s3.ap-northeast-2.amazonaws.com
亚太地区（孟买）	https://iot-sitewise-gateway-ap-south-1-677656657204.s3.ap-south-1.amazonaws.com
亚太地区（新加坡）	https://iot-sitewise-gateway-ap-southeast-1-475191558554.s3.ap-southeast-1.amazonaws.com
亚太地区（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
中国（北京）	https://iot-sitewise-gateway-cn-north-1-237124890262.s3.cn-north-1.amazonaws.com.cn
欧洲地区（法兰克福）	https://iot-sitewise-gateway-eu-central-1-748875242063.s3.eu-central-1.amazonaws.com
欧洲地区（爱尔兰）	https://iot-sitewise-gateway-eu-west-1-383414315062.s3.eu-west-1.amazonaws.com
美国东部（弗吉尼亚州北部）	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/
美国东部（俄亥俄州）	https://iot-sitewise-gateway-us-east-2-005072661813.s3.us-east-2.amazonaws.com
Amazon GovCloud （美国西部）	https://iot-sitewise-gateway-us-gov-west-1-599984565679.s3.us-gov-west-1.amazonaws.com/

区域	终端节点
美国西部（俄勒冈州）	https://iot-sitewise-gateway-us-west-2-502577205460.s3.us-west-2.amazonaws.com

数据处理包要求

- 如果您计划在边缘使用数据处理包 Amazon IoT SiteWise，则您的本地设备还必须满足以下要求：
  - 配备 x86 64 位四核处理器。
  - 至少有 16 GB 内存。
  - 如果使用，则至少有 32 GB 的内存空间Microsoft Windows。
  - 至少有 256 GB 可用磁盘空间。
  - 本地设备必须支持端口 443 的网络入站流量。
  - 以下端口保留供以下用户使用 Amazon IoT SiteWise：
    80、443、3001、4569、4572、8000、8081、8082、8084、8085、8445、8085、8445、8086、9000、和 50010。使用预留端口通信可能导致连接终止。

    > **Note**
    > Amazon IoT Greengrass V2 直播管理器组件有自己的要求。有关更多信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的 Configuration。

- 磁盘空间和计算容量最低要求取决于您的实施和用例所特有的各项因素。
  - 缓存间歇性 Internet 连接数据时所需的磁盘空间取决于以下因素：
    - 上传的数据流数
    - 每秒每个数据流的数据点数
    - 每个数据点的大小
    - 通信速度
    - 预期网络中断
  - 轮询并上传数据时所需的计算容量取决于以下因素：
    - 上传的数据流数
    - 每秒每个数据流的数据点数

配置使用 SiteWise Edge 网关的权限

您必须具有以下权限才能使用 SiteWise Edge 网关：

> **Note**
>
> 如果您使用 Amazon IoT SiteWise 控制台创建 SiteWise Edge 网关，则会为您添加这些权限。

- 您的 SiteWise Edge 网关的 IAM 角色必须允许您在 Amazon IoT Greengrass V2 设备上使用 SiteWise 边缘网关来处理资产模型数据和资产数据。

  该角色允许以下服务来代入角色：`credentials.iot.amazonaws.com`。

  权限详细信息

  该角色必须具有以下权限：

  - `iotsitewise` — 允许主体在边缘检索资产模型数据和资产数据。
  - `iot`— 允许您的 Amazon IoT Greengrass V2 设备与之交互 Amazon IoT。
  - `logs`— 允许您的 Amazon IoT Greengrass V2 设备向 Amazon Logs 发送 CloudWatch 日志。
  - `s3`— 允许您的 Amazon IoT Greengrass V2 设备从 Amazon S3 下载自定义组件工件。

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

创建自托管 SiteWise Edge 网关

使用 Amazon IoT SiteWise 控制台或 Amazon CLI 创建自托管 SiteWise Edge 网关。此过程详细介绍了如何创建将安装在自己的硬件上的自托管 SiteWise Edge 网关。有关创建在 Siemens In SiteWise dustrial Edge 上运行的 Edge 网关的信息，请参阅在上托管 SiteWise Edge 网关 Siemens Industrial Edge。

创建 SiteWise Edge 网关

Console

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择创建网关。
4. 对于选择部署目标，请选择自托管网关。
5. 选择启用 MQTT 的 V3 网关或经典直播、 V 2 网关。有关每个选项的更多信息，请参阅使用自托管 Amazon IoT SiteWise Edge 网关 Amazon IoT Greengrass V2。建议使用支持 MQTT 的 V3 网关，因为它具有面向未来的功能。
6. 在网关配置部分，输入 SiteWise Edge 网关的名称或使用生成的名称 Amazon IoT SiteWise。
7. 在 Greengrass 设备操作系统下，选择要安装此 Edge 网关的设备的操作系统。 SiteWise

> **Note**
> 数据处理包仅在 x86 平台上可用。它仅在 Classic 直播、V2 网关上可用

8. （可选）要在边缘处理和组织数据，请在边缘功能下选择数据处理包。
9. （可选）在高级配置下，执行以下操作：
   - 对于 Greengrass 核心设备，请在以下选项中选择一项：
     - 默认设置- Amazon 自动使用默认设置在中创建 Greengrass 核心设备。 Amazon IoT Greengrass V2
       1. 输入 Greengrass 核心设备的名称或使用生成的名称。 Amazon IoT SiteWise
     - 高级设置 — 如果您想使用现有 Greengrass 核心设备或手动创建一个核心设备，请选择此项。
       1. 选择 Greengrass 核心设备或选择 “创建 Greengrass 核心设备” 在主机中创建一个。 Amazon IoT Greengrass V2 有关更多信息，请参阅 Amazon IoT Greengrass V2 开发人员指南中的设置Amazon IoT Greengrass Version 2 核心设备。
10. 选择创建网关。
11. 在 “生成 SiteWise Edge 网关安装程序” 对话框中，选择 “生成并下载”。 Amazon IoT SiteWise 会自动生成一个安装程序，您可以使用该安装程序来配置本地设备。

> **Important**
> 您无法重新生成此文件。请务必将安装程序文件保存在安全位置，因为您稍后会用到该文件。

Amazon CLI

要使用创建自托管网关 Amazon CLI，请提供网关名称、指定平台和网关版本。在创建网关时，您可以指定许多其他选项。有关更多信息，请参阅《Amazon CLI 命令参考》中的 create-gatew ay Amazon IoT SiteWise

要使用此示例，请将用户输入占位符 替换为您自己的信息。

```
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`— 网关的唯一名称。
- `gateway-platform`— 指定网关平台配置。对于自托管网关，请输入`greengrassV2`。有关更多信息，请参阅《Amazon CLI 命令参考》中 create-gateway 部分中的选项。 Amazon IoT SiteWise
  - `coreDeviceThingName`— Amazon IoT Greengrass V2 核心设备的物 Amazon 联网设备名称。
  - `coreDeviceOperatingSystem`— 中核心设备的操作系统 Amazon IoT Greengrass V2。必须指定操作系统`gateway-version 3`，但不适用于`gateway-version 2`。选项包括：`LINUX_AARCH64``LINUX_AMD64`、和 `WINDOWS_AMD64`
- `gateway-version`— 网关的版本。
  - 要创建支持 MQTT 的 V3 网关，请使用网关3版本。
  - 要创建经典流，即 V2 网关，请使用2网关版本。
- `cli-input-json`— 包含请求参数的 JSON 文件。

在本地设备上安装 Amazon IoT SiteWise Edge 网关软件

创建 Amazon IoT SiteWise Edge 网关后，在本地设备上安装 SiteWise Edge 网关软件。 SiteWise 边缘网关软件可以安装在安装了 Linux 或Microsoft Windows服务器操作系统的本地设备上。

> **Important**
>
> 确保您的本地设备已连接到 Internet。

Linux

以下过程使用 SSH 连接到您的本地设备。或者，您可以使用 USB 闪存驱动器或其他工具将安装程序文件传输到本地设备。如果您不想使用 SSH，请跳至下面的步骤 2：安装 SiteWise Edge 网关软件。

SSH 先决条件

在使用 SSH 连接到您的设备之前，请先完成以下先决条件。

- Linux 和 macOS – 下载并安装 OpenSSH。有关更多信息，请参阅 https://www.openssh.com。

步骤 1：将安装程序复制到您的 SiteWise Edge 网关设备

以下说明介绍了如何使用 SSH 客户端连接到您的本地设备。

1. 要连接到您的设备，请在计算机的终端窗口中运行以下命令，将*username*和*IP*替换为具有更高权限和 IP 地址的用户名。

```
ssh username@IP
```

2. 要将 Amazon IoT SiteWise 生成的安装程序文件传输到您的 SiteWise Edge 网关设备，请运行以下命令。

> **Note**
>
> - *path-to-saved-installer*替换为计算机上用于保存安装程序文件的路径和安装程序文件的名称。
> - *IP-address*替换为本地设备的 IP 地址。
> - *directory-to-receive-installer*替换为本地设备上用于接收安装程序文件的路径。

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

步骤 2：安装 SiteWise Edge 网关软件

在以下步骤中，在 SiteWise Edge 网关设备的终端窗口中运行命令。

1. 向安装程序文件授予执行权限。

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

2. 运行安装程序。

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

Windows Server

先决条件

要安装 SiteWise Edge 网关软件，您必须具备以下先决条件：

- Microsoft Windows已安装 2019 或更高版本的服务器
- 管理员权限
- PowerShell 已安装版本 5.1 或更高版本
- SiteWise Edge 网关安装程序已下载到将在其中进行配置的 Windows 服务器

步骤 1：以管理员 PowerShell 身份运行

1. 在要安装 SiteWise 边缘网关的 Windows 服务器上，以管理员身份登录。
2. 在 Windows 搜索栏PowerShell中输入。
3. 在搜索结果中，打开 Windows PowerShell 应用程序上的上下文（右键单击）菜单。选择以管理员身份运行。

步骤 2：安装 SiteWise Edge 网关软件

在 SiteWise Edge Gateway 设备的终端窗口中运行以下命令。

1. 解除封锁 SiteWise Edge 网关安装程序。

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

2. 运行安装程序。

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

> **Note**
>
> 如果在系统上禁用脚本执行，请将脚本执行策略更改为 `RemoteSigned`。

```
Set-ExecutionPolicy RemoteSigned
```

下一步取决于您需要的自托管网关的类型。继续执行支持 MQTT 的边缘版 V3 网关 Amazon IoT SiteWise或经典直播，适用 Amazon IoT SiteWise 于 Edge 的 V2 网关。

支持 MQTT 的边缘版 V3 网关 Amazon IoT SiteWise

Amazon IoT SiteWise 可以使用支持 MQTT 的 V3 网关，这代表了 SiteWise Edge 网关架构的重大进步。这种网关类型利用 MQTT（消息队列遥测传输）协议进行数据通信，为工业物联网部署提供了更高的灵活性和效率。

支持 MQTT 的 V3 网关使用 MQTT 进行数据传输，从而实现了轻量级的发布-订阅网络协议，可在设备和云之间高效传输消息。您可以设置各种数据目标，包括直接将实时数据摄入 Amazon IoT SiteWise 和使用 Amazon S3 进行缓冲数据摄取。要实现精确的数据收集，您可以实现路径过滤器来订阅特定的 MQTT 主题。

支持 MQTT 的 V3 网关带有预先配置的实时目的地，过滤器设置为 “#”（所有主题），您可以根据需要对其进行自定义或删除。为了简化数据管理，每个网关中只能存在一个实时目的地。

支持 MQTT 的架构与经典流 V2 网关有很大不同。虽然 V2 使用基于流的方法，但 V3 采用 MQTT，提供了更多可配置的数据目的地和筛选选项。但是，请注意，V3 不支持数据处理包，该包在 V2 中可用。

支持 MQTT 的 V3 网关具有以下优点：

- 由于 MQTT 的轻量级特性，提高了可扩展性，可以更好地处理大量设备和高频数据传输。
- 通过路径筛选器增强数据控制，实现对数据收集的精细管理，减少不必要的数据传输和处理。
- 灵活的数据处理，允许根据特定需求在实时处理和缓冲存储之间进行配置。
- 与现代物联网通信标准保持一致，为未来的增强和集成奠定基础。

考虑采用支持 MQTT 的 V3 网关进行新部署，尤其是在您需要灵活的数据摄取选项和精确控制数据收集时。

> **Note**
>
> 对于需要数据处理包（Classic Streams）的现有部署或场景，V2 网关仍然是一个可行的选择。

通过提供两种网关类型，无论您优先考虑高级 MQTT 功能还是与现有系统的兼容性，都可以 Amazon IoT SiteWise 确保您可以选择最适合您的特定工业物联网需求的解决方案。

目的地和路径过滤器

查看以下主题，详细了解启用 MQTT 的网关中的目的地和路径过滤器：

- 了解 Amazon IoT SiteWise 边缘目的地
- 添加 Amazon IoT SiteWise Edge 实时目的地
- 使用 Amazon S3 添加 Amazon IoT SiteWise 缓冲目的地
- 了解 Amazon IoT SiteWise Edge 目标的路径过滤器
- 向 Amazon IoT SiteWise Edge 目标添加路径过滤器
- 管理 Amazon IoT SiteWise 边缘目的地

将外部应用程序连接到 EMQX 代理

本指南介绍如何通过已部署的支持 MQTT 的 V Amazon IoT SiteWise 3 网关上的 EMQX 代理将外部应用程序连接到 Edge 网关。外部应用程序可能包括自定义监控工具、第三方可视化软件或需要在边缘与工业数据交互的传统系统。

我们将介绍 Linux 和Microsoft Windows环境的配置步骤，包括 EMQX 部署配置、用于安全连接的 TLS 设置以及用于控制对特定主题的访问的授权规则。

> **Note**
>
> EMQX 不是 Edge 的供应商或供应商。 Amazon IoT SiteWise

> **Important**
>
> 为了保护与网关的连接，我们强烈建议通过 Amazon IoT Greengrass 客户端设备身份验证功能使用基于证书的身份验证。此方法通过双向 TLS (mTLS) 身份验证提供强大的安全性。有关更

多信息，请参阅《Amazon IoT Greengrass Version 2 开发者指南》中的 Connect 客户端设备与核心设备。

如果您无法使用基于证书的身份验证，请按照本指南使用用户名和密码设置身份验证。

先决条件

- 支持 SiteWise Edge MQTT 的 V3 网关，该网关已部署并处于联机状态
- 访问网关主机
- 访问 Amazon IoT SiteWise 和 Amazon IoT Greengrass 控制台

主题

- Edge 上的 EMQX 代理的消息有效载荷格式 Amazon IoT SiteWise
- 配置 EMQX 代理
- 将应用程序连接到 Edge 上的 EMQX 代理 Amazon IoT SiteWise
- 在 EMQX 中为 Amazon IoT SiteWise Edge 设置授权规则

Edge 上的 EMQX 代理的消息有效载荷格式 Amazon IoT SiteWise

要让 IoT SiteWise 发布者组件使用来自外部应用程序的数据并将其发布到 Amazon IoT SiteWise 云端，发送到代理的有效负载必须满足特定要求。

了解有效载荷格式是 MQTT 与 Amazon IoT SiteWise Edge 成功通信的关键。虽然连接设置过程将在后面的章节中介绍，但我们将首先介绍有效负载要求，以帮助您规划实施。

MQTT 主题要求

对 MQTT 主题结构没有限制，包括使用的关卡数量或字符数。但是，我们建议主题与有效载荷中的`propertyAlias`字段相匹配。

Example 属性别名示例

如果 MQTT 主题是`site1/line1/compressor1/temperature`，请确保`propertyAlias`匹配。

```
{
  "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 负载结构

MQTT 消息负载以 JSON 编写，并遵循 Amazon IoT SiteWise API 参考中定义的`PutAssetPropertyValueEntry`消息格式。

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

> **Note**
>
> 要使消息被视为有效，只能满足以下条件之一：

- `propertyAlias`已设置，或者
- `assetId`和`propertyId`都设置好了

`PutAssetPropertyValueEntry`有一个在此上下文中不是必填的`entryId`字段。

配置 EMQX 代理

本节介绍如何添加用户名和密码。它还介绍了如何使用添加的用户名和密码从外部来源建立 TLS 连接。您可以使用 Linux 来配置 EMQX 代理，也可以使用。Microsoft Windows

> **Note**
>
> 要配置代理，您需要在支持 MQTT 的 V3 网关中使用默认 EMQX 配置设置的核心设备。

> **Important**
>
> 完成此过程后，我们强烈建议您配置授权规则。有关更多信息，请参阅 在 EMQX 中为 Amazon IoT SiteWise Edge 设置授权规则。针对已添加用户的授权规则可增强安全性。

更新 EMQX 部署配置以进行身份验证

更新 EMQX 部署配置以进行身份验证

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择要配置的网关。
4. 在 Edge 网关配置部分，复制您的 Greengrass 核心设备值。将其保存以备后用。
5. 打开 Amazon IoT 管理控制台。
6. 在左侧导航栏的 “管理” 部分下，选择 Greengrass 设备，然后选择 “部署”。
7. 找到您之前保存的核心设备值，然后选择该链接以打开部署。
8. 选择 “操作” 下拉按钮，然后选择 “修改”。
9. 阅读显示的消息，然后选择修订部署。将出现 “指定目标” 页面。

10. 选择 “下一步”，直到进入 “配置组件” 步骤。
11. 选择 `aws.greengrass.clientdevices.mqtt.EMQX` 单选按钮。
12. 选择 “配置组件” 按钮。此时将显示该组件的配置页面。
13. 在 “配置更新” 下，选择 “重置为组件版本的默认配置：2.*”。 *。
14. 根据您的操作系统，在 “要合并的配置” 部分中输入以下配置。

    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
 -p 127.0.0.1:18083:18083 -v emqx-data:/opt/emqx/data -e
 EMQX_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"
}
```

该`dockerOptions`字段仅适用于 Linux 网关。

15. 选择确认。
16. 选择 “下一步”，直到进入 “查看” 步骤。
17. 选择部署。
18. 部署成功后，继续执行下一步。

启用用户名和密码认证

本节介绍如何通过 EMQX 控制面板 GUI 添加用户名和密码。

> **Note**
>
> 所提供的 EMQX 相关说明仅供参考。由于 EMQX 的文档和功能可能会随着时间的推移而发生变化，并且我们不保留其文档，因此我们建议您查阅 EMQX 的官方文档以获取最新信息。

EMQX Dashboard

通过 EMQX 控制面板启用用户名和密码身份验证

1. 确保您在网关主机内。
2. 打开浏览器窗口并访问http://localhost:18083/。
3. 输入的默认用户名**admin**和默认的密码**public**。有关更多信息，请参阅 EMQX 文档中的 EMQX 控制面板。
4. 登录后，系统会提示您更改密码。更新您的密码以继续进入 EMQX 控制面板。
5. 在左侧导航栏中，选择盾牌图标，然后选择身份验证。
6. 在 “内置数据库” 行中，选择 “用户” 按钮。
7. 选择加号图标按钮以添加用户。将出现 “添加” 屏幕。
8. 输入外部应用程序用户的用户名和密码。
9. 选择保存。您选择的用户名将显示在 “身份验证” 页面的表格中。

> **Note**
>
> 现有或默认的授权规则适用于新用户。建议根据您的外部应用程序需求对其进行审查和调整。

EMQX REST API with Linux

通过 shell 环境和 EMQX 代理添加用户名和密码 REST APIs

1. 确保已安装./jq命令行处理器工具。如果不是，请安装它。有关更多信息，请参阅中的下载 jq。 /jq 文档。
2. 使用以下命令更新默认public密码，**new_admin_password**替换为您为经纪商管理员用户选择的密码。

```
curl -s -X 'POST' 'http://localhost:18083/api/v5/login' -H'accept: application/
json' \
-H 'Content-Type: application/json' \
 -d '{"username": "admin", "password": "public" }' \
| jq -r '.token' \
| xargs -I {} curl -s -w '%{http_code}' -X 'POST' 'http://localhost:18083/api/
v5/users/admin/change_pwd' \
```

```
     -H 'Authorization: Bearer {}' \
     -H 'Content-Type: application/json' \
     -d '{
           "old_pwd": "public",
           "new_pwd": "new_admin_password"
         }' \
| grep -q '^2[0-9][0-9]$' && echo "Admin password changed successfully" || echo
 "Failed to change admin password"
```

如果密码更改成功，则会显示以下消息：

```
Admin password changed successfully
```

3. 为您的外部应用程序添加用户名和密码。

   - **username_to_add**替换为要在外部应用程序上使用的用户名。
   - 替换**password_to_add**要用于用户名的密码。
   - **new_admin_password**替换为您更新的仪表板管理员密码。

```
curl -s -X 'POST' 'http://localhost:18083/api/v5/login' -H'accept: application/
json' \
-H 'Content-Type: application/json' \
 -d '{"username": "admin", "password": "new_admin_password" }' \
| jq -r '.token' \
| xargs -I {} curl -s -X 'POST' \
'http://localhost:18083/api/v5/authentication/password_based
%3Abuilt_in_database/users' \
-H 'accept: application/json' \
-H 'Authorization: Bearer {}' -H 'Content-Type: application/json' \
-d '{ "password": "password_to_add", "user_id": "username_to_add"}' \
| jq '.'
```

如果更新成功，则会显示类似以下内容的消息：

```
{
  "is_superuser": false,
  "user_id": "your-added-username"
}
```

如果命令失败，请确保使用在上一步中更新的正确仪表板管理员密码。

EMQX REST API with Windows

要添加用户名和密码，请通过 EMQ PowerShell X 代理 REST APIs

1. 使用以下命令更新默认public密码，**new-admin-password**替换为您为经纪商管理员用户选择的密码。

```
try {
    $loginResponse = Invoke-RestMethod -Uri 'http://localhost:18083/api/v5/
login' -Method Post -ContentType 'application/json' -Body '{"username": "admin",
 "password": "public"}'
    $token = $loginResponse.token

    $changePwdResponse = Invoke-RestMethod -Uri 'http://localhost:18083/api/v5/
users/admin/change_pwd' -Method Post -ContentType 'application/json' -Headers
 @{"Authorization" = "Bearer $token"} -Body '{"old_pwd": "public", "new_pwd":
 "new-admin-password"}'

    Write-Output "Admin password changed successfully"
} catch {
    Write-Output "Failed to change admin password: $_"
}
```

如果密码更改成功，则会显示以下消息：

```
Admin password changed successfully
```

2. 为您的外部应用程序添加用户名和密码。

   - **username-to-add**替换为要在外部应用程序上使用的用户名。
   - 替换**password-to-add**要用于用户名的密码。
   - **new-admin-password**替换为您更新的仪表板管理员密码。

```
try {
    $loginResponse = Invoke-RestMethod -Uri 'http://localhost:18083/api/v5/
login' -Method Post -ContentType 'application/json' -Body '{"username": "admin",
 "password": "new-admin-password"}'
    $token = $loginResponse.token
```

```
    $addUserResponse = Invoke-RestMethod -Uri 'http://localhost:18083/api/
v5/authentication/password_based%3Abuilt_in_database/users' -Method Post -
ContentType 'application/json' -Headers @{"Authorization" = "Bearer $token"} -
Body '{"password": "password-to-add", "user_id": "username-to-add"}'

    $addUserResponse | ConvertTo-Json
} catch {
    Write-Output "Failed to add user: $_"
}
```

如果更新成功，则会显示类似以下内容的消息：

```
{
  "is_superuser": false,
  "user_id": "your-added-username"
}
```

如果命令失败，请确保使用在上一步中更新的正确仪表板管理员密码。

将应用程序连接到 Edge 上的 EMQX 代理 Amazon IoT SiteWise

EMQX 代理在端口 8883 上使用传输层安全 (TLS) 来加密所有通信，确保您的数据在传输过程中受到保护。本节将引导您完成在应用程序和 EMQX 代理之间建立连接的步骤。遵循这些步骤有助于维护工业数据的完整性和机密性。连接过程包括两种主要方法：通过组件使用自动 IP 发现，或者在 TLS 证书中手动将 DNS 名称和 IP 地址配置为主题备用名称 (SANs)。每种方法都有其自身的优势，具体取决于您的网络设置和安全要求。本文档将指导您完成这两个选项。

主题

- 配置 TLS 以实现与 Edge 上的 EMQX 代理的安全连接 Amazon IoT SiteWise
- 在 Edge 上测试 EMQX 代理连接 Amazon IoT SiteWise
- 使用你自己的 CA
- 为外部防火墙连接打开 8883 端口

配置 TLS 以实现与 Edge 上的 EMQX 代理的安全连接 Amazon IoT SiteWise

默认情况下，为 EMQX 代理 Amazon IoT Greengrass 生成由核心设备证书颁发机构 (CA) 签名的 TLS 服务器证书。有关更多信息，请参阅使用 MQTT 代理将客户端设备连接到 Amazon IoT Greengrass 核心设备。

检索 TLS 证书

要获取 CA 证书，请在网关主机上运行以下命令：

Linux

在网关主机上的 shell 会话中运行以下命令：

```
cat /greengrass/v2/work/aws.greengrass.clientdevices.Auth/ca.pem
```

Windows

在网关主机的 PowerShell 会话中运行以下命令：

```
Get-Content C:\greengrass\v2\work\aws.greengrass.clientdevices.Auth\ca.pem
```

将 ca.pem 文件的内容复制到要连接到代理的外部应用程序中。将其保存为 `BrokerCoreDeviceCA.pem`。

向 TLS 服务器证书添加自定义 DNS 名称/IP 地址

由生成的证书上的主题备用名称 (SAN) Amazon IoT Greengrass 是`localhost`。从网关主机外部建立 TLS 连接时，TLS 验证步骤会失败，因为代理的主机名与服务器证书`localhost`上的主机名不匹配。

为了解决主机名不匹配的问题， Amazon IoT Greengrass 提供了两种管理核心设备端点的方法。本节涵盖这两个选项。有关更多详细信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的管理核心设备端点。

- 要使用核心设备的 IP 地址连接到 EMQX 代理，请使用自动 IP 发现部分。
- 要使用 DNS 名称而不是 IP 地址连接到 EMQX 代理，请使用手动管理部分。

Automated IP discovery

此选项允许您的核心设备自动发现其 IP 地址，并将其作为主题备用名称 (SAN) 添加到代理证书中。

1. 将该`aws.greengrass.clientdevices.IPDetector`组件添加到核心设备的部署中。
2. 将更改部署到您的设备
3. 等待部署完成。

部署完成后，您可以使用代理的 IP 地址建立安全的 TLS 连接。

IP 地址将作为 SAN 自动添加到代理证书中。

Manual DNS and IP Configuration

您可以手动将 DNS 名称和 IP 地址作为主题备用名称 (SANs) 添加到您的 TLS 证书。当您为网关主机配置了 DNS 名称时，此方法很有用。

> **Important**
>
> 如果您正在使用该 IPDetector 组件，请在继续操作之前将其从部署中删除。该 IPDetector 组件会覆盖手动端点配置。

手动配置终端节点

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择要配置的网关。
4. 在 Edge 网关配置部分，选择您的 Greengrass 核心设备网址。将显示核心设备的页面。
5. 选择 “客户端设备” 选项卡。
6. 选择管理端点。
7. 在 “管理终端节点” 对话框中，输入 DNS 名称和要添加的任何 IP 地址 SANs。使用端口 8883。
8. 选择更新。

代理的 TLS 服务器证书会自动更新以包含您的新端点。

使用 Linux 验证 TLS 服务器证书更新

1. 在您的网关主机上启动 shell 会话。

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

2. 该命令返回的输出类似于以下内容：

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

3. 验证您的终端节点是否出现在列表中 SANs。

使用 Windows 验证 TLS 服务器证书的更新

1. 在您的网关主机上启动 shell 会话。

```
(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. 该命令返回的输出类似于以下内容：

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

3. 验证您添加的终端节点是否在列表中 SANs。

在 Edge 上测试 EMQX 代理连接 Amazon IoT SiteWise

使用 TLS 证书和身份验证凭证配置 EMQX 代理后，请务必验证您的设置是否正常运行。测试连接有助于确保您的安全配置得到正确实施，并且客户端可以成功地与代理建立加密连接。本节演示如何使用 Mosquitto 命令行接口 (CLI) 客户端（一种广泛使用的支持 TLS 加密和身份验证的 MQTT 客户端工具）测试您的代理连接。

使用 Mosqitto CLI 客户端测试 EMQX 代理连接

在此步骤中，我们将使用 mosquitto CLI 客户端来测试我们的设置，并确保我们可以使用之前创建的用户名和密码成功连接到代理。要获取`BrokerCoreDeviceCA.pem`步骤 3：设置 TLS 下的以下步骤。

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

> **Note**
>
> 在步骤 3：设置 TLS 以获取具有正确的 SAN 的证书时，如果 TLS 服务器hostname/IP address you are connecting to does not match the Subject Alternative Name (SAN) that is on the CA cert you're passing to the client. See "Adding custom DNS names/IP的地址为 “证书”，则可能会出现 SSL: Verify 错误。

此时，所有用户都可以发布和订阅经纪商上的所有主题。继续执行在 EMQX 中为 Amazon IoT SiteWise Edge 设置授权规则。

使用你自己的 CA

Amazon IoT Greengrass 概述了如何将自己的客户端设备身份验证组件配置为使用自己的证书颁发机构 (CA)。客户端设备身份验证组件（`aws.greengrass.clientdevices.Auth`）对客户端设备进行身份验证并授权客户端设备操作。有关更多信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的使用自己的证书颁发机构。

要使用自己的 CA，请将该`aws.greengrass.clientdevices.Auth`组件添加到您的部署中，以便您可以指定自定义配置。

为外部防火墙连接打开 8883 端口

Linux

在 Linux 主机防火墙规则中，为端口 8883 添加入站规则，以允许来自网关主机外部的传入连接。如果有任何防火墙，请确保允许端口 8883 上的传入 TLS 连接。

Windows

在您的Microsoft Windows主机防火墙规则中，为端口 8883 添加入站规则，以允许来自网关主机外部的传入连接。确保该规则是允许规则，类型为端口，指定端口 8883。您可以根据自己的网络配置进行配置，以允许从外部应用程序连接到代理。

在 EMQX 中为 Amazon IoT SiteWise Edge 设置授权规则

EMQX 支持根据用户名、IP 地址或客户端 ID 等标识符添加授权规则。如果您想限制连接到各种操作或主题的外部应用程序的数量，这很有用。以下步骤向您展示了如何在 EMQX 中使用文件授权提供程序。有关更多信息，请参阅 EMQX 文档中的 “使用 ACL 文件”。

主题

- 在 Linux 中使用内置数据库配置授权
- 在 Windows 中使用内置数据库配置授权
- 更新 EMQX 部署配置以进行授权
- 通过 EMQX 控制面板或 REST 添加规则 APIs

在 Linux 中使用内置数据库配置授权

配置授权规则时，有两种配置选项取决于您的部署设置。

- Docker— 如果您在没有的情况下运行标准Docker安装Litmus Edge，请使用 Docker 网桥网关配置。当你只部署了 Amazon IoT SiteWise 组件时，通常会出现这种情况。
- Litmus Edge— 如果您已在网关上Litmus Edge安装，请使用Litmus Edge网络子网配置。

> **Note**
>
> 如果您最初配置网Docker桥网关，然后再进行安装Litmus Edge，请使用Litmus Edge网络子网选项重新配置授权规则，以确保所有组件之间的通信正常。

添加基本授权规则

1. 验证 EMQX 代理是否已部署并正在运行。
2. 在您的网关主机上启动 shell 会话。
3. 通过运行以下Docker命令之一来确定网Docker桥网络网关或 Litmus Edge 网络子网 IP 地址：

   Docker bridge gateway

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

   查看类似于以下内容的输出：

   ```
   "IPAM": {
       "Driver": "default",
       "Options": null,
       "Config": [
           {
               "Subnet": "172.17.0.0/16",
   ```

```
            "Gateway": "172.17.0.1"
        }
    ]
},
```

记下网关值。默认情况下，172.17.0.1除非您有自定义Docker设置。

Litmus Edge network subnet

如果您已经Litmus Edge部署，请改为运行以下命令：

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

记下子网值。

4. 通过运行以下命令在 EMQX 代理容器上创建访问控制列表 (ACL) 文件：

```
docker exec emqx mkdir /opt/emqx/data/authz/ && docker exec -it emqx vi /opt/emqx/
data/authz/acl.conf
```

5. 按 “i” 键进入编辑模式。
6. 粘贴以下内容：

Docker bridge gateway

*docker_bridge_gateway*替换为上一步中的网关值。

```
%%-----------------------------------------------------------------------
%% -type(ipaddr() :: {ipaddr, string()}).
%%
%% -type(ipaddrs() :: {ipaddrs, string()}).
%%
%% -type(username() :: {user | username, string()} | {user | username, {re,
 regex()}}).
%%
%% -type(clientid() :: {client | clientid, string()} | {client | clientid, {re,
 regex()}}).
%%
%% -type(who() :: ipaddr() | ipaddrs() |username() | clientid() |
%%                {'and', [ipaddr() | ipaddrs()| username() | clientid()]} |
%%                {'or',  [ipaddr() | ipaddrs()| username() | clientid()]} |
%%                all).
%%
```

```
%% -type(action() :: subscribe | publish | all).
%%
%% -type(topic_filters() :: string()).
%%
%% -type(topics() :: [topic_filters() | {eq, topic_filters()}]).
%%
%% -type(permission() :: allow | deny).
%%
%% -type(rule() :: {permission(), who(), access(), topics()} | {permission(),
 all}).
%%----------------------------------------------------------------------

{allow, {ipaddr, "docker_bridge_gateway"}, all, ["$SYS/#", "#"]}.

{deny, all}.
```

Litmus Edge network subnet

*litmus_network_subnet*替换为上一步中的子网值。

```
%%----------------------------------------------------------------------
%% -type(ipaddr() :: {ipaddr, string()}).
%%
%% -type(ipaddrs() :: {ipaddrs, string()}).
%%
%% -type(username() :: {user | username, string()} | {user | username, {re,
 regex()}}).
%%
%% -type(clientid() :: {client | clientid, string()} | {client | clientid, {re,
 regex()}}).
%%
%% -type(who() :: ipaddr() | ipaddrs() |username() | clientid() |
%%                {'and', [ipaddr() | ipaddrs()| username() | clientid()]} |
%%                {'or',  [ipaddr() | ipaddrs()| username() | clientid()]} |
%%                all).
%%
%% -type(action() :: subscribe | publish | all).
%%
%% -type(topic_filters() :: string()).
%%
%% -type(topics() :: [topic_filters() | {eq, topic_filters()}]).
%%
%% -type(permission() :: allow | deny).
```

```
%%
%% -type(rule() :: {permission(), who(), access(), topics()} | {permission(),
 all}).
%%----------------------------------------------------------------------

{allow, {ipaddr, "litmus_network_subnet"}, all, ["$SYS/#", "#"]}.

{deny, all}.
```

此策略允许网关主机内的所有连接连接到代理、发布和订阅所有主题。这包括物联网 SiteWise OPC UA 收集器和物联网 SiteWise 发布者。

7. 键入 “: wq”，然后按键保存`Enter`并退出文件编辑器。

> **Note**
>
> 此时，规则已写入文件但尚未应用。

8. 继续执行更新 EMQX 部署配置以进行授权。

在 Windows 中使用内置数据库配置授权

本节介绍如何使用内置数据库为 Windows 部署配置授权规则。

添加基本授权规则

1. 验证 EMQX 代理是否已部署并正在运行。
2. 导航到以下目录：

```
C:\greengrass\v2\work\aws.greengrass.clientdevices.mqtt.EMQX\v2\data
```

3. 打开 `authz` 文件夹。如果该文件夹不存在，请创建它。
4. 创建名为的访问控制列表 (ACL) 文件`acl.conf`。
5. 将以下内容粘贴到`acl.conf`文件中：

```
%%----------------------------------------------------------------------
%% -type(ipaddr() :: {ipaddr, string()}).
%%
%% -type(ipaddrs() :: {ipaddrs, string()}).
%%
```

```
%% -type(username() :: {user | username, string()} | {user | username, {re,
 regex()}}).
%%
%% -type(clientid() :: {client | clientid, string()} | {client | clientid, {re,
 regex()}}).
%%
%% -type(who() :: ipaddr() | ipaddrs() |username() | clientid() |
%%                {'and', [ipaddr() | ipaddrs()| username() | clientid()]} |
%%                {'or',  [ipaddr() | ipaddrs()| username() | clientid()]} |
%%                all).
%%
%% -type(action() :: subscribe | publish | all).
%%
%% -type(topic_filters() :: string()).
%%
%% -type(topics() :: [topic_filters() | {eq, topic_filters()}]).
%%
%% -type(permission() :: allow | deny).
%%
%% -type(rule() :: {permission(), who(), access(), topics()} | {permission(),
 all}).
%%--------------------------------------------------------------------

{allow, {ipaddr, "127.0.0.1"}, all, ["$SYS/#", "#"]}.

{deny, all}.
```

此策略允许网关主机内的所有连接连接到代理、发布和订阅所有主题。这包括物联网 SiteWise OPC UA 收集器和物联网 SiteWise 发布者。

6. 保存该文件。

> **Note**
>
> 此时，规则已写入文件但尚未应用。

7. 继续执行更新 EMQX 部署配置以进行授权。

更新 EMQX 部署配置以进行授权

更新 EMQX 部署配置以进行授权

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择要配置的网关。
4. 在 Edge 网关配置部分，复制您的 Greengrass 核心设备值。将其保存以备后用。
5. 打开 Amazon IoT 管理控制台。
6. 在左侧导航栏的 “管理” 部分下，选择 Greengrass 设备，然后选择 “部署”。
7. 找到您之前保存的核心设备值，然后选择该链接以打开部署。
8. 选择 “操作” 下拉按钮，然后选择 “修改”。
9. 阅读显示的消息，然后选择修订部署。将出现 “指定目标” 页面。
10. 选择 “下一步”，直到进入 “配置组件” 步骤。
11. 选择 `aws.greengrass.clientdevices.mqtt.EMQX` 单选按钮。
12. 选择 “配置组件” 按钮。此时将显示该组件的配置页面。
13. 在 “配置更新” 下，选择 “重置为组件版本的默认配置：2.*”。 *。
14. 根据您的操作系统将以下内容粘贴到要合并的配置部分。

    Linux

```
{
    "emqxConfig": {
        "authorization": {
            "no_match": "deny",
            "sources": [
                {
                    "enable": true,
                    "path": "/opt/emqx/data/authz/acl.conf",
                    "type": "file"
                }
            ]
        },
        "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
 -p 127.0.0.1:18083:18083 -v emqx-data:/opt/emqx/data -e
 EMQX_NODE__NAME=emqx@local",
    "requiresPrivilege": "true"
}
```

Windows

```
{
    "emqxConfig": {
```

```
        "authorization": {
            "no_match": "deny",
            "sources": [
                {
                    "enable": true,
                    "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. 选择确认。
16. 选择 “下一步”，直到进入 “查看” 步骤。
17. 选择部署。

> **Note**
>
> 此后，您无法编辑 ACL 文件来更新授权规则。或者，您可以在成功部署通过 EMQX 控制面板或 REST 添加规则 APIs后继续。

通过 EMQX 控制面板或 REST 添加规则 APIs

编辑先前添加的 ACL 文件不会更新授权规则。相反，您可以使用 EMQX 控制面板或 EMQX REST 来添加或更新授权规则。 APIs

> **Note**
>
> 添加授权规则是高级配置步骤，需要熟悉 ACL 文件格式化选项。有关使用 EMQX 创建授权规则的更多信息，请参阅 EMQX 文档中的使用 ACL 文件部分

> **Note**
>
> 所提供的 EMQX 相关说明仅供参考。由于 EMQX 的文档和功能可能会随着时间的推移而发生变化，并且我们不保留其文档，因此我们建议您查阅 EMQX 的官方文档以获取最新信息。

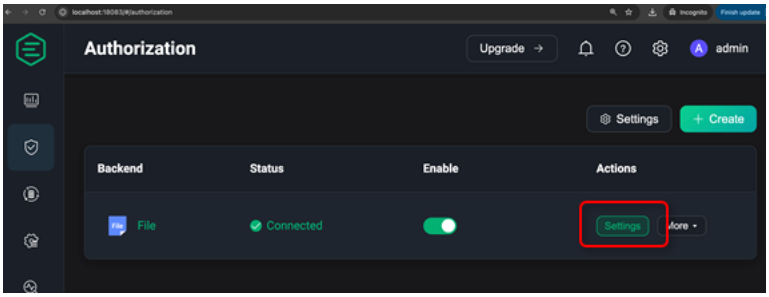
EMQX dashboard

此过程说明如何在 EMQX 控制面板上添加授权规则。

EMQX 控制面板只能从网关主机内部访问。如果您尝试从网关主机外部进行连接，则无法访问控制面板。

使用 EMQX 控制面板添加授权规则

1. 确保您在网关主机内。
2. 打开浏览器窗口并访问http://localhost:18083/。
3. 登录 EMQX 控制面板。此过程假设您已将默认登录凭证更改为自己选择的登录凭证。有关初始设置的更多信息，请参阅启用用户名和密码认证。
4. 选择盾牌图标，然后从下拉菜单中选择 “授权”。
5. 在 “授权” 屏幕的表格中选择 “设置” 按钮。请注意，这不是整个页面的 “设置” 按钮。



6. 在 ACL 文件部分，根据您的业务需求添加或更新授权规则。应在现有规则之后添加新规则。有关创建规则的更多指导，请参阅 EMQX 文档中的 “使用 ACL 文件” 部分。

EMQX REST APIs using Linux

使用 shell 环境和 EMQX 代理查看和添加授权规则 REST APIs

1. 登录 EMQX 控制面板。此过程假设您已将默认登录凭证更改为自己选择的登录凭证。有关初始设置的更多信息，请参阅启用用户名和密码认证。
2. 运行以下命令查看您现有的授权规则。用您自己的登录信息替换管理员密码。

```
curl -s -X 'POST' 'http://localhost:18083/api/v5/login' -H'accept: application/
json' \
-H 'Content-Type: application/json' \
 -d '{"username": "admin", "password": "your-admin-password" }' \
| jq -r '.token' \
| xargs -I {} curl 'http://localhost:18083/api/v5/authorization/sources/file' \
  -H 'Authorization: Bearer {}' \
| jq .
```

3. 使用以下命令更新 ACL 文件：

```
docker exec -it emqx vi /opt/emqx/data/authz/acl.conf
```

按 “i” 键进入编辑模式，然后根据业务需求添加或更新规则。有关创建规则的更多指导，请参阅 EMQX 文档中的 “使用 ACL 文件” 部分。

完成编辑后，按 “: wq”，然后按 “Enter” 或 “return” 来保存和编辑文件编辑器。

4. 添加之前创建的基本授权规则。有关更多信息，请参阅 在 EMQX 中为 Amazon IoT SiteWise Edge 设置授权规则。
5. 根据需要添加或更新授权规则。
6. 使用以下命令调用 EMQX API 以应用更新的规则。用您自己的登录信息替换管理员密码。将`path-to-rules-file`替换为新授权规则的文件路径。

```
UPDATED_RULES=$(cat path-to-rules-file)
        curl -s -X 'POST' 'http://localhost:18083/api/v5/login' -H'accept:
 application/json' \
        -H 'Content-Type: application/json' \
         -d '{"username": "admin", "password": "your-admin-password" }' \
        | jq -r '.token' \
        | xargs -I {} curl 'http://localhost:18083/api/v5/authorization/sources/
file' \
          -H 'Authorization: Bearer {}' \
          -X 'PUT' \
          --data-raw "{\"enable\":true,\"rules\":\"$UPDATED_RULES\",\"type\":
\"file\"}"
```

您的新授权规则应在您调用 EMQX API 后生效。

EMQX REST APIs using Windows

要查看现有授权规则，请使用Windows PowerShell和 EMQX 代理 REST APIs

1. 登录 EMQX 控制面板。此过程假设您已将默认登录凭证更改为自己选择的登录凭证。有关初始设置的更多信息，请参阅启用用户名和密码认证。
2. 运行以下命令查看您现有的授权规则。用您自己的登录信息替换管理员密码。

```
try {
```

```
    $loginResponse = Invoke-RestMethod -Uri 'http://localhost:18083/api/v5/
login' -Method Post -ContentType 'application/json' -Body '{"username": "admin",
 "password": "your-admin-password"}'
    $token = $loginResponse.token

    $viewAuthorizationRuleResponse = Invoke-RestMethod -Uri 'http://
localhost:18083/api/v5/authorization/sources/file' -Method Get -Headers
 @{"Authorization" = "Bearer $token"}

    $viewAuthorizationRuleResponse | ConvertTo-Json
} catch {
    Write-Output "Failed to fetch current authorization rules: $_"
}
```

3. 使用以下命令更新 ACL 文件：

```
C:\greengrass\v2\work\aws.greengrass.clientdevices.mqtt.EMQX\v2\data\authz
\acl.conf
```

根据您的业务需求添加或更新规则。有关创建规则的更多指导，请参阅 EMQX 文档中的 “使用 ACL 文件” 部分。

4. 根据需要添加或更新授权规则。
5. 使用以下命令调用 EMQX API 以应用更新的规则。用您自己的登录信息替换管理员密码。

```
try {
    # Login and get token
    $loginResponse = Invoke-RestMethod -Uri 'http://localhost:18083/api/v5/
login' -Method Post -ContentType 'application/json' -Body '{"username": "admin",
 "password": "your-admin-password"}'
    $token = $loginResponse.token

    # Read the content of the ACL file from the C volume
    $aclContent = Get-Content -Path "C:\greengrass\v2\work
\aws.greengrass.clientdevices.mqtt.EMQX\v2\data\authz\acl.conf" | Out-String

    # Prepare the request body
    $body = @{
        enable = $true
        rules = $aclContent
        type = "file"
    } | ConvertTo-Json
```

```
    # Update authorization rules
    $updateAuthorizationRuleResponse = Invoke-RestMethod -Uri 'http://
localhost:18083/api/v5/authorization/sources/file' -Method Put -Headers @{
        "Authorization" = "Bearer $token"
        "Content-Type" = "application/json"
    } -Body $body

    # Output the response
    $updateAuthorizationRuleResponse | ConvertTo-Json

    Write-Output "Updated authorization rules successfully!"
} catch {
    Write-Output "Failed to update authorization rules: $_"
}
```

经典直播，适用 Amazon IoT SiteWise 于 Edge 的 V2 网关

了解 Classic 直播的功能和局限性，即适用于 Amazon IoT SiteWise Edge 的 V2 网关。

Classic Streams，V2 网关保留了在推出支持 MQTT 的 V3 网关之前的早期 Amazon IoT SiteWise 部署中熟悉的传统功能。这些 SiteWise Edge 网关被视为经典流，即 V2 网关。它们保持向后兼容性，并且可以与数据处理包配合使用。虽然 Classic 直播，但 V2 网关为现有设置提供了可靠的性能，但与较新的网关选项相比，它存在局限性。具体而言，这种网关类型与支持 MQTT 的 V3 网关目标中提供的高级功能不完全兼容。要使用 MQTT 消息协议，您可以创建一个支持 MQTT 的新的 V3 网关。有关更多信息，请参阅 支持 MQTT 的边缘版 V3 网关 Amazon IoT SiteWise。

配置 Amazon IoT SiteWise 发布者组件

创建 Amazon IoT SiteWise Edge 网关并安装软件后，您可以设置发布者组件，以便您的 SiteWise Edge 网关可以将数据导出到 Amazon 云端。使用发布者组件启用其他特征或配置默认设置。有关更多信息，请参阅《Amazon IoT Greengrass Version 2 开发者指南》中的 Amazon IoT SiteWise publisher。

> **Note**
>
> 发布者配置因您使用的网关类型而异。对于经典流，V2 网关，请使用`iotsitewise:publisher:2`命名空间。对于支持 MQTT 的 V3 网关，请使用命名空间。`iotsitewise:publisher:3`

Console

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要为其配置发布商的 SiteWise Edge 网关。
4. 在发布者配置部分，选择编辑
5. 对于发布顺序，请选择下列选项之一：
   - 先发布最旧的数据-默认情况下， SiteWise Edge 网关会先将最旧的数据发布到云端。
   - 首先发布最新数据 — SiteWise Edge 网关首先将最新数据发布到云端。
6. （可选）如果您不希望 SiteWise Edge 网关压缩您的数据，请取消选择 “上传数据时激活压缩”。
7. （可选）如果您不想发布旧数据，请选择排除已过期数据，然后执行以下操作：
   - 在截止期中输入一个值并选择一个单位。截止期必须介于 5 分钟到 7 天之间。例如，如果截止期为三天，则早于三天的数据不会被发布到云端。
8. （可选）要设置有关如何在本地设备上处理数据的自定义设置，请选择本地存储设置并执行以下操作：
   a. 在保留期中输入一个数字并选择一个单位。保留期必须介于 1 分钟到 30 天之间，且大于或等于轮换期。例如，如果保留期为 14 天，则 SiteWise Edge 网关会在边缘存储超过 14 天后删除所有早于指定截止期限的数据。
   b. 在轮换期中输入一个数字并选择一个单位。轮换期必须多于 1 分钟，且等于或小于保留期。例如，假设轮换周期为两天， SiteWise Edge 网关会批量处理并将早于截止期限的数据保存到单个文件中。对于通过自托管的网关 Amazon IoT Greengrass V2， SiteWise Edge 网关每两天将一批数据传输到以下本地目录一次：`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`。
   c. 对于存储容量，请输入一个大于或等于 1 的值。如果存储容量为 2 GB，则当本地存储的数据超过 2 GB 时， SiteWise Edge 网关会开始删除数据。
9. 选择保存。

Amazon CLI

使用 UpdateGatewayCapabilityConfigurationAPI 配置发布者。

将 `capabilityNamespace` 参数设置为 `iotsitewise:publisher:2`。

Example：Classic Stream、V2 网关的发行商配置

发布者命名空间：`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
                }
            }
        ]
    }
}
```

发布者提供了您可以自定义的以下配置参数：

`SiteWisePublisherConfiguration`

`publishingOrder`

数据发布到云的顺序。此参数值可能为下列其中之一：

- `TIME_ORDER`（先发布最早的数据）— 默认情况下，网关会先将最早的数据发布到云。

- RECENT_DATA（先发布最新数据）— 网关会先将最新数据发布到云。

enableCompression

将其设置true为可在发布之前压缩数据。数据压缩可以减少带宽使用量。

dropPolicy

（可选）控制将数据发布到云的策略。

cutoffAge

要发布的数据的最大期限，以天、小时和分钟为单位指定。例如，7d 或 1d7h16m。不会将早于您指定时间的数据发送到 Amazon IoT SiteWise。

早于截止期的数据不会发布到云端。截止期必须介于 5 分钟到 7 天之间。

当您指定截止期时，可以使用 m、h 和 d。请注意，m 表示分钟，h 表示小时，d 表示天。

exportPolicy

（可选）管理边缘数据存储的策略。本策略适用于早于截止期的数据。

retentionPeriod

您的 SiteWise Edge 网关在存储指定保留期后，会从本地存储中删除所有早于截止期限的数据。保留期必须介于 1 分钟到 30 天之间，且大于或等于轮换期。

当您指定保留期时，可以使用 m、h 和 d。请注意，m 表示分钟，h 表示小时，d 表示天。

rotationPeriod

将早于截止期的数据进行批处理和保存至单个文件的时间间隔。 SiteWise Edge 网关在每个轮换周期结束时将一批数据传输到以下本地目录:/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports. 轮换期必须多于 1 分钟，且等于或小于保留期。

当您指定轮换期时，可以使用 m、h 和 d。请注意，m 表示分钟，h 表示小时，d 表示天。

exportSizeLimitGB

允许存储在本地的数据最大以 GB 为单位。如果超过此配额， SiteWise Edge 网关会开始删除最早的数据，直到本地存储的数据大小等于或小于配额。该参数值必须大于或等于 1。

`SiteWiseS3PublisherConfiguration`

`accessRoleArn`

授予管理您要发布到的 Amazon S3 存储桶的 Amazon IoT SiteWise 权限的访问角色。

`streamToS3ConfigMapping`

将流映射到 Amazon S3 配置的一组配置。

`streamName`

要读取并发布到 Amazon S3 配置的流。

`targetBucketArn`

要发布到的 ARN 存储桶。

`publishPolicy`

`publishFrequency`

SiteWise 边缘网关向 Amazon S3 存储桶发布内容的频率。

`localSizeLimitGB`

写入本地磁盘的文件的最大大小。如果突破此阈值，则发布者会将所有缓冲的数据发布到其目的地。

`siteWiseImportPolicy`

`enableSiteWiseStorageImport`

将其设置为可`true`将数据从 Amazon S3 存储桶导入 Amazon IoT SiteWise 存储。

`enableDeleteAfterImport`

将其设置为 `true`，以在将 Amazon S3 存储桶中的文件摄取到 Amazon IoT SiteWise 存储空间后将其删除。

目的地和 Amazon IoT Greengrass 直播管理器

Amazon IoT Greengrass 流管理器允许您将数据发送到以下 Amazon Web Services 云 目的地：传入的频道 Amazon IoT Analytics、Amazon Kinesis Data Streams 中的流、 Amazon IoT SiteWise中的资产属性或亚马逊简单存储服务 (Amazon S3) 中的对象。有关更多信息，请参阅《Amazon IoT

Greengrass Version 2 开发人员指南》中的 Manage data streams on the Amazon IoT Greengrass Core。

Example：数据流消息结构

以下示例显示了流管理器传输的所需数据 Amazon IoT Greengrass 流消息结构。

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

> **Note**
>
> 数据流消息的结构中必须包含（`assetId` 和 `propertyId`）或 `propertyAlias`。

`assetId`

（可选）要更新的资产的 ID。

`propertyAlias`

（可选）标识属性的别名，例如 OPC UA 服务器数据流路径。例如：

```
/company/windfarm/3/turbine/7/temperature
```

有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的管理数据流。

`propertyId`

（可选）此条目的资产属性的 ID。

`propertyValues`

（必需）要上传的属性值的列表。您最多可以指定 10 个 `propertyValues` 数组元素。

`quality`

（可选）资产属性值的质量。

`timestamp`

（必需）资产属性值的时间戳。

`offsetInNanos`

（可选）来自 `timeInSeconds` 的纳秒偏移量。

`timeInSeconds`

（必需）采用 Unix 纪元时间格式的时间戳日期（以秒为单位）。分数纳秒数据由 `offsetInNanos` 提供。

`value`

（必需）资产属性的值。

> **Note**
>
> `value` 字段仅可以是以下值之一。

`booleanValue`

（可选）布尔型（`true` 或 `false`）资产属性数据。

`doubleValue`

（可选）双精度型（浮点数）资产属性数据。

`integerValue`

（可选）整数型（整数）资产属性数据。

`stringValue`

（可选）字符串型（字符序列）资产属性数据。

在 Edge 上配置 Amazon IoT SiteWise 边缘功能

您可以使用 Amazon IoT SiteWise Edge 收集和临时存储数据，以便可以在本地组织和处理设备数据。通过启用边缘处理，您可以选择仅向发送聚合数据， Amazon Web Services 云 以优化带宽使用率和云存储成本。使用带有的 Amazon IoT SiteWise 组件 Amazon IoT Greengrass，您可以在将数据发送到之前在边缘收集和处理数据 Amazon Web Services 云，或者使用 E SiteWise dge 在本地对其进行管理 APIs。

数据收集是通过数据包和在其上运行的 Amazon IoT SiteWise 组件进行的 Amazon IoT Greengrass。

> **Note**
>
> - Amazon IoT SiteWise 在 Edge 网关上保留您的 SiteWise 边缘数据长达 30 天。数据的保留期取决于您的设备的可用磁盘空间。
> - 如果您的 SiteWise Edge 网关已与 Edge 网关断开连接 30 天，则数据处理包将自动禁用。Amazon Web Services 云

主题

- 在 Edge 中设置 SiteWise 边缘功能

在 Edge 中设置 SiteWise 边缘功能

Amazon IoT SiteWise 提供了以下数据包，您的 SiteWise Edge 网关可用来确定如何收集和处理您的数据。选择包以为 Edge 网关启用 SiteWise 边缘功能。

- 数据收集包使您的 SiteWise Edge 网关能够从多台 OPC UA 服务器收集数据，然后将数据从边缘导出到 Amazon 云端。向 SiteWise Edge 网关添加数据源后，它将变为活动状态。
- 数据处理包使您的 SiteWise Edge 网关能够在边缘处理您的设备数据。例如，您可以使用资产模型来计算指标和转换。有关资产模型和资产的更多信息，请参阅 为工业资产建模。

> **Note**
> - 数据处理包仅在 x86 平台上可用。

要配置边缘功能，请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要为其激活 SiteWise 边缘功能的 Edge 网关。
4. 在边缘功能部分，选择编辑
5. 在边缘功能部分，选择启用数据处理包（会产生额外费用）。
6. （可选）在 Edge LDAP 连接部分，您可以授予公司目录中的用户组访问此 SiteWise Edge 网关的权限。用户组可以使用轻型目录访问协议 (LDAP) 凭据访问 SiteWise Edge 网关。然后，他们可以使用 Amazon IoT SiteWise 应用程序、 Amazon IoT SiteWise API 操作或其他工具来管理 SiteWise Edge 网关。 Amazon OpsHub 有关更多信息，请参阅 管理 SiteWise Edge 网关。

   > **Note**
   > 您也可以使用 Linux 或Microsoft Windows凭据访问边 SiteWise 缘网关。有关更多信息，请参阅 使用 Linux 操作系统凭据访问您的 SiteWise Edge 网关。

   a. 选择已激活。
   b. 对于提供者名称，为您的 LDAP 提供者输入一个名称。
   c. 对于主机名或 IP 地址，输入 LDAP 服务器的主机名或 IP 地址。
   d. 对于端口，输入端口号。
   e. 对于基本可分辨名称 (DN)，输入基本的可分辨名称 (DN)。

      支持以下属性类型：公用名 (CN)、LocalityName (L)、名称 (ST)、组织 stateOrProvince名称 (O)、(OU)、国家名称 organizationalUnitName (C)、街道地址 (STREET)、域名组件 (DC) 和用户 ID (UID)。
   f. 对于管理员组 DN，输入 DN。
   g. 对于用户组 DN，输入 DN。

7. 选择保存。

现在，您已经在 Edge 网关上激活了 SiteWise 边缘功能，您需要为边缘配置资产模型。您的资产模型边缘配置指定计算资产属性的位置。您可以在边缘计算所有属性，或单独配置资产模型属性。资产模型属性包括指标、转换和测量值。

有关资产属性的更多信息，请参阅 the section called “定义数据属性”。

创建资产模型后，您可以为边缘进行配置。有关您的边缘资产模型配置的更多信息，请参阅 the section called “创建资产模型（控制台）”。

> **Note**
>
> 资产模型和仪表板每 10 分钟在 Amazon 云端和 SiteWise 边缘网关之间自动同步一次。您也可以从本地 SiteWise Edge 网关应用程序手动同步。

为 Amazon IoT SiteWise 模型和资产配置边缘数据处理

您可以使用 Amazon IoT SiteWise Edge 在本地收集、存储、组织和监控设备数据。您可以使用 SiteWise Edge 对工业数据进行建模，使用 M SiteWise onitor 创建仪表板供操作人员在本地可视化数据。您可以在本地处理数据并将其发送到 Amazon 云端，也可以使用 Amazon IoT SiteWise API 在本地进行处理。

借 Amazon IoT SiteWise 助 Edge，您可以在本地处理原始数据，并选择仅将聚合数据发送到 Amazon 云端，以优化带宽使用率和云存储成本。

> **Note**
>
> - Amazon IoT SiteWise 在 Edge 网关上保留您的 SiteWise 边缘数据长达 30 天。数据的保留期取决于您的设备的可用磁盘空间。
> - 如果您的 SiteWise Edge 网关已与 Amazon 云端断开连接 30 天，在 Edge 中 SiteWise 设置 OPC UA 来源则会自动禁用。

在 E SiteWise dge 上配置用于数据处理的资产模型

资产属性包括指标、转换和测量值：

- 指标是在指定的时间段内资产的汇总数据。您可以使用现有指标数据计算新的指标。 Amazon IoT SiteWise 始终将您的指标发送到 Amazon 云端以进行长期存储。 Amazon IoT SiteWise 默认情况下，在 Amazon 云端计算指标。您可以配置资产模型以计算边缘指标。 Amazon IoT SiteWise 将处理后的结果发送到 Amazon 云端。
- 转换是将资产属性的数据点从一种形式映射到另一种形式的数学表达式。转换可以使用指标作为输入数据，并且必须在与其输入相同的位置进行计算和存储。如果将指标输入配置为在边缘计算，则 Amazon IoT SiteWise 还会在边缘计算其关联的转换。
- 默认情况下，测量值格式为原始数据，您的设备会收集这些数据并发送到 Amazon 云端。您可以将资产模型配置为将数据存储于本地设备。

有关资产属性的更多信息，请参阅 the section called “定义数据属性”。

创建资产模型后，您可以为边缘进行配置。有关您的边缘资产模型配置的更多信息，请参阅 the section called “创建资产模型（控制台）”。

> **Note**
>
> 资产模型和仪表板每 10 分钟在 Amazon 云端和 SiteWise 边缘网关之间自动同步一次。您也可以从 管理 SiteWise Edge 网关 手动同步。

您可以使用 Amazon IoT SiteWise REST APIs 和 Amazon Command Line Interface (Amazon CLI) 来查询 SiteWise Edge 网关以获取边缘的数据。在向 SiteWise Edge 网关查询边缘数据之前，必须满足以下先决条件：

- 必须为 REST 设置您的证书 APIs。有关设置凭证的更多信息，请参阅 the section called “管理网关”。
- SDK 终端节点必须指向您的 SiteWise Edge 网关的 IP 地址。您可以在您的 SDK 文档中了解更多信息。例如，请参阅 Amazon SDK for Java 2.x Developer Guide（开发人员指南）中的 Specifying Custom Endpoints（指定自定义端点）。
- 必须注册您的 SiteWise Edge 网关证书。您可以在软件开发工具包的文档中找到有关注册 SiteWise Edge 网关证书的更多信息。例如，请参阅 Amazon SDK for Java 2.x 开发人员指南中的 在 Node.js 中注册证书包。

有关使用查询数据的更多信息 Amazon IoT SiteWise，请参阅从中查询数据 Amazon IoT SiteWise。

将数据源添加到您的 Amazon IoT SiteWise Edge 网关

设置 Amazon IoT SiteWise Edge 网关后，您可以添加和配置数据源，以便从本地工业设备摄取数据。Amazon IoT SiteWise SiteWise Edge 支持各种协议，包括 OPC UA 以及通过合作伙伴数据源提供的许多其他协议。这些来源使您的网关能够连接到本地服务器并检索您的工业数据。通过配置数据源，您可以从各种数据源提取数据，然后将数据流与资产属性相关联，从而在中 Amazon IoT SiteWise实现全面的工业资产建模和数据映射。

主题

- Amazon IoT SiteWise 边缘网关的 OPC UA 数据源

Amazon IoT SiteWise 边缘网关的 OPC UA 数据源

设置 Amazon IoT SiteWise Edge 网关后，您可以配置数据源，以便您的 SiteWise Edge 网关可以将来自本地工业设备的数据摄取到 Amazon IoT SiteWise。每个源都代表一个本地服务器，例如 OPC UA 服务器，您的 SiteWise 边缘网关可以连接和检索工业数据流。有关设置 SiteWise Edge 网关的更多信息，请参阅创建自托管 SiteWise Edge 网关。

网关类型（支持 MQTT）、V3 网关与经典流（V2 网关）会影响 OPC UA 数据的处理方式。在 Classic Stream 中，V2 网关、OPC UA 数据源直接添加到网关 IoT SiteWise 发布者配置中。每个数据源都与网关耦合，并且为每个源单独配置数据路由。相比之下，使用支持 MQTT 的 V3 网关，OPC UA 数据源会转换为 MQTT 主题，并通过集中式目的地进行管理。有关每种类型的更多信息，请参见支持 MQTT 的边缘版 V3 网关 Amazon IoT SiteWise和经典直播，适用 Amazon IoT SiteWise 于 Edge 的 V2 网关。

> **Note**
>
> Amazon IoT SiteWise 每次添加或编辑源时都会重新启动 SiteWise Edge 网关。您的 SiteWise Edge 网关在更新源配置时不会提取数据。重启 SiteWise Edge 网关的时间取决于 E SiteWise dge 网关源上的标签数量。重启时间可以从几秒钟（对于标签较少的 SiteWise Edge 网关）到几分钟（对于具有许多标签的 SiteWise Edge 网关）不等。

创建源后，您可以将数据流与资产属性关联。有关如何创建和使用资产的更多信息，请参阅为工业资产建模。

您可以查看 CloudWatch 指标以验证数据源是否已连接 Amazon IoT SiteWise。有关更多信息，请参阅 Amazon IoT Greengrass Version 2 网关指标。

目前， Amazon IoT SiteWise 支持以下数据源协议：

- OPC UA — 一种用于工业自动化的通信协议 machine-to-machine（M2M）。

Support 支持其他工业协议

SiteWise 通过与数据源合作伙伴的集成，Edge 支持各种工业协议。这些合作伙伴关系可实现与 200 多种不同协议的连接，适用于各种工业系统和设备。

在 Edge 中 SiteWise 设置 OPC UA 来源

您可以使用 Amazon IoT SiteWise 控制台或 SiteWise Edge 网关功能来定义并向边缘网关添加 OPC UA 源，以表示本地 OPC UA 服务器。 SiteWise

主题

- 配置 OPC UA 源（控制台）
- 配置 OPC UA 来源 ()Amazon CLI

配置 OPC UA 源（控制台）

您可以使用控制台按照以下步骤配置 OPC UA 源。

> **Note**
>
> 警告：重复 TQVs 可能会导致双重充电。

使用控制台配置 OPC UA 源 Amazon IoT SiteWise

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择边 SiteWise 缘网关以添加 OPC UA 源。
4. 选择添加数据来源。
5. 输入源名称。
6. 输入数据源服务器的 本地终端节点。端点可以是 IP 地址或主机名。您也可以向本地端点添加端口号。例如，您的本地端点可能类似于 **`opc.tcp://203.0.113.0:49320`**。
7. 启用 MQTT 的 V3 网关和经典直播 V2 网关之间的@@ 目的地各不相同。

- Classic Steam、V2 网关目的地与源有 1:1 的关系。每个源都将数据发送到特定的目的地。
- 支持 MQTT 的 V3 网关目的地是单独设置的，因为中心和分支模型允许您集中配置和管理跨不同网关的多个数据源。要在 V3 网关中设置目的地，请参阅了解 Amazon IoT SiteWise 边缘目的地。

Classic steams, V2 gateway destinations

- Amazon IoT SiteWise 实时-选择此选项可将数据直接发送到 Amazon IoT SiteWise 存储器。在边缘实时摄取和监控数据。
- Amazon IoT SiteWise 使用 Amazon S3 进行缓冲 — 以 Parquet 格式将数据发送到 Amazon S3，然后导入 Amazon IoT SiteWise 存储。选择此选项可批量摄取数据，并以经济高效的方式存储历史数据。您可以配置您的首选 Amazon S3 存储桶位置，以及您希望将数据上传到 Amazon S3 的频率。您还可以选择在将数据摄取到 Amazon IoT SiteWise后如何处理数据。您可以选择同时在 Amazon S3 Amazon IoT SiteWise 和 Amazon S3 中提供数据，也可以选择在数据导入 Amazon S3 后自动将其从 Amazon S3 中删除 Amazon IoT SiteWise。
  - Amazon S3 存储桶是一种暂存和缓冲机制，支持 Parquet 格式的文件。
  - 如果您选中 “将数据导入 Amazon IoT SiteWise 存储” 复选框，则会先将数据上传到 Amazon S3，然后上传到 Amazon IoT SiteWise 存储中。
    - 如果您选中 “从 Amazon S3 中删除数据” 复选框，则在数据导入 SiteWise 存储后，数据将从 Amazon S3 中删除。
    - 如果您清除 “从 Amazon S3 中删除数据” 复选框，则数据将存储在 Amazon S3 和 SiteWise 存储中。
  - 如果您清除将数据导入 Amazon IoT SiteWise 存储复选框，则数据将仅存储在 Amazon S3 中，它不会导入到 SiteWise 存储中。

  有关 Amazon IoT SiteWise 提供的各种存储选项的详细信息，请访问管理数据存储。要了解有关定价选项的更多信息，请参阅 Amazon IoT SiteWise 定价。

- Amazon IoT Greengrass 流管理器 — 使用 Amazon IoT Greengrass 流管理器将数据发送到以下 Amazon 云端目标：传入的频道、Amazon Kinesis Data Streams 中的流 Amazon IoT Analytics、中的资产属性或亚马逊简单存储服务 (Amazon S3) Simple Storage Service Amazon IoT SiteWise中的对象。有关更多信息，请参阅《Amazon IoT Greengrass Version 2 开发者指南》中的 “在 Amazon IoT Greengrass 核心上管理数据流”。

  输入 Amazon IoT Greengrass 直播的名称。

MQTT-enabled, V3 gateway destinations

1. 支持 MQTT 的边缘版 V3 网关 Amazon IoT SiteWise有关添加相关目的地的信息，请参阅。
2. 添加源目标后，返回此过程。

8. 在高级配置窗格中，您可以执行以下操作：

   a. 为源服务器和 SiteWise Edge 网关之间传输的连接和数据选择消息安全模式。此字段是 OPC UA 安全策略和消息安全模式的组合。选择与您为 OPC UA 服务器指定的相同的安全策略和消息安全模式。

   b. 如果您的来源需要身份验证，请从身份验证配置列表中选择一个 Amazon Secrets Manager 密钥。 SiteWise Edge 网关在连接到此数据源时使用此密钥中的身份验证凭据。您必须将密钥附加到 SiteWise Edge 网关的 Amazon IoT Greengrass 组件，才能将其用于数据源身份验证。有关更多信息，请参阅 the section called “配置数据源身份验证”。

   > **Tip**
   > 您的数据服务器可能有一个名为 允许匿名登录 的选项。如果此选项为 是，则说明您的源不需要身份验证。

   c. （可选）您可以通过选择激活数据流前缀 - 可选来激活数据流前缀。

      - （可选）输入数据流前缀。 SiteWise Edge 网关将此前缀添加到来自该源的所有数据流。使用数据流前缀，可以区分来自不同源的同名数据流。在您的账户中，每个数据流都应具有唯一的名称。

   d. （可选）选择数据类型转换选项，将不支持的 OPC UA 数据类型转换为字符串，然后再将其摄取到 Amazon IoT SiteWise中。将具有简单数据类型的数组值转换为 JSON 字符串，将 DateTime 数据类型转换为 ISO 8601 字符串。有关更多信息，请参阅 转换不支持的数据类型。

   e. （可选）对于属性组，选择添加新组。

      i. 为属性组输入名称。

      ii. 对于属性：

         1. 对于节点路径，添加 OPC UA 节点筛选条件以限制上传到 Amazon IoT SiteWise的 OPC UA 路径。其格式类似于要选择的节点 ID。

iii. 对于 组设置，请执行以下操作：

1. 在数据质量设置中，选择您希望 C Amazon IoT SiteWise ollector 采集的数据质量类型。
2. 对于扫描模式设置，请使用扫描模式配置标准订阅属性。您可以选择订阅或轮询。有关扫描模式的更多信息，请参阅 the section called “筛选数据摄取范围”。

   Subscribe

   要发送每个数据点，请执行以下步骤：

   i. 选择订阅并设置以下内容：

      A. 数据更改触发器 — 启动数据变更警报的条件。

      B. 订阅队列大小 — OPC-UA 服务器上特定指标的队列深度，其中受监控项目的通知已排队。

      C. 订阅发布间隔 — 创建订阅时指定的发布周期间隔（以毫秒为单位）。

      D. 快照间隔-可选- 用于确保 Amazon IoT SiteWise Edge 摄取稳定的数据流的快照频率超时设置。

      E. 扫描速率 — 您希望 SiteWise Edge 网关读取寄存器的速率。 Amazon IoT SiteWise 自动计算您的 SiteWise Edge 网关允许的最低扫描速率。

      F. 时间戳 — 要包含在 OPC UA 数据点中的时间戳。您可以使用服务器时间戳或设备的时间戳。

      > **Note**
      >
      > 使用 2.5.0 或更高版本的 IoT SiteWise OPC UA 收集器组件。如果您在早期版本中使用时间戳功能，则配置更新会失败。有关更多信息，请参阅 更新 Amazon IoT SiteWise 组件的版本。

   ii. 在死区设置中，配置死区类型。死区类型控制您的源发送给您的 Amazon IoT SiteWise数据以及它丢弃了哪些数据。有关死区设置的更多信息，请参阅 the section called “筛选数据摄取范围”。

      - 无 — 关联的服务器发送此属性组的所有数据点。
      - 百分比 — 关联的服务器仅发送超出数据范围指定百分比的数据。该范围由服务器根据为每个节点定义的最小和最大工程单位计算得出。如果服务器不

支持百分比死区或缺少定义的工程单元，则网关将使用下面提供的最小值和最大值来计算范围。

- 绝对 — 关联的服务器仅发送超出特定范围的数据。

A. 将死区值设置为数据范围与死区的百分比。

B. （可选）使用最小范围 - 可选和最大范围 - 可选，为死区范围指定最小值和最大值。

Poll

要按特定间隔发送数据点，请执行以下步骤：

- 选择轮询并设置以下内容：

A. 扫描速率 — 您希望 SiteWise Edge 网关读取寄存器的速率。 Amazon IoT SiteWise 自动计算您的 SiteWise Edge 网关允许的最低扫描速率。

B. 时间戳 — 要包含在 OPC UA 数据点中的时间戳。您可以使用服务器时间戳或设备的时间戳。

> **Note**
>
> 使用 2.5.0 或更高版本的 IoT SiteWise OPC UA 收集器组件。如果您在早期版本中使用时间戳功能，则配置更新会失败。有关更多信息，请参阅 更新 Amazon IoT SiteWise 组件的版本。

> **Note**
>
> 当您在扫描模式设置中选择订阅时，死区设置适用。

9. 选择保存。

配置 OPC UA 来源 ()Amazon CLI

您可以使用为 SiteWise Edge 网关定义 OPC UA 数据源。 Amazon CLI为此，请创建一个 OPC UA 功能配置 JSON 文件，然后使用 update-gateway-capability-configuration命令更新 SiteWise Edge 网关配置。您必须在单个功能配置中定义所有 OPC UA 源。

MQTT-enabled, V3 gateway

此功能具有以下命名空间。

- `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_RSAOAEP" | "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

此功能具有以下命名空间。

- iotsitewise:opcuacollector:2

请求语法

```
{
  "sources": [
    {
      "name": "string",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny" | "X509",
          "certificateBody": "string",
          "certificateChain": "string",
        },
        "endpointUri": "string",
        "securityPolicy": "NONE" | "BASIC128_RSA15" | "BASIC256" | "BASIC256_SHA256"
 | "AES128_SHA256_RSAOAEP" | "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
          }
        }
      ]
    }
  ]
}
```

请求正文

sources

OPC UA 源定义结构的列表，每个结构均包含以下信息：

`name`

源的唯一友好名称。

`endpoint`

一个包含以下信息的终端节点结构：

`certificateTrust`

一个包含以下信息的证书信任策略结构：

`type`

源的证书信任模式。选择下列选项之一：

- `TrustAny`— 边 SiteWise 缘网关在连接到 OPC UA 源时信任任何证书。
- `X509`— 边 SiteWise 缘网关在连接到 OPC UA 源时信任 X.509 证书。如果选择此选项，则必须在 `certificateTrust` 中定义 `certificateBody`。也可以在 `certificateTrust` 中定义 `certificateChain`。

`certificateBody`

（可选）X.509 证书的正文。

如果您在 `certificateTrust` 中为 `type` 选择 `X509`，则此字段为必填字段。

`certificateChain`

（可选）X.509 证书的信任链。

仅当您在 `certificateTrust` 中为 `type` 选择 `X509` 时使用此字段。

`endpointUri`

OPC UA 源的本地端点。例如，您的本地终端节点可能类似于 `opc.tcp://203.0.113.0:49320`。

`securityPolicy`

要使用的安全策略，可保护从 OPC UA 源读取的消息。选择下列选项之一：

- `NONE`— 边 SiteWise 缘网关不保护来自 OPC UA 来源的消息。建议您选择不同的安全策略。如果选择此选项，则还必须为 `messageSecurityMode` 选择 `NONE`。
- `BASIC256_SHA256` — `Basic256Sha256` 安全政策。
- `AES128_SHA256_RSAOAEP` — `Aes128_Sha256_RsaOaep` 安全政策。
- `AES256_SHA256_RSAPSS` — `Aes256_Sha256_RsaPss` 安全政策。

- BASIC128_RSA15 — （已弃用）OPC UA 规范中已弃用 Basic128Rsa15 安全策略，因为它被视为不再安全。建议您选择不同的安全策略。有关更多信息，请参阅配置文件 SecurityPolicy — Basic 128Rsa15。
- BASIC256 — （已弃用）OPC UA 规范中已弃用 Basic256 安全策略，因为它被视为不再安全。建议您选择不同的安全策略。有关更多信息，请参阅 SecurityPolicy — Basic256。

> **Important**
>
> 如果您选择除 NONE 外的其他安全策略，则 messageSecurityMode 必须选择 SIGN 或 SIGN_AND_ENCRYPT。您还必须将源服务器配置为信任 SiteWise Edge 网关。有关更多信息，请参阅 将 OPC UA 服务器设置为信任 Amazon IoT SiteWise Edge 网关。

messageSecurityMode

用于保护与 OPC UA 源的连接的消息安全模式。选择下列选项之一：

- NONE— 边 SiteWise 缘网关无法保护与 OPC UA 源的连接。建议您选择不同的消息安全模式。如果选择此选项，则还必须为 securityPolicy 选择 NONE。
- SIGN— 在 SiteWise Edge 网关和 OPC UA 源之间传输的数据已签名，但未加密。
- SIGN_AND_ENCRYPT — 对网关和 OPC UA 源之间的传输中数据进行签名和加密。

> **Important**
>
> 如果您选择除 NONE 外的消息安全模式，则必须选择除 NONE 外的 securityPolicy。您还必须将源服务器配置为信任 SiteWise Edge 网关。有关更多信息，请参阅 将 OPC UA 服务器设置为信任 Amazon IoT SiteWise Edge 网关。

identityProvider

一个包含以下信息的身份提供程序结构：

type

源所需的身份验证凭证的类型。选择下列选项之一：

- Anonymous — 源不需要身份验证即可连接。

- `Username` — 源需要用户名和密码才能连接。如果选择此选项，则必须在 `identityProvider` 中定义 `usernameSecretArn`。

`usernameSecretArn`

（可选）密钥的 ARN。 Amazon Secrets Manager SiteWise Edge 网关在连接到此源时使用此密钥中的身份验证凭证。您必须将密钥附加到 SiteWise Edge 网关的 IoT SiteWise 连接器，才能使用它们进行源身份验证。有关更多信息，请参阅 为 SiteWise Edge 配置数据源身份验证。

如果您在 `identityProvider` 中为 `type` 选择 `Username`，则此字段为必填字段。

`nodeFilterRules`

此列表中的每个结构都必须包含以下信息：

`action`

此节点筛选器规则的操作。可以选择以下选项：

- `INCLUDE`— SiteWise Edge 网关仅包含符合此规则的数据流。

`definition`

一个包含以下信息的节点筛选器规则结构：

`type`

此规则的节点筛选器路径的类型。可以选择以下选项：

- `OpcUaRootPath`— SiteWise Edge 网关根据 OPC UA 路径层次结构的根目录评估此节点筛选路径。

`rootPath`

根据 OPC UA 路径层次结构的根进行评估的节点筛选条件路径。此路径必须以 / 开头。

`measurementDataStreamPrefix`

要在源中所有数据流的前面加上的字符串。 SiteWise Edge 网关将此前缀添加到来自该源的所有数据流。使用数据流前缀，可以区分来自不同源的同名数据流。在您的账户中，每个数据流都应具有唯一的名称。

`typeConversions`

不支持的 OPC UA 数据类型可用的转换类型。每种数据类型都转换为字符串。有关更多信息，请参阅 转换不支持的数据类型。

array

转换为字符串的简单数组数据类型。可以选择以下选项：

- JsonArray — 表示您选择将简单数组数据类型转换为字符串。

datetime

转换为字符串 DateTime 的数据类型。可以选择以下选项：

- ISO8601String — 表示您选择将 ISO 8601 数据类型转换为字符串。

destination

OPC UA 标签目标的配置。支持 v2 和 MQTT 的经典直播、V3 网关对目的地的配置不同。

type

目标的类型。

streamName— 仅适用于经典直播、V2 网关

流的名称。流名称应该唯一。

streamBufferSize— 仅适用于经典直播、V2 网关

流的缓冲区大小。这对于管理来自 OPC UA 源的数据流非常重要。

propertyGroups

（可选）定义协议要求的 deadband 和 scanMode 的属性组列表。

name

属性组的名称。这应该是一个唯一的标识符。

deadband

deadband 值定义了在将数据发送到云端之前数据点的值必须发生的最小变化。它包含以下信息：

type

支持的死区类型。可以选择以下选项：

- ABSOLUTE — 一个固定值，它指定了将数据点视为足够重要以发送到云端所需的最小绝对变化。

- PERCENT — 一个动态值，它以占上次发送的数据点值的百分比形式指定所需的最小变化。当数据值随时间变化显著时，这种类型的死区非常有用。

value

死区值。如果 type 是 ABSOLUTE，则此值为无单位的双精度。如果 type 是 PERCENT，则此值是 1 和 100 之间的双精度。

eguMin

（可选）使用 PERCENT 死区时的工程单位最小值。如果 OPC UA 服务器未配置工程单元，则可以设置此项。

eguMax

（可选）使用 PERCENT 死区时的工程单位最大值。如果 OPC UA 服务器未配置工程单元，则可以设置此项。

timeoutMilliseconds

超时前的持续时间，以毫秒为单位。最少为 100。

scanMode

scanMode 结构包含以下信息：

type

支持的 scanMode 类型。接受的值为：POLL 和 EXCEPTION。

rate

扫描模式的采样间隔。

timestampToReturn

时间戳的来源。可以选择以下选项：

- SOURCE_TIME — 使用设备上的时间戳。
- SERVER_TIME — 使用服务器上的时间戳。

> **Note**
>
> TimestampToReturn与 2.5.0 或更高版本的 IoT SiteWise OPC UA 收集器组件一起使用。如果您在早期版本中使用此功能，则配置更新会失败。有关更多信息，请参阅 更新 Amazon IoT SiteWise 组件的版本。

`nodeFilterRuleDefinitions`

（可选）包含在属性组中的节点路径列表。属性组不能重叠。如果您不为此字段指定值，则该组包含根目录下的所有路径，且您无法创建其他属性组。`nodeFilterRuleDefinitions` 结构包含以下信息：

`type`

`OpcUaRootPath` 是唯一支持的类型。这指定 `rootPath` 值是相对于 OPC UA 浏览空间根目录的路径。

`rootPath`

以逗号分隔的列表，用于指定要包含在属性组中的路径（相对于根路径）。

经典流、V2 网关的其他功能配置示例 ()Amazon CLI

以下示例根据存储在 JSON 文件中的有效负载定义了 OPC UA SiteWise Edge 网关功能配置。

```
aws iotsitewise update-gateway-capability-configuration \
--capability-namespace "iotsitewise:opcuacollector:2" \
--capability-configuration file://opc-ua-configuration.json
```

Example ：OPC UA 源配置

以下 `opc-ua-configuration.json` 文件定义了基本的、不安全的 OPC UA 源配置。

```
{
    "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 ：带有已定义属性组的 OPC UA 源配置

以下 `opc-ua-configuration.json` 文件定义了一个基本的、不安全的 OPC UA 源配置，其中包含已定义的属性组。

```
{
    "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 ：带属性的 OPC UA 源配置

针对 `opc-ua-configuration.json` 的以下 JSON 示例定义了具有以下属性的 OPC UA 源配置：

- 信任任何证书。
- 使用 `BASIC256` 安全策略保护消息。
- 使用 `SIGN_AND_ENCRYPT` 模式保护连接。
- 使用存储在 Secrets Manager 密钥中的身份验证凭证。
- 筛选出数据流（路径以 `/WindFarm/2/WindTurbine/` 开头的数据流除外）。
- 在每个数据流路径的开头添加 `/Washington` 以将此“Wind Farm #2”与另一个区域中的“Wind Farm #2”区分开来。

```
{
    "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:greengrass-windfarm2-auth-1ABCDE"
                },
                "nodeFilterRules": [
                  {
                      "action": "INCLUDE",
                      "definition": {
                          "type": "OpcUaRootPath",
                          "rootPath": "/WindFarm/2/WindTurbine/"
                    }
                  }
                ]
            },
            "measurementDataStreamPrefix": "/Washington"
        }
```

```
    ]
}
```

Example ：具有证书信任的 OPC UA 源配置

针对 `opc-ua-configuration.json` 的以下 JSON 示例定义了具有以下属性的 OPC UA 源配置：

- 信任给定的 X.509 证书。
- 使用 `BASIC256` 安全策略保护消息。
- 使用 `SIGN_AND_ENCRYPT` 模式保护连接。

```
{
    "sources": [
        {
            "name": "Wind Farm #3",
            "endpoint": {
                "certificateTrust": {
                    "type": "X509",
                    "certificateBody": "-----BEGIN CERTIFICATE-----
          MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
          -----END CERTIFICATE-----",
                    "certificateChain": "-----BEGIN CERTIFICATE-----
          MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
```

```
WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
3rrszlaEXAMPLE=
         -----END CERTIFICATE-----"
               },
               "endpointUri": "opc.tcp://203.0.113.2:49320",
               "securityPolicy": "BASIC256",
               "messageSecurityMode": "SIGN_AND_ENCRYPT",
               "identityProvider": {
                   "type": "Anonymous"
               },
               "nodeFilterRules": []
             },
           "measurementDataStreamPrefix": ""

        }
    ]
}
```


将 OPC UA 服务器设置为信任 Amazon IoT SiteWise Edge 网关

如果在配置 OPC UA 源时选择 “无” 以`messageSecurityMode`外的选项，则必须使源服务器能够信任 Amazon IoT SiteWise Edge 网关。E SiteWise dge 网关会生成您的源服务器可能需要的证书。过程因源服务器而异。有关更多信息，请参阅您的服务器的文档。

以下过程概述了基本步骤。

使 OPC UA 服务器能够信任 SiteWise Edge 网关

1. 打开用于配置 OPC UA 服务器的界面。
2. 输入 OPC UA 服务器管理员的用户名和密码。
3. 在界面中找到可信客户端，然后选择Amazon IoT SiteWise 网关客户端。
4. 选择 信任。

导出 OPC UA 客户端证书

某些 OPC UA 服务器需要访问 OPC UA 客户端证书文件才能信任 SiteWise Edge 网关。如果这适用于您的 OPC UA 服务器，则可以使用以下步骤从 SiteWise Edge 网关导出 OPC UA 客户端证书。然后，可以在您的 OPC UA 服务器上导入证书。

要导出某个来源的 OPC UA 客户端证书文件，请执行以下步骤：

1. 运行以下命令来更改为包含证书文件的目录。*sitewise-work*替换为 *aws.iot.SiteWiseEdgeCollectorOpcua* Greengrass 工作文件夹的本地存储路径，并*source-name*替换为数据源的名称。

   默认情况下，Greengrass 工作文件夹*/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua*在 Linux 及以上版本上。*C:/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua* Microsoft Windows

   ```
   cd /sitewise-work/source-name/opcua-certificate-store
   ```

2. 该来源的 SiteWise Edge 网关的 OPC UA 客户端证书位于`aws-iot-opcua-client.pfx`文件中。

   运行以下命令来将证书导出到名为 `aws-iot-opcua-client-certificate.pem` 的 `.pem` 文件。

   ```
   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. 将证书文件从 SiteWise Edge 网关传输到 OPC UA 服务器。`aws-iot-opcua-client-certificate.pem`

   为此，您可以使用常用软件（如 `scp` 程序）来通过 SSH 协议传输文件。有关更多信息，请参阅 Wikipedia 上的 Secure copy。

   > **Note**
   >
   > 如果您的 SiteWise Edge 网关在亚马逊弹性计算云 (Amazon EC2) 上运行，并且您是首次连接到该网关，则必须配置连接的先决条件。有关更多信息，请参阅亚马逊 EC2 用户指南中的使用 SSH 连接您的 Linux 实例。

4. 在 OPC UA 服务器上导入证书文件以信任 SiteWise Edge 网关。`aws-iot-opcua-client-certificate.pem`步骤因所使用的源服务器而异。请参阅服务器的文档。

使用 OPC UA 筛选数据摄取范围

您可以使用扫描模式和死区范围来控制通过 OPC UA 源摄取数据的方式。这些功能使您可以控制要采集的数据类型，以及服务器和 Edg SiteWise e 网关交换这些信息的方式和时间。

根据质量收集或筛选数据

您可以配置数据质量设置，以控制从 OPC UA 源收集哪些数据。数据来源会在发送数据时将质量评级作为元数据包括在内。您可以选择下列选项之一或下列全部选项：

- `Good`
- `Bad`
- `Uncertain`

处理 NaN 或空值

SiteWise Edge 支持对 NaN 和空值的收集和处理。

- NaN（不是数字）：表示未定义或无法表示的数值结果。
- 空：表示缺少数据。

物联网 SiteWise OPC UA 收集器捕获质量差或不确定的 NaN 和空值。这些特殊值会写入本地流，从而实现更全面的数据收集。

使用扫描模式控制数据收集频率

您可以配置 OPC UA 扫描模式来控制从 OPC UA 源收集数据的方式。您可以选择订阅或轮询模式。

- 订阅模式 — OPC UA 源收集数据，按照您的扫描速率定义的频率发送到您的 SiteWise Edge 网关。服务器仅在值发生变化时才发送数据，因此这是您的 SiteWise Edge 网关接收数据的最大频率。
- 轮询模式 — 您的 SiteWise Edge 网关以您的扫描速率定义的设定频率轮询 OPC UA 源。无论值是否发生变化，服务器都会发送数据，因此您的 SiteWise Edge 网关始终以此间隔接收数据。

> **Note**
> 轮询模式选项会覆盖此源的死区设置。

使用死区范围筛选 OPC UA 数据摄取

> **Note**
> 您的服务器会删除死区指定的窗口内的所有数据。您无法恢复此丢弃的数据。

死区类型

您可以为 OPC UA 服务器属性组指定两种类型的死区。它们允许您选择将发送到 Amazon 云的数据，以及丢弃的数据。

- 百分比 — 使用测量值中预期波动的百分比来指定窗口。服务器根据此百分比计算出确切的窗口，并将超出窗口范围的数据发送到 Amazon 云端。例如，在传感器上指定 2% 的死区值，范围从 -100 华氏度到 +100 华氏度，会告诉服务器在值变化 4 华氏度或更高时将数据发送到 Amazon 云端。

  > **Note**
  > 如果您的源服务器未定义工程单位，则可以选择为此窗口指定最小和最大死区值。如果未提供工程单位范围，OPC UA 服务器将默认为测量值数据类型的完整范围。

- 绝对 — 使用精确单位指定窗口。例如，在传感器上将死区值指定为 2，告诉服务器在数据值变化至少 2 个单位时将数据发送到 Amazon 云端。在正常操作期间经常会出现波动的动态环境中，您可以使用绝对死区。

死区超时

您可以选择配置死区超时设置。在此超时之后，即使当前测量值处于预期的死区波动范围内，OPC UA 服务器也将发送当前测量值。您可以使用超时设置来确保 Amazon IoT SiteWise 始终采集稳定的数据流，即使值不超过定义的死区窗口也是如此。

在 SiteWise Edge 中使用 OPC UA 节点过滤器

在为 SiteWise Edge 网关定义 OPC UA 数据源时，可以定义节点筛选器。节点过滤器允许您限制 SiteWise Edge 网关发送到云的数据流路径。您可以使用节点筛选器通过仅包含您建模的数据的路径来减少 SiteWise Edge 网关的启动时间和 CPU 使用率 Amazon IoT SiteWise。默认情况下， SiteWise 边缘网关会上传除以开头的路径之外的所有 OPC UA 路径。`/Server/`您可以使用在节点筛选条件中使用 `*` 和 `**` 通配符来包含多个数据流路径和一个筛选条件。要了解如何为 SiteWise Edge 网关设置 OPC UA 来源，请参阅Amazon IoT SiteWise 边缘网关的 OPC UA 数据源。

> **Note**
>
> Amazon IoT SiteWise 每次添加或编辑源时都会重新启动 SiteWise Edge 网关。您的 SiteWise Edge 网关在更新源配置时不会提取数据。重启 SiteWise Edge 网关的时间取决于 E SiteWise dge 网关源上的标签数量。重启时间可以从几秒钟（对于标签较少的 SiteWise Edge 网关）到几分钟（对于具有许多标签的 SiteWise Edge 网关）不等。

下表列出了可用于筛选 OPC UA 数据来源的通配符。

OPC UA 节点筛选条件通配符

通配符	描述
*	匹配数据流路径中的单个级别。
**	匹配数据流路径中的多个级别。

> **Note**
>
> 如果您为源配置了宽筛选器，然后将源更改为使用更严格的筛选器，则 Amazon IoT SiteWise 会停止存储与新筛选器不匹配的数据。

Example：使用节点过滤器的场景

考虑以下各项假定数据流：

- `/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

使用前面的数据流，您可以定义节点筛选条件，限制 OPC UA 源中要包含的数据。

- 要选择此示例中的所有节点，请使用 / 或 /**/。您可以使用 ** 通配符包含多个目录或文件夹。
- 要选择所有 PLC 数据流，请使用 /*/*/*/PLC* 或 /**/PLC*。
- 要选择此示例中的所有计数器，请使用 /**/Counter* 或 /*/*/*/Counter*。
- 要选择 Line 2 中的所有计数器，请使用 /**/Line 2/Counter*。

转换不支持的数据类型

（可选）Amazon IoT SiteWise 为简单数组和 DateTime 数据类型启用数据类型转换。 Amazon IoT SiteWise 不支持所有 OPC UA 数据类型。当您将不支持的数据发送到 Amazon IoT Greengrass 数据流时，该数据就会丢失。但是，通过将不支持的本机数据类型转换为字符串，您可以将数据摄取到 Amazon IoT SiteWise 而不是丢弃它。 Amazon IoT SiteWise 序列化转换后的数据，以便以后可以根据需要使用自己的函数在下游将字符串转换回其原始数据类型。

您可以随时更新数据来源的数据类型转换设置，并且每个数据来源都可以具有自己的设置。

在中添加数据源时 Amazon IoT SiteWise 控制台，“高级配置” 中的 “数据类型转换” 下有两个复选框。您可以指明要将哪些数据类型转换为字符串。

此外，物联网 SiteWise OPC UA 收集器可以在边缘接受 NaN 或空值。

- 将具有简单数据类型的数组值转换为 JSON 字符串
- 将 DateTime 值转换为 ISO 8601 字符串

先决条件

- 使用 2.5.0 或更高版本的 IoT SiteWise OPC UA 收集器。

限制

这些是 Amazon IoT SiteWise中对 OPC UA 数据类型转换为字符串的限制。

- 不支持复杂数据类型转换。
- 转换后的字符串限制为 1024 字节。如果字符串长度超过 1024 字节，则该字符串将被拒绝 Amazon IoT SiteWise。

为 SiteWise Edge 配置数据源身份验证

如果您的 OPC UA 服务器需要身份验证凭据才能连接，则可以使用创建密钥并将其部署 Amazon Secrets Manager 到您的 SiteWise Edge 网关。 Amazon Secrets Manager 对设备上的机密进行加密，以确保您的用户名和密码的安全，直到您需要使用它们为止。有关 Amazon IoT Greengrass 密钥管理器组件的更多信息，请参阅《Amazon IoT Greengrass Version 2 开发者指南》中的密钥管理器。

有关管理 Secrets Manager 密钥访问权限的信息，请参阅：

- 谁有权访问你的 Amazon Secrets Manager 秘密。
- 确定账户内的请求是被允许还是被拒绝。

步骤 1：创建源身份验证密钥

您可以使用 Amazon Secrets Manager 为数据源创建身份验证密钥。在密钥中，定义 **`username`** 和 **`password`** 密钥/值对，其中包含数据来源的身份验证详细信息。

创建密钥（控制台）

1. 导航至 Amazon Secrets Manager 控制台。
2. 选择存储新密钥。
3. 在密钥类型下，选择其他密钥类型。
4. 在键/值对下，执行以下操作：
   1. 在第一个输入框中输入 **`username`**，在第二个输入框中输入用户名。
   2. 选择添加行。
   3. 在第一个输入框中输入 **`password`**，在第二个输入框中输入密码。
5. 为加密密钥选择 aws/secretsmanager，然后选择下一步。
6. 在存储新密钥页面，输入密钥名称。
7. （可选）输入可帮助您识别此密钥的描述，然后选择下一步。
8. （可选）在存储新密钥页面，启用自动轮换。有关更多信息，请参阅 Amazon Secrets Manager 用户指南中的轮换密钥。

9. 指定轮换计划。
10. 选择可以轮换此密钥的 Lambda 函数，然后选择下一步。
11. 检查您的密钥配置，然后选择存储。

要授权您的 SiteWise Edge 网关与之交互 Amazon Secrets Manager，您的 SiteWise Edge 网关的 IAM 角色必须允许该`secretsmanager:GetSecretValue`操作。您可以使用 Greengrass 核心设备来搜索 IAM 策略。有关更新 IAM 策略的更多信息，请参阅Amazon Identity and Access Management 用户指南中的编辑 IAM 策略。

Example policy

*secret-arn*替换为您在上一步中创建的密钥的 Amazon 资源名称 (ARN)。有关如何获取密钥的 ARN 的更多信息，请参阅Amazon Secrets Manager 用户指南Amazon Secrets Manager中的查找密钥。

```
{
"Version":"2012-10-17",
"Statement":[
  {
     "Action":[
        "secretsmanager:GetSecretValue"
     ],
     "Effect":"Allow",
     "Resource":[
        "secret-arn"
     ]
  }
]
}
```

步骤 2：将密钥部署到您的 SiteWise Edge 网关设备

您可以使用 Amazon IoT SiteWise 控制台将密钥部署到您的 SiteWise Edge 网关。

部署密钥（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择网关。
3. 从网关列表中，选择目标 SiteWise Edge 网关。
4. 在网关配置部分，选择 Greengrass 核心设备链接以打开与 Edge 网关关联 Amazon IoT Greengrass 的核心。 SiteWise

5. 在导航窗格中，选择部署。
6. 选择目标部署，然后选择修订。
7. 在指定目标页面，选择下一步。
8. 在选择组件页面的公共组件部分，关闭仅显示所选组件。
9. 搜索并选择 aws.greengrass。 SecretManager组件，然后选择 “下一步”。
10. 从 “选定组件” 列表中选择 aws.greengrass。 SecretManager组件，然后选择配置组件。
11. 在要合并的配置字段，添加以下 JSON 对象。

> **Note**
>
> *secret-arn*替换为您在上一步中创建的密钥的 ARN。有关如何获取密钥的 ARN 的更多信息，请参阅Amazon Secrets Manager 用户指南Amazon Secrets Manager中的查找密钥。

```
{
"cloudSecrets":[
  {
     "arn":"secret-arn"
  }
]
}
```

12. 选择确认。
13. 选择下一步。
14. 在配置高级设置页面上，选择下一步。
15. 查看您的部署配置，再选择部署。

步骤 3：添加身份验证配置

您可以使用 Amazon IoT SiteWise 控制台向 SiteWise Edge 网关添加身份验证配置。

添加身份验证配置（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 从网关列表中，选择目标 SiteWise Edge 网关。

3. 从数据来源列表中选择目标数据来源，再选择编辑。
4. 在添加数据来源页面选择高级配置。
5. 对于身份验证配置，请选择您在上一步中部署的密钥。
6. 选择保存。

Amazon IoT GreengrassAmazon IoT SiteWise Edge 的组件

SiteWise Edge 使用 Amazon IoT Greengrass 组件在边缘收集、处理和传输工业数据。这些组件协同工作，可实现本地数据处理并与 Amazon IoT SiteWise 云服务无缝集成。

物联网 SiteWise 发行商

物联网 SiteWise 发布者组件 (`aws.iot.SiteWiseEdgePublisher`) 负责：

- 安全地将收集的数据传输到 Amazon IoT SiteWise 云服务
- 在连接问题期间管理数据缓冲和重试

有关为 SiteWise Edge 配置发布器的更多信息，请参阅配置 Amazon IoT SiteWise 发布者组件。而且，有关发布者组件的更多信息，请参阅《Amazon IoT Greengrass Version 2 开发者指南》中的 IoT SiteWise 发布者。

物联网 SiteWise 处理器

物联网 SiteWise 处理器组件 (`aws.iot.SiteWiseEdgeProcessor`) 执行以下任务：

- 在边缘执行数据转换和计算
- 在本地实现资产属性定义和计算
- 通过在传输前聚合或筛选数据来减少数据量

有关处理器组件的更多信息，请参阅《Amazon IoT Greengrass Version 2 开发者指南》中的 IoT SiteWise 处理器。

物联网 SiteWise OPC UA 采集器

物联网 SiteWise OPC UA 收集器 (`aws.iot.SiteWiseEdgeCollectorOpcua`) 组件设计用于：

- 在工业环境中连接 OPC UA 服务器
- 高效地从 OPC UA 数据源收集数据
- 将 OPC UA 数据转换为兼容的格式 Amazon IoT SiteWise

有关 OPC UA 收集器组件的更多信息，请参阅《Amazon IoT Greengrass Version 2 开发者指南》中的 IoT SiteWise OPC UA 收集器。

物联网 SiteWise OPC UA 数据源模拟器

IoT SiteWise OPC UA 数据源模拟器组件 (`aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator`) 提供以下功能：

- 启动生成示例数据的本地 OPC UA 服务器
- 模拟网关上的 Amazon IoT SiteWise OPC UA 收集器组件可以读取的数据源 Amazon IoT SiteWise
- 允许使用生成的样本数据探索 Amazon IoT SiteWise 特征

该组件对于测试和开发目的特别有用，使您无需使用物理设备即可模拟工业数据源。

有关数据源仿真组件的更多信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的 IoT SiteWise OPC UA 数据源模拟器。

这些 Amazon IoT Greengrass 组件用于启用 SiteWise Edge 功能。物联网 SiteWise 发布者确保数据可靠地发送到云端，物联网 SiteWise 处理器负责本地计算和数据优化，物联网 SiteWise OPC UA 收集器可促进与常见工业协议的集成。

> **Note**
>
> 要使用这些组件，必须先在边缘设备上安装 Amazon IoT Greengrass V2 或稍后安装。每个组件的正确配置对于 SiteWise Edge 的最佳性能非常重要。

筛选 SiteWise Edge 网关上的资产

您可以使用边缘筛选来更有效地管理资产，方法是仅将一部分资产发送到特定的 SiteWise 边缘网关以用于数据处理。如果您的资产按树结构或父子结构排列，则可以设置附加到 SiteWise Edge 网关的 IAM 角色的 IAM 策略，该策略仅允许将树的根（父级）及其子项发送到特定的 SiteWise Edge 网关。

> **Note**
>
> 如果要将现有资源排列为树形结构，则在创建结构后，请进入添加到结构中的每项现有资源并选择 “编辑”，然后选择 “保存” 以确保 Amazon IoT SiteWise 识别新结构。

设置边缘筛选

在 Edge 网关上设置 SiteWise 边缘筛选，方法是将以下 IAM 策略添加到 SiteWise Edge 网关的 IAM 角色，*<root-asset-id>*替换为要发送到 SiteWise Edge 网关的根资产的 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>*"
                }
            }
        }
    ]
}
```

如果您的 SiteWise Edge 网关上当前有您想要移除的资产，请登录您的 SiteWise Edge 网关并运行以下命令， Amazon IoT SiteWise 通过删除缓存来强制 SiteWise Edge 网关与之同步。

```
sudo rm /greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/sync-app/
sync_resource_bundles/edge.json
```

为 E Amazon IoT SiteWise dge 配置代理支持并管理信任存储

在 Amazon IoT SiteWise Edge 中，配置和管理信任存储库，为边缘设备设置代理支持。首先，设置代理配置，然后配置信任存储。您可以在网关安装期间配置信任存储，也可以在网关建立后手动配置信任存储。

- 代理-在各种网络环境中促进边缘设备和 Amazon 服务之间的连接。
- 信任存储-通过管理可信证书来确保连接安全。适当的配置可帮助您遵守网络安全策略，在受限的网络环境中实现通信，并优化边缘设备和云服务之间的数据传输。

SiteWise Edge 针对不同的组件类型使用多个信任存储，确保数据从边缘设备安全高效地流向云端。您可以在现有网关上配置信任存储和代理，也可以在安装过程中创建新网关时配置信任存储和代理。

信任存储和代理配置的要求

在配置信任存储或使用代理设置安装 SiteWise Edge 之前，请确保满足先决条件。根据您的组件使用情况和功能要求，有不同的实现要求。

代理支持要求

- 您的代理服务器的 URL。URL 应包含用户信息，即主机的端口号。例如 `scheme://[userinfo@]host[:port]`。
  - `scheme`— 必须是 HTTP 或 HTTPS
  - （可选）`userinfo`-用户名和密码信息
  - `host`— 代理服务器的主机名或 IP 地址
  - `port`— 端口号
- 绕过代理的地址列表。
- （可选）如果您使用带有自签名证书的 HTTPS 代理，则为代理 CA 证书文件。

信托商店要求

- 要使用 HTTPS 代理获得完整的数据处理包功能，您应该更新所有三个信任存储库。
- 如果您仅使用 IoT SiteWise OPC UA 收集器和物联网 SiteWise 发布器，请将证书 C Amazon IoT Greengrass ore 和 Java 信任存储更新到最新版本。

信任存储和代理服务器边缘配置的最佳实践

要进行持续维护并保持边缘环境的最高安全级别，请执行以下操作：

- 定期检查和更新代理设置，使其符合您的网络安全要求。
- 监控网关连接和数据流，确保代理通信正确
- 根据贵组织的证书管理政策维护和更新信任存储
- 您可以实施并遵循我们推荐的最佳实践，以便在边缘环境中实现安全通信，例如：
- 记录您的代理和信任存储库配置，以实现运营可见性
- 按照贵组织的安全惯例进行凭证管理

这些做法有助于保持 SiteWise Edge 网关的安全可靠运行，同时与更广泛的安全策略保持一致。

在 Amazon IoT SiteWise Edge 网关安装期间配置代理设置

您可以将 Amazon IoT SiteWise Edge 配置为在网关安装期间使用代理服务器。安装脚本同时支持 HTTP 和 HTTPS 代理，并且可以自动配置信任存储以实现安全的代理连接。

使用代理设置运行安装脚本时，它会执行几项重要任务：

- 验证代理 URL 格式和参数，确保它们指定正确。
- 通过配置的代理下载并安装所需的依赖项。
- 如果提供了代理 CA 证书，则会将其附加到 Amazon IoT Greengrass 根 CA 证书并导入到 Java KeyStore 中。
- 将 Amazon IoT Greengrass （ SiteWise Edge 使用的）配置为对所有出站连接使用代理。
- 使用适当的代理和信任存储配置完成 SiteWise Edge 安装。

在安装网关软件时配置代理设置

1. 创建 SiteWise Edge 网关。有关更多信息，请参阅创建自托管 SiteWise Edge 网关和在本地设备上安装 Amazon IoT SiteWise Edge 网关软件。
2. 使用适合您环境的代理设置运行安装脚本。用您的特定代理信息替换占位符

   替换以下各项：

   - `-p`，`--proxy-url`— 代理服务器的 URL。网址必须为`http`或`https`。
   - `-n`，`--no-proxy`— 以逗号分隔的地址列表，用于绕过代理。
   - （可选）`-c`，`--proxy-ca-cert`— 代理 CA 证书文件的路径。
   - （可选）`-j`，`--javastorepass`— Java KeyStore 密码。默认密码为 `changeit`。

   Linux

   对于 Linux 系统，请使用以下命令结构：

```
sudo ./install.sh -p proxy-url -n no-proxy-addresses [-c proxy-ca-cert-path] [-
j javastorepass]
```

Windows

对于使用的Microsoft Windows系统 PowerShell，请使用以下命令结构：

```
.\install.ps1 -ProxyUrl proxy-url -NoProxyAddresses no-proxy-addresses [-
ProxyCaCertPath proxy-ca-cert-path] [-JavaStorePass javastorepass]
```

在启用代理的安装过程中进行故障排除

有关解决与 SiteWise Edge 网关相关的信任存储问题的更多信息，请参阅支持代理的安装问题。

在 E Amazon IoT SiteWise dge 中手动配置信任存储以支持 HTTPS 代理

将 Amazon IoT SiteWise Edge 组件配置为通过 HTTPS 代理进行连接时，请将代理服务器的证书添加到相应的信任存储中。 SiteWise Edge 使用多个信任存储库来保护通信。有三个信任存储库，您对它们的使用取决于网关实现中的 SiteWise Edge 组件类型。

如果提供了代理设置，则会在安装过程中自动更新信任存储库。

- 配置 Amazon IoT Greengrass 核心组件信任存储— Amazon IoT Greengrass 根CA证书包含在信任库中，用于验证 Amazon 服务的真实性。

  此信任存储可帮助 Amazon IoT Greengrass 组件通过代理安全地与 Amazon 服务通信，同时验证这些服务的真实性。
- 配置基于 Java 的组件信任存储— Java KeyStore (JKS) 是基于 Java 的组件用于 SSL/TLS 连接的主要信任存储库。

  Java 应用程序依靠 JKS 来建立安全连接。例如，如果您使用的是基于 Java 的 IoT SiteWise 发布者或 IoT SiteWise OPC UA 收集器，则需要配置此信任存储库。这样可以确保在向云端发送数据或从 OPC UA 服务器收集数据时，这些组件可以安全地通过 HTTPS 代理进行通信。
- 系统级组件信任存储配置— 使用 HTTPS 代理时，必须将其证书添加到相应的信任存储库中以启用安全连接。

  使用 HTTPS 代理时，必须将其证书添加到相应的信任存储中，以启用安全连接。这是必要的，因为通常用 Rust 或 Go 等语言编写的系统级组件依赖于系统的信任存储库，而不是 Java 的 JKS。例如，如果您使用的系统实用程序需要通过代理进行通信（例如软件更新或时间同步），则需要配置系统级信任存储。这样可以确保这些组件和实用程序可以通过代理建立安全连接。

配置 Amazon IoT Greengrass 核心组件信任存储

对于使用亚马逊根 CA 的 Amazon IoT Greengrass 核心函数：

1. 在以下位置找到证书文件 `/greengrass/v2/AmazonRootCA1.pem`
2. 将 HTTPS 代理根证书（自签名）附加到此文件中。

```
-----BEGIN CERTIFICATE-----
MIIEFTCCAv2gAwIQWgIVAMHSAzWG/5YVRYtRQOxXUTEpHuEmApzGCSqGSIb3DQEK
\nCwUAhuL9MQswCQwJVUzEPMAVUzEYMBYGA1UECgwP1hem9uLmNvbSBJbmMuMRww
... content of proxy CA certificate ...
+vHIRlt0e5JAm5\noTIZGoFbK82A0/nO7f/t5PSIDAim9V3Gc3pSXxCCAQoFYnui
GaPUlGk1gCE84a0X\n7Rp/lND/PuMZ/s8YjlkY2NmYmNjMCAXDTE5MTEyN2cM216
gJMIADggEPADf2/m45hzEXAMPLE=
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIDQTCCAimgF6AwIBAgITBmyfz/5mjAo54vB4ikPmljZKyjANJmApzyMZFo6qBg
ADA5MQswCQYDVQQGEwJVUzEPMA0tMVT8QtPHRh8jrdkGA1UEChMGDV3QQDExBBKW
... content of root CA certificate ...
o/ufQJQWUCyziar1hem9uMRkwFwYVPSHCb2XV4cdFyQzR1KldZwgJcIQ6XUDgHaa
5MsI+yMRQ+hDaXJiobldXgjUka642M4UwtBV8oK2xJNDd2ZhwLnoQdeXeGADKkpy
rqXRfKoQnoZsG4q5WTP46EXAMPLE
-----END CERTIFICATE-----
```

在已建立的网关上配置 HTTPS 代理

您可以通过连接到端口 443 而不是端口 8883 来为已建立的网关添加代理支持。有关使用代理服务器的更多信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的通过端口 443 或通过网络代理进行连接。如果您创建了新的网关，则可以在网关安装期间设置代理配置。有关更多信息，请参阅在 Amazon IoT SiteWise Edge 网关安装期间配置代理设置。

当你 Amazon IoT Greengrass 在 SiteWise Edge 上使用 HTTPS 代理时，软件会根据提供的网址自动在 HTTP 和 HTTPS 之间进行代理连接。

> **Important**
>
> 在尝试通过 HTTPS 代理进行连接之前，请更新所有必需的信任存储库。

配置基于 Java 的组件信任存储

对于数据处理包中的物联网 SiteWise 发行商、物联网 SiteWise OPC UA 收集器和 Java 服务，默认 Java 信任存储位置为 `$JAVA_HOME/jre/lib/security/cacerts`

添加证书

1. 创建用于存储代理服务器证书的文件，例如`proxy.crt`。

   > **Note**
   > 使用代理服务器的证书提前创建文件。

2. 使用以下命令将文件添加到 Java 的信任存储区：

   ```
   sudo keytool -import -alias proxyCert -keystore /usr/lib/jvm/java-11-openjdk-amd64/
   lib/security/cacerts -file proxy.crt
   ```

3. 出现提示时，使用默认密码：`changeit`

系统级组件信任存储配置

对于用 Rust、Go 和其他使用系统信任存储库的语言编写的组件：

Linux

Linux 系统：将证书添加到 `/etc/ssl/certs/ca-certificates.crt`

Windows

Microsoft Windows系统：要配置信任存储库，请按照 Microsoft Ignite 文档中的证书存储区步骤进行操作。

Windows 提供多个证书存储区，包括用于用户和计算机范围的单独存储区，每个存储区都有多个子存储区。对于大多数 SiteWise Edge 设置，我们建议在`COMPUTER | Trusted Root Certification Authorities`商店中添加证书。但是，根据您的特定配置和安全要求，您可能需要使用不同的商店。

对信任存储库问题进行故障排除

有关解决与 SiteWise Edge 网关相关的信任存储问题的更多信息，请参阅信任商店问题。

Amazon IoT SiteWise APIs 在边缘使用

Amazon IoT SiteWise 提供了其 APIs中的一个子集以及特定于边缘的资源 APIs，从而实现了与部署在边缘的资产模型及其关联资产的无缝交互。这些资产模型必须配置为在边缘运行。有关更多信息，请参阅在 E SiteWise dge 上配置用于数据处理的资产模型，获取此设置流程的详细说明。

配置这些之后 APIs，您可以检索有关您的资产模型和单个资产的全面数据。检索资产模型、资产、控制面板、门户和项目信息可以帮助您监控已部署的门户和控制面板，并访问在边缘级别收集的资产数据。这在您的网络中提供了一个中央主机， Amazon IoT SiteWise 无需调用 Web API 即可与之进行交互。

主题

- 所有可用的 Amazon IoT SiteWise Edge 设备 APIs
- 仅限 Edge APIs ，适用于边缘设备 Amazon IoT SiteWise
- 在 Edge 上 Amazon IoT SiteWise 启用 CORS APIs
- 为 Edge 配置会话超时 Amazon IoT SiteWise
- 教程：在 Amazon IoT SiteWise Edge 网关上列出资产模型

所有可用的 Amazon IoT SiteWise Edge 设备 APIs

Amazon IoT SiteWise 提供了多种可在边缘设备上使用的选项，以便您可以在设备上本地完成任务。APIs 一些可用的优势 APIs 包括检索资产模型、创建和更新资产属性以及将数据流发送到云端。通过利用这些功能 APIs，您可以构建可在网络连接断断续续或有限的环境中运行的解决方案。

可用 Amazon IoT SiteWise APIs

边缘设备上提供 Amazon IoT SiteWise APIs 以下功能：

- ListAssetModels
- DescribeAssetModel
- ListAssets
- DescribeAsset
- DescribeAssetProperty
- ListAssociatedAssets
- GetAssetPropertyAggregates
- GetAssetPropertyValue

- GetAssetPropertyValueHistory
- ListDashboards
- ListPortals
- ListProjectAssets
- ListProjects
- DescribeDashboard
- DescribePortal
- DescribeProject

仅限边缘可用 APIs

APIs 以下内容在边缘设备上本地使用：

- 身份验证 — 使用此 API 获取用于进行 API 调用的 Sigv4 临时凭证。

仅限 Edge APIs ，适用于边缘设备 Amazon IoT SiteWise

除了边缘上可用的之外，还有一些特定于边缘的。 Amazon IoT SiteWise APIs 这些边缘特定的内容如下 APIs 所述。

身份验证

从 SiteWise Edge 网关获取证书。您需要添加本地用户或使用 LDAP 或 Linux 用户群体连接到您的系统。有关添加用户的更多信息，请参阅 LDAP 或 Linux 用户群体。

请求语法

```
POST /authenticate HTTP/1.1
Content-type: application/json
{
  "username": "string",
  "password": "string",
  "authMechanism": "string"
}
```

URI 请求参数

该请求不使用任何 URI 参数。

请求正文

请求接受采用 JSON 格式的以下数据。

username

用于验证请求调用的用户名。

类型：字符串

必需：是

password

请求凭证的用户密码。

类型：字符串

必需：是

authMechanism

在主机中验证此用户的身份验证方法。

类型：字符串

有效值：`ldap`、`linux`、`winnt`

必需：是

响应语法

```
HTTP/1.1 200
Content-type: application/json
{
  "accessKeyId": "string",
  "secretAccessKey": "string",
  "sessionToken": "string",
  "region": "edge"
}
```

响应元素

如果此操作成功，则该服务将会发送回 HTTP 200 响应。

以下数据以 JSON 格式返回。

accessKeyId

用于标识临时安全凭证的访问密钥 ID。

长度限制：最小长度为 16。最大长度为 128。

模式：`[\w]*`

secretAccessKey

用于签署请求的秘密访问密钥。

类型：字符串

sessionToken

用户必须传递给服务 API 才能使用临时凭证的令牌。

类型：字符串

区域

您针对 API 调用的目标区域。

类型：常量 — `edge`

错误

IllegalArgumentException

由于提供的正文文件格式错误，请求被拒绝。错误消息描述了具体的错误。

HTTP 状态代码：400

AccessDeniedException

根据当前的身份提供商，用户无有效凭证。错误消息描述了身份验证机制。

HTTP 状态代码：403

TooManyRequestsException

该请求已达到身份验证尝试次数上限。错误消息包含在进行新的身份验证尝试之前需要等待的时间。

HTTP 状态代码：429

在 Edge 上 Amazon IoT SiteWise 启用 CORS APIs

在 E Amazon IoT SiteWise dge 上启用 CORS（跨源资源共享）APIs 允许 Web 应用程序 APIs 跨不同域直接与之通信。这可以实现无缝集成、实时数据交换和跨域数据访问，无需中间服务器或变通方法。可以将 CORS 设置配置为指定允许的来源，从而确保受控的跨域访问。

> **Note**
>
> CORS 可用于 3.3.1 及更高版本。此功能适用于该组件的 3.3.1 及更高版本。`aws.iot.SiteWiseEdgeProcessor`有关更多信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的Amazon IoT SiteWise 处理器。

在 Edge 上 SiteWise 启用 CORS APIs

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要为其启用 CORS 的 SiteWise Edge 网关。您可以对 Amazon IoT Greengrass V2 部署类型启用 CORS。
4. 在网关配置部分，选择关联的 Greengrass 核心设备。
5. 在 “部署” 选项卡的 Greengrass 设备下，选择相应的部署链接。
6. 在 “操作” 下选择 “修订”，然后选择 “修订部署”。

   > **Important**
   >
   > 创建经过修订的启用 CORS 的配置会替换设备的当前配置。

7. 在步骤 1 “指定目标” 中，提供一个可选的名称来标识部署。
8. 在步骤 2，选择组件（可选）中，您可以将所有当前选择保持不变，然后选择 “下一步”。
9. 在步骤 3 “配置组件-可选” 中，选择 aws.iot。 SiteWiseEdgeProcessor，然后选择配置组件。
10. 在配置更新部分的要合并的配置下，输入以下 JSON：

```
{
```

```
    "AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN": "*"
}
```

> **Note**
>
> 使用*作为的值AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN允许所有来源。对于生产环境，为了提高安全性，建议指定确切 URLs 的来源。

11. 选择确认。
12. 选择 “下一步” 继续完成剩余的步骤，直到进入步骤 5 “查看”。
13. 查看您的配置更改，然后选择 Deploy 将更改应用到您的 SiteWise Edge 网关。

> **Note**
>
> 或者，您可以通过在 Amazon IoT SiteWise 网关*上将环境变量设置为全局AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN来启用 CORS。

> **Note**
>
> 对于经过身份验证的代理，userinfo必须包含在代理配置的url字段中，而不是作为分隔的username和password字段。

部署完成后，将在您的 SiteWise Edge API 上启用 CORS，从而允许指定的源向该 API 发出跨域请求。

为 Edge 配置会话超时 Amazon IoT SiteWise

SiteWise Edge 允许您为 Ed SiteWise ge API 配置会话超时。此功能通过在指定时间段后自动终止非活动会话来增强安全性。本节将指导您完成使用配置会话超时的过程 Amazon IoT SiteWise 控制台。

> **Note**
>
> 会话超时配置适用于该aws.iot.SiteWiseEdgeProcessor组件的 3.4.0 及更高版本。有关更多信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的Amazon IoT SiteWise 处理器。

为 SiteWise Edge 网关配置会话超时

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要在其中配置会话超时的 SiteWise Edge 网关。

> **Note**
> 您可以根据 Amazon IoT Greengrass V2 部署类型配置会话超时。

4. 在网关配置部分，选择关联的 Greengrass 核心设备。
5. 在 “部署” 选项卡的 Greengrass 设备下，选择相应的部署链接。
6. 在 “操作” 下选择 “修订”。阅读警告，然后选择 “修订部署”。

> **Important**
> 创建修改后的会话超时配置会替换设备的当前配置。

7. 在步骤 1 “指定目标” 中，提供可选名称以标识修改后的部署，然后选择 “下一步”。
8. 在步骤 2，选择组件（可选）中，您可以将所有当前选择保持不变，然后选择 “下一步”。
9. 在步骤 3 “配置组件-可选” 中，选择 aws.iot。 SiteWiseEdgeProcessor，然后选择配置组件。
10. 在 “配置更新” 部分的 “要合并的配置” 下，输入以下 JSON：

```
{
    "AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES": "240"
}
```

11. 设置的值（以分钟`AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES`为单位）。会话超时值可以从 1 分钟到 10080 分钟（7 天）不等。默认值为 240 分钟（4 小时）。
12. 选择确认。
13. 选择 “下一步” 继续完成剩余步骤，直到进入步骤 5 “查看”。
14. 查看您的配置更改，然后选择 Deploy 将更改应用到您的 SiteWise Edge 网关。

> **Note**
>
> 或者，您可以通过在 Edge 网关上将全局环境变量 AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES 设置为所需的值（以分钟为单位）来配置会话超时。 SiteWise

部署完成后，新的会话超时配置将应用于您的 SiteWise Edge API。

教程：在 Amazon IoT SiteWise Edge 网关上列出资产模型

您可以使用可用的子集 Amazon IoT SiteWise APIs 以及特定于边缘的资源 APIs 来与资产模型及其边缘资产进行交互。本教程将引导您获取 Amazon IoT SiteWise Edge 网关的临时凭证以及获取 SiteWise Edge 网关上的资产模型列表。

先决条件

在本教程的步骤中，您可以使用各种工具。要使用这些工具，请确保您安装了相应的必备工具。

要完成本教程，您需要：

- 已部署且正在运行 Amazon IoT SiteWise Edge 自托管网关要求
- 通过 SiteWise 端口 443 访问同一网络中的 Edge 网关。
- 安装 OpenSSL
- (f Amazon OpsHub o Amazon IoT SiteWise r) f Amazon OpsHub or Amazon IoT SiteWise 应用程序
- (curl) curl 已安装
- (Python) urllib3 已安装
- (Python) Python3 已安装
- (Python) Boto3 已安装
- (Python) BotoCore已安装

步骤 1：获取 SiteWise Edge 网关服务签名证书

要与 SiteWise Edge 网关上的 APIs 可用设备建立 TLS 连接，您需要一个可信证书。您可以使用 OpenSSL 或 Amazon OpsHub 为生成此证书。 Amazon IoT SiteWise

OpenSSL

> **Note**
> 您需要安装 OpenSSL 才能运行此命令。

打开终端并运行以下命令从 SiteWise Edge 网关获取签名证书。<sitewise_gateway_ip>替换为 SiteWise 边缘网关的 IP。

```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | openssl
 x509 -outform PEM > GatewayCert.pem
```

Amazon OpsHub for Amazon IoT SiteWise

你可以用 Amazon OpsHub 于 Amazon IoT SiteWise。有关更多信息，请参阅 管理 SiteWise Edge 网关。

本教程使用下载的 SiteWise Edge 网关证书的绝对路径。运行以下命令导出证书的完整路径，将 <absolute_path_to_certificate> 替换为证书的路径：

```
export PATH_TO_CERTIFICATE='<absolute_path_to_certificate>'
```

步骤 2：获取您的 SiteWise Edge 网关主机名

> **Note**
> 您需要安装 OpenSSL 才能运行此命令。

要完成本教程，您需要提供 SiteWise Edge 网关的主机名。要获取 SiteWise Edge 网关的主机名，请运行以下命令，<sitewise_gateway_ip>替换为 SiteWise Edge 网关的 IP：

```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | grep -Po
 'CN = \K.*'| head -1
```

运行以下命令导出主机名供以后使用，<your_edge_gateway_hostname>替换为 SiteWise Edge 网关的主机名：

```
export GATEWAY_HOSTNAME='<your_edge_gateway_hostname>'
```

步骤 3：获取 SiteWise Edge 网关的临时证书

现在，您已获得 SiteWise Edge 网关的签名证书和主机名，您需要获取临时证书，以便可以在网关 APIs 上运行。您可以使用 Edge 网关 Amazon IoT SiteWise 或直接从 SiteWise Edge 网关获取这些证书 APIs。 Amazon OpsHub

> **Important**
>
> 凭证每 4 小时过期一次，因此您应在使用 SiteWise Edge 网关之前获取证书。 APIs 请勿将凭证缓存超过 4 小时。

使用 fo Amazon OpsHub r 获取临时证书 Amazon IoT SiteWise

> **Note**
>
> 您需要安装 f Amazon OpsHub o Amazon IoT SiteWise r 应用程序。

要使用 Amazon IoT SiteWise 应用程序 Amazon OpsHub 获取您的临时证书，请执行以下操作：

1. 登录应用程序。
2. 选择设置。
3. 对于身份验证，选择复制凭证。
4. 展开适合您环境的选项，再选择复制。
5. 保存凭证以供稍后使用。

使用 SiteWise Edge 网关 API 获取临时证书

要使用 SiteWise Edge 网关 API 获取临时证书，可以使用 Python 脚本或 curl，首先需要有 SiteWise Edge 网关的用户名和密码。边 SiteWise 缘网关使用 Sigv4 身份验证和授权。有关添加用户的更多信息，请参阅 LDAP 或 Linux 用户群体。以下步骤将使用这些凭证来获取您的 SiteWise Edge 网关上使用所需的本地证书 Amazon IoT SiteWise APIs。

Python

> **Note**
>
> 您需要安装 urllib3 和 Python3。

获取 Python 使用凭证

1. 创建一个名为 get_credentials.py 的文件，再将以下代码复制到其中。

```
'''
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 Amazon 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. 使用您创建的凭证替换 <gateway_username> 和 <gateway_password> 并在终端运行 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**
>
> 您需要安装 curl。

获取 curl 使用凭证

1. 使用您创建的凭证替换 <gateway_username> 和 <gateway_password> 并在终端运行命令。

```
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"
}'
```

该响应应当与以下内容相似：

```
{
    "username": "sweuser",
    "accessKeyId": "<accessKeyId>",
    "secretAccessKey": "<secretAccessKey>",
```

```
    "sessionToken": "<sessionToken>",
    "sessionExpiryTime": "2022-11-17T04:51:40.927095Z",
    "authMechanism": "linux",
    "role": "edge-user"
}
```

2. 从您的终端运行以下命令。

```
export AWS_ACCESS_KEY_ID=<accessKeyId>
export AWS_SECRET_ACCESS_KEY=<secretAccessKey>
export AWS_SESSION_TOKEN=<sessionToken>
export AWS_REGION=edge
```

第 4 步：获取 SiteWise Edge 网关上的资产模型列表

现在，您已拥有签名证书、 SiteWise Edge 网关主机名和 SiteWise Edge 网关的临时证书，您可以使用 `ListAssetModels` API 获取 SiteWise Edge 网关上的资产模型列表。

Python

> **Note**
>
> 你需要安装 Python3、Boto 3。BotoCore

使用 Python 获取资产模型列表

1. 创建一个名为 list_asset_model.py 的文件，再将以下代码复制到其中。

```
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. 从终端运行 list_asset_model.py。

```
python3 list_asset_model.py
```

curl

> **Note**
>
> 您需要安装 curl。

使用 curl 获取资产模型列表

从终端运行以下命令。

```
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"
```

该响应应当与以下内容相似：

```
{
    "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
}
```

在上托管 SiteWise Edge 网关 Siemens Industrial Edge

使用 Edge 应用程序将您的网关托管在西门子工业 Amazon IoT SiteWise 边缘上。与之类似 Amazon IoT Greengrass V2，您可以使用 SiteWise Edge on 来优化制造流程或改善操作工作流程Siemens Industrial Edge。

通过在设备上运行 Edge 网关，您可以将来自西门子工业 SiteWise 边缘设备的数据提取到您的 Amazon 账户。为此，请向 Edge 支持团队申请 Amazon IoT SiteWise Ed SiteWise ge 应用程序的访问权限。然后，创建 SiteWise 边缘网关资源，部署目标为西门子工业边缘设备（全新）。接下来，下载配置文件，然后通过 Siemens Industrial Edge Management 门户将其上传到您的应用程序。有关在 Siemens Industrial Edge 上运行应用程序的更多信息，包括如何设置所需的 Siemens 资源，请参阅 Siemens 文档中的 What is Industrial Edge?。

> **Note**
>
> 西门子不是 SiteWise Edge 的供应商或供应商。Siemens Industrial Edge 商城是一个独立的市场。

主题

- 安全性
- Siemens Secure Storage和 Amazon IoT SiteWise Edge 应用程序
- Siemens Industrial Edge设备的目的地
- 从预览版应用程序迁移
- 故障排除

- Amazon IoT SiteWise Edge 应用程序更新日志
- Amazon IoT SiteWise Edge 应用程序的要求
- 为 Siemens Industrial Edge 创建网关
- 为应用程序创建 Siemens Databus user
- 访问 Amazon IoT SiteWise Edge 应用程序
- 将应用程序安装到Siemens设备上
- 更新 Amazon IoT SiteWise Edge 应用程序配置

安全性

作为我们的客户和合作伙伴之间Amazon责任共担模式的一部分，以下内容描述了谁对安全的不同方面负责：

客户责任

- 审查合作伙伴。
- 配置授予合作伙伴的网络访问权限。
- 对运行 SiteWise Edge 的设备进行物理保护。

Amazon 责任

- 将合作伙伴与客户的 Amazon 云资源隔离开来。

合作伙伴的责任

- 使用安全的默认值。
- 通过补丁和其他适当更新确保解决方案随着时间的推移而保持安全。
- 对客户数据保密。
- 审查合作伙伴市场上可用的其他应用程序。

Siemens Secure Storage和 Amazon IoT SiteWise Edge 应用程序

为了保护运行 Amazon IoT SiteWise Edge 应用程序所需的凭据和机密，Siemens Industrial Edge提供了在设备上安全存储凭据的机制。如果 Amazon IoT SiteWise Edge 应用程序不支持安全存储这些凭证，则该应用程序将无法在设备上运行。由于缺少安全存储支持而导致的运行失败会被记录在日志文件中。

安装和运行 E Amazon IoT SiteWise dge 应用程序需要以下最低操作系统版本。将您的设备升级到最新版本以安装该应用程序。

- 对于虚拟设备：IEVD 版本 1.19 或更高版本
- 对于物理设备：IED-OS 版本 2.2 或更高版本

在升级设备之前，开启的 Amazon IoT SiteWise Edge 应用程序Siemens Industrial Edge将无法运行。

Siemens Industrial Edge设备的目的地

在上使用 Amazon IoT SiteWise Edge 应用程序时Siemens Industrial Edge，目的地会帮助准备数据，然后再将其发送到以 Amazon IoT SiteWise 供进一步分析和分发。您可以使用 Amazon S3 或使用实时数据提取配置缓冲数据摄取的数据目标设置。两者都允许您使用Siemens Industrial Edge设备部署目标上的路径过滤器订阅 MQTT 主题。

网关上的Siemens Industrial Edge部署目标支持两种主要的数据处理方法：

- Amazon IoT SiteWise 实时设置-数据在收集 Amazon IoT SiteWise 时直接发送到
- Amazon IoT SiteWise 使用 Amazon S3 设置进行缓冲-数据在发送到 Amazon S3 之前会分批临时收集和存储

有关配置这些选项的更多信息，请参见使用 Amazon S3 添加 Amazon IoT SiteWise 缓冲目的地和添加 Amazon IoT SiteWise Edge 实时目的地。

路径过滤器的前缀

使用Siemens Industrial Edge部署目标的网关路径筛选器将主题和数据流名称结合起来，为您的数据创建唯一标识符。带有数据流名称的组合主题在Siemens Industrial Edge网关中被称为前缀。这与自托管网关不同，在自托管网关中，路径过滤器仅基于 MQTT 主题。

Example 西门子数据流的路径过滤器结构

西门子数据流的典型路径过滤器包括主题路径和数据流名称：

```
ie/d/device1/application1/datastream1
```

其中：

- `ie/d/`是西门子数据流的必需前缀
- `device1/application1`表示分层路径
- `datastream1`是特定的数据流名称

> **Note**
>
> 处理Siemens Industrial Edge数据流时，请确保在路径筛选器中同时包含元数据 (`ie/m/ie/d/`) 和数据 () 主题，以接收有关数据流的完整信息。

目的地和路径过滤器

查看以下主题，详细了解启用 MQTT 的网关中的目的地和路径过滤器：

- 了解 Amazon IoT SiteWise 边缘目的地
- 添加 Amazon IoT SiteWise Edge 实时目的地
- 使用 Amazon S3 添加 Amazon IoT SiteWise 缓冲目的地
- 了解 Amazon IoT SiteWise Edge 目标的路径过滤器
- 向 Amazon IoT SiteWise Edge 目标添加路径过滤器
- 管理 Amazon IoT SiteWise 边缘目的地

从预览版应用程序迁移

如果您Siemens Industrial Edge在预览阶段运行了 SiteWise Edge，则需要从预览版 1.0.1 升级到最新版本。按照以下步骤进行迁移：

1. 创建新的 SiteWise Edge 网关。有关更多信息，请参阅 为 Siemens Industrial Edge 创建网关。
2. 为每个新网关创建一个新 Siemens Databus user。有关更多信息，请参阅 为应用程序创建 Siemens Databus user。
3. 在 IED 上卸载 1.0.1 版 Amazon IoT SiteWise 边缘网关应用程序。

   > **Note**
   >
   > 在重新配置 E Amazon IoT SiteWise dge 应用程序预览版之前使用的 Amazon IoT SiteWise 资产时，为数据流中断做好准备。在保留数据历史记录的同时，重新安装新网关时可能会导致数据丢失。

4. 在中删除您在预览期间创建的 SiteWise Edge 网关Amazon IoT SiteWise 控制台。
5. 使用新的网关配置文件在 I Amazon IoT SiteWise ED 上安装 Edge 网关应用程序。有关更多信息，请参阅 将应用程序安装到Siemens设备上。

> **Important**
>
> 安装新网关会覆盖 SiteWise Edge 应用程序的预览版。安装 2.0.0 版后，将无法退回到 1.0.1 版。

配置新网关和 Siemens Databus user 后，您的数据将流向您的属性。

您也可以直接从版本升级 SiteWise Edge 应用程序。但是，仍然需要新的网关配置。

故障排除

要对Siemens Industrial Edge设备上的 SiteWise Edge 网关进行故障排除，请参阅对 Amazon IoT SiteWise Edge 应用程序进行故障排除 Siemens Industrial Edge。

您也可以访问 Amazon re:Post 来查找问题的答案。

Amazon IoT SiteWise Edge 应用程序更新日志

下表描述了每个版本的 Amazon IoT SiteWise Edge 应用程序中的更改。

版本	更改	

3.0.0	新特征
	
	• 在 Amazon IoT SiteWise 控制台中为安装在以下位置的网关添加其他配置选项Siemens Industrial Edge：
	  • 添加和配置 Amazon S3 目的地。
	  • 移除 Amazon IoT SiteWise 实时目的地。
	  • 为每个目的地添加过滤器以控制数据路由。
	• 添加全局前缀配置选项，用于在Siemens IE安装程序中发送来自物联网 SiteWise 发布者的所有数据。过滤后应用前缀。
	
	错误修复和改进
	
	• 添加了在边缘应用程序的初始尝试失败时重试连接西门子工业 Amazon IoT SiteWise 边缘数据总线的功能。	

版本	更改
2.0.1	错误修复和改进 • 修复了应用程序在启动时无法检索 Amazon 凭据时会进入失败状态并退出的问题。增加了对在成功检索凭据之前重试的支持。
2.0.0	• Amazon IoT SiteWise Edge 应用程序现已正式上市。 • 应用程序需要 Siemens IEVD 1.19 版或 Siemens IED-OS 2.2 版。 • 性能改进：降低了内存消耗和 CPU 使用率。 • 调试改进：您现在可以上传可选的配置文件来启用调试日志。 • 安全增强功能：该应用程序使用 SecureStorage API 来在设备上安全地存储凭证。 • Docker摘要值：`sha256:4a960f29234a190ebb5224c1fd0f3e99faafccc4cb3d93ca13fef247b6656d18`
1.0.1	初始版本

Amazon IoT SiteWise Edge 应用程序的要求

要在上面运行 Amazon IoT SiteWise EdgeSiemens Industrial Edge，你需要满足以下条件：

- 一个 Siemens Digital Exchange 平台账户。
- 一个 Siemens Industrial Edge Hub（iehub）账户。
- 一个 Siemens Industrial Edge Management 实例。
  - IE 应用程序配置服务。要了解更多信息，请参阅 Siemens Industrial Edge Management 文档中的Installing the IE App Configuration Service manually。
- 访问 Amazon IoT SiteWise Edge 应用程序的 2.0.1 或更高版本。有关更多信息，请参阅 访问 Amazon IoT SiteWise Edge 应用程序。
- Siemens Industrial Edge Device（IED）或 Siemens Industrial Edge Virtual Device（IEVD）。
  - 至少 15 GB 的磁盘空间，以满足硬件要求。
  - 1 GB 内存和额外 1 GB 的交换内存。

- 设备配置以支持端口 443 和 8883 的出站流量。
- x86-64 位处理器。
- Siemens Industrial Edge Management 版本 1.13.10 或更高版本。
- 设备符合 Siemens Secure Storage 要求。
  - 在虚拟设备上，使用 IEVD 版本 1.19 或更高版本。
  - 在物理设备上，使用 IED-OS 版本 2.2 或更高版本。
- 最新版本的 Docker Compose。
- Docker Engine 版本 18.091 或更高版本。

- 必需的域访问权限。有关更多信息，请参阅 Amazon IoT SiteWise 端点。

为 Siemens Industrial Edge 创建网关

拥有适当的西门子帐户和 IEM 实例后，您可以创建部署类型为西门子工业 SiteWise 边缘设备的边缘网关。

> **Note**
>
> 请确保您满足在 Siemens Industrial Edge Management 上运行设备的所有要求。有关更多信息，请参阅 Amazon IoT SiteWise Edge 应用程序的要求。

要创建配置文件，请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择创建网关。
4. 对于部署类型，请选择西门子工业边缘设备。
5. 输入 E SiteWise dge 网关的名称或使用生成的名称 Amazon IoT SiteWise。
6. （可选）在高级配置下，执行以下操作：
   - 输入 Amazon IoT Core 事物的名称或使用生成的名称 Amazon IoT SiteWise。
7. 选择创建网关。
8. 在生成 SiteWise Edge 网关配置文件对话框中，选择生成并下载。 Amazon IoT SiteWise 自动生成用于配置 Amazon IoT SiteWise Edge 应用程序的配置文件。

> **Important**
>
> 您可以使用网关配置文件来备份和恢复 Amazon IoT SiteWise Edge 应用程序。保存您的 SiteWise Edge 网关配置文件Amazon Secrets Manager以安全地存储和管理该文件。Secrets Manager 可以安全地存储、管理和检索敏感信息。

为应用程序创建 Siemens Databus user

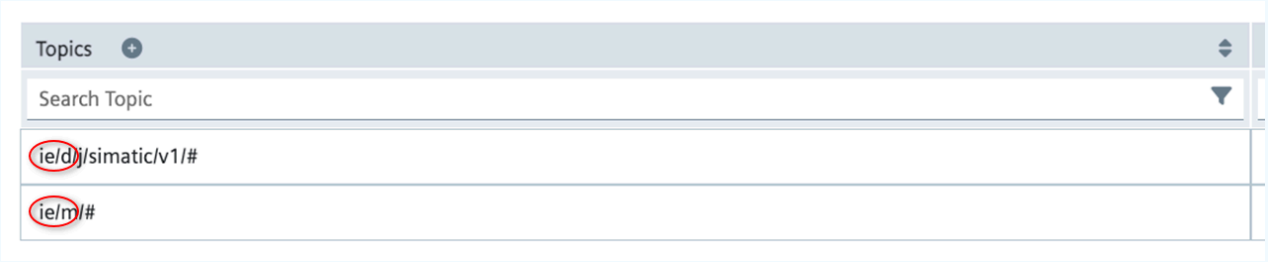
Amazon IoT SiteWise Ed Siemens Industrial Edge ge on 从Siemens Databus应用程序中提取数据。要将 SiteWise Edge 连接到Siemens Databus，您需要一个Siemens Databus user能够访问要安全传输到的数据的访问权限 Amazon IoT SiteWise。首先，创建一个，Siemens Databus user然后向 SiteWise Edge 应用程序提供证书。

要创建 Siemens Databus user，请执行以下步骤：

1. 在您的 Siemens Industrial Edge Management 实例中，在平台应用程序部分中选择边缘管理。
2. 选择数据连接图标。
3. 选择 Databus。此时，系统会显示已连接设备列表。
4. 选择要连接到 Amazon IoT SiteWise Edge 应用程序的设备。
5. 选择启动。此时，系统将显示适用于您所选设备的 Databus Configurator。
6. 在用户下为您的 Edge 设备创建用户。有关创建用户的更多信息，请参阅 Siemens 工业边缘管理文档中的用户。
7. 选择此 Siemens Databus 应有权访问的主题。这些主题限制了 Amazon IoT SiteWise Edge 可以访问的内容。

> **Important**
>
> Siemens Databus user有权访问的所有主题都已发布到 Amazon IoT SiteWise。

> **Note**
>
> Siemens Databus user需要访问数据和元数据主题。以 `ie/d` 开头的主题是数据主题。以 `ie/m` 开头的主题是元数据主题。成对共享主题，这样 SiteWise Edge 就可以访问每个相应主题的数据和元数据。



8. 为您的 Siemens Databus 配置设置适当的权限。

创建Siemens Databus配置后，您可以在上安装 Amazon IoT SiteWise Edge 应用程序Siemens Industrial Edge Management。有关更多信息，请参阅 将应用程序安装到Siemens设备上。

您也可以选择为Siemens Industrial Edge网关配置目的地和路径过滤器。有关更多信息，请参阅 目的地和路径过滤器。

访问 Amazon IoT SiteWise Edge 应用程序

要访问上的 Amazon IoT SiteWise Edge 应用程序Siemens Industrial Edge，请发送一封电子邮件请求访问 SiteWise Edge 支持团队。

在您的电子邮件中，请包含以下信息：

- 您的姓名和联系信息
- 公司名称
- Siemens Industrial Edge 租户 ID

将应用程序安装到Siemens设备上

通过向 Amazon IoT SiteWise Edge 支持团队发送电子邮件获得 E SiteWise dge 应用程序的访问权限后Siemens Industrial Edge，将该应用程序分配给的Siemens Industrial Edge Management实例。然后，您可以在设备上安装 Amazon IoT SiteWise Edge 应用程序。

安装 Ed Amazon IoT SiteWise ge 应用程序

1. 验证中提供的Docker摘要是否与中列出的最新版本Siemens Industrial Edge Management相匹配Amazon IoT SiteWise Edge 应用程序更新日志。

   有关查找 Siemens Docker 摘要值的更多信息，请参阅Siemens文档中的 “在Siemens Industrial Edge设备中管理应用程序”。

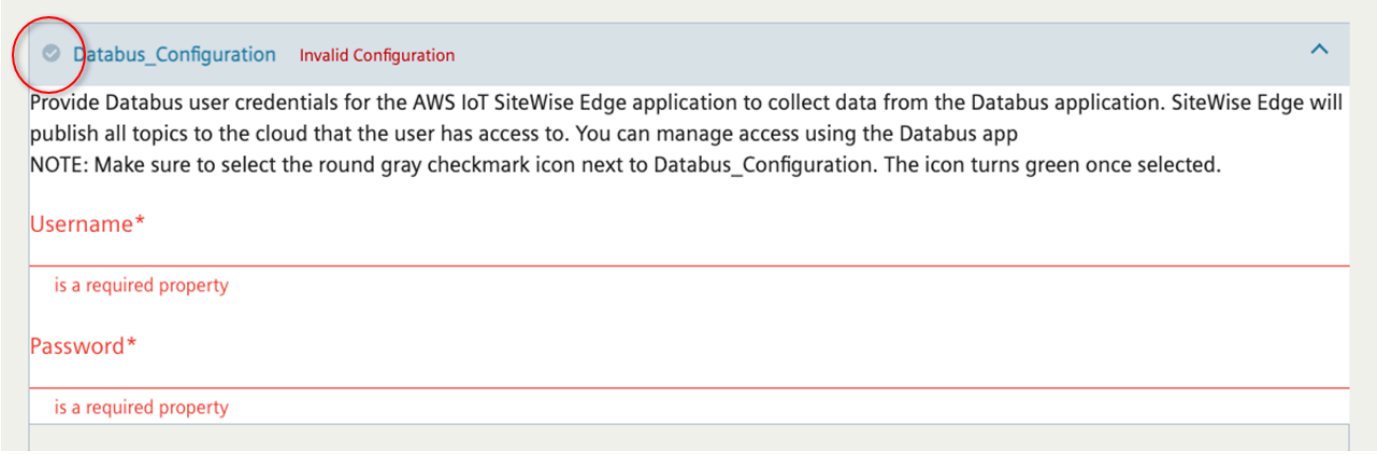
Siemens Industrial Edge Management一次支持一个版本的 Amazon IoT SiteWise Edge 应用程序。在Siemens Industrial Edge设备上安装 Amazon IoT SiteWise Edge 应用程序之前，请执行此步骤以确保您使用的是最新版本的应用程序。

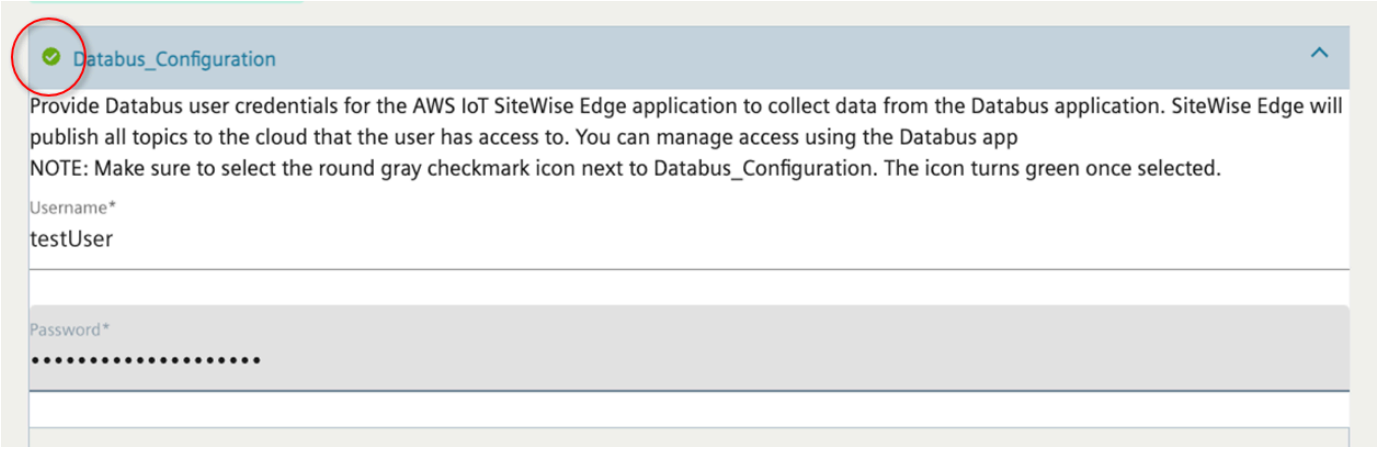
2. 将 Amazon IoT SiteWise Edge 应用程序分配给 Siemens Industrial Edge Management。有关更多信息，请参阅 Siemens 文档“Industrial Edge Management”部分的 Managing an app。
3. 在边缘管理中，浏览 Amazon IoT SiteWise Edge 的目录并选择它。
4. 选择安装。

> **Note**
>
> 如果显示 “联系我们” 按钮，请选择该按钮，然后按照步骤请求访问 Amazon IoT SiteWise Edge 应用程序Siemens Industrial Edge。有关更多信息，请参阅 访问 Amazon IoT SiteWise Edge 应用程序。

5. 在“架构配置”选项中选择 Databus_Configuration。
6. 为 Databus 配置输入用户名和密码。有关创建 Siemens Databus user 的更多信息，请参阅 为应用程序创建 Siemens Databus user。
7. 选择 Databus_Configuration 旁边的灰色圆形小对勾图标，将图标变为绿色。

> **Note**
>
> 仅当对勾图标从灰色变为绿色时，输入配置才适用。否则，输入配置将被忽略。





8. 选择下一步转到其他配置，您可以在其中上传网关配置文件。
9. 选择 SiteWise_Edge_Gateway_Config 作为上传网关配置文件的位置。

> **Note**
>
> 确保选择 SiteWise_Edge_Gateway_Config 而不是 _Edge_Support_Config_Config_Optional SiteWise

10. 选择要安装该应用程序的设备。
11. 选择立即安装。

您可以选择将发布者组件配置为将数据导出到 Amazon 云端。有关更多信息，请参阅 configure the Amazon IoT SiteWise publisher component。

要为Siemens Industrial Edge网关配置目的地，请参阅目的地和路径过滤器。

更新 Amazon IoT SiteWise Edge 应用程序配置

在上更新 Amazon IoT SiteWise Edge 应用程序配置时，需要考虑一些事项Siemens Industrial Edge。

> **Note**
>
> 对 Amazon IoT SiteWise Edge 应用程序配置的任何更改都需要重新启动应用程序。

重新启动 E Amazon IoT SiteWise dge 应用程序的原因

- Amazon IoT SiteWise Edge 应用程序的新增Siemens Databus user功能。

- 对网关配置文件（您的 SiteWise_Edge_Gateway_Config 文件）的更改。
- 代理配置更新（还需要完全重启 IEVD）
- 要为调试问题启用调试日志，请执行以下步骤：

重新启动应用程序

1. 在您的 Siemens Industrial Edge Management 实例中，在平台应用程序部分中选择边缘管理。
2. 选择我安装的应用程序。
3. 选择 Amazon IoT SiteWise Edge 应用程序。
4. 选择重新启动。

目的地和路径过滤器

Amazon IoT SiteWise Edge 中的目的地提供了一种灵活高效的方式来管理工业数据如何从边缘设备流向云端。本节介绍如何配置目的地、使用路径过滤器路由特定的数据流，以及如何为您的用例选择正确的目标类型。

您可以在支持 MQTT 的自托管、V3 网关和与托管的 Amazon IoT SiteWise Edge 应用程序一起使用的网关上使用目的地和路径过滤器。Siemens Industrial Edge目标和路径过滤器不适用于 Classic Streams、V2 网关。

主题

- 了解 Amazon IoT SiteWise 边缘目的地
- 了解 Amazon IoT SiteWise Edge 目标的路径过滤器
- 添加 Amazon IoT SiteWise Edge 实时目的地
- 使用 Amazon S3 添加 Amazon IoT SiteWise 缓冲目的地
- 向 Amazon IoT SiteWise Edge 目标添加路径过滤器
- 管理 Amazon IoT SiteWise 边缘目的地

了解 Amazon IoT SiteWise 边缘目的地

使用 Amazon IoT SiteWise Edge 目的地来确定将源数据发送到何处。您可以根据所需的特定特征来选择数据目的地，例如成本效益、低延迟或存储要求。集成我们的合作伙伴或自定义应用程序捕获

Amazon IoT SiteWise的设备数据，以便在边缘发布和订阅路径过滤器（主题）。然后，您可以在云端对设备数据进行建模、传输和存储。

> **Note**
>
> 要充分使用自托管网关上的所有目标功能，请升级到最新版本的 IoT SiteWise 发布者和 IoT SiteWise OPC UA 收集器。Classic Streams、V2 网关继续支持直播，以保持与现有设置的兼容性。有关更多信息，请参阅 经典直播，适用 Amazon IoT SiteWise 于 Edge 的 V2 网关。

主题

- SiteWise 边缘目的地如何增强数据管理
- 目标类型
- 比较网关版本之间的目标功能
- 目的地限制
- SiteWise 边缘目的地的用例

SiteWise 边缘目的地如何增强数据管理

使用 Amazon S3 将 Amazon IoT SiteWise 数据从边缘实时导出或批量导出。

目的地可增强您 Amazon IoT SiteWise 环境的灵活性和可扩展性。目的地实施集中式数据管理模型，即数据源将数据发布到中央系统。目的地使用路径筛选器确定您的数据发送到哪里。目的地可以订阅多个路径过滤器。

支持 MQTT 的网关，无论是自托管的还是运行在上面的网关Siemens Industrial Edge，都使用 MQTT 进行本地通信，并带有默认的实时目的地，过滤器设置为。#这意味着，默认情况下，所有主题的所有消息都将发布到 Amazon IoT SiteWise 实时目标。有关更多信息，请参阅 了解 Amazon IoT SiteWise Edge 目标的路径过滤器。您可以在每个网关中添加一个实时目的地。

目标类型

为网关配置目标时，您有两个主要选项：使用实时配置和使用 Amazon IoT SiteWise Amazon S3 进行缓冲配置。每种目的地类型都有自己的一组设置和注意事项。

Amazon IoT SiteWise 实时设置

选择此选项可将数据直接发送到 Amazon IoT SiteWise 热层存储，以便于实时摄取和监控数据。实时设置管理数据流，尤其是在网关遇到云连接问题时。在连接丢失期间，数据会临时存储在网关本地。重新建立连接后，存储的数据将自动发送到云端。

您可以调整数据发布过程的各个方面，例如要在本地存储的最大数据量、重新连接时向云端发送数据的速率以及在存储空间达到其容量后何时删除数据。

有关 Amazon IoT SiteWise 存储层的更多信息，请参阅管理中的数据存储 Amazon IoT SiteWise。

Amazon IoT SiteWise 使用 Amazon S3 设置进行缓冲

此目标类型允许您在网关上本地缓冲数据，并定期将其分批发送到 Amazon S3 存储桶。数据以高效的 Parquet 格式存储，该格式针对分析工作负载进行了优化。数据存入 Amazon S3 后，您可以将其导入 Amazon IoT SiteWise 以进行存储、处理和分析。

选择此选项可批量摄取数据，并以经济高效的方式存储历史数据。您可以配置您的首选 Amazon S3 存储桶位置，以及您希望将数据上传到 Amazon S3 的频率。您还可以选择在将数据摄取到 Amazon IoT SiteWise后如何处理数据。您可以选择同时在 Amazon S3 SiteWise 和 Amazon S3 中提供数据，也可以选择自动将其从 Amazon S3 中删除。

比较网关版本之间的目标功能

支持 MQTT 的网关中的目的地功能简化了数据流管理。目的地通过集中配置到各个端点的数据路由来简化数据管理。这种方法消除了对复杂的单个直播设置的需求，使整个系统更加灵活且更易于管理。

相比之下，Classic Streams、V2 网关、 SiteWise Edge 通过 Amazon IoT Greengrass 流将数据从数据源传输到发布者，为每个数据源单独配置数据目的地。

使用 Amazon IoT SiteWise 目标功能，可以整合发布者路由配置。目标配置允许您以集中方式管理目的地和路径过滤器。您可以根据需要轻松添加目的地、管理路径过滤器、删除不必要的过滤器或目的地。

此外，目标功能利用了 MQTT（消息队列遥测传输），这是一种广泛用于工业物联网应用的行业标准协议。采用 MQTT Amazon IoT SiteWise 有助于更轻松地与各种设备和系统集成。

目的地限制

目前 SiteWise Edge 网关上目的地的限制包括：

- 支持 MQTT 的网关不支持数据处理包。

- 数据类型支持仅限于 Amazon IoT SiteWise 数据类型。有关启用数据类型转换的信息，请参阅转换不支持的数据类型。

SiteWise 边缘目的地的用例

SiteWise 边缘目的地用于各种应用程序。以下是一些关键示例：

工业自动化, 实时监控和预测性维护

在工业环境中，工厂车间的传感器和设备可以将数据发布到 SiteWise Edge。可以将目的地配置为筛选和路由相关数据，从而实现对机器性能的实时监控和分析。您可以使用路径过滤器订阅相关的 MQTT 主题，处理数据，然后发布处理后的数据。通过这种方式，您可以有选择地将处理过的数据路由到 Amazon 云分析服务或本地系统。然后，制造商可以实施预测性维护策略，优化生产流程并减少停机时间。

智能建筑, 能源效率和占用率优化

楼宇自动化系统生成数据流，用于监视和控制建筑物的各个方面，例如暖通空调系统、照明和门禁控制。借 SiteWise 助 Edge，可以摄取、处理这些数据流并将其路由到不同的目的地。设施经理可以配置目的地以筛选和转发相关数据，从而实现能源效率措施和占用优化等高级功能，同时确保数据隐私和合规性。

这些用例演示了如何在各行各业中利用 SiteWise Edge 中的 “目的地” 功能来高效地摄取、处理和路由数据。这可以实现实时监控、预测性维护、能源效率和远程诊断等高级功能，同时确保数据隐私和合规性。

了解 Amazon IoT SiteWise Edge 目标的路径过滤器

主题

- 路径过滤器要求
- 路径过滤器的最佳实践
- OPC UA 服务器的路径过滤器
- 路径过滤器名称中的特殊字符

每个目标都配置为将数据路由到 Amazon IoT SiteWise 或 Amazon S3。路径过滤器允许您在接收目的地的 MQTT 消息时选择要筛选的特定数据。路径过滤器代表数据流的逻辑名称，充当对所需 MQTT 主题的订阅。

在 MQTT 中，数据被组织成主题，这些主题是由正斜杠 () / 分隔的分层字符串。例如，设备可能会向主题发布温度数据`home/livingroom/sensor1/temperature`。这里，`home/livingroom/sensor1`表示传感器的路径或逻辑名称，`temperature`也是要发布的数据类型。

您可以使用路径过滤器通过通配符（+和#）订阅特定主题或一系列主题。通+配符匹配主题层次结构中的单个级别。例如，`home/+/sensor1/temperature`将匹配`home/livingroom/sensor1/temperature`和`home/bedroom/sensor1/temperature`。当在过滤器的末尾使用#通配符时，可以匹配多个级别。

您还可以在路径过滤器名称中使用 MQTT 规范中通常不允许使用的各种字符。这些字符在名称中使用时不起通配符的作用。 Amazon IoT SiteWise 使用编码转换这些字符，以确保 MQTT 合规性，同时保留原始命名结构。此功能对于适应其他系统的现有命名约定特别有用。有关更多信息，请参阅 路径过滤器名称中的特殊字符。

通过仔细选择适当的路径过滤器，您可以控制将哪些数据发送到特定目的地。使用路径筛选器根据物联网系统的要求定制数据流。

路径过滤器要求

使用输入路径过滤器时 Amazon IoT SiteWise 控制台，请记住以下几点：

- 路径过滤器由新行分隔，每行代表一个单独的路径过滤器。
- 单个路径过滤器的字节可以介于 1 到 65,535 之间。
- 路径过滤器不能为空。
- 不允许使用空值 (U+0000)。
- 一次最多可以输入 100 个路径过滤器或 65,535 个字符，以先达到的限制为准。
- 网关上所有目的地的总路径过滤器限制为 20,000 个。
- 您可以在路径过滤器名称中使用%#+、、和$字符，但是它们 Amazon IoT SiteWise 会自动转换为 URI 编码。

路径过滤器的最佳实践

在为 Amazon IoT SiteWise 目的地创建路径筛选器时，请考虑以下策略来有效管理数据。

- 构造过滤器以反映您的设备层次结构。例如，在制造环境中`factory/+/machine/#`，捕获来自不同生产线的所有机器的数据。

- 对设备类型、位置或功能使用特定的级别。例如 `factory/assembly-line/robot/temperature`。或者，在智能农业中`farm/+/crop/+/moisture`，监测不同田地上各种作物的水分含量。
- 策略性地利用通配符：+用于单个关卡的变体#并捕获所有后续关卡。例如`building/+/+/energy-consumption`，跟踪建筑物中不同区域和楼层的能源使用情况。这假设第一个+捕获所有楼层，第二个+捕获所有区域。
- 通过创建足够具体的筛选器来平衡特异性和灵活性，这些筛选器既要足够具体，可以捕获相关数据，又要足够灵活，可以适应未来的变化。例如，`site/+/equipment-type/+/measurement`允许在不更改过滤器结构的情况下添加新的场地或设备类型。

彻底测试您的过滤器，确保它们捕获预期的数据并与您的物联网系统的架构和目标保持一致。

OPC UA 服务器的路径过滤器

对于 OPC UA 服务器，您的路径筛选器必须与 OPC UA 标签名称相对应。路径过滤器的最终级别必须与 OPC UA 标签名称完全匹配。例如，如果您的 OPC UA 标签为`Device1.Temperature`，则路径过滤器可能`factory/line1/Device1.Temperature`是。您可以在前面的关卡中使用通配符，例如`factory/+/Device1.Temperature`在多条生产线上捕获标签。如果您的路径筛选器名称中有特殊字符，路径过滤器名称中的特殊字符请参见以获取更多信息。

路径过滤器名称中的特殊字符

Amazon IoT SiteWise 可容纳 OPC UA 等工业协议中常用的字符，而标准 MQTT 主题名称中通常不允许使用这些字符。此功能有助于更顺畅地将工业系统与基于 MQTT 的架构集成。

> **Note**
>
> 虽然我们的特殊字符处理有助于集成和迁移，但建议尽可能与新实现的标准 MQTT 命名约定保持一致，以确保更广泛的兼容性。

从工业来源接收数据时，使用特殊字符的 URI 编码对主题名称进行 Amazon IoT SiteWise 标准化：

- %变成%25（首先编码为转义字符）
- # 变为 %23
- + 变为 %2B
- $变成%24（仅在主题开头时）

这种编码可确保包含这些特殊 MQTT 字符的源数据可以安全地用作 MQTT 主题名称，同时保留原始的工业命名惯例。

Example : 路径过滤器名称中的特殊字符

以下是工业主题名称在 Amazon IoT SiteWise 路径过滤器中可能如何显示的示例：

- `Factory1/Line#2/Sensor+3` 变为 `Factory1/Line%232/Sensor%2B3`
- `Plant%A/Unit$1/Temp` 变为 `Plant%25A/Unit%241/Temp`
- `Site1/#Section/+Node` 变为 `Site1/%23Section/%2BNode`

在中创建订阅或查看主题名称时 Amazon IoT SiteWise，您将看到未编码的原始版本。编码会自动处理，以确保 MQTT 合规性。

添加 Amazon IoT SiteWise Edge 实时目的地

实时目标类型使您能够将物联网数据直接从设备和网关实时流式传输到 Amazon IoT SiteWise 存储中。此选项非常适合需要在生成数据时立即摄取和处理数据，而无需进行批处理或缓冲的用例。您只能在每个网关中配置一个实时目的地，因为它会持续向其传输数据 Amazon IoT SiteWise。

> **Note**
>
> 重复 TQVs 可能会导致双重充电。

添加实时目的地

使用 Amazon IoT SiteWise 控制台或 Amazon CLI 向支持 SiteWise Edge MQTT 的网关添加实时目的地。

Console

1. 打开 Amazon IoT SiteWise 管理控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要向其添加目的地的网关。
4. 在目的地部分，选择添加目的地。
5. 在添加目的地页面上，输入目的地详细信息：
   a. 目的地名称字段中的目的地名称。

b. 为目标类型选择Amazon IoT SiteWise 实时。

6. 通过将发布顺序设置为 “先发布旧数据” 或 “先发布最新数据” 来配置网关发布顺序。默认情况下，网关会先发布最旧的数据。
7. 使用最大批处理等待时间来设置发布者在向其发送批量数据之前等待的最长时间 Amazon IoT SiteWise。此设置适用于每个别名。数据存储在本地，直到：

   - 设定的时间已过，或者
   - 已收到别名的 10 个 time-quality-value (TQV) 条目

   无论先满足哪个条件，都会触发将批次发送到云端。
8. 要压缩上传的数据，请选中 “上传数据时激活压缩” 复选框。让网关在将您的数据上传到云端之前对其进行压缩，可以减少带宽使用量。
9. 要筛选出过期的发布者数据，请选中 “排除过期数据” 复选框。此选择仅将活动和当前数据发送到 Amazon IoT SiteWise。
10. 在截止时间段字段中，输入数据集中应视为过期数据的频率。您可以确定数据是以分钟还是天为单位进行计数。最短截止时间为五分钟。最长截止期为七天。
11. （可选）配置本地存储设置：

    a. 设置保留期频率-网关在本地存储超过截止期限的数据的时间。最短保留期为一分钟。

       最长保留期为 30 天，且大于或等于轮换期。

    b. 设置轮换周期-保存超过单个文件截止时间段的数据时要指定的时间间隔。网关在每个轮换期结束时将一批数据传输到以下本地目录：`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`。

       保留时间必须大于一分钟且等于保留期。

    c. 提供存储容量 (GB) 值以设置本地存储的最大数据大小（以 GB 为单位）。如果数据超过确定的最大本地存储大小，网关将首先开始删除最旧的数据。网关会继续删除，直到本地存储的数据大小等于或小于配额。

       存储容量必须大于或等于 1 GB。
12. 为您的目的地添加路径过滤器。有关更多信息，请参阅向 Amazon IoT SiteWise Edge 目标添加路径过滤器。

有关更多信息，请参阅 目标类型。

Amazon CLI

Example：创建新的 Amazon IoT SiteWise 实时目的地

使用 UpdateGatewayCapabilityConfigurationAPI 配置发布者。

将 `capabilityNamespace` 参数设置为 `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": [
                            "#"
                        ]
                    }
                }
            ]
        }
    ]
}
```

要更新现有的 Amazon IoT SiteWise 实时目的地，请先使用 `DescribeGatewayCapabilityConfiguration` API 查找`destinationId`。

Example：更新 Amazon IoT SiteWise 实时目的地

使用 UpdateGatewayCapabilityConfigurationAPI 配置发布者。

将 capabilityNamespace 参数设置为 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": [
                            "#"
                        ]
                    }
                }
            ]
        }
    ]
}
```

以下配置选项特定于使用iotsitewise:publisher:3命名空间的网关。

`sources`

定义要将数据从工业设备传输到的数据源 Amazon IoT SiteWise。对于支持 MQTT 的网关，请使用。`MQTT`

类型： 对象数组

必需：是

`destinations`

定义将数据发送到何处。目的地要么是实时的，要么是使用 Amazon S3 进行缓冲的。至少需要一个目标对象，但你可以添加一个空数组。每个网关可以有一个实时目的地。有关更多信息，请参阅 了解 Amazon IoT SiteWise 边缘目的地。

类型： 对象数组

必需：是

`id`

目的地的唯一标识符。您可以提供现有的目的地 ID，也可以将其留空。如果您未指定 ID，则默认情况下会生成 UUID。

类型：字符串

必需：否

`type`

目标类型。选项包括：`SITEWISE_REALTIME`和`SITEWISE_BUFFERED`。

- `SITEWISE_REALTIME`— 实时将数据直接发送到 Amazon IoT SiteWise 存储器。
- `SITEWISE_BUFFERED`— 以 Parquet 格式将数据批量发送到 Amazon S3，然后导入 Amazon IoT SiteWise 存储。

类型：字符串

必需：是

`name`

目的地的唯一名称。

类型：字符串

必需：是

`config`

特定于目标类型的配置，采用 JSON 格式。配置因实时目的地和缓冲目的地而异。

类型：对象

必需：是

发布顺序

确定数据的发布顺序。数据根据其时间戳发布。选项包括`TIME_ORDER`和`RECENT_DATA`。

- `TIME_ORDER`（默认）-先发布较旧的数据。
- `RECENT_DATA`— 首先发布最新数据。

类型：字符串

必需：否

启用压缩

如果设置为`true`，则在发送到之前启用数据压缩 Amazon IoT SiteWise。让网关在将您的数据上传到云端之前对其进行压缩，可以减少带宽使用量。默认值为 `true`。

类型：布尔值

必需：否

丢弃政策

定义如何处理较旧的数据。

类型： 对象

必需：否

- `cutoffAge`

  要发布的数据的最大期限，以天、小时和分钟为单位指定。例如，`7d` 或 `1d7h16m`。不会将早于您指定时间的数据发送到 Amazon IoT SiteWise。

早于截止期的数据不会发布到云端。截止期必须介于 5 分钟到 7 天之间。

当您指定截止期时，可以使用 m、h 和 d。请注意，m 表示分钟，h 表示小时，d 表示天。

类型：字符串

必需：是

- `exportPolicy`

  定义如何处理超过截止时间的数据。

  类型：对象

  必需：否

  - `retentionPeriod`

    您的 SiteWise Edge 网关在存储指定保留期后，会从本地存储中删除边缘上所有早于截止期的数据。保留期必须介于 1 分钟到 30 天之间，且大于或等于轮换期。

    当您指定保留期时，可以使用 m、h 和 d。请注意，m 表示分钟，h 表示小时，d 表示天。

    类型：字符串

    必需：否

  - `rotationPeriod`

    将早于截止期的数据进行批处理和保存至单个文件的时间间隔。 SiteWise Edge 网关在每个轮换周期结束时将一批数据传输到以下本地目录:`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`. 轮换期必须多于 1 分钟，且等于或小于保留期。

    当您指定轮换期时，可以使用 m、h 和 d。请注意，m 表示分钟，h 表示小时，d 表示天。

    类型：字符串

    必需：否

  - `exportSizeLimitGB`

允许存储在本地的数据最大以 GB 为单位。如果超过此配额， SiteWise Edge 网关会开始删除最早的数据，直到本地存储的数据大小等于或小于配额。该参数值必须大于或等于 1。

类型：整数

必需：否

`maxBatchWaitTime`

设置发布者在向其发送批量数据之前等待的最长时间 Amazon IoT SiteWise。此设置适用于每个别名。数据存储在本地，直到：

- 设定的时间已过，或者
- 已收到别名的 10 个 time-quality-value (TQV) 条目

使用mh、和d来指定截止时间。请注意，m 表示分钟，h 表示小时，d 表示天。

类型：字符串

必需：否

`filters`

要应用于数据的筛选器。至少需要一个过滤器。

类型：字符串

必需：是

`type`

过滤器的类型。使用 PATH。

类型：字符串

必需：是

`config`

特定于过滤器类型的配置，采用 JSON 格式。至少需要一个对象，但数组可以为空。

类型：对象

必需：是

- `paths`

  路径过滤器数组。有关更多信息，请参阅 了解 Amazon IoT SiteWise Edge 目标的路径过滤器。默认路径为 #。

  类型：字符串数组

  必需：是

使用 Amazon S3 添加 Amazon IoT SiteWise 缓冲目的地

Amazon IoT SiteWise 如果您不需要实时数据，则缓冲目标类型允许您节省摄取成本。它允许您在将物联网数据导入 Amazon S3 存储桶之前将其临时存储在 Amazon S3 存储桶中 Amazon IoT SiteWise。或者，无论您是否计划将其导入，您都可以简单地将数据上传到 S3 进行存储 Amazon IoT SiteWise。这对于在将设备和网关的数据摄取到之前对其进行批处理和缓冲非常有用。 Amazon IoT SiteWise使用此选项，数据将以配置的频率以 Parquet 格式上传到指定的 S3 存储桶。然后，您可以将这些数据导入 Amazon IoT SiteWise 存储器以进行进一步的分析和处理。

添加使用 Amazon S3 缓冲的目标

使用 Amazon IoT SiteWise 控制台或 Amazon CLI 将使用 Amazon S3 缓冲数据的目标添加到您的 SiteWise Edge 网关。

Console

使用 Amazon Web Services Management Console 添加使用 Amazon S3 缓冲的 Amazon IoT SiteWise 目标。

1. 打开 Amazon IoT SiteWise 管理控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要向其添加目的地的网关。
4. 在目的地部分，选择添加目的地。
5. 在添加目的地页面上，输入目的地详细信息：
   a. 目的地名称字段中的目的地名称。
   b. 为目标类型选择使用 Amazon S3 进行Amazon IoT SiteWise 缓冲。 Amazon IoT SiteWise 使用 Amazon S3 缓冲以 Parquet 格式将数据分批发送到亚马逊简单存储服务，然后将数据导入 Amazon IoT SiteWise 存储。

6. 输入您要存储网关数据的位置的 Amazon S3 网址。您可以通过选择 “浏览 S3” 来浏览路径。添加存储桶后，您还可以通过选择 “查看” 来查看该存储桶。
7. 通过输入时间范围并选择数据上传频率的时间增量，指定网关应多久将数据上传到 Amazon S3 的频率。频率值应大于 0 且小于或等于 30 天。
8. 在数据存储设置中，确定将网关数据导入后如何处理网关数据 Amazon IoT SiteWise。关于数据存储，需要做出两个决定：
   - 如果要将导入的数据复制到 Amazon IoT SiteWise 存储中，请选中 “将数据复制到存储” 复选框。此选项会将导入的数据从您配置的 Amazon S3 存储桶复制到 Amazon IoT SiteWise 存储中。
   - 如果您选择将数据从 Amazon S3 存储桶导入 Amazon IoT SiteWise 存储，则还可以指定是否应在导入完成后删除导入的数据。选中 “从 Amazon S3 中删除数据” 复选框，在将导入的日期导入到存储空间后，将其从配置的 Amazon S3 Amazon IoT SiteWise 存储桶中删除。
9. 为您的目的地添加路径过滤器。有关更多信息，请参阅向 Amazon IoT SiteWise Edge 目标添加路径过滤器。

Amazon CLI

Example：创建一个使用 Amazon S3 进行缓冲的新 Amazon IoT SiteWise 目标

使用 UpdateGatewayCapabilityConfigurationAPI 配置发布者。

将 `capabilityNamespace` 参数设置为 `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：更新使用 Amazon S3 缓冲的 Amazon IoT SiteWise 目标

要更新现有的 Amazon IoT SiteWise 实时目的地，请先使用 `DescribeGatewayCapabilityConfiguration` API 查找`destinationId`。

发布者命名空间：`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": [
                "#"
              ]
            }
          }
        ]
      }
    ]
  }
```

以下配置选项特定于使用命名空间的启用 MQTT 的网关。`iotsitewise:publisher:3`

`sources`

定义要将数据从工业设备传输到的数据源 Amazon IoT SiteWise。对于支持 MQTT 的网关，请使用。`MQTT`

类型： 对象数组

必需：是

`destinations`

定义将数据发送到何处。目的地要么是实时的，要么是使用 Amazon S3 进行缓冲的。至少需要一个目标对象，但你可以添加一个空数组。每个网关可以有一个实时目的地。有关更多信息，请参阅 了解 Amazon IoT SiteWise 边缘目的地。

类型： 对象数组

必需：是

`id`

目的地的唯一标识符。您可以提供现有的目的地 ID，也可以将其留空，以便为目的地自动生成新 ID。

类型：字符串

必需：否

`type`

目标类型。选项包括：`SITEWISE_REALTIME`和`SITEWISE_BUFFERED`。选择`SITEWISE_BUFFERED`。

- `SITEWISE_REALTIME`（默认）-实时将数据直接发送到 Amazon IoT SiteWise 存储器。有关更多信息，请参阅 添加 Amazon IoT SiteWise Edge 实时目的地。
- `SITEWISE_BUFFERED`— 以 Parquet 格式将数据批量发送到 Amazon S3，然后导入 Amazon IoT SiteWise 存储。

类型：字符串

必需：是

`name`

目的地的唯一名称。

类型：字符串

必需：是

`config`

特定于目标类型的配置，采用 JSON 格式。配置因实时目的地和缓冲目的地而异。

类型：对象

必需：是

`targetBucketArn`

要发布到的 ARN 存储桶。 Amazon Web Services 区域 为两者 Amazon IoT SiteWise 和 Amazon S3 选择相同的选项。如果选择了前缀，则其长度必须介于 1-255 个字符之间。

> **Note**
>
> Amazon IoT SiteWise（包括网关）将有权访问整个指定的 S3 存储桶。我们建议使用专用的存储桶进行缓冲数据摄取。

类型：字符串

必需：是

`publishPolicy`

发布政策的详细信息。

类型：对象

必需：是

`publishFrequency`

SiteWise 边缘网关向 Amazon S3 存储桶发布内容的频率。向 Amazon S3 上传数据的频率必须大于 0 分钟且小于或等于 30 天。在指定发布频率期d限时，可以使用mh、和。请注意，m 表示分钟，h 表示小时，d 表示天。默认值为 15 分钟。

类型：字符串

必需：是

`localSizeLimitGB`

写入本地磁盘的文件的最大大小，以 GB 为单位。如果突破此阈值，则发布者会将所有缓冲的数据发布到其目的地。

类型：整数

必需：是

`siteWiseImportPolicy`

用于将数据导入的导入策略的详细信息 Amazon IoT SiteWise。

类型：对象

必需：是

`enableSiteWiseStorageImport`

将其设置为可`true`将数据从 Amazon S3 存储桶导入 Amazon IoT SiteWise 存储。它最初会复制中的数据 Amazon IoT SiteWise。然后，如果您设置`enableDeleteAfterImport`为 true，则 S3 中的数据将在复制到后删除 Amazon IoT SiteWise。定价含义适用。默认值为 `true`。

类型：布尔值

必需：是

`enableDeleteAfterImport`

将其设置为，`true`以便在将文件提取到存储空间后删除 Amazon S3 存储桶中的文件。 Amazon IoT SiteWise 默认值为 `true`。

类型：布尔值

必需：是

`bulkImportJobRoleArn`

Amazon IoT SiteWise 假设在数据摄取期间从 Amazon S3 读取缓冲数据的 IAM 角色的 ARN。当边缘设备调用 Amazon IoT SiteWise APIs 以启动批量导入过程时，将使用此角色。

> **Note**
>
> 如果设置`enableSiteWiseStorageImport`为`true`，则此参数为必填项。

类型：字符串

必需：否

为您的目的地添加路径过滤器。有关更多信息，请参阅 [向 Amazon IoT SiteWise Edge 目标添加路径过滤器](#)。

向 Amazon IoT SiteWise Edge 目标添加路径过滤器

向目标添加路径过滤器。路径过滤器使用 MQTT 主题语法，其中#是匹配任意数量级别的通配符，+是匹配单个级别的通配符。您可以向网关添加多个目的地，每个目的地都有自己的一组路径过滤器，这些过滤器已订阅到您的设备遥测中。

Siemens Industrial Edge网关使用前缀来实现兼容性。有关更多信息，请参阅 路径过滤器的前缀。

Console

添加路径过滤器

1. 打开 Amazon IoT SiteWise 管理控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要向其添加路径过滤器的网关。
4. 在 “添加目标” 下的 “路径过滤器” 部分中，选择 “添加路径过滤器”。
5. 输入您希望此目的地订阅的路径过滤器。您可以使用通配符（#和+）来订阅多个路径。
6. 选择 “添加路径过滤器”，将路径过滤器添加到列表中。
7. 如果需要，重复步骤以添加其他路径过滤器。
8. 添加完所有必需的路径过滤器后，选择 “创建”。

Amazon CLI for self-hosted gateways

Example：路径过滤器配置

```
{
  "destinations": [
    {
      ...
    }
  ],
  "filters": [
    {
      "type": "PATH",
      "config": {
        "paths": [
          "home/+/sensor1/temperature",
          "home/livingroom/sensor1/temperature",
          "home/livingroom/sensor1/temperature",
          "building/#"
        ]
      }
    }
  ]
```

```
}
```

Amazon CLI for 西门子 IEgateways

Example：路径过滤器的前缀配置

使用每个路径筛选器的数据 (`ie/d`) 和元数据 (`ie/m`) 前缀来捕获所有数据。

```
{
  "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**
>
> 通过下载路径过滤器列表在目的地之间复制路径过滤器。有关更多信息，请参阅 下载目标中的所有路径过滤器（控制台）。

批量上传路径过滤器

要批量上传路径过滤器，请使用 CSV 或文本文件。 Amazon IoT SiteWise 上传文件时会自动删除完全重复的内容。例如，`windfarm/site1/`和`windfarm/site1/`是精确的重复项 Amazon IoT SiteWise ，因为字符串完全相同。部分重复项不会被删除，因此会产生额外费用。例如，`windfarm/#`和`windfarm/site1`是重叠的主题`windfarm/site1`，因为已经包含在。`windfarm/#`

> **Note**
>
> 避免重复以免产生额外费用。上传的文件必须为.csv 或.txt 格式。它不能包含任何标题，并且应由一列组成。在该列中，列出您的路径过滤器，每个筛选器都位于单独的行中。文件中不应包含任何其他信息。

文件上传要求

这些是额外的路径过滤器要求。

- 您可以上传一个.csv 或.txt 文件。不支持其他文件格式。
- CSV (.csv) 文件不能有标题，只能包含一列。
- 每行可以有一个路径过滤器。
- 上传的文件不能为空。
- #用作通配符时，它必须是主题筛选器中的最后一个字符。例如，`topic/#`或作为特定主题级别的独立角色。但是，请注意，它#也可以用作主题级别名称中的常规字符，例如`factory/machine#1/topic`。有关更多信息，请参阅路径过滤器名称中的特殊字符
  - 您也可以使用该+字符。例如，使用`factory/+/temp`来获取工厂的所有温度，而不是`factory/machine3/temp`单独获取。`factory/machine2/temp`

管理 Amazon IoT SiteWise 边缘目的地

添加目的地后，您可以执行各种操作来管理它们，例如编辑目标配置、删除目标和管理路径过滤器。

编辑目的地（控制台）

选择表格中目的地旁边的单选按钮，然后选择 “编辑” 按钮来编辑目的地。

编辑目的地

1. 打开 Amazon IoT SiteWise 管理控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择相应的网关。
4. 在目的地部分，选择要编辑的目的地，然后选择编辑。
5. 修改目的地，然后选择 “保存”。

删除目的地（控制台）

如果您不再需要目的地，可以将其从 SiteWise Edge 网关中删除。

删除目的地

1. 打开 Amazon IoT SiteWise 管理控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择相应的网关。
4. 在目的地部分，选择要删除的目的地，然后选择删除。将出现确认屏幕。
5. 要确认您选择删除目的地，请在确认框中键入 “删除”。

下载目标中的所有路径过滤器（控制台）

在 Amazon IoT SiteWise 控制台中下载包含所有路径过滤器的 CSV 文件。您可以使用下载的路径过滤器列表在网关目标之间轻松共享路径过滤器列表。

下载包含所有路径过滤器的 CSV 文件

1. 打开 Amazon IoT SiteWise 管理控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择包含您的路径过滤器的网关。
4. 选择添加目的地或编辑目的地。
5. 导航到路径过滤器部分，然后选择下载 CSV。

> **Note**
>
> 无论您从路径筛选器列表中选择了哪个路径筛选器，CSV 文件都包含特定目标中的所有路径过滤器。

编辑路径过滤器（控制台）

使用 Amazon IoT SiteWise 控制台，可以在每个相应的文本框中编辑每个单独的路径过滤器。

编辑路径过滤器

1. 打开 Amazon IoT SiteWise 管理控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择包含您的路径过滤器的网关。
4. 选择相应的目的地。
5. 选择编辑。
6. 为包含要编辑的路径过滤器的行选择文本框。
7. 更新路径过滤器的文本，确保已编辑路径过滤器的复选框处于选中状态。
8. 选择保存。

删除路径过滤器（控制台）

您可以删除目标的路径筛选器，以控制其从 MQTT 源和数据处理管道接收的数据。

删除路径过滤器

1. 打开 Amazon IoT SiteWise 管理控制台。
2. 在左侧导航栏中，选择 Edge 部分的 Edge 网关。
3. 选择包含您的路径过滤器的网关。
4. 选择相应的目的地。
5. 选择编辑。
6. 在 “编辑目标” 屏幕的 “路径过滤器” 部分，选择一个或多个要删除的路径过滤器。
7. 选择删除。将显示一条删除确认消息。如果要继续删除路径过滤器，请在确认屏幕上选择删除。

管理 SiteWise Edge 网关

您可以使用 Amazon IoT SiteWise 控制台和 API 操作来管理 Amazon IoT SiteWise Edge 网关。您也可以使用 f Amazon OpsHub or Window Amazon IoT SiteWise s 应用程序从本地设备管理 SiteWise Edge 网关的某些方面。

我们强烈建议您使用 f Amazon OpsHub or Amazon IoT SiteWise 应用程序来监控本地设备上的磁盘使用情况。您还可以监控`Gateway.AvailableDiskSpace`和`Gateway.UsedPercentageDiskSpace` Amazon CloudWatch 指标，并创建警报，以便在磁盘空间不足时收到通知。有关 Amazon CloudWatch 警报的更多信息，请参阅基于静态阈值创建 CloudWatch 警报。

确保您的设备有足够的空间容纳后续数据。当本地设备的空间即将用完时，该服务会自动删除时间戳最早的少量数据，为后续数据腾出空间。

如要检查服务是否删除了您的数据，请执行以下操作：

1. 登录以 Amazon OpsHub 获取 Amazon IoT SiteWise 应用程序。
2. 选择设置。
3. 对于日志，指定时间范围，再选择下载。
4. 解压缩该日志文件。
5. 如果日志文件包含以下消息，则服务会删除您的数据：已删除*number*字节数据，以防止 SiteWise Edge 网关存储空间不足。

使用控制 Amazon IoT SiteWise 台管理您的 SiteWise Edge 网关

您可以使用 Amazon IoT SiteWise 控制台配置、更新和监控您 Amazon 账户中的所有 SiteWise Edge 网关。

您可以通过导航到Amazon IoT SiteWise 控制台中的 SiteWise Edge 网关页面来查看您的 Edge 网关。要访问特定网关的 Edge 网关详细信息页面，请选择 Edge 网关的名称。

在 Edge 网关详细信息页面的概述选项卡中，您可以执行以下操作：

- 在数据来源部分，更新数据来源配置并配置其他数据来源
- 选择 Open metrics 可在 CloudWatch 指标控制台中查看每个数据源摄取的数据点数 CloudWatch
- 在 Edge 功能部分，单击编辑将数据包添加到您的 SiteWise Edge 网关
- 在网关配置部分，查看 SiteWise Edge 网关的连接状态

- 在发布器配置部分，查看 SiteWise Edge 网关同步状态和 Amazon IoT SiteWise 发布者组件的配置

在 Edge 网关详细信息页面的更新选项卡中，您可以查看部署到 Edge 网关的当前组件和包版本。当新版本可用时，您也可以在此处部署新版本。

使用 Amazon OpsHub 管理 SiteWise 边缘网关 Amazon IoT SiteWise

您可以使用 f Amazon OpsHub or Amazon IoT SiteWise 应用程序来管理和监控您的自托管 SiteWise Edge 网关。此应用程序提供以下监控和管理选项：

- 在概述下，您可以执行以下操作：
  - 查看 SiteWise Edge 网关详细信息，帮助您深入了解 SiteWise Edge 网关设备数据、发现问题并提高 SiteWise Edge 网关的性能。
  - 查看 SiteWise 监控来自本地服务器和边缘设备的数据的监控门户。有关更多信息，请参阅 Amazon IoT SiteWise Monitor 应用程序指南 中的 什么是 Amazon IoT SiteWise Monitor。
- 在 He alth 下，有一个控制面板，用于显示来自 SiteWise 边缘网关的数据。领域专家（例如流程工程师）可以使用仪表板来查看 SiteWise 边缘网关行为的概述。
- 在资产下，查看部署到本地设备的资产以及上次为资产属性收集或计算的价值。
- 在 VPC 设置下，执行下列操作：
  - 如果安装了数据处理包，请查看 SiteWise Edge 网关配置信息并将资源与 Amazon 云同步。
  - 下载可用于使用其他工具访问 SiteWise Edge 网关的身份验证文件。
  - 下载可用于对 SiteWise Edge 网关进行故障排除的日志。
  - 查看部署到 SiteWise Edge 网关的 Amazon IoT SiteWise 组件。

> **Important**
>
> 以下内容需要用 Amazon OpsHub 于 Amazon IoT SiteWise：
>
> - 您的本地设备和 f Amazon OpsHub o Amazon IoT SiteWise r 应用程序必须连接到同一个网络。
> - 必须启用数据处理包。

使用管理 SiteWise Edge 网关 Amazon OpsHub

1. 下载并安装适用Amazon OpsHubAmazon IoT SiteWise 于 Windows 的应用程序。

2. 打开 应用程序。
3. 如果您没有为网关设置本地凭证，请按照 使用本地操作系统凭据访问您的 SiteWise Edge 网关 中的步骤进行设置。
4. 您可以使用 Linux 或轻型目录访问协议 (LDAP) 凭据登录 SiteWise Edge 网关。要登录您的 SiteWise Edge 网关，请执行以下任一操作：

   Linux

   1. 对于主机名或 IP 地址，输入本地设备的主机名或 IP 地址。
   2. 在身份验证中，请选择 Linux。
   3. 在用户名中，输入您的 Linux 操作系统的用户名。
   4. 在密码中，输入您的 Linux 操作系统的密码。
   5. 选择登录。

   LDAP

   1. 对于主机名或 IP 地址，输入本地设备的主机名或 IP 地址。
   2. 在身份验证中，请选择 LDAP。
   3. 对于用户名，请输入您的 LDAP 用户名。
   4. 对于 密码，请输入您的 LDAP 密码。
   5. 选择登录。

使用本地操作系统凭据访问您的 SiteWise Edge 网关

除了轻型目录访问协议 (LDAP) 之外，您还可以使用 Linux 或 Windows 凭据来访问您的自托管 SiteWise Edge 网关。

> **Important**
>
> 要使用 Linux 凭据访问您的 SiteWise Edge 网关，必须激活 SiteWise Edge 网关的数据处理包。

使用 Linux 操作系统凭据访问您的 SiteWise Edge 网关

下列步骤假定您使用的是装有 Ubuntu 的设备。如果您使用的是其他 Linux 发行版，请查阅您设备的相关文档。

创建 Linux 用户群体

1. 要创建管理员组，请运行以下命令。

```
sudo groupadd --system SWE_ADMIN_GROUP
```

   SWE_ADMIN_GROUP群组中的用户可以允许 SiteWise Edge 网关的管理员访问权限。

2. 要创建用户组，请运行以下命令。

```
sudo groupadd --system SWE_USER_GROUP
```

   SWE_USER_GROUP群组中的用户可以允许 SiteWise Edge 网关的只读访问权限。

3. 要向管理员组添加用户，请运行以下命令。*password*用要添加的用户名和密码替换*user-name*和。

```
sudo useradd -p $(openssl passwd -1 password) user-name
```

4. 要将用户添加到SWE_ADMIN_GROUP或中SWE_USER_GROUP，请*user-name*替换为您在上一步中添加的用户名。

```
sudo usermod -a -G SWE_ADMIN_GROUP user-name
```

现在，您可以使用用户名和密码登录 for Amazon IoT SiteWise 应用程序上的 SiteWise Edge 网关。Amazon OpsHub

使用 Windows 凭据访问您的 SiteWise Edge 网关

下列步骤假定您使用的是装有 Windows 的设备。

> **Important**
>
> 安全是双方共同承担 Amazon 的责任。创建包含至少 12 个字符以及大写、小写、数字和符号组合的强密码策略。此外，将 Windows 防火墙规则设置为允许端口 443 的传入流量，并阻止所有其他端口的传入流量。

创建 Windows Server 用户群体

1. 以管理员 PowerShell 身份运行。
   a. 在要安装 SiteWise 边缘网关的 Windows 服务器上，以管理员身份登录。
   b. 在 Windows 搜索栏PowerShell中输入。
   c. 在搜索结果中，右键单击 Windows PowerShell 应用程序。选择以管理员身份运行。
2. 要创建管理员组，请运行以下命令。

   ```
   net localgroup SWE_ADMIN_GROUP /add
   ```

   您必须是SWE_ADMIN_GROUP群组中的用户才能允许 SiteWise Edge 网关的管理员访问权限。
3. 要创建用户组，请运行以下命令。

   ```
   net localgroup SWE_USER_GROUP /add
   ```

   您必须是该SWE_USER_GROUP组中的用户才能允许 SiteWise Edge 网关的只读访问权限。
4. 要添加用户，请运行以下命令。*password*用要创建的用户名和密码替换*user-name*和。

   ```
   net user user-name password /add
   ```
5. 要向管理员组添加用户，请运行以下命令。*user-name*替换为要添加的用户名。

   ```
   net localgroup SWE_ADMIN_GROUP user-name /add
   ```

现在，您可以使用用户名和密码登录 for Amazon IoT SiteWise 应用程序上的 SiteWise Edge 网关。Amazon OpsHub

管理 SiteWise Edge 网关证书

您可以在边缘 SiteWise 网关设备上 SiteWise 使用监控器和第三方应用程序，例如 Grafana。这些应用程序需要与服务建立 TLS 连接。 SiteWise 边缘网关目前使用自签名证书。如果您使用浏览器打开应用程序（例如 Monito SiteWise r 门户），则可能会收到证书不可信的警告。

以下内容显示如何从 for Amazon IoT SiteWise 应用程序下载可信证书。 Amazon OpsHub

1. 登录到应用程序。
2. 选择设置。
3. 对于身份验证，请选择下载证书。

以下假设您使用的是谷歌浏览器或 FireFox。如果您使用其他浏览器，请查阅您的浏览器相关文档。要将上一步中下载的证书添加到浏览器，请执行下列操作之一：

- 如果您使用 Google Chrome，请按照 Google Chrome 企业帮助文章中的设置证书进行操作。
- 如果您使用 Firefox，请按照 Oracle 文档中的将证书加载至 Mozilla 或 Firefox 浏览器进行操作。

更改 SiteWise Edge 网关组件包的版本

您可以使用 Amazon IoT SiteWise 控制台更改 SiteWise Edge 网关上组件包的版本。

更改 SiteWise Edge 网关组件包的版本

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择 网关。
3. 选择要更改其包版本的 SiteWise Edge 网关。
4. 在网关配置下，选择查看软件版本。
5. 对于要更新版本的包，在编辑软件版本页面上，选择要部署的版本，然后选择部署。
6. 选择完成。

更新 Amazon IoT SiteWise 组件的版本

更新 Amazon IoT Greengrass 核心设备上的 Amazon IoT SiteWise 网关组件，确保您可以访问最新功能、性能改进和安全补丁。

要更新组 Amazon IoT SiteWise 件 Amazon IoT Greengrass

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择 Edge 网关。
3. 选择要编辑的网关，然后选择编辑。
4. 在 Edge 功能的软件版本下，选择可用的软件更新。将出现 “编辑软件版本” 页面。
5. 选择组件版本。

> **Note**
>
> 建议选择可用的最新版本。保留网关组件 up-to-date有助于保持工业数据收集和处理的最佳功能。

6. 选择部署。这将启动 Amazon IoT Greengrass V2 部署以更新网关上的 Amazon IoT SiteWise 组件。

删除 SiteWise Edge 网关

删除 E SiteWise dge 网关

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要删除的网关。
4. 选择删除。
5. 要确认要删除网关，请键入 “删除”，然后在出现的窗口中选择 “删除”。

备份和恢复 SiteWise Edge 网关

本主题介绍如何恢复 SiteWise Edge 网关和备份指标数据。如果您在同一台计算机上遇到 SiteWise Edge 网关损坏的问题，并且需要对问题进行故障排除，请阅读 Amazon IoT SiteWise 文档 SiteWise Edge 网关问题疑难解答。

> **Note**
>
> 本主题中介绍的指南适用于安装在 2.1.0 或更高 Amazon IoT Greengrass V2 版本上的 SiteWise Edge 网关。

指标数据的每日备份

如果您想在新机器上传输或恢复数据，则创建备份非常重要。备份数据可以大大降低在传输或恢复过程中丢失操作数据的风险。

本部分内容适用于使用数据处理包的网关。有关数据处理包的更多信息，请参阅在 E SiteWise dge 上配置用于数据处理的资产模型。

influxdb 文件夹路径如下所示：

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/influxdb
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeProcessor\influxdb
```

我们建议您备份整个文件夹及其项下的所有内容。

我们建议您定期将指标数据从 1.0 SiteWise Edge 备份到外部硬盘驱动器或 Amazon 云端。

恢复 SiteWise Edge 网关

在尝试恢复 SiteWise Edge 网关之前，请确保所有连接到该网关的 Edge 设备都已停止或断开连接。

使用以下步骤恢复 SiteWise Edge 网关：

1. 使用创建 SiteWise Edge 网关时下载的安装脚本在新计算机上恢复 SiteWise Edge 网关。阅读在本地设备上安装 SiteWise Edge 网关软件过程以设置 SiteWise Edge 网关。

   如果您丢失或找不到安装脚本，请联系 Amazon 客户支持。

2. 安装 E SiteWise dge 网关后，登录Amazon IoT Greengrass 控制台。

3. 要重新部署组件，请导航到管理，再在Amazon IoT Greengrass 设备下选择核心设备。
4. 在Amazon IoT Greengrass 核心设备表中，选择与您的 SiteWise Edge 网关对应的核心设备。
5. 进入设备页面后，打开部署选项卡并选择您的部署 ID，这将会使用您选择的 ID 打开部署页面。
6. 进入部署页面后，按右上角的操作按钮，再选择修订选项启动新的部署。配置部署。如果您想保持部署原样，请跳至查看并部署。
7. 等待部署状态变为 `Completed`。

> **Note**
>
> SiteWise Edge 上的所有组件也需要几分钟才能完全设置和运行。

恢复 Amazon IoT SiteWise 数据

使用以下过程在新计算机上恢复数据。

1. 将该 `influxdb` 文件夹复制到新机器上。
2. 通过在终端中运行以下命令来停止 SiteWise EdgeProcessor 组件：

   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
   ```

3. 找到您备份数据的路径，然后运行以下命令：

   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. 重启组 SiteWiseEdgeProcessor 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
   ```

验证备份和恢复是否成功

使用此过程验证您的备份数据和 E SiteWise dge 网关恢复。

> **Note**
>
> 此过程要求您已经安装 Amazon OpsHub 了 Amazon IoT SiteWise。有关更多信息，请参阅使用 Amazon OpsHub 用于管理 SiteWise Edge 网关 Amazon IoT SiteWise。

1. 开放 Amazon OpsHub 给 Amazon IoT SiteWise.
2. 在 SiteWise Edge 网关设置页面上，检查组件表中列出的每个组件的状态。确认状态颜色为绿色且读数显示为正在运行。
3. 在门户控制面板上验证您过去数据，检查过去数据和新数据是否都设置正确。过去数据和新数据之间会有停机时间。您应该可以看到一个未收集数据点的持续时间。

如果您在备份或恢复 SiteWise Edge 网关时遇到问题，请参阅以下疑难解答主题对 E Amazon IoT SiteWise dge 网关进行故障排除。

传统网关（Amazon IoT Greengrass Version 1）

> **Note**
>
> SiteWise 只有在 2021 年 7 月 29 日之前开始使用此功能时，在上 Amazon IoT Greengrass V1 运行的 Edge 网关才可用。有关使用运行 Amazon IoT SiteWise 网关的更多信息 Amazon IoT Greengrass V2，请参阅使用自托管 Amazon IoT SiteWise Edge 网关 Amazon IoT Greengrass V2。

SiteWise 边缘网关现在只能在上面运行 Amazon IoT Greengrass V2，为您的工业物联网应用提供增强的功能和更高的性能。这个最新版本 Amazon IoT Greengrass V2 代表了架构的演变，它建立在基于组件的现代框架之上，支持模块化软件部署。它通过统一的安装程序简化了安装，同时为开发人员提供了更大的灵活性来部署自定义组件和进行本地测试。基于组件的模型可以实现更有效的资源管理，并通过组件配方引入了简化的配置方法。这种设计有助于更好地处理组件之间的依赖关系，支持持续部署实践，并为本地开发提供增强的 CLI 功能。此外，通过 Amazon IoT Greengrass V2 集中配置管理 Amazon IoT Core 并提供改进的日志记录和监控功能，所有这些功能都受到更精细的安全权限模型的保护。

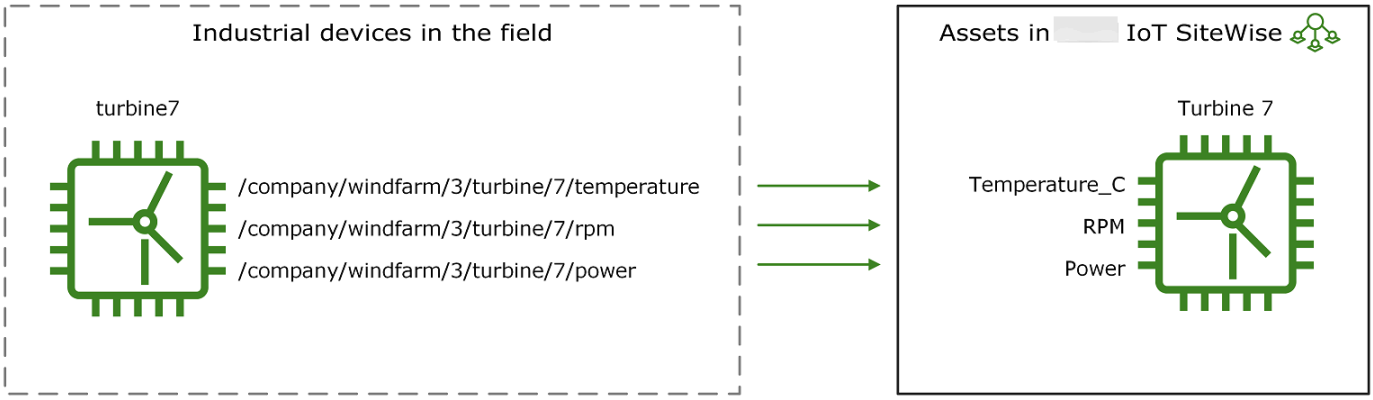
有关开始使用 SiteWise Edge 网关的更多信息 Amazon IoT Greengrass V2，请使用Amazon IoT SiteWise Edge 自托管网关要求。这些资源提供了 step-by-step有关设置网关、配置数据源和管理工业物联网基础设施的说明。

> **Note**
>
> 随着物联网服务的 Amazon 不断创新和改进，建议随时了解最新功能和增强功能。定期查看 Amazon IoT SiteWise 和 Amazon IoT Greengrass 文档，了解可以进一步优化您的工业物联网解决方案的新功能。

为工业资产建模

您可以使用 Amazon IoT SiteWise 资产创建工业运营的虚拟表现形式。资产代表将一个或多个数据流上传到 Amazon 云端的设备、设备或流程。例如，资产设备可以是风力涡轮机，它将空气温度、螺旋桨旋转速度和功率输出时间序列测量结果发送到 Amazon IoT SiteWise中的资产属性。

每个数据流对应一个唯一的属性别名。例如，别名 `/company/windfarm/3/turbine/7/temperature` 唯一标识来自风电场 #3 的涡轮机 #7 的温度数据流。您可以将 Amazon IoT SiteWise 资产配置为使用数学表达式转换传入的测量数据，例如将温度数据从摄氏度转换为华氏度。



资产还可以表示设备的逻辑分组，如整个风电场。您可以将资产与其他资产相关联，来创建表示复杂工业运营的资产层次结构。资产可以访问其关联的子资产中的数据。这样，您就可以使用 Amazon IoT SiteWise 表达式来计算汇总指标，例如风力发电场的净功率输出。

您必须从资产模型创建每个资产。资产模型是标准化资产格式的声明式结构。资产模型在同一类型的多个资产之间强制提供一致信息，以便您可以处理表示设备组的资产中的数据。在上图中，您对所有三个涡轮机使用相同的资产模型，因为所有涡轮机共享一组公有属性。

您也可以创建组件模型。组件模型是一种特殊类型的资产模型，您可以将其包含在资产模型或其他组件模型中。您可以使用组件模型来定义在多个资产模型之间共享的可重复使用的常见子装配件，例如传感器、电机等。

定义资产模型后，您可以创建工业资产。要创建资产，请选择 `ACTIVE` 资产模型以从该模型创建资产。然后，您可以填充特定于资产的信息，例如数据流别名和属性。在上图中，您可以从一个资产模型创建三个涡轮机资产，然后为每个涡轮机关联像 `/company/windfarm/3/turbine/7/temperature` 一样的数据流别名。

您还可以更新和删除现有资产、资产模型和组件模型。更新资产模型时，基于该资产模型的每个资产都会反映您对基础模型所做的任何更改。更新组件模型时，这适用于基于引用该组件模型的每个资产模型的每个资产。

您的资产模型可能非常复杂，例如在对具有许多子组件的复杂设备进行建模时。为了帮助保持此类资产模型的有序性和可维护性，您可以使用自定义复合模型对相关属性进行分组或重复使用共享组件。有关更多信息，请参阅 自定义复合模型（组件）。

主题

- 资产和模型状态
- 资产模型版本
- 自定义复合模型（组件）
- 设置 Amazon IoT SiteWise 对象 IDs
- 为创建资产和组件模型 Amazon IoT SiteWise
- 在中为资产模型创建资产 Amazon IoT SiteWise
- 在上搜索资产 Amazon IoT SiteWise 控制台
- 更新属性值
- 关联资产和取消关联资产
- 更新资产和模型
- 删除中的资源和模型 Amazon IoT SiteWise
- 使用资产和模型进行批量操作

资产和模型状态

创建、更新或删除资产、资产模型或组件模型时，更改需要一段时间才能传播。 Amazon IoT SiteWise 异步解析这些操作并更新每个资源的状态。每个资产、资产模型和组件模型都有一个状态字段，包含资源状态和任何错误消息（如适用）。状态可以是以下值之一：

- `ACTIVE` — 资源处于活动状态。这是唯一状态，您可以在其中查询资产、资产模型和组件模型并与之交互。
- `CREATING` — 正在创建资源。
- `UPDATING` — 正在更新资源。
- `DELETING` — 正在删除资源。

- PROPAGATING — （仅限资产模型和组件模型）更改将传播到所有依赖资源（从资产模型传播到资产，或从组件模型传播到资产模型）。
- FAILED — 资源在创建或更新操作期间无法验证，可能是由于表达式中的循环引用。您可以删除处于 FAILED 状态的资源。

一些创建、更新和删除操作ACTIVE会将资产、资产模型或组件模型 Amazon IoT SiteWise 置于操作解析之外的状态。如要在执行其中一项操作后查询资源或与之进行交互，必须等到状态更改为 ACTIVE。否则，您的请求将失败。

主题

- 检查资产的状态
- 检查资产或组件模型的状态

检查资产的状态

您可以使用 Amazon IoT SiteWise 控制台或 API 来检查资产的状态。

主题

- 检查资产的状态（控制台）
- 检查资产的状态（Amazon CLI）

检查资产的状态（控制台）

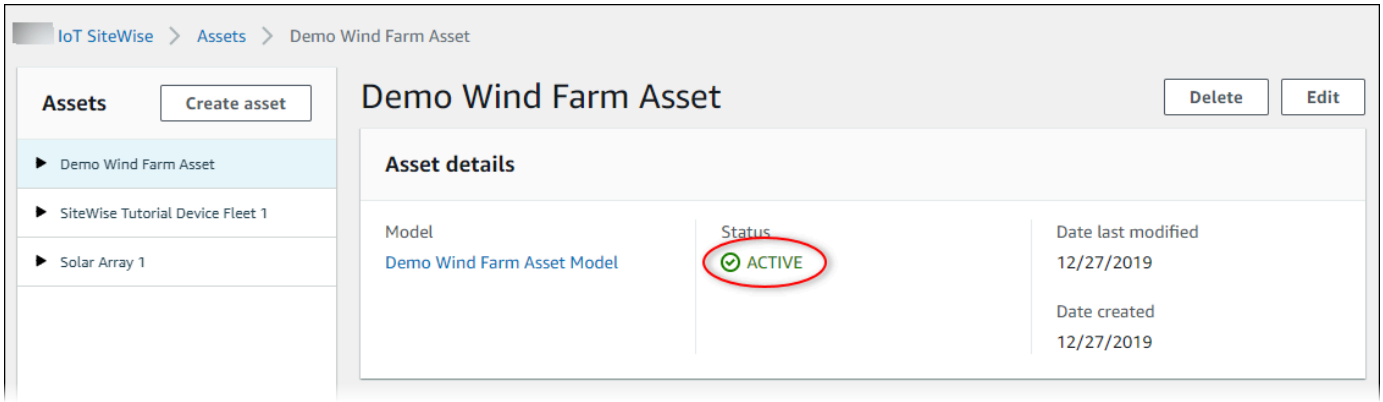
使用以下过程可以在 Amazon IoT SiteWise 控制台中检查资产的状态。

检查资产的状态（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要检查的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 在 资源详细信息面板中查找 状态。



检查资产的状态（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 来检查资产的状态。

要检查资产的状态，请使用带`assetId`参数的DescribeAsset操作。

要检查资产的状态（Amazon CLI），请执行以下步骤：

- 运行以下命令以描述资产。*`asset-id`*替换为资产的 ID 或外部 ID。外部 ID 是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

```
aws iotsitewise describe-asset --asset-id asset-id
```

此操作将返回包含资产详细信息的响应。响应中包含具有以下结构的 `assetStatus` 对象：

```
{
    ...
    "assetStatus": {
      "state": "String",
      "error": {
         "code": "String",
         "message": "String"
      }
    }
  }
```

资产的状态位于 JSON 对象的 `assetStatus.state` 中。

检查资产或组件模型的状态

您可以使用 Amazon IoT SiteWise 控制台或 API 来检查资产模型或组件模型的状态。

主题

- 检查资产模型或组件模型的状态（控制台）
- 检查资产模型或组件模型的状态（Amazon CLI）

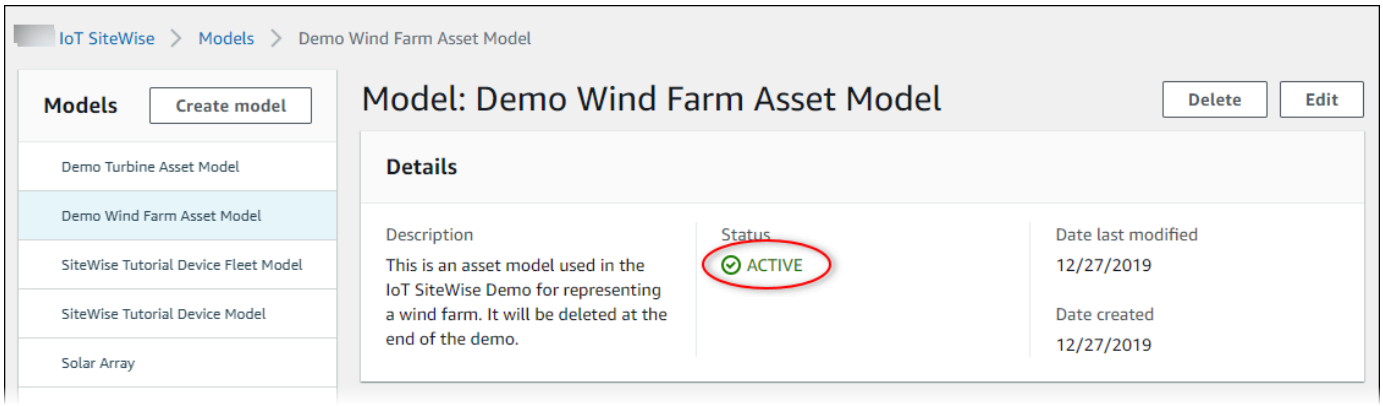
检查资产模型或组件模型的状态（控制台）

使用以下步骤在 Amazon IoT SiteWise 控制台中检查资产模型或组件模型的状态。

> **Tip**
> 资产模型和组件模型均列在导航窗格的模型下。所选资源模型或组件模型的详细信息面板会显示其类型。

要检查资产模型或组件模型的状态（控制台），请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择要检查的模型。
4. 在详细信息面板中查找状态。



检查资产模型或组件模型的状态（Amazon CLI）

您可以使用 Amazon CLI 来检查资产模型或组件模型的状态。

要检查资产模型或组件模型的状态，请使用带`assetModelId`参数的DescribeAssetModel操作。

> **Tip**
>
> Amazon CLI 将组件模型定义为一种资产模型。因此，您可以对两种类型的模型使用相同的DescribeAssetModel操作。响应中的 `assetModelType` 字段表示它是 `ASSET_MODEL` 还是 `COMPONENT_MODEL`。

要检查资产模型或组件模型的状态（Amazon CLI），请执行以下步骤：

- 运行以下命令以描述该模型。*`asset-model-id`*替换为资产模型或组件模型的 ID 或外部 ID。外部 ID 是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

  ```
  aws iotsitewise describe-asset-model --asset-model-id asset-model-id
  ```

  此操作将返回包含模型详细信息的响应。响应中包含具有以下结构的 `assetModelStatus` 对象。

  ```
  {
      ...
      "assetModelStatus": {
        "state": "String",
        "error": {
           "code": "String",
           "message": "String"
        }
      }
    }
  ```

  模型的状态位于 JSON 对象的 `assetModelStatus.state` 中。

资产模型版本

Amazon IoT SiteWise 支持对资产模型和组件模型的创建和更新操作进行异步处理。它还会更新模型的状态。

Amazon IoT SiteWise 将有效模型在创建和更新请求中的更改传播到其依赖资源（从资产模型传播到资产，或者从组件模型传播到资产模型）。然后，它会将该模型置于 ACTIVE 状态。

如果提供的模型定义无效，则将模型 Amazon IoT SiteWise 置于FAILED状态。更改不会传播到依赖资源。依赖资源是指模型处于 ACTIVE 状态时传播的最后一个模型定义。

根据以上信息，模型定义有两种类型的模型版本：

1. 最新版本 - 作为创建或更新请求的一部分接受的最新定义。
2. 活动版本 - 已成功处理最新定义，模型状态为 ACTIVE。

默认情况下，在资产模型或组件模型上调用 descri APIs be 时，会返回模型最新版本的详细信息。在某些情况下，需要资产模型或组件模型的活动版本。参见下面的示例场景：

- 一个定义无效的更新操作使您的资产模型处于 FAILED 状态。您必须通过检索资产模型的活动版本并创建另一个引用此有效定义的更新请求来恢复更改。
- 上 Amazon IoT SiteWise 有一个应用程序，客户可以在其中查看资产及其相应的资产模型。当用户引用与特定资产相对应的资产模型定义，并且该资产模型暂时处于 UPDATING、PROPAGATING 或 FAILED 状态时，最新版本返回尚未传播到其资产的资产模型定义。在这种情况下，您必须为客户检索资产模型的活动版本。

主题

- 检索资产模型或组件模型的活动版本（控制台）
- 检索资产模型或组件模型的活动版本（Amazon CLI）

检索资产模型或组件模型的活动版本（控制台）

按照以下步骤在 Amazon IoT SiteWise 控制台中检索资产模型或组件模型的活动版本。

> **Tip**
>
> 资产模型和组件模型均列在导航窗格的模型下。所选资源模型或组件模型的详细信息面板会显示其类型。

要检索资产模型或组件模型的活动版本（控制台），请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择要检索其活动版本的模型。
   a. 如果模型处于 ACTIVE 状态，则表示您正在查看其活动版本。
   b. 如果模型暂时处于 UPDATING、PROPAGATING 或 FAILED 状态，请在详细信息面板的状态下查找查看活动版本。

检索资产模型或组件模型的活动版本（Amazon CLI）

Amazon CLI 使用检索资产模型或组件模型的活动版本。

要检索资产模型或组件模型的活动版本，请使用带assetModelVersion参数的DescribeAssetModel操作。

> **Tip**
> Amazon CLI 将组件模型定义为一种资产模型。因此，您可以对两种类型的模型使用相同的DescribeAssetModel操作。响应中的 assetModelType 字段表示它是 ASSET_MODEL 还是 COMPONENT_MODEL。

要检索资产模型或组件模型的活动版本（Amazon CLI），请执行以下步骤：

- 运行以下命令以描述该模型。*asset-model-id*替换为资产模型或组件模型的 ID 或外部 ID。外部 ID 是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

  ```
  aws iotsitewise describe-asset-model --asset-model-id asset-model-id --asset-model-
  version ACTIVE
  ```

  此操作将返回包含模型详细信息的响应。响应中包含具有以下结构的 assetModelStatus 对象。

  ```
  {
      ...
  ```

```
    "assetModelName": "string",
    "assetModelProperties": [ ... ],
    ...,
    "assetModelVersion": "string"
}
```

自定义复合模型（组件）

当您对特别复杂的工业资产（例如包含许多零件的复杂机器）进行建模时，保持资产模型的有序性和可维护性可能会成为一项挑战。

在这种情况下，您可以将自定义复合模型或组件（如果您使用的是控制台）添加到现有资产模型和组件模型中。它们会对相关属性进行分组，并会重复使用子组件定义，以此帮助您保持井然有序。

自定义复合模型分为两类：

- 内联自定义复合模型定义了一组已分组属性，这些属性适用于自定义复合模型所属的资产模型或组件模型。您可以使用它们对相关属性进行分组。它们由名称、描述和一组资产模型属性组成，不可重复使用。
- C omponent-model-based 自定义复合模型引用要包含在资产模型或组件模型中的组件模型。您可以使用它们在模型中包含标准子装配件。它们由名称、描述及其引用的组件模型的 ID 组成，没有自己的属性；引用的组件模型会将其关联属性提供给任何创建的资产。

以下各节介绍了如何在您的设计中使用自定义复合模型。

主题

- 内联自定义复合模型
- Component-model-based 自定义复合模型
- 使用路径来引用自定义复合模型属性

内联自定义复合模型

内联自定义复合模型提供了一种通过对相关属性进行分组来组织资产模型的方法。

例如，假设您需要对机器人资产进行建模。该机器人包括伺服马达、电源和电池。这些组成部分都有其自己的属性，需要包含在模型中。您可以定义一个名为 `robot_model` 的资产模型，其属性如下所示。

- robot_model
  - servo_status （整数）
  - servo_position （双精度）
  - powersupply_status （整数）
  - powersupply_temperature （双精度）
  - battery_status （整数）
  - battery_charge （双精度）

但是，在某些情况下，可能有许多子装配件，或者子装配件本身可能具有许多属性。在这些情况下，可能有太多属性，以至于在模型根目录的单个平面列表中引用和维护它们变得很麻烦，就像前面的示例一样。

要处理此类情况，您可以使用内联自定义复合模型对属性进行分组。内联自定义复合模型是定义其自身属性的自定义复合模型。例如，您可以通过以下方式为您的机器人建模。

- robot_model
  - servo
    - status（整数）
    - position（双精度）
  - powersupply
    - status（整数）
    - temperature （双精度）
  - battery
    - status（整数）
    - charge（双精度）

在前面的示例中，servo、powersupply 和 battery 是在 robot_model 资产模型中定义的内联自定义复合模型的名称。然后，这些复合模型都定义了自己的属性。

> **Note**
>
> 在本例中，每个自定义复合模型都定义了自己的属性，因此，所有属性都是资产模型（在本例中为 `robot_model`）本身的一部分。这些属性不会与任何其他资产模型或组件模型共享。例如，如果您创建了其他资产模型，其中也有一个名为 `servo` 的内联自定义复合模型，那么，在 `robot_model` 中对 `servo` 进行更改不会影响其他资产模型的 `servo` 定义。
> 如果要实现这样的共享（例如，只有一个伺服系统的定义，所有资产模型都可以共享），则可以改为为其创建一个组件模型，然后创建引用它的component-model-based复合模型。有关详细信息，请参阅下一节。

有关创建内联自定义复合模型的信息，请参阅创建自定义复合模型（组件）。

Component-model-based 自定义复合模型

可以在中创建元件模型 Amazon IoT SiteWise 来定义标准的、可重复使用的子组件。创建组件模型后，您可以在其他资产模型和组件模型中添加对该组件模型的引用。为此，您可以将component-model-based 自定义复合模型添加到要引用该组件的任何模型中。您可以从多个模型添加对组件的引用，也可以在同一个模型中多次添加引用。

通过这种方式，您可以避免在模型之间重复相同的定义。它还可以简化模型的维护工作，因为您对组件模型所做的任何更改都将反映在使用该模型的所有资产模型中。

例如，假设您的工业设施中有许多类型的设备，这些设备都使用相同类型的伺服电机。其中某些设备在单台设备中就有多个伺服电机。您为每种设备类型创建了一个资产模型，但不想每次都重复使用 `servo` 的定义。您只想建模一次，然后将其用于各种资产模型。如果您后续更改 `servo` 的定义，则它将在所有模型和资源中同步更新。

要以这种方式对前一个示例中的机器人进行建模，可以将伺服电机、电源和电池定义为组件模型，如下所示。

- `servo_component_model`
  - `status`（整数）
  - `position`（双精度）

- `powersupply_component_model`

- status（整数）
- temperature （双精度）

- battery__component_model
  - status（整数）
  - charge（双精度）

然后，您可以定义引用这些组件的资产模型，例如 robot_model。多个资产模型可以引用同一个组件模型。您还可以在一个资产模型中多次引用同一个组件模型，例如，如果您的机器人中有多个伺服电机。

- robot_model
  - servo1（引用：servo_component_model ）
  - servo2（引用：servo_component_model ）
  - servo3（引用：servo_component_model ）
  - powersupply （引用：powersupply_component_model ）
  - battery（引用：battery_component_model ）

有关如何创建组件模型的信息，请参阅创建组件模型。

有关如何在其他模型中引用您的组件模型的信息，请参阅创建自定义复合模型（组件）。

使用路径来引用自定义复合模型属性

在资产模型、组件模型或自定义复合模型上创建属性时，您可以从使用其值的其他属性（例如转换和指标）中引用该属性。

Amazon IoT SiteWise 为您提供了不同的方式来引用您的房产。最简单的方法通常是使用其属性 ID。但是，如果您要引用的属性位于自定义复合模型上，则您可能会发现，按路径引用该属性会更有用。

路径是路径段的有序序列，指定了属性在资产模型和复合模型中的嵌套复合模型间的位置。

获取属性路径

你可以从属性的path字段中获取属性的路径AssetModelProperty。

例如，假设您的资产模型 robot_model 包含一个具有 position 属性的自定义复合模型 servo。如果您调用 DescribeAssetModelCompositeModelservo，则该position属性将列出一个如下所示的path字段：

```
"path": [
    {
       "id": "asset model ID",
       "name": "robot_model"
    },
    {
       "id": "composite model ID",
       "name": "servo"
    },
    {
       "id": "property ID",
       "name": "position"
    }
]
```

使用属性路径

在定义引用其他属性（例如转换或指标）的属性时，您可以使用属性路径。

一个属性使用一个变量来引用另一个属性。有关使用变量的更多信息，请参阅在公式表达式中使用变量。

在定义变量来引用属性时，您可以使用该属性的 ID 或路径。

要定义使用被引用属性路径的变量，请指定其值的 propertyPath 字段。

例如，要定义具有通过路径引用属性的指标的资产模型，可以将这样的负载传递给 CreateAssetModel：

```
{
    ...
    "assetModelProperties": [
        {
            ...
            "type": {
                "metric": {
                    ...
                    "variables": [
```

```
                        {
                            "name": "variable name",
                            "value": {
                                "propertyPath": [
                                    path segments
                                ]
                            }
                        }
                    ],
                    ...
                }
            },
            ...
        },
        ...
    ],
    ...
}
```

设置 Amazon IoT SiteWise 对象 IDs

Amazon IoT SiteWise 定义各种类型的永久对象，例如资产、资产模型、属性和层次结构。所有这些对象都有唯一的标识符，可用于对其进行检索、更新和删除。

Amazon IoT SiteWise 为客户提供了不同的身份创建选项。 Amazon IoT SiteWise 默认情况下，在创建对象时会为您生成一个。用户也可以为你的对象 IDs 提供自己的对象。

主题

- 处理对象 UUIDs
- 使用外部 IDs

处理对象 UUIDs

中的每个永久对象都 Amazon IoT SiteWise 有一个用于标识它的 UUID。例如，资产模型具有资产模型 ID，资产具有资产 ID，依此类推。此 ID 是在您创建对象时分配的，并且在该对象的生命周期内保持不变。

创建新对象时，默认情况下会为您 Amazon IoT SiteWise 生成一个唯一的 ID。您也可以在创建时以 UUID 格式提供您自己的 ID。

> **Note**
>
> UUIDs 对于相同的对象类型，在创建它的 Amazon 区域内必须是全局唯一的。 Amazon IoT SiteWise 自动为您生成 ID 时，它始终是唯一的。如果您自行选择 ID，请确保其唯一性。

例如，如果您通过调用创建新的资产模型 CreateAssetModel，则可以在请求的可选`assetModelId`字段中提供自己的 UUID。

相比之下，如果您在该请求中省略 `assetModelId`，则 Amazon IoT SiteWise 会为新的资产模型生成一个 UUID。

使用外部 IDs

要以 UUID 以外的某种格式定义您自己的 ID，您可以分配一个外部 ID。例如，如果您重复使用系统中未使用的 ID，或者为了更便于人阅读 Amazon，则可以这样做。外部 IDs 有更灵活的格式。您可以在 Amazon IoT SiteWise API 操作中使用它们来引用您的对象，而不是使用 UUID。

与一样 UUIDs，每个外部 ID 在其上下文中都必须是唯一的。例如，您无法拥有具有同一外部 ID 的两个资产模型。此外，与一样 UUIDs，一个对象在其生命周期中只能有一个外部 ID，这是无法更改的。

外部和外部 IDs 之间的区别 UUIDs

外部 IDs 区别 UUIDs 在以下方面：

- 每个对象都有一个 UUID，但外部 IDs 是可选的。
- Amazon IoT SiteWise 从不生成外部 IDs。这些都是您自己提供的。
- 如果对象还没有外部 ID，您可以随时为其分配一个外部 ID。

外部格式 IDs

有效的外部 ID 具有以下属性：

- 介于 2 到 128 个字符之间。
- 第一个和最后一个字符必须是字母数字（A-Z、a-z、0-9）。
- 除第一个和最后一个以外的字符必须是字母数字，或者是以下形式之一：`_-.:`

例如，外部 ID 必须符合以下正则表达式：

`[a-zA-Z0-9][a-zA-Z0-9_\-.:]*[a-zA-Z0-9]+`

使用外部引用对象 IDs

在很多可以使用 UUID 来引用对象的场景中，如果对象拥有外部 ID，则您可以改用其外部 ID。为此，请将外部 ID 附加到字符串 `externalId:` 中。

例如，假设您有一个资产模型，其 UUID（资产模型 ID）为 `a1b2c3d4-5678-90ab-cdef-11111EXAMPLE`，并且还具有外部 ID `myExternalId`。DescribeAssetModel致电了解详情。您可以使用以下任意一项作为 `assetModelId` 的值：

- 使用资产模型 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:` 前缀本身不是外部 ID 的一部分。只有在为接受外部 UUIDs 或外部的 API 操作提供外部 ID 时，才需要提供前缀 IDs。例如，在查询或更新现有对象时提供前缀。
> 在为对象定义外部 ID 时（例如创建资产模型时），请不要包含前缀。

IDs 在中的许多 API 操作 UUIDs 中，你可以使用外部代替这种方式 Amazon IoT SiteWise，但不是全部。例如 GetAssetPropertyValue，必须使用 UUIDs；它不支持使用外部 ID。

要确定特定的 API 操作是否支持这种用法，请查阅 API 参考。

为创建资产和组件模型 Amazon IoT SiteWise

Amazon IoT SiteWise 资产模型和组件模型可推动工业数据的标准化。资产和组件模型代表您的工业资产及其组成部分的结构和属性。资产模型定义了整体资产，例如风力涡轮机或生产线。组件模型表示构成资产的各个组件，例如叶片、发电机或传感器。通过创建这些模型，您可以以反映工业设备在现实世界中的关系和层次结构的方式组织和结构化您的资产数据，从而使其更易于监控、分析和维护。

资产模型或组件模型包含名称、描述、资产属性，以及（可选）将属性组合在一起或引用子装配件的组件模型的自定义复合模型。

在中 Amazon IoT SiteWise，您可以创建资产模型和组件模型来表示工业资产及其组件的结构和属性。

- 使用资产模型创建资产。除了上面列出的特征外，资产模型还可以包含定义资产之间关系的层次结构定义。
- 组件模型表示资产模型或其他组件模型中的子装配件。创建组件模型时，您可以在资产模型和其他组件模型中添加对该组件模型的引用。但是，您无法直接从组件模型创建资源。

创建资产模型或组件模型后，您可以为其创建自定义复合模型，以将属性组合在一起或引用现有的组件模型。

有关如何创建资产模型和组件模型的详细信息，请参阅以下部分。

主题

- 在中创建资产模型 Amazon IoT SiteWise
- 创建组件模型
- 定义数据属性
- 创建自定义复合模型（组件）

在中创建资产模型 Amazon IoT SiteWise

Amazon IoT SiteWise 资产模型推动工业数据的标准化。资产模型包含名称、描述、资产属性和资产层次结构定义。例如，您可以使用温度、每分钟旋转次数 (RPM) 和功率属性来定义风力涡轮机模型。然后，可以使用净功率输出属性和风力涡轮机层次结构定义来定义风电场模型。

> **Note**
>
> - 我们建议您从最低级别的节点开始对运营进行建模。例如，在创建风电场模型之前创建风力涡轮机模型。资产层次结构定义包含对现有资产模型的引用。通过这种方法，您可以在创建模型时定义资产层次结构。
> - 资产模型无法包含其他资产模型。如果必须定义一个可以在另一个模型中作为子装配件引用的模型，则应改为创建一个组件--> 模型。有关更多信息，请参阅 创建组件模型。

以下各节介绍如何使用 Amazon IoT SiteWise 控制台或 API 创建资产模型。以及可用于创建模型的不同类型的资产属性和资产层次结构。

主题

- 创建资产模型（控制台）
- 创建资产模型（Amazon CLI）
- 资产模型示例
- 定义资产模型层次结构

创建资产模型（控制台）

您可以使用 Amazon IoT SiteWise 控制台创建资产模型。 Amazon IoT SiteWise 控制台提供各种功能，例如公式 auto 补全，可以帮助您定义有效的资产模型。

创建资产模型（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择创建模型。
4. 在 创建模型 页面上，执行以下操作：
   a. 为资产模型输入 名称，例如 **Wind Turbine** 或 **Wind Turbine Model**。此名称对于您账户在此区域的所有模型都必须是唯一的。
   b. （可选）为模型添加外部 ID。这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。
   c. （可选）为模型添加 测量值定义。测量值代表来自您的设备的数据流。有关更多信息，请参阅 定义来自设备的数据流（测量值）。
   d. （可选）为模型添加 转换定义。转换是将数据从一种表单映射到另一种表单的公式。有关更多信息，请参阅 转换数据（转换）。
   e. （可选）为模型添加 指标定义。指标是按时间间隔汇总数据的公式。指标可以从关联资产的指标中输入数据，这样您可以计算统计信息，从而提供对整个操作或部分操作的见解。有关更多信息，请参阅 聚合来自属性和其他资产（指标）的数据。
   f. （可选）为模型添加 层次结构定义。层次结构是资产之间的关系。有关更多信息，请参阅 定义资产模型层次结构。
   g. （可选）为资产模型添加标签。有关更多信息，请参阅 标记您的 Amazon IoT SiteWise 资源。
   h. 选择创建模型。

创建资产模型时， Amazon IoT SiteWise 控制台会导航到新模型的页面。在此页面上，您可以查看模型的 状态，最初为 创建。此页面会自动更新，因此您可以等待模型的状态更新。

> **Note**
>
> 复杂模型的资产模型创建流程最多可能需要几分钟时间。在资产模型状态为活动后，您可以使用资产模型创建资产。有关更多信息，请参阅 资产和模型状态。

创建资产模型（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 来创建资产模型。

使用 CreateAssetModel 操作创建具有属性和层次结构的资产模型。此操作需要具有以下结构的负载。

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition
}
```

要创建资产模型（Amazon CLI），请执行以下操作：

1. 创建一个名为 `asset-model-payload.json` 的文件，然后将以下 JSON 对象复制到该文件中。

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ],
  "assetModelHierarchies": [

  ],
  "assetModelCompositeModels": [
```

```
  ]
}
```

2. 使用首选的 JSON 文本编辑器编辑 `asset-model-payload.json` 文件的以下内容：

   a. 输入资产模型的名称 (`assetModelName`)，如 **`Wind Turbine`** 或 **`Wind Turbine Model`**。此名称对于您账户在此 Amazon Web Services 区域的所有资产模型和组件模型都必须是唯一的。

   b. （可选）输入该资产模型的外部 ID（`assetModelExternalId`）。这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

   c. （可选）输入资产模型的说明（`assetModelDescription`），或删除 `assetModelDescription` 键/值对。

   d. （可选）定义模型的资产属性 (`assetModelProperties`)。有关更多信息，请参阅 定义数据属性。

   e. （可选）定义模型的资产层次结构 (`assetModelHierarchies`)。有关更多信息，请参阅 定义资产模型层次结构。

   f. （可选）为资产模型添加标签（`tags`）。有关更多信息，请参阅 标记您的 Amazon IoT SiteWise 资源。

3. 运行以下命令，根据 JSON 文件中的定义创建资产模型。

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-payload.json
```

   此操作将返回包含您在创建资产时引用的 `assetModelId` 的响应。响应中还包含模型的状态（`assetModelStatus.state`），最初为 `CREATING`。在传播更改之前，资产模型的状态为 `CREATING`。

   > **Note**
   >
   > 复杂模型的资产模型创建流程最多可能需要几分钟时间。要检查您的资产模型的当前状态，请通过指定来使用DescribeAssetModel操作`assetModelId`。在资产模型状态为 `ACTIVE` 后，您可以使用资产模型创建资产。有关更多信息，请参阅 资产和模型状态。

4. （可选）为您的资产模型创建自定义复合模型。借助自定义复合模型，您可以在模型内对属性进行分组，也可以通过引用组件模型来包含子装配件。有关更多信息，请参阅 创建自定义复合模型（组件）。

资产模型示例

本节包含资产模型定义示例，您可以使用这些示例使用 Amazon CLI 和创建资产模型 Amazon IoT SiteWise SDKs。这些资产模型代表风力涡轮机和风电场。风力涡轮机资产会摄取原始传感器数据并计算功率和平均风速等值。风电场资产计算诸如风电场中所有风力涡轮机的总功率之类的值。

主题

- 风力涡轮机资产模型
- 风电场资产模型

风力涡轮机资产模型

以下资产模型表示风电场中的涡轮机。风力涡轮机摄取传感器数据来计算功率和平均风速等值。

> **Note**
>
> 此示例模型类似于 Amazon IoT SiteWise 演示中的风力涡轮机模型。有关更多信息，请参阅 使用演 Amazon IoT SiteWise 示。

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

风电场资产模型

以下资产模型表示由多台风力涡轮机组成的风电场。该资产模型定义了风力涡轮机模型的层次结构。这使风电场能够根据风电场中所有风力涡轮机的数据计算值（例如平均功率）。

> **Note**
>
> 此示例模型类似于 Amazon IoT SiteWise 演示中的风力发电场模型。有关更多信息，请参阅 使用演 Amazon IoT SiteWise 示。

此资产模型依赖于风力涡轮机资产模型。将 `propertyId` 和 `childAssetModelId` 值替换为现有风力涡轮机资产模型中的对应值。

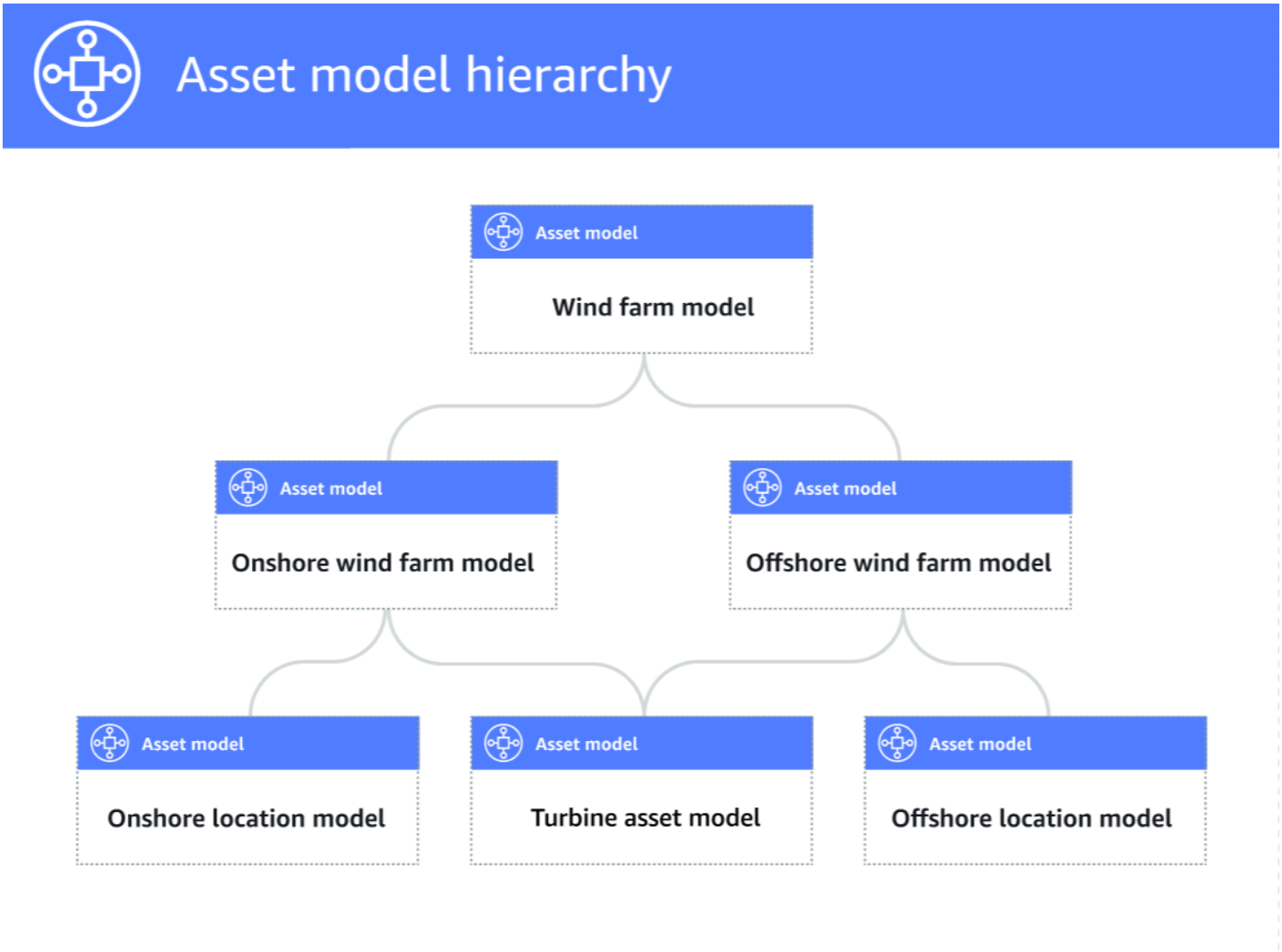
```
{
  "assetModelName": "Wind Farm Asset Model",
  "assetModelDescription": "Represents a wind farm.",
  "assetModelProperties": [
    {
      "name": "Code",
      "dataType": "INTEGER",
      "type": {
        "attribute": {
          "defaultValue": "300"
        }
      }
    },
    {
      "name": "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"
    }
  ]
}
```

定义资产模型层次结构

您可以定义资产模型层次结构，以便在工业运营中的资产模型之间创建逻辑关联。例如，您可以定义由陆上和海上风电场组成的风电场。陆上风电场包含涡轮机和陆上位置。海上风电场包含涡轮机和海上位置。



当您通过层次将子资产模型与父资产模型关联时，父资产模型的指标可以从子资产模式的指标输入数据。您可以使用资产模型层次结构和指标来计算统计信息，从而提供对整个操作或部分操作的见解。有关更多信息，请参阅 聚合来自属性和其他资产（指标）的数据。

每个层次结构定义父资产模型和子资产模型之间的关系。在父资产模型中，您可以为同一个子资产模型定义多个层次结构。例如，如果您的风电场中有两种不同类型的风力涡轮机，其中所有风力涡轮机都由相同的资产模型表示，则可以为每种类型定义一个层次结构。然后，您可以在风电场模型中定义指标，以计算每种风力涡轮机的独立和组合统计数据。

一个父资产模型可以与多个子资产模型相关联。例如，如果您的陆上风电场和海上风电场由两种不同的资产模型表示，则可以将这些资产模型与相同的父风电场资产模型关联起来。

子资产模型也可以与多个父资产模型相关联。例如，如果您有两种不同类型的风电场，其中所有风力涡轮机都由相同的资产模型表示，则可以将风力涡轮机资产模型与不同的风电场资产模型相关联。

> **Note**
>
> 定义资产模型层次结构时，子资产模型必须是 `ACTIVE` 或具有之前的 `ACTIVE` 版本。有关更多信息，请参阅 资产和模型状态。

定义分层资产模型并创建资产后，您可以关联资产以完成父子关系。有关更多信息，请参阅在中为资产模型创建资产 Amazon IoT SiteWise和关联资产和取消关联资产。

主题

- 定义资产模型层次结构（控制台）
- 定义资产层次结构（Amazon CLI）

定义资产模型层次结构（控制台）

在 Amazon IoT SiteWise 控制台中为资产模型定义层次结构时，需要指定以下参数：

- 层次结构名称 – 层次结构的名称，如 **`Wind Turbines`**。
- 层次模型 – 子资产模型。
- 层次结构外部 ID（可选）- 这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

有关更多信息，请参阅 创建资产模型（控制台）。

定义资产层次结构（Amazon CLI）

使用 Amazon IoT SiteWise API 为资产模型定义层次结构时，需要指定以下参数：

- `name` – 层次结构的名称，如 **`Wind Turbines`**。
- `childAssetModelId` – 层次结构的子资产模型的 ID 或外部 ID。您可以使用该ListAssetModels操作来查找现有资产模型的 ID。

Example 层次结构定义示例

以下示例演示一个表示风电场与风力涡轮机的关系的资产模型层次结构。此对象就是一个示例AssetModelHierarchy。有关更多信息，请参阅 创建资产模型（Amazon CLI）。

```
{
  ...
  "assetModelHierarchies": [
    {
      "name": "Wind Turbines",
      "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
    },
  ]
}
```

创建组件模型

使用 Amazon IoT SiteWise 组件模型定义子组件，您可以从资产模型或其他组件模型中引用这些子组件。这样，您就可以在多个其他模型中重复使用组件的定义，也可以在同一个模型中多次重复使用该组件的定义。

定义组件模型的过程与定义资产模型非常相似。与资产模型一样，组件模型也有名称、描述和资产属性。但是，组件模型不能包含资产层次结构定义，因为组件模型本身不能直接用于创建资产。组件模型也无法定义警报。

例如，您可以使用电机温度、编码器温度和绝缘电阻属性为伺服电机定义组件。然后，您可以为包含伺服电机的设备（例如数控机床）定义资产模型。

> **Note**
>
> - 我们建议您从最低级别的节点开始对运营进行建模。例如，在创建数控机床的资产模型之前，先创建伺服电机组件。资产模型包含对现有组件模型的引用。
> - 您无法直接从组件模型创建资源。要创建使用您的组件的资产，您必须为该资产创建资产模型。然后，为其创建引用您的组件的自定义复合模型。有关创建资产模型的更多信息，请参阅在中创建资产模型 Amazon IoT SiteWise。有关创建自定义复合模型的更多信息，请参阅创建自定义复合模型（组件）。

以下各节介绍如何使用 Amazon IoT SiteWise API 创建组件模型。

主题

- 创建组件模型（Amazon CLI）
- 示例组件模型

创建组件模型（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 来创建组件模型。

使用CreateAssetModel操作创建具有属性的组件模型。此操作需要具有以下结构的有效载荷：

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
}
```

要创建组件模型（Amazon CLI），请执行以下步骤：

1. 创建一个名为 `component-model-payload.json` 的文件，然后将以下 JSON 对象复制到该文件中：

   ```
   {
     "assetModelType": "COMPONENT_MODEL",
     "assetModelName": "",
     "assetModelDescription": "",
     "assetModelProperties": [

     ]
   }
   ```

2. 使用首选的 JSON 文本编辑器编辑 `component-model-payload.json` 文件的以下内容：

   a. 为组件模型输入一个名称（`assetModelName`），如 **`Servo Motor`** 或 **`Servo Motor Model`**。此名称对于您账户在此 Amazon Web Services 区域的所有资产模型和组件模型都必须是唯一的。

   b. （可选）为组件模型输入一个外部 ID（`assetModelExternalId`）。这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

   c. （可选）输入资产模型的说明（`assetModelDescription`），或删除 `assetModelDescription` 键/值对。

   d. （可选）定义组件模型的资产属性（`assetModelProperties`）。有关更多信息，请参阅定义数据属性。

   e. （可选）为资产模型添加标签（`tags`）。有关更多信息，请参阅 标记您的 Amazon IoT SiteWise 资源。

3. 运行以下命令，根据 JSON 文件中的定义创建组件模型。

```
aws iotsitewise create-asset-model --cli-input-json file://component-model-
payload.json
```

该操作会返回一个响应，其中包含您在资产模型或其他组件模型中添加对组件模型的引用时所引用的 `assetModelId`。响应中还包含模型的状态（`assetModelStatus.state`），最初为 `CREATING`。在传播更改之前，组件模型的状态为 `CREATING`。

> **Note**
>
> 复杂模型的组件模型创建流程最多可能需要几分钟时间。要检查组件模型的当前状态，请通过指定来使用DescribeAssetModel操作`assetModelId`。组件模型状态变为 `ACTIVE` 后，您可以在资产模型或其他组件模型中添加对该组件模型的引用。有关更多信息，请参阅 资产和模型状态。

4. （可选）为您的组件模型创建自定义复合模型。借助自定义复合模型，您可以在模型内对属性进行分组，也可以通过引用其他组件模型来包含子装配件。有关更多信息，请参阅 创建自定义复合模型（组件）。

示例组件模型

本节包含一个组件模型定义示例，您可以使用该示例创建带有 Amazon CLI 和的组件模型 Amazon IoT SiteWise SDKs。此组件模型表示可以在其他设备（例如数控机床）中使用的伺服电机。

主题

- 伺服电机组件模型

伺服电机组件模型

以下组件模型表示可在数控机床等设备中使用的伺服电机。伺服电机提供各种测量值，例如温度和电阻。这些测量值可用作从引用伺服电机组件模型的资产模型创建的资产的属性。

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

定义数据属性

资产属性是每个资产中包含资产数据的结构。资产属性可以是以下任一类型：

- 属性 – 通常为静态的资产特征，如设备制造商或地理区域。有关更多信息，请参阅 定义静态数据（属性）。
- 测量值 – 资产的原始设备的传感器数据流，如时间戳转速值或以摄氏度为单位的时间戳温度值。测量值由数据流别名定义。有关更多信息，请参阅 定义来自设备的数据流（测量值）。
- 转换 – 资源转换后的时间序列值，如以华氏度为单位的时间戳温度值。转换由表达式和随该表达式一起使用的变量定义。有关更多信息，请参阅 转换数据（转换）。
- 指标 – 在指定时间间隔内聚合的资产数据，如每小时平均温度。指示由时间间隔、表达式和随该表达式一起使用的变量定义。指标表达式可以输入关联资产的指标属性，这样您就可以计算出代表您的操作或操作子集的指标。有关更多信息，请参阅 聚合来自属性和其他资产（指标）的数据。

有关更多信息，请参阅 在中创建资产模型 Amazon IoT SiteWise。

有关如何使用测量值、转换和指标计算整体设备效率 (OEE) 的示例，请参阅计算 OEE Amazon IoT SiteWise。

主题

- 定义静态数据（属性）

- 定义来自设备的数据流（测量值）
- 转换数据（转换）
- 聚合来自属性和其他资产（指标）的数据
- 使用公式表达式

定义静态数据（属性）

资产属性表示通常为静态的信息，如设备制造商或地理位置。您从资产模型创建的每个资产都包含该模型的属性。

主题

- 定义属性（控制台）
- 定义属性（Amazon CLI）

定义属性（控制台）

在 Amazon IoT SiteWise 控制台中为资产模型定义属性时，需要指定以下参数：

- 名称 – 属性的名称。
- 默认值 –（可选）此属性的默认值。从模型创建的资产具有此属性值。有关如何覆盖从模型创建的资产中的默认值的更多信息，请参阅更新属性值。
- 数据类型 – 属性的数据类型，如以下类型之一：
  - 字符串 – 最多 1024 字节的字符串。
  - 整数 – 有符号的 32 位整数，范围是 [-2,147,483,648, 2,147,483,647]。
  - 双精度 – 具有 [-10^100, 10^100] 范围和 IEEE 754 双精度的浮点数。
  - 布尔值 – `true` 或 `false`。
- 外部 ID -（可选）这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

有关更多信息，请参阅 创建资产模型（控制台）。

定义属性（Amazon CLI）

使用 Amazon IoT SiteWise API 为资产模型定义属性时，需要指定以下参数：

- `name` – 属性的名称。

- `defaultValue` – （可选）此属性的默认值。从模型创建的资产具有此属性值。有关如何覆盖从模型创建的资产中的默认值的更多信息，请参阅更新属性值。
- `dataType` – 属性的数据类型，如以下类型之一：
  - `STRING` – 最多 1024 字节的字符串。
  - `INTEGER` – 有符号的 32 位整数，范围是 [-2,147,483,648, 2,147,483,647]。
  - `DOUBLE` – 具有 [-10^100, 10^100] 范围和 IEEE 754 双精度的浮点数。
  - `BOOLEAN` – `true` 或 `false`。
- `externalId` -（可选）这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

Example 属性定义示例

以下示例演示一个表示具有默认值的资产型号的属性。此对象是包含属AssetModelProperty性的示例。您可以将此对象指定为 CreateAssetModel 请求负载的一部分，以创建属性特性。有关更多信息，请参阅 创建资产模型（Amazon CLI）。

```
{
...
"assetModelProperties": [
{
  "name": "Model number",
  "dataType": "STRING",
  "type": {
    "attribute": {
      "defaultValue": "BLT123"
    }
  }
}
],
...
}
```

定义来自设备的数据流（测量值）

测量值表示设备的原始传感器数据流，如带有时间戳的温度值或带有时间戳的每分钟转数 (RPM) 值。

主题

- 定义测量值（控制台）

- 定义测量值（Amazon CLI）

定义测量值（控制台）

在 Amazon IoT SiteWise 控制台中为资产模型定义测量值时，需要指定以下参数：

- 名称 – 属性的名称。
- 单位 –（可选）属性的科学单位，例如毫米或摄氏度。
- 数据类型 – 属性的数据类型，如以下类型之一：
  - 字符串 – 最多 1024 字节的字符串。
  - 整数 – 有符号的 32 位整数，范围是 [-2,147,483,648, 2,147,483,647]。
  - 双精度 – 具有 [-10^100, 10^100] 范围和 IEEE 754 双精度的浮点数。
  - 布尔值 – `true` 或 `false`。
- 外部 ID -（可选）这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

有关更多信息，请参阅 创建资产模型（控制台）。

定义测量值（Amazon CLI）

使用 Amazon IoT SiteWise API 定义资产模型的衡量标准时，需要指定以下参数：

- `name` – 属性的名称。
- `dataType` – 属性的数据类型，如以下类型之一：
  - `STRING` – 最多 1024 字节的字符串。
  - `INTEGER` – 有符号的 32 位整数，范围是 [-2,147,483,648, 2,147,483,647]。
  - `DOUBLE` – 具有 [-10^100, 10^100] 范围和 IEEE 754 双精度的浮点数。
  - `BOOLEAN` – `true` 或 `false`。
- `unit` –（可选）属性的科学单位，例如毫米或摄氏度。
- `externalId` -（可选）这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

Example 测量值定义示例

以下示例演示一个表示资产温度传感器读数的测量值。此对象是包含测量值AssetModelProperty的示例。您可以将此对象指定为 CreateAssetModel 请求负载的一部分，以创建测量值属性。有关更多信息，请参阅 创建资产模型（Amazon CLI）。

在定义资产模型时，测量值结构为空结构，因为您以后将每个资产配置为使用唯一的设备数据流。有关如何将资产的测量值属性连接到设备的传感器数据流的更多信息，请参阅管理数据流 Amazon IoT SiteWise。

```
{
      ...
      "assetModelProperties": [
      {
          "name": "Temperature C",
          "dataType": "DOUBLE",
          "type": {
              "measurement": {}
          },
          "unit": "Celsius"
      }
  ],
      ...
}
```

转换数据（转换）

转换是将资产属性的数据点从一种形式映射到另一种形式的数学表达式。转换表达式由资产属性变量、文字、运算符和函数组成。变换后的数据点与输入数据点有关 one-to-one系。 Amazon IoT SiteWise 每当任何输入属性接收到新的数据点时，都会计算一个新的转换数据点。

> **Note**
>
> 对于具有相同时间戳的属性更新，输出值可能会被来自其他传入属性的更新所覆盖。

例如，如果您的资产具有以摄氏度为单位的名为 `Temperature_C` 的温度测量流，您可以使用公式 `Temperature_F = 9/5 * Temperature_C + 32` 将每个数据点转换为华氏度。每次在`Temperature_C`测量流中 Amazon IoT SiteWise 接收到一个数据点时，相应的`Temperature_F`值将在几秒钟内计算出来，并作为`Temperature_F`属性提供。

如果您的转换包含多个变量，则较早到达的数据点会立即启动计算。举一个例子，其中零件制造商使用转换来监控产品质量。制造商根据零件类型使用不同的标准，使用以下测量值来表示过程：

- `Part_Number` - 用于标识零件类型的字符串。
- `Good_Count` - 一个整数，如果零件符合标准，则该整数将增加 1。
- `Bad_Count` - 一个整数，如果零件不符合标准，则该整数会增加 1。

制造商还创建了一个等于 `if(eq(Part_Number, "BLT123") and (Bad_Count / (Good_Count + Bad_Count) > 0.1), "Caution", "Normal")` 的转换 `Quality_Monitor`。

此转换可监控为特定零件类型生产的不良零件的百分比。如果部件号为， BLT123 且不良部件的百分比超过 10% (0.1)，则转换将返回`"Caution"`。否则，转换将返回 `"Normal"`。

> **Note**
>
> - 如果 `Part_Number` 在其他测量之前收到新的数据点，则 `Quality_Monitor` 转换将使用新 `Part_Number` 值和最新的 `Good_Count` 和 `Bad_Count` 值。为避免错误，请在下次制造运行之前重置 `Good_Count` 和 `Bad_Count`。
> - 如果您只想在所有变量都收到新的数据点后才计算表达式，请使用指标。

主题

- 定义转换（控制台）
- 定义转换（Amazon CLI）

定义转换（控制台）

在 Amazon IoT SiteWise 控制台中为资产模型定义转换时，需要指定以下参数：

- 名称 – 属性的名称。
- 单位 –（可选）属性的科学单位，例如毫米或摄氏度。
- 数据类型 - 转换的数据类型，可以是双精度或 字符串。
- 外部 ID -（可选）这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

- 公式 – 转换表达式。转换表达式不能使用聚合函数或时间函数。要打开自动完成特征，则开始键入或按向下箭头键。有关更多信息，请参阅 使用公式表达式。

> **Important**
>
> 转换可以输入整型、双精度、布尔或字符串类型的属性。布尔值转换为 0（假）和 1（真）。
>
> 转换必须输入一个或多个非特性属性。每次输入非特性属性接收新数据点时， Amazon IoT SiteWise 都会计算一个新的转换数据点。新属性值不会启动转换更新。资产属性数据 API 操作的请求速率同样适用于转换计算结果。
>
> 公式表达式只能输出双精度值或字符串值。嵌套表达式可以输出其他数据类型，例如字符串，但整个公式的计算结果必须为数字或字符串。您可以使用 jp 函数将字符串转换为数字。布尔值必须为 1（真）或 0（假）。有关更多信息，请参阅 未定义值、无限值和溢出值。

有关更多信息，请参阅 创建资产模型（控制台）。

定义转换（Amazon CLI）

使用 Amazon IoT SiteWise API 为资产模型定义转换时，需要指定以下参数：

- `name` – 属性的名称。
- `unit` –（可选）属性的科学单位，例如毫米或摄氏度。
- `dataType` – 转换的数据类型，必须为 `DOUBLE` 或`STRING`。
- `externalId` -（可选）这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。
- `expression` – 转换表达式。转换表达式不能使用聚合函数或时间函数。有关更多信息，请参阅 使用公式表达式。
- `variables` – 定义要在表达式中使用的资产的其他属性的变量列表。每个变量结构包含一个在表达式中使用的简单名称，以及一个指定哪个属性与该变量关联的 `value` 结构。`value` 结构包含以下信息：
  - `propertyId` – 要从中输入值的属性的 ID。您可以使用属性的名称而不是其 ID。

> **Important**
>
> 转换可以输入整型、双精度、布尔或字符串类型的属性。布尔值转换为 0（假）和 1（真）。

转换必须输入一个或多个非特性属性。每次输入非特性属性接收新数据点时， Amazon IoT SiteWise 都会计算一个新的转换数据点。新属性值不会启动转换更新。资产属性数据 API 操作的请求速率同样适用于转换计算结果。
公式表达式只能输出双精度值或字符串值。嵌套表达式可以输出其他数据类型，例如字符串，但整个公式的计算结果必须为数字或字符串。您可以使用 jp 函数将字符串转换为数字。布尔值必须为 1（真）或 0（假）。有关更多信息，请参阅 未定义值、无限值和溢出值。

Example 转换定义

以下示例演示了将资产的温度测量数据从摄氏度转换为华氏度的转换属性。此对象是包含变换AssetModelProperty的示例。您可以将此对象指定为 CreateAssetModel 请求负载的一部分，以创建转换属性。有关更多信息，请参阅 创建资产模型（Amazon CLI）。

```
{
...
"assetModelProperties": [
...
{
  "name": "Temperature F",
  "dataType": "DOUBLE",
  "type": {
    "transform": {
      "expression": "9/5 * temp_c + 32",
      "variables": [
        {
          "name": "temp_c",
          "value": {
            "propertyId": "Temperature C"
          }
        }
      ]
    }
  },
  "unit": "Fahrenheit"
}
],
...
}
```

Example 包含三个变量的转换定义

以下示例演示了一个变换属性，如果超过 10% 的 BLT123 部件不符合标准，则该属性会返回警告消息 ("Caution")。否则，它将返回一条信息消息 ("Normal")。

```
{
...
"assetModelProperties": [
...
{
"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"
                }
            }
        ]
    }
}
}
...
}
```

聚合来自属性和其他资产（指标）的数据

指标是使用聚合函数处理所有输入数据点并按指定时间间隔输出单个数据点的数学表达式。例如，指标可以从温度数据流计算每小时平均温度。

指标可以从关联资产的指标中输入数据，这样您可以计算统计信息，从而提供对整个操作或部分操作的见解。例如，指标可以计算风电场中所有风力涡轮机的每小时平均温度。有关如何定义资产之间的关联的更多信息，请参阅定义资产模型层次结构。

指标也可以输入来自其他属性的数据，而无需在每个时间间隔内聚合数据。如果您在公式中指定了属性，则 Amazon IoT SiteWise 在计算公式时会使用该属性的最新值。如果您在公式中指定指标，则 Amazon IoT SiteWise 使用最后一个值作为计算公式的时间间隔。这意味着您可以定义 `OEE = Availability * Quality * Performance` 这样的指标，其中 `Availability`、`Quality` 和 `Performance` 都是同一资产模型上的其他指标。

Amazon IoT SiteWise 还会为所有资产属性自动计算一组基本聚合指标。要降低计算成本，您可以使用这些聚合，而不是为基本计算定义自定义指标。有关更多信息，请参阅 在中查询资产属性聚合 Amazon IoT SiteWise。

主题

- 定义指标（控制台）
- 定义指标（Amazon CLI）

定义指标（控制台）

在 Amazon IoT SiteWise 控制台中为资产模型定义指标时，需要指定以下参数：

- 名称 – 属性的名称。
- 数据类型 - 转换的数据类型，可以是双精度或 字符串。
- 外部 ID -（可选）这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。
- 公式 – 指标表达式。指标表达式可以使用聚合函数从属性中输入层次结构中所有关联资产的数据。开始键入或按向下箭头键以打开自动完成功能。有关更多信息，请参阅 使用公式表达式。

> **Important**
>
> 指标只能输入整型、双精度、布尔或字符串类型的属性。布尔值转换为 `0`（假）和 `1`（真）。

如果您在指标表达式中定义任何指标输入变量，这些输入必须与输出指标具有同一个时间间隔。

公式表达式只能输出双精度值或字符串值。嵌套表达式可以输出其他数据类型，例如字符串，但整个公式的计算结果必须为数字或字符串。您可以使用 jp 函数将字符串转换为数字。布尔值必须为 1（真）或 0（假）。有关更多信息，请参阅 未定义值、无限值和溢出值。

- 时间间隔 – 时间间隔指标。Amazon IoT SiteWise 支持以下滚动窗口时间间隔，其中每个间隔在前一个间隔结束时开始：
  - 1 分钟 – 1 分钟，在每分钟结束时计算（凌晨 12:00:00、凌晨 12:01:00、凌晨 12:02:00，依此类推）。
  - 5 分钟 – 5 分钟，在每 5 分钟结束时计算，从整点开始（中午 12:00:00、中午 12:05:00、中午 12:10:00，依此类推）。
  - 15 分钟 – 15 分钟，在每 15 分钟结束时计算，从整点开始（凌晨 12:00:00、凌晨 12:15:00、凌晨 12:30:00，依此类推）。
  - 1 小时 – 1 小时（60 分钟），在每小时结束时计算，采用 UTC 格式（凌晨 12:00:00、凌晨 01:00:00、凌晨 02:00:00，依此类推）。
  - 1 天 – 1 天（24 小时），在每天结束时计算，采用 UTC 格式（星期一凌晨 12:00:00、星期二凌晨 12:00:00，依此类推）。
  - 1 周 – 1 周（7 天），在每个星期日结束时计算，采用 UTC 格式（每个星期一凌晨 12:00:00）。
  - 自定义间隔 - 您可以输入一分钟到一周之间的任何时间间隔。
- 偏移日期 - 或者，聚合数据的参考日期。
- 偏移时间 - 或者，聚合数据的参考时间。偏移时间必须介于 00:00:00 和 23:59:59 之间。
- 偏移时区 - 或者，偏移的时区。如果未指定，默认偏移时区为通用协调时间 (UTC)。

支持的时区

- (UTC+ 00:00) 协调世界时间
- (UTC+ 01:00) 欧洲中部时间
- (UTC+ 02:00) 东欧
- (UTC03+:00) 东非时间
- (UTC+ 04:00) 近东时间
- (UTC+ 05:00) 巴基斯坦拉合尔时间
- (UTC+ 05:30) 印度标准时间
- 孟加拉国标准时间

- (UTC+ 07:00) 越南标准时间
- (UTC+ 08:00) 中国台北时间
- (UTC+ 09:00) 日本标准时间
- (UTC+ 09:30) 澳大利亚中部时间
- (UTC+ 10:00) 澳大利亚东部时间
- (UTC+ 11:00) 所罗门标准时间
- (UTC+12:00) 新西兰标准时间
- (UTC-11:00) 中途岛时间
- (UTC-10:00) 夏威夷标准时间
- (UTC-09:00) 阿拉斯加标准时间
- (UTC-08:00) 太平洋标准时间
- (UTC-07:00) 凤凰城标准时间
- (UTC-06:00) 中部标准时间
- (UTC-05:00) 东部标准时间
- (UTC-04:00) 波多黎各和美属维尔京群岛时间
- (UTC-03:00) 阿根廷标准时间
- (UTC-02:00) 南乔治亚时间
- (UTC-01:00) 中非时间

Example 带偏移量的自定义时间间隔（控制台）

以下示例向您展示如何定义 12 小时的时间间隔，偏移时间为 2021 年 2 月 20 日下午 6:30:30（太平洋标准时间）。

使用偏移量定义自定义间隔

1. 对于 时间间隔，选择自定义间隔。
2. 对于 时间间隔，执行以下操作之一：
   - 输入 **12**，然后选择 小时。
   - 输入 **720**，然后选择 分钟。
   - 输入 **43200**，然后选择 秒。

> **Important**
>
> 无论单位如何，时间间隔都必须为整数。

3. 对于 偏移日期，选择 2021 年 2 月 20 日。
4. 对于 偏移时间，输入 **18:30:30**。
5. 对于 偏移时区，选择 (UTC-08:00) 太平洋标准时间。

如果您在 2021 年 7 月 1 日下午 06:30:30（太平洋标准时间）或之前创建指标，您将在 2021 年 7 月 1 日下午 06:30:30（太平洋标准时间）获得第一个聚合结果。第二个聚合结果将在 2021 年 7 月 2 日上午 06:30:30（太平洋标准时间）获得，依此类推。

定义指标（Amazon CLI）

使用 Amazon IoT SiteWise API 为资产模型定义指标时，需要指定以下参数：

- `name` – 属性的名称。
- `dataType` – 指标的数据类型，必须为 `DOUBLE` 或`STRING`。
- `externalId` -（可选）这是用户定义的 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。
- `expression` – 指标表达式。指标表达式可以使用聚合函数从属性中输入层次结构中所有关联资产的数据。有关更多信息，请参阅 使用公式表达式。
- `window` – 指标翻滚窗口的时间间隔和偏移量，其中每个间隔从前一个间隔结束时开始：
  - `interval` – 滚动窗口的时间间隔。间隔时间必须介于 1 分钟和 1 周之间。
  - `offsets` – 翻滚窗口的偏移。

  有关更多信息，请参阅 Amazon IoT SiteWise API 参考中的 TumblingWindow。

  Example 带有偏移量 (Amazon CLI) 的自定义时间间隔

  以下示例向您展示如何定义 12 小时的时间间隔，偏移时间为 2021 年 2 月 20 日下午 06:30:30（太平洋标准时间）。

```
{
```

```
    "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（太平洋标准时间）或之前创建指标，您将在 2021 年 7 月 1 日下午 06:30:30（太平洋标准时间）获得第一个聚合结果。第二个聚合结果将在 2021 年 7 月 2 日上午 06:30:30（太平洋标准时间）获得，依此类推。

- `variables` – 定义要在表达式中使用的资产或子资产的其他属性的变量列表。每个变量结构包含一个在表达式中使用的简单名称，以及一个指定哪个属性与该变量关联的 `value` 结构。`value` 结构包含以下信息：
  - `propertyId` – 从中提取值的属性的 ID。如果属性是在当前模型中定义的（而不是在层次结构的模型中定义的），您可以使用属性的名称而不是其 ID。
  - `hierarchyId` –（可选）从中查询属性的子资产的层次结构的 ID。您可以使用层次结构定义的名称而不是其 ID。如果省略此值，则会在当前模型中 Amazon IoT SiteWise 找到该属性。

> **Important**
>
> 指标只能输入整型、双精度、布尔或字符串类型的属性。布尔值转换为 `0`（假）和 `1`（真）。
>
> 如果您在指标表达式中定义任何指标输入变量，这些输入必须与输出指标具有同一个时间间隔。
>
> 公式表达式只能输出双精度值或字符串值。嵌套表达式可以输出其他数据类型，例如字符串，但整个公式的计算结果必须为数字或字符串。您可以使用 jp 函数将字符串转换为数字。布尔值必须为 1（真）或 0（假）。有关更多信息，请参阅 未定义值、无限值和溢出值。

- `unit` –（可选）属性的科学单位，例如毫米或摄氏度。

Example 指标定义示例

以下示例演示一个指标属性，该属性可聚合资产的温度测量值数据，以计算每小时最高温度（以华氏度为单位）。此对象是包含指标AssetModelProperty的示例。您可以将此对象指定为 CreateAssetModel 请求负载的一部分，以创建指标属性。有关更多信息，请参阅 创建资产模型（Amazon 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 输入关联资产数据的指标定义示例

以下示例演示了一个指标属性，该属性汇总了多台风力涡轮机的平均功率数据，以计算风电场的总平均功率。此对象是包含指标AssetModelProperty的示例。您可以将此对象指定为 CreateAssetModel 请求负载的一部分，以创建指标属性。

```
{
      ...
      "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"
      }
    ],
    ...
}
```

使用公式表达式

通过使用公式表达式，您可以定义数学函数以转换和聚合原始工业数据，从而深入了解您的运营情况。公式表达式结合使用文字、运算符、函数和变量来处理数据。有关如何定义使用公式表达式的资产属性的更多信息，请参阅转换数据（转换）和聚合来自属性和其他资产（指标）的数据。转换和指标是公式属性。

主题

- 在公式表达式中使用变量
- 在公式表达式中使用文字
- 在公式表达式中使用运算符
- 在公式表达式中使用常量
- 在公式表达式中使用函数
- 公式表达式教程

在公式表达式中使用变量

变量表示公式表达式中的 Amazon IoT SiteWise 资产属性。使用变量在表达式中输入来自其他资产属性的值，以便您可以处理来自常量属性（属性）、原始数据流（测量）和其他公式属性的数据。

变量可以表示来自同一资产模型或关联子资产模型的资产属性。只有指标公式才能输入子资产模型中的变量。

您可以在控制台和 API 中使用不同的名称来识别变量。

- Amazon IoT SiteWise 控制台-在表达式中使用资产属性名称作为变量。
- Amazon IoT SiteWise API (Amazon CLI, Amazon SDKs)-使用ExpressionVariable结构定义变量，该结构需要变量名和对资产属性的引用。变量名称可包含大小写字母、数字和下划线。然后，使用变量名在表达式中引用资产属性。

变量名称区分大小写。

有关更多信息，请参阅定义转换和定义指标。

使用变量来引用属性

变量的值定义了它所引用的属性。 Amazon IoT SiteWise 提供了不同的方式来实现这一目的。

- 按属性 ID：您可以指定属性的唯一 ID（UUID）来对其进行标识。
- 按名称：如果该属性属于同一资产模型，则可以在属性 ID 字段中指定其名称。
- 按路径：变量值可以通过其路径来引用属性。有关更多信息，请参阅 使用路径来引用自定义复合模型属性。

> **Note**
>
> Amazon IoT SiteWise 控制台不支持变量。它们由 Amazon IoT SiteWise API 使用，包括 Amazon Command Line Interface Amazon CLI) 和 Amazon SDKs。

您在响应中收到的变量 Amazon IoT SiteWise 包含有关该值的完整信息，包括 ID 和路径。

但是，当您将变量传递到时 Amazon IoT SiteWise （例如，在 “创建” 或 “更新” 调用中），您只需要指定其中一个即可。例如，如果您指定路径，则无需提供 ID。

在公式表达式中使用文字

Amazon IoT SiteWise 支持在表达式和公式中使用文字。文字是表示特定数据类型的固定值。在中 Amazon IoT SiteWise，您可以在公式表达式中定义数字和字符串文字。文字可以在各种环境中使用，包括数据转换、警报条件和可视化计算。

- 数字

  使用数字和科学记数法来定义整数和双精度。您可以使用 E 记数法来表示带有科学记数法的数字。

  示例：`1`、`2.0`、`.9`、`-23.1`、`7.89e3`、`3.4E-5`

- 字符串

  使用 '（引号）和 "（双引号）字符来定义字符串。起点和结尾的报价类型必须匹配。要对与您用来声明字符串的引号相匹配的引号进行转义，请将该引号字符包含两次。这是 Amazon IoT SiteWise 字符串中唯一的转义字符。

  示例：`'active'`、`"inactive"`、`'{"temp": 52}'`、`"{""temp"": ""high""}"`。

在公式表达式中使用运算符

您可以在公式表达式中使用以下常见的运算符。

运算符	描述
+	如果两个操作数都是数字，则此运算符将左右操作数相加。 如果任一操作数是字符串，则此运算符将左右操作数连接为字符串。例如，表达式的 `1 + 2 + " is three"` 计算结果为 `"3 is three"`。连接字符串最多可以包含 1024个字符。如果字符串超过 1024 个字符，则 Amazon IoT SiteWise 不会为该计算输出数据点。
-	从左侧操作数中减去右侧操作数。

运算符	描述
	只能将此运算符与数字操作数一起使用。
/	左侧操作数除以右侧操作数。 只能将此运算符与数字操作数一起使用。
*	将左右操作数相乘。 只能将此运算符与数字操作数一起使用。
^	将左操作数作为右操作数的幂（指数）。 只能将此运算符与数字操作数一起使用。
%	返回左侧操作数除以右侧操作数得到的余数。结果具有与左侧操作数相同的符号。此行为不同于 modulo 操作。 只能将此运算符与数字操作数一起使用。
`x < y`	如果 `x` 小于 `y`，则返回 `1`，否则，返回 `0`。
`x > y`	如果 `x` 大于 `y`，则返回 `1`，否则，返回 `0`。
`x <= y`	如果 `x` 小于或等于 `y`，则返回 `1`，否则，返回 `0`。
`x >= y`	如果 `x` 大于或等于 `y`，则返回 `1`，否则，返回 `0`。
`x == y`	如果 `x` 等于 `y`，则返回 `1`，否则，返回 `0`。
`x != y`	如果 `x` 不等于 `y`，则返回 `1`，否则，返回 `0`。

运算符	描述
`!x`	如果 x 等于 0（假），则返回 1，否则，返回 0。 如果出现以下情况，则 x 的计算结果为假： • x 是一个数字操作数，其计算结果为 0。 • x 计算结果为空字符串。 • x 计算结果为空数组。 • x 计算结果为 None。
`x and y`	如果 x 的计算结果为 0（假），则返回 0。否则，返回 y 的计算结果。 如果出现以下情况，则 x 或 y 的计算结果为假： • x 或 y 是一个数字操作数，其计算结果为 0。 • x 或 y 的计算结果为空字符串。 • x 或 y 的计算结果为空数组。 • x 或 y 的计算结果为 None。
`x or y`	如果 x 等于 1（真），则返回 1。否则，返回 y 的计算结果。 如果出现以下情况，则 x 或 y 的计算结果为假： • x 或 y 是一个数字操作数，其计算结果为 0。 • x 或 y 的计算结果为空字符串。 • x 或 y 的计算结果为空数组。 • x 或 y 的计算结果为 None。

运算符	描述
`not x`	如果 x 等于 0（假），则返回 1，否则，返回 0。 如果出现以下情况，则 x 的计算结果为假： • x 是一个数字操作数，其计算结果为 0。 • x 计算结果为空字符串。 • x 计算结果为空数组。 • x 计算结果为 None。

`[]`
 	
`s[index]`
	返回字符串 s 索引 index 上的字符。这等同于 Python 中的索引语法。
	
	Example 示例
	
	• `"Hello!"[1]` 返回 e。
	• `"Hello!"[-2]` 返回 o。

运算符	描述

`[]`
 	
`s[start:end:step]`
	返回字符串 `s` 的片段。这等同于 Python 中的切片语法。此运算符具有以下参数：
	
	• `start` – 或者，切片的包含起始索引。默认值为 `0`。
	• `end` – 或者，切片的独占结束索引。默认为字符串的长度。
	• `step` – 或者，切片中每个步骤要递增的数字。例如，您可以指定 2 返回一个包含所有其他字符的切片，或者指定 `-1` 反转切片。默认值为 1。
	
	您可以省略 `step` 参数以使用其默认值。例如，`s[1:4:1]` 等同于 `s[1:4]`。
	
	参数必须是整数或 无 常量。如果指定none，则 Amazon IoT SiteWise 使用该参数的默认值。
	
	Example 示例
	
	• `"Hello!"[1:4]` 返回 `"ell"`。
	• `"Hello!"[:2]` 返回 `"He"`。
	• `"Hello!"[3:]` 返回 `"lo!"`。
	• `"Hello!"[:-4]` 返回 `"He"`。
	• `"Hello!"[::2]` 返回 `"Hlo"`。
	• `"Hello!"[::-1]` 返回 `"!olleH"`。

在公式表达式中使用常量

在中 Amazon IoT SiteWise，您可以在表达式和公式中使用常量来表示固定值或预定义参数。常量可以在各种环境中使用，例如数据转换、警报条件或可视化计算。通过使用常量，您可以简化表达式，使其更具可读性和可维护性。

您可以在表达式中使用以下常见的数学常数。所有常量均不区分大小写。

> **Note**
> 如果定义的变量与常量同名，则该变量会覆盖该常量。

常量	描述
`pi`	数字 pi (π)：`3.141592653589793`
`e`	数字 e：`2.718281828459045`
`true`	等同于数字 1。在中 Amazon IoT SiteWise，布尔值转换为其等效数字。
`false`	等同于数字 0。在中 Amazon IoT SiteWise，布尔值转换为其等效数字。
`none`	等同于无值。您可以使用此常量将任何内容输出为条件表达式的结果。

在公式表达式中使用函数

您可以使用以下函数对公式表达式中的数据进行操作。

转换和指标支持不同的功能。下表显示了哪些类型的函数与每种类型的公式属性兼容。

> **Note**
> 一个公式表达式中最多可以包含 10 个函数。

函数类型	转换	Metrics
在公式表达式中使用常用函数	是	是

函数类型	转换	Metrics
在公式表达式中使用比较函数	是	是
在公式表达式中使用条件函数	是	是
在公式表达式中使用字符串函数	是	是
在公式表达式中使用聚合函数	没有	是
在公式表达式中使用时间函数	是	是
在公式表达式中使用日期和时间函数	是	是

函数语法

您可以使用以下语法来创建函数：

常规语法

在常规语法中，函数名称后面是带有零个或多个参数的圆括号。

`function_name(argument1, argument2, argument3, ...)`。例如，使用常规语法的函数可能看起来像 `log(x)` 和 `contains(s, substring)`。

统一函数调用语法 (UFCS)

UFCS 允许您在面向对象编程中使用方法调用的语法来调用函数。在 UFCS 中，第一个参数后面跟着点 (`.`)，然后是函数名和括号内的其余参数（如果有）。

`argument1.function_name(argument2, argument3, ...)`。例如，带有 UFCS 的函数可能看起来像 `x.log()` 和 `s.contains(substring)`。

您也可以使用 UFCS 来链接后续函数。 Amazon IoT SiteWise 使用当前函数的计算结果作为下一个函数的第一个参数。

例如，您可以使用 `message.jp('$.status').lower().contains('fail')`，而不使用 `contains(lower(jp(message, '$.status')),'fail')`。

有关更多信息，请参阅 D Programming Language 页面。

> **Note**
>
> 您可以将 UFCS 用于所有 Amazon IoT SiteWise 功能。
>
> Amazon IoT SiteWise 函数不区分大小写。例如，可以互换使用 `lower(s)` 和 `Lower(s)`。

在公式表达式中使用常用函数

在转换和指标中，您可以使用以下函数来计算转换和指标中的常见数学函数。

函数	描述
`abs(x)`	返回 x 的绝对值。
`acos(x)`	返回 x 的反余弦值。
`asin(x)`	返回 x 的反正弦值。

函数	描述
atan(x)	返回 x 的反正切值。
cbrt(x)	返回 x 的立方根。
ceil(x)	返回大于 x 的最接近整数。
cos(x)	返回 x 的余弦值。
cosh(x)	返回 x 的双曲余弦值。
cot(x)	返回 x 的余切值。
exp(x)	返回 e 的 x 次幂。
expm1(x)	返回 exp(x) - 1。对于较小的 exp(x) - 1 值，使用此函数可以更准确地计算 x。
floor(x)	返回小于 x 的最接近整数。
log(x)	返回 x 的 $\log_e$（以 e 为底）。
log10(x)	返回 x 的 $\log_{10}$（以 10 为底）。
log1p(x)	返回 log(1 + x)。对于较小的 log(1 + x) 值，使用此函数可以更准确地计算 x。
log2(x)	返回 x 的 $\log_2$（以 2 为底）。
pow(x, y)	返回 x 的 y 次幂。这等同于 x ^ y。
signum(x)	返回 x 的符号（-1 表示负输入，0 表示零输入，+1 表示正输入）。
sin(x)	返回 x 的正弦值。
sinh(x)	返回 x 的双曲正弦值。
sqrt(x)	返回 x 的平方根。

函数	描述
`tan(x)`	返回 x 的正切值。
`tanh(x)`	返回 x 的双曲正切值。

在公式表达式中使用比较函数

在变换和指标中，您可以使用以下比较函数来比较两个值并输出1（真）或0（假）。 Amazon IoT SiteWise 按字典顺序比较字符串。

函数	描述
`gt(x, y)`	如果 x 大于 y，则返回 1，否则，返回 0 (x > y)。 如果 x 和 y 的类型不兼容，例如数字和字符串，则此函数不返回值。
`gte(x, y)`	如果 x 大于或等于 y，则返回 1，否则，返回 0 (x ≥ y)。 Amazon IoT SiteWise 如果参数在相对容差范围内，则认为它们是相等的1E-9。其行为类似于 Python 中的 isclose 函数。 如果 x 和 y 的类型不兼容，例如数字和字符串，则此函数不返回值。
`eq(x, y)`	如果 x 等于 y，则返回 1，否则，返回 0 (x == y)。 Amazon IoT SiteWise 如果参数在相对容差范围内，则认为它们是相等的1E-9。其行为类似于 Python 中的 isclose 函数。 如果 x 和 y 的类型不兼容，例如数字和字符串，则此函数不返回值。

函数	描述
`lt(x, y)`	如果 x 小于 y，则返回 1，否则，返回 0 (x < y)。 如果 x 和 y 的类型不兼容，例如数字和字符串，则此函数不返回值。
`lte(x, y)`	如果 x 小于或等于 y，则返回 1，否则，返回 0 (x ≤ y)。 Amazon IoT SiteWise 如果参数在相对容差范围内，则认为它们是相等的1E-9。其行为类似于 Python 中的 isclose 函数。 如果 x 和 y 的类型不兼容，例如数字和字符串，则此函数不返回值。
`isnan(x)`	如果 x 等于 NaN，则返回 1，否则，返回 0。 如果 x 是字符串，则此函数不返回值。

在公式表达式中使用条件函数

在转换和指标中，您可以使用以下函数来检查条件并返回不同的结果，无论该条件的计算结果为真还是假。

函数	描述
`if(condition, result_if_true, result_if_false)`	计算 condition ，如果条件计算结果为真，则返回 result_if_true ，如果条件计算结果为 false，则返回 result_if_false 。 condition 必须是数字。此函数将 0 和一个空字符串视为 false，其他所有内容（包括 NaN）都视为 true。布尔值转换为 0（假）和 1（真）。

函数	描述
	您可以从此函数返回 无 常量，以丢弃特定条件的输出。这意味着您可以筛选出不符合条件的数据点。有关更多信息，请参阅 筛选数据点。 Example 示例 • `if(0, x, y)` 返回变量 `y`。 • `if(5, x, y)` 返回变量 `x`。 • 如果变量 `temp` 大于 `300`，则 `if(gt(temp, 300), x, y)` 返回变量 `x`。 • 如果 `if(gt(temp, 300), temp, none)` 大于或等于 `300`，则返回变量 `temp`，如果 `temp` 小于 `300`，则返回 `none`（无值）。 我们建议您将 UFCS 用于嵌套条件函数，其中一个或多个参数是条件函数。您可以使用 `if(condition, result_if_true)` 计算条件和 `elif(condition, result_if_true, result_if_false)` 计算其他条件。 例如，您可以使用 `if(condition1, result1_if_true).elif(condition2, result2_if_true, result2_if_false)`，而不使用 `if(condition1, result1_if_true, if(condition2, result2_if_true, result2_if_false))`。 您还可以链接其他中间条件函数。例如，您可以使用 `if(condition1, result1_if_true).elif(condition2, result2_if_true).elif(condition3, result3_if_true, result3_i`

函数	描述
	f_false) 而不是嵌套多个 if 语句，例如 if(condition1, result1_if_true, if(condition2, result2_if_true, if(condition3, result3_if_true result3_if_false))) 。 **Important** 您必须将 elif(condition, result_if_true, result_if _false) 与 UFCS 一起使用。

在公式表达式中使用字符串函数

在转换和指标中，您可以使用以下函数对字符串进行操作。有关更多信息，请参阅 在公式中使用字符串。

> **Important**
> 公式表达式只能输出双精度值或字符串值。嵌套表达式可以输出其他数据类型，例如字符串，但整个公式的计算结果必须为数字或字符串。您可以使用 jp 函数将字符串转换为数字。布尔值必须为 1（真）或 0（假）。有关更多信息，请参阅 未定义值、无限值和溢出值。

函数	描述
len(s)	返回字符串 s 的长度。
find(s, substring)	返回字符串 substring 中字符串 s 的索引。
contains(s, substring)	如果字符串 s 包含字符串 substring ，则返回 1，否则，返回 0。
upper(s)	以大写形式返回字符串 s。

函数	描述
`lower(s)`	以小写形式返回字符串 s。

函数	描述
`jp(s, json_path)`	s使用JsonPath表达式计算字符串`json_path` 并返回结果。 请使用该函数执行以下操作： • 从序列化的 JSON 结构中提取值、数组或对象。 • 将字符串转换为数字。例如，公式 `jp('111', '$')` 以数字形式返回 111。 要从 JSON 结构中提取字符串值并将其作为数字返回，必须使用多个嵌套 `jp` 函数。外部 `jp` 函数从 JSON 结构中提取字符串，内部 `jp` 函数将字符串转换为数字。 该字符串 `json_path` 必须包含一个字符串文字。这意味着 `json_path` 不能是计算结果为字符串的表达式。 Example 示例 • `jp('{"status":"active","value":15}', '$.value')` 返回 15。 • `jp('{"measurement":{"reading":25,"confidence":0.95}}', '$.measurement.reading')` 返回 25。 • `jp('[2,8,23]', '$[2]')` 返回 23。 • `jp('{"values":[3,6,7]}', '$.values[1]')` 返回 6。 • `jp('111', '$')` 返回 111。 • `jp(jp('{"measurement":{"reading":25,"confidence":"0.95`

函数	描述
	"}}', '$.measurement.confidence'), '$') 返回 0.95。
join(s0, s1, s2, s3, ...)	返回带有一个分隔符的串联字符串。此函数使用第一个输入字符串作为分隔符，并将剩余的输入字符串连接在一起。其行为类似于 Java 中的连接（CharSequence 分隔符， CharSequence... 元素）函数。 Example 示例 • join("-", "aa", "bb", "cc") 返回 aa-bb-cc
format(expression: "format") 或 format("format", expression)	返回指定格式的字符串。此函数将 expression 求值，然后以指定格式返回该值。其行为类似于 Java 中的 format(String format, Object... args) 函数。有关支持的格式的更多信息，请参阅 Java 平台标准版 7 API 规范中的类格式化程序下的转换。 Example 示例 • format(100+1: "d") 返回字符串，101。 • format("The result is %d", 100+1) 返回字符串，The result is 101。

函数	描述
`f'expression'`	返回一个连接的字符串。使用此格式化函数，您可以使用简单的表达式来连接和格式化字符串。这些函数可能包含嵌套表达式。可以使用 `{}`（大括号）对表达式进行插值。这与 Python 中的格式化字符串字面值类似。 Example 示例 • `f'abc{1+2: "f"}d'` 返回 `abc3.000000d` 。要计算此示例表达式，请执行以下操作： 1. `format(1+2: "f")` 返回浮点数 `3.000000`。 2. `join('', "abc", 1+2, 'd')` 返回字符串，`abc3.000000d` 。 您也可以用以下方式编写表达式：`join('', "abc", format(1+2: "f"), 'd')` 。

在公式表达式中使用聚合函数

仅在指标中，您可以使用以下函数以聚合每个时间间隔的输入值，并计算单个输出值。聚合函数可以从关联资产聚合数据。

聚合函数参数可以是变量、数字文字、时间函数、嵌套表达式或聚合函数。公式 `max(latest(x), latest(y), latest(z))` 使用聚合函数作为参数，并返回 x、y、和 z 属性的最大当前值。

可以在聚合函数中使用嵌套表达式。当您使用嵌套表达式时，以下规则将适用：

- 每个参数都只能有一个变量。

  Example

  例如，支持 `avg(x*(x-1))` 和 `sum(x/2 )/avg(y^2 )`。

  例如，不支持 `min(x/y)`。
- 每个参数可以有多级嵌套表达式。

Example

例如，支持 `sum(avg(x^2 )/2)`。

- 不同的参数可以有不同的变量。

Example

例如，支持 `sum(x/2, y*2)`。

> **Note**
>
> - 如果您的表达式包含测量值，则 Amazon IoT SiteWise 使用当前时间间隔内测量值的最后一个值来计算聚合。
> - 如果您的表达式包含属性，则 Amazon IoT SiteWise 使用属性的最新值来计算聚合。

函数	描述
avg(x_0, ..., x_n)	返回给定变量在当前时间间隔内的平均值。 仅当给定变量在当前时间间隔内至少有一个数据点时，此函数才会输出数据点。
sum(x_0, ..., x_n)	返回给定变量在当前时间间隔内的值总和。 仅当给定变量在当前时间间隔内至少有一个数据点时，此函数才会输出数据点。
min(x_0, ..., x_n)	返回给定变量在当前时间间隔内的最小值。 仅当给定变量在当前时间间隔内至少有一个数据点时，此函数才会输出数据点。
max(x_0, ..., x_n)	返回给定变量在当前时间间隔内的最大值。 仅当给定变量在当前时间间隔内至少有一个数据点时，此函数才会输出数据点。

函数	描述
count(x_0, ..., x_n)	返回给定变量在当前时间间隔内的数据点总数。有关如何计算满足条件的数据点数量的更多信息，请参阅计算与条件匹配的数据点。 此函数会计算每个时间间隔的数据点。
stdev(x_0, ..., x_n)	返回给定变量在当前时间间隔内的标准差。 仅当给定变量在当前时间间隔内至少有一个数据点时，此函数才会输出数据点。

在公式表达式中使用时间函数

使用时间函数根据数据点的时间戳返回值。

在指标中使用时间函数

仅在指标中，您可以使用以下函数，这些函数根据数据点的时间戳返回值。

时态函数参数必须是来自本地资产模型或嵌套表达式的属性。这意味着您不能在时态函数中使用子资源模型中的属性。

可以在时态函数中使用嵌套表达式。当您使用嵌套表达式时，以下规则将适用：

- 每个参数都只能有一个变量。

  例如，支持 `latest( t*9/5 + 32 )`。
- 参数不能是聚合函数。

  例如，不支持 `first( sum(x) )`。

函数	描述
first(x)	返回给定变量在当前时间间隔内具有最早时间戳的值。

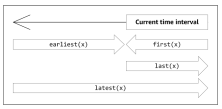
函数	描述
`last(x)`	返回给定变量在当前时间间隔内具有最晚时间戳的值。
`earliest(x)`	返回给定变量在当前时间间隔开始之前的最后一个值。 如果输入属性在其历史记录中具有至少一个数据点，则此函数将计算每个时间间隔的数据点。有关详细信息，请参阅time-range-defintion。
`latest(x)`	返回给定变量在当前时间间隔结束之前具有最晚时间戳的最后一个值。 如果输入属性在其历史记录中具有至少一个数据点，则此函数将计算每个时间间隔的数据点。有关详细信息，请参阅time-range-defintion。
`statetime(x)`	返回给定变量在当前时间间隔内为正值的时间（以秒为单位）。您可以使用比较函数创建转换属性，以供 `statetime` 函数使用。 例如，如果 `Idle` 属性为 `0` 或 `1`，您可以使用以下表达式计算每个时间间隔的闲置时间：`IdleTime = statetime(Idle)` 。有关更多信息，请参阅 statetime 使用情景示例。 此函数不支持将指标属性作为输入变量。 如果输入属性在其历史记录中具有至少一个数据点，则此函数将计算每个时间间隔的数据点。

函数	描述
`TimeWeightedAvg(x, [interpolation])`	返回输入数据的平均值，使用点之间的时间间隔进行加权。 有关计算和间隔的详细信息，请参阅时间加权函数参数。 可选参数 `interpolaton` 必须是字符串常量： • `locf` – 这是默认值。该计算使用上次观测到的向前结转计算算法计算数据点之间的间隔。在这种方法中，数据点是作为最后一个观测值计算的，直到下一个输入数据点的时间戳。 将一个好的数据点之后的值外推为其值，直到下一个数据点的时间戳。 • `linear` – 计算使用线性插值计算算法计算数据点之间的间隔。 两个良好数据点之间的值是通过这些数据点的值之间的线性插值推断出来的。 良好数据点和不良数据点之间的值或最后一个良好数据点之后的值，将被推断为一个良好数据点。

函数	描述
`TimeWeightedStDev(x, [algo])`	返回输入数据的标准差，使用点之间的时间间隔进行加权。 有关计算和间隔的详细信息，请参阅时间加权函数参数。 该计算使用上次观测到的向前结转计算算法计算数据点之间的间隔。在这种方法中，数据点是作为最后一个观测值计算的，直到下一个输入数据点的时间戳。权重计算为数据点或窗口边界之间的时间间隔（以秒为单位）。 可选参数 `algo` 必须是字符串常量： • f – 这是默认值。它返回带有频率权重的无偏加权样本方差，其中以秒 TimeWeight 为单位计算。这种算法通常是在标准差下假设的，被称为加权样本的贝塞尔标准差校正。 • p – 返回有偏差的加权样本方差，也称为总体方差。 以下公式用于计算，其中： • S_p = 总体标准差 • S_f = 频率标准差 • X_i = 传入数据 • ω_i = 等于以秒为单位的时间间隔的权重 • μ^* = 传入数据的加权平均值 总体标准差方程： $$S_p^2 = \frac{\sum_{i=1}^{N} \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^{N} \omega_i}$$

函数	描述
	频率标准差方程： $$S_f^2 = \frac{\sum_{i=1}^{N} \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^{N} \omega_i - 1}$$

下图显示了如何相对于当前时间间隔 Amazon IoT SiteWise 计算时间函数`first``last``earliest``latest`、、和。



> **Note**
>
> - `first(x)`, `last(x)` 的时间范围是 (当前窗口开始, 当前窗口结束]。
> - `latest(x)` 的时间范围是 (起始时间, 当前窗口结束]。
> - `earliest(x)` 的时间范围是 (起始时间, 上一窗口结束]。

时间加权函数参数

为聚合窗口计算的时间加权函数会考虑以下因素：

- 窗口内的数据点
- 数据点之间的时间间隔
- 窗口前的最后一个数据点
- 窗口之后的第一个数据点（对于某些算法）

术语：

- 不良数据点 - 任何质量不佳或非数字值的数据点。在窗口结果计算中不考虑这一点。
- 不良间隔 - 不良数据点之后的间隔。第一个已知数据点之前的间隔也被视为不良间隔。
- 良好数据点 - 任何具有良好质量和数值的数据点。

> **Note**
>
> - Amazon IoT SiteWise 仅在计算变换和指标时才消耗GOOD质量数据。它会忽略 UNCERTAIN 和 BAD 数据点。
> - 第一个已知数据点之前的间隔被视为不良间隔。请参阅the section called “公式表达式教程”了解更多信息。

最后一个已知数据点之后的间隔无限期地持续下去，影响所有后续窗口。当新的数据点到达时，该函数会重新计算间隔。

按照上述规则，将计算聚合窗口结果并仅限于窗口边界。默认情况下，只有当整个窗口都是良好间隔时，该函数才会发送窗口结果。

如果窗口良好间隔小于窗口长度，则该函数不会发送窗口。

当影响窗口结果的数据点发生变化时，即使数据点在窗口之外，该函数也会重新计算窗口。

如果输入属性的历史记录中至少有一个数据点，并且计算已启动，则该函数计算每个时间间隔的时间加权聚合函数。

Example statetime 使用情景示例

考虑这样一个使用情景示例，您有一个具有以下属性的资产：

- `Idle` – 一个为 `0` 或 `1` 的测量值。当该值为时 `1`，机器处于空闲状态。
- `Idle Time` – 一个指标，该指标使用 `statetime(Idle)` 公式计算机器在每分钟内处于空闲状态的时间（以秒为单位）。

`Idle` 属性具有以下数据点。

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

Amazon IoT SiteWise 根据的值每分钟计算一次`Idle Time`属性`Idle`。此计算完成后，`Idle Time` 属性将具有以下数据点。

Timestamp	2:00:00 PM	2:01:00 PM	2:02:00 PM	2:03:00 PM	2:04:00 PM
Idle Time	不适用	30	60	45	0

Amazon IoT SiteWise 在每分钟结束`Idle Time`时执行以下计算。

- At 2:00 PM (for 1:59 PM to 2:00 PM)
  - 在 2:00 PM 之前没有 `Idle` 数据，因此计算不会得出任何数据点。
- At 2:01 PM (for 2:00 PM to 2:01 PM)
  - 在 2:00:00 PM，机器处于活动状态（`Idle` 为 `0`）。
  - 在 2:00:30 PM，机器处于空闲状态（`Idle` 为 `1`）。
  - `Idle` 在 2:01:00 PM 的时间间隔结束之前没有再次变化，因此 `Idle Time` 是 30 秒。
- At 2:02 PM (for 2:01 PM to 2:02 PM)
  - 在 2:01:00 PM，机器处于空闲状态（基于 2:00:30 PM 的上一个数据点）。
  - 在 2:01:15 PM，机器仍处于空闲状态。
  - `Idle` 在 2:02:00 PM 的时间间隔结束之前没有再次变化，因此 `Idle Time` 是 60 秒。
- At 2:03 PM (for 2:02 PM to 2:03 PM)
  - 在 2:02:00 PM，机器处于空闲状态（基于 2:01:15 PM 的上一个数据点）。
  - 在 2:02:45 PM，机器处于活动状态。
  - `Idle` 在 2:03:00 PM 的时间间隔结束之前没有再次变化，因此 `Idle Time` 是 45 秒。
- At 2:04 PM (for 2:03 PM to 2:04 PM)
  - 在 2:03:00 PM，机器处于活动状态（基于 2:02:45 PM 的上一个数据点）。
  - `Idle` 在 2:04:00 PM 的时间间隔结束之前没有再次变化，因此 `Idle Time` 是 0 秒。

Example 示例 TimeWeightedAvg 和 TimeWeightedStDev 场景

下表提供了这些一分钟窗口指标的示例输入和输出：`Avg(x)`, `TimeWeightedAvg(x)`, `TimeWeightedAvg(x, "linear")`, `stDev(x)`, `timeWeightedStDev(x)`, `timeWeightedStDev(x, 'p')`。

一分钟聚合窗口的示例输入：

> **Note**
>
> 这些数据点的质量为 GOOD。

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

汇总结果输出：

> **Note**
>
> 无 – 未生成此窗口的结果。

Time	**Avg(x)**	**TimeWeightedAvg(x)**	**TimeWeightedAvg(X, "linear")**	**stDev(X)**	**timeWeightedStDev(x)**	**timeWeightedStDev(x, 'p')**
3:00:00	4	无	无	0	无	无
3:01:00	2	4	3	0	0	0

Time	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	8.5440037 4531753	7.724054437220943	7.659416862050705
3:04:00	0	10	2.5	0	10.084389681792215	10
3:05:00	无	0	0	无	0	0

在转换中使用时间函数

仅在转换中，您可以使用 `pretrigger()` 函数在启动当前转换计算的属性更新之前，检索变量的 GOOD 质量值。

以一个制造商 Amazon IoT SiteWise 用来监控机器状态的例子为例。制造商使用以下测量值和转换来表示过程：

- 测量值，`current_state`，可以是 0 或 1。
  - 如果本机处于清洁状态，则 `current_state` 等于 1。
  - 如果计算机处于制造状态，则 `current_state` 等于 0。
- 转换，`cleaning_state_duration`，等于 `if(pretrigger(current_state) == 1, timestamp(current_state) - timestamp(pretrigger(current_state)), none)`。此转换以 Unix 纪元格式返回机器处于清洁状态的时间（以秒为单位）。有关更多信息，请参阅 在公式表达式中使用条件函数 时间戳（ ） 函数。

如果机器处于清洁状态的时间超过预期，制造商可能会对机器进行调查。

您也可以在多变量转换中使用 `pretrigger()` 函数。例如，您有两个名为 x 和 y 的测量值和一个等于 `x + y + pretrigger(y)` 的转换 z。下表显示了 x、y 和 z 从上午 9:00 到 9:15 的值。

> **Note**
>
> - 此示例假设测量值按时间顺序到达。例如，上午 09:00 x 的值在上午 09:05 x 的值之前到达。
> - 如果上午 9:05 的数据点在上午 9:00 的数据点之前到达，则 z 不会在上午 9:05 进行计算。
> - 如果上午 9:05 x 的值在上午 09:00 x 的值之前到达，并且 y 的值按时间顺序到达，则 z 在上午 9:05 等于 `22 = 20 + 1 + 1`。

	09:00 AM	09:05 AM	09:10 AM	09:15 AM
x	10	20		30
y	1	2	3	
`z = x + y + pretrigger(y)`	y 在上午 09:00 之前没有收到任何数据点。因此，z 不是在上午 09:00 计算的。	23 = 20 + 2 + 1 `pretrigger(y)` 等于 1。	25 = 20 + 3 + 2 x 没有收到新的数据点。`pretrigger(y)` 等于 2。	36 = 30 + 3 + 3 y 没有收到新的数据点。因此，在上午 09:15 `pretrigger(y)` 等于 3。

在公式表达式中使用日期和时间函数

在转换和指标中，您可以通过以下方式使用日期和时间函数：

- 以 UTC 或本地时区检索数据点的当前时间戳。
- 使用参数（例如`year`、`month` 和 `day_of_month`）构造时间戳。
- 使用 `unix_time` 参数提取一个时间段，例如年或月。

函数	描述
now()	采用 Unix 纪元时间格式返回当前日期和时间（以秒为单位）。

函数	描述
`timestamp()`	• 在转换中，该函数采用 Unix 纪元时间格式的时间戳（以秒为单位）。 仅在转换中，您可以执行以下操作之一： • 提供一个变量作为函数的参数。`timestamp( variable-name )` 函数以 Unix 纪元格式返回指定变量的最新 GOOD 质量值的时间戳（以秒为单位）。 例如，如果您的资产有一个名为的转换属性 `Temperature_F` ，该属性使用 `9/5 * Temperature_C` 公式将每个温度数据点从摄氏度转换为华氏度，则可以使用 `timestamp(Temperature_F)` 函数获取 `Temperature_F` 属性最新 GOOD 质量值的时间戳。 • 使用 `pretrigger()` 函数作为函数的参数。`timestamp(pretrigger( variable-name ))` 函数返回以 Unix 纪元格式启动当前转换计算的属性更新之前，指定变量的 GOOD 质量值的时间戳（以秒为单位）。有关更多信息，请参阅在转换中使用时间函数。 • 在指标中，该函数以 Unix 纪元格式返回在当前窗口末尾检索到的时间戳（以秒为单位）。

函数	描述
`mktime(time_zone, year, month, day_of_month, hour, minute, second)`	返回 Unix 纪元时间格式的输入时间（以秒为单位）。 使用此函数存在以下要求： • 时区参数必须是带引号的字符串 (`'UTC'`)。如果未指定，默认时区为 UTC。 时区参数可以是第一个或最后一个参数。 • 年、月、月、日、时、分和秒参数必须按顺序排列。 • 需要年、月和日参数。 使用此函数存在以下限制： • `year` - 有效值在 1970 到 2250 之间。 • `month` - 有效值在 1 到 12 之间。 • `day-of-month` - 有效值在 1 到 31 之间。 • `hour` - 有效值在 0 到 23 之间。 • `minute` - 有效值在 0 到 59 之间。 • `second` - 有效值在 0 到 60 之间。它可以是浮点数。 示例： • `mktime(2020, 2, 29)` • `mktime('UTC+3', 2021, 12, 31, 22)` • `mktime(2022, 10, 13, 2, 55, 13.68, 'PST')`

函数	描述
`localtime(unix_time, time_zone)`	从 Unix 时间返回指定时区中的某年、一个月中的某天、一周中的某天、一年中的某天、小时、分钟或秒。 使用此函数存在以下要求： • 时区参数必须是带引号的字符串 (`'UTC'`)。如果未指定，默认时区为 UTC。 • 采用 Unix 纪元时间格式是以秒为单位的时间（采用 Unix 纪元时间格式）。有效范围在 1-31556889864403199 之间。它可以是浮点数。 响应示例：`2007-12-03T10:15:30+01:00[Europe/Paris]` `localtime(unix_time, time_zone)` 不是一个独立的函数。`year()`、`mon()`、`mday`、`wday()`、`yday()`、`hour()` 和 `sec()` 函数采用 `localtime(unix_time, time_zone)` 作为参数。 示例： • `year(localtime('GMT', 1605898608.8113723))` • `now().localtime().year()` • `timestamp().localtime('PST').year()` • `localtime(1605289736, 'Europe/London').year()`
`year(localtime(unix_time, time_zone)`	返回 `localtime(unix_time, time_zone)` 的年份。

函数	描述
mon(localtime(unix_time, time_zone))	返回 localtime(unix_time, time_zone) 的月份。
mday(localtime(unix_time, time_zone))	返回 localtime(unix_time, time_zone) 一个月中的某天。
wday(localtime(unix_time, time_zone))	返回 localtime(unix_time, time_zone) 一周中的某天。
yday(localtime(unix_time, time_zone))	返回 localtime(unix_time, time_zone) 一年中的某天。
hour(localtime(unix_time, time_zone))	返回 localtime(unix_time, time_zone) 的某小时。
minute(localtime(unix_time, time_zone))	返回 localtime(unix_time, time_zone) 的某分钟。
sec(localtime(unix_time, time_zone))	返回 localtime(unix_time, time_zone) 的某秒。

支持的时区格式

您可以通过以下方式指定时区参数：

- 时区偏移量 - 指定 'Z' 表示 UTC 或偏移量（'+2' 或 '-5'）。
- 偏移量 IDs -将时区缩写和偏移量合并。例如，'GMT+2' 和 'UTC-01:00'。时区缩写必须仅包含三个字母。
- 基于区域 IDs -例如，'Etc/GMT+12'和'Pacific/Pago_Pago'。

支持的时区缩写

日期和时间函数支持以下三个字母的时区缩写：

- 美国东部标准时间 (EST) - -05:00
- 美国夏威夷时间 (HST) - -10:00

- 北美山区时间 (MST) - -07:00
- ACT - 澳大利亚/达尔文
- AET - 澳大利亚/悉尼
- AGT-America/Argentina/Buenos _Aires
- ART - 非洲/开罗
- AST - 美洲/安克雷奇
- BET - 美洲/圣保罗
- BST - 亚洲/达卡
- CAT - 非洲/哈拉雷
- CET - 欧洲/巴黎
- CNT - 美洲/圣约翰斯
- CST - 美洲/芝加哥
- CTT - 亚洲/上海
- EAT - 非洲/亚的斯亚贝巴
- IET-America/Indiana/Indianapolis
- IST - 亚洲/加尔各答
- JST - 亚洲/东京
- MIT - 太平洋/阿皮亚
- NET - 亚洲/埃里温
- NST - 太平洋/奥克兰
- PLT - 亚洲/卡拉奇
- PRT - 美洲/波多黎各
- America/Los_Angeles
- SST - 太平洋/瓜达尔卡纳尔岛
- VST - 亚洲/胡志明市

支持的基于区域 IDs

日期和时间函数支持以下基于区域的函数 IDs，按其与 UTC+ 00:00 的关系进行组织：

- Etc/GMT+12 (UTC-12:00)

- 太平洋/帕果帕果 (UTC-11:00)
- 太平洋/萨摩亚 (UTC-11:00)
- 太平洋/纽埃岛 (UTC-11:00)
- 美国/萨摩亚 (UTC-11:00)
- Etc/GMT+11 (UTC-11:00)
- 太平洋/中途岛 (UTC-11:00)
- 太平洋/檀香山 (UTC-10:00)
- 太平洋/拉罗汤加岛 (UTC-10:00)
- 太平洋/塔希提岛 (UTC-10:00)
- 太平洋/约翰斯顿岛 (UTC-10:00)
- 美国/夏威夷 (UTC-10:00)
- Systemv/ HST1 0 (UTC-10:00)
- Etc/GMT+10 (UTC-10:00)
- 太平洋/马克萨斯群岛 (UTC-09:30)
- Etc/GMT+9 (UTC-09:00)
- 太平洋/甘比尔 (UTC-09:00)
- 美国/阿特卡 (UTC-09:00)
- Systemv/ YST9 (UTC-09:00)
- 美洲/埃达克岛 (UTC-09:00)
- 美国/阿留申 (UTC-09:00)
- Etc/GMT+8 (UTC-08:00)
- 美国/阿拉斯加 (UTC-08:00)
- 美洲/朱诺 (UTC-08:00)
- 美洲/梅特拉卡特拉 (UTC-08:00)
- 美洲/亚库塔特 (UTC-08:00)
- 太平洋/皮特凯恩岛 (UTC-08:00)
- 美洲/锡特卡 (UTC-08:00)
- 美洲/安克雷奇 (UTC-08:00)
- Systemv/ PST8 (UTC-08:00)

- 美洲/诺姆 (UTC-08:00)
- Systemv/ YST9 YDT (UTC-08:00)
- 加拿大/育空地区 (UTC-07:00)
- 美国/太平洋-新 (UTC-07:00)
- Etc/GMT+7 (UTC-07:00)
- 美国/亚利桑那州 (UTC-07:00)
- 美国/道森克里克 (UTC-07:00)
- 加拿大/太平洋地区 (UTC-07:00)
- PST8太平洋夏令时 (UTC-07:00)
- Systemv/ MST7 (UTC-07:00)
- 美洲/道森 (UTC-07:00)
- 墨西哥/ BajaNorte (世界标准时间 07:00)
- 美洲/蒂华纳 (UTC-07:00)
- 美洲/克雷斯顿 (UTC-07:00)
- 美洲/埃莫西约 (UTC-07:00)
- 美洲/圣伊莎贝尔 (UTC-07:00)
- 美洲/温哥华 (UTC-07:00)
- 美洲/恩塞纳达 (UTC-07:00)
- 美洲/凤凰城 (UTC-07:00)
- 美洲/怀特霍斯 (UTC-07:00)
- 美洲/纳尔逊堡 (UTC-07:00)
- Systemv/ PST8 PDT (UTC-07:00)
- 美洲/洛杉矶 (UTC-07:00)
- 美国/太平洋地区 (UTC-07:00)
- 美洲/萨尔瓦多 (UTC-06:00)
- 美洲/危地马拉 (UTC-06:00)
- 美洲/伯利兹 (UTC-06:00)
- 美洲/马那瓜 (UTC-06:00)
- 美洲/特古西加尔巴 (UTC-06:00)

- Etc/GMT+6 (UTC-06:00)
- 太平洋/复活节岛 (UTC-06:00)
- 墨西哥/ BajaSur (世界标准时间 06:00)
- 美洲/里贾纳 (UTC-06:00)
- 美洲/丹佛 (UTC-06:00)
- 太平洋/加拉帕戈斯群岛 (UTC-06:00)
- 美洲/耶洛奈夫 (UTC-06:00)
- 美洲/斯威夫特卡伦特 (UTC-06:00)
- 美洲/伊努维克 (UTC-06:00)
- 美洲/马萨特兰 (UTC-06:00)
- 美洲/博伊西 (UTC-06:00)
- 美洲/哥斯达黎加 (UTC-06:00)
- MST7MDT (世界标准时间 06:00)
- Systemv/ CST6 (世界标准时间 06:00)
- 美洲/奇瓦瓦州 (UTC-06:00)
- 美洲/奥吉纳加 (UTC-06:00)
- 智利/ EasterIsland (世界标准时间 06:00)
- 美国/山区 (UTC-06:00)
- 美洲/埃德蒙顿 (UTC-06:00)
- 加拿大/山区 (UTC-06:00)
- 美洲/剑桥湾 (UTC-06:00)
- 纳瓦霍 (UTC-06:00)
- Systemv/ MST7 MDT (UTC-06:00)
- 加拿大/萨斯喀彻温省 (UTC-06:00)
- 美洲/希普罗克 (UTC-06:00)
- 美洲/巴拿马 (UTC-05:00)
- 美洲/芝加哥 (UTC-05:00)
- 美洲/埃鲁内佩 (UTC-05:00)
- Etc/GMT+5 (UTC-05:00)

- 墨西哥/整体 (UTC-05:00)
- 美洲/阿克里港 (UTC-05:00)
- 美洲/瓜亚基尔 (UTC-05:00)
- 美洲/兰京海口 (UTC-05:00)
- 美国/中部 (UTC-05:00)
- 美洲/雷尼河 (UTC-05:00)
- America/Indiana/Knox(世界标准时间 05:00)
- America/North_Dakota/Beulah(世界标准时间 05:00)
- 美洲/蒙特雷 (UTC-05:00)
- 美洲/牙买加 (UTC-05:00)
- 美洲阿蒂科肯 (UTC-05:00)
- 美洲/科勒尔港 (UTC-05:00)
- America/North_Dakota/Center(世界标准时间 05:00)
- 美洲/开曼群岛 (UTC-05:00)
- America/Indiana/Tell_City (世界标准时间 05:00)
- 美洲/墨西哥城 (UTC-05:00)
- 美洲/马塔莫罗斯 (UTC-05:00)
- CST6中部夏令时间 (UTC-05:00)
- 美洲/诺克斯县 (UTC-05:00)
- 美洲/波哥大 (UTC-05:00)
- 美洲/梅诺米尼 (UTC-05:00)
- 美洲/雷索卢特 (UTC-05:00)
- Systemv/ EST5 (UTC-05:00)
- 加拿大/中部 (UTC-05:00)
- 巴西/阿克里州 (UTC-05:00)
- 美洲/坎昆 (UTC-05:00)
- 美洲/利马 (UTC-05:00)
- 美洲/Bahia_Banderas (UTC-05:00)
- 美国/印第安纳-斯塔克 (UTC-05:00)

- 美洲/里奥布朗库 (UTC-05:00)
- Systemv/ CST6 CDT (UTC-05:00)
- 牙买加 (UTC-05:00)
- 美洲/梅里达 (UTC-05:00)
- America/North_Dakota/New_Salem (世界标准时间 05:00)
- 美洲/温尼伯 (UTC-05:00)
- 美洲/库亚巴 (UTC-04:00)
- 美洲/马里戈特 (UTC-04:00)
- America/Indiana/Petersburg(世界标准时间 04:00)
- 智利/大陆 (UTC-04:00)
- 美洲/大特克岛 (UTC-04:00)
- 古巴 (UTC-04:00)
- Etc/GMT+4 (UTC-04:00)
- 美洲/马瑙斯 (UTC-04:00)
- 美洲/韦恩堡 (UTC-04:00)
- 美洲/圣托马斯 (UTC-04:00)
- 美洲/安圭拉 (UTC-04:00)
- 美洲/哈瓦那 (UTC-04:00)
- 美国/密歇根州 (UTC-04:00)
- 美洲/巴巴多斯 (UTC-04:00)
- 美洲/路易斯维尔 (UTC-04:00)
- 美洲/库拉索 (UTC-04:00)
- 美洲/圭亚那 (UTC-04:00)
- 美洲/马提尼克岛 (UTC-04:00)
- 美洲/波多黎各 (UTC-04:00)
- 美洲/西班牙港 (UTC-04:00)
- Systemv/ AST4 (UTC-04:00)
- America/Indiana/Vevay(世界标准时间 04:00)
- America/Indiana/Vincennes(世界标准时间 04:00)

- 美洲/克拉伦代克 (UTC-04:00)
- 美洲/安提瓜 (UTC-04:00)
- 美洲/印第安纳波利斯 (UTC-04:00)
- 美洲/伊卡卢伊特 (UTC-04:00)
- 美洲/圣文森特 (UTC-04:00)
- America/Kentucky/Louisville(世界标准时间 04:00)
- 美洲/多米尼加 (UTC-04:00)
- 美洲/亚松森 (UTC-04:00)
- EST5美国东部时间（世界标准时间 04:00）
- 美洲/拿骚 (UTC-04:00)
- America/Kentucky/Monticello(世界标准时间 04:00)
- 巴西/西部 (UTC-04:00)
- 美洲/阿鲁巴 (UTC-04:00)
- America/Indiana/Indianapolis(世界标准时间 04:00)
- 美洲/圣地亚哥 (UTC-04:00)
- 美洲/拉巴斯 (UTC-04:00)
- 美洲/桑德贝 (UTC-04:00)
- America/Indiana/Marengo(世界标准时间 04:00)
- 美洲/巴朗克萨伯隆 (UTC-04:00)
- 美洲/圣多明哥 (UTC-04:00)
- 美国/东部 (UTC-04:00)
- 加拿大/东部 (UTC-04:00)
- America/ Port-au-Prince (世界标准时间 04:00)
- 美洲/圣巴泰勒米 (UTC-04:00)
- 美洲/尼皮贡 (UTC-04:00)
- 美国/印第安纳州东部 (UTC-04:00)
- 美洲/圣露西亚 (UTC-04:00)
- 美洲/蒙特塞拉特 (UTC-04:00)
- 美洲/Lower_Princes (UTC-04:00)

- 美洲/底特律 (UTC-04:00)
- 美洲/托尔托拉 (UTC-04:00)
- 美洲/波多韦 (UTC-04:00)
- 美洲/大坎普 (UTC-04:00)
- 美洲/弗吉尼亚 (UTC-04:00)
- 美洲/潘纳通 (UTC-04:00)
- 美洲/蒙特利尔 (UTC-04:00)
- America/Indiana/Winamac(世界标准时间 04:00)
- 美洲/博阿维斯塔 (UTC-04:00)
- 美洲/格林纳达 (UTC-04:00)
- 美洲/纽约 (UTC-04:00)
- 美洲/圣基茨 (UTC-04:00)
- 美洲/加拉加斯 (UTC-04:00)
- 美洲/瓜德罗普岛 (UTC-04:00)
- 美洲/多伦多 (UTC-04:00)
- Systemv/ EST5 美国东部时间 (UTC-04:00)
- America/Argentina/Catamarca(世界标准时间 03:00)
- 加拿大/大西洋地区 (UTC-03:00)
- America/Argentina/Cordoba(世界标准时间 03:00)
- 美洲/阿拉瓜伊纳 (UTC-03:00)
- America/Argentina/Salta(世界标准时间 03:00)
- Etc/GMT+3 (UTC-03:00)
- 美洲/蒙得维的亚 (UTC-03:00)
- 巴西/东部 (UTC-03:00)
- America/Argentina/Mendoza(世界标准时间 03:00)
- America/Argentina/Rio_Gallegos (世界标准时间 03:00)
- 美洲/卡塔马卡 (UTC-03:00)
- 美洲/科尔多瓦 (UTC-03:00)
- 美洲/圣保罗 (UTC-03:00)

- America/Argentina/Jujuy(世界标准时间 03:00)
- 美洲/卡宴 (UTC-03:00)
- 美洲/累西腓市 (UTC-03:00)
- 美洲/布宜诺斯艾利斯 (UTC-03:00)
- 美洲/帕拉马里博 (UTC-03:00)
- 美洲/蒙克顿 (UTC-03:00)
- 美洲/门多萨 (UTC-03:00)
- 美洲/圣塔伦 (UTC-03:00)
- 大西洋/百慕大 (UTC-03:00)
- 美洲/马塞约 (UTC-03:00)
- 大西洋/斯坦利 (UTC-03:00)
- 美洲/哈利法克斯 (UTC-03:00)
- 南极洲/罗瑟拉 (UTC-03:00)
- America/Argentina/San_Luis (世界标准时间 03:00)
- America/Argentina/Ushuaia(世界标准时间 03:00)
- 南极洲/帕尔默 (UTC-03:00)
- 美洲/蓬塔阿雷纳斯 (UTC-03:00)
- 美洲/格莱斯贝 (UTC-03:00)
- 美洲/福塔雷萨 (UTC-03:00)
- 美洲/图勒岛 (UTC-03:00)
- America/Argentina/La_Rioja (世界标准时间 03:00)
- 美洲/贝伦 (UTC-03:00)
- 美洲/胡胡伊 (UTC-03:00)
- 美洲/巴伊亚 (UTC-03:00)
- 美洲/古斯湾 (UTC-03:00)
- America/Argentina/San_Juan (世界标准时间 03:00)
- America/Argentina/ComodRivadavia(世界标准时间 03:00)
- America/Argentina/Tucuman(世界标准时间 03:00)
- 美洲/罗萨里奥 (UTC-03:00)

- Systemv/ AST4 ADT (UTC-03:00)
- America/Argentina/Buenos_Aires (世界标准时间 03:00)
- 美洲/圣约翰斯 (UTC-02:30)
- 加拿大/纽芬兰 (UTC-02:30)
- 美洲/密克隆群岛 (UTC-02:00)
- Etc/GMT+2 (UTC-02:00)
- 美洲/戈特霍布 (UTC-02:00)
- 美洲/诺罗尼亚 (UTC-02:00)
- 巴西/ DeNoronha (世界标准时间 02:00)
- 大西洋/南乔治亚岛 (UTC-02:00)
- Etc/GMT+1 (UTC-01:00)
- 大西洋/佛得角 (UTC-01:00)
- 太平洋/基里巴斯圣诞岛 (UTC+14:00)
- Etc/GMT-14 (UTC+14:00)
- 太平洋/法考福环礁 (UTC+13:00)
- 太平洋/恩德伯里 (UTC+13:00)
- 太平洋/阿皮亚 (UTC+13:00)
- 太平洋/汤加塔布 (UTC+13:00)
- Etc/GMT-13 (UTC+13:00)
- NZ-CHAT (UTC+12:45)
- 太平洋/查塔姆群岛 (UTC+12:45)
- 太平洋/夸贾林 (UTC+12:00)
- 南极洲/（世界标准时间+ 12 McMurdo :00）
- 太平洋/瓦利斯 (UTC+12:00)
- 太平洋/斐济 (UTC+12:00)
- 太平洋/富纳富提 (UTC+12:00)
- 太平洋/瑙鲁 (UTC+12:00)
- 夸贾林岛 (UTC+ 12:00)
- 新西兰 (UTC+12:00)

- 太平洋/威克岛 (UTC+12:00)
- 南极洲/南极 (UTC+ 12:00)
- 太平洋/塔拉瓦 (UTC+12:00)
- 太平洋/奥克兰 (UTC+12:00)
- 亚洲/堪察加半岛 (UTC+12:00)
- Etc/GMT-12 (UTC+12:00)
- 亚洲/阿纳德尔 (UTC+12:00)
- 太平洋/马朱罗 (UTC+12:00)
- 太平洋/波纳佩岛 (UTC+11:00)
- 太平洋/布干维尔 (UTC+11:00)
- 南极洲/麦格理 (UTC+11:00)
- 太平洋/波纳佩岛 (UTC+11:00)
- 太平洋/埃法特岛 (UTC+11:00)
- 太平洋/诺福克 (UTC+11:00)
- 亚洲/马加丹 (UTC+11:00)
- 太平洋/科斯雷 (UTC+11:00)
- 亚洲/萨哈林 (UTC+11:00)
- 太平洋/努美阿 (UTC+11:00)
- Etc/GMT-11 (UTC+11:00)
- 亚洲/Srednekolymsk (UTC+11:00)
- 太平洋/瓜达尔卡纳尔岛 (UTC+11:00)
- 澳大利亚/豪勋爵 (UTC+ 10:30)
- 澳大利亚/LHI (UTC+10:30)
- 澳大利亚/霍巴特 (UTC+10:00)
- 太平洋/雅浦岛 (UTC+10:00)
- 澳大利亚/塔斯马尼亚州 (UTC+10:00)
- 太平洋/莫尔兹比港 (UTC+10:00)
- 澳大利亚/澳大利亚首都直辖区 (UTC+10:00)
- 澳大利亚/维多利亚州 (UTC+10:00)

- 太平洋/楚克岛 (UTC+10:00)
- 澳大利亚/昆士兰州 (UTC+10:00)
- 澳大利亚/堪培拉 (UTC+10:00)
- 澳大利亚/居里 (UTC+10:00)
- 太平洋/关岛 (UTC+10:00)
- 太平洋/特鲁克岛 (UTC+10:00)
- 澳大利亚/新南威尔士州 (UTC+10:00)
- 亚洲/符拉迪沃斯托克 (UTC+10:00)
- 太平洋/塞班岛 (UTC+10:00)
- 南极洲/杜DUrville 蒙特（世界标准时间+ 10:00）
- 澳大利亚/悉尼（世界标准时间+ 10:00）
- 澳大利亚/布里斯班 (UTC+10:00)
- Etc/GMT-10 (UTC+10:00)
- 亚洲/乌斯季挪拉 (UTC+10:00)
- 澳大利亚/墨尔本 (UTC+10:00)
- 澳大利亚/利达曼 (UTC+10:00)
- 澳大利亚/北部 (UTC+09:30)
- 澳大利亚/Yancowinna (UTC+ 09:30)
- 澳大利亚/阿德莱德 (UTC+09:30)
- 澳大利亚/布罗肯希尔 (UTC+ 09:30)
- 澳大利亚/南部 (UTC+09:30)
- 澳大利亚/达尔文 (UTC+09:30)
- Etc/GMT-9 (UTC+09:00)
- 太平洋/帕劳 (UTC+09:00)
- 亚洲/赤塔 (UTC+09:00)
- 亚洲/帝力 (UTC+09:00)
- 亚洲/查亚普拉 (UTC+09:00)
- 亚洲/雅库茨克 (UTC+09:00)
- 亚洲/平壤 (UTC+09:00)

- 韩国 (UTC+09:00)
- 亚洲/首尔 (UTC+09:00)
- 亚洲/汉德加 (UTC+ 09:00)
- 日本 (UTC+09:00)
- 亚洲/东京 (UTC+09:00)
- 澳大利亚/尤克拉 (UTC+ 08:45)
- 亚洲/古晋 (UTC+08:00)
- 亚洲/重庆 (UTC+08:00)
- Etc/GMT-8 (UTC+08:00)
- 澳大利亚/珀斯 (UTC+08:00)
- 亚洲/澳门 (UTC+08:00)
- 亚洲/澳门 (UTC+08:00)
- 亚洲/乔巴山 (UTC+08:00)
- 亚洲/上海 (UTC+08:00)
- 南极洲/凯西 (UTC+ 08:00)
- 亚洲/乌兰巴托 (UTC+ 08:00)
- 亚洲/重庆 (UTC+08:00)
- 亚洲/乌兰巴托 (UTC+08:00)
- 亚洲/台北 (UTC+08:00)
- 亚洲/马尼拉 (UTC+08:00)
- 中国 (UTC+08:00)
- 亚洲/乌戎潘当 (UTC+ 08:00)
- 亚洲/哈尔滨 (UTC+08:00)
- 新加坡 (UTC+08:00)
- 亚洲/文莱 (UTC+08:00)
- 澳大利亚/西部 (UTC+08:00)
- 亚洲/香港 (UTC+ 08:00)
- 亚洲/望加锡 (UTC+08:00)
- 中国香港 (UTC+08:00)

- 亚洲/吉隆坡 (UTC+ 08:00)
- 亚洲/伊尔库茨克 (UTC+08:00)
- 亚洲/新加坡 (UTC+08:00)
- 亚洲/坤甸 (UTC+ 07:00)
- Etc/GMT-7 (UTC+07:00)
- 亚洲/金边 (UTC+07:00)
- 亚洲/新西伯利亚 (UTC+07:00)
- 南极洲/戴维斯 (UTC+07:00)
- 亚洲/托木斯克 (UTC+ 07:00)
- 亚洲/雅加达 (UTC+07:00)
- 亚洲/巴尔瑙尔 (UTC+07:00)
- 印度洋/圣诞岛 (UTC+07:00)
- 亚洲/胡志明市 (UTC+07:00)
- 亚洲/科布多 (UTC+ 07:00)
- 亚洲/曼谷 (UTC+07:00)
- 亚洲/万象 (UTC+07:00)
- 亚洲/新库兹涅茨克 (UTC+07:00)
- 亚洲/克拉斯诺亚尔斯克 (UTC+ 07:00)
- 亚洲/西贡 (UTC+07:00)
- 亚洲/仰光 (UTC+06:30)
- 亚洲/仰光 (UTC+06:30)
- 印度洋/科科斯群岛 (UTC+06:30)
- 亚洲/喀什 (UTC+06:00)
- Etc/GMT-6 (UTC+06:00)
- 亚洲/阿拉木图 (UTC+06:00)
- 亚洲/达卡 (UTC+06:00)
- 亚洲/鄂木斯克 (UTC+06:00)
- 亚洲/达卡 (UTC+06:00)
- 印度洋/查戈斯群岛 (UTC+06:00)

- 亚洲/Qyzylorda (UTC+06:00)
- 亚洲/比什凯克 (UTC+06:00)
- 南极洲/东方站（UTC+ 06:00）
- 亚洲/乌鲁木齐 (UTC+06:00)
- 亚洲/廷布 (UTC+06:00)
- 亚洲/廷布 (UTC+06:00)
- 亚洲/加德满都 (UTC+05:45)
- 亚洲/加德满都 (UTC+05:45)
- 亚洲/加尔各答 (UTC+05:30)
- 亚洲/科伦坡 (UTC+05:30)
- 亚洲/加尔各答 (UTC+05:30)
- 亚洲/阿克套 (UTC+ 05:00)
- Etc/GMT-5 (UTC+05:00)
- 亚洲/撒马尔罕 (UTC+05:00)
- 亚洲/卡拉奇 (UTC+05:00)
- 亚洲/叶卡捷琳堡 (UTC+05:00)
- 亚洲/杜尚别 (UTC+05:00)
- 印度洋/马尔代夫 (UTC+05:00)
- 亚洲/Oral (UTC+ 05:00)
- 亚洲/塔什干 (UTC+05:00)
- 南极洲/莫森 (UTC+05:00)
- 亚洲/阿克托比 (UTC+ 05:00)
- 亚洲/阿什哈巴德 (UTC+05:00)
- 亚洲/阿什哈巴德 (UTC+ 05:00)
- 亚洲/阿特劳 (UTC+05:00)
- 印度洋/凯尔盖朗岛 (UTC+ 05:00)
- 伊朗 (UTC+04:30)
- 亚洲/德黑兰 (UTC+04:30)
- 亚洲/喀布尔 (UTC+04:30)

- 亚洲/埃里温 (UTC+04:00)
- Etc/GMT-4 (UTC+04:00)
- Etc/GMT-4 (UTC+04:00)
- 亚洲/迪拜 (UTC+04:00)
- 印度洋/留尼汪岛 (UTC+04:00)
- 欧洲/萨拉托夫 (UTC+ 04:00)
- 欧洲/萨马拉 (UTC+04:00)
- 印度洋/马埃岛 (UTC+04:00)
- 亚洲/巴库 (UTC+04:00)
- 亚洲/马斯喀特 (UTC+04:00)
- 欧洲/伏尔加格勒 (UTC+ 04:00)
- 欧洲/阿斯特拉罕 (UTC+04:00)
- 亚洲/第比利斯 (UTC+04:00)
- 欧洲/乌里扬诺夫斯克 (UTC+ 04:00)
- 亚洲/亚丁 (UTC+03:00)
- 非洲/内罗毕 (UTC+03:00)
- 欧洲/伊斯坦布尔 (UTC+03:00)
- Etc/GMT-3 (UTC+03:00)
- 欧洲/扎波罗热 (UTC+03:00)
- 以色列 (UTC+03:00)
- 印度洋/科摩罗 (UTC+03:00)
- 南极洲/昭和 (UTC+03:00)
- 非洲/摩加迪沙 (UTC+03:00)
- 欧洲/布加勒斯特 (UTC+03:00)
- 非洲/阿斯马拉 (UTC+03:00)
- 欧洲/玛丽港 (UTC+ 03:00)
- 亚洲/伊斯坦布尔 (UTC+03:00)
- 欧洲/蒂拉斯波尔 (UTC+ 03:00)
- 欧洲/莫斯科 (UTC+03:00)

- 欧洲/基希讷乌 (UTC+03:00)
- 欧洲/赫尔辛基 (UTC+03:00)
- 亚洲/贝鲁特 (UTC+03:00)
- 亚洲/特拉维夫 (UTC+ 03:00)
- 非洲/吉布提 (UTC+03:00)
- 欧洲/辛菲罗波尔 (UTC+03:00)
- 欧洲/索非亚 (UTC+03:00)
- 亚洲/加沙 (UTC+03:00)
- 非洲/阿斯马拉 (UTC+03:00)
- 欧洲/里加 (UTC+03:00)
- 亚洲/巴格达 (UTC+03:00)
- 亚洲/大马士革 (UTC+03:00)
- 非洲/达累斯萨拉姆 (UTC+ 03:00)
- 非洲/亚的斯亚贝巴 (UTC+ 03:00)
- 欧洲/乌日哥罗德 (UTC+ 03:00)
- 亚洲/耶路撒冷 (UTC+03:00)
- 亚洲/利雅得 (UTC+03:00)
- 亚洲/科威特 (UTC+03:00)
- 欧洲/基洛夫 (UTC+03:00)
- 非洲/坎帕拉 (UTC+03:00)
- 欧洲/明斯克 (UTC+03:00)
- 亚洲/卡塔尔 (UTC+03:00)
- 欧洲/基辅 (UTC+03:00)
- 亚洲/巴林 (UTC+03:00)
- 欧洲/维尔纽斯 (UTC+03:00)
- 印度洋/塔那那利佛 (UTC+03:00)
- 印度洋/马约特岛 (UTC+03:00)
- 欧洲/塔林 (UTC+03:00)
- 土耳其 (UTC+03:00)

- 非洲/朱巴 (UTC+03:00)
- 亚洲/尼科西亚 (UTC+03:00)
- 亚洲/法马古斯塔 (UTC+03:00)
- W-SU (UTC+03:00)
- EET (UTC+03:00)
- 亚洲/希伯伦 (UTC+03:00)
- 亚洲/安曼 (UTC+03:00)
- 欧洲/尼科西亚 (UTC+03:00)
- 欧洲/雅典 (UTC+03:00)
- 非洲/开罗 (UTC+02:00)
- 非洲/姆巴巴内 (UTC+02:00)
- 欧洲/布鲁塞尔 (UTC+02:00)
- 欧洲/华沙 (UTC+02:00)
- CET (UTC+02:00)
- 欧洲/卢森堡 (UTC+02:00)
- Etc/GMT-2 (UTC+02:00)
- 利比亚 (UTC+02:00)
- 非洲/基加利 (UTC+02:00)
- 非洲/的黎波里 (UTC+ 02:00)
- 欧洲/加里宁格勒 (UTC+ 02:00)
- 非洲/温得和克 (UTC+02:00)
- 欧洲/马耳他 (UTC+02:00)
- 欧洲/布辛根 (UTC+02:00)
-
- 欧洲/斯科普里 (UTC+ 02:00)
- 欧洲/萨拉热窝 (UTC+ 02:00)
- 欧洲/罗马 (UTC+02:00)
- 欧洲/苏黎世 (UTC+02:00)
- 欧洲/直布罗陀 (UTC+02:00)

- 非洲/卢本巴希 (UTC+02:00)
- 欧洲/瓦杜兹 (UTC+02:00)
- 欧洲/卢布尔雅那 (UTC+ 02:00)
- 欧洲/柏林 (UTC+02:00)
- 欧洲/斯德哥尔摩 (UTC+02:00)
- 欧洲/布达佩斯 (UTC+02:00)
- 欧洲/萨格勒布 (UTC+02:00)
- 欧洲地区（巴黎）(UTC+02:00)
- 非洲/休达 (UTC+02:00)
- 欧洲/布拉格 (UTC+02:00)
- 南极洲/巨魔 (UTC+ 02:00)
- 非洲/哈博罗内 (UTC+02:00)
- 欧洲/哥本哈根 (UTC+02:00)
- 欧洲/维也纳 (UTC+02:00)
- 欧洲/地拉那 (UTC+02:00)
- MET (UTC+02:00)
- 欧洲/阿姆斯特丹 (UTC+02:00)
- 非洲/马普托 (UTC+02:00)
- 欧洲/圣马力诺 (UTC+ 02:00)
- 波兰 (UTC+02:00)
- 欧洲/安道尔 (UTC+02:00)
- 欧洲/奥斯陆 (UTC+02:00)
- 欧洲/波德戈里察 (UTC+02:00)
- 非洲/布琼布拉 (UTC+02:00)
- 大西洋/扬马延岛 (UTC+02:00)
- 非洲/马塞卢 (UTC+02:00)
- 欧洲/马德里 (UTC+02:00)
- 非洲/布兰太尔 (UTC+02:00)
- 非洲/卢萨卡 (UTC+02:00)

- 非洲/哈拉雷 (UTC+02:00)
- 非洲/喀土穆 (UTC+02:00)
- 非洲/约翰内斯堡 (UTC+02:00)
- 欧洲/贝尔格莱德 (UTC+ 02:00)
- 欧洲/布拉迪斯拉发 (UTC+02:00)
- 北极洲/朗伊尔城 (UTC+02:00)
- 埃及 (UTC+02:00)
- 欧洲/梵蒂冈 (UTC+02:00))
- 欧洲/摩纳哥 (UTC+02:00)
- 欧洲/伦敦 (UTC+01:00)
- Etc/GMT-1 (UTC+01:00)
- 欧洲/泽西岛 (UTC+01:00)
- 欧洲/根西岛 (UTC+01:00)
- 欧洲/马恩岛 (UTC+ 01:00)
- 非洲/突尼斯 (UTC+01:00)
- 非洲/马拉博 (UTC+01:00)
- GB-Eire (UTC+01:00)
- 非洲/拉各斯 (UTC+01:00)
- 非洲/阿尔及尔 (UTC+01:00)
- GB (UTC+01:00)
- 葡萄牙 (UTC+01:00)
- 非洲/圣多美 (UTC+ 01:00)
- 非洲/恩贾梅纳 (UTC+01:00)
- 大西洋/法罗群岛 (UTC+ 01:00)
- 爱尔兰 (UTC+ 01:00)
- 大西洋/法罗群岛 (UTC+01:00)
- 欧洲/都柏林 (UTC+01:00)
- 非洲/利伯维尔 (UTC+01:00)
- 非洲/埃丰阿莱耶 (UTC+ 01:00)

- 非洲/埃丰阿莱耶 (UTC+ 01:00)
- 非洲/杜阿拉 (UTC+01:00)
- 非洲/布拉柴维尔 (UTC+01:00)
- 非洲/新港 (UTC+ 01:00)
- 大西洋/马德拉 (UTC+01:00)
- 欧洲/里斯本 (UTC+01:00)
- 大西洋/加那利 (UTC+01:00)
- 非洲/卡萨布兰卡 (UTC+01:00)
- 欧洲/贝尔法斯特 (UTC+01:00)
- 非洲/罗安达 (UTC+ 01:00)
- 非洲/金沙萨 (UTC+01:00)
- 非洲/班吉 (UTC+01:00)
- WET (UTC+01:00)
- 非洲/尼亚美（UTC+ 01:00）
- GMT (UTC+00:00)
- Etc/GMT-0 (UTC+00:00)
- 大西洋/圣海伦娜 (UTC+ 00:00)
- Etc/GMT+0 (UTC+00:00)
- 非洲/班珠尔 (UTC+00:00)
- Etc/GMT (UTC+00:00)
- 非洲/弗里敦 (UTC+00:00)
- 非洲/巴马科 (UTC+00:00)
- 非洲/科纳克里 (UTC+00:00)
- 通用 (UTC+00:00)
- 非洲/努瓦克肖特 (UTC+00:00)
- UTC (UTC+00:00)
- Etc/通用 (UTC+00:00)
- 大西洋/亚速尔群岛 (UTC+00:00)
- 非洲/阿比让 (UTC+00:00)

- 非洲/阿克拉 (UTC+00:00)
- Etc/UCT (UTC+00:00)
- GMT0 (UTC+00:00)
- 祖鲁 (UTC+ 00:00) 祖鲁 (UTC+ 00:00)
- 非洲/瓦加杜古 (UTC+00:00)
- 大西洋/雷克雅未克 (UTC+00:00)
- Etc/祖鲁 (UTC+00:00)
- 冰岛 (UTC+00:00)
- 非洲/洛美 (UTC+00:00)
- 格林威治 (UTC+00:00)
- Etc/GMT0 (UTC+00:00)
- 美洲/丹马港 (UTC+ 00:00)
- 非洲/达喀尔 (UTC+00:00)
- 非洲/比绍 (UTC+00:00)
- Etc/格林威治 (UTC+00:00)
- 非洲/廷巴克图 (UTC+00:00)
- UCT (UTC+00:00)
- 非洲/蒙罗维亚 (UTC+00:00)
- Etc/UTC (UTC+00:00)

公式表达式教程

您可以按照这些教程在 Amazon IoT SiteWise中使用公式表达式。

主题

- 在公式中使用字符串
- 筛选数据点
- 计算与条件匹配的数据点
- 公式中的最新数据
- 公式中的数据质量

- 未定义值、无限值和溢出值

在公式中使用字符串

您可以对公式表达式中的字符串进行操作。您也可以从引用属性和测量属性的变量中输入字符串。

> **Important**
>
> 公式表达式只能输出双精度值或字符串值。嵌套表达式可以输出其他数据类型，例如字符串，但整个公式的计算结果必须为数字或字符串。您可以使用 jp 函数将字符串转换为数字。布尔值必须为 1（真）或 0（假）。有关更多信息，请参阅 未定义值、无限值和溢出值。

Amazon IoT SiteWise 提供了可用于对字符串进行操作的以下公式表达式功能：

- 字符串文字
- 索引运算符 (`s[index]`)
- 切片运算符 (`s[start:end:step]`)
- 比较函数，您可以使用它按字典顺序比较字符串
- 字符串函数，包括可以解析序列化的 JSON 对象并将字符串转换为数字的 `jp` 函数

筛选数据点

您可以使用 if 函数筛选出不符合条件的数据点。`if` 函数计算条件并返回 `true` 和 `false` 结果的不同值。您可以使用 无 常量`if` 函数的一个案例的输出，以丢弃该案例的数据点。

筛选出与条件匹配的数据点

- 创建一个转换，该转换使用 `if` 函数定义一个条件，该条件检查条件是否满足，并以 `result_if_true` 或 `result_if_false` 值的形式返回 `none`。

Example 示例：过滤掉水未沸腾的数据点

考虑您在其中有一个测量值的场景 `temp_c`，它提供水在机器中的温度（以摄氏度为单位）。您可以定义以下转换来过滤掉水未沸腾的数据点：

- 转换：`boiling_temps = if(gte(temp_c, 100), temp_c, none)` – 如果温度大于或等于 100 摄氏度，则返回温度，否则不返回任何数据点。

计算与条件匹配的数据点

您可以使用比较函数和 求和 来计算条件为真的数据点的数量。

计算与条件匹配的数据点

1. 创建一个使用比较函数在另一个属性上定义筛选条件的转换。
2. 创建一个汇总满足该条件的数据点的指标。

Example 示例：计算水沸腾的数据点数量

考虑您在其中有一个测量值的场景 `temp_c`，它提供水在机器中的温度（以摄氏度为单位）。您可以定义以下转换和指标属性来计算水沸腾的数据点数量：

- 转换：`is_boiling = gte(temp_c, 100)` – 如果温度大于或等于 100 摄氏度，则计算结果为返回 `1`，否则计算结果为返回 `0`。
- 指标：`boiling_count = sum(is_boiling)` – 返回水沸腾的数据点数量。

公式中的最新数据

Amazon IoT SiteWise 支持延迟摄取最长 7 天的数据。当 Amazon IoT SiteWise 收到延迟数据时，它会重新计算过去窗口中输入延迟数据的任何指标的现有值。这些重新计算会产生数据处理费用。

> **Note**
>
> 在 Amazon IoT SiteWise 计算输入后期数据的属性时，它使用每个属性的当前公式表达式。

在 Amazon IoT SiteWise 重新计算某个指标的过去窗口后，它将替换该窗口的先前值。如果您为该指标启用了通知，则 Amazon IoT SiteWise 还会发出属性值通知。这意味着，您可以对您以前收到了通知的相同属性和时间戳收到新属性值更新通知。如果应用程序或数据湖使用属性值通知，则必须使用新值更新上一个值，以便其数据准确无误。

公式中的数据质量

在中 Amazon IoT SiteWise，每个数据点都有一个质量代码，可以是以下代码之一：

- `GOOD` – 数据不受任何问题的影响。

- BAD – 数据受传感器故障等问题的影响。
- UNCERTAIN – 数据受传感器不准确等问题的影响。

Amazon IoT SiteWise 在计算变换和指标时仅消耗GOOD质量数据。 Amazon IoT SiteWise 仅输出成功计算所需的GOOD质量数据。如果计算不成功，则 Amazon IoT SiteWise 不会为该计算输出数据点。如果计算生成未定义值、无限值或溢出值，则会发生这种情况。

有关如何查询数据和按数据质量筛选的更多信息，请参阅从中查询数据 Amazon IoT SiteWise。

未定义值、无限值和溢出值

某些公式表达式（例如`x / 0``sqrt(-1)`、或`log(0)`）会计算在实数系统中未定义的值、无限值或超出所 Amazon IoT SiteWise支持的范围。当资产属性的表达式计算未定义、无限或溢出值时， Amazon IoT SiteWise 不会为该计算输出数据点。

Amazon IoT SiteWise 如果将非数值作为公式表达式的结果进行计算，也不会输出数据点。这意味着，如果您定义了一个计算字符串、数组或 无 常量的公式，则 Amazon IoT SiteWise 不会为该计算输出数据点。

Example 示例

以下每个公式表达式生成的值都 Amazon IoT SiteWise 不能表示为数字。 Amazon IoT SiteWise 在计算这些公式表达式时不输出数据点。

- `x / 0` 未定义。
- `log(0)` 未定义。
- `sqrt(-1)` 在实数系统中未定义。
- `"hello" + " world"` 是一个字符串。
- `jp('{"values":[3,6,7]}', '$.values')` 是一个数组。
- 当 `temp` 小于 `300` 时，`if(gte(temp, 300), temp, none)` 等于 `none`。

创建自定义复合模型（组件）

自定义复合模型或组件（如果您使用的是控制台）为您的资产模型和组件模型提供了另一种组织方式。您可以使用它们，通过对属性进行分组或引用其他模型来构建模型。有关使用自定义复合模型的更多信息，请参阅自定义复合模型（组件）。

您可以在现有资产模型或组件模型中创建自定义复合模型。自定义复合模型分为两类。要对模型中的相关属性进行分组，可以创建内联自定义复合模型。要在资产模型或组件模型中引用组件模型，可以创建component-model-based自定义复合模型。

以下各节介绍如何使用 Amazon IoT SiteWise API 创建自定义复合模型。

主题

- 创建内联组件（控制台）
- 创建内联自定义复合模型（Amazon CLI）
- 创建组 component-model-based件（控制台）
- 创建 component-model-based自定义复合模型 (Amazon CLI)

创建内联组件（控制台）

您可以使用 Amazon IoT SiteWise 控制台创建定义其自身属性的内联组件。

> **Note**
>
> 由于这是一个内联组件，因此，这些属性仅适用于当前的资产模型，不会共享到其他任何位置。
> 如果您需要生成一个可重复使用的模型（例如，在多个资产模型之间共享，或者在一个资产模型中包含多个实例），则应改为基于组件模型创建一个组件。有关详细信息，请参阅下一节。

要创建组件（控制台），请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择要为其添加组件的资产模型。
4. 在属性选项卡上，选择组件。
5. 选择创建组件。
6. 在创建组件页面上，执行以下操作：
   a. 为组件输入一个名称，例如 **`ServoMotor`** 或 **`ServoMotor Model`**。此名称对于您账户在此区域的所有组件都必须是唯一的。

b. （可选）为模型添加 属性定义。属性表示很少更改的信息。有关更多信息，请参阅 定义静态数据（属性）。

c. （可选）为模型添加 测量值定义。测量值代表来自您的设备的数据流。有关更多信息，请参阅 定义来自设备的数据流（测量值）。

d. （可选）为模型添加 转换定义。转换是将数据从一种表单映射到另一种表单的公式。有关更多信息，请参阅 转换数据（转换）。

e. （可选）为模型添加 指标定义。指标是按时间间隔汇总数据的公式。指标可以从关联资产的指标中输入数据，这样您可以计算统计信息，从而提供对整个操作或部分操作的见解。有关更多信息，请参阅 聚合来自属性和其他资产（指标）的数据。

f. 选择创建组件。

创建内联自定义复合模型（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 来创建定义其自身属性的内联自定义复合模型。

使用CreateAssetModelCompositeModel操作创建具有属性的内联模型。此操作需要具有以下结构的负载。

> **Note**
>
> 由于这是一个内联复合组件，因此，这些属性仅适用于当前的资产模型，不会共享到其他任何位置。之所以称其为“内联”，是因为它没有为 `composedAssetModelId` 字段提供值。
> 如果您需要生成可重复使用的模型（例如，在多个资产模型之间共享，或者在一个资产模型中包含多个实例），则应改为创建component-model-based复合模型。有关详细信息，请参阅下一节。

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

创建组 component-model-based件（控制台）

您可以使用 Amazon IoT SiteWise 控制台基于组件模型创建组件。

创建组 component-model-based件（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择要为其添加组件的资产模型。
4. 在属性选项卡上，选择组件。
5. 选择创建组件。
6. 在创建组件页面上，执行以下操作：
   a. 选择要作为组件基础的组件模型。
   b. 为组件输入一个名称，例如 **ServoMotor** 或 **ServoMotor Model**。此名称对于您账户在此区域的所有组件都必须是唯一的。
   c. 选择创建组件。

创建 component-model-based自定义复合模型 (Amazon CLI)

您可以使用在 Amazon CLI 资产模型中创建 component-model-based自定义复合模型。 component-model-based自定义复合模型是对已在其他地方定义的组件模型的引用。

使用CreateAssetModelCompositeModel操作创建 component-model-based自定义复合模型。此操作需要具有以下结构的负载。

> **Note**
>
> 在此示例中，`composedAssetModelId` 的值是现有组件模型的资产模型 ID 或外部 ID。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。有关如何创建组件模型的示例，请参阅创建组件模型（Amazon CLI）。

```
{
    "assetModelCompositeModelName": "CNCLathe_ServoMotorA",
    "assetModelCompositeModelType": "CUSTOM",
    "composedAssetModelId": component model ID
]
```

由于它只是一个参考，因此 component-model-based自定义复合模型除了名称之外没有自己的属性。

如果要将同一组件的多个实例添加到资产模型中（例如，具有多个伺服电动机的数控机床），则可以添加多个 component-model-based自定义复合模型，每个模型都有自己的名称，但所有模型的引用都相同`composedAssetModelId`。

您可以将组件嵌套在其他组件中。为此，您可以向其中一个组件模型添加 component-model-based复合模型，如本示例所示。

在中为资产模型创建资产 Amazon IoT SiteWise

您可以从资产模型中创建资产。您必须具有资产模型，然后才能创建资产。如果您尚未创建资产模型，请参阅在中创建资产模型 Amazon IoT SiteWise。

> **Note**
>
> 您只能从 `ACTIVE` 模型创建资产。如果模型的状态不是 `ACTIVE`，您可能需要等待几分钟（最长），然后才能从该模型创建资产。有关更多信息，请参阅 资产和模型状态。

主题

- 创建资产（控制台）
- 创建资产（Amazon CLI）
- 配置新资产

创建资产（控制台）

您可以使用 Amazon IoT SiteWise 控制台创建资产。

创建资产（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择创建资产。
4. 在 创建资产 页上，执行以下操作：

   a. 对于 模型，选择要从中创建资产的资产模型。

   > **Note**
   > 如果您的模型状态不为 活动，则必须等待其变为活动状态，如果状态为 失败，则必须解决问题。

   b. 为资产输入 名称。

   c. （可选）为资产添加标签。有关更多信息，请参阅 标记您的 Amazon IoT SiteWise 资源。

   d. 选择创建资产。

   创建资产时， Amazon IoT SiteWise 控制台会导航到新资产的页面。在此页面上，您可以查看资产的 状态，最初为 正在创建。此页面会自动更新，因此您可以等待资产的状态更新。

   > **Note**
   > 资产创建流程最多可能需要一分钟。在状态为活动后，您可以对资产执行更新操作。有关更多信息，请参阅 资产和模型状态。

创建资产后，请参阅配置新资产。

创建资产（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 根据资产模型创建资产。

您必须具有 `assetModelId` 才能创建资产。如果您创建了资产模型，但不知道该模型`assetModelId`，请使用 ListAssetModelsAPI 查看所有资产模型。

要根据资产模型创建资产，请使用带有以下参数的 CreateAssetAPI：

- `assetName` – 新资产的名称。为资产命名，以帮助您识别它。
- `assetModelId` – 资产的 ID。这是 UUID 格式的实际 ID，或者 `externalId:myExternalId`（如果有）。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

要创建资产（Amazon CLI），请执行以下步骤：

- 运行以下命令以创建资产。*`asset-name`*替换为资产的*`asset-model-id`*名称以及资产模型的 ID 或外部 ID。

```
aws iotsitewise create-asset \
  --asset-name asset-name \
  --asset-model-id asset-model-id
```

此操作将按以下格式返回包含新资产的详细信息和状态的响应。

```
{
  "assetId": "String",
  "assetArn": "String",
  "assetStatus": {
    "state": "String",
    "error": {
      "code": "String",
      "message": "String"
    }
  }
}
```

资产的 `state` 为 `CREATING`，直至资产创建完毕。

> **Note**
>
> 资产创建流程最多可能需要一分钟。要检查资产的状态，请使用以您的资产 ID 作为`assetId`参数的DescribeAsset操作。资产的 `state` 为 `ACTIVE`后，您可以对资产执行更新操作。有关更多信息，请参阅 资产和模型状态。

创建资产后，请参阅配置新资产。

配置新资产

在中创建资产后 Amazon IoT SiteWise，您可以采取几个后续步骤来充分利用该资产及其数据。这些步骤可能包括配置数据流以从资产中提取数据，设置警报和通知以监控资产的性能，创建可视化和仪表板以显示资产的数据，以及将资产与其他 Amazon 服务或第三方应用程序集成以进行进一步分析或自动化。

使用以下可选操作完成资产配置：

- 管理数据流 Amazon IoT SiteWise，如果您的资产具有测量值属性。
- 更新属性值，如果您的资产具有唯一属性值。
- 关联资产和取消关联资产，如果您的资产是父资产。

在上搜索资产 Amazon IoT SiteWise 控制台

使用 Amazon IoT SiteWise 控制台 搜索功能根据元数据和实时属性值筛选器查找资产。

先决条件

Amazon IoT SiteWise 需要权限才能与之集成， Amazon IoT TwinMaker 以便更好地组织和建模工业数据。如果您已授予权限 Amazon IoT SiteWise，请使用 ExecuteQueryAPI。如果您尚未授予权限 Amazon IoT SiteWise，并且需要入门方面的帮助，请参阅整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker。

高级搜索开启 Amazon IoT SiteWise 控制台

元数据搜索

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产下的高级搜索。
3. 在高级搜索下，选择元数据搜索选项。
4. 填写参数。尽可能多地填写字段，以便实现高效搜索。
   a. 资产名称 - 输入资产的完整名称或部分名称进行广泛搜索。
   b. 属性名称 - 输入完整的属性名称或部分名称进行广泛搜索。

c. 运算符 - 从以下选项中选择一个运算符：

- =
- <
- >
- <=
- >=

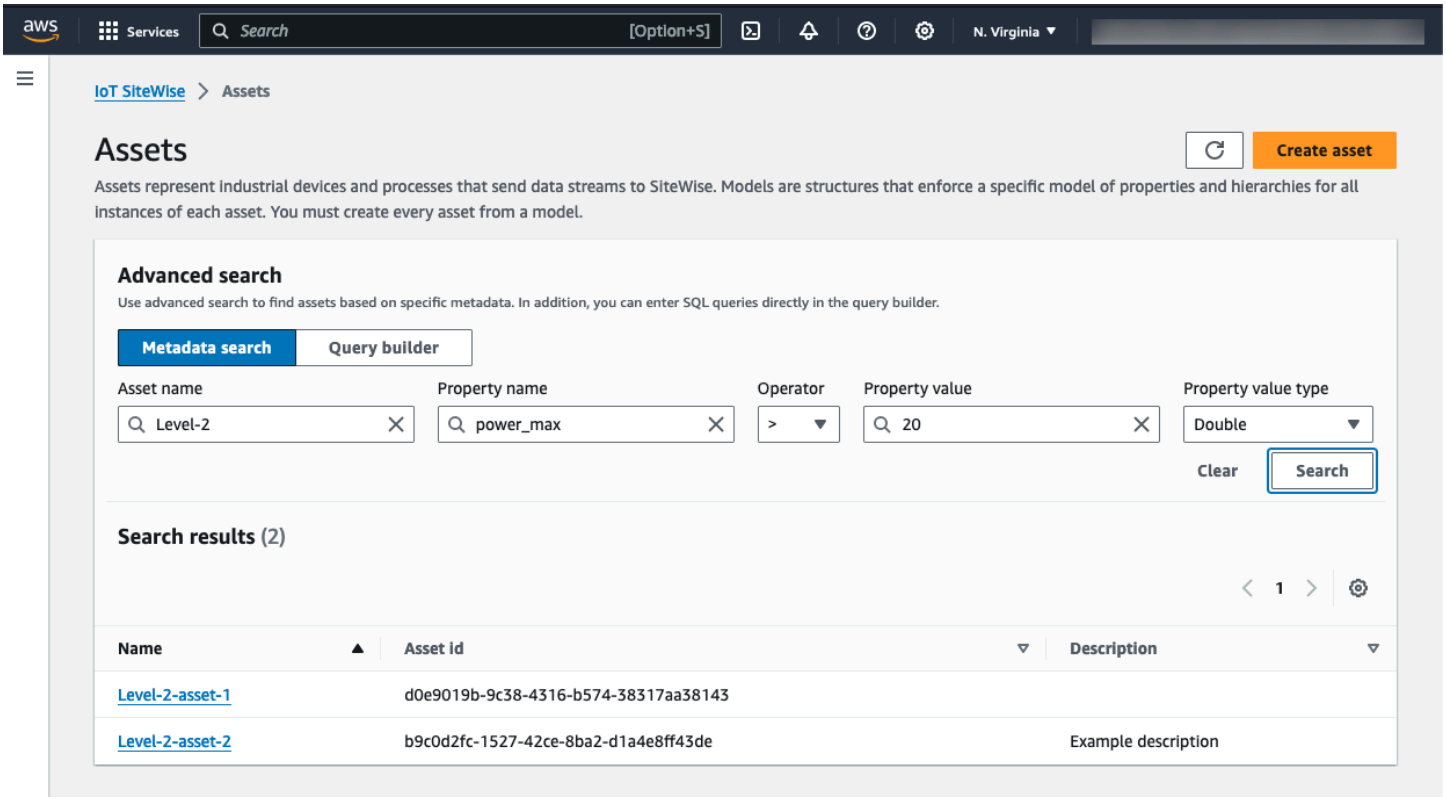
d. 属性值 - 此值会与属性的最新值进行比较。

e. 属性值类型 - 属性的数据类型。请从以下内容中选择：

- 双精度
- 整数
- String
- 布尔值

5. 选择搜索。

6. 在搜索结果表中，从名称列中选择资产。这会将您转到该资产的资产详细信息页面。



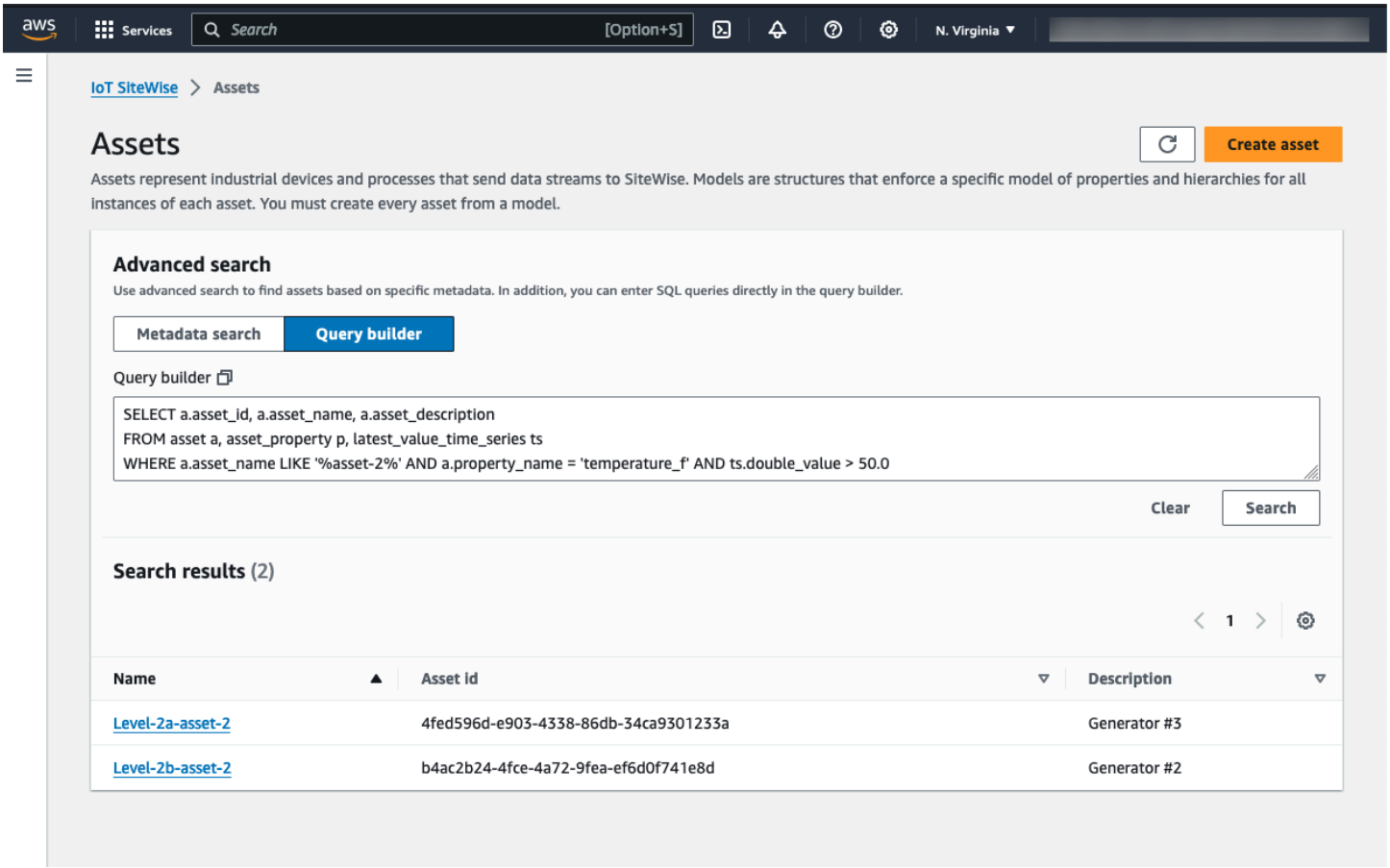
部分搜索

无需为资产搜索提供所有参数。以下是一些使用元数据搜索选项进行部分搜索的示例：

- 按名称查找资产：
  - 仅在资产名称字段中输入一个值。
  - 属性名称和属性值字段留空。
- 查找包含具有特定名称的属性的资产：
  - 仅在属性名称字段中输入一个值。
  - 资产名称和属性值字段留空。
- 根据属性的最新值查找资产：
  - 在属性名称和属性值字段中输入值。
  - 选择运算符和属性值类型。

查询生成器搜索

1. 导航到 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产下的高级搜索。
3. 在高级搜索下，选择查询生成器选项。
4. 在查询生成器窗格中，编写您的 SQL 查询以检索 `asset_name`、`asset_id` 和 `asset_description`。
5. 选择搜索。
6. 在搜索结果表中，从名称列中选择资产。这会将您转到该资产的资产详细信息页面。



> **Note**
>
> - SQL 查询的 `SELECT` 子句必须包含 `asset_name` 和 `asset_id` 字段，以确保搜索结果表中的资产有效。
> - 查询生成器仅在结果表中显示名称、资产 ID 和描述。向 `SELECT` 子句添加更多字段不会在结果表中添加更多列

更新属性值

资产继承其资产模型的属性，包括属性的默认值。某些情况下，您需要保留资产模型的默认属性，如资产制造商特征。其他情况下，您需要更新继承的属性，如资产的纬度和经度。

Updating an attribute value (console)

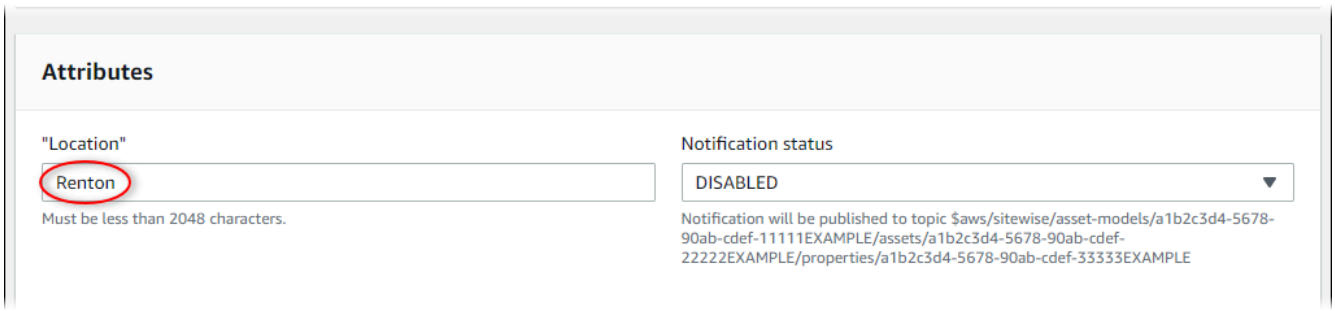
您可以使用 Amazon IoT SiteWise 控制台更新属性资产属性的值。

更新属性值（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要更新其属性的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择编辑。
5. 查找要更新的属性，然后输入其新值。



6. 选择保存。

Updating an attribute value (Amazon CLI)

您可以使用 Amazon Command Line Interface (Amazon CLI) 来更新属性值。

您必须知道资产的 `assetId` 和属性的 `propertyId` 才能完成此过程。您还可以使用外部 ID。如果您创建了资产但不知道该资产`assetId`，请使用 ListAssetsAPI 列出特定模型的所有资产。使用该DescribeAsset操作查看资产的属性，包括财产 IDs。

使用 BatchPutAssetPropertyValue 操作将属性值分配给您的资产。您可以使用此操作一次设置多个属性。此操作的负载包含条目列表，每个条目都包含资产 ID、属性 ID 和属性值。

要更新属性值（Amazon CLI），请执行以下步骤：

1. 创建一个名为 `batch-put-payload.json` 的文件，并将以下 JSON 对象复制到该文件中。此示例负载演示如何设置风力涡轮机的纬度和经度。更新 IDs、值和时间戳以修改您的用例的有效负载。

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

- 负载中的每个条目都包含一个 `entryId`，您可以将其定义为任何唯一的字符串。如果任何请求条目失败，则每个错误都将包含相应请求的 `entryId`，以便您知道要重试的请求。
- 要设置属性值，可以在每个属性属性的列表中包含一个 timestamp-quality-value (TQV) 结构。`propertyValues`此结构必须包含新的 `value` 和当前的 `timestamp`。
  - `value` – 包含下列字段之一的结构，具体取决于要设置的属性的类型：

    - `booleanValue`
    - `doubleValue`
    - `integerValue`
    - `stringValue`
    - `nullValue`
  - `timestamp`— 一种包含当前 Unix 纪元时间（以秒为单位）的结构，`timeInSeconds`。Amazon IoT SiteWise 拒绝任何时间戳在过去存在超过 7 天或将来超过 5 分钟的数据点。

有关如何为 BatchPutAssetPropertyValue 准备负载的更多信息，请查阅使用摄取数据 Amazon IoT SiteWise APIs。

2. 运行以下命令以将属性值发送到 Amazon IoT SiteWise：

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

关联资产和取消关联资产

如果资产的模型定义了任何子资产模型层次结构，则可以将子资产关联到您的资产。父资产可以访问和聚合关联资产中的数据。有关分层资产模型的更多信息，请参阅定义资产模型层次结构。

主题

- 关联和取消关联资产（控制台）
- 关联资产和取消关联资产（Amazon CLI）

关联和取消关联资产（控制台）

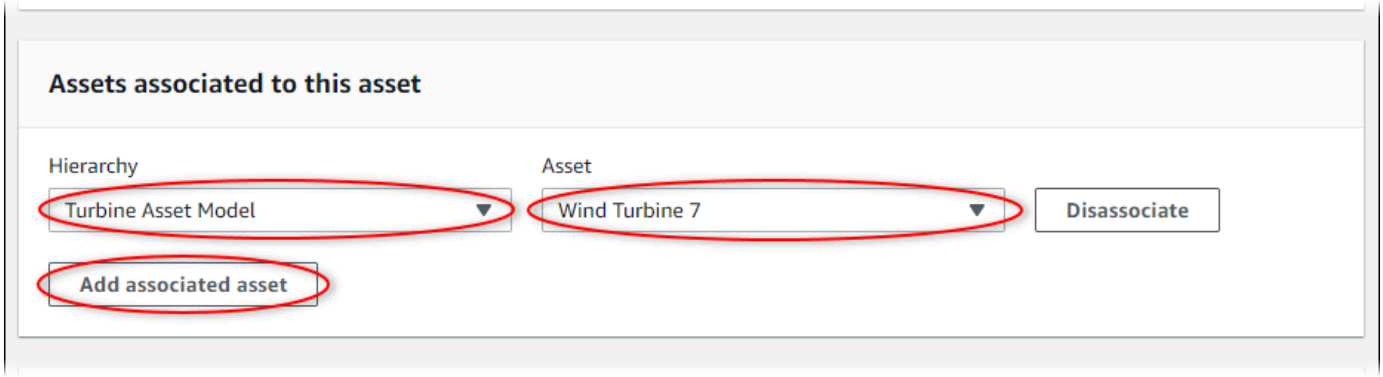
您可以使用 Amazon IoT SiteWise 控制台来关联和取消关联资产。

关联资产（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要为其关联子资产的父资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择编辑。
5. 在 与此资产关联的资产 中，选择 添加关联的资产。



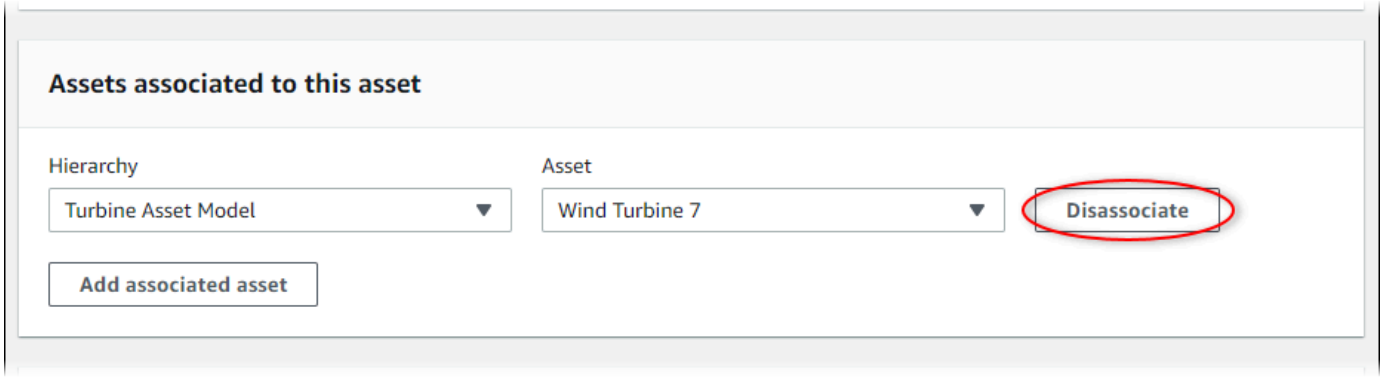
6. 对于 层次结构，选择定义父资产与子资产之间关系的层次结构。
7. 对于 资产，选择要关联的子资产。
8. 选择保存。

取消资产关联（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要取消其子资产关联的父资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择编辑。
5. 在 与此资产关联的资产 中，为资产选择 取消关联。



6. 选择保存。

关联资产和取消关联资产（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 来关联和取消关联资产。

对于此过程，您必须知道父资产模型中定义与子资产模型关系的层次结构的 ID (`hierarchyId`)。使用DescribeAsset操作在响应中查找层次结构 ID。

查找层次结构 ID

- 运行以下命令以描述父资产。*`parent-asset-id`*替换为父资产的 ID 或外部 ID。

```
aws iotsitewise describe-asset --asset-id parent-asset-id
```

此操作将返回包含资产详细信息的响应。响应中包含具有以下结构的 `assetHierarchies` 列表：

```
{
  ...
  "assetHierarchies": [
    {
      "id": "String",
      "name": "String"
    }
  ],
  ...
}
```

层次结构 ID 是资产层次结构列表中的层次结构的 `id` 值。

有了层次结构 ID 后，您可以将资产与该层次结构关联或取消关联。

要将子资产与父资产相关联，请使用AssociateAssets操作。要取消子资产与父资产的关联，请使用DisassociateAssets操作。请指定以下参数（对于这两个操作是相同的）：

- `assetId` - 父资产的 ID 或外部 ID。
- `hierarchyId` - 父资产中的层次结构 ID 或外部 ID。
- `childAssetId` - 子资产的 ID 或外部 ID。

要关联资产（Amazon CLI），请执行以下步骤：

- 运行以下命令以将子资产与父资产关联。将*parent-asset-id*、*hierarchy-id*、和*child-asset-id*替换为相应的 IDs：

```
aws iotsitewise associate-assets \
  --asset-id parent-asset-id \
  --hierarchy-id hierarchy-id \
  --child-asset-id child-asset-id
```

要取消资产关联（Amazon CLI），请执行以下步骤：

- 运行以下命令以取消子资产与父资产的关联。将*parent-asset-id*、*hierarchy-id*、和*child-asset-id*替换为相应的 IDs：

```
aws iotsitewise disassociate-assets \
  --asset-id parent-asset-id \
  --hierarchy-id hierarchy-id \
  --child-asset-id child-asset-id
```

更新资产和模型

您可以在中更新您的资产、资产模型和组件模型， Amazon IoT SiteWise 以修改其名称和定义。这些更新操作是异步的，需要一段时间才能传 Amazon IoT SiteWise播。在进行其他更改之前，请检查资产或模型的状态。您必须等待更改传播之后才能继续使用更新的资产或模型。

主题

- 更新中的资产 Amazon IoT SiteWise

- 更新资产模型和组件模型
- 更新自定义复合模型（组件）
- 对资产模型写入进行乐观锁定

更新中的资产 Amazon IoT SiteWise

您可以使用 Amazon IoT SiteWise 控制台或 API 来更新资产的名称。

更新资产时，资产的状态为`UPDATING`直至更改传播为止。有关更多信息，请参阅 资产和模型状态。

主题

- 更新资产（控制台）
- 更新资产（Amazon CLI）

更新资产（控制台）

您可以使用 Amazon IoT SiteWise 控制台更新资产详细信息。

更新资产（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要更新的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择编辑。
5. 更新资产的 名称。
6. （可选）在此页面上，更新资产的其他信息。有关更多信息，请参阅下列内容：
   - 管理数据流 Amazon IoT SiteWise
   - 更新属性值
   - 与其他 Amazon 服务互动
7. 选择保存。

更新资产（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 来更新资产的名称。

使用UpdateAsset操作来更新资产。指定以下参数：

- `assetId` – 资产的 ID。这是 UUID 格式的实际 ID，或者 `externalId:myExternalId`（如果有）。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。
- `assetName` – 资产的新名称。

要更新资产的名称（Amazon CLI），请执行以下步骤：

- 运行以下命令以更新资产的名称。*asset-id*替换为资产的 ID 或外部 ID。*asset-name*使用资源的新名称更新。

```
aws iotsitewise update-asset \
  --asset-id asset-id \
  --asset-name asset-name
```

更新资产模型和组件模型

使用 Amazon IoT SiteWise 控制台或 API 更新资产模型或组件模型。

您无法更改现有属性的类型或数据类型，或现有指标的窗口。您也无法将模型的类型从资产模型更改为组件模型，反之亦然。

> **Important**
>
> - 如果您从资产模型或组件模型中移除某个属性，则 Amazon IoT SiteWise 会删除该属性之前的所有数据。对于组件模型，这会影响使用该组件模型的所有资产模型，因此，请特别注意了解更改可能会应用到的范围。
> - 如果您从资产模型中移除层次结构定义，则 Amazon IoT SiteWise 取消关联该层次结构中的所有资产。

更新资产模型时，基于该模型的每个资产都会反映您对基础模型所做的任何更改。在更改传播之前，每个资产都具有 `UPDATING` 状态。您必须等待这些资产恢复到 `ACTIVE` 状态，然后才能与它们进行交互。在此期间，更新的资产模型的状态将为 `PROPAGATING`。

更新组件模型时，包含该组件模型的每个资产模型都会反映所作出的更改。如前一段所述，在对组件模型的更改完成传播之前，每个受影响的资产模型都会先处于 UPDATING 状态，然后在更新其关联资产时变为 PROPAGATING 状态。您必须等待这些资产模型恢复到 ACTIVE 状态，然后才能与它们进行交互。在此期间，更新后的组件模型的状态将为 PROPAGATING。

有关更多信息，请参阅 资产和模型状态。

主题

- 更新资产或组件模型（控制台）
- 更新资产或组件模型（Amazon CLI）

更新资产或组件模型（控制台）

您可以使用 Amazon IoT SiteWise 控制台更新资产模型或组件模型。

要更新资产模型或组件模型（控制台），请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择要更新的资产模型或组件模型。
4. 选择编辑。
5. 在 编辑模型 页面上，执行以下任何操作：
   - 在 模型详细信息 中，更改模型的 名称。
   - 更改任何 属性定义。您无法更改现有属性的 数据类型。有关更多信息，请参阅 定义静态数据（属性）。
   - 更改任何 测量值定义。您无法更改现有测量值的 数据类型。有关更多信息，请参阅 定义来自设备的数据流（测量值）。
   - 更改任何 转换定义。有关更多信息，请参阅 转换数据（转换）。
   - 更改任何 指标定义。您无法更改现有指标的 时间间隔。有关更多信息，请参阅 聚合来自属性和其他资产（指标）的数据。
   - （仅限资产模型）更改任何层次结构定义。您无法更改现有层次结构的 层次结构模型。有关更多信息，请参阅 定义资产模型层次结构。
6. 选择 Save。

> **Note**
>
> 如果自您上次打开编辑模型页面以来，其他用户成功更新了资产模型，则在控制台中发出的更新请求将被拒绝。控制台会提示用户刷新编辑模型页面，以获取更新后的模型。您必须重新进行更新，然后重试保存。有关更多信息，请参阅对资产模型写入进行乐观锁定。

更新资产或组件模型（Amazon CLI）

使用 Amazon Command Line Interface (Amazon CLI) 更新资产模型或组件模型。

使用 UpdateAssetModelAPI 更新资产模型或组件模型的名称、描述和属性。仅对于资产模型，您可以更新层次结构。指定以下参数：

- `assetModelId` – 资产的 ID。这是 UUID 格式的实际 ID，或者 `externalId:myExternalId`（如果有）。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

指定有效载荷中更新的模型。要了解资产模型或组件模型的预期格式，请参阅在中创建资产模型 Amazon IoT SiteWise。

> **Warning**
>
> UpdateAssetModelAPI 会使用您在有效载荷中提供的模型覆盖现有模型。为避免删除模型的属性或层次结构，您必须在更新的模型负载中包含它们的 IDs 和定义。要了解如何查询模型的现有结构，请参阅DescribeAssetModel操作。

> **Note**
>
> 通过以下过程只能更新 `AWS/ALARM` 类型的复合模型。如果要更新`CUSTOM`复合模型，请UpdateAssetModelCompositeModel改用。有关更多信息，请参阅 更新自定义复合模型（组件）。

要更新资产模型或组件模型（Amazon CLI），请执行以下步骤：

1. 运行以下命令以检索现有模型定义。*`asset-model-id`*替换为要更新的资产模型或组件模型的 ID 或外部 ID。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

上面的命令会返回与模型的最新版本相对应的模型定义。

对于资产模型处于 FAILED 状态的用例，请检索与其活动版本相对应的有效模型定义，以构建您的更新请求。有关详细信息，请参阅资产模型版本。运行以下命令以检索活动模型定义：

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id --asset-model-
version ACTIVE
```

此操作将返回包含模型详细信息的响应。该响应具有以下结构。

```
{
    "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"
}
```

有关更多信息，请参阅 DescribeAssetModel 操作。

2. 创建一个名为 `update-asset-model.json` 的文件，并将上一命令的响应复制到该文件中。
3. 从 `update-asset-model.json` 中的 JSON 对象中删除以下键/值对：

- assetModelId
- assetModelArn
- assetModelCompositeModelSummaries
- assetModelCreationDate
- assetModelLastUpdateDate
- assetModelStatus
- assetModelType
- assetModelVersion
- eTag

该UpdateAssetModel操作需要具有以下结构的有效负载：

```
{
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition,
  "assetModelCompositeModels": Array of AssetModelCompositeModel
}
```

4. 在 update-asset-model.json 中，执行以下任何操作：
   - 更改资产模型的名称 (assetModelName)。
   - 更改、添加或删除资产模型的描述 (assetModelDescription)。
   - 更改、添加或删除资产模型的任何属性 (assetModelProperties)。您无法更改现有属性的 dataType 或现有指标的 window。有关更多信息，请参阅 定义数据属性。
   - 更改、添加或删除资产模型的任何层次结构 (assetModelHierarchies)。您无法更改现有层次结构的 childAssetModelId。有关更多信息，请参阅 定义资产模型层次结构。
5. 运行以下命令，使用存储在 update-asset-model.json 中的定义更新资产模型。*asset-model-id*替换为资产模型的 ID：

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://model-payload.json
```

> **Important**
>
> 当多个用户同时更新一个资产模型时，一个用户所作的更改可能会无意中被另一个用户覆盖。为防止出现这种情况，您必须定义一个有条件的更新请求。请参阅对资产模型写入进行乐观锁定。

更新自定义复合模型（组件）

您可以使用 Amazon IoT SiteWise API 更新自定义复合模型，也可以使用 Amazon IoT SiteWise 控制台更新组件。

主题

- 更新组件（控制台）
- 更新自定义复合模型（Amazon CLI）

更新组件（控制台）

您可以使用 Amazon IoT SiteWise 控制台更新组件。

要创建组件（控制台），请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择组件所属的资产模型。
4. 在属性选项卡上，选择组件。
5. 选择要更新的组件。
6. 选择编辑。
7. 在编辑组件页面上，执行以下任何操作：
   - 在 模型详细信息 中，更改模型的 名称。
   - 更改任何 属性定义。您无法更改现有属性的 数据类型。有关更多信息，请参阅 定义静态数据（属性）。
   - 更改任何 测量值定义。您无法更改现有测量值的 数据类型。有关更多信息，请参阅 定义来自设备的数据流（测量值）。
   - 更改任何 转换定义。有关更多信息，请参阅 转换数据（转换）。

- 更改任何 指标定义。您无法更改现有指标的 时间间隔。有关更多信息，请参阅 聚合来自属性和其他资产（指标）的数据。

8. 选择 Save。

更新自定义复合模型（Amazon CLI）

使用 Amazon Command Line Interface (Amazon CLI) 更新自定义复合模型。

要更新名称或描述，请使用UpdateAssetModelCompositeModel操作。您也可以更新属性（仅限于内联自定义复合模型）。您无法更新 component-model-based自定义复合模型的属性，因为其引用的组件模型提供了其关联的属性。

> **Important**
>
> 如果您从自定义复合模型中移除某个属性，则 Amazon IoT SiteWise 会删除该属性的所有先前数据。您无法更改现有属性的类型或数据类型。
> 要将现有的复合模型属性替换为具有相同 `name` 的新属性，请执行以下操作：
>
> 1. 提交移除所有现有属性的 `UpdateAssetModelCompositeModel` 请求。
> 2. 提交第二个包含新属性的 `UpdateAssetModelCompositeModel` 请求。新的资产属性将与前一个资产属性name相同， Amazon IoT SiteWise 并将生成一个新的唯一资产id。

要更新自定义复合模型（Amazon CLI），请执行以下步骤：

1. 要检索现有复合模型定义，请运行以下命令。*composite-model-id*替换为要更新的自定义复合模型的 ID 或外部 ID，以及*asset-model-id*与自定义复合模型关联的资产模型。有关更多信息，请参阅 Amazon IoT SiteWise 《用户指南》。

   a. 运行以下命令：

   ```
   aws iotsitewise describe-asset-model-composite-model \
   --asset-model-composite-model-id composite-model-id \
   --asset-model-id asset-model-id
   ```

   b. 上面的命令会返回与关联模型的最新版本相对应的复合模型定义。对于资产模型处于 `FAILED` 状态的用例，请检索与其活动版本相对应的有效模型定义，以构建您的更新请求。有关详细信息，请参阅资产模型版本。

c. 运行以下命令以检索活动模型定义：

```
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. 有关更多信息，请参阅 DescribeAssetModelCompositeModel 操作。

2. 创建一个名为 `update-custom-composite-model.json` 的文件，然后将上一命令的响应复制到该文件中。

3. 从 `update-custom-composite-model.json` 的 JSON 对象中删除每个键/值对，但保留以下字段：

   - `assetModelCompositeModelName`
   - `assetModelCompositeModelDescription`（如果存在）
   - `assetModelCompositeModelProperties`（如果存在）

4. 在 `update-custom-composite-model.json` 中，执行以下任何操作：

   - 更改 `assetModelCompositeModelName` 的值。
   - 添加或移除 `assetModelCompositeModelDescription`，或更改其值。
   - 仅适用于内联自定义复合模型：更改、添加或移除 `assetModelCompositeModelProperties` 中的资产模型的任何属性。

   有关此文件所需格式的更多信息，请参阅的请求语法UpdateAssetModelCompositeModel。

5. 运行以下命令，使用存储在 `update-custom-composite-model.json` 中的定义更新自定义复合模型。*`composite-model-id`*替换为复合模型的 ID 和它所在*`asset-model-id`*的资产模型的 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**
>
> 当多个用户同时更新一个资产模型时，一个用户所作的更改可能会无意中被另一个用户覆盖。为防止出现这种情况，您必须定义一个有条件的更新请求。请参阅对资产模型写入进行乐观锁定。

对资产模型写入进行乐观锁定

更新资产模型时，用户会执行以下操作：

1. 阅读当前的资产模型定义。
2. 根据所需的更改，编辑资产模型定义。
3. 使用新定义更新资产模型。

在有两个用户更新模型的场景中，可能出现以下情况：

- 用户 A 读取资产模型 X 的定义。
- 用户 B 读取资产模型 X 的定义并提交更改，这会修改 X 的定义。
- 用户 A 提交并覆盖用户 B 对资产模型 X 所作的更改，而没有验证或合并用户 B 的更改。

乐观锁定是一种用于防止 Amazon IoT SiteWise 意外覆盖的机制，如上面的场景。乐观锁是一种策略，可确保正在更新或删除的资产模型的当前版本与其在 Amazon IoT SiteWise中的当前版本相同。这样可以防止资产模型写入被意外的更新所覆盖。

按照以下步骤，使用乐观锁执行资产模型写入：

主题

- 使用乐观锁执行资产模型写入（控制台）
- 使用乐观锁执行资产模型写入（Amazon CLI）

使用乐观锁执行资产模型写入（控制台）

以下过程描述了如何在控制台中对资产模型的活动版本进行乐观锁定，以执行资产模型写入。

1. 导航至 Amazon IoT SiteWise 控制台。

2. 在导航窗格中，选择模型。
3. 选择要更新的资产模型或组件模型。
4. 选择编辑。
5. 在编辑模型页面上进行更改。
6. 选择保存。

> **Note**
>
> 有时，在用户开始编辑模型与保存对模型所作的编辑之间，会发生一次或多次成功的模型更新。
> 为确保用户不会意外覆盖成功的新更新，用户的写入将被拒绝。控制台禁用保存按钮，并会提示用户刷新编辑模型页面。用户必须再次更新该模型的新活动版本。用户必须执行以下额外步骤：

7. 选择刷新。
8. 再次执行步骤 5 和 6。

使用乐观锁执行资产模型写入（Amazon CLI）

以下过程描述了如何在 Amazon CLI中通过乐观锁执行资产模型写入。

1. 获取与当前模型定义 ETag 关联的

   ETag 是为资产模型的每个新表示形式生成的唯一令牌。调用 DescribeAssetModelAPI 以获取当前的资产模型定义，并ETag从响应中关联起来。

   在并发更新期间，用户更新可能成功（模型处于 ACTIVE 状态），也可能不成功（模型处于 FAILED 状态）。为确保用户不会意外覆盖成功的更新，您必须从 资产模型版本 中检索资产模型的活动版本并获取 ETag 值。

   运行以下命令：

   ```
   aws iotsitewise describe-asset-model --asset-model-id asset-model-id \
   --asset-model-version ACTIVE
   ```

   该响应会返回以下结构：

```
{
  "assetModelId": "String",
  "assetModelArn": "String",
  "assetModelName": "String",
  ...
  "eTag": "String"
}
```

> **Note**
>
> 您必须检索资产模型的最新版本及其 ETag，以免覆盖任何更新。

2. 使用写入条件执行更新和删除操作

   以下资产模型 APIs 支持乐观锁定：

   - UpdateAssetModel
   - DeleteAssetModel
   - CreateAssetModelCompositeModel
   - UpdateAssetModelCompositeModel
   - DeleteAssetModelCompositeModel

   > **Note**
   >
   > 以下场景使用 UpdateAssetModel API 作为参考。这些条件适用于上面列出的所有操作。

   以下场景描述了不同的写入条件，具体取决于并发控制要求：

   - 为了不覆盖任何成功的更新，请运行以下命令。自上次读取活动版本以来，不得存在新的活动版本。将 e-tag 替换为读取活动版本时使用的 API 操作中返回的 ETag。

```
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
```

- 当模型创建失败时，由于处于 FAILED 状态，所以其活动版本尚不存在。在提交更改之前，仍然可以覆盖当前存在的新活动版本。如果在上次读取期间不存在活动版本，则运行以下命令，以免覆盖新的活动版本。

```
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
```

- 为避免覆盖任何成功或失败的更新，请运行以下命令。此命令定义了一个写入条件，可确保自上次读取最新版本以来未创建最新版本。将 e-tag 替换为读取活动版本时使用的 API 操作中返回的 ETag。

```
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
```

  如果写入条件的计算结果为 FALSE，则写入请求将失败，并显示 PreconditionFailedException。

删除中的资源和模型 Amazon IoT SiteWise

当你使用完资源和模型 Amazon IoT SiteWise 后，你可以将其删除。删除操作是异步的，需要一段时间才能传 Amazon IoT SiteWise播。

主题

- 删除中的资产 Amazon IoT SiteWise
- 删除中的资产模型 Amazon IoT SiteWise

删除中的资产 Amazon IoT SiteWise

您可以使用 Amazon IoT SiteWise 控制台或 API 删除环境中不再需要的资产。删除资产模型还会删除所有关联的资产和组件模型。但是，请务必注意，删除资产或模型是一项永久性操作，与已删除资源关联的所有数据也将被删除。在删除资产或模型之前，建议先查看任何可能受到影响的依赖项或集成，并确保您已备份所有重要数据。

您必须先取消资产与其子资产和父资产的关联，然后才能删除该资产。有关更多信息，请参阅 关联资产和取消关联资产。如果使用 Amazon Command Line Interface (Amazon CLI)，则可以使用ListAssociatedAssets操作列出资产的子项。

删除资产时，资产的状态为 `DELETING`，直至更改传播为止。有关更多信息，请参阅 资产和模型状态。删除资产后，您将无法再查询该资产。如果查询，API 将返回 HTTP 404 响应。

> **Important**
>
> Amazon IoT SiteWise 删除已删除资产的所有属性数据。

主题

- 删除资产（控制台）
- 删除资产（Amazon CLI）

删除资产（控制台）

您可以使用 Amazon IoT SiteWise 控制台删除资产。

删除资产（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要删除的资产。

> **Tip**
>
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 如果该资产具有任何 关联的资产，请删除每个资产。您可以选择资产的名称以导航到其页面，从中可以将其删除。
5. 在资产页面上，选择 删除。
6. 在删除资产对话框中，执行以下操作：
   a. 输入 **Delete** 以确认删除。
   b. 选择删除。

删除资产（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 删除资产。

使用DeleteAsset操作删除资产。指定以下参数：

- `assetId` – 资产的 ID。这是 UUID 格式的实际 ID，或者 `externalId:myExternalId`（如果有）。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

要删除资产（Amazon CLI），请执行以下步骤：

1. 运行以下命令以列出资产的层次结构。*`asset-id`*替换为资产的 ID 或外部 ID：

```
aws iotsitewise describe-asset --asset-id asset-id
```

   此操作将返回包含资产详细信息的响应。响应中包含具有以下结构的 `assetHierarchies` 列表：

```
{
  ...
  "assetHierarchies": [
    {
      "id": "String",
      "name": "String"
    }
  ],
  ...
}
```

   有关更多信息，请参阅 DescribeAsset 操作。

2. 对于每个层次结构，运行以下命令以列出与该层次结构关联的资产子项。*asset-id*替换为资产的 ID 或外部 ID 以及*hierarchy-id*层次结构的 ID 或外部 ID。

```
aws iotsitewise list-associated-assets \
  --asset-id asset-id \
  --hierarchy-id hierarchy-id
```

有关更多信息，请参阅 ListAssociatedAssets 操作。

3. 运行以下命令以删除每个关联的资产，然后删除该资产。*asset-id*替换为资产的 ID 或外部 ID。

```
aws iotsitewise delete-asset --asset-id asset-id
```

删除中的资产模型 Amazon IoT SiteWise

您可以使用 Amazon IoT SiteWise 控制台或 API 删除资产模型。

您必须先从资产模型中删除创建的所有资产，然后才能删除资产模型。

删除资产模型时，它的状态为 DELETING，直至更改传播为止。有关更多信息，请参阅 资产和模型状态。删除资产模型后，您将无法再查询该资产模型。如果查询，API 将返回 HTTP 404 响应。

主题

- 删除资产模型（控制台）
- 删除资产模型（Amazon CLI）

删除资产模型（控制台）

您可以使用 Amazon IoT SiteWise 控制台删除资产模型。

删除资产模型（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 模型。
3. 选择要删除的资产模型。
4. 如果模型具有任何 资产，请删除每个资产。选择资产的名称以导航到其页面，您可以从中将其删除。有关更多信息，请参阅 删除资产（控制台）。

5. 在模型页面上，选择 删除。
6. 在删除模型对话框中，执行以下操作：
   a. 输入 **Delete** 以确认删除。
   b. 选择删除。

删除资产模型（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 删除资产模型。

使用DeleteAssetModel操作删除资产模型。指定以下参数：

- `assetModelId` – 资产的 ID。这是 UUID 格式的实际 ID，或者 `externalId:myExternalId`（如果有）。有关更多信息，请参阅《Amazon IoT SiteWise 用户指南》中的 使用外部引用对象 IDs。

要删除资产模型（Amazon CLI），请执行以下步骤：

1. 运行以下命令以列出从模型创建的所有资产。*`asset-model-id`*替换为资产模型的 ID 或外部 ID。

   ```
   aws iotsitewise list-assets --asset-model-id asset-model-id
   ```

   有关更多信息，请参阅 ListAssets 操作。
2. 如果上一命令返回模型中的任何资产，请删除每个资产。有关更多信息，请参阅 删除资产（Amazon CLI）。
3. 运行以下命令以删除资产模型。*`asset-model-id`*替换为资产模型的 ID 或外部 ID。

   ```
   aws iotsitewise delete-asset-model --asset-model-id asset-model-id
   ```

> **Important**
> 为避免删除自上次读取操作以来同时更新的资产模型，您必须定义有条件删除请求。请参阅对资产模型写入进行乐观锁定。

使用资产和模型进行批量操作

要处理大量资产或资产模型，请使用批量操作将资源批量导入和导出到其他位置。例如，您可以创建一个数据文件来定义 Amazon S3 存储桶中的资产或资产模型，然后使用批量导入在 Amazon IoT SiteWise中创建或更新它们。或者，如果您有大量资产或资产模型 Amazon IoT SiteWise，则可以将其导出到 Amazon S3。

> **Note**
>
> 您可以 Amazon IoT SiteWise 通过调用 Amazon IoT TwinMaker API 中的操作在中执行批量操作。无需设置 Amazon IoT TwinMaker 或创建 Amazon IoT TwinMaker 工作区即可执行此操作。您只需要一个可在其中放置 Amazon IoT SiteWise 内容的 Amazon S3 存储桶。

主题

- 关键概念和术语
- 支持的功能
- 批量操作先决条件
- 运行批量导入作业
- 运行批量导出作业
- 作业进度跟踪和错误处理
- 导入元数据示例
- 导出元数据示例
- Amazon IoT SiteWise 元数据传输任务架构

关键概念和术语

Amazon IoT SiteWise 批量导入和导出功能依赖于以下概念和术语：

- 导入：将资产或资产模型从 Amazon S3 存储桶中的文件移动到 Amazon IoT SiteWise的操作。
- 导出：将资产或资产模型从 Amazon S3 存储桶移 Amazon IoT SiteWise 至 Amazon S3 存储桶的操作。
- 源：您要从中移动内容的起始位置。

  例如，Amazon S3 存储桶既是导入源，又 Amazon IoT SiteWise 是导出源。

- 目标：您要将内容移动到的所需位置。

  例如，Amazon S3 存储桶是导出目的地， Amazon IoT SiteWise 也是导入目的地。
- Amazon IoT SiteWise 架构：此架构用于从中导入和导出元数据 Amazon IoT SiteWise。
- 顶级资源：可以单独创建或更新的 Amazon IoT SiteWise 资源，例如资产或资产模型。
- 子资源：顶级 Amazon IoT SiteWise 资源中的嵌套资源。示例包括属性、层次结构和复合模型。
- 元数据：成功导入或导出资源所需的关键信息。元数据的示例包括资产和资产模型的定义。
- metadataTransferJob：运行时创建的对象`CreateMetadataTransferJob`。

支持的功能

本主题说明了在运行批量操作时您可以完成哪些任务。批量操作支持以下功能：

- 顶级资源创建：导入未定义 ID 的资产或资产模型或者其 ID 与现有资产或资产模型不匹配的资产或资产模型时，该资产或资产模型将被创建为新资源。
- 顶级资源替换：导入其 ID 与已存在的资产或资产模型相匹配的资产或资产模型时，该资产或资产模型将替换现有资源。
- 子资源创建、替换或删除：当您的导入内容替换顶级资源（例如资产或资产模型）时，新定义将替换所有子资源，例如属性、层次结构或复合模型。

  例如，如果您在批量导入期间更新了资产模型，并且更新后的版本定义了原始版本中不存在的属性，则会创建一个新属性。如果定义的是一个已经存在的属性，则现有属性将被更新。如果更新后的资产模型省略了原始版本中存在的属性，则该属性将被删除。
- 不删除顶级资源：批量操作不会删除资产或资产模型。批量操作只能创建或更新它们。

批量操作先决条件

本节介绍批量操作的先决条件，包括在 Amazon 服务和本地计算机之间交换资源的 Amazon Identity and Access Management (IAM) 权限。在开始批量操作之前，必须满足以下先决条件：

- 创建 Amazon S3 存储桶以存储资源。有关使用 Amazon S3 的更多信息，请参阅什么是 Amazon S3？

IAM 权限

要执行批量操作，您必须创建一个 Amazon Identity and Access Management (IAM) 策略，该策略具有允许 Amazon S3 和您的本地计算机之间交换 Amazon 资源的权限。 Amazon IoT SiteWise有关创建 IAM 策略的更多信息，请参阅创建 IAM 策略。

要执行批量操作，您需要以下策略。

Amazon IoT SiteWise 政策

此策略允许访问批量操作所需 Amazon IoT SiteWise 的 API 操作：

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

Amazon IoT TwinMaker 政策

此策略允许访问您用来处理批量操作的 Amazon IoT TwinMaker API 操作：

```
{
    "Sid": "MetadataTransferJobApiAccess",
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:CreateMetadataTransferJob",
        "iottwinmaker:CancelMetadataTransferJob",
        "iottwinmaker:GetMetadataTransferJob",
        "iottwinmaker:ListMetadataTransferJobs"
    ],
    "Resource": "*"
}
```

Amazon S3 策略

此策略提供对 Amazon S3 存储桶的访问权限，以传输用于批量操作的元数据。

For a specific Amazon S3 bucket

如果您使用一个特定的存储桶来处理批量操作元数据，则此策略提供对该存储桶的访问权限：

```
{
    "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

如果您要使用许多不同的存储桶来处理批量操作元数据，则此策略提供对任何存储桶的访问权限：

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": "*"
}
```

有关对导入和导出操作进行故障排除的信息，请参阅批量导入和导出故障排除。

运行批量导入作业

批量导入是将元数据移动到 Amazon IoT SiteWise 工作区的操作。例如，批量导入可以将元数据从本地文件或 Amazon S3 存储桶中的文件移动到 Amazon IoT SiteWise 工作空间。

步骤 1：准备要导入的文件

下载本 Amazon IoT SiteWise 机格式文件以导入资源和资产模型。有关更多信息，请参阅Amazon IoT SiteWise 元数据传输任务架构。

步骤 2：将准备好的文件上传到 Amazon S3

将文件上传到 Amazon S3。有关详细信息，请参阅《Amazon Simple Storage Service 用户指南》中的将文件上传到 Amazon S3。

导入元数据（控制台）

您可以使用批量导入元数据。 Amazon IoT SiteWise 控制台 按照 步骤 1：准备要导入的文件 和 步骤 2：将准备好的文件上传到 Amazon S3 准备一个可以导入的文件。

将数据从 Amazon S3 导入到 Amazon IoT SiteWise 控制台

1. 导航至 Amazon IoT SiteWise 控制台。
2. 从导航窗格中选择新批量操作。
3. 选择新导入以开始导入流程。
4. 在导入元数据页面上：
   - 选择浏览 Amazon S3，查看 Amazon S3 存储桶和文件。
   - 导航到包含准备好的导入文件的 Amazon S3 存储桶。
   - 选择要导入的文件。
   - 查看所选文件，然后选择导入。
5. 的 “ SiteWise 元数据批量操作” 页面在 “作业进度” 表中 Amazon IoT SiteWise 控制台 显示新创建的导入任务。

导入元数据（Amazon CLI）

要执行导入操作，请使用以下过程：

将数据从 Amazon S3 导入到 Amazon CLI

1. 按照 Amazon IoT SiteWise 元数据传输任务架构，创建指定要导入的资源的元数据文件。将此文件存储在 Amazon S3 存储桶中。

   有关要导入的元数据文件的示例，请参阅导入元数据示例。
2. 现在，使用请求正文创建一个 JSON 文件。请求正文指定传输作业的源和目标。此文件与上一步中的文件无关。请务必将您的 Amazon S3 存储桶指定为源，将 `iotsitewise` 指定为目标。

   以下示例显示的是请求正文：

```
{
      "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. `CreateMetadataTransferJob`通过运行以下 Amazon CLI 命令来调用。在此示例中，上一步中的请求正文文件名为 `createMetadataTransferJobExport.json`。

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
  --cli-input-json file://createMetadataTransferJobImport.json
```

这将创建一个元数据传输作业，并开始传输所选资源的过程。

运行批量导出作业

批量导出是将元数据从 Amazon IoT SiteWise 工作空间移动到 Amazon S3 存储桶的操作。

将 Amazon IoT SiteWise 内容批量导出到 Amazon S3 时，您可以指定筛选条件来限制要导出的特定资产模型和资产。

必须在 JSON 请求的源部分中的 `iotSiteWiseConfiguration` 部分指定筛选条件。

> **Note**
>
> 您可以在请求中包含多个筛选条件。批量操作将导出与任何筛选条件匹配的资产模型和资产。如果您未提供任何筛选条件，则批量操作会导出所有资产模型和资产。

Example 带筛选条件的请求正文

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

导出元数据（控制台）

以下过程介绍了控制台导出操作：

在中创建导出任务 Amazon IoT SiteWise 控制台

1. 导航至 Amazon IoT SiteWise 控制台。
2. 从导航窗格中选择新批量操作。
3. 选择新导出以开始导出过程。
4. 在导出元数据页面上：

- 为导出作业输入一个名称。这是为您的 Amazon S3 存储桶中的导出文件使用的名称。
- 选择要导出的资源，这将为作业设置筛选条件：
  - 导出所有资产和资产模型。针对资产和资产模型使用筛选条件。
  - 导出资产。筛选您的资产。
    - 选择要用于导出筛选条件的资产。
    - （可选）添加子代或关联的资产模型。
  - 导出资产模型。筛选您的资产模型。
    - 选择要用于导出筛选条件的资产模型。
    - （可选）添加子代或关联的资产，或同时添加两者。
  - 选择下一步。
- 导航到 Amazon S3 存储桶：
  - 选择浏览 Amazon S3，查看 Amazon S3 存储桶和文件。
  - 导航到必须将文件放置在的 Amazon S3 存储桶。
  - 选择下一步。
- 查看导出任务并选择导出。

5. 的 “ SiteWise 元数据批量操作” 页面在 “作业进度” 表中 Amazon IoT SiteWise 控制台 显示新创建的导入任务。

有关导出元数据时使用筛选条件的不同方法，请参阅导出元数据示例。

导出元数据（Amazon CLI）

以下过程解释了 Amazon CLI 导出操作：

将数据从导出 Amazon IoT SiteWise 到 Amazon S3

1. 使用请求正文创建 JSON 文件。请求正文指定传输作业的源和目标。以下示例是请求正文示例：

```
{
    "metadataTransferJobId": "your-transfer-job-Id",
    "sources": [{
        "type": "iotsitewise"
    }],
    "destination": {
```

```
        "type": "s3",
        "s3Configuration": {
            "location": "arn:aws:s3:::amzn-s3-demo-bucket"
        }
    }
}
```

请务必将您的 Amazon S3 存储桶指定为元数据传输作业的目标。

> **Note**
>
> 此示例将导出您的所有资产模型和资产。要将导出限制为特定的资产模型或资产，您可以在请求正文中加入筛选条件。有关应用导出筛选条件的更多信息，请参阅导出元数据示例。

2. 保存您的请求正文文件以供下一步使用。在本示例中，文件命名为 `createMetadataTransferJobExport.json`。
3. `CreateMetadataTransferJob`通过运行以下 Amazon CLI 命令来调用：

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
        --cli-input-json file://createMetadataTransferJobExport.json
```

使用您自己的传输文件名称替换输入 JSON 文件 `createMetadataTransferJobExport.json`。

作业进度跟踪和错误处理

批量处理作业需要时间来处理。每个任务都按照 Amazon IoT SiteWise 收到请求的顺序进行处理。它 one-at-a-time针对每个账户进行处理。任务完成后，队列中的下一个任务会自动开始处理。 Amazon IoT SiteWise 异步解析任务，并在任务进行时更新每个任务的状态。每个作业都有一个状态字段，包含资源状态和错误消息（如果适用）。

状态可以是以下值之一：

- `VALIDATING` — 正在验证作业，包括提交的文件格式及其内容。
- `PENDING` — 作业在队列中。您可以从 Amazon IoT SiteWise 控制台取消处于此状态的作业，但所有其他状态都将持续到最后。

- RUNNING — 正在处理作业。它正在创建和更新导入文件所定义的资源，或者根据所选的导出作业筛选条件导出资源。如果取消，系统不会删除此作业导入的任何资源。请参阅查看作业进度和详细信息（控制台）了解更多信息。
- CANCELLING — 正在主动取消作业。
- ERROR — 一个或多个资源处理失败。有关更多信息，请查看详细的作业报告。请参阅检查错误详情（控制台）了解更多信息。
- COMPLETED — 作业已完成，没有错误。
- CANCELLED — 作业已取消且未排队。如果您取消了某项RUNNING任务，则不会从中删除取消时该任务已导入的资源 Amazon IoT SiteWise。

主题

- 作业进度跟踪
- 检查错误 Amazon IoT SiteWise

作业进度跟踪

查看作业进度和详细信息（控制台）

要启动批量作业，请参阅导入元数据（控制台）或导出元数据（控制台）。

Amazon IoT SiteWise 控制台中的 Job 进度概览：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 从导航窗格中选择新批量操作。
3. Amazon IoT SiteWise 控制台中的作业进度表显示了批量操作任务的列表。
4. 作业类型列描述了是导出作业还是导入作业。导入日期列显示了作业的开始日期。
5. 状态列显示了作业的状态。您可以选择一个作业来查看有关该作业的详细信息。
6. 所选作业成功完成后会显示成功，如果作业失败，则会显示失败列表。同时还会显示每种资源类型的错误描述。

Amazon IoT SiteWise 控制台中的 Job 详情概述：

Amazon IoT SiteWise 控制台中的作业进度表显示了批量操作任务的列表。

1. 选择一个作业以查看更多详情。

2. 对于导入作业，Data source ARN 表示导入文件的 Amazon S3 位置。
3. 对于导出作业，Data destination ARN 表示文件在导出后的 Amazon S3 位置。
4. Status 和 Status reason 提供有关当前作业的更多详细信息。有关更多信息，请参阅作业进度跟踪和错误处理。
5. Queued position 表示作业在处理队列中的位置。一次只能处理一个作业。排队位置为 1 表示接下来将处理该作业。
6. 作业详情页面还会显示作业进度计数。
   - 作业进度计数类型包括：
     i. Total resources — 表示传输过程中的资产总数。
     ii. Succeeded — 表示在此过程中成功传输的资产数量。
     iii. Failed — 表示在此过程中失败的资产数量。
     iv. Skipped — 表示在此过程中跳过的资产数量。
7. 作业状态为 PENDING 或 VALIDATING，则所有作业进度计数会显示为 –。这表示正在评估作业进度计数。
8. 作业状态 RUNNING 显示 Total resources 计数，即已提交待处理的作业。详细计数（Succeeded、Failed 和 Skipped）适用于已处理的资源。在作业的状态变为 COMPLETED 或 ERROR 之前，详细计数的总和小于 Total resources 计数。
9. 如果作业的状态为 COMPLETED 或 ERROR，则 Total resources 计数等于详细计数（Succeeded、Failed 和 Skipped）的总和。
10. 如果作业的状态为 ERROR，请查看作业失败表，了解有关特定错误和失败的详细信息。有关更多信息，请参阅检查错误详情（控制台）。

查看作业进度和详细信息（Amazon CLI）

开始批量操作后，您可以使用以下 API 操作检查或更新其状态：

- 要检索有关特定任务的信息，请使用 GetMetadataTransferJobAPI 操作。

  使用 **GetMetadataTransferJob** API 检索信息：

  1. 创建并运行传输作业。调用 GetMetadataTransferJob API。

Example Amazon CLI 命令：

```
aws iottwinmaker get-metadata-transfer-job \
        --metadata-transfer-job-id your_metadata_transfer_job_id \
        --region your_region
```

2. `GetMetadataTransferJob` API 返回一个带有以下参数的 `MetadataTransferJobProgress` 对象：
   - succeededCount — 表示在此过程中成功传输的资产数量。
   - failedCount — 表示在此过程中失败的资产数量。
   - skippedCount — 表示在此过程中跳过的资产数量。
   - totalCount — 表示传输过程中的资产总数。

   这些参数表示作业进度状态。如果状态为 `RUNNING`，它们会帮助跟踪仍有待处理的资源数量。

   如果您遇到架构验证错误，或者 failedCount 大于或等于 1，则作业进度状态将变为 `ERROR`。作业的完整错误报告位于您的 Amazon S3 存储桶中。有关更多信息，请参阅检查错误 Amazon IoT SiteWise。

- 要列出当前作业，请使用 ListMetadataTransferJobsAPI 操作。

  使用 JSON 文件根据返回作业的当前状态对其进行筛选。请参见以下过程：

1. 要指定要使用的过滤器，请创建一个 Amazon CLI 输入 JSON 文件。要使用：

```
{
    "sourceType": "s3",
    "destinationType": "iottwinmaker",
    "filters": [{
        "state": "COMPLETED"
    }]
}
```

   有关有效`state`值的列表，请参阅 Amazon IoT TwinMaker API 参考指南ListMetadataTransferJobsFilter中的。

2. 在以下 Amazon CLI 示例命令中使用 JSON 文件作为参数：

```
aws iottwinmaker list-metadata-transfer-job --region your_region \
        --cli-input-json file://ListMetadataTransferJobsExample.json
```

- 要取消任务，请使用 CancelMetadataTransferJobAPI 操作。此 API 会取消特定的元数据传输作业，但不会影响任何已导出或导入的资源：

```
aws iottwinmaker cancel-metadata-transfer-job \
        --region your_region \
        --metadata-transfer-job-id job-to-cancel-id
```

检查错误 Amazon IoT SiteWise

检查错误详情（控制台）

Amazon IoT SiteWise 控制台中的错误详情：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 有关批量操作作业的列表， Amazon IoT SiteWise 控制台 请参阅中的作业进度表。
3. 选择一项作业以查看该作业的详细信息。
4. 如果作业的状态为 `COMPLETED` 或 `ERROR`，则 `Total resources` 计数等于详细计数（`Succeeded`、`Failed` 和 `Skipped`）的总和。
5. 如果作业的状态为 `ERROR`，请查看作业失败表，了解有关特定错误和失败的详细信息。
6. 作业失败表显示了作业报告中的内容。`Resource type` 字段表示错误或失败的位置，如下方所示：
   - 例如，`Resource type` 字段 `Bulk operations template` 中的验证错误表示导入模板和元数据架构文件格式不匹配。请参阅Amazon IoT SiteWise 元数据传输任务架构了解更多信息。
   - `Resource type` 字段中的失败 `Asset` 表示该资产不是因为与其他资产发生冲突而创建的。有关 Amazon IoT SiteWise 资源错误和冲突的信息，请参阅常见错误。

检查错误详情（Amazon CLI）

要处理和诊断传输作业期间产生的错误，请参阅以下有关使用 `GetMetadataTransferJob` API 操作的过程：

1. 创建并运行转接任务后，请致电 GetMetadataTransferJob：

```
aws iottwinmaker get-metadata-transfer-job \
        --metadata-transfer-job-id your_metadata_transfer_job_id \
        --region us-east-1
```

2. 看到作业状态变为 COMPLETED 后，您就可以开始验证作业的结果了。
3. 调用 GetMetadataTransferJob 时，它会返回一个名为 MetadataTransferJobProgress 的对象。

   该 MetadataTransferJobProgress 对象包含以下参数：

   - failedCount：表示在传输过程中失败的资产数量。
   - skippedCount：表示在传输过程中跳过的资产数量。
   - succeededCount：表示在传输过程中成功的资产数量。
   - totalCount：表示传输过程中涉及的资产总数。
4. 此外，API 调用还会返回一个包含预签名 URL 的元素 reportUrl。如果您的传输作业有任何问题需要进一步调查，请访问此 URL。

导入元数据示例

本节介绍了如何通过一次批量导入操作创建元数据文件，以导入资产模型和资产。

批量导入示例

通过一次批量导入操作导入多个资产模型和资产。以下示例说明了如何创建元数据文件来完成此操作。

在此示例场景中，您拥有不同的工作场所，其工作单元中包含工业机器人。

此示例定义了两种资产模型：

- RobotModel1：此资产模型表示您在工作场所拥有的一种特定类型的机器人。机器人具有测量值属性 Temperature。
- WorkCell：此资产模型表示您的一个工作场所中的机器人集合。此资产模型定义了一个层次结构 robotHierarchyOEM1，用于表示工作单元包含机器人的关系。

此示例还定义了一些资产：

- WorkCell1：波士顿站点内的一个工作单元

- `RobotArm123456`：该工作单元中的一个机器人
- `RobotArm987654`：该工作单元中的另一个机器人

以下 JSON 元数据文件定义了这些资产模型和资产。使用此元数据进行批量导入会在其中创建资产模型和资产 Amazon IoT SiteWise，包括它们的层次关系。

用于导入的元数据文件

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

模型和资产的初始载入示例

在此示例场景中，您拥有不同的工作场所，包含一家公司的工业机器人。

此示例定义了多个资产模型：

- `Sample_Enterprise` — 此资产模型表示这些站点所属的公司。此资产模型定义了一个层次结构 `Enterprise to Site`，用于表示站点与企业的关系。
- `Sample_Site` — 此资产模型表示公司内部的制造站点。此资产模型定义了一个层次结构 `Site to Line`，用于表示装配线与站点的关系。

- Sample_Welding Line — 此资产模型表示工作场所内的装配线。此资产模型定义了一个层次结构 Line to Robot，用于表示机器人与装配线的关系。
- Sample_Welding Robot — 此资产模型表示您的工作场所中的一种特定类型的机器人。

此示例还根据资产模型定义资产。

- Sample_AnyCompany Motor – 此资产是从 Sample_Enterprise 资产模型创建的。
- Sample_Chicago – 此资产是从 Sample_Site 资产模型创建的。
- Sample_Welding Line 1 – 此资产是从 Sample_Welding Line 资产模型创建的。
- Sample_Welding Robot 1 – 此资产是从 Sample_Welding Robot 资产模型创建的。
- Sample_Welding Robot 2 – 此资产是从 Sample_Welding Robot 资产模型创建的。

以下 JSON 元数据文件定义了这些资产模型和资产。使用此元数据进行批量导入会在其中创建资产模型和资产 Amazon IoT SiteWise，包括它们的层次关系。

用于载入要导入的资产和模型的 JSON 文件

```
{
    "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",
```

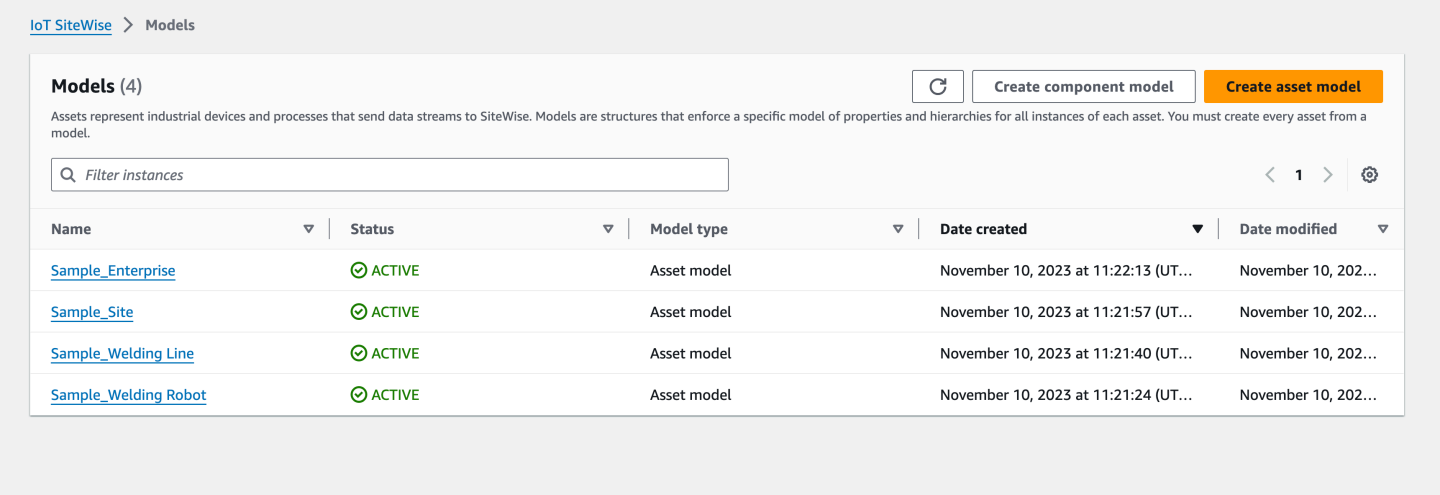
```
            "assetModelProperties": [
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Line_Availability",
                    "name": "Availability",
                    "type": {
                        "measurement": {}
                    },
                    "unit": "%"
                }
            ],
            "assetModelHierarchies": [
                {
                    "externalId": "External_Id_Welding_Line_TO_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_TO_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_TO_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"
                }
            ]
        },
```

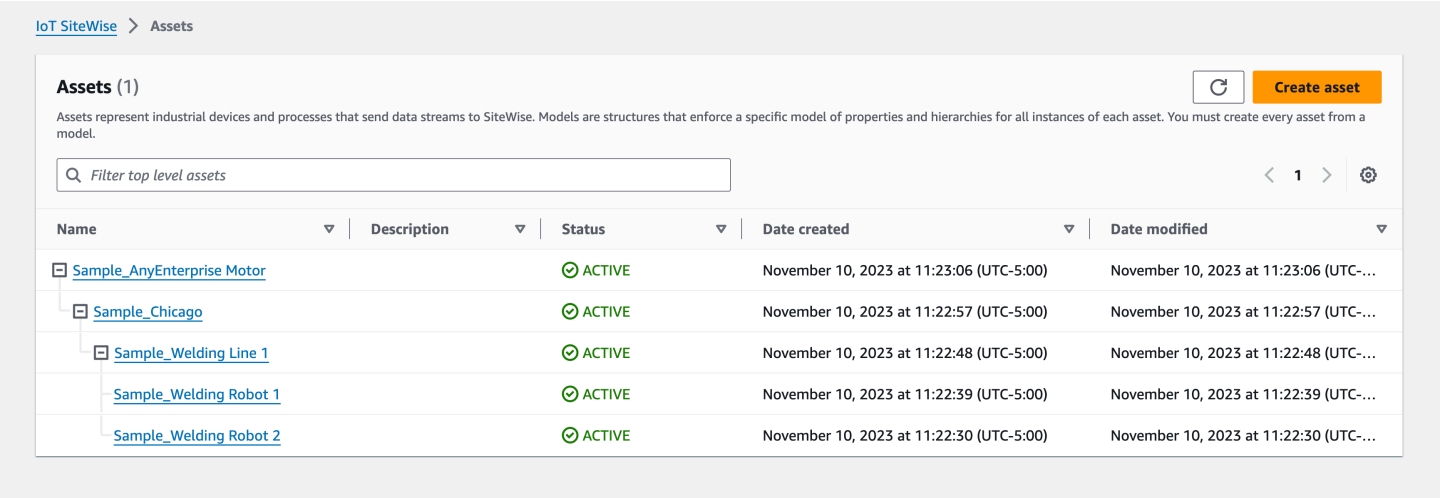
```
        {
            "assetExternalId": "External_Id_Welding_Robot_2",
            "assetName": "Sample_Welding Robot 2",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "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_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_1"
                },
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_2"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Site_Chicago",
            "assetName": "Sample_Chicago",
            "assetModelExternalId": "External_Id_Site",
```

```
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Site_TO_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_TO_Site",
                    "childAssetExternalId": "External_Id_Site_Chicago"
                }
            ]
        }
    ]
}
```

以下屏幕截图显示的是运行前面的代码示例 Amazon IoT SiteWise 控制台 之后显示的模型。



以下屏幕截图显示了运行上一个代码示例 Amazon IoT SiteWise 控制台 之后显示在中的模型、资产和层次结构。



载入额外资产的示例

此示例定义了要导入到您账户中现有资产模型的其他资产：

- `Sample_Welding Line 2` – 此资产是从 `Sample_Welding Line` 资产模型创建的。
- `Sample_Welding Robot 3` – 此资产是从 `Sample_Welding Robot` 资产模型创建的。
- `Sample_Welding Robot 4` – 此资产是从 `Sample_Welding Robot` 资产模型创建的。

要为此示例创建初始资产，请参阅模型和资产的初始载入示例。

以下 JSON 元数据文件定义了这些资产模型和资产。使用此元数据进行批量导入会在其中创建资产模型和资产 Amazon IoT SiteWise，包括它们的层次关系。

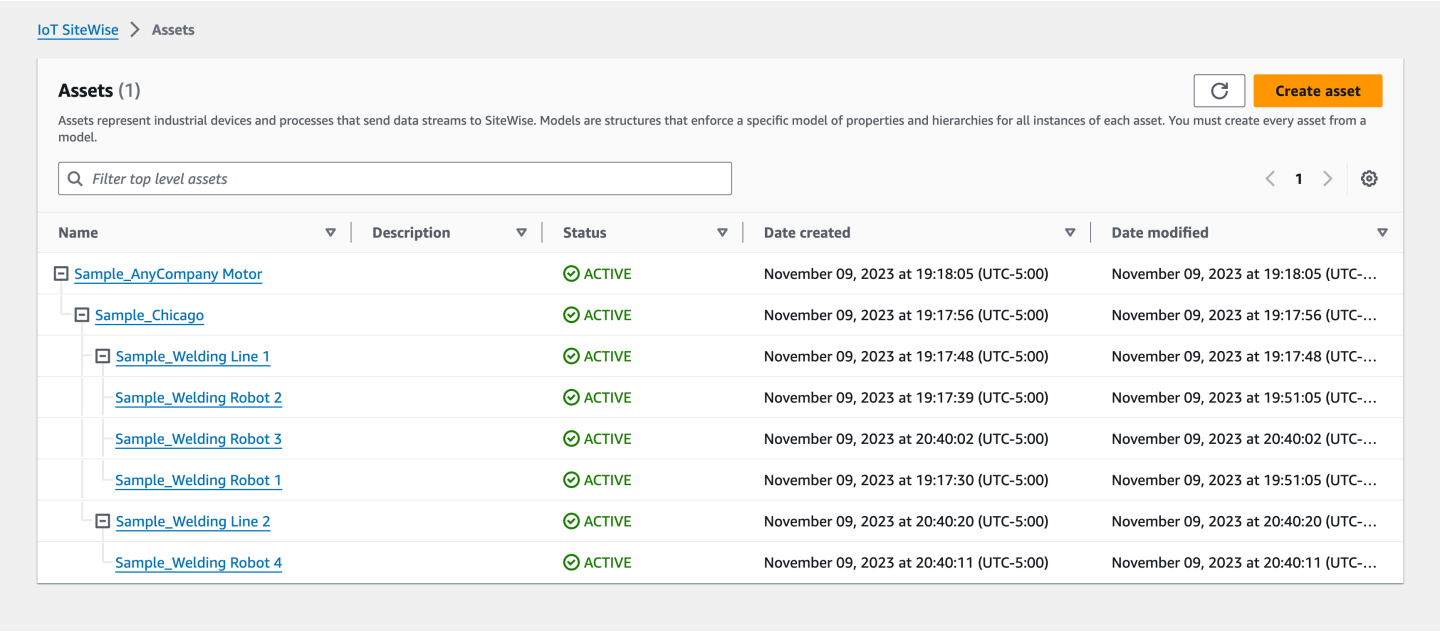
用于载入其他资产的 JSON 文件

```
{
    "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_TO_Robot",
                "childAssetExternalId": "External_Id_Welding_Robot_1"
            },
            {
                "externalId": "External_Id_Welding_Line_TO_Robot",
                "childAssetExternalId": "External_Id_Welding_Robot_2"
            },
            {
                "externalId": "External_Id_Welding_Line_TO_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_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_4"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Site_Chicago",
            "assetName": "Sample_Chicago",
            "assetModelExternalId": "External_Id_Site",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "childAssetExternalId": "External_Id_Welding_Line_1"
                },
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "childAssetExternalId": "External_Id_Welding_Line_2"
                }
            ]
        }
    ]
}
```

以下屏幕截图显示了运行上一个代码示例 Amazon IoT SiteWise 控制台 之后显示在中的模型、资产和层次结构。



载入新属性的示例

此示例定义了现有资产模型上的新属性。请参阅载入额外资产的示例，以载入其他资产和模型。

- `Joint 1 Temperature` – 此属性已添加到 `Sample_Welding Robot` 资产模型中。此新属性还将传播到从 `Sample_Welding Robot` 资产模型创建的每个资产。

要向现有资产模型添加新属性，请参阅以下 JSON 元数据文件示例。如 JSON 所示，必须提供新属性以及现有 `Sample_Welding Robot` 资产模型的完整定义。如果未提供现有定义中的整个属性列表，则 Amazon IoT SiteWise 删除省略的属性。

用于载入新属性的 JSON 文件

此示例向资产模型添加了一个新属性 `Joint 1 Temperature`。

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

管理数据流的示例

此示例介绍了管理与资产属性关联的数据流的两种方法。重命名资产属性别名时，对于当前存储在资产属性的数据流中的历史数据，有两个选项。

- 选项一 — 保留当前数据流并仅重命名别名，这会支持使用新别名访问历史数据。

  在 JSON 元数据文件示例中，带有 ID `External_Id_Welding_Robot_Cycle_Count` 的资产属性将其别名更改为 `AnyCompany/Chicago/Welding Line/S3000/Count-Updated`。该资产属性的历史数据不受此次更改的影响，将保持不变。
- 选项二 — 为资产属性分配新的数据流，该数据流可使用新别名进行访问。旧数据流及其历史数据仍可使用旧别名进行访问，但不与任何资产属性相关联。

  在 JSON 元数据文件示例中，带有 ID `External_Id_Welding_Robot_Joint_1_Current` 的资产属性将其别名更改为 `AnyCompany/Chicago/Welding Line/S4999/1/Current`。这次，存在其他值 `retainDataOnAliasChange` 并将其设置为 `False`。通过此设置，原始数据流将与资产属性断开关联，并会创建一个不包含历史数据的新数据流。

要使用原始历史数据访问旧数据流，请在中 Amazon Web Services Console Home，转到 “数据流” 页面并搜索旧别名`AnyCompany/Chicago/Welding Line/S3000/1/Current`。

用于更新属性别名的 JSON 文件

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

导出元数据示例

将 Amazon IoT SiteWise 内容批量导出到 Amazon S3 时，您可以指定筛选条件来限制要导出的特定资产模型和资产。

您在请求正文 `sources` 部分内的 `iotSiteWiseConfiguration` 部分中指定筛选条件。

> **Note**
>
> 您可以包含多个筛选条件。批量操作将导出与任何筛选条件匹配的任何资产模型或资产。如果您未提供任何筛选条件，则该操作将导出所有资产模型和资产。

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

按资产模型筛选

您可以筛选特定的资产模型。您还可以使用该模型的所有资产或其层次结构中的所有资产模型。不能同时包含资产和层次结构。

有关层次结构的更多信息，请参阅定义资产模型层次结构。

Asset model

此筛选条件包括指定的资产模型：

```
"filterByAssetModel": {
    "assetModelId": "asset model ID"
}
```

Asset model and its assets

此筛选条件包括指定的资产模型，以及使用该资产模型的所有资产：

```
"filterByAssetModel": {
    "assetModelId": "asset model ID",
    "includeAssets": true
}
```

Asset model and its hierarchy

此筛选条件包括指定的资产模型，以及其层次结构中的所有关联资产模型：

```
"filterByAssetModel": {
```

```
    "assetModelId": "asset model ID",
    "includeOffspring": true
}
```

按资产筛选

您可以筛选特定的资产。您还可以包含其资产模型或其层次结构中的所有关联资产。不能同时包含资产模型和层次结构。

有关层次结构的更多信息，请参阅定义资产模型层次结构。

Asset

此筛选条件包括指定的资产：

```
"filterByAsset": {
    "assetId": "asset ID"
}
```

Asset and its asset model

此筛选条件包括指定的资产，以及其使用的资产模型：

```
"filterByAsset": {
    "assetId": "asset ID",
    "includeAssetModel": true
}
```

Asset and its hierarchy

此筛选条件包括指定的资产，以及其层次结构中的所有关联资产：

```
"filterByAsset": {
    "assetId": "asset ID",
    "includeOffspring": true
}
```

Amazon IoT SiteWise 元数据传输任务架构

执行自己的批量导入和导出操作时，请使用 Amazon IoT SiteWise 元数据传输任务架构作为参考：

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "IoTSiteWise",
  "description": "Metadata transfer job resource schema for IoTSiteWise",
  "definitions": {
    "Name": {
      "type": "string",
      "minLength": 1,
      "maxLength": 256,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "Description": {
      "type": "string",
      "minLength": 1,
      "maxLength": 2048,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "ID": {
      "type": "string",
      "minLength": 36,
      "maxLength": 36,
      "pattern": "^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$"
    },
    "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"
      }
    }
  }
```

```
}
```

使用警报监控数据 Amazon IoT SiteWise

您可以为您的数据配置警报，以便在设备或流程运行不佳时提醒您的团队。计算机或进程的最佳性能意味着特定指标的值应在上限和下限的范围内。当这些指标超出其运作范围时，必须通知设备操作员以便修复问题。使用警报功能快速识别问题并通知操作员，以最大限度地提高设备和进程的性能。

主题

- 警报类型
- 警报状态
- 警报状态属性
- 在中定义资产模型的警报 Amazon IoT SiteWise
- 在中为资产配置警报 Amazon IoT SiteWise
- 回应中的警报 Amazon IoT SiteWise
- 摄取外部警报状态 Amazon IoT SiteWise

警报类型

您可以定义在 Amazon 云端检测到的警报和通过外部进程检测到的警报。 Amazon IoT SiteWise 支持以下类型的警报：

- Amazon IoT Events 警报

  Amazon IoT Events 警报是检测到的警报 Amazon IoT Events。 Amazon IoT SiteWise 将资产属性值发送到中的警报模型 Amazon IoT Events。然后， Amazon IoT Events 将警报状态发送到 Amazon IoT SiteWise。您可以配置选项，例如警报何时检测到以及在警报状态发生变化时通知谁。您还可以定义警报状态发生变化时发生的Amazon IoT Events 操作。

  中的警报 Amazon IoT Events 是警报模型的实例。警报模型指定了警报的阈值和严重性、警报状态发生变化时该怎么做等等。配置警报模型的每个特征时，您可以指定警报监控的资产模型的特征属性。所有基于资产模型的资产在 Amazon IoT Events 评估警报的特征时都使用该属性的值。有关更多信息，请参阅《开发人员指南》中的Amazon IoT Events 使用警报。

  当 Amazon IoT Events 警报状态发生变化时，您可以对其做出响应。例如，您可以在警报激活时确认或暂停警报。您还可以启用、禁用和重置警报。

SiteWise 监控用户可以在监控器门户中对 Amazon IoT Events 警报进行可 SiteWise 视化、配置和响应。有关更多信息，请参阅《Amazon IoT SiteWise Monitor 应用程序指南》中的使用警报进行监控。

> **Note**
>
> Amazon IoT Events 评估这些警报并在 Amazon IoT SiteWise 和之间传输数据需要付费 Amazon IoT Events。有关更多信息，请参阅Amazon IoT Events 定价。

- 外部警报

  外部警报是您在外部评估的警报 Amazon IoT SiteWise。如果您有报告警报状态的数据来源，请使用外部警报。外部警报包含一个测量属性，您可以从中提取警报状态数据。

  当外部警报改变状态时，您无法确认或暂停该警报。

  SiteWise 监控用户可以在 SiteWise 监控器门户中看到外部警报的状态，但他们无法配置或响应这些警报。

  Amazon IoT SiteWise 不评估外部警报的状态。

警报状态

工业警报包括有关其监控的设备或流程状态的信息，以及有关操作员对警报状态的响应的信息（可选）。

定义 Amazon IoT Events 警报时，您可以指定是否启用确认流。默认情况下，启用确认流。启用此选项后，操作员可以确认警报并留下备注，详细说明警报或其为解决警报采取的措施。如果操作员在活动警报变为非活动状态之前未确认该活动警报，则该警报将被锁定。锁定状态表示警报已激活且未被确认，因此操作员需要检查设备或流程，并确认已锁定的警报。

警报具有以下状态：

- 正常 (`Normal`) - 警报已启用但处于非活动状态。工业流程或设备按预期运行。
- 活动 (`Active`) - 警报处于活动状态。工业流程或设备超出了其操作范围，需要引起注意。
- 已确认 (`Acknowledged`) - 操作员已确认警报的状态。

  此状态仅适用于您启用确认流的警报。

- 已锁定 (Latched) - 警报恢复正常，但处于活动状态，没有操作员确认。工业流程或设备需要操作员注意才能将警报重置为正常。

  此状态仅适用于您启用确认流的警报。
- 已暂停 (SnoozeDisabled) - 警报被停用，因为操作员暂停了警报。操作员定义警报暂停的持续时间。在该持续时间之后，警报将恢复正常状态。
- 已禁用 (Disabled) - 警报已禁用，无法检测。

警报状态属性

Amazon IoT SiteWise 将警报状态数据存储为序列化为字符串的 JSON 对象。此对象包含警报的状态和其他信息，例如操作员的响应操作和警报评估的规则。

您可以通过名称和结构类型 AWS/ALARM_STATE 来识别警报状态属性。有关更多信息，请参阅 在中定义资产模型的警报 Amazon IoT SiteWise。

警报状态数据对象包含以下信息：

stateName

警报的状态。有关更多信息，请参阅 警报状态。

数据类型：STRING

customerAction

（可选）包含有关操作员响应警报信息的对象。操作员可以启用、禁用、确认和暂停警报。当他们这样做时，警报状态数据包括他们的响应以及他们响应时可以留下的备注。该对象包含以下信息：

actionName

操作员为响应警报而采取的操作名称。此值包含以下字符串之一：

- ENABLE
- DISABLE
- SNOOZE
- ACKNOWLEDGE
- RESET

数据类型：STRING

enable

（可选）操作员启用警报时 customerAction 中存在的对象。当操作员启用警报时，警报状态将变为 Normal。该对象包含以下信息：

note

（可选）客户在启用警报时留下的备注。

数据类型：STRING

最大长度：128 个字符

disable

（可选）操作员禁用警报时 customerAction 中存在的对象。当操作员启用警报时，警报状态将变为 Disabled。该对象包含以下信息：

note

（可选）客户在禁用警报时留下的备注。

数据类型：STRING

最大长度：128 个字符

acknowledge

（可选）操作员确认警报时 customerAction 中存在的对象。当操作员启用警报时，警报状态将变为 Acknowledged。该对象包含以下信息：

note

（可选）客户在确认警报时留下的备注。

数据类型：STRING

最大长度：128 个字符

snooze

（可选）操作员暂停警报时 customerAction 中存在的对象。当操作员启用警报时，警报状态将变为 SnoozeDisabled。该对象包含以下信息：

snoozeDuration

操作员暂停警报的持续时间（以秒为单位）。在此持续时间之后，警报将变为 Normal 状态。

数据类型：INTEGER

note

（可选）客户在暂停警报时留下的备注。

数据类型：STRING

最大长度：128 个字符

ruleEvaluation

（可选）包含有关警报评估规则信息的对象。该对象包含以下信息：

simpleRule

包含有关简单规则信息的对象，该规则使用比较运算符将属性值与阈值进行比较。该对象包含以下信息：

inputProperty

此警报评估属性的值。

数据类型：DOUBLE

operator

此警报用于将属性与阈值进行比较的比较运算符。此值包含以下字符串之一：

- < - 小于
- <= - 小于或等于
- == - 等于
- != - 不等于
- >= - 大于或等于
- > - 大于

数据类型：STRING

threshold

此警报与属性值进行比较的阈值。

数据类型：DOUBLE

在中定义资产模型的警报 Amazon IoT SiteWise

资产模型可推动您的工业数据和警报实现标准化。您可以在资产模型上规定警报定义，以标准化基于资产模型的所有资产的警报。

您可以使用组合资产模型来定义资产模型上的警报。复合资产模型是在另一种资产模型上标准化一组特定属性的资产模型。复合资产模型可确保资产模型上存在某些属性。警报具有类型、状态和（可选）源属性，因此警报复合模型强制这些属性存在。

每个复合模型都有一个类型，用于定义复合模型的属性。警报复合模型定义警报类型、警报状态和（可选）警报源的属性。当您使用复合模型从资产模型创建资产时，该资产将包括复合模型中的属性以及您在资产模型中指定的属性。

复合模型中的每个属性都必须具有用于标识其复合模型类型的名称。复合模型属性支持具有复杂数据类型的属性。这些属性具有 `STRUCT` 数据类型和 `dataTypeSpec` 特征，用于指定属性的复杂数据类型。复杂数据类型属性包含序列化为字符串的 JSON 数据。

警报复合模型具有以下属性。对于此类复合模型，每个属性都必须具有标识该属性的名称。

警报类型

警报的类型。指定下列项之一：

- `IOT_EVENTS`— Amazon IoT Events 警报。 Amazon IoT SiteWise 向发送数据 Amazon IoT Events 以评估此警报的状态。必须指定警报源属性才能定义此 Amazon IoT Events 警报定义的警报模型。
- `EXTERNAL` – 外部警报。您摄取警报状态作为测量值。

属性名称：`AWS/ALARM_TYPE`

属性类型：属性

数据类型：`STRING`

警报状态

警报状态的时间序列数据。这是一个序列化为字符串的对象，其中包含警报的状态和其他信息。有关更多信息，请参阅 警报状态属性。

属性名称：`AWS/ALARM_STATE`

属性类型：测量值

数据类型：`STRUCT`

数据结构类型：AWS/ALARM_STATE

警报源

（可选）评估警报状态的资源的 Amazon 资源名称（ARN）。对于 Amazon IoT Events 警报，这是警报模型的 ARN。

属性名称：AWS/ALARM_SOURCE

属性类型：属性

数据类型：STRING

Example 警报复合模型示例

以下资产模型表示带有警报以监控其温度的锅炉。 Amazon IoT SiteWise 将温度数据发送到 Amazon IoT Events 以检测警报。

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

主题

- 为其定义 Amazon IoT Events 警报 Amazon IoT SiteWise
- 在中定义外部警报 Amazon IoT SiteWise

为其定义 Amazon IoT Events 警报 Amazon IoT SiteWise

创建 Amazon IoT Events 警报时， Amazon IoT SiteWise 会将资产属性值发送 Amazon IoT Events 到以评估警报的状态。 Amazon IoT Events 警报定义取决于您在中定义的警报模型 Amazon IoT

Events。要在资产模型上定义 Amazon IoT Events 警报，您需要定义一个警报组合模型，该模型将 Amazon IoT Events 警报模型指定为其警报源属性。

Amazon IoT Events 警报取决于警报阈值和警报通知设置等输入。您可以将这些输入定义为资产模型的属性。然后，您可以根据模型在每个资产上自定义这些输入。 Amazon IoT SiteWise 控制台可以为您创建这些属性。如果您使用 Amazon CLI 或 API 定义警报，则必须在资产模型上手动定义这些属性。

您还可以定义检测到警报时发生的其他操作，例如自定义警报通知操作。例如，您可以配置向 Amazon SNS 主题发送推送通知的操作。有关您可以定义的操作的更多信息，请参阅《Amazon IoT Events 开发人员指南》中的使用其他 Amazon 服务。

更新或删除资产模型时， Amazon IoT SiteWise 可以检查中的警报模型是否 Amazon IoT Events 正在监视与该资产模型关联的资产属性。这可以防止您删除 Amazon IoT Events 警报当前正在使用的资产属性。要在中启用此功能 Amazon IoT SiteWise，您必须拥有`iotevents:ListInputRoutings`权限。此权限 Amazon IoT SiteWise 允许调用所支持ListInputRoutings的 API 操作 Amazon IoT Events。有关更多信息，请参阅 （可选）ListInputRoutings 权限。

> **Note**
>
> 警报通知功能在中国（北京）区域中不可用。

主题

- 定义 Amazon IoT Events 警报（Amazon IoT SiteWise 控制台）
- 定义 Amazon IoT Events 警报（Amazon IoT Events 控制台）
- 定义 Amazon IoT Events 警报 (Amazon CLI)

定义 Amazon IoT Events 警报（Amazon IoT SiteWise 控制台）

您可以使用 Amazon IoT SiteWise 控制台在现有资产模型上定义 Amazon IoT Events 警报。要在新资产模型上定义 Amazon IoT Events 警报，请创建资产模型，然后完成以下步骤。有关更多信息，请参阅 在中创建资产模型 Amazon IoT SiteWise。

> **Important**
>
> 每个警报都需要一个属性，来指定要与警报进行比较的阈值。必须先在资产模型上定义阈值属性，然后才能定义警报。

举例说明，如果您想定义一个警报，用于检测风力涡轮机何时超过其最大额定风速 50 英里/小时。在定义警报之前，您必须定义一个默认值为 50 的属性（最大风速）。

在资产模型上定义 Amazon IoT Events 警报

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择要为其定义警报的资产模型。
4. 选择警报选项卡。
5. 选择添加警报。
6. 在警报类型选项部分，选择Amazon IoT Events 警报。
7. 在基本详细信息部分中，执行以下操作：
   a. 输入警报的名称。
   b. （可选）输入警报的描述。
8. 在阈值定义部分，您可以定义警报何时检测以及警报的严重程度。执行以下操作：
   a. 选择警报检测的属性。每次此属性收到新值时，都会 Amazon IoT SiteWise 将该值发送 Amazon IoT Events 到以评估警报的状态。
   b. 选择用于将属性与阈值进行比较的运算符。从以下选项中进行选择：
      - 小于
      - 小于或等于
      - = (等于)
      - != 不等于
      - 大于或等于
      - 大于
   c. 对于 “值”，选择要用作阈值的属性属性。 Amazon IoT Events 将属性的值与该属性的值进行比较。
   d. 输入警报的严重程度。使用您的团队能理解的数字来反映此警报的严重程度。
9. 在默认资产状态部分，您可以设置从此资产模型创建的警报的默认状态。

> **Note**
>
> 您可以在后续步骤中为从此资产模型创建的资产激活或停用此警报。

10. 在高级设置部分，您可以配置权限、其他通知设置、警报状态操作、 SiteWise 监控器中的警报模式以及确认流。

> **Note**
>
> Amazon IoT Events 警报需要以下服务角色：
>
> - Amazon IoT Events 假定向发送警报状态值的角色 Amazon IoT SiteWise。

在附加权限策略页面上，执行以下操作：

- 对于 Amazon IoT Events 角色，请使用现有角色或创建具有所需权限的角色。此角色需要允许 iotevents.amazonaws.com 担任角色的 `iotsitewise:BatchPutAssetPropertyValue` 许可和信任关系。

11. （可选）在设置状态操作部分，执行以下操作：

    a. 选择编辑操作。

    b. 在添加警报状态操作下，添加操作。然后选择保存。

    您最多可以添加 10 个操作。

    Amazon IoT Events 可以在警报处于活动状态时执行操作。您可以定义内置操作以使用计时器或设置变量，或者将数据发送到其他 Amazon 资源。有关更多信息，请参阅 Amazon IoT Events 开发人员指南 中的 支持的操作。

12. （可选）在 “ SiteWise 监控器-可选” 中的 “管理警报模式” 下，选择 “活动” 或 “非活动”。

    使用此选项可以在 M SiteWise onitorss 中更新警报模型。默认情况下，此选项处于启用状态。

13. 在确认流下，选择活动或非活动。有关确认流的更多信息，请参阅 警报状态。

14. 选择添加警报。

> **Note**
>
> Amazon IoT SiteWise 控制台发出多个 API 请求，将警报添加到资产模型中。当您选择添加警报时，控制台会打开一个对话框，显示这些 API 请求的进度。在每个 API 请求成功或一个 API 请求失败之前，请保持停留在此页面。如果请求失败，请关闭对话框，修复问题，然后选择添加警报重试。

定义 Amazon IoT Events 警报（Amazon IoT Events 控制台）

您可以使用 Amazon IoT Events 控制台在现有资产模型上定义 Amazon IoT Events 警报。要在新资产模型上定义 Amazon IoT Events 警报，请创建资产模型，然后完成以下步骤。有关更多信息，请参阅在中创建资产模型 Amazon IoT SiteWise。

> **Important**
>
> 每个警报都需要一个属性，来指定要与警报进行比较的阈值。必须先在资产模型上定义阈值属性，然后才能定义警报。
>
> 举例说明，如果您想定义一个警报，用于检测风力涡轮机何时超过其最大额定风速 50 英里/小时。在定义警报之前，您必须定义一个默认值为 50 的属性（最大风速）。

在资产模型上定义 Amazon IoT Events 警报

1. 导航至 Amazon IoT Events 控制台。
2. 在导航窗格中，选择警报模型。
3. 选择创建警报模型。
4. 输入警报的名称。
5. （可选）输入警报的描述。
6. 在 警报目标 部分，执行以下操作：
   a. 对于目标选项，选择Amazon IoT SiteWise 资产属性。
   b. 选择要为其添加警报的资产模型。
7. 在阈值定义部分，您可以定义警报何时检测以及警报的严重程度。执行以下操作：

a. 选择警报检测的属性。每次此属性收到新值时，都会 Amazon IoT SiteWise 将该值发送 Amazon IoT Events 到以评估警报的状态。

b. 选择用于将属性与阈值进行比较的运算符。从以下选项中进行选择：

- 小于
- 小于或等于
- = (等于)
- != 不等于
- 大于或等于
- 大于

c. 对于 “值”，选择要用作阈值的属性属性。 Amazon IoT Events 将属性的值与该属性的值进行比较。

d. 输入警报的严重程度。使用您的团队能理解的数字来反映此警报的严重程度。

8. 在实例部分，指定此警报的默认状态。您可以在稍后的步骤中为通过该资产模型创建的所有资产激活或停用此警报。

9. 在高级设置设置中，您可以配置权限、其他通知设置、警报状态操作、 SiteWise 监控器中的警报模式以及确认流。

> **Note**
>
> Amazon IoT Events 警报需要以下服务角色：
>
> - Amazon IoT Events 假定向发送警报状态值的角色 Amazon IoT SiteWise。

a. 在确认流部分，选择已启用或已禁用。有关确认流的更多信息，请参阅 警报状态。

b. 在附加权限策略页面上，执行以下操作：

- 对于 Amazon IoT Events 角色，请使用现有角色或创建具有所需权限的角色。此角色需要允许 iotevents.amazonaws.com 担任角色的 `iotsitewise:BatchPutAssetPropertyValue` 许可和信任关系。

c. （可选）在设置状态操作 - 可选部分中，执行以下操作：

- 在警报状态操作下，添加操作，然后选择保存。

您最多可以添加 10 个操作。

Amazon IoT Events 可以在警报处于活动状态时执行操作。您可以定义内置操作以使用计时器或设置变量，或者将数据发送到其他 Amazon 资源。有关更多信息，请参阅 Amazon IoT Events 开发人员指南 中的 支持的操作。

10. 选择创建。

> **Note**
>
> Amazon IoT Events 控制台发出多个 API 请求，将警报添加到资产模型中。当您选择添加警报时，控制台会打开一个对话框，显示这些 API 请求的进度。在每个 API 请求成功或一个 API 请求失败之前，请保持停留在此页面。如果请求失败，请关闭对话框，修复问题，然后选择添加警报重试。

定义 Amazon IoT Events 警报 (Amazon CLI)

您可以使用 Amazon Command Line Interface (Amazon CLI) 来定义监控资产属性的 Amazon IoT Events 警报。您可以在新的或现有的资产模型上定义警报。在资产模型上定义警报后，可以在中创建警报 Amazon IoT Events 并将其连接到资产模型。在此过程中，您将执行以下操作：

步数

- 步骤 1：为资产模型定义警报
- 步骤 2：定义 Amazon IoT Events 警报模型
- 步骤 3：启用 Amazon IoT SiteWise 和之间的数据流 Amazon IoT Events

步骤 1：为资产模型定义警报

向新的或现有的资产模型添加警报定义和关联属性。

在资产模型上定义警报 (CLI)

1. 创建名为 `asset-model-payload.json` 的文件。按照其他部分中的步骤将资产模型的详细信息添加到文件中，但不要提交创建或更新资产模型的请求。在本节中，您将向 `asset-model-payload.json` 文件中的资产模型详细信息添加警报定义。
   - 有关如何创建资产模型的更多信息，请参阅创建资产模型（Amazon CLI）。

- 有关如何更新现有资产模型的更多信息，请参阅 更新资产或组件模型（Amazon CLI）。

> **Note**
>
> 您的资产模型必须定义至少一个资产属性，包括要通过警报监控的资产属性。

2. 向资产模型添加警报复合模型 (`assetModelCompositeModels`)。 Amazon IoT Events 警报组合模型指定警报`IOT_EVENTS`类型并指定警报源属性。在中创建警报模型后，您可以添加警报源属性 Amazon IoT Events。

> **Important**
>
> 警报组合模型必须与您稍后创建的 Amazon IoT Events 警报模型同名。警报模型名称只能包含字母数字字符。指定一个唯一的字母数字名称，这样您就可以对警报模型使用相同的名称。

```
{
  ...
  "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. 向资产模型添加警报阈值属性。指定用于此阈值的默认值。您可以基于此模型覆盖每个资产的此默认值。

> **Note**
>
> 警报阈值属性必须为 INTEGER 或 DOUBLE。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "Temperature Max Threshold",
      "dataType": "DOUBLE",
      "type": {
        "attribute": {
          "defaultValue": "105.0"
        }
      }
    }
  ]
}
```

4. 创建资产模型或更新现有资产模型。请执行以下操作之一：
   - 要创建资源模型，请运行以下命令。

     ```
     aws iotsitewise create-asset-model --cli-input-json file://asset-model-
     payload.json
     ```

   - 要更新现有资源模型，请运行以下命令。*asset-model-id*替换为资产模型的 ID。

     ```
     aws iotsitewise update-asset-model \
       --asset-model-id asset-model-id \
       --cli-input-json file://asset-model-payload.json
     ```

运行命令之后，请记下响应中的 assetModelId。

示例：锅炉资产模型

以下资产模型表示报告温度数据的锅炉。该资产模型定义了一个警报，用于检测锅炉何时过热。

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

步骤 2：定义 Amazon IoT Events 警报模型

在中创建警报模型 Amazon IoT Events。在中 Amazon IoT Events，您可以使用表达式来指定警报模型中的值。您可以使用表达式来指定 Amazon IoT SiteWise 要评估的值，并将其用作警报的输入。向警报模型 Amazon IoT SiteWise 发送资产属性值时， Amazon IoT Events 会评估表达式以获取该属性的值或资产的 ID。您可以在警报模型中使用以下表达式：

- 资产属性值

  要获取资产属性的值，请使用以下表达式。*assetModelId*替换为资产模型的 ID，*propertyId*替换为属性的 ID。

  ```
  $sitewise.assetModel.`assetModelId`.`propertyId`.propertyValue.value
  ```

- 资产 IDs

要获取资产的 ID，请使用以下表达式。*assetModelId*替换为资产模型的 ID，*propertyId*替换为属性的 ID。

```
$sitewise.assetModel.`assetModelId`.`propertyId`.assetId
```

> **Note**
>
> 创建警报模型时，您可以定义字面值，而不是计算结果为 Amazon IoT SiteWise 值的表达式。这可以减少您在资产模型上定义的属性数量。但是，如果您将值定义为文字，则无法根据资产模型自定义资产的该值。您的 Amazon IoT SiteWise Monitor 用户也无法自定义警报，因为他们只能在资产上配置警报设置。

创建 Amazon IoT Events 警报模型 (CLI)

1. 在中创建警报模型时 Amazon IoT Events，必须指定警报使用的每个属性的 ID，其中包括以下内容：

   - 复合资产模型中的警报状态属性
   - 警报监控的属性
   - 阈值属性
   - （可选）IAM Identity Center 身份存储 ID 属性
   - （可选）IAM Identity Center 用户 ID 属性
   - （可选）短信发送人 ID 属性
   - （可选）电子邮件发送人地址属性
   - （可选）电子邮件主题属性
   - （可选）附加消息属性

   运行以下命令以检索资产模型上的这些属性。 IDs *asset-model-id*替换为上一步中的资产模型的 ID。

   ```
   aws iotsitewise describe-asset-model --asset-model-id asset-model-id
   ```

此操作将返回包含资产模型详细信息的响应。记下警报使用的每个属性的 ID。您可以在下一步创建 Amazon IoT Events 警报模型 IDs 时使用它们。

2. 在中创建警报模型 Amazon IoT Events。执行以下操作：

   a. 创建名为 `alarm-model-payload.json` 的文件。

   b. 将以下 JSON 对象复制到文件中。

   c. 为您的警报输入名称（`alarmModelName`）、说明（`alarmModelDescription`）和严重性（`severity`）。对于严重性，请指定一个反映贵公司严重性级别的整数。

   > **Important**
   >
   > 警报模型必须与您之前在资产模型上定义的警报复合模型同名。
   > 警报模型名称只能包含字母数字字符。

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3
}
```

   d. 将比较规则 (`alarmRule`) 添加到警报中。此规则定义了要监控的属性 (`inputProperty`)、要比较的阈值 (`threshold`) 和要使用的比较运算符 (`comparisonOperator`)。

   - *`assetModelId`*替换为资产模型的 ID。
   - *`alarmPropertyId`*替换为警报监控的属性的 ID。
   - *`thresholdAttributeId`*替换为阈值属性的 ID。
   - 替换*`GREATER`*为用于将属性值与阈值进行比较的运算符。从以下选项中进行选择：
     - `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. 添加一个操作 (`alarmEventActions`)，当警报状态发生更改时将警报状态发送给 Amazon IoT SiteWise 。

> **Note**
>
> 对于高级配置，您可以定义警报状态更改时要执行的其他操作。例如，您可以调用 Amazon Lambda 函数或发布到 MQTT 主题。有关更多信息，请参阅《Amazon IoT Events 开发人员指南》中的使用其他 Amazon 服务。

- *`assetModelId`*替换为资产模型的 ID。
- *`alarmPropertyId`*替换为警报监控的属性的 ID。
- *`alarmStatePropertyId`*替换为警报复合模型中警报状态属性的 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. （可选）将警报功能 (alarmCapabilities) 添加到 alarm-model-payload.json 的有效负载中。在此对象中，您可以根据资产模型指定是否启用确认流程以及资产的默认启用状态。有关确认流的更多信息，请参阅 警报状态。

```
{
  "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.valu
          }
        ],
        "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.valu
            }
          }
        ]
      }
    ]
  },
  "alarmCapabilities": {
    "initializationConfiguration": {
      "disabledOnInitialization": false
    },
    "acknowledgeFlow": {
      "enabled": true
    }
  }
}
```

g. 添加 Amazon IoT Events 可以假设向其发送数据的 IAM 服务角色 (`roleArn`) Amazon IoT SiteWise。此角色需要 `iotsitewise:BatchPutAssetPropertyValue` 权限以及允许 `iotevents.amazonaws.com` 担任角色的信任关系。有关更多信息，请参阅 Amazon IoT Events 开发人员指南 中的 警报服务角色。

- 将`roleArn`替换为 Amazon IoT Events 可以代入执行这些操作的角色的 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":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
          }
        ],
        "emailConfigurations": [
          {
            "from":
"$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.valu
            }
          }
        ]
      }
    ]
  },
  "alarmCapabilities": {
    "initializationConfiguration": {
      "disabledOnInitialization": false
    },
    "acknowledgeFlow": {
      "enabled": false
    }
  },
  "roleArn": "arn:aws:iam::123456789012:role/MyIoTEventsAlarmRole"
}
```

h. 运行以下命令从中的有效负载创建 Amazon IoT Events 警报模型`alarm-model-payload.json`。

```
aws iotevents create-alarm-model --cli-input-json file://alarm-model-
payload.json
```

i. 操作将返回响应，其中包含警报模型的 ARN，`alarmModelArn`。复制此 ARN，以便下一步在资产模型的警报定义中进行设置。

步骤 3：启用 Amazon IoT SiteWise 和之间的数据流 Amazon IoT Events

在 Amazon IoT SiteWise 和中创建所需资源后 Amazon IoT Events，您可以启用资源之间的数据流以启用警报。在本节中，您将更新资产模型中的警报定义以使用您在上一步中创建的警报模型。

启用 Amazon IoT SiteWise 和之间的数据流 Amazon IoT Events (CLI)

- 在资产模型中将警报模型设置为警报的来源。执行以下操作：

  a. 运行以下命令以检索现有资产模型定义。*asset-model-id*替换为资产模型的 ID。

  ```
  aws iotsitewise describe-asset-model --asset-model-id asset-model-id
  ```

  此操作将返回包含资产模型详细信息的响应。

  b. 创建一个名为 `update-asset-model-payload.json` 的文件，并将上一命令的响应复制到该文件中。

  c. 从 `update-asset-model-payload.json` 文件中删除以下键/值对：

  - `assetModelId`
  - `assetModelArn`
  - `assetModelCreationDate`
  - `assetModelLastUpdateDate`
  - `assetModelStatus`

  d. 将警报源属性 (`AWS/ALARM_SOURCE`) 添加到您之前定义的警报复合模型中。*alarmModelArn*替换为警报模型的 ARN，后者设置警报源属性的值。

  ```
  {
    ...
    "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. 运行以下命令，使用存储在 `update-asset-model-payload.json` 文件中的定义更新资产模型。*`asset-model-id`*替换为资产模型的 ID。

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://update-asset-model-payload.json
```

现在，您的资产模型定义了一个在 Amazon IoT Events中检测到的警报。报警根据该资产模型对所有资产中的目标属性进行监控。您可以为每项资产配置警报，以自定义每项资产的阈值或 IAM Identity Center 接收人等属性。有关更多信息，请参阅 在中为资产配置警报 Amazon IoT SiteWise。

在中定义外部警报 Amazon IoT SiteWise

外部警报包含您在 Amazon IoT SiteWise外部检测到的警报状态。

定义外部警报（控制台）

您可以使用 Amazon IoT SiteWise 控制台在现有资产模型上定义外部警报。要在新资产模型上定义外部警报，请创建资产模型，然后完成以下步骤。有关更多信息，请参阅 在中创建资产模型 Amazon IoT SiteWise。

在资产模型上定义警报

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型。
3. 选择要为其定义警报的资产模型。
4. 选择警报定义选项卡。
5. 选择添加警报。
6. 在警报类型选项中，选择外部警报。
7. 输入警报的名称。
8. （可选）输入警报的描述。
9. 选择添加警报。

定义外部警报（CLI）

您可以使用在新资产模型或现有资产模型上定义外部警报。 Amazon CLI

要向资产模型添加外部警报，请向资产模型添加警报复合模型。外部警报复合模型指定 EXTERNAL 类型，但不指定警报源属性。以下复合警报示例定义了外部温度警报。

```
{
  ...
  "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": {}
          }
        }
      ]
    }
  ]
}
```

有关如何向新的或现有的资产模型添加复合模型的更多信息，请参阅以下内容：

- 创建资产模型（Amazon CLI）
- 更新资产或组件模型（Amazon CLI）

定义外部警报后，您可以根据资产模型将警报状态推送到资产。有关更多信息，请参阅 摄取外部警报状态 Amazon IoT SiteWise。

在中为资产配置警报 Amazon IoT SiteWise

在资产模型上定义 Amazon IoT Events 警报后，您可以根据资产模型为每项资产配置警报。您可以编辑警报的阈值和通知设置。其中每个值都是资产的一个属性，因此您可以更新属性的默认值来配置这些值。

> **Note**
>
> 您可以为 Amazon IoT Events 警报配置这些值，但不能为外部警报配置这些值。

主题

- 配置阈值（控制台）
- 配置阈值（Amazon CLI）

配置阈值（控制台）

您可以使用 Amazon IoT SiteWise 控制台更新用于指定警报阈值的属性的值。

更新警报阈值（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要更新警报阈值的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择编辑。
5. 找到警报用于其阈值的属性，然后输入其新值。
6. 选择保存。

配置阈值（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 更新指定警报阈值的属性的值。

您必须知道资产的 `assetId` 和属性的 `propertyId` 才能完成此过程。您还可以使用外部 ID。如果您创建了资产但不知道该资产`assetId`，请使用 ListAssetsAPI 列出特定模型的所有资产。使用该DescribeAsset操作查看资产的属性，包括财产 IDs。

使用 BatchPutAssetPropertyValue 操作将属性值分配给您的资产。您可以使用此操作一次设置多个属性。此操作的负载包含条目列表，每个条目都包含资产 ID、属性 ID 和属性值。

要更新属性值（Amazon CLI），请执行以下步骤：

1. 创建一个名为 `batch-put-payload.json` 的文件，并将以下 JSON 对象复制到该文件中。此示例负载演示如何设置风力涡轮机的纬度和经度。更新 IDs、值和时间戳以修改您的用例的有效负载。

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

- 负载中的每个条目都包含一个 `entryId`，您可以将其定义为任何唯一的字符串。如果任何请求条目失败，则每个错误都将包含相应请求的 `entryId`，以便您知道要重试的请求。
- 要设置属性值，可以在每个属性属性的列表中包含一个 timestamp-quality-value (TQV) 结构。`propertyValues`此结构必须包含新的 `value` 和当前的 `timestamp`。
  - `value` – 包含下列字段之一的结构，具体取决于要设置的属性的类型：
    - `booleanValue`
    - `doubleValue`
    - `integerValue`
    - `stringValue`

- `nullValue`
- `timestamp`— 一种包含当前 Unix 纪元时间（以秒为单位）的结构，`timeInSeconds`。Amazon IoT SiteWise 拒绝任何时间戳在过去存在超过 7 天或将来超过 5 分钟的数据点。

有关如何为 BatchPutAssetPropertyValue 准备负载的更多信息，请查阅使用摄取数据 Amazon IoT SiteWise APIs。

2. 运行以下命令以将属性值发送到 Amazon IoT SiteWise：

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

回应中的警报 Amazon IoT SiteWise

当 Amazon IoT Events 警报状态发生变化时，您可以执行以下操作来响应警报：

- 确认警报以表明您正在处理问题。
- 暂停警报以将其暂时停用。
- 禁用警报可将其永久禁用，直到您再次启用警报。
- 启用已禁用的警报以检测警报状态。
- 重置警报以清除其状态和最新值。

您可以使用 Amazon IoT SiteWise 控制台或 Amazon IoT Events API 来响应警报。

> **Note**
> 您可以响应 Amazon IoT Events 警报，但不能响应外部警报。

主题

- 响应警报（控制台）
- 响应警报（API）

响应警报（控制台）

您可以使用 Amazon IoT SiteWise 控制台确认、暂停、禁用或启用警报。

主题

- 确认警报（控制台）
- 暂停警报（控制台）
- 禁用警报（控制台）
- 启用警报（控制台）
- 重置警报（控制台）

确认警报（控制台）

您可以确认警报以表明您正在处理问题。

> **Note**
> 您必须在警报上启用确认流，这样才能确认警报。如果您从 Amazon IoT SiteWise 控制台定义警报，则默认情况下会启用此选项。

确认警报（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要确认警报的资产。

   > **Tip**
   > 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择 警报 选项卡。
5. 选择要确认的警报，然后选择 操作 以打开响应操作菜单。
6. 选择确认。警报的状态更改为 已确认。

暂停警报（控制台）

您可以暂停警报以将其暂时停用。指定暂停警报的持续时间。

对警报设置稍后提醒 (控制台)

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要对警报设置稍后提醒的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择 警报 选项卡。
5. 选择要设置稍后提醒的警报，然后选择操作以打开响应操作菜单。
6. 选择暂停。将打开一个模型，您可以在其中指定暂停的持续时间。
7. 选择 暂停时长或输入 自定义暂停时长。
8. 选择保存。警报的状态更改为 已暂停。

禁用警报（控制台）

您可以禁用警报，使其不再检测。禁用警报后，如果您希望警报检测，则必须重新启用警报。

禁用警报（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要禁用其警报的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择 警报 选项卡。
5. 选择要禁用的警报，然后选择操作以打开响应操作菜单。

6. 选择 禁用。警报的状态更改为 已禁用。

启用警报（控制台）

您可以启用警报，使其在禁用或暂停后再次检测。

若要启用警报（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要启用警报的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择 警报 选项卡。
5. 选择要启用的警报，然后选择操作以打开响应操作菜单。
6. 请选择 启用。警报的状态更改为 正常。

重置警报（控制台）

您可以重置警报以清除其状态和最新值。

若要重置警报 (控制台)

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要重置警报的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择 警报 选项卡。
5. 选择要启用的警报，然后选择操作以打开响应操作菜单。
6. 选择 重置。警报的状态更改为 正常。

响应警报（API）

您可以使用 Amazon IoT Events API 来确认、暂停、禁用、启用或重置警报。有关更多信息，请参阅 Amazon IoT Events API 参考 中的以下操作：

- BatchAcknowledgeAlarm
- BatchSnoozeAlarm
- BatchDisableAlarm
- BatchEnableAlarm
- BatchResetAlarm

有关更多信息，请参阅 Amazon IoT Events 开发人员指南 中的 响应警报。

摄取外部警报状态 Amazon IoT SiteWise

外部警报是您在外部评估的警报 Amazon IoT SiteWise。当您有一个数据来源报告您想要获取到 Amazon IoT SiteWise的警报状态时，您可以使用外部警报。

警报状态属性要求警报状态数据值采用特定的格式。每个数据值都必须是序列化为字符串的 JSON 对象。然后，将序列化的字符串作为字符串值提取。有关更多信息，请参阅 警报状态属性。

Example 警报状态数据值示例（未序列化）

```
{
  "stateName": "Active"
}
```

Example 警报状态数据值示例（序列化）

```
{\"stateName\":\"Active\"}
```

> **Note**
>
> 如果您的数据来源无法报告这种格式的数据，或者您无法在采集数据之前将其转换为此格式，则可以选择不使用警报属性。相反，例如，您可以将数据作为具有字符串数据类型的测量属性进行获取。有关更多信息，请参阅定义来自设备的数据流（测量值）和将数据摄取到 Amazon IoT SiteWise。

映射外部警报状态流 Amazon IoT SiteWise

您可以定义属性别名，将数据流映射到警报状态属性。这有助于您在获取或检索数据时轻松识别警报状态属性。有关属性别名的更多信息，请参阅 管理数据流 Amazon IoT SiteWise。

主题

- 映射外部警报状态流（控制台）
- 映射外部警报状态流（Amazon CLI）

映射外部警报状态流（控制台）

您可以定义属性别名，将数据流映射到警报状态属性。这有助于您在获取或检索数据时轻松识别警报状态属性。有关属性别名的更多信息，请参阅 管理数据流 Amazon IoT SiteWise。

您可以使用 Amazon IoT SiteWise 控制台为警报状态属性设置别名。

为警报状态属性设置属性别名（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择要为其设置属性别名的资产。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择编辑。
5. 滚动到警报并展开该部分。
6. 在外部警报下，在属性别名 - 可选中输入别名。
7. 选择保存。

映射外部警报状态流（Amazon CLI）

您可以定义属性别名，将数据流映射到警报状态属性。这有助于您在获取或检索数据时轻松识别警报状态属性。有关属性别名的更多信息，请参阅 管理数据流 Amazon IoT SiteWise。

您可以使用 Amazon Command Line Interface (Amazon CLI) 为警报状态属性设置别名。

您必须知道资产的 `assetId` 和属性的 `propertyId` 才能完成此过程。您还可以使用外部 ID。如果您创建了资产但不知道该资产`assetId`，请使用 ListAssetsAPI 列出特定模型的所有资产。使用该DescribeAsset操作查看资产的属性，包括财产 IDs。

> **Note**
>
> DescribeAsset响应包括该资产的复合资产模型列表。每个警报都是一个复合模型。要查找 `propertyId`，请找到警报的复合模型，然后在该复合模型中找到 `AWS/ALARM_STATE` 属性。

有关如何设置属性别名的更多信息，请参阅 更新资产属性的别名。

摄取警报状态数据 Amazon IoT SiteWise

警报状态属性期望警报状态为序列化的 JSON 字符串。要将警报状态引入外部警报 Amazon IoT SiteWise，请将此序列化字符串作为带时间戳的字符串值提取。以下示例演示了活动警报的状态数据值。

```
{\"stateName\":\"Active\"}
```

要标识警报状态属性，您可以指定以下项之一：

- 您要向其发送数据的警报属性的 `assetId` 和 `propertyId` 。
- `propertyAlias`，这是一个数据流别名（例如，`/company/windfarm/3/turbine/7/temperature/high`）。要使用此选项，您必须首先设置警报属性的别名。要了解如何为警报状态属性设置属性别名，请参阅 映射外部警报状态流 Amazon IoT SiteWise。

以下 BatchPutAssetPropertyValueAPI 负载示例，演示了如何格式化外部警报的状态。当风力涡轮机的每分钟转速 (RPM) 读数过高时，此外部警报会报告。

Example 警报状态数据的 BatchPutAssetPropertyValue 有效负载示例

```
{
    "entries": [
      {
        "entryId": "unique entry ID",
        "propertyAlias": "/company/windfarm/3/turbine/7/temperature/high",
        "propertyValues": [
```

```
          {
            "value": {
              "stringValue": "{\"stateName\":\"Active\"}"
            },
            "timestamp": {
              "timeInSeconds": 1607550262
            }
          }
        ]
      }
    ]
  }
```

有关如何使用 `BatchPutAssetPropertyValue` API 获取数据的更多信息，请参阅 使用摄取数据 Amazon IoT SiteWise APIs。

要进一步了解有关采集数据的其他方式，请参阅 将数据摄取到 Amazon IoT SiteWise。

Amazon IoT SiteWise 助手

As Amazon IoT SiteWise sistant 是一款由 AI 驱动的生成式助手。它允许工厂经理、质量工程师和维护技术人员等用户直接从运营和企业数据中获得见解、解决问题并采取行动。

该 Amazon IoT SiteWise 助理将 Amazon IoT 数据、资产模型、手册和文档中的信息整合为可理解的关键事件摘要。它还支持交互式深入问答环节，便于诊断、根本原因探索和指导性建议。

主题

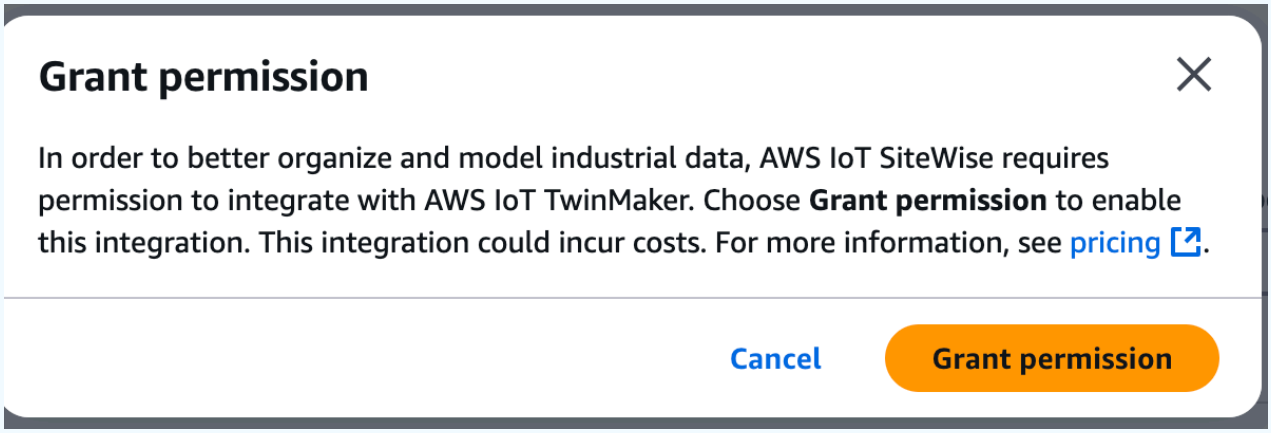
- 配置 Amazon IoT SiteWise 助手
- 创建数据集
- 编辑数据集
- 删除数据集
- Amazon IoT SiteWise 助手问题

配置 Amazon IoT SiteWise 助手

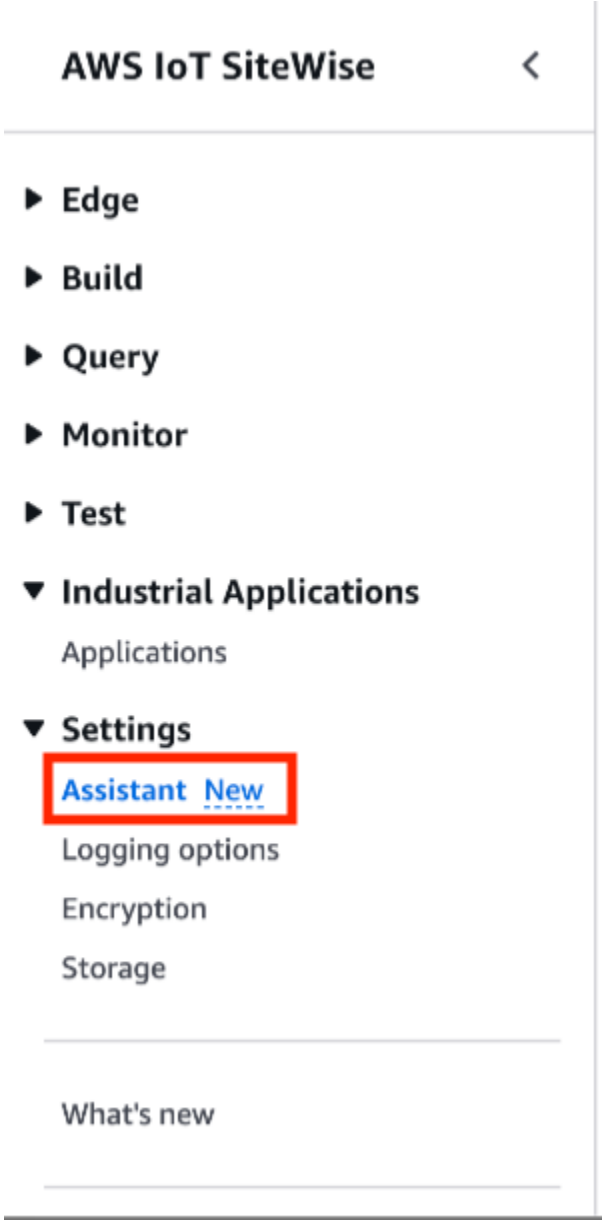
Amazon IoT SiteWise 助手配置

1. 登录 Amazon IoT SiteWise 控制台。

> **Note**
>
> 授予权限以启用与 Amazon IoT TwinMaker 服务的集成。这是 Assi Amazon IoT SiteWise stant 和仪表板在 Amazon IoT SiteWise 资源中执行 SQL 查询所必需的。参见集成 Amazon IoT SiteWise 和 Amazon IoT TwinMaker。



2. 从左侧导航面板中选择 “助手”。



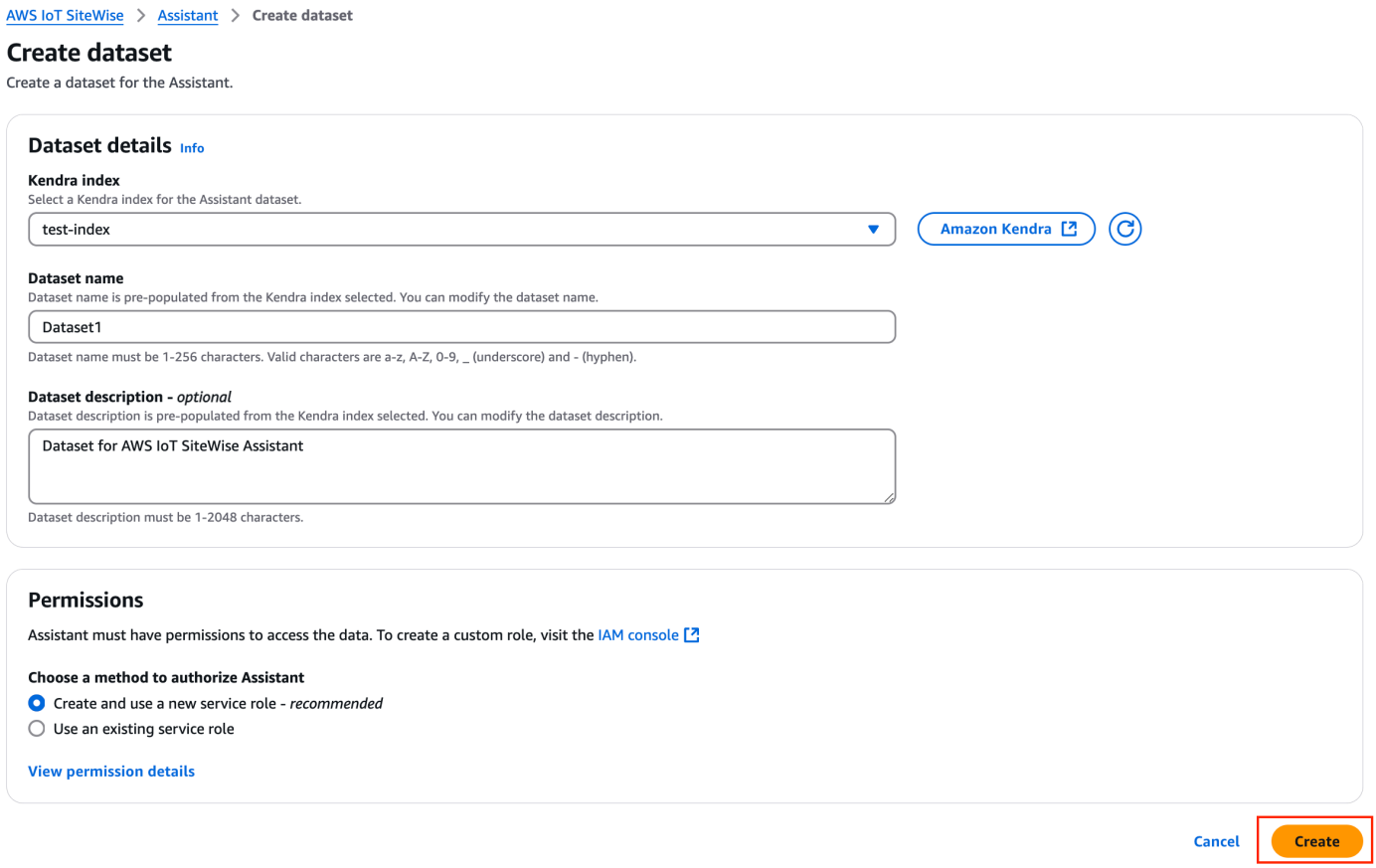
创建数据集

> **Note**
>
> Amazon IoT SiteWise 助手必须使用带有 Amazon Kendra 索引的数据集来获得企业级知识和指导。如果您没有 Amazon Kendra 索引，请参阅创建索引来创建索引。添加数据集可以提高助手的响应质量，并最大限度地减少幻觉。

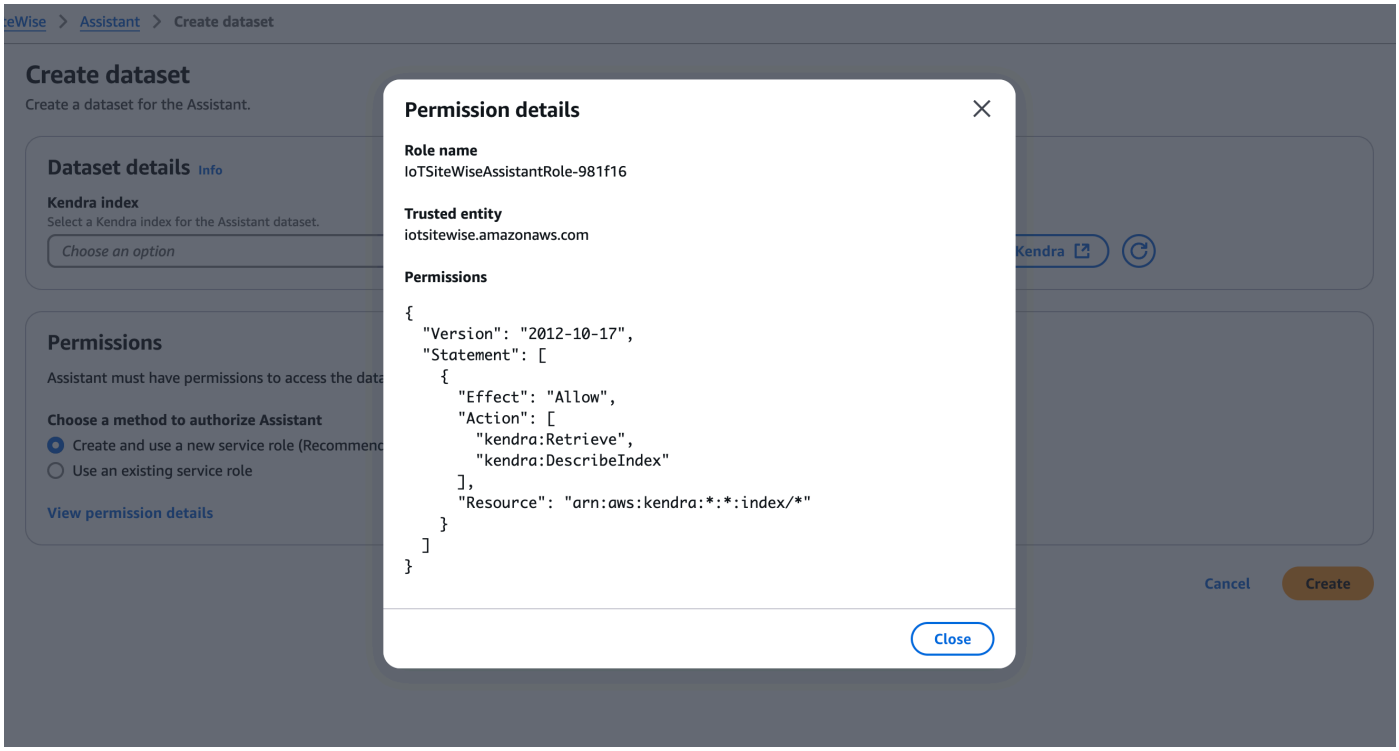
Console

在 Amazon IoT SiteWise 控制台中创建数据集

1. 数据集显示在 “Amazon IoT SiteWise 助手” 页面的 “数据集” 部分。
2. 如果不存在数据集，请选择创建数据集。
3. 在数据集详细信息页面中，从下拉菜单中选择要与数据集关联的 Kendra 索引。
4. 数据集名称由步骤 3 中选择的 Kendra 索引填充。如果需要，可以编辑名称。
5. （可选）数据集描述由步骤 3 中选择的 Kendra 索引填充。如果需要，可以编辑描述。
6. 在 “权限” 部分中，从以下选项中进行选择：
   a. 请选择创建和使用新的服务角色。默认情况下， Amazon IoT SiteWise 会自动创建服务角色。此角色允许 Amazon IoT SiteWise 助手访问您的 Kendra 索引。
   b. 选择使用现有服务角色，然后选择目标角色。
7. 选择创建。



Amazon IoT SiteWise 为用户创建的服务角色（如果用户选择创建并使用新的服务角色）。



Amazon CLI

在中创建数据集 Amazon CLI

1. 创建用于创建数据集的 IAM 角色。使用以下权限策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kendra:Retrieve"
            ],
            "Resource": "arn:aws:kendra:*:*:index/*"
        }
    ]
}
```

使用以下信任关系：

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iotsitewise.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. 使用示例中提供的模板创建一个文件 create-dataset.json。填充`datasetId``kendra knowledgeBaseArn`并连接`roleArn`此数据集。

```
{
    "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. 使用以下命令创建数据集：

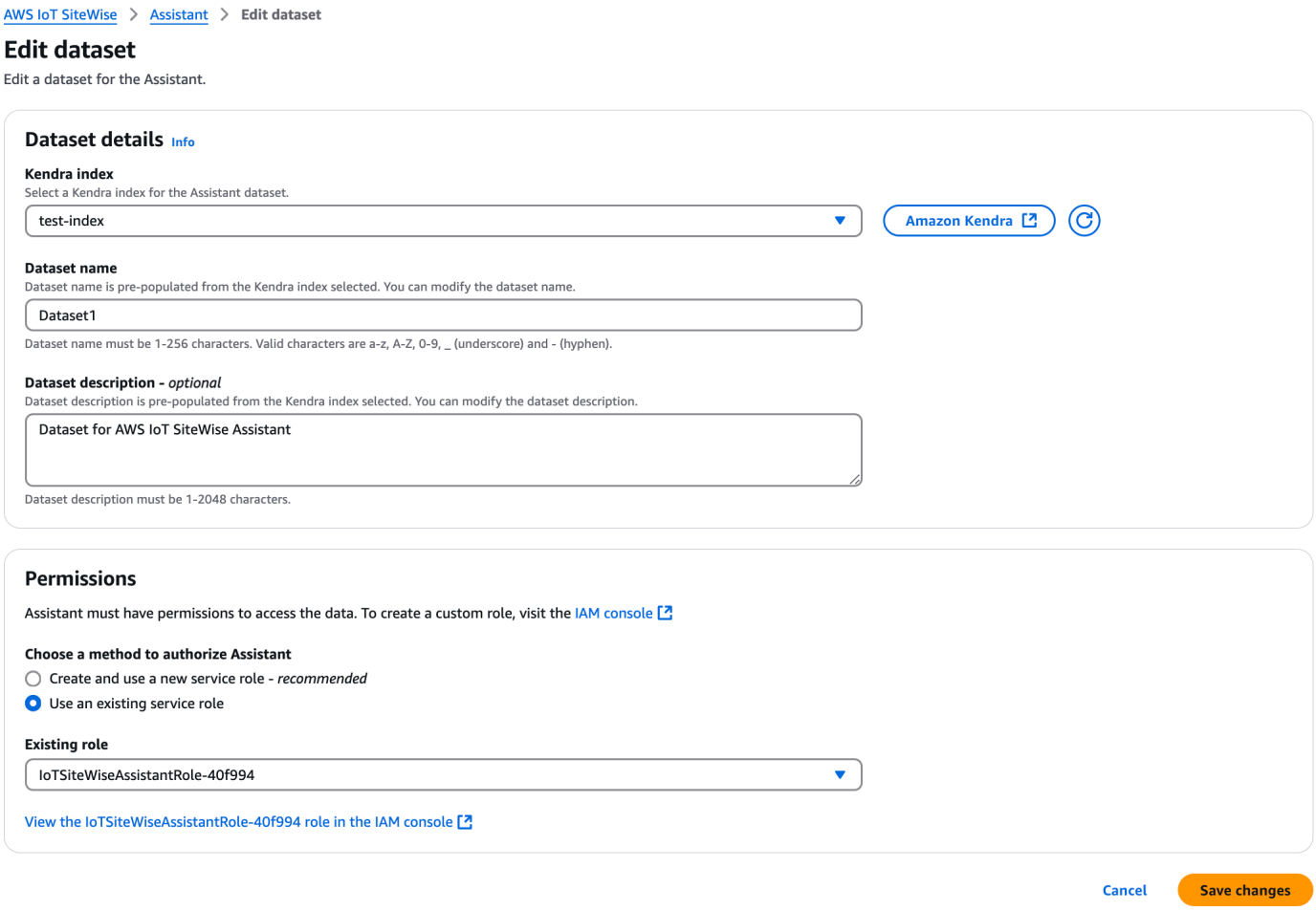
```
aws iotsitewise create-dataset --cli-input-json file://create-dataset.json --
region us-east-1
```

编辑数据集

Console

编辑数据集

1. 数据集显示在 “助手” 页面的 “数据集” 部分。选择要编辑的数据集。选择 “编辑” 开始编辑。
2. 在数据集详细信息页面中，从下拉菜单中选择要与数据集关联的 Kendra 索引。
3. 数据集名称由步骤 2 中选择的 Kendra 索引填充。如果需要，可以编辑名称。
4. （可选）数据集描述由步骤 2 中选择的 Kendra 索引填充。如果需要，可以编辑描述。
5. 在 “权限” 部分中，从以下选项中进行选择：
   a. 请选择创建和使用新的服务角色。默认情况下， Amazon IoT SiteWise 会自动创建服务角色。此角色允许 Amazon IoT SiteWise 助手访问您的 Kendra 索引。
   b. 选择使用现有服务角色，然后选择目标角色。
6. 选择 “保存更改” 以保存您的选择。



Amazon CLI

在中编辑数据集 Amazon CLI

1. 使用示例中提供的模板创建文件 update-dataset.json。填充`datasetId``kendra knowledgeBaseArn`并连接`roleArn`此数据集。

```
{
    "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. 使用以下命令更新数据集：

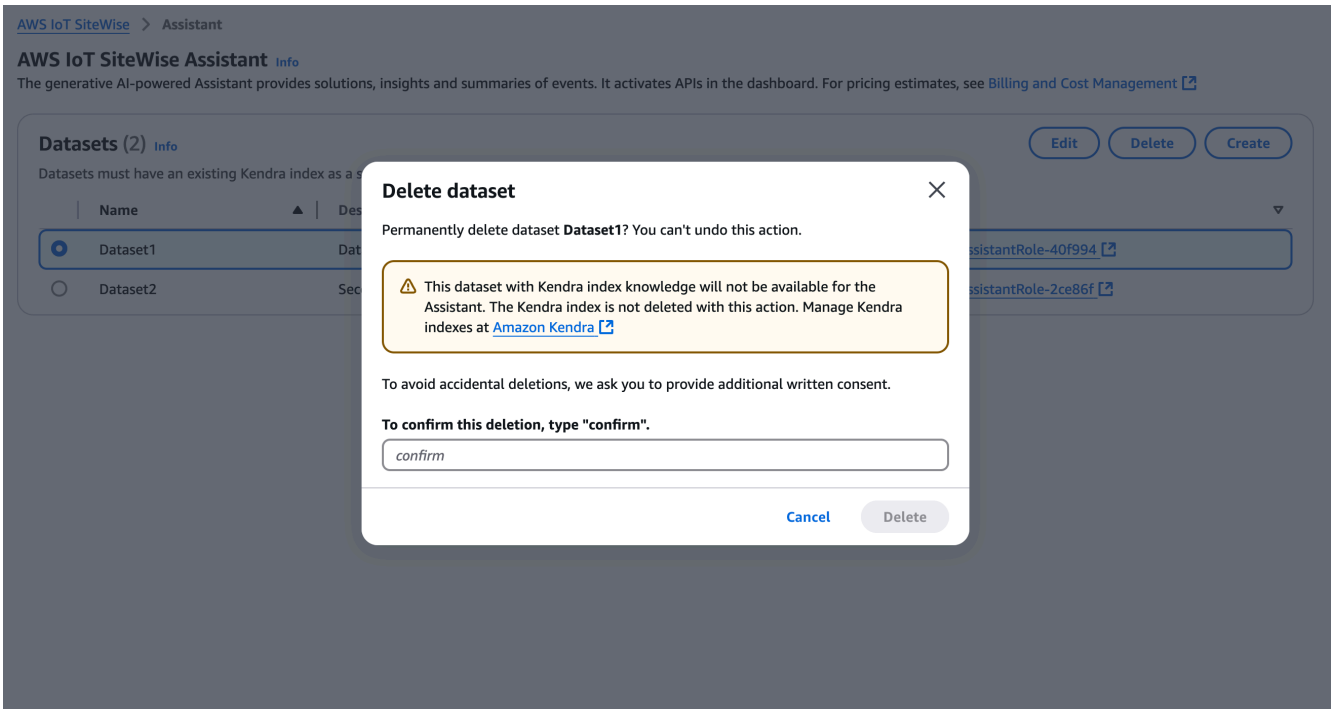
```
aws iotsitewise update-dataset --cli-input-json file://update-dataset.json --
region us-east-1
```

删除数据集

Console

删除数据集

1. 数据集显示在 “助手” 页面的 “数据集” 部分。选择一个数据集。选择删除。
2. 在弹出窗口中键入确认以确认删除。



3. 选择删除。

Amazon CLI

删除数据集

- 使用删除数据集`datasetId`。

```
aws iotsitewise delete-dataset --region us-east-1 --dataset-id <UUID>
```

Amazon IoT SiteWise 助手问题

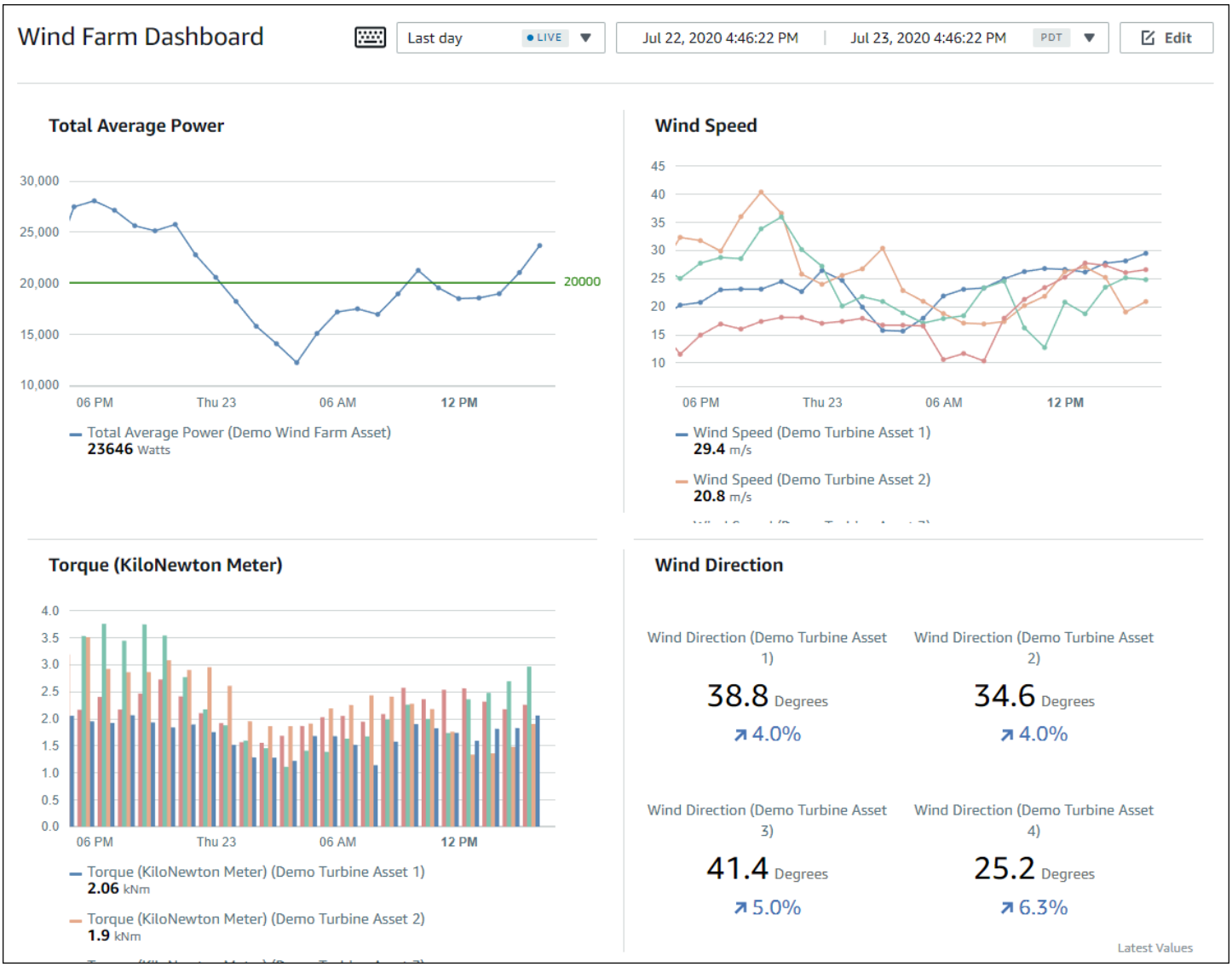
要问 Amazon IoT SiteWise 助手的问题示例要了解有关查询 Amazon IoT SiteWise 助手的更多信息，请参阅。

使用监控数据 Amazon IoT SiteWise Monitor

您可以通过 Amazon IoT SiteWise 创建 Monitor Web 门户来 SiteWise 监控来自流程、设备和设备的数据。 SiteWise Monitor 是一 Amazon IoT SiteWise 项功能，可用于以托管 Web 应用程序的形式创建门户。然后，您可以使用这些门户查看和共享您的操作数据。您可以使用控制面板创建项目来可视化流程、设备以及连接到 Amazon IoT的装置中的数据。

领域专家（如流程工程师）可以使用这些门户快速查看操作数据，进而了解设备和装置的行为。

以下是显示风力发电厂数据的示例控制面板。



由于可以 Amazon IoT SiteWise 捕获一段时间内的数据，因此您可以使用 M SiteWise onitor 查看一段时间内的操作数据，或者查看特定时间点上次报告的值。这样，您就可以发掘以其他方式可能很难发现的情况。

SiteWise 监控角色

四个角色与 SiteWise 监视器交互：

Amazon 管理员

Amazon 管理员使用 Amazon IoT SiteWise 控制台创建门户。 Amazon 管理员还可以分配门户管理员和添加门户用户。门户管理员可在以后将门户用户分配到项目，作为拥有者或查看者。 Amazon 管理员只能在 Amazon 控制台中工作。

门户管理员

每个 M SiteWise onitor 门户都有一个或多个门户管理员。门户管理员使用门户创建包含资产和控制面板集合的项目。门户管理员随后会将资产和拥有者分配给每个项目。通过控制对项目的访问，门户管理员可以指定项目拥有者和查看者可以查看哪些资产。

项目拥有者

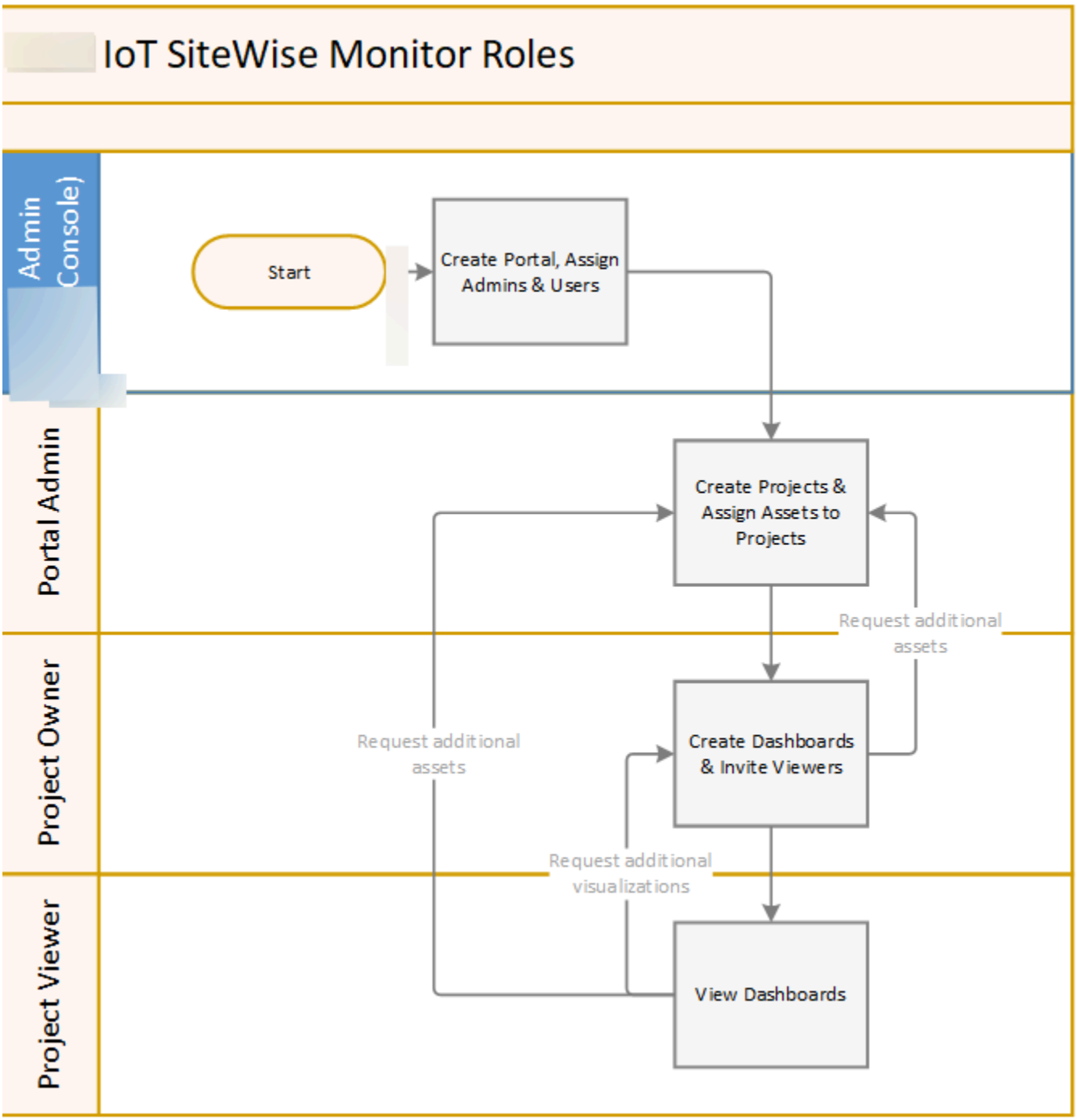
每个 M SiteWise onitor 项目都有所有者。项目所有者以控制面板形式创建可视化效果，来一致地呈现您的操作数据。当控制面板准备好共享时，项目所有者可以邀请查看者加入项目。项目所有者还可以将其他所有者分配给项目。项目所有者可以为警报配置阈值和通知设置。

项目查看者

每个 M SiteWise onitor 项目都有查看器。项目查看者可以连接到门户以查看项目拥有者创建的控制面板。在每个控制面板中，项目查看者可以调整时间范围以更好地了解运营数据。项目查看者只能查看其具有访问权限的项目中的控制面板。项目查看者可以确认和暂停警报。

根据您的组织，同一人员可能承担多种角色。

下图说明了这四个角色在 Monito SiteWise r 门户中的交互方式。



您可以使用 IAM 来管理谁有权访问您的数据。您的数据用户可以使用他们的 凭证从桌面或移动浏览器登录 M SiteWise onitor。

SAML 联合身份验证

IAM 通过 SAML（安全断言标记语言）2.0支持身份联合验证。SAML 2.0 是一个开放标准，许多外部身份提供商 (IdPs) 使用它来对用户进行身份验证并将其身份和安全信息传递给服务提供商 (SPs)。

SPs 通常是应用程序或服务。SAML 联合使您的 M SiteWise onitor 门户管理员和用户能够使用外部凭据（例如公司用户名和密码）登录他们分配的门户。

您可以将 IAM 配置为使用基于 SAML 的联合身份验证访问您的 SiteWise 监控器门户。

IAM

您的门户管理员和用户可以申请临时安全证书，以访问他们分配的 Monito SiteWise r 门户。您可以在 IAM 中创建 SAML 身份提供商身份，以便在您的身份提供商与 Amazon之间建立信任关系。有关更多信息，请参阅 I AM 用户指南中的使用基于 SAML 的联合身份 Amazon验证进行 API 访问权限。

您的门户管理员和用户可以登录贵公司的门户，然后选择进入 Amazon 管理控制台的选项。然后，他们可以导航到分配给他们的 SiteWise 监视器入口。贵公司的门户网站负责处理您的身份提供商与之间的信任交换 Amazon。有关更多信息，请参阅 IAM 用户指南中的允许 SAML 2.0 联合用户访问 Amazon 管理控制台。

> **Note**
>
> 向门户添加用户或管理员时，请避免创建限制用户权限（例如受限 IP）的 IAM policy。任何具有受限权限的附加策略都将无法连接到 Amazon IoT SiteWise 门户。

SiteWise 显示器概念

要使用 M SiteWise onitor，您应该熟悉以下概念：

门户

Amazon IoT SiteWise Monitor 门户是一个 Web 应用程序，可用于可视化和共享 Amazon IoT SiteWise 数据。门户具有一个或多个管理员，并且包含零个或多个项目。

Project

每个 M SiteWise onitor 门户都包含一组项目。每个项目都有一个与之关联的 Amazon IoT SiteWise 资产子集。项目拥有者创建一个或多个控制面板，以提供一致的方式来查看与这些资产关联的数据。项目拥有者可以邀请查看者加入项目，以允许他们查看项目中的资产和控制面板。该项目是 Monito SiteWise r 内部共享的基本单元。项目所有者可以邀请 Amazon 管理员授予门户访问权限的用户。用户必须具有门户访问权限，然后才能与该用户共享该门户中的项目。

资产

当 Amazon IoT SiteWise 从您的工业设备中摄取数据时，您的设备、设备和流程将分别表示为资产。每项资产都有与之关联的属性和警报。门户管理员还将资产组分配给每个项目。

属性

属性是与资产关联的时间序列数据。例如，一套设备可能具有序列号、位置、品牌和型号以及安装日期。它还可能具有可用性、性能、质量、温度、压力等时间序列值。

警报

警报监控属性，以识别设备何时超出其工作范围。每个警报都定义了一个阈值和一个要监控的属性。当属性超过阈值时，警报将变为活动状态，并表示您或您的团队中的某人应解决问题。项目所有者可以自定义警报的阈值和通知设置。项目查看者可以确认和暂停警报，他们可以留言，详细说明警报或他们为解决警报而采取的措施。

控制面板

每个项目都包含一组控制面板。控制面板为一组资产的值提供一组可视化效果。项目拥有者创建控制面板及其包含的可视化效果。当项目拥有者准备好共享控制面板集时，拥有者可以邀请查看者加入项目，从而使他们能够访问项目中的所有控制面板。如果希望为不同的控制面板使用一组不同的查看器，则必须在项目之间划分控制面板。在查看者查看控制面板时，他们可以自定义用于查看特定数据的时间范围。

可视化

在每个控制面板中，项目拥有者决定如何显示与项目关联的资产的属性和警报。可用性可以用折线图表示，而其他值可能显示为条形图或关键绩效指标 (KPIs)。警报最好以状态网格和状态时间轴的形式显示。项目所有者可以自定义每个可视化效果，以便更好地了解该资产的数据。

开始使用 Amazon IoT SiteWise Monitor（经典）

如果您是组织的 Amazon 管理员，则可以通过 Amazon IoT SiteWise 控制台创建门户。完成以下步骤以创建门户，以便您的组织成员可以查看您的 Amazon IoT SiteWise 数据：

1. 配置和创建门户
2. 添加门户管理员并发送邀请电子邮件
3. 添加门户用户

创建门户后，门户管理员可以查看您的 Amazon IoT SiteWise 资产并将其分配给门户中的项目。然后，项目拥有者可以创建控制面板来显示资产的属性，以帮助项目查看者了解设备、流程和装置的运行情况。

> **Note**
>
> 向门户添加用户或管理员时，请避免创建限制用户权限 Amazon Identity and Access Management（例如受限 IP）的 (IAM) 策略。任何具有受限权限的附加策略都将无法连接到 Amazon IoT SiteWise 门户。

您可以按照教程的说明，逐步设置具有项目、控制面板和多个用户的门户，用于使用风电场数据的特定场景。有关更多信息，请参阅 在 Monito SiteWise r 中可视化和共享风电场数据。

主题

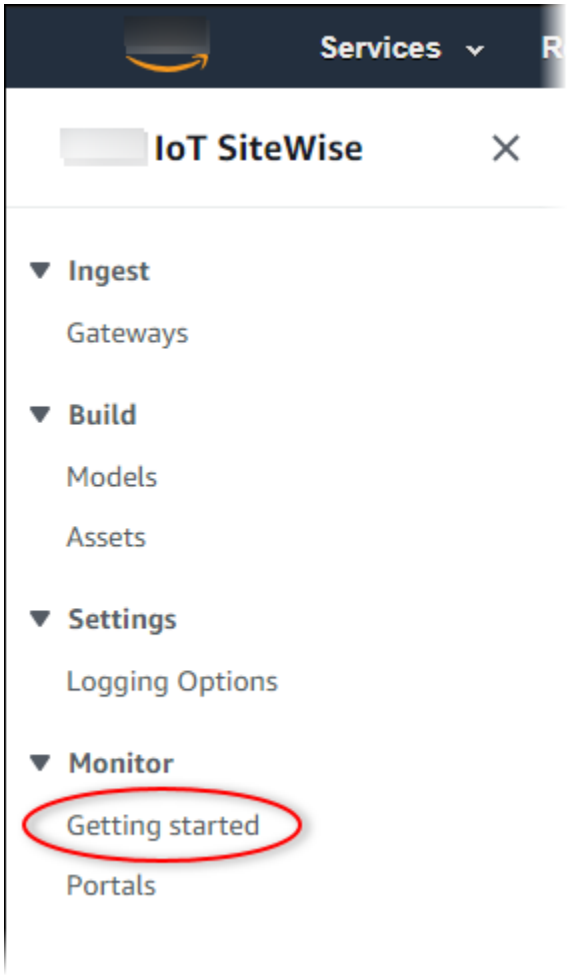
- 在 Monito SiteWise r 中创建门户
- 在 Monito SiteWise r 中配置您的门户
- 在 SiteWise 监控器中邀请管理员
- 在 SiteWise 监控器中添加门户用户
- 创建 Amazon IoT SiteWise 仪表板 (Amazon CLI)
- 在中为你的传送门开启警报 Amazon IoT SiteWise
- 在边缘启用您的 Amazon IoT SiteWise 门户
- 管理您的 SiteWise 监控器门户

在 Monito SiteWise r 中创建门户

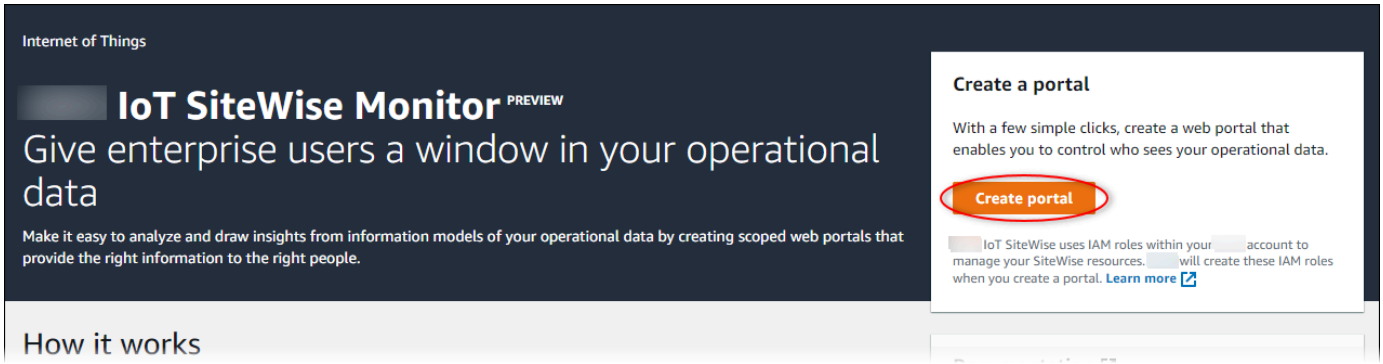
您可以在 Amazon IoT SiteWise 控制台中创建 SiteWise 监控器门户。

创建门户

1. 登录 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 监控、入门。



3. 选择 创建门户。



接下来，您必须提供一些基本信息来配置门户。

在 Monito SiteWise r 中配置您的门户

用户使用门户查看您的数据。您可以自定义门户的名称、描述、品牌、、支持联系人电子邮件和权限。

IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2- *optional*
Additional features

Step 3
Invite administrators

Step 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 Single Sign-On SSO) or Identity and Access Management (IAM) credentials. If you choose SSO, you must enable the service for your account.

You haven't enabled in your account yet. When you create your first portal user, this automatically enables SSO in your account.

Create user

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 IoT SiteWise resources. Learn

配置门户

1. 输入门户的名称。
2. （可选）输入门户的描述。如果您有多个门户，请使用有意义的描述来帮助您跟踪每个门户包含的内容。
3. （可选）上传图片以在门户中显示您的品牌。选择一个正方形 PNG 图片。如果上传非方形图片，门户会将图片缩减为正方形。
4. 输入门户用户在遇到门户问题需要帮助时可以联系的电子邮件地址。
5. （可选）为门户添加标签。有关更多信息，请参阅 标记您的 Amazon IoT SiteWise 资源。
6. 请选择以下选项之一：
   - 请选择创建和使用新的服务角色。默认情况下，M SiteWise onitor 会自动为每个门户创建一个服务角色。此角色允许您的门户用户访问您的 Amazon IoT SiteWise 资源。有关更多信息，请参阅 将服务角色用于 Amazon IoT SiteWise Monitor。
   - 选择使用现有服务角色，然后选择目标角色。
7. 选择 下一步。
8. 选择 “创建”。 Amazon IoT SiteWise 将创建您的门户。

> **Note**
>
> 如果关闭控制台，您可以通过添加管理员和用户来完成设置过程。有关更多信息，请参阅 在中添加或移除门户管理员 Amazon IoT SiteWise。如果您不想保留此门户，请将其删除，以免使用资源。有关更多信息，请参阅 删除中的传送门 Amazon IoT SiteWise。

状态列可以是以下值之一。

- 正在创建- Amazon IoT SiteWise 正在处理您创建门户的请求。此过程可能需要几分钟才能完成。
- 更新- Amazon IoT SiteWise 正在处理您更新门户的请求。此过程可能需要几分钟才能完成。
- 待处理- Amazon IoT SiteWise 正在等待 DNS 记录传播完成。此过程可能需要几分钟完成。当门户的状态为 PENDING 时，您可以将其删除。
- 正在删除- Amazon IoT SiteWise 正在处理您删除门户的请求。此过程可能需要几分钟才能完成。
- ACTIVE - 当门户变为活动状态时，您的门户用户可以访问它。

- 失败- Amazon IoT SiteWise 无法处理您创建、更新或删除门户的请求。如果您启用 Amazon IoT SiteWise 向 Amazon Logs 发送 CloudWatch 日志，则可以使用这些日志来解决问题。有关更多信息，请参阅Amazon IoT SiteWise 使用 CloudWatch 日志进行监控。

门户创建时将显示一条消息。

接下来，您必须为门户邀请一个或多个门户管理员。到目前为止，您创建了一个门户，但没有人可以访问它。

在 SiteWise 监控器中邀请管理员

要开始使用新门户，您必须分配门户管理员。门户管理员创建项目、选择项目拥有者并将资产分配给项目。门户管理员可以查看您的所有资 Amazon IoT SiteWise 产。

根据用户身份验证服务，请选择以下选项之一：

IAM

您可以选择用户或角色作为门户管理员。如果您想添加其他用户或角色作为门户管理员，可以在 IAM 控制台中创建用户或角色。有关更多信息，请参阅 IAM 用户指南中的在您的 Amazon 账户中创建 IAM 用户和创建 IAM 角色。

邀请管理员

1. 执行以下操作：
   - 选择 IAM 用户以添加一个 IAM 用户作为您的门户管理员。
   - 选择 IAM 角色以将 IAM 角色添加为门户管理员。
2. 选中您希望作为门户管理员的用户或角色对应的复选框。这会将用户或角色添加到门户管理员列表中。
3. 如果您添加了不希望其作为管理员的用户或角色，请清除该用户或角色对应的复选框。
4. 邀请门户管理员步骤完成后，选择 下一步。

> **Important**
>
> 用户或角色必须具有登录门户的 `iotsitewise:DescribePortal` 权限。

> **Note**
>
> 如果您使用 IAM Identity Center 作为身份存储，并且已登录 Amazon Organizations 管理账户，则可以选择创建用户来创建 IAM Identity Center 用户。IAM Identity Center 会向新用户发送一封电子邮件，让他们设置密码。然后，您可以将该用户作为管理员分配到门户。有关更多信息，请参阅管理 IAM Identity Center 中的身份。

您可以在以后更改门户管理员列表。有关更多信息，请参阅 在中添加或移除门户管理员 Amazon IoT SiteWise。

> **Note**
>
> 由于只有门户管理员可以创建项目并将资产分配给项目，因此在您指定至少一个门户管理员之前，门户的用途非常有限。

在最后一步中，您添加可以访问新门户的用户。

在 SiteWise 监控器中添加门户用户

您控制哪些用户可以访问您的门户。在每个门户中，门户管理员创建一个或多个项目，并分配门户用户作为每个项目的拥有者或查看者。每个项目的拥有者可以邀请其他门户用户来拥有或查看项目。

根据用户身份验证服务，请选择以下选项之一：

IAM

如果您看到要在 IAM 用户或 IAM 角色列表中添加的用户或角色，请完成以下步骤。

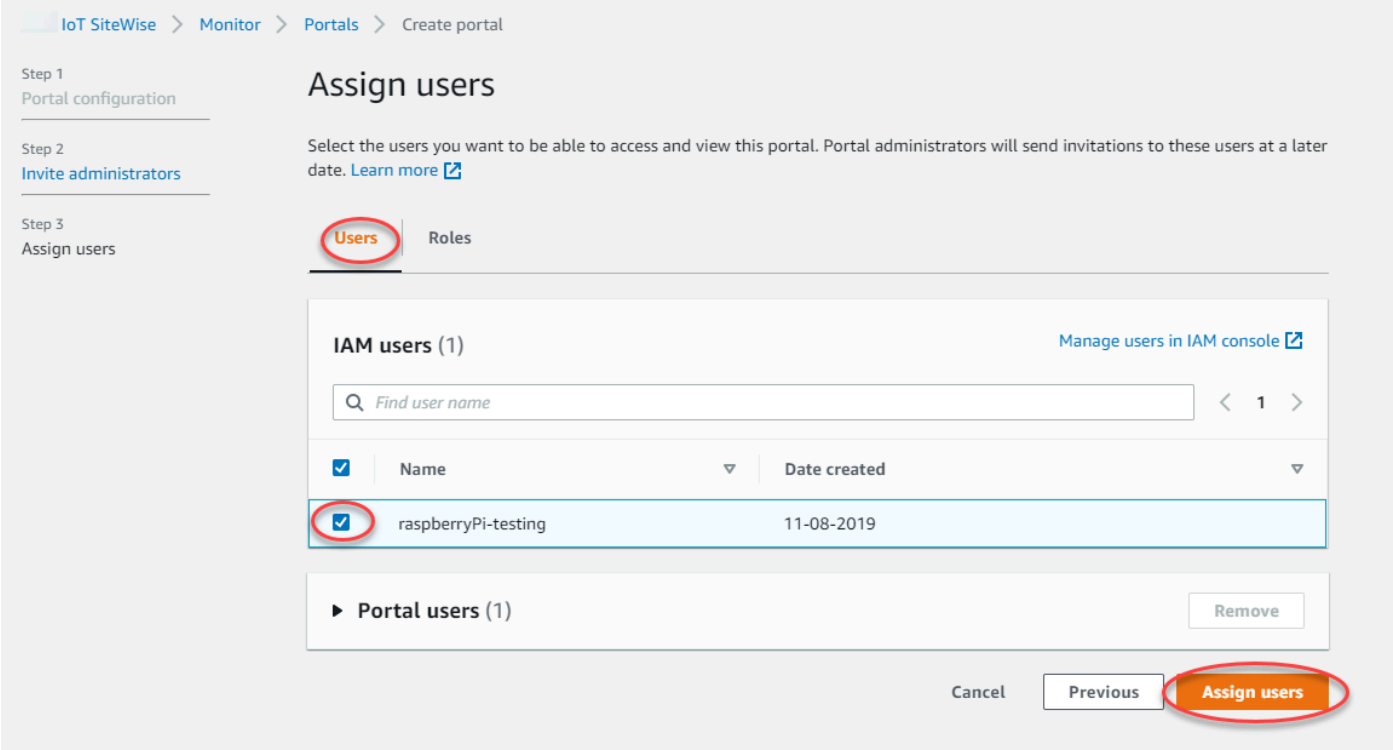
添加门户用户

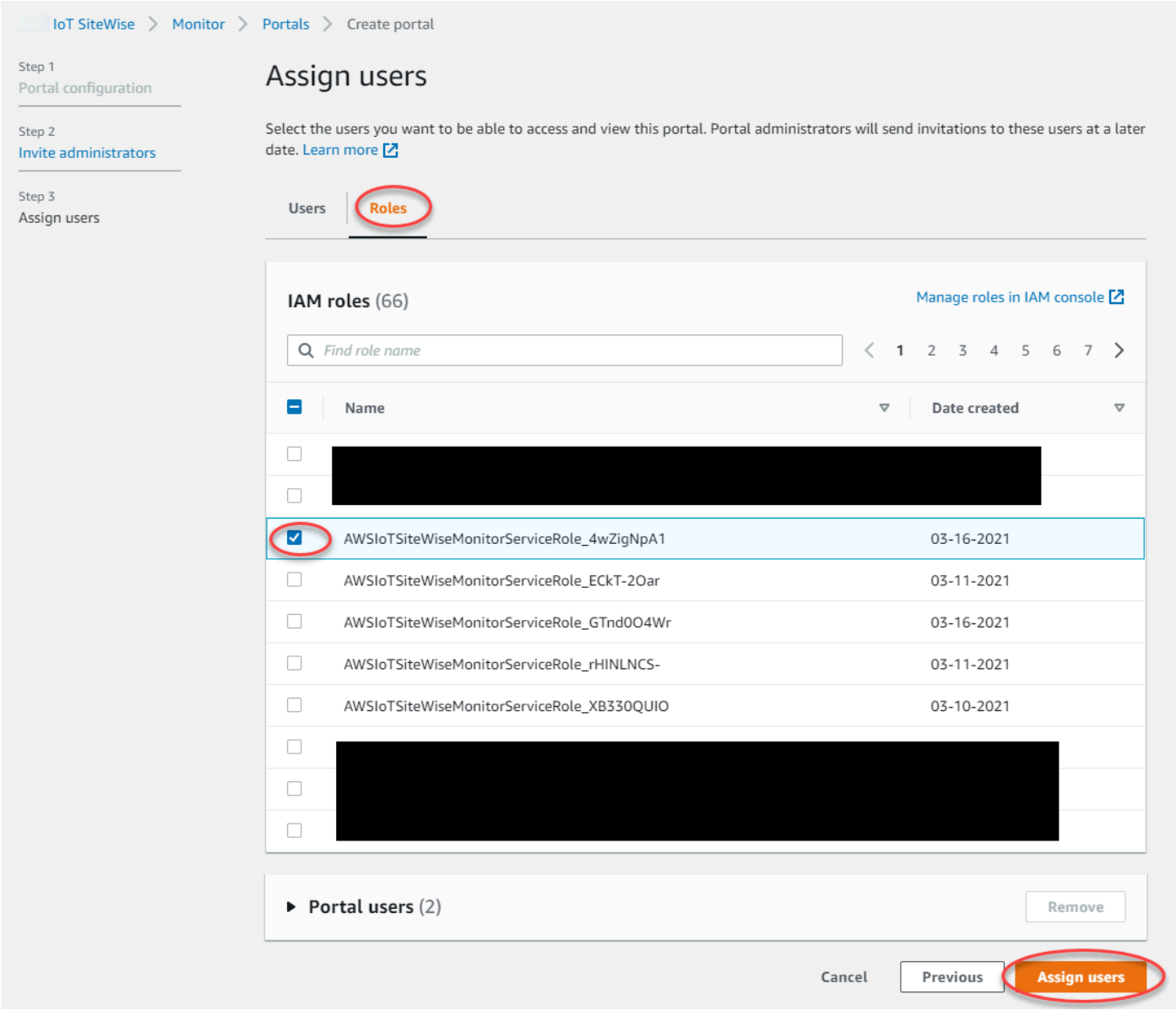
1. 执行以下操作：
   - 选择 IAM 用户以将 IAM 用户添加为门户用户。
   - 选择 IAM 角色以将 IAM 角色添加为门户用户。

   如果您是首次使用 M SiteWise onitor，则无需将门户管理员添加为门户用户。

2. 选中要作为门户用户的用户或角色所对应的复选框。这会将用户或角色添加到门户用户列表中。
3. 如果您添加了不希望其访问门户的用户，请清除该用户对应的复选框。
4. 选择完用户后，选择分配用户。

> **Important**
>
> 用户或角色必须具有登录门户的 `iotsitewise:DescribePortal` 权限。





恭喜您！您已成功创建门户、分配门户管理员并分配在收到邀请时可以使用该门户的用户。门户管理员现在可以创建项目并将资产添加到这些项目。然后，项目拥有者可以创建控制面板来可视化各个项目资产的数据。

您可以在以后更改门户用户列表。有关更多信息，请参阅 在中添加或移除门户用户 Amazon IoT SiteWise。

如果需要对门户进行更改，请参阅管理您的 SiteWise 监控器门户。

要开始使用门户，请参阅《SiteWise 监控器应用程序指南》中的 “入门”。

创建 Amazon IoT SiteWise 仪表板 (Amazon CLI)

使用在仪表板中定义可视化（或微件）时 Amazon CLI，必须在 `dashboardDefinition` JSON 文档中指定以下信息。此定义是CreateDashboard和UpdateDashboard操作的参数。

`widgets`

小部件定义结构的列表，其中每个结构均包含以下信息：

`type`

控件的类型。 Amazon IoT SiteWise 提供以下控件类型：

- `sc-line-chart` – 折线图。有关更多信息，请参阅 Amazon IoT SiteWise Monitor 应用指南中的折线图。
- `sc-scatter-chart` – 散点图。有关更多信息，请参阅Amazon IoT SiteWise Monitor 应用程序指南中的散点图。
- `sc-bar-chart` – 条形图。有关更多信息，请参阅Amazon IoT SiteWise Monitor 应用指南中的条形图。
- `sc-status-grid` – 一个状态控件，以网格形式显示资产属性的最新值。有关更多信息，请参阅Amazon IoT SiteWise Monitor 应用程序指南中的状态控件。
- `sc-status-timeline` – 一个状态控件，以时间轴的形式显示资产属性的历史值。有关更多信息，请参阅Amazon IoT SiteWise Monitor 应用程序指南中的状态控件。
- `sc-kpi` – 关键绩效指标 (KPI) 可视化。有关更多信息，请参阅Amazon IoT SiteWise Monitor 应用指南中的 KPI 控件。
- `sc-table` – 表格控件。有关更多信息，请参阅Amazon IoT SiteWise Monitor 应用指南中的表格控件。

`title`

小部件的标题。

`x`

小部件的水平位置，从网格的左侧开始。此值表示小部件在控制面板网格中的位置。

`y`

小部件的垂直位置，从网格的顶部开始。此值表示小部件在控制面板网格中的位置。

`width`

小部件的宽度，用控制面板网格上的空格数表示。

`height`

小部件的高度，用控制面板网格上的空格数表示。

`metrics`

指标结构的列表，其中每个结构均为此小部件定义一个数据流。此列表中的每个结构都必须包含以下信息：

`label`

要为此指标显示的标签。

`type`

该指标的数据源类型。 Amazon IoT SiteWise 提供以下指标类型：

- `iotsitewise` – 控制面板提取 Amazon IoT SiteWise中的资产属性的数据。如果选择此选项，则必须为此指标定义 `assetId` 和 `propertyId`。

`assetId`

（可选）Amazon IoT SiteWise中的资产的 ID。

如果您在该指标中为 `type` 选择 `iotsitewise`，则此字段是必填字段。

`propertyId`

（可选）Amazon IoT SiteWise中的资产属性的 ID。

如果您在该指标中为 `type` 选择 `iotsitewise`，则此字段是必填字段。

`analysis`

或者，一种结构，用于定义要为小组件显示的分析（例如趋势线）。有关更多信息，请参阅Amazon IoT SiteWise Monitor 应用指南中的配置趋势线。您可以在小组件中为每个属性添加每种类型的趋势线。分析结构包含以下信息：

`trends`

或者，趋势结构列表，每个趋势结构都为该小组件定义了趋势分析。此列表中的每个结构包含以下信息：

`type`

趋势线的类型。选择以下选项：

- `linear-regression`— 显示线性回归线。 SiteWise Monitor 使用最小二乘法计算线性回归。

`annotations`

或者，定义小组件阈值的注释结构。有关更多信息，请参阅 版本指南中的Amazon IoT SiteWise Monitor 配置阈值。每个控件可以添加最多 6 个注释。注释结构包含以下信息：

`y`

或者，注释结构列表，每个注释结构都定义了此小组件的水平阈值。此列表中的每个结构包含以下信息：

`comparisonOperator`

阈值的比较运算符。选择下列选项之一：

- `LT` – 突出显示至少有一个数据点小于 `value` 的属性。
- `GT` – 突出显示至少有一个数据点大于 `value` 的属性。
- `LTE` – 突出显示至少有一个数据点小于或等于 `value` 的属性。
- `GTE` – 突出显示至少有一个数据点大于或等于 `value` 的属性。
- `EQ` – 突出显示至少有一个数据点等于 `value` 的属性。

`value`

用于将数据点与 `comparisonOperator` 进行比较的阈值。

`color`

（可选）阈值颜色的 6 位十六进制代码。对于至少有一个数据点符合阈值规则的属性，可视化效果会以这种颜色显示属性图例。默认值为黑色（`#000000`）。

`showValue`

或者，是否在控件的边距中显示阈值的值。默认值为 `true`。

`properties`

（可选）小部件属性的平面字典。该结构的成员取决于上下文。 Amazon IoT SiteWise 提供了以下使用的小部件`properties`：

- 折线图、散点图和条形图具有以下属性：

  `colorDataAcrossThresholds`

  或者，是否更改此小组件中跨越阈值的数据的颜色。启用此选项后，超过阈值的数据将以您选择的颜色显示。默认值为 `true`。

- 状态网格具有以下属性：

  `labels`

  或者，一种结构，用于定义要在状态网格上显示的标签。标签结构包含以下信息：

  `showValue`

  或者，是否在此小组件中显示每个资产属性的单位和值。默认值为 `true`。

Example 示例控制面板定义

以下示例从存储在 JSON 文件中的负载定义控制面板。

```
aws iotsitewise create-dashboard \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeEXAMPLE \
  --dashboard-name "Wind Farm Dashboard" \
  --dashboard-definition file://dashboard-definition.json
```

针对 `dashboard-definition.json` 的以下 JSON 示例定义具有以下可视化小部件的控制面板：

- 控制面板左上角的直观显示风电场总功率的折线图。该折线图包括一个阈值，该阈值指示风电场何时输出功率少于其最低预期输出。该折线图还包括一条线性回归趋势线。
- 控制面板右上角的直观显示四个涡轮机的风速的折线图。

> **Note**
>
> 此示例表示控制面板上的折线图和条形图可视化效果。此控制面板与示例风电场控制面板类似。

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

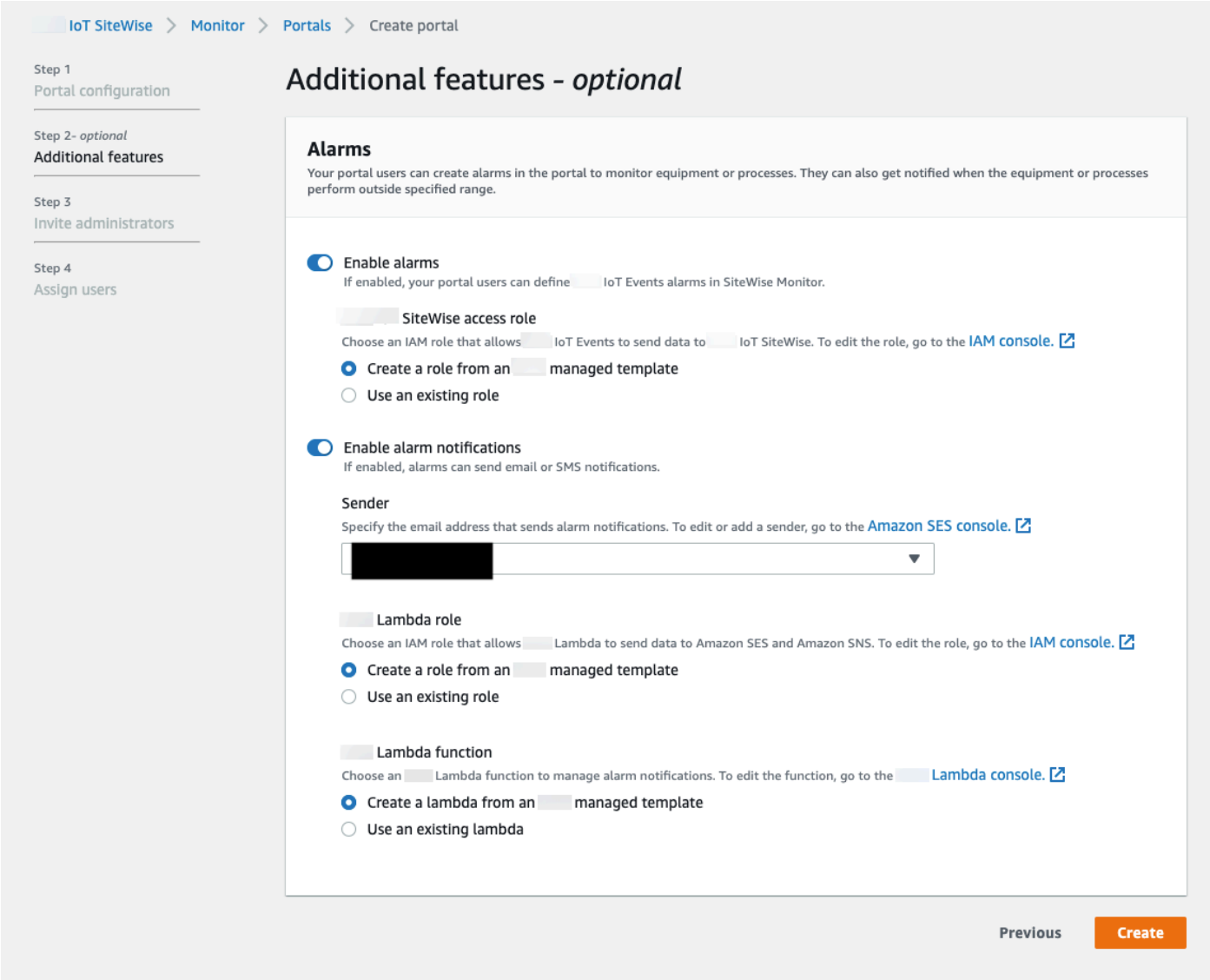
在中为你的传送门开启警报 Amazon IoT SiteWise

您可以 Amazon IoT Events 为门户启用支持的警报功能，以便门户管理员可以在您的 SiteWise 监控器入口中创建、编辑和删除 Amazon IoT Events 警报模型。项目所有者可以配置警报。项目查看者可以查看警报详情。本节介绍如何使用 Amazon IoT SiteWise 控制台为门户启用警报功能。

> **Important**
>
> - 您无法在门户中创建外部警报。
> - 如果要发送警报通知，则必须为用户身份验证服务选择 IAM Identity Center。
> - 警报通知功能在中国（北京 Amazon Web Services 区域）不可用。

配置和创建门户时，可以在步骤 2 其他功能中启用警报和警报通知。根据用户身份验证服务，请选择以下选项之一：

IAM Identity Center



为入口启用警报

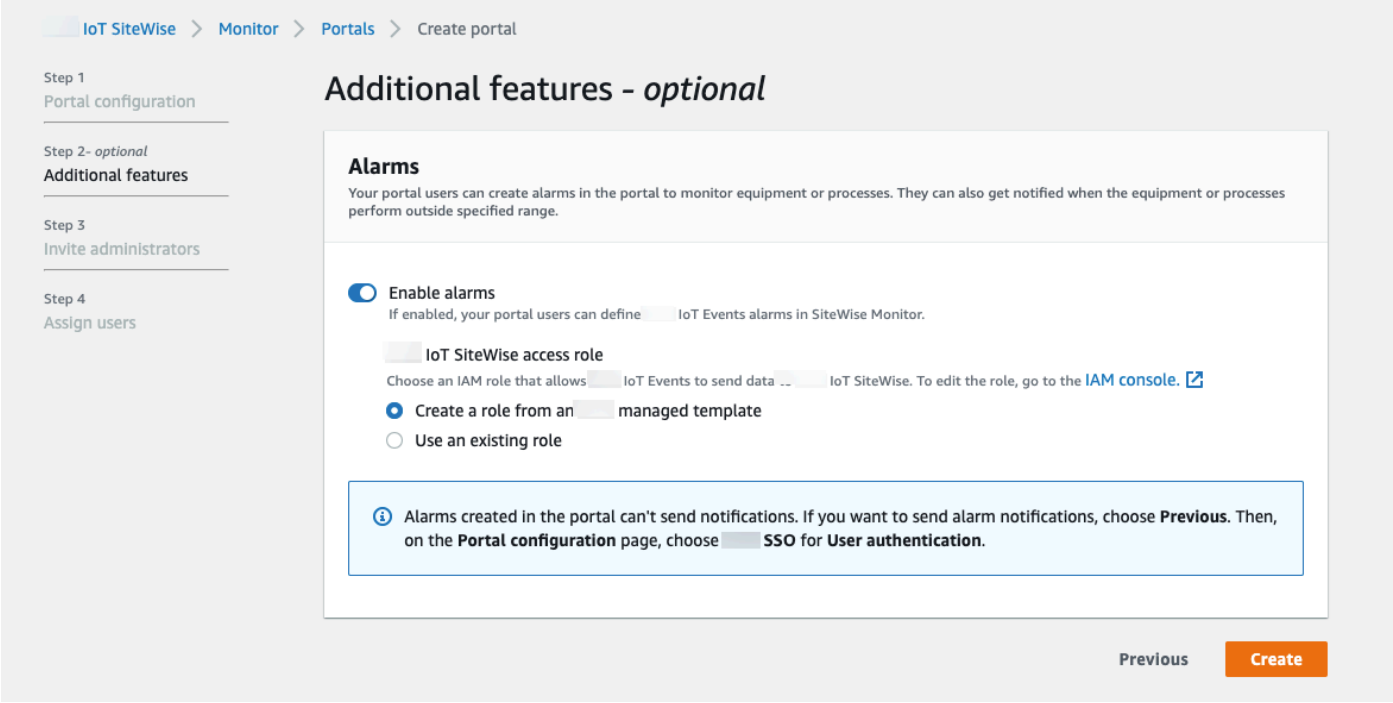
1. 或者，选择启用警报。
   - 对于 Amazon IoT SiteWise 访问角色，请使用现有角色或创建具有所需权限的角色。此角色需要 `iotevents:BatchPutMessage` 权限和信任关系，以允许 `iot.amazonaws.com` 和 `iotevents.amazonaws.com` 代入该角色。
2. 或者，选择启用警报通知。
   a. 对于 发送人，请选择发送人。

> **Important**
>
> 您必须在 Amazon SES 中验证发送人的电子邮件地址。有关更多信息，请参阅 Amazon Simple Email Service 开发人员指南 中的 在 Amazon SES 中验证电子邮件地址。

b. 对于 Amazon Lambda 角色，请使用现有角色或创建具有所需权限的角色。此角色需要 `lambda:InvokeFunction` 和 `sso-directory:DescribeUser` 权限以及允许 `iotevents.amazonaws.com` 和 `lambda.amazonaws.com` 代入该角色的信任关系。

c. 对于 Amazon Lambda 函数，请选择现有 Lambda 函数或或创建用于管理警报通知的函数。有关详细信息，请参阅 Amazon IoT Events 开发人员指南 中的 管理警报通知。

IAM



为入口启用警报

- 或者，选择启用警报。
  - 对于 Amazon IoT SiteWise 访问角色，请使用现有角色或创建具有所需权限的角色。此角色需要 `iotevents:BatchPutMessage` 权限和信任关系，以允许 `iot.amazonaws.com` 和 `iotevents.amazonaws.com` 代入该角色。

有关 Monito SiteWise r 中警报的更多信息，请参阅《Amazon IoT SiteWise 应用指南》中的 “使用警报进行监控”。

在边缘启用您的 Amazon IoT SiteWise 门户

在边缘启用门户后，在您的账户中启用了数据处理包的所有 SiteWise Edge 网关上均可使用此门户。

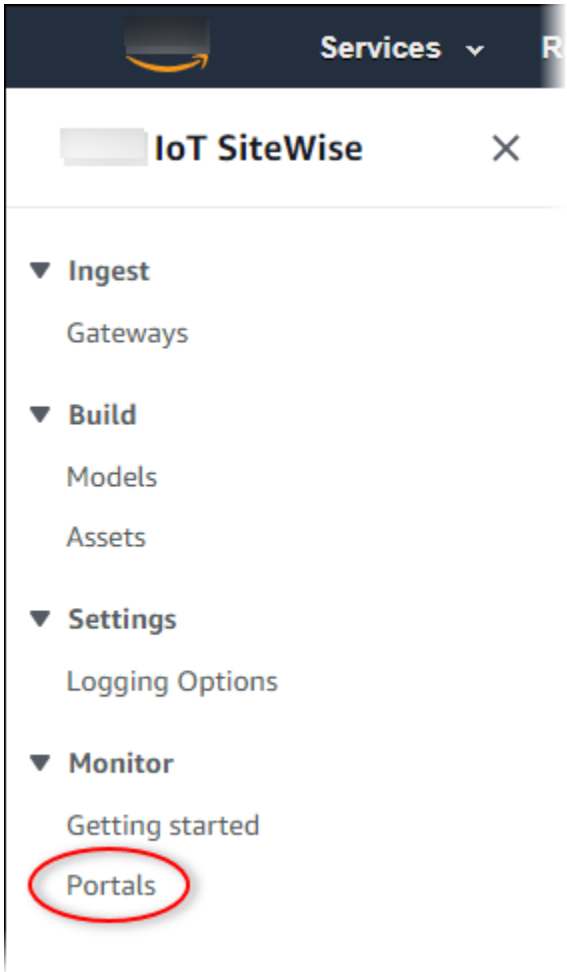
在边缘启用传送门

1. 在边缘配置部分，开启在边缘启用此门户。
2. 选择创建。

管理您的 SiteWise 监控器门户

您可以管理和配置门户的各个方面，包括添加和删除用户或管理员、设置用户权限和角色、自定义门户的 URL 和名称、设置支持联系信息以及向门户管理员发送电子邮件邀请。

1. 登录 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 监控、门户。



3. 选择门户，然后选择 查看详细信息（或选择门户的 名称）。
4. 您可以执行以下任一管理任务：
   - 在中更改门户详细信息 Amazon IoT SiteWise
   - 在中添加或移除门户管理员 Amazon IoT SiteWise
   - 向门户管理员发送电子邮件邀请
   - 在中添加或移除门户用户 Amazon IoT SiteWise
   - 删除中的传送门 Amazon IoT SiteWise

有关如何创建门户的信息，请参阅开始使用 Amazon IoT SiteWise Monitor（经典）。

主题

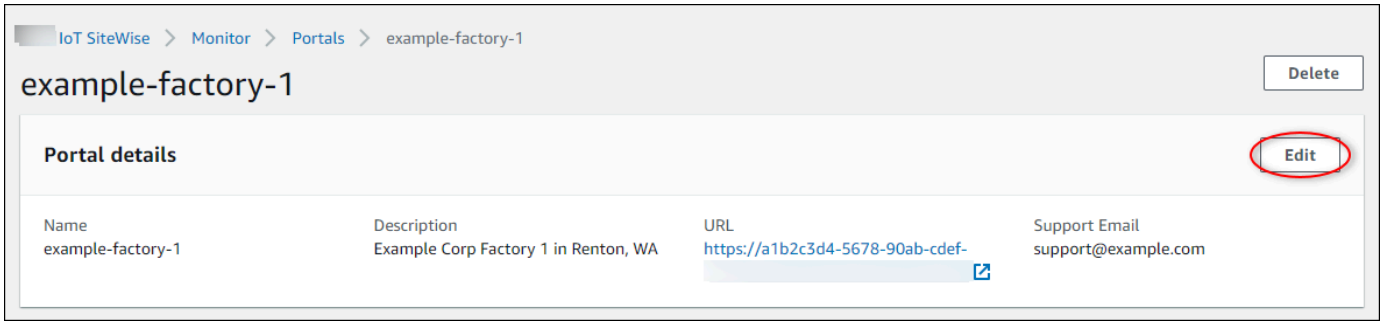
- 在中更改门户详细信息 Amazon IoT SiteWise
- 在中添加或移除门户管理员 Amazon IoT SiteWise

- 向门户管理员发送电子邮件邀请
- 在中添加或移除门户用户 Amazon IoT SiteWise
- 删除中的传送门 Amazon IoT SiteWise

在中更改门户详细信息 Amazon IoT SiteWise

您可以更改门户的名称、描述、品牌、支持电子邮件和权限。

1. 在门户详细信息页面的门户详细信息部分，选择 编辑。

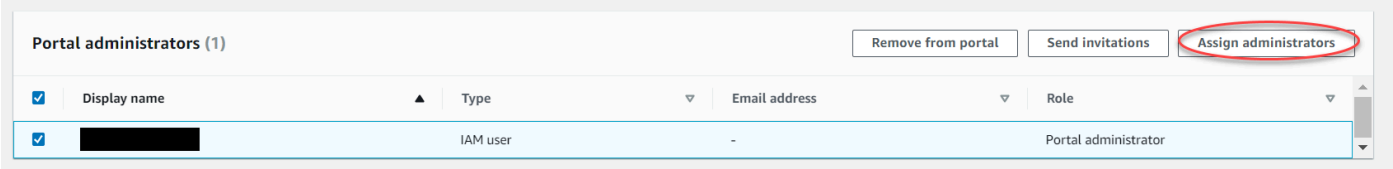


2. 更新 名称、描述、门户品牌、支持联系人电子邮件 或 权限。
3. 完成后，选择 保存。

在中添加或移除门户管理员 Amazon IoT SiteWise

通过几个步骤，您可以添加或删除作为门户管理员的用户。根据用户身份验证服务，请选择以下选项之一。

IAM

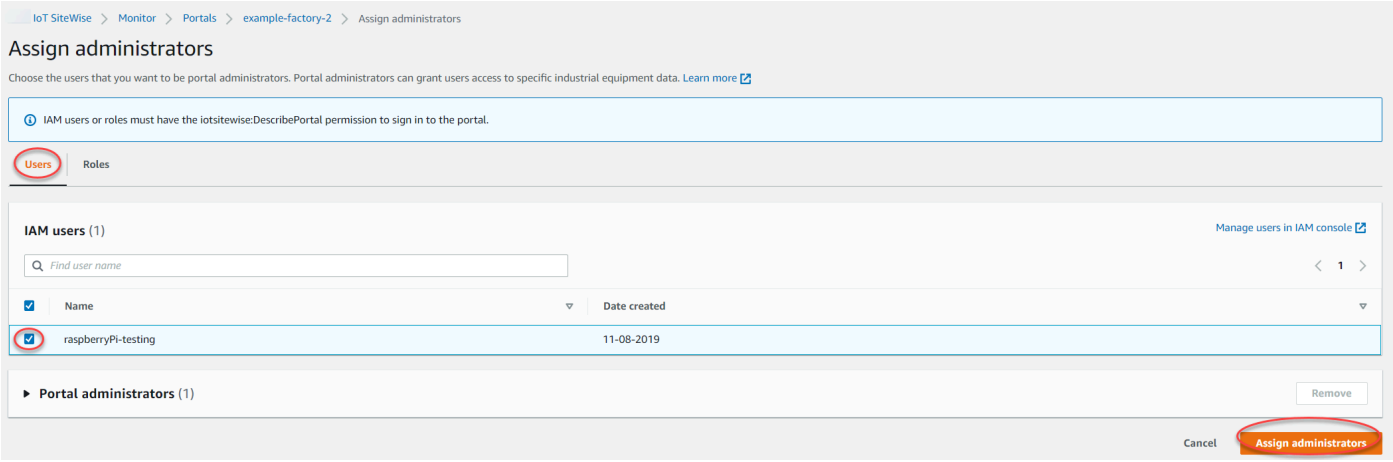


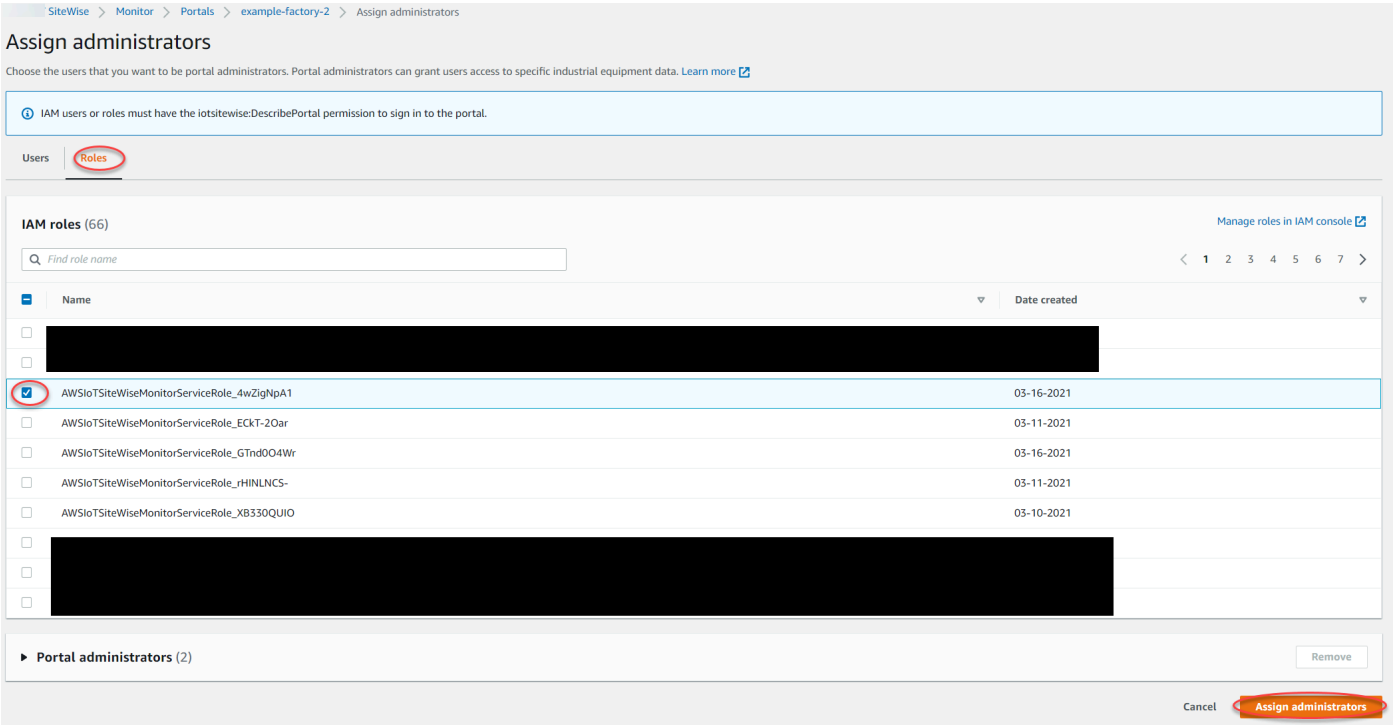
添加门户管理员

1. 在门户详细信息页面的门户管理员部分，选择分配管理员。
2. 在分配管理员页面上，执行以下操作：
   - 如果您想添加IAM 用户作为门户管理员，请选择 IAM 用户。
   - 如果您想以门户管理员身份添加 IAM 角色，请选择 IAM 角色。

3. 选中您希望作为门户管理员的用户或角色对应的复选框。这会将用户或角色添加到门户管理员列表中。
4. 选择分配管理员。

> **Important**
>
> 用户或角色必须具有登录门户的 `iotsitewise:DescribePortal` 权限。





删除门户管理员

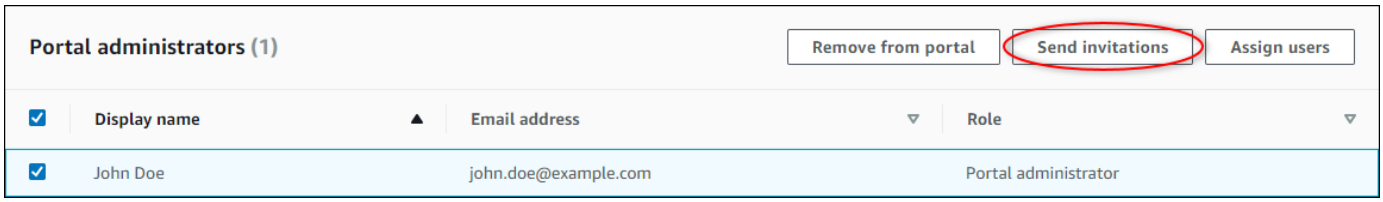
- 在门户详细信息页面的 门户管理员 部分，选中要删除的各个用户的复选框，然后选择 从门户中删除。

> **Note**
>
> 不建议不为门户添加管理员。

向门户管理员发送电子邮件邀请

向门户管理员发送电子邮件邀请。

1. 在门户详细信息页面的 门户管理员 部分，选中门户管理员对应的复选框。



2. 选择 发送邀请。您的电子邮件客户端将打开，邮件正文中将填充邀请内容。

   在将电子邮件发送给门户管理员之前您可以对其进行自定义。

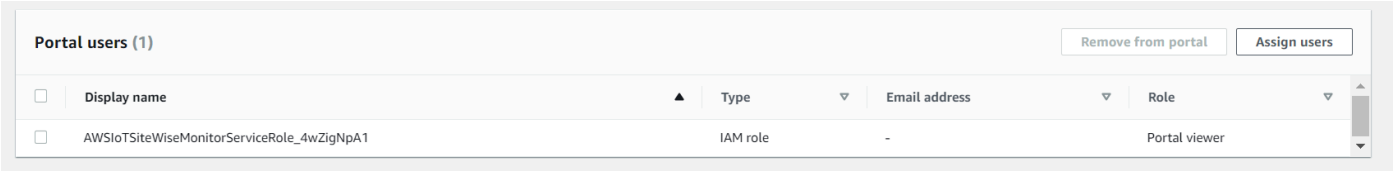
在中添加或移除门户用户 Amazon IoT SiteWise

您选择哪些用户可以访问您的门户。门户用户显示在 Monito SiteWise r 门户中的用户列表中。从此列表，门户管理员可以添加项目拥有者，而项目拥有者可以添加项目查看者。

> **Note**
>
> 如果您的门户管理员和门户用户需要添加或删除用户，可能会通过门户的支持电子邮件与您联系。

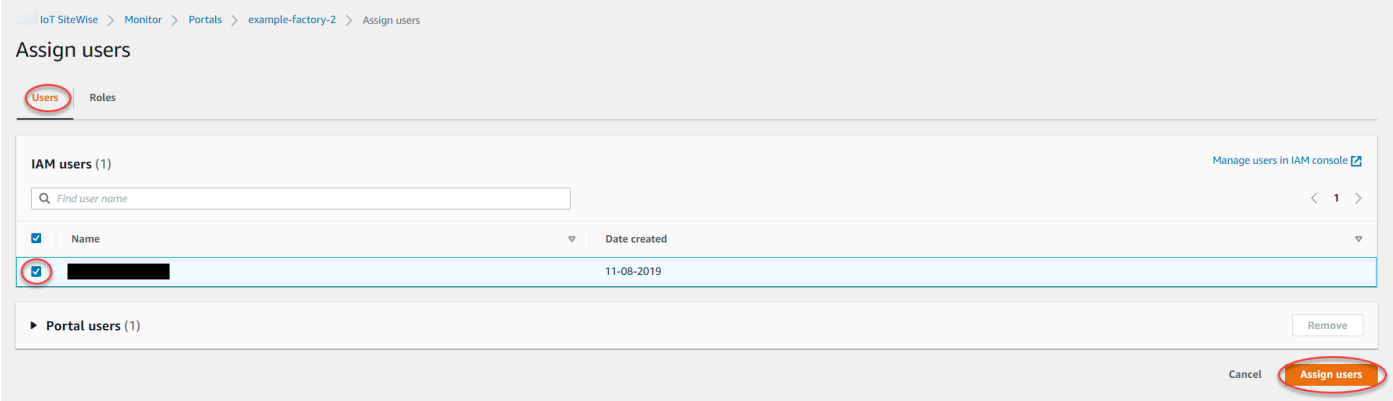
根据用户身份验证服务，请选择以下选项之一。

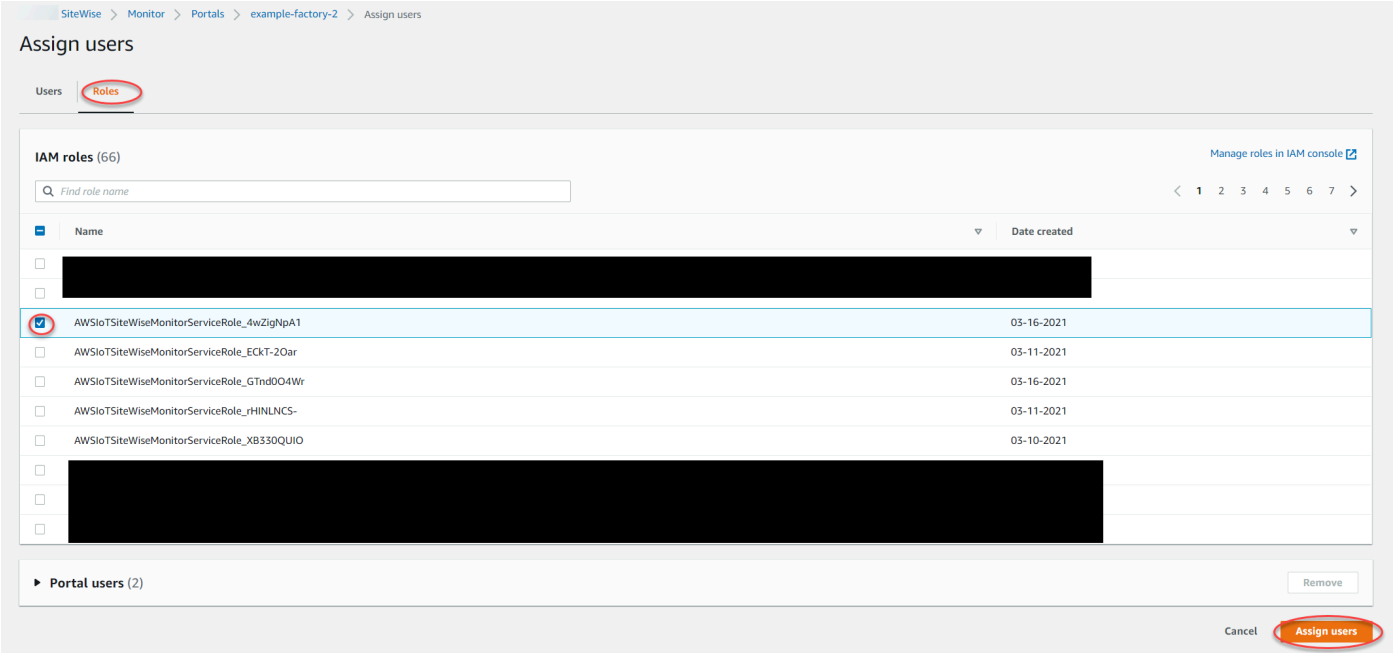
IAM



添加门户用户

1. 在门户详细信息页面的 门户用户 部分，选择 分配用户。
2. 在分配用户页面上，执行以下操作：
   - 选择 IAM 用户以将 IAM 用户添加为您的门户用户。
   - 选择 IAM 角色以将 IAM 角色添加为您的门户用户。
3. 选中您希望作为门户用户添加的用户或角色对应的复选框。这会将用户或角色添加到门户用户列表中。
4. 选择 分配用户。





删除门户用户

- 在门户详细信息页面的门户用户部分，选中要从门户中删除的用户对应的复选框，然后选择从门户中删除。

> **Important**
>
> 用户或角色必须具有登录门户的 `iotsitewise:DescribePortal` 权限。

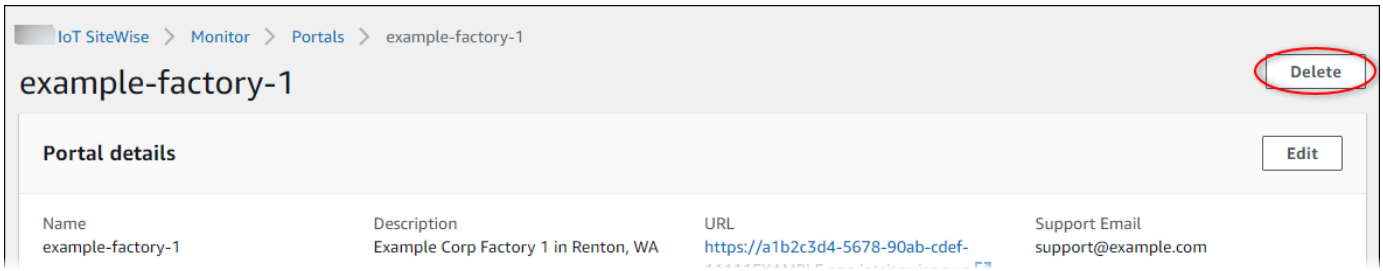
删除中的传送门 Amazon IoT SiteWise

如果您是出于测试目的创建门户，或者您重复创建了已经创建的门户，您可以删除该门户。

> **Note**
>
> 您必须先手动删除门户中的所有控制面板和项目，然后才能删除门户。有关更多信息，请参阅《SiteWise 监控器应用程序指南》中的删除项目和删除仪表板。

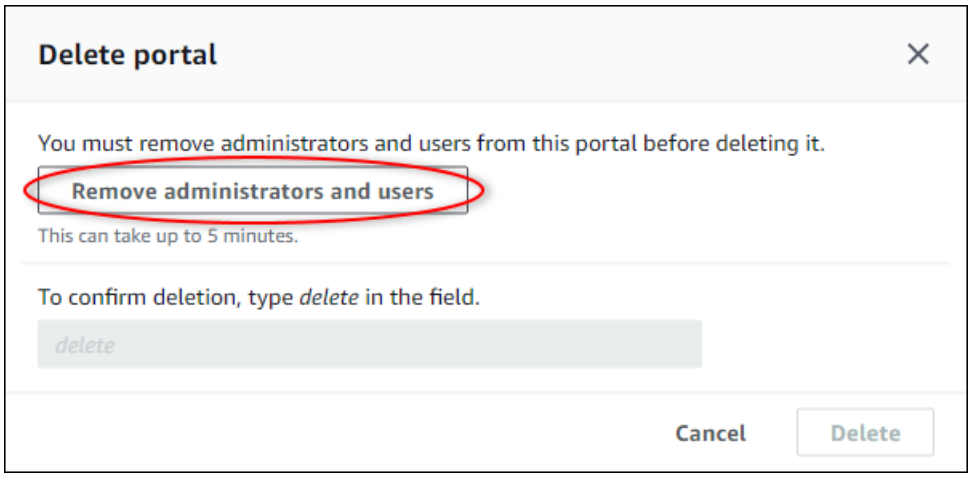
1. 在门户详细信息页面上，选择 删除。

> **Important**
>
> 删除门户时，您将丢失门户包含的所有项目以及每个项目中的所有控制面板。此操作无法撤消。您的资产数据不受影响。

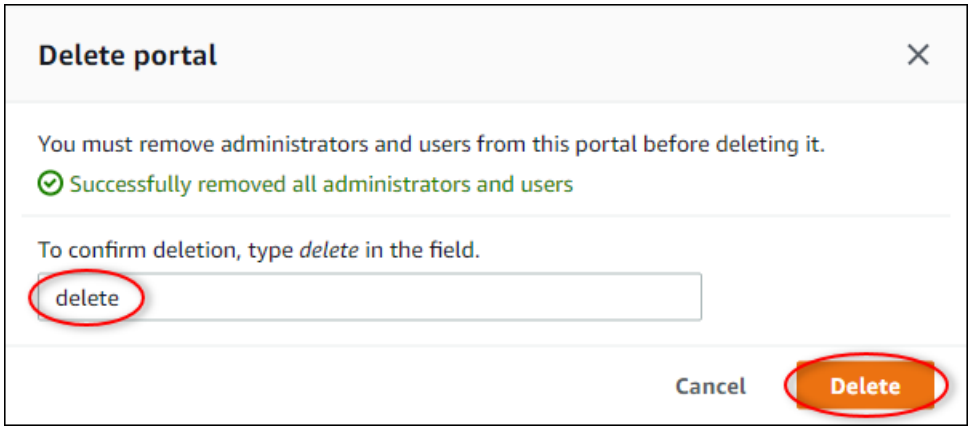


2. 在删除门户对话框中，选择删除管理员和用户。

   您必须先从门户中删除管理员和用户，然后才能删除门户。如果门户没有管理员或用户，则不会显示按钮，您可以跳到下一步。



3. 如果您确定要删除整个门户，请在字段中输入 **`delete`** 以确认删除。



4. 选择 Delete（删除）。

入门 Amazon IoT SiteWise Monitor （AI 感知）-预览

作为组织的 Amazon 管理员，您可以通过 Amazon IoT SiteWise 控制台创建门户，使组织成员能够查看您的 Amazon IoT SiteWise 数据。完成以下步骤即可开始使用。

1. 配置和创建门户。
2. 添加门户管理员并发送邀请电子邮件。
3. 添加门户用户。

创建门户后，门户管理员可以创建项目并将用户添加到项目中。然后，项目成员创建仪表板以可视化连接的数据 Amazon IoT SiteWise，使他们能够监控连接的设备、流程和设备的性能。

> **Note**
>
> 向门户添加用户或管理员时，请避免创建限制用户权限 Amazon Identity and Access Management （例如受限 IP）的 (IAM) 策略。任何权限受限的附加策略都将无法连接到 Amazon IoT SiteWise 门户。

创建要与团队共享的项目。然后，项目所有者可以创建仪表板来可视化资产的属性，从而帮助项目查看者了解设备、流程和设备的性能。它还为您的团队提供一致的运营视图。

仪表板有助于可视化和理解您的项目数据。它可以帮助企业和应用程序用户跟踪他们的 Amazon IoT 设备和数据。根据您的需求，选择最能显示数据的可视化类型。重新排列可视化并调整其大小，以创建适合您团队的布局。探索您的设备、流程和设备资产和数据，快速发现问题并提高运营效率。

主题

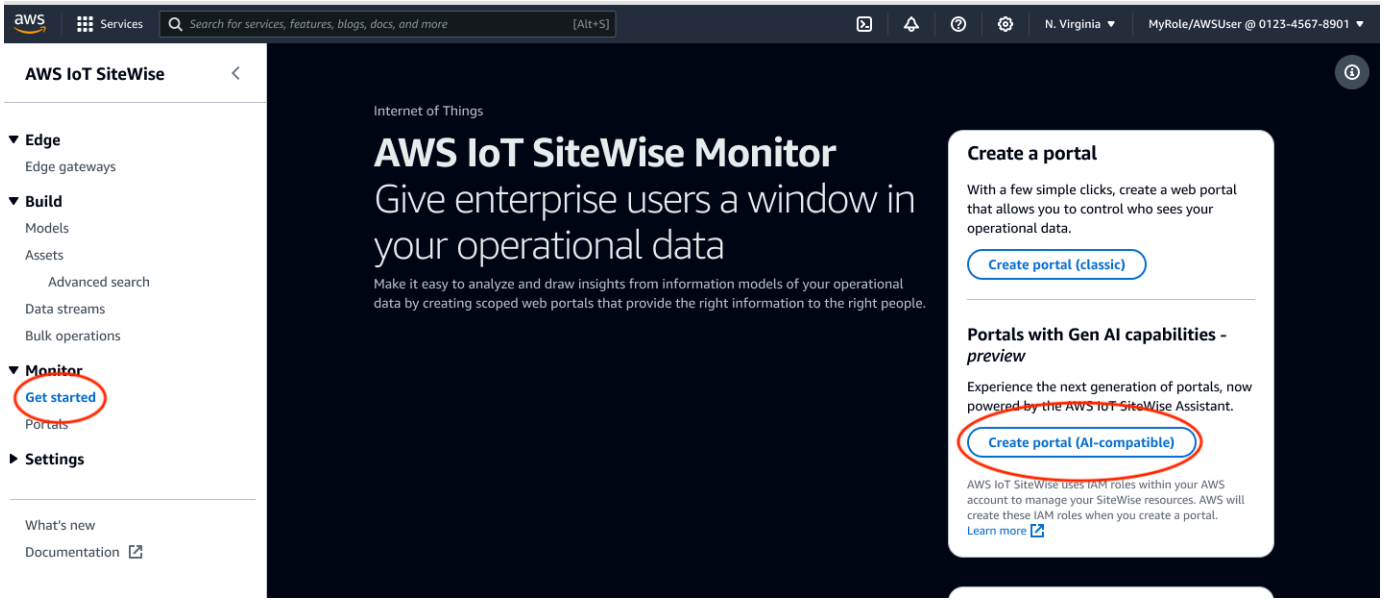
- 创建门户
- 配置门户
- 管理门户
- 删除门户
- 使用创建仪表板 Amazon CLI
- 门户网站登录
- 创建项目
- 更新项目
- 删除项目
- 创建控制面板
- 更新控制面板
- 删除控制面板
- 配置控制面板

创建门户

您可以在 Amazon IoT SiteWise 控制台中创建 SiteWise 监控器门户。

创建门户

1. 登录 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择监控，开始使用。
3. 选择创建门户（AI 感知）。



接下来，您必须提供一些基本信息来配置门户。

配置门户

用户使用门户查看您的数据。您可以自定义门户的名称、描述、品牌、、支持联系人电子邮件和权限。

配置门户的步骤：

1. 输入门户的名称。
2. （可选）输入门户的描述。如果您有多个门户，请使用有意义的描述来帮助您跟踪每个门户包含的内容。
3. （可选）上传图片以在门户中显示您的品牌。选择一个正方形 PNG 图片。如果上传非方形图片，门户会将图片缩减为正方形。
4. 在 Support 联系人邮箱中输入支持问题的电子邮件地址。
5. 在用户身份验证框中，选择以下选项：
   - 如果您的门户用户使用其公司用户名和密码登录此门户，请选择 IAM Identity Center。

     如果您尚未在账户中启用 IAM Identity Center，请执行以下操作：

     a. 选择创建用户。

     b. 在 创建用户页面上，要创建第一个门户，请输入用户的电子邮件地址、名字和姓氏，然后选择创建用户。

> **Note**
>
> 对 IAM 凭证的 Support 即将推出。

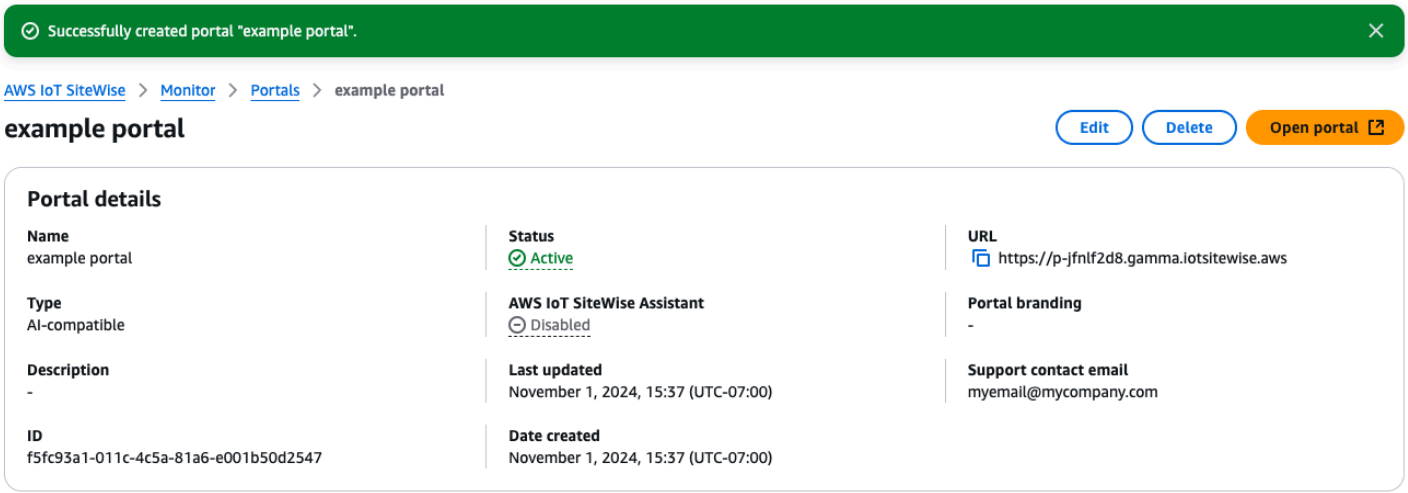
6. 在 “服务访问权限” 部分中选择以下选项之一：
   - 请选择创建和使用新的服务角色。默认情况下，M SiteWise onitor 会自动为每个门户创建一个服务角色。此角色允许您的门户用户访问您的 Amazon IoT SiteWise 资源。有关更多信息，请参阅 将服务角色用于 Amazon IoT SiteWise Monitor。
   - 选择使用现有服务角色，然后选择目标角色。
7. 选择启用此门户网站的 Amazon IoT SiteWise 助手。As Amazon IoT SiteWise sistant 提供快速的数据分析、实时见解和指导性建议。

> **Note**
>
> 启用 Amazon IoT SiteWise 助手将产生费用。要使用企业级知识解决方案和指南，您必须拥有与 Amazon Kendra 索引关联的数据集。

8. （可选）为门户添加标签。有关更多信息，请参阅 标记您的 Amazon IoT SiteWise 资源。
9. 选择创建门户。 Amazon IoT SiteWise 将创建您的门户。

> **Note**
>
> 如果关闭控制台，您可以通过添加管理员和用户来完成设置过程。有关更多信息，请参阅 添加或删除门户管理员。如果您不想保留此门户，请将其删除，以免使用资源。有关更多信息，请参阅 删除门户。

门户创建时将显示一条消息。



创建门户后，它将列在 “门户” 部分中。门户详细信息部分列出了每个门户的名称、描述、ID、URL、状态、上次更新和创建日期、门户品牌和支持电子邮件。

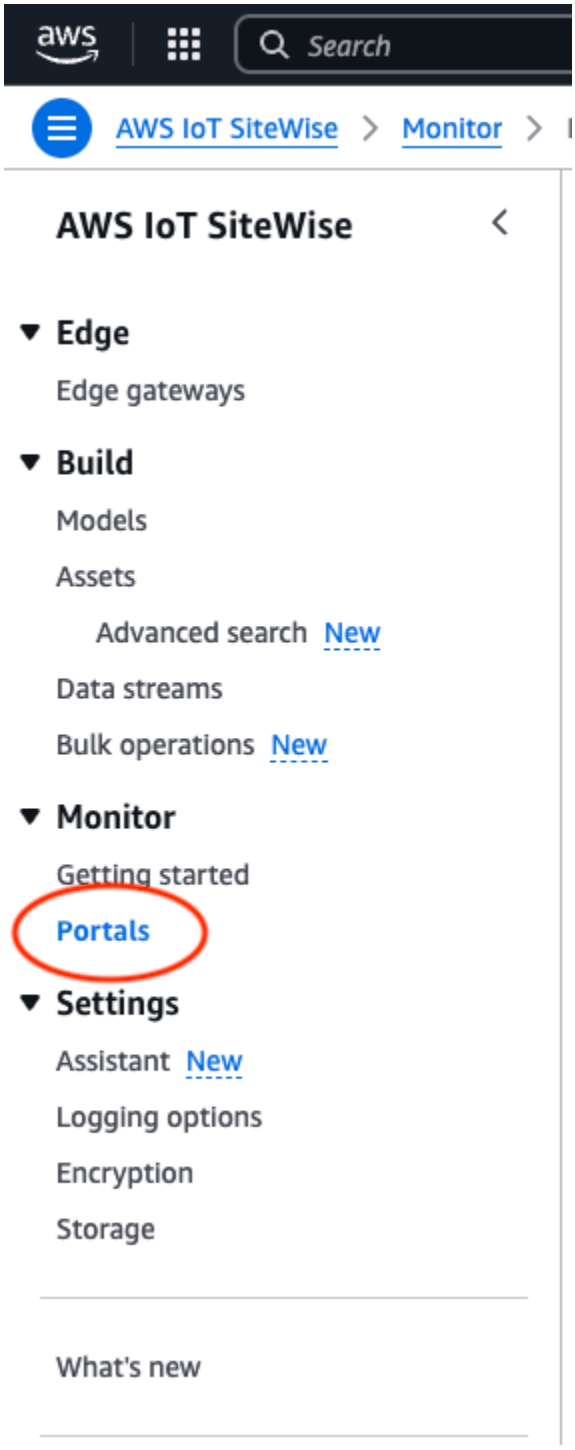
状态列可以是以下值之一。

- C@@ RE AT Amazon IoT SiteWise ING-正在处理您创建门户的请求。此过程可能需要几分钟才能完成。
- 更新- Amazon IoT SiteWise 正在处理您更新门户的请求。此过程可能需要几分钟才能完成。
- 待处理 —— Amazon IoT SiteWise 正在等待 DNS 记录传播完成。此过程可能需要几分钟完成。当门户的状态为 PENDING 时，您可以将其删除。
- 删除- Amazon IoT SiteWise 正在处理您删除门户的请求。此过程可能需要几分钟才能完成。
- 活动- 当门户变为活动状态时，您的门户用户可以访问它。
- 失败- Amazon IoT SiteWise 无法处理您创建、更新或删除门户的请求。如果您启用 Amazon IoT SiteWise 向 Amazon Logs 发送 CloudWatch 日志，则可以使用这些日志来解决问题。有关更多信息，请参阅Amazon IoT SiteWise 使用 CloudWatch 日志进行监控。

管理门户

您可以管理和配置门户的各个方面，这包括添加和移除管理员、设置权限和角色、自定义姓名、描述、设置支持电子邮件以及邀请门户管理员。

1. 登录 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 监控、门户。



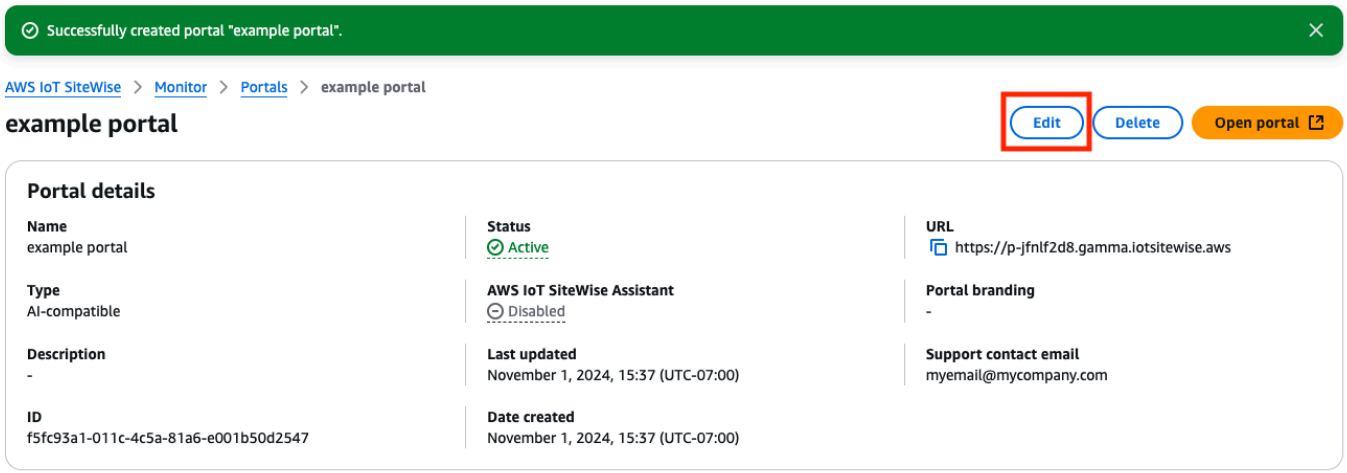
3. 选择一个入口，然后选择打开门户（或选择门户的名称）。
4. 您可以执行以下任一管理任务：
   - 编辑入口属性
   - 添加或删除门户管理员

- 向门户管理员发送电子邮件邀请
- 删除中的传送门 Amazon IoT SiteWise

编辑入口属性

您可以更改门户的名称、描述、品牌、支持电子邮件和服务访问权限。

1. 在门户详细信息页面的门户详细信息部分，选择 编辑。

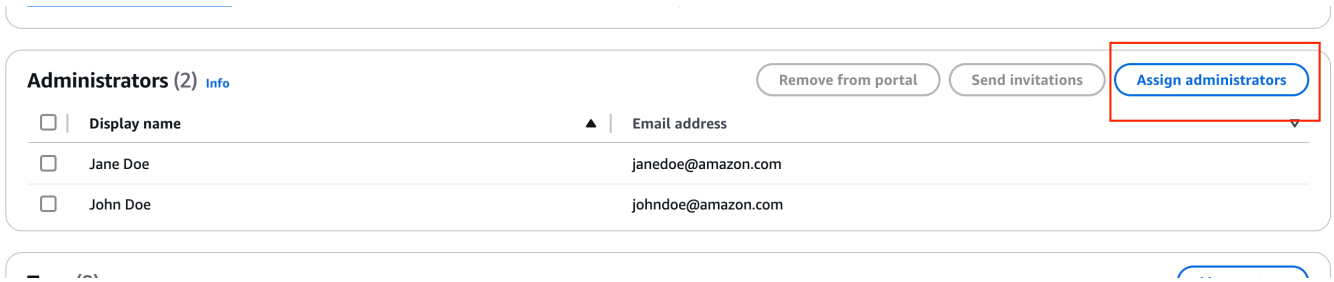


2. 更新名称、描述、Portal 品牌、Su pport 联系人电子邮件、Amazon IoT SiteWise 助手或服务访问权限。
3. 完成后，选择保存更改。

添加或删除门户管理员

通过几个步骤，您可以添加或删除作为门户管理员的用户。根据用户身份验证服务，请选择以下选项之一。

IAM Identity Center

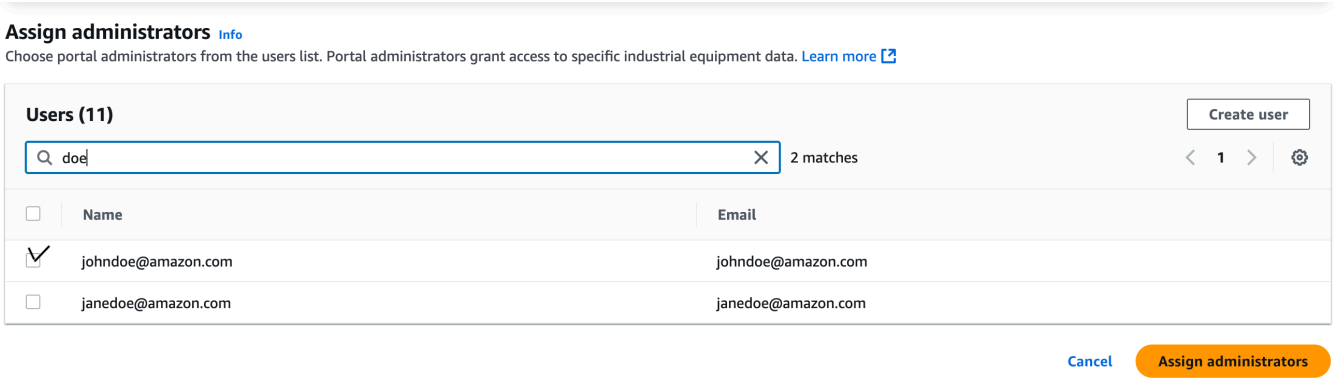


添加门户管理员

1. 在门户详细信息页面的管理员部分，选择分配管理员。
2. 在分配管理员页面上，选择要作为管理员添加到门户的用户。

> **Note**
>
> 如果您使用 IAM Identity Center 作为身份存储，并且已登录 Amazon Organizations 管理账户，则可以选择创建用户来创建 IAM Identity Center 用户。IAM Identity Center 会向新用户发送一封电子邮件，让他们设置密码。然后，您可以将该用户作为管理员分配到门户。有关更多信息，请参阅管理 IAM Identity Center 中的身份。

3. 选择分配管理员。



删除门户管理员

- 在门户详细信息页面的 门户管理员 部分，选中要删除的各个用户的复选框，然后选择 从门户中删除。

> **Note**
>
> 管理员 (#) 列出了门户的管理员人数。您可以添加多个门户管理员来管理和处理项目。

向门户管理员发送电子邮件邀请

向门户管理员发送电子邮件邀请。

1. 在门户详细信息页面的管理员部分，选中门户管理员对应的复选框。

2. 选择 发送邀请。您的电子邮件客户端将打开，邮件正文中将填充邀请内容。

   在将电子邮件发送给门户管理员之前您可以对其进行自定义。

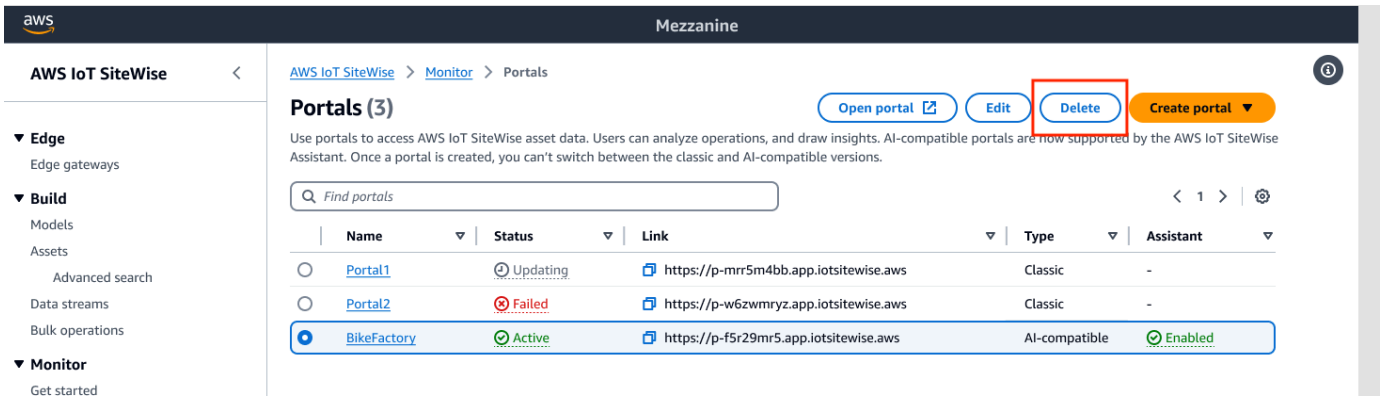
删除门户

如果您是出于测试目的创建门户，或者您重复创建了已经创建的门户，您可以删除该门户。

> **Note**
>
> 您必须先手动删除门户中的所有控制面板和项目，然后才能删除门户。

1. 在门户详细信息页面上，选择 删除。

   > **Important**
   >
   > 删除门户时，您将丢失门户包含的所有项目以及每个项目中的所有控制面板。此操作无法撤消。您的资产数据不受影响。



2. 在删除门户对话框中，选择移除管理员和用户。

   您必须先从门户中删除管理员和用户，然后才能删除门户。如果门户没有管理员或用户，则不会显示按钮，您可以跳到下一步。

3. 如果您确定要删除整个门户，请在字段中输入 **`confirm`** 以确认删除。

4. 选择删除。

使用创建仪表板 Amazon CLI

使用在仪表板中定义可视化（或微件）时 Amazon CLI，必须在 `dashboardDefinition` JSON 文档中指定以下信息。此定义是CreateDashboard和UpdateDashboard操作的参数。

`displaySettings`

具有以下参数的显示设置：

- `numRows`-仪表板布局中的行数。每行都宽度为单元格大小。
- `numColumbs`-仪表板布局中的列数。每列的单元格大小都很宽。
- `cellSize`—（可选）布局中单元格的大小（以像素为单位）。它必须是正数。默认值为 10。
- `significantDigits`—（可选）要在控制面板中显示的有效数字数。默认值为 4。

`querySettings`

带有以下参数的查询信息：

- `refreshRate`—（可选）以毫秒为单位的数据刷新速率。接受以下值——1000、5000、10000、60000、300000。

`defaultViewport`

如果未提供，则默认为最后五分钟。包含以下参数：

- `duration`—（可选）确定从现在开始查询过去多久的数据。
- `start`—（可选）它的类型为日期。查询数据的开始时间范围。需要指定`end`日期。
- `end`—（可选）它的类型为日期。查询数据的结束时间范围。需要指定`start`日期。

`widgets`

包含以下信息的小组件定义结构列表：

`type`

控件的类型。 Amazon IoT SiteWise 提供以下控件类型：

- `xy-plot`— 折线图或散点图，具体取决于配置。
- `bar-chart` – 条形图。
- `kpi-chart`— 关键绩效指标图表。
-

status-timeline— 一种状态控件，用于可视化和浏览来自一个或多个数据源的时间序列数据。

- text— 文本控件。
- table – 表格控件。

id

小组件的唯一标识符。

x

控件的水平位置，从仪表板的左侧开始。此值表示小部件在控制面板网格中的位置。

y

小组件的垂直位置，从仪表板顶部开始。此值表示小部件在控制面板网格中的位置。

z

控件的相对顺序。如果它们重叠，则较大的 Z 值控件会显示在较低的 Z 值构件的前面。

width

控件的宽度，以仪表板上的单元格数量表示。

height

控件的高度，以仪表板上的单元格数量表示。

properties

控件的属性列表。它因小部件的类型而异。有关详细信息，请参阅 IoT 应用程序套件。

Example 示例控制面板定义

以下示例从存储在 JSON 文件中的负载定义控制面板。

```
aws iotsitewise create-dashboard \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeEXAMPLE \
  --dashboard-name "Example Dashboard" \
  --dashboard-definition file://dashboard-definition.json
```

针对 `dashboard-definition.json` 的以下 JSON 示例定义具有以下可视化小部件的控制面板：

```
{
    "displaySettings": {
        "numColumns": 200,
        "numRows": 1000,
        "cellSize": 20,
        "significantDigits": 4
    },
    "widgets": [{
        "id": "Ot73JcxUoc6oEXAMPLE",
        "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": {
```

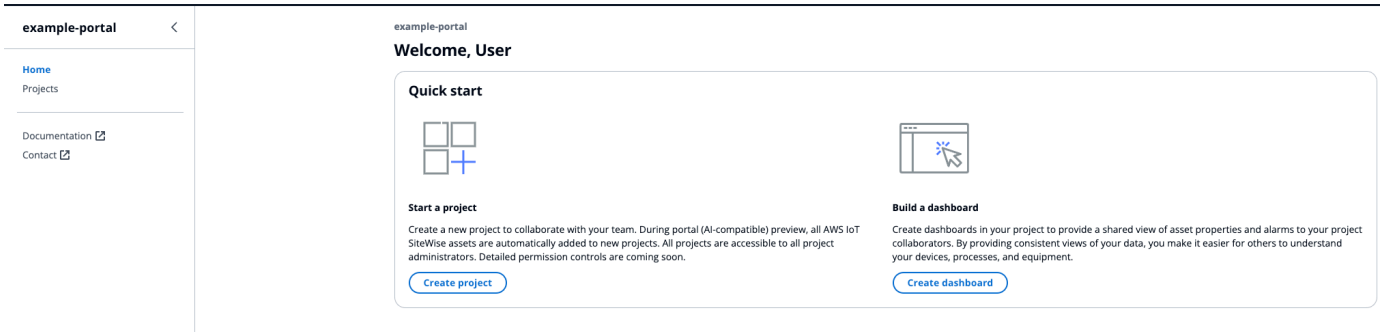
```
                "style": "filled-circle"
            },
            "axis": {
                "yVisible": true,
                "xVisible": true
            },
            "legend": {
                "visible": true,
                "position": "right",
                "width": "30%",
                "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
    }
}
```

门户网站登录

用户登录

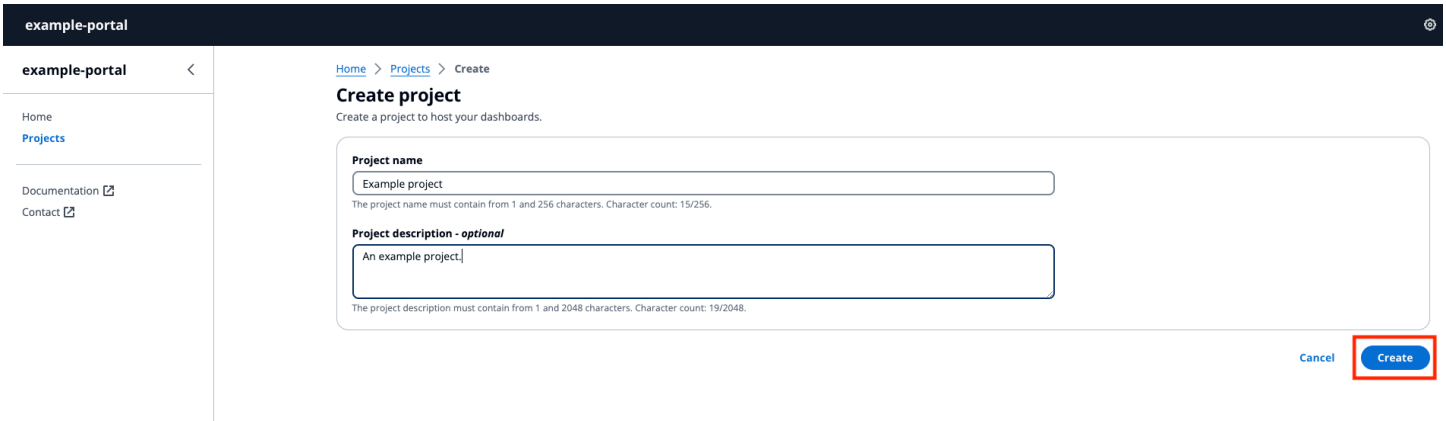
1. 在浏览器上，输入应用程序 URL。
2. 输入您的用户名和密码，然后单击 “登录” 按钮。
3. 现在，您已登录到该应用程序。



创建项目

创建项目

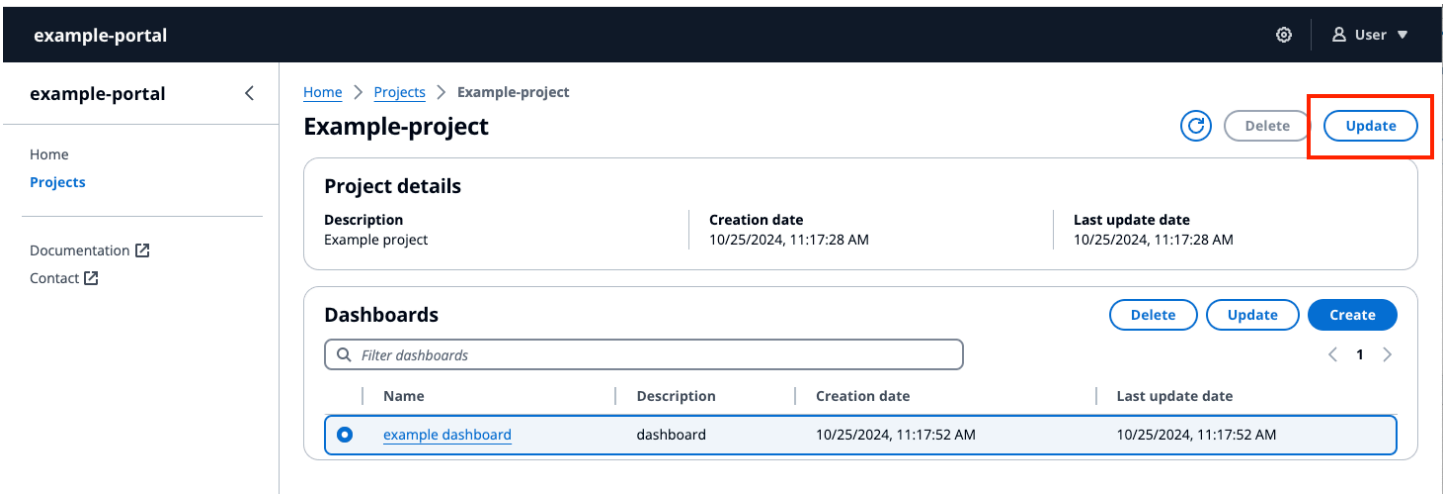
1. 项目的创建方式有两种：
   a. 在 “主页” 页面的 “快速入门” 下的 “欢迎” 部分中，选择 “创建项目”。
   b. 从左侧导航窗格中选择 “项目”。选择右上角的 “创建” 来创建项目。
2. 在 “创建项目” 部分，输入项目名称，并提供可选的描述。
3. 选择创建。



更新项目

编辑项目

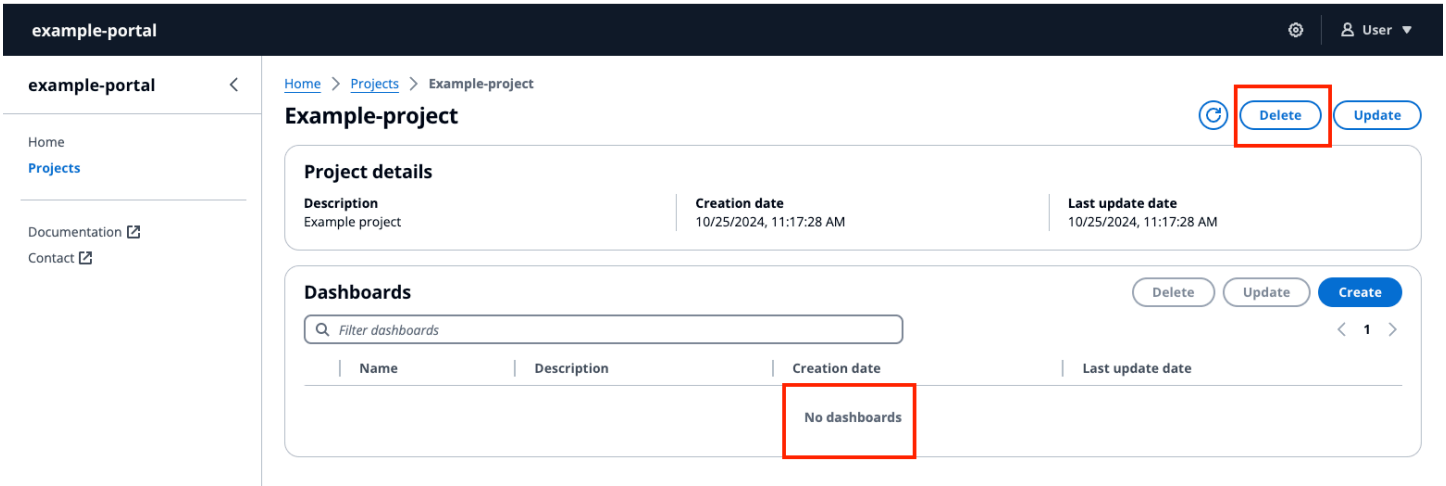
1. 选择 “项目” 页面右上角的 “更新” 按钮，编辑项目详细信息。
2. 通过编辑项目名称来更改项目名称。
3. 通过编辑描述详细信息来更改项目的描述。
4. 选择 “更新” 以保存您的更改。



删除项目

删除项目

1. 只有在项目中的所有仪表板都被删除后，才能删除该项目。
2. 选择 “项目” 页面右上角的 “删除” 按钮。
3. 再次确认您要删除该项目。
4. 选择 “删除” 以删除该项目。



创建控制面板

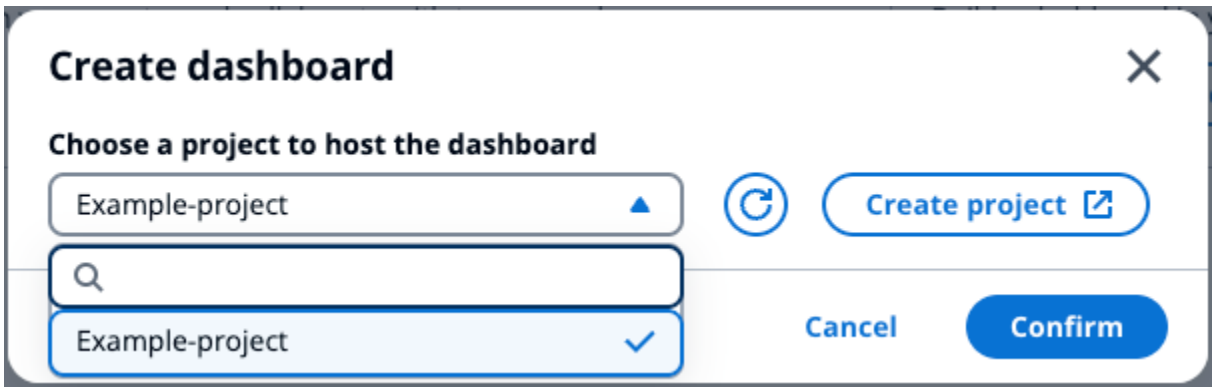
创建控制面板

1. 通过两种方式创建仪表板：

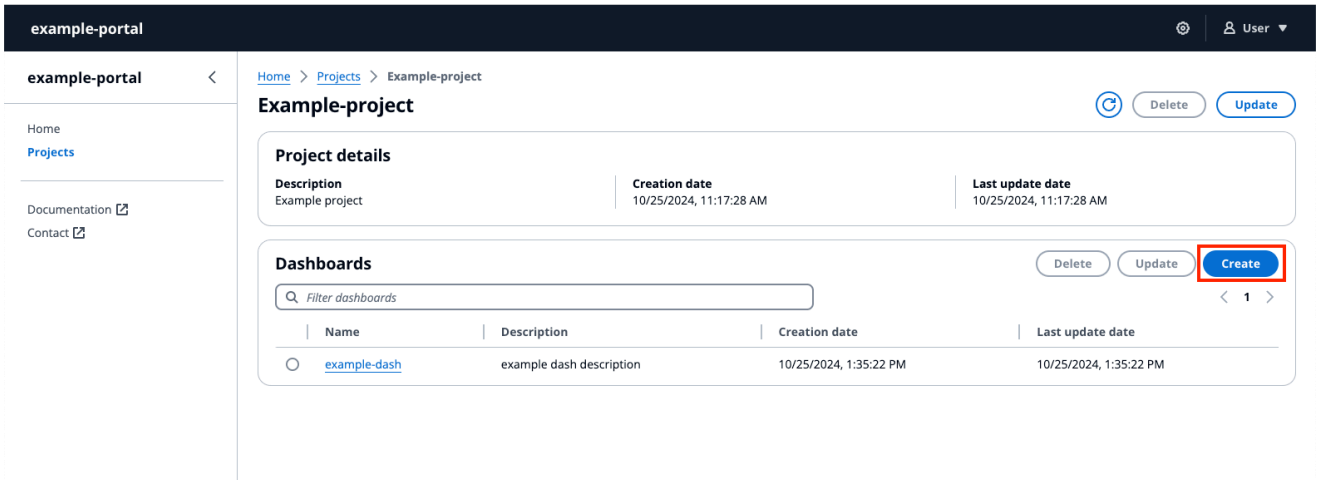
a. 从 “主页” 的 “构建仪表板” 中创建仪表板。

   i. 要在现有项目中创建仪表板，请从 “选择要托管仪表板的项目” 的下拉菜单中选择一个项目名称。

   ii. 如果您没有项目，请选择 “创建项目”，然后选择 “确认”。



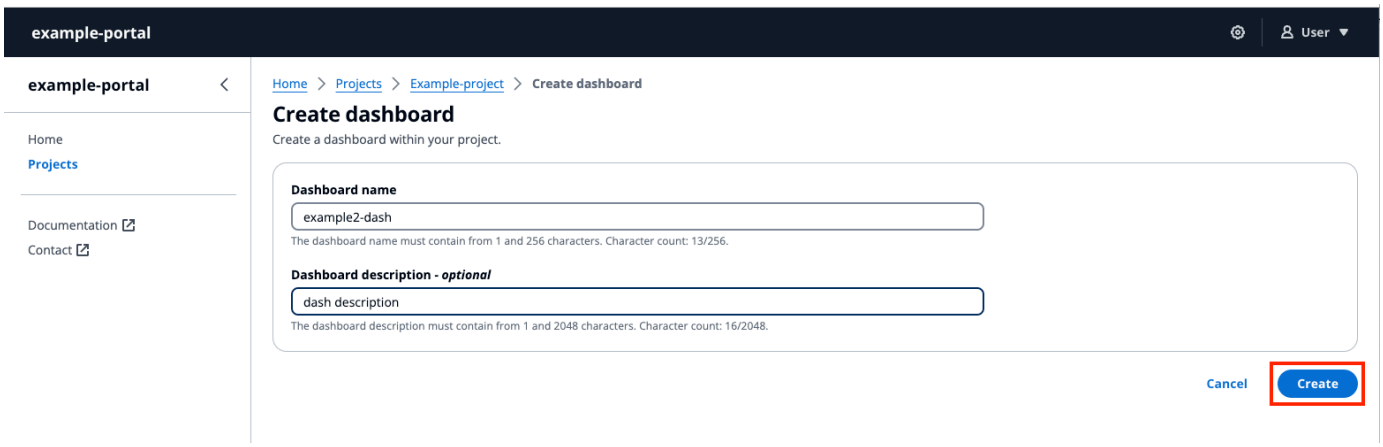
b. 在 “项目” 部分的 “仪表板” 下，根据项目创建仪表板。



2. 选择右上角的创建。

3. 输入仪表板名称，并提供可选的仪表板描述。

4. 选择创建。



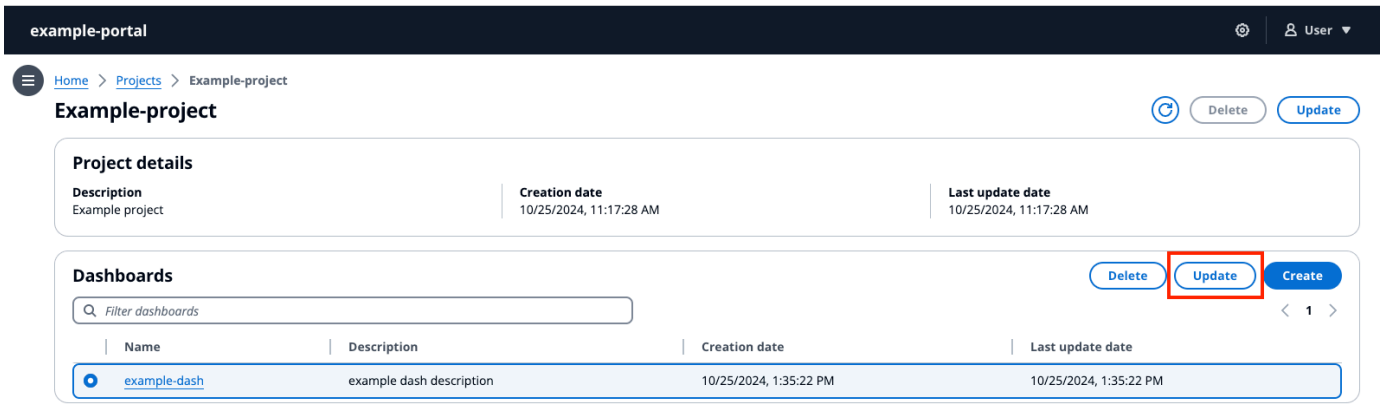
5. 配置您新创建的仪表板。

更新控制面板

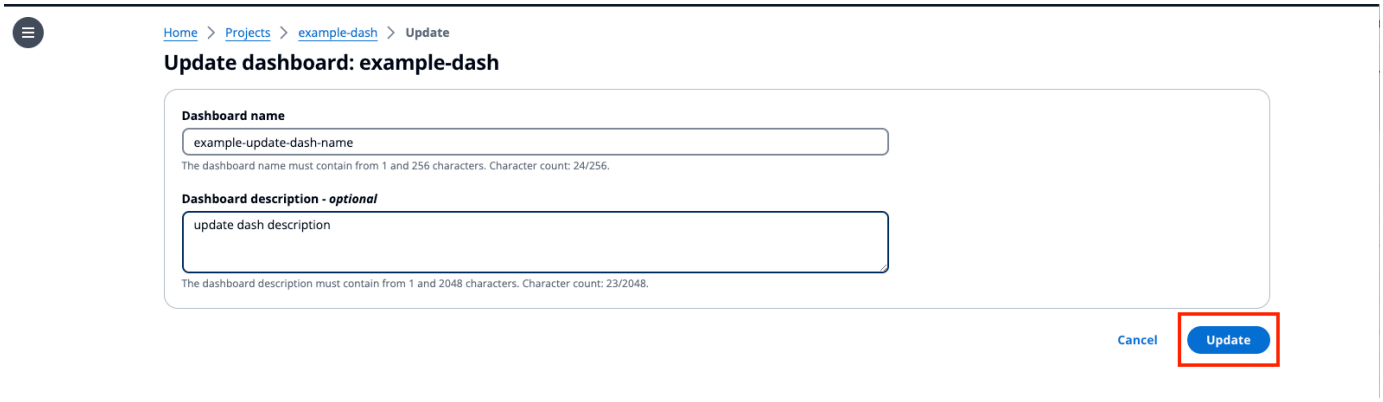
仪表板部分列出了项目中的仪表板。从列表中选择一个仪表板。

更新控制面板

1. 选择要更新的仪表板。



2. 更新仪表板名称和控制板描述（可选）。选择 “更新” 以保存更改。

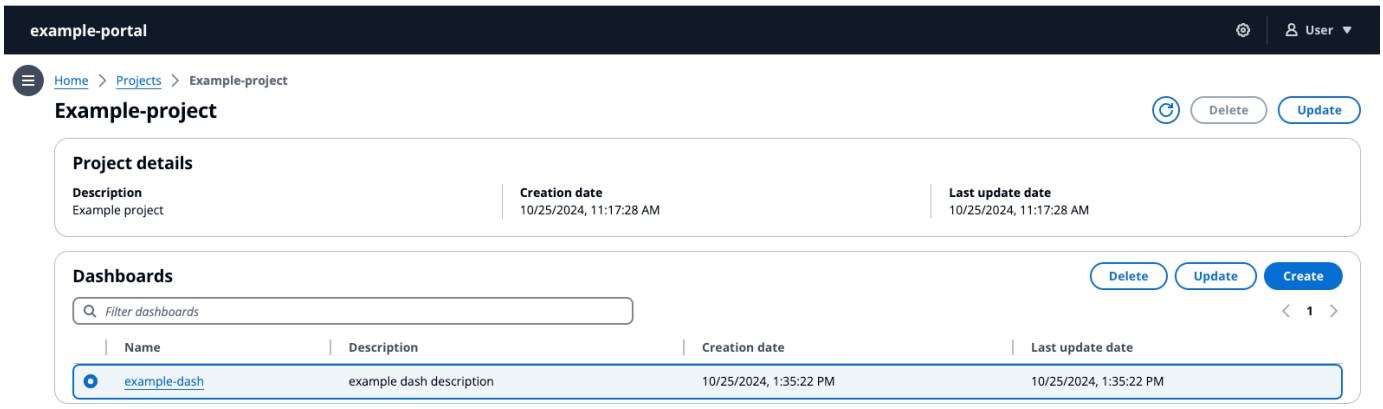


删除控制面板

仪表板部分列出了项目中的仪表板。从列表中选择一个仪表板。

删除控制面板

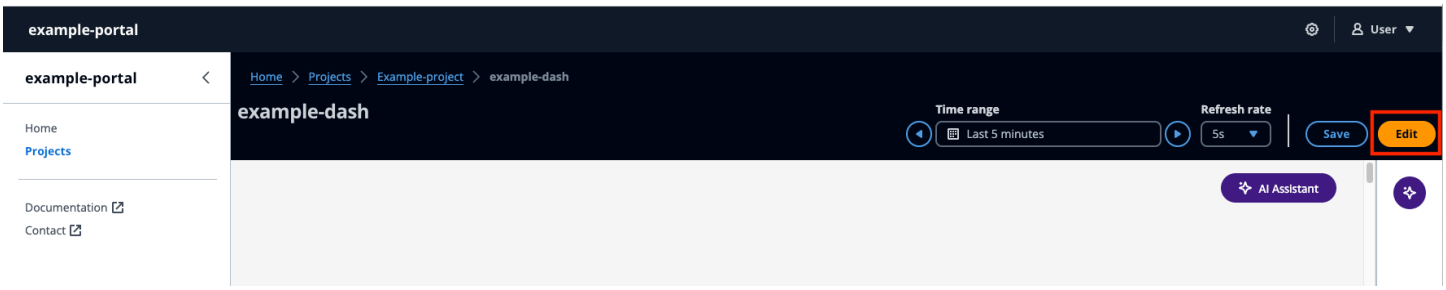
1. 选择要删除的仪表板。



2. 选择 “删除” 以删除控制面板。此操作无法撤消。

配置控制面板

仪表板部分列出了项目中的仪表板。从列表中选择一个仪表板。编辑模式允许您通过添加小组件并对其进行配置来配置仪表板。使用 “预览” 按钮可以直观显示所做的更改。



配置仪表板的步骤：

- 将不同类型的数据控件拖放到仪表板画布上以实现数据可视化。
- 通过左侧的资源管理器向所需的小组件添加数据。资源浏览器由 “建模”、“未建模” 和 “动态资产” 部分组成。按资产名称或属性名称进行搜索。选择要添加的属性，然后选择添加。
- 通过更改小组件上的配置来微调布局和样式。配置组件，包括标题、阈值和其他配置细节。
- 配置显示数据的时间范围。
  - 选择显示数据的时间范围。从右上角选择时间范围和刷新率，然后对范围进行个性化设置。从菜单中选择刷新数据的速率。
  - 使用轨迹球鼠标滚轮或右键单击，在控件上选择时间范围。这会移动显示的时间范围。
- 选择保存。

主题

- 资源浏览器
- 小组件
- 配置小部件
- 使用小部件
- 小部件中的警报
- Amazon IoT SiteWise 助手在小部件中使用
- 要问 Amazon IoT SiteWise 助手的问题示例

资源浏览器

本节介绍建模、未建模和动态资产。从三个资源中选择任意一个，然后将它们添加到您的控件中并对其进行可视化。

主题

- 建模
- 未建模
- 动态资产

建模

本节介绍选择和可视化建模资产的过程。

资产的选择

可以按如下方式查询资产：

- 搜索资产名称。使用通配符*。例如，Wind*返回以文本开头的资产名称Wind。您必须与集成Amazon IoT TwinMaker才能使用此功能。
- 默认情况下，会列出所有资产。

从列出的资产中，按名称、描述、ID 或资产模型 ID 进行筛选。选择一项资产以列出其属性（数据流）和警报。

数据流选择

数据流在 “数据流” 菜单下方列出。筛选https://docs.aws.amazon.com/iot-sitewise/最新// APIReference中按属性元数据列出的数据流。根据所选小组件选择一个或多个数据流。

- KPI 和 Ga uge 仅支持单个数据流。
- 其余控件支持具有多选功能的多个数据流。

警报选择

Amazon IoT SiteWise 警报列在 “警报数据流” 菜单下方。筛选按警报元数据列出的警报数据流。 名称、输入属性和复合模型 ID 是用于筛选的一些元数据。根据所选小组件选择一个或多个数据流。

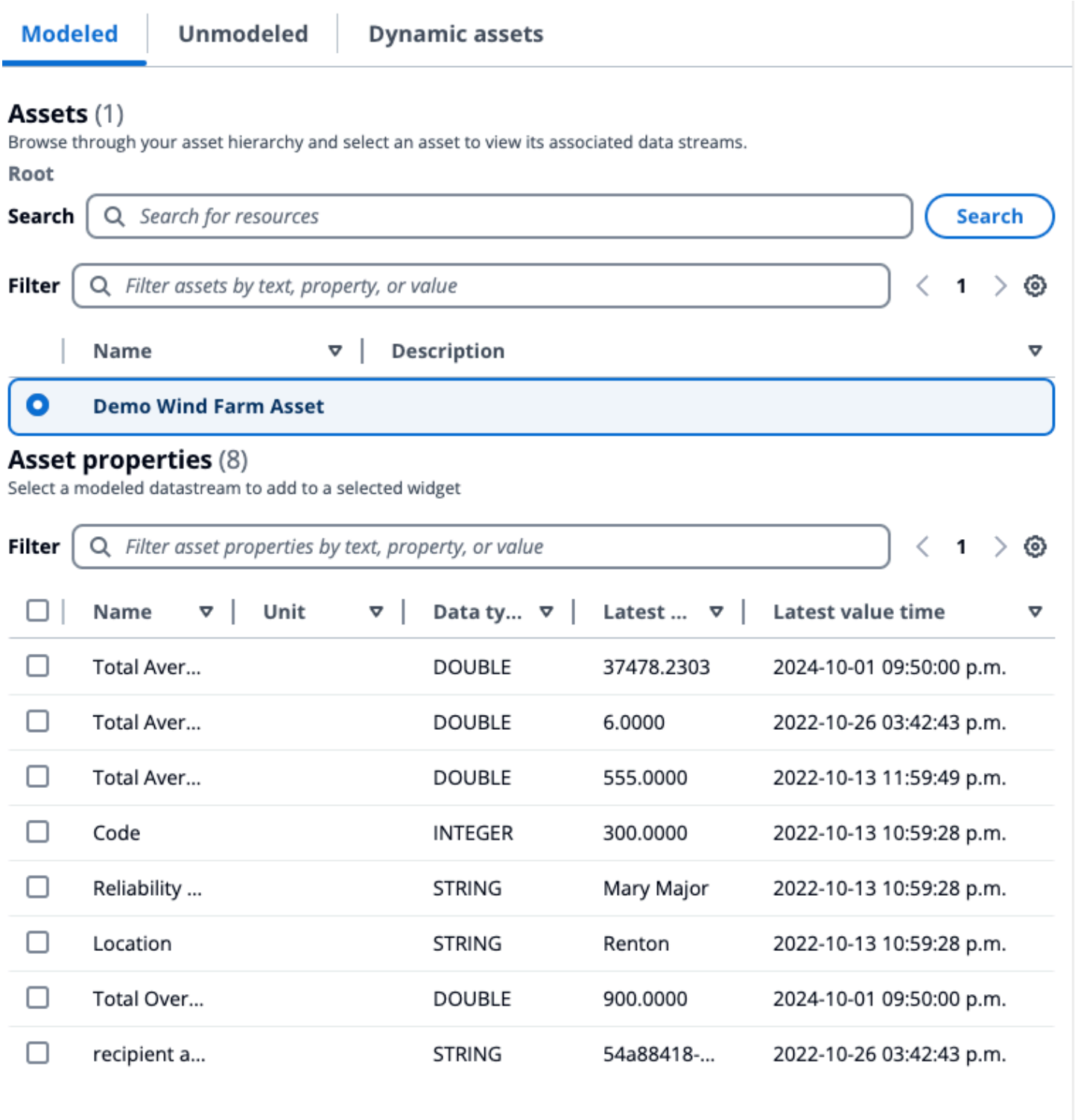
- KPI 和 G auge 仅支持单个警报。
- 其余小部件支持多选多重警报。

建模资产可视化

1. 将控件拖到画布上。为每个小组件面板选择属性以构造仪表板。

2. “筛选” 选项筛选资源以选择要可视化的资源。筛选由文本、属性或值完成。筛选是针对加载到浏览器中的资产，而不是后端筛选。
3. 搜索以列出要添加到您的控件中的资产。
4. 将资源@@ 添加到画布中的控件中。
5. 选择 “重置” 以选择其他资产，或者对所选资产进行修改。
6. 保存控制面板。在预览模式下，从下拉菜单中选择不同的资源以监控每个资源下的属性，而无需重建数据面板。

> **Note**
>
> 右侧的配置设置轮显示首选项供用户选择，例如页面大小、粘性第一列、粘滞最后一列和列首选项。自定义您的首选项，然后选择 “确认” 以应用更改。



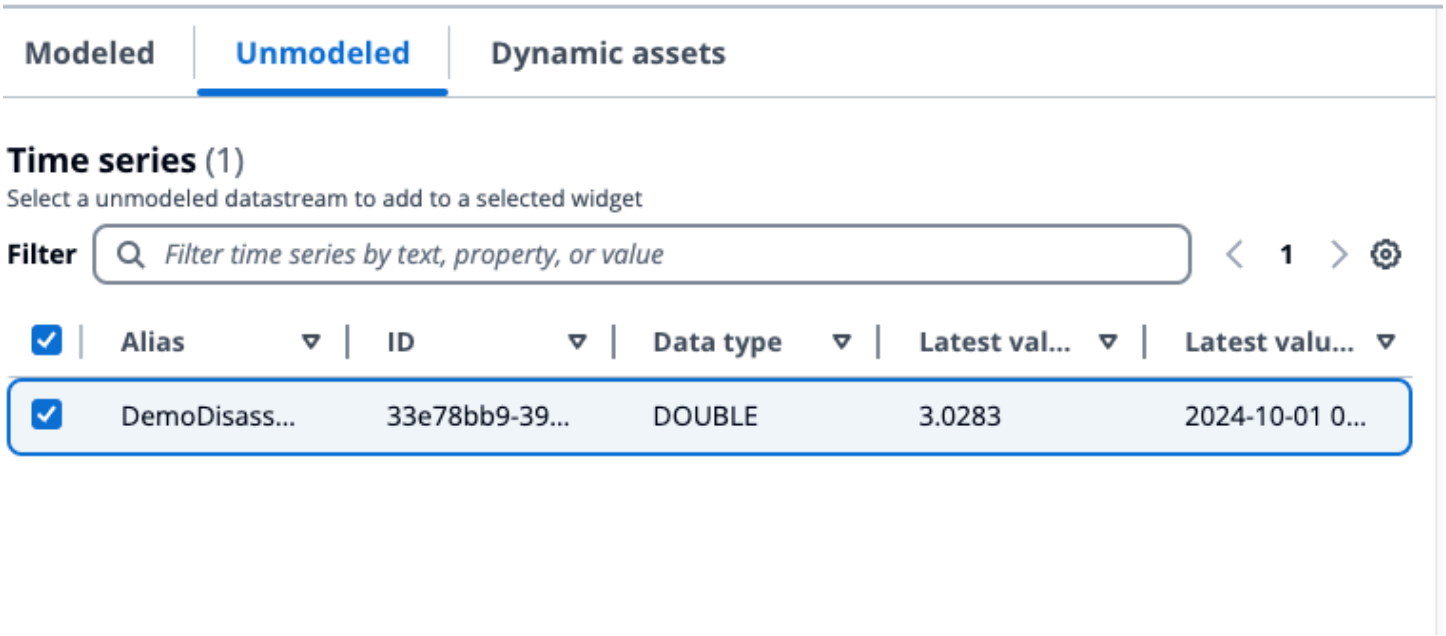
未建模

本节介绍如何搜索未建模的数据流并将其添加到微件中进行可视化。

未建模的数据流可视化

1. 将控件拖到画布上。为每个小组件面板选择属性以构造仪表板。
2. 未建模的数据流列在 “时间序列” 部分下。它们具有可自定义的属性。
3. “筛选” 选项筛选数据流以进行可视化。筛选适用于加载到浏览器的数据流，而不是后端筛选。
4. 将数据流@@ 添加到画布中的控件。
5. 选择 “重置” 以取消选择数据流。
6. 保存控制面板。在预览模式下，从下拉菜单中选择不同的资源以监控每个资源下的属性，而无需重建数据面板。

> **Note**
>
> 右侧的配置设置轮显示首选项供用户选择，例如页面大小、粘性第一列、粘滞最后一列和列首选项。自定义您的首选项，然后选择 “确认” 以应用更改。

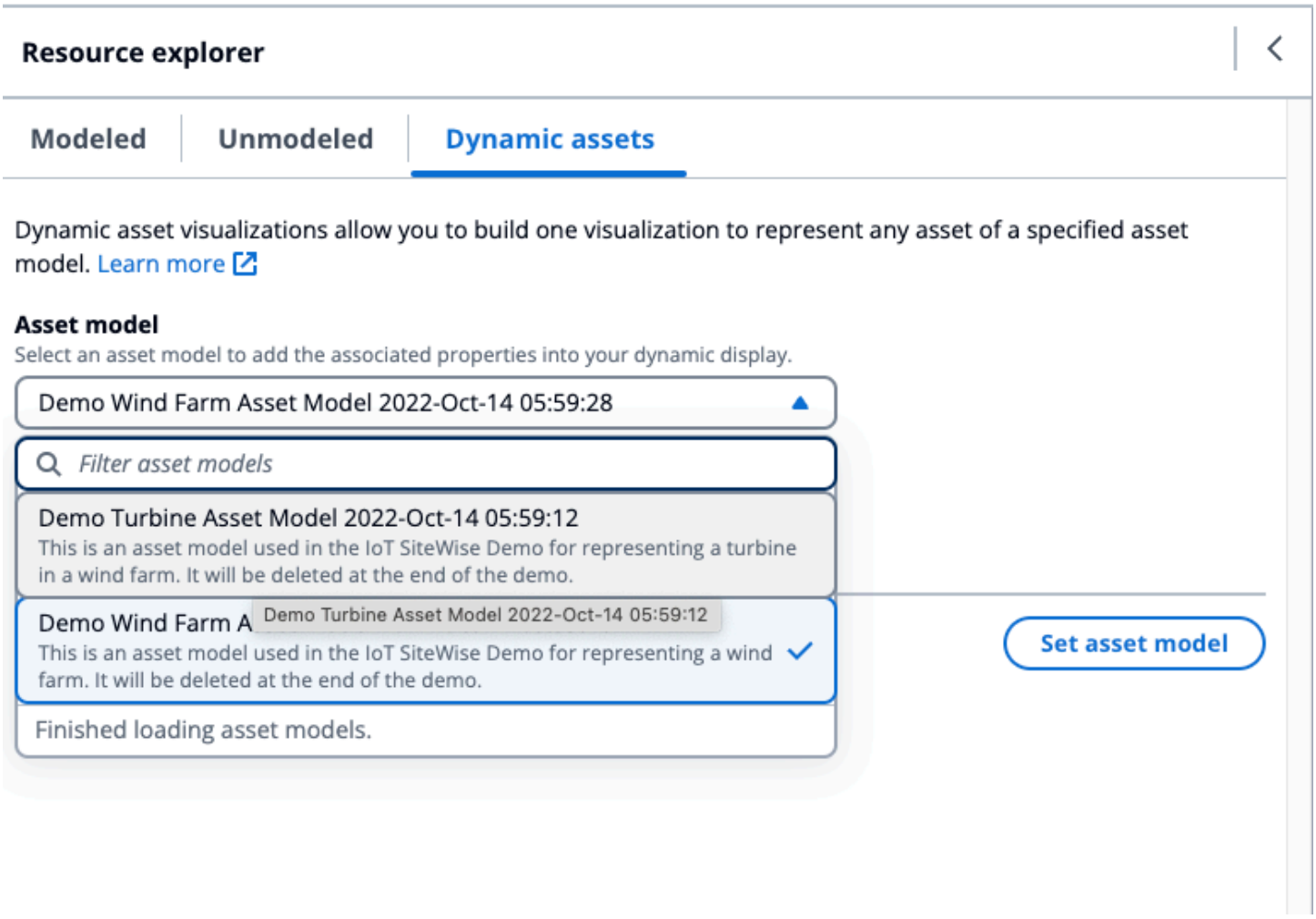


动态资产

新的 SiteWise 监控器允许客户动态切换选定资产模型的资产。您可以通过从下拉菜单中进行选择来可视化来自不同资源的属性。

动态资产可视化

1. 在资源浏览器中选择 “动态资产” 选项卡。
2. 从下拉菜单中选择要列出其资源的资源模型。
3. 从下拉菜单中选择默认资源。
4. 选择 “设置资产模型” 以选择资产模型。
5. 保存控制面板。在预览模式下，从下拉菜单中选择不同的资源来监视每个资源下的属性，而无需重建数据面板。



小组件

控件支持多种功能，包括警报、高性能直播以及与其他 IoT App Kit 组件的流畅同步。仪表板支持以下小部件：

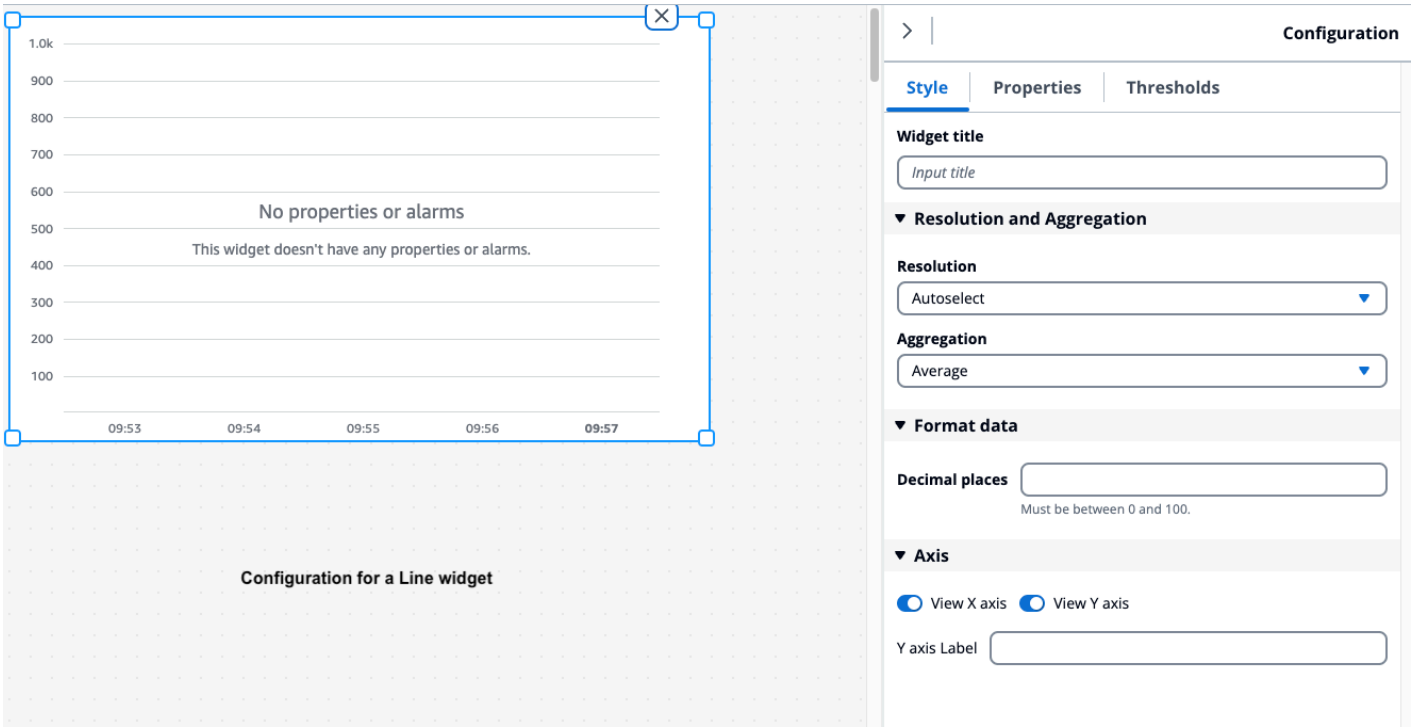
- 线 — 线控件是一个可视化控件，可显示一段时间内的趋势和变化。它由一系列数据点组成，每个数据点都由一个点或标记表示，通过直线段连接以创建折线图。它支持多种功能，包括警报、阈值、高性能直播以及与其他 IoT App Kit 组件的流畅同步。此控件可定制，可清晰简洁地传达复杂的数据。
- 条形图-条形图是一款功能强大的可视化工具，可显示时间序列数据。它支持多种功能，包括警报、高性能直播以及与其他 IoT App Kit 组件的流畅同步。
- 时间轴-时间轴控件提供了一种可视化和浏览来自数据源的时间序列数据的方法。它在时间轴上以不同的颜色显示数据流值是独一无二的。它支持一系列丰富的功能，包括警报、高性能直播以及与其他物联网应用套件组件之间的流畅同步。它最适合用于显示非数字数据类型/
- KPI — 关键绩效指标 (KPI) 组件提供了资产属性的简要概述。它支持警报和阈值。本概述提供了有关您的设备、设备和流程整体性能的重要见解。KPI 仅支持单个数据流或警报，不支持多个数据流。
- G auge — Gauge 组件提供了资产属性概述的简洁表示。它用于可视化对您的设备、设备或流程整体性能的关键见解。它在功能上与 KPI 相同，但视觉上有所不同。Gauge 显示数据流值、阈值和值范围。您可以使用 Gauge 与 Amazon IoT 来自一个或多个数据源的数据进行交互。
- 表-表组件提供了一种紧凑的表单，用于查看来自一个或多个时序数据源的一个或多个数据流。它以表格形式显示带有 “属性”、“最新值” 和 “单位” 的资产。支持 Amazon IoT SiteWise 警报。
- 文本-文本控件可帮助书写具有各种颜色和字体的文本。您可以通过将文本与 URL 关联来创建链接。此小组件未启用 “属性” 和 “阈值” 字段。

配置小部件

将小组件添加到仪表板后，您可以通过选择右侧面板中的配置图标来配置该小组件。

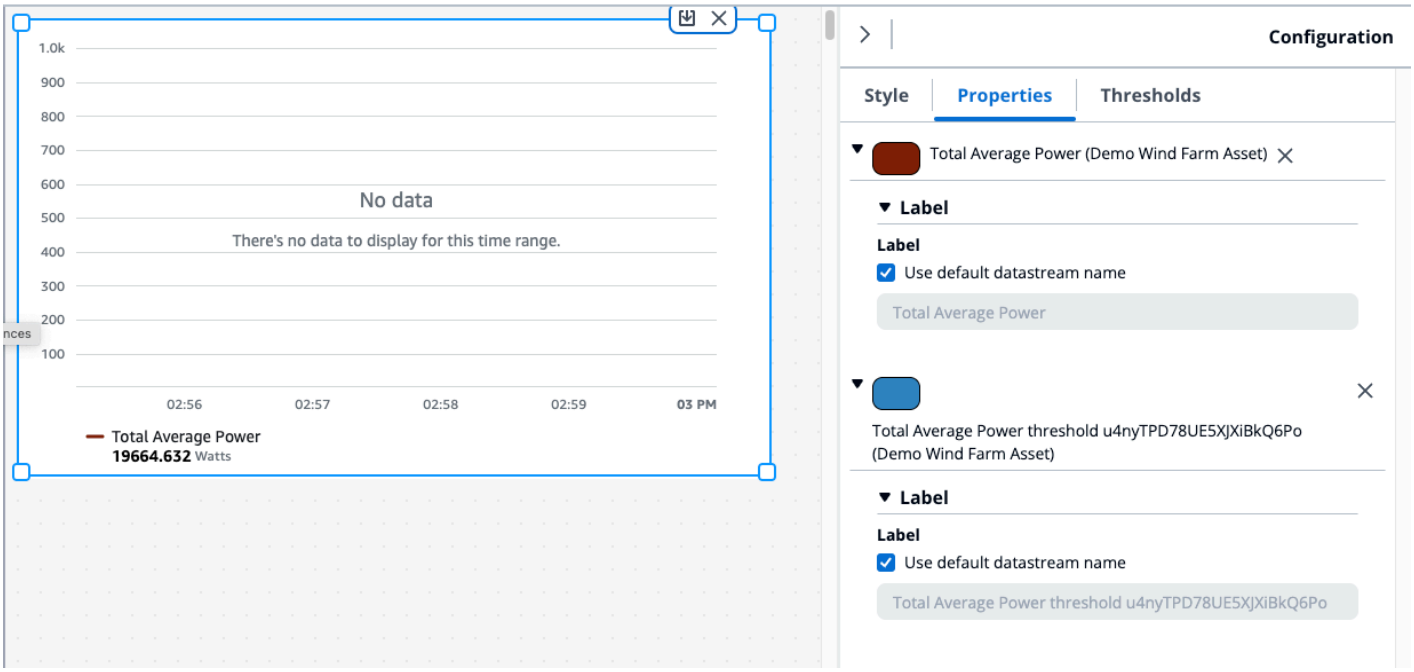
- 样式-在控件标题中添加标题。不同的控件具有不同的配置。下面列出了几个例子。
  - 栏控件：
    - 分辨率和聚合-在此处设置分辨率和聚合值。
    - 格式化数据-将小数位设置为要显示的小数位数。
    - 显示样式-选择要显示的值。
    - 轴-选择显示轴。

- 线路小部件：
  - 分辨率和聚合-在此处设置分辨率和聚合值。
  - 格式化数据-将小数位设置为要显示的小数位数。
  - Y 轴-添加标签以及最小值和最大值。
  - 控件样式-选择线条类型、线条样式、线条粗细和数据点形状值。
  - 图例-选择 “对齐”，然后选择 “显示”。
- 仪表小部件：
  - 分辨率和聚合-在此处设置分辨率和聚合值。
  - 格式化数据-将小数位设置为要显示的小数位数。
  - 显示样式-选择要显示的值。
  - Y 轴-添加标签以及最小值和最大值。
  - 字体-选择字体大小、单位字体大小和标签字体大小值。

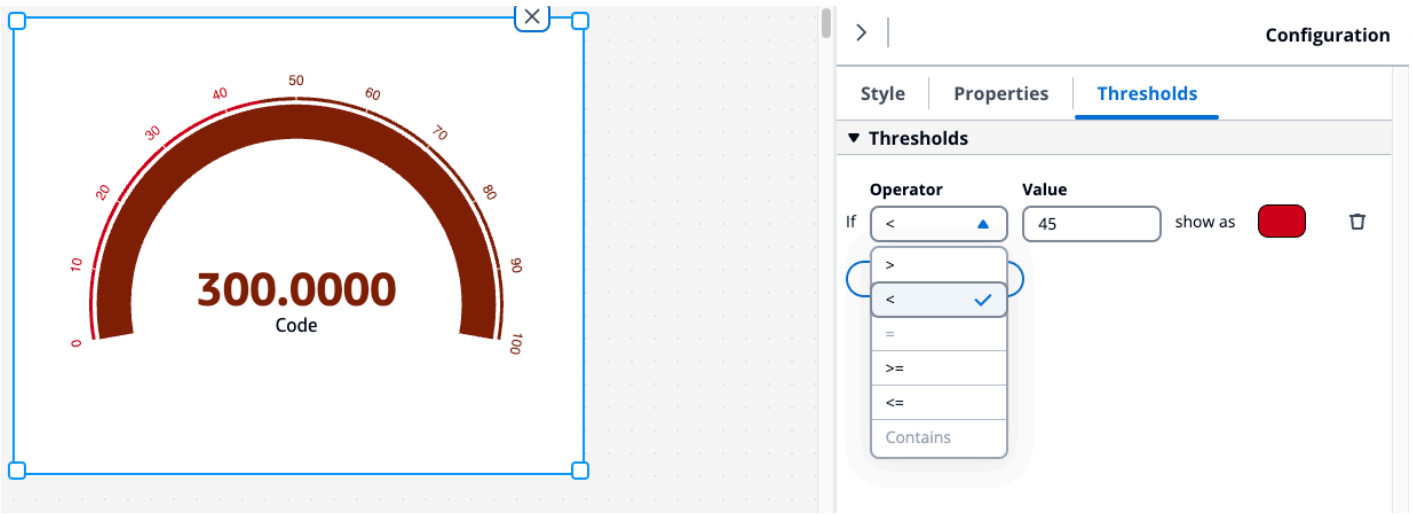


- 属性-此部分列出了小组件的所有属性。不同的控件具有不同的属性。下面列出了几个例子。
  - 线路小部件：
    - 标签-选择使用默认数据流名称或指定新名称。
    - 样式 — 将线型、线条样式设置为要显示的小数位数。
    - Y 轴 — 选择默认样式的值，显示 Y 轴控件并设置最小值和最大值。

- 表格控件：
  - 标签-选择使用默认数据流名称或指定新名称。
- 表格控件：
  - 标签-选择使用默认数据流名称或指定新名称。



- 阈值-为小组件添加阈值。不同的控件具有不同的配置。下面列出了几个例子。
  - 条形图控件：
    - 选择添加要添加到构件的阈值。
    - 选择运算符并给出阈值的值。使用调色板中的颜色自定义阈值。
    - 您可以选择对所有数据应用阈值。
  - 线路小部件：
    - 选择添加要添加到构件的阈值。
    - 选择运算符并给出阈值的值。使用调色板中的颜色自定义阈值。
    - 从下拉菜单中选择如何显示阈值。
  - 仪表小部件：
    - 选择添加要添加到构件的阈值。
    - 选择运算符并给出阈值的值。使用调色板中的颜色自定义阈值。



使用小部件

您可以单独使用仪表板中的小部件，也可以通过多选来使用它们。

在仪表板中编辑微件

选择一个控件并对其进行编辑。要编辑仪表板中的多个微件，请按住 Shift + 左键单击，然后选择仪表板中的所有微件。选择后，用户可以添加新的数据流，并在样式配置设置中修改控件标题。仪表板中所有小部件的标题都已更改。

右键单击画布，然后执行以下操作：

- 复制 — 将控件的副本添加到画布。
- 删除-删除控件。
- 置于最前面-将所选控件置于画布的正面。
- 返回底层-将所选控件发送到画布背面。

调整小部件的大小

单独调整小组件的大小，或者通过在仪表板中多选小组件来调整小组件的大小。

要更改小组件的大小，请执行以下操作：

- 要更改单个构件的大小，请选择该小组件，然后将其拖到一个角落以更改其大小。
- 要更改多个小组件的大小，请按住 Shift + 左键单击选择多个小部件，然后将其拖到一个角落以更改其大小。

删除仪表板中的小部件

单独删除小组件，或者通过在仪表板中多选小组件来删除小组件。

要删除小部件：

- 要删除单个构件，请选择该构件，然后右键单击并选择删除。您也可以选择并单击右上角的 X 来删除该构件。
- 要删除多个小组件，请按住 Shift + 左键单击选择多个小组件，然后右键单击并选择 “删除”。

小部件中的警报

当设备或流程运行不佳时，警报会向您和您的团队发出警报。计算机或进程的最佳性能意味着特定指标的值应在上限和下限的范围内。当这些指标超出其运作范围时，必须通知设备操作员以便修复问题。警报功能有助于您快速识别问题并通知操作员，以最大限度地提高设备和进程的性能。

您可以在资源浏览器的 “建模” 选项卡中找到与资产关联的警报。

- 搜索并选择资产。
- 向下滚动 “数据流” 表至 “警报数据流” 部分，然后将其展开。
- 在 “警报” 表中选择一个警报，然后选择 “添加”。

主题

- 不同小部件中的警报

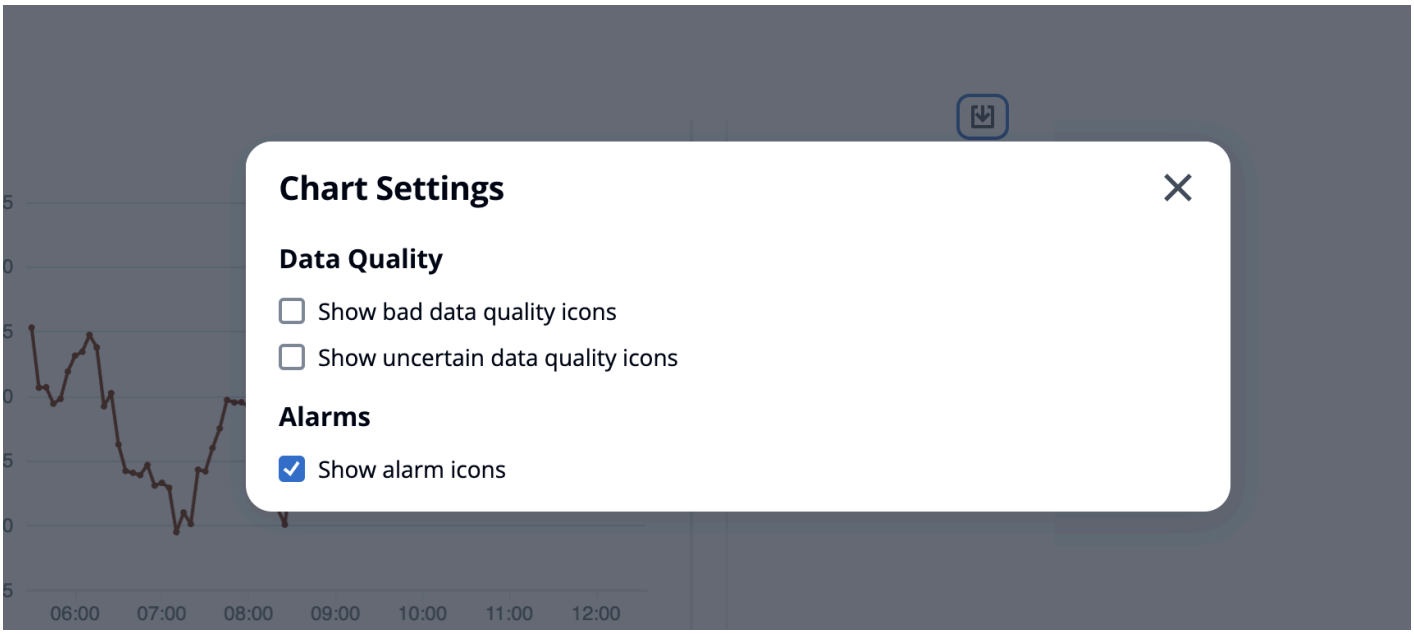
不同小部件中的警报

对于所有小部件：

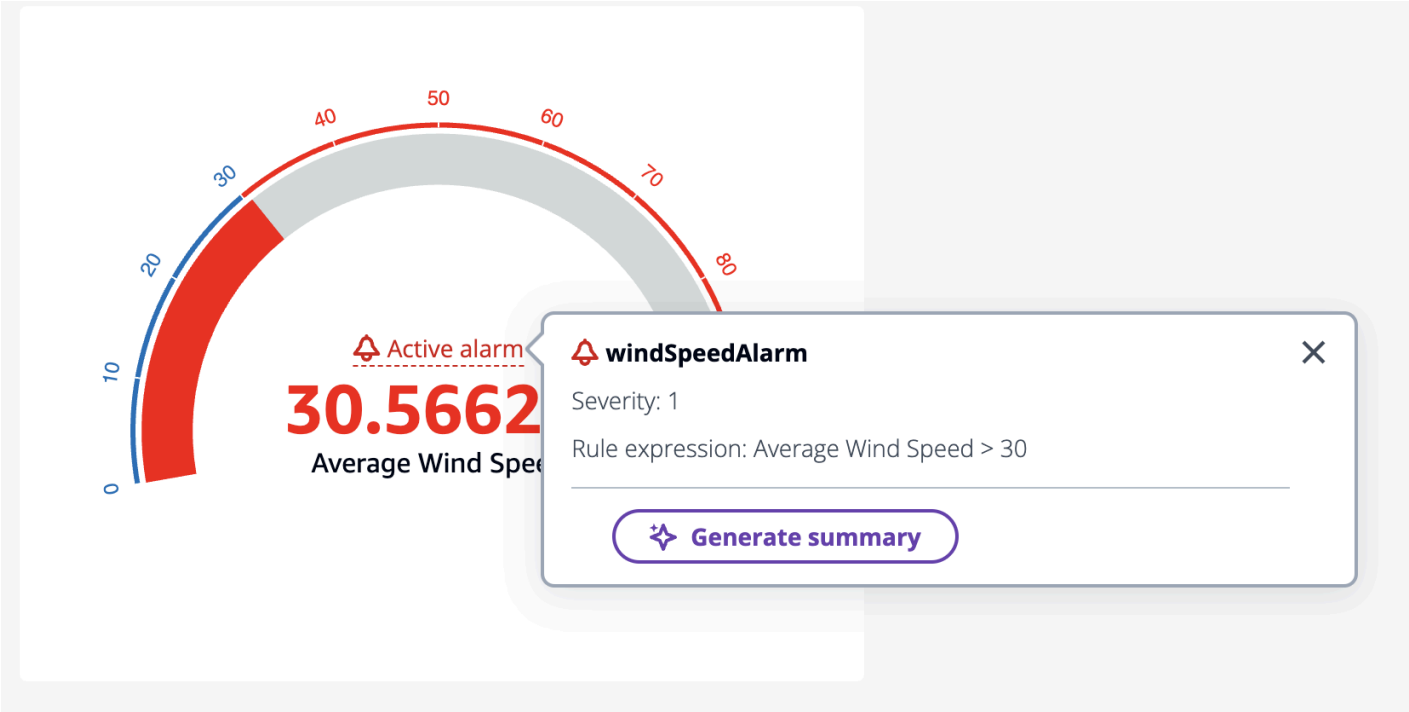
- 数据流属性设置取决于向小组件添加的属性类型。数据流属性具有完整的属性设置支持，而警报属性目前不允许配置属性设置。
- 如果您添加警报数据流，则其关联的输入属性数据流也会添加到图表中。如果您移除警报数据流，则其输入属性也会被移除。
- 要单独控制警报的输入属性数据流，必须将它们分别添加。

以下示例说明了某些小部件如何使用警报。

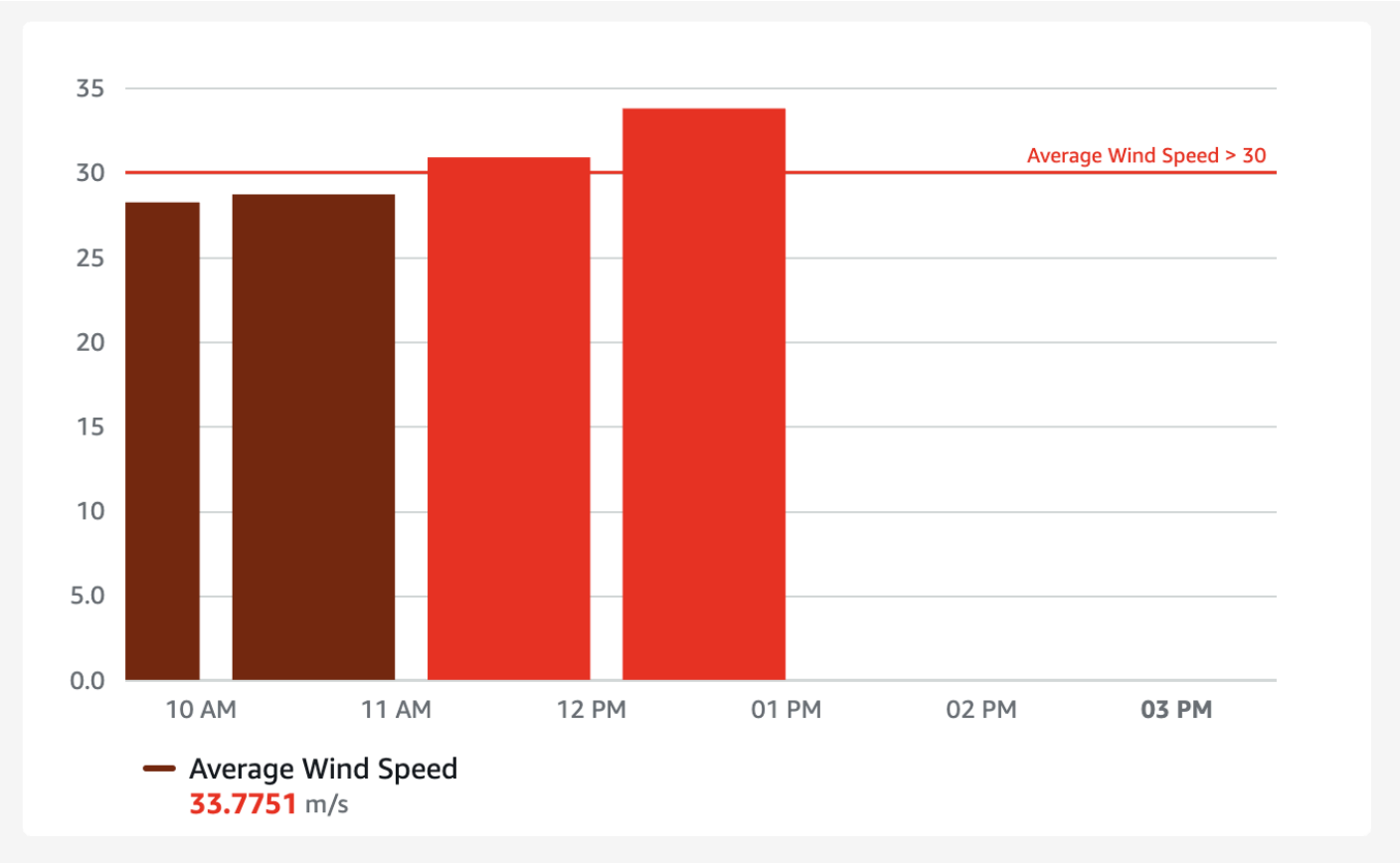
- 折线图
  - 警报及其输入属性数据流已添加到图表中。
  - 你可以在图表图例中看到警报状态，当警报状态发生变化时，你可以看到悬停在数据流上的图标。
  - 您可以从图表设置中关闭警报图标。



- 关键绩效指标和衡量标准
  - 警报及其输入属性数据流将添加到所选控件中。
  - 警报阈值已添加到控件中，该控件会根据其配置更改颜色。
  - 您可以在小组件上选择警报状态，查看警报详细信息，然后单击 “生成摘要” 以调用 Amazon IoT SiteWise 以获取警报摘要。



- 表
  - 警报及其输入属性作为一行添加到表格中。
- 条形图
  - 警报作为阈值添加到图表中，图表会更改任何突破阈值的数据流的颜色。
  - 您可以单独添加任何关联的数据流。
  - 您无法通过小组件与 Amazon IoT SiteWise 助手进行交互。

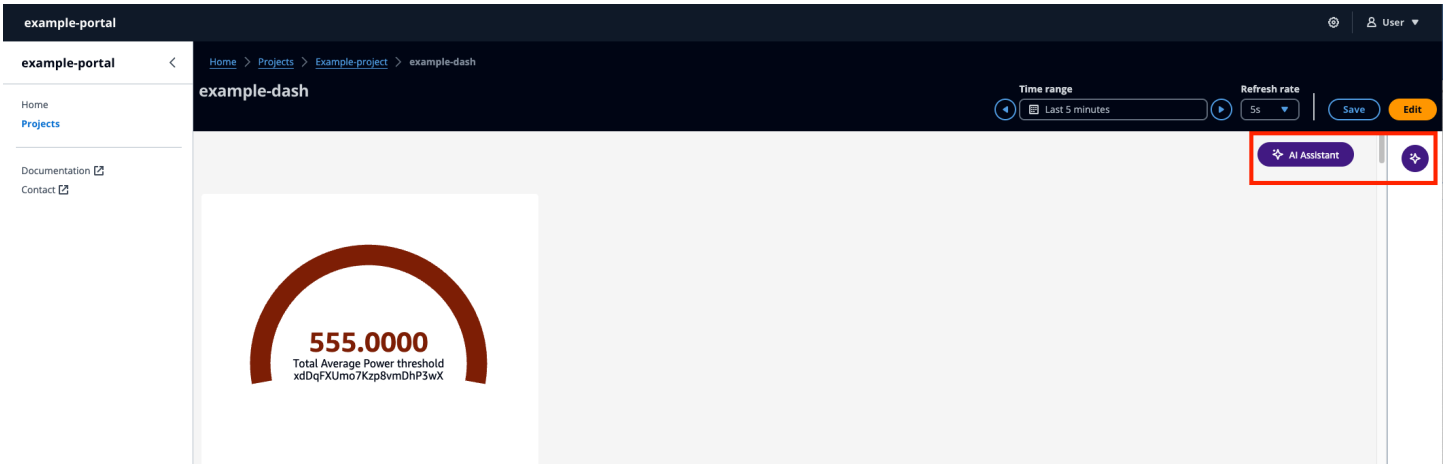


- 状态时间表
  - 警报将作为阈值添加到时间轴中。
  - 将警报状态及其输入属性数据添加到时间轴的工作正在进行中。
  - 您无法通过小组件与 Amazon IoT SiteWise 助手进行交互。

Amazon IoT SiteWise 助手在小部件中使用

As Amazon IoT SiteWise sistant 是一款由 AI 驱动的生成式助手。它允许工厂经理、质量工程师和维护技术人员等用户直接从运营和企业数据中获得见解、解决问题并采取行动。该 Amazon IoT SiteWise 助理将 Amazon IoT 数据、资产模型、手册和文档中的信息整合为可理解的关键事件摘要。它还支持交互式深入问答环节，便于诊断、根本原因探索和指导性建议。

Amazon IoT SiteWise 助手按钮位于仪表板的右上角。点击它激活助手。只能与仪表板的预览模式一起使用。



在以下情况下使用 Amazon IoT SiteWise 助手：

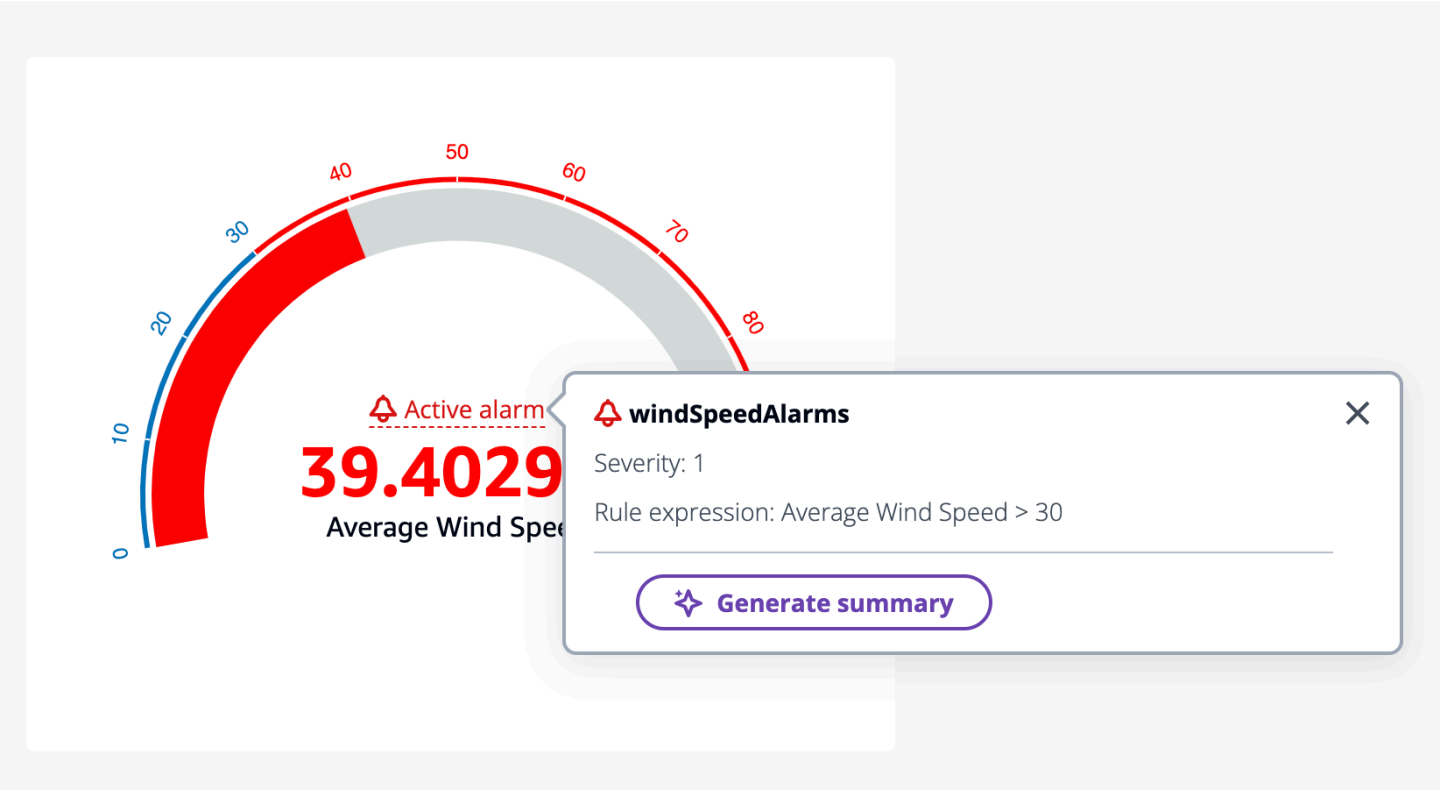
主题

- 用例-警报摘要
- 用例-情境摘要
- 用例-深入研究摘要

用例-警报摘要

汇总仪表板上选定面板的当前警报。“线路”、“KPI”、“仪表” 和 “表格” 控件支持警报。选择带有警报的小部件并对其进行总结。

- 在小部件上选择 “激活警报”。
- 将显示警报的严重性和规则表达式。
- 选择 “生成摘要” 以生成摘要。



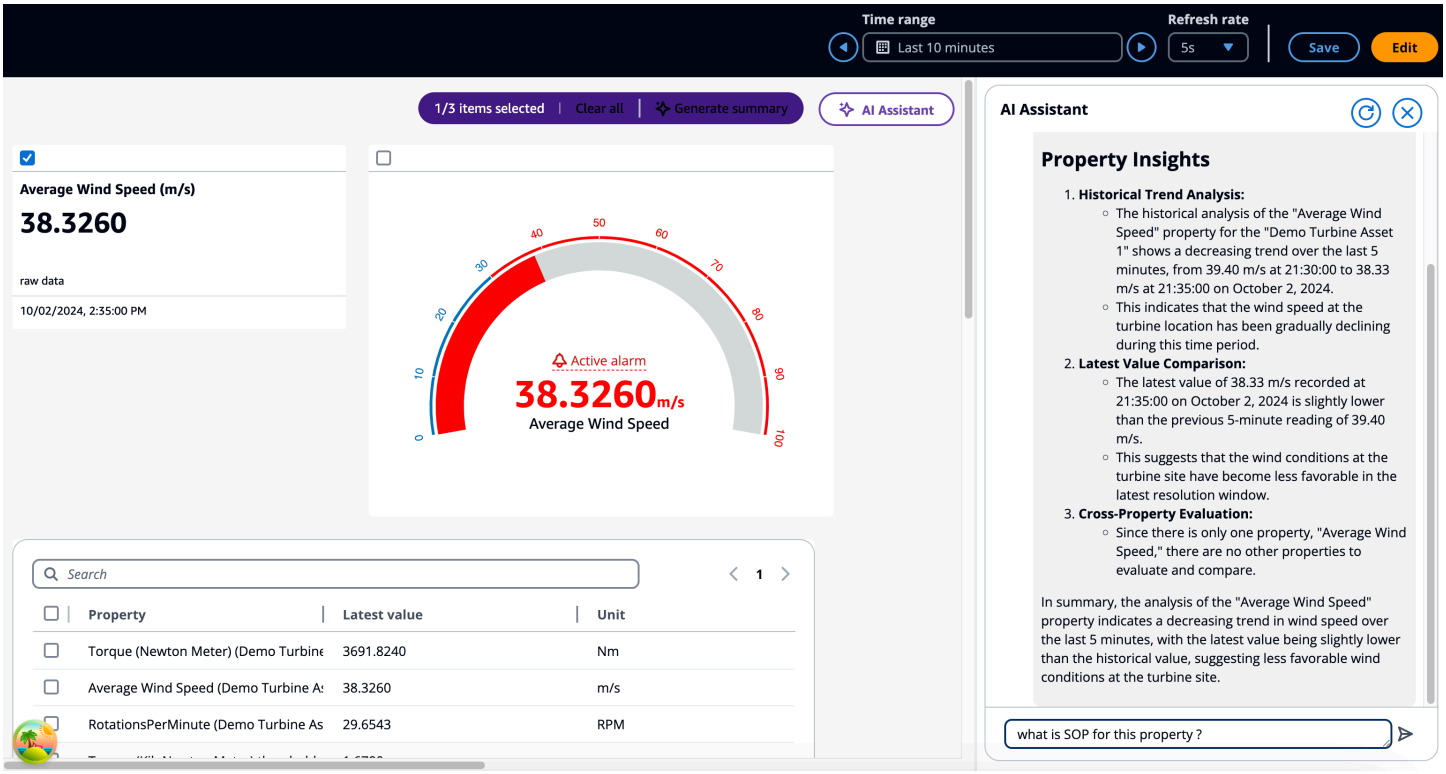
用例-情境摘要

最多选择三个小部件进行汇总。它们可以是控件和属性的组合。如果选择的选项超过三个，助手将返回错误。

使用 Amazon IoT SiteWise 助手生成情况摘要

1. 点击 AI 助手。它显示一个包含三个选项的菜单。
    a. 已选择的项目-仅选择三个。您不能选择超过三个。
    b. 全部清除-清除您的选择。
    c. 生成摘要-生成有关所选项目的摘要。
2. 选择 “生成摘要” 以生成有关所选项目的摘要。

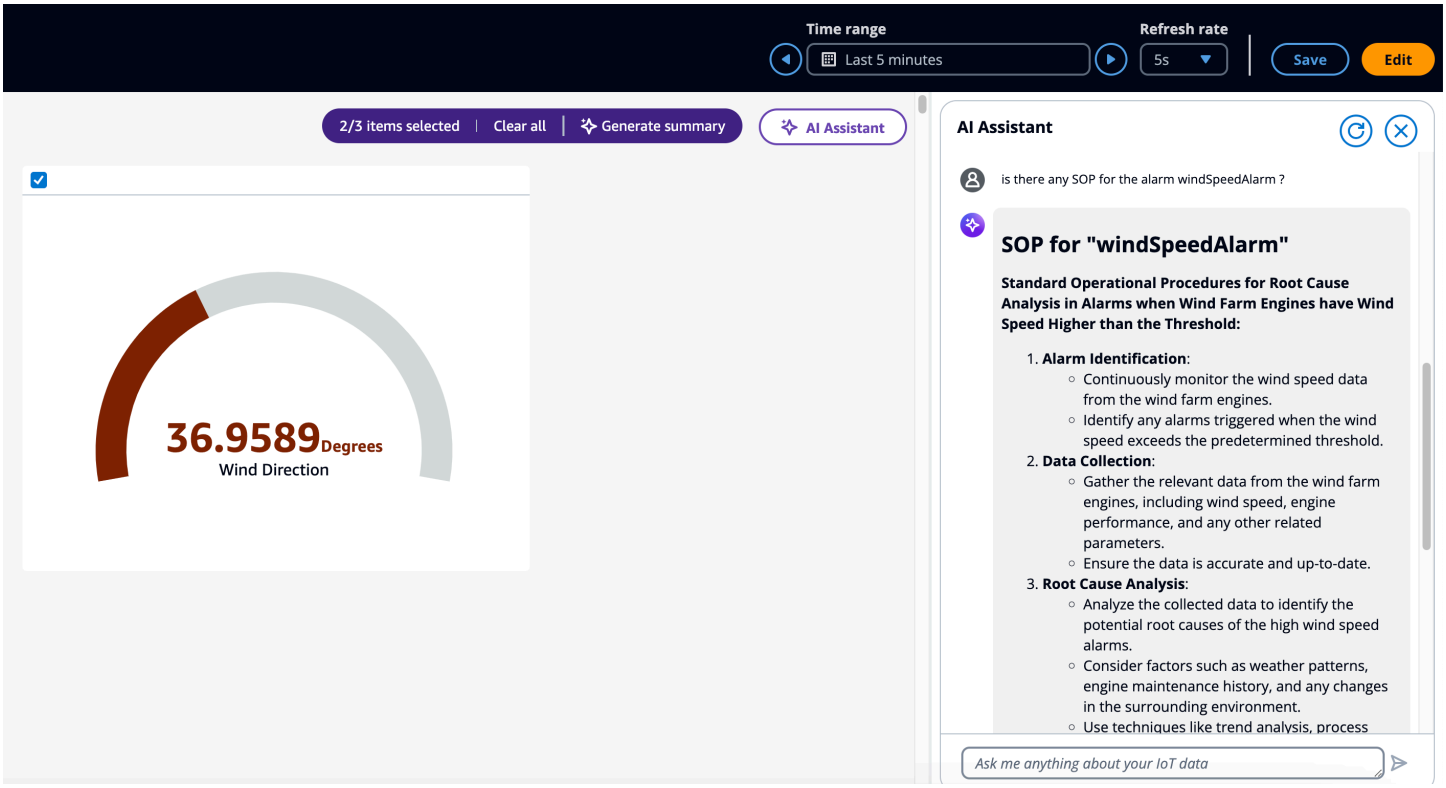
下图选择了一个小部件和来自 Amazon IoT SiteWise 助手的摘要。



用例-深入研究摘要

在这个用例中，用户可以进行深入研究，访问 SOPs（标准操作程序）、手册、文档，并考虑下一步行动。对于上一节中的示例，如果用户选择了解有关此属性的 SOP 的更多信息，请向助手询问此属性的 SOP。这会向用户显示有关 SOP 的深入信息。

以下示例显示了“警报是否有 SOP windSpeedAlarm？”的答案“



要问 Amazon IoT SiteWise 助手的问题示例

> **Note**
>
> - Amazon IoT SiteWise 助手必须使用带有 Amazon Kendra 索引的数据集来获得企业级知识和指导。如果您没有 Amazon Kendra 索引，请参阅创建索引来创建索引。添加数据集可以提高助手的响应质量。要了解更多信息，请参阅创建数据集。
> - 有些问题需要 Amazon IoT TwinMaker 集成。有关详细信息，请参阅集成 Amazon IoT TwinMaker 和 Amazon IoT SiteWise。

作为同一次对话的一部分，在仪表板中获得警报摘要后，需要向助理提问一些后续问题。

- 显示上面摘要中资产的详细信息？
- 从根到上述资产的分层路径是什么？
- 上述资产的受抚养后代资产有哪些？
- 上述资产中哪些具有主动警报功能的依赖资产？
- 查找所有具有活动警报功能的资产。

作为同一次对话的一部分，在仪表板中获得房产摘要后，需要向助理询问一些后续问题。

- 对过去 24 小时进行相同的分析。
- 查找与上述属性相关的文档。
- 提供资产 ID 1da67d28-14f8-4f71-a06a-386f0425a21d/asset 名称 Demo Turbine Asset 1 的详细信息。

从 API 调用 Amazon IoT SiteWise 助手。

- 为资产 ID windSpeedAlarm中的警报名称生成警报摘要d591e153-e5cf-4206-96bb-ce3c119d9d2d。
- 为资产 ID 中的警报名称windSpeedAlarm生成过去 12 小时/2 天/1 周的警报摘要。d591e153-e5cf-4206-96bb-ce3c119d9d2d
- 为资产 ID ab187fb7-d74b-44d9-bd9b-f2f19a9137cc 中的属性 ID 生成属性摘要 d591e153-e5cf-4206-96bb-ce3c119d9d2d
- 为资产 ID 中的房产 ID ab187fb7-d74b-44d9-bd9b-f2f19a9137cc 生成过去 12 小时/2 天/1 周的房产摘要。d591e153-e5cf-4206-96bb-ce3c119d9d2d
- 查找资产名称为 Turbine 的资产。
- 给我资产ID 5356168c-3390-456f-802c-9f6e047810d4 中房产ID的当前属性值d591e153-e5cf-4206-96bb-ce3c119d9d2d，3cbb084e-1ded-4b08-9f21-1b47b2fb86fd。
- 资产 ID 和资产 ID d591e153-e5cf-4206-96bb-ce3c119d9d2d 之间有什么关系3cbb084e-1ded-4b08-9f21-1b47b2fb86fd。
- 查找有关如何修复风力涡轮机低转速问题的文档。
- 为属性别名生成属性摘要WindSpeed。
- 根据我的知识库，术前检查有哪些？

从中查询数据 Amazon IoT SiteWise

您可以使用 Amazon IoT SiteWise API 操作来查询资产属性的当前值、历史值以及特定时间间隔内的聚合。

使用这些特征深入了解您的数据。例如，发现具有给定属性值的所有资产，或者创建数据的自定义表示形式。您还可以使用 API 操作来开发与 Amazon IoT SiteWise 资产中存储的工业数据集成的软件解决方案。您还可以在 Amazon IoT SiteWise Monitor中实时浏览您的资产数据。要了解如何配置 SiteWise 监控器，请参阅使用监控数据 Amazon IoT SiteWise Monitor。

本节中介绍的操作返回包含时间戳、质量、值（TQV）结构的属性值对象：

- `timestamp` 包含当前的 Unix 纪元时间（以秒为单位），偏移量为纳秒。
- `quality` 包含以下字符串之一，用于指示数据点的质量：
  - `GOOD` – 数据不受任何问题的影响。
  - `BAD` – 数据受传感器故障等问题的影响。
  - `UNCERTAIN` – 数据受传感器不准确等问题的影响。
- `value` 包含下列字段之一，具体取决于属性的类型：
  - `booleanValue`
  - `doubleValue`
  - `integerValue`
  - `stringValue`
  - `nullValue`

主题

- 在中查询当前资产属性值 Amazon IoT SiteWise
- 在中查询历史资产属性值 Amazon IoT SiteWise
- 在中查询资产属性聚合 Amazon IoT SiteWise
- Amazon IoT SiteWise 查询语言

在中查询当前资产属性值 Amazon IoT SiteWise

本教程展示了两种获取资产属性当前值的方法。您可以使用 Amazon IoT SiteWise 控制台或在 Amazon Command Line Interface (Amazon CLI) 中使用 API。

主题

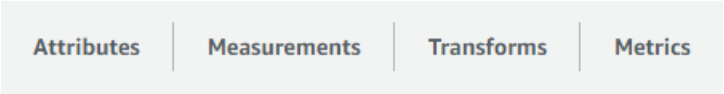
- 查询资产属性的当前值（控制台）
- 查询资产属性的当前值（Amazon CLI）

查询资产属性的当前值（控制台）

您可以使用 Amazon IoT SiteWise 控制台查看资产属性的当前值。

获取资产属性的当前值（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产。
3. 选择具有要查询的属性的资产。
4. 选择箭头图标来展开资产层次结构以查找资产。
5. 选择属性类型对应的选项卡。例如，选择 测量值 可以查看测量值属性的当前值。



6. 找到要查看的属性。当前值将显示在 最新值 列中。

查询资产属性的当前值（Amazon CLI）

您可以使用 Amazon Command Line Interface (Amazon CLI) 来查询资产属性的当前值。

使用GetAssetPropertyValue操作查询资产属性的当前值。

要标识资产属性，请指定以下项之一：

- 向其发送数据的资产属性的 `assetId` 和 `propertyId`。
- `propertyAlias`，这是一个数据流别名（例如，`/company/windfarm/3/turbine/7/temperature`）。要使用此选项，您必须首先设置资产属性的别名。要设置属性别名，请参阅管理数据流 Amazon IoT SiteWise。

要获取资产属性的当前值（Amazon CLI），请执行以下步骤：

- 运行以下命令以获取资产属性的当前值。*`asset-id`*替换为资产 ID *`property-id`* 和属性 ID。

```
aws iotsitewise get-asset-property-value \
  --asset-id asset-id \
  --property-id property-id
```

此操作将按以下格式返回包含属性当前的 TQV 的响应。

```
{
  "propertyValue": {
    "value": {
      "booleanValue": Boolean,
      "doubleValue": Number,
      "integerValue": Number,
      "stringValue": "String",
      "nullValue": {
          "valueType": "String"
      }
    },
    "timestamp": {
      "timeInSeconds": Number,
      "offsetInNanos": Number
    },
    "quality": "String"
  }
}
```

在中查询历史资产属性值 Amazon IoT SiteWise

您可以使用 Amazon IoT SiteWise API GetAssetPropertyValueHistory操作来查询资产属性的历史值。

要标识资产属性，请指定以下项之一：

- 向其发送数据的资产属性的 `assetId` 和 `propertyId`。
- `propertyAlias`，这是一个数据流别名（例如，`/company/windfarm/3/turbine/7/temperature`）。要使用此选项，您必须首先设置资产属性的别名。要设置属性别名，请参阅管理数据流 Amazon IoT SiteWise。

传递以下参数来优化结果：

- `startDate`— 查询历史数据范围的唯一起始点，以 Unix 纪元时间（以秒为单位）表示。

- endDate – 查询历史数据范围的结束端，以 Unix 纪元时间（以秒为单位）表示。
- maxResults – 一个请求中要返回的最大结果数。默认为 20 结果。
- nextToken – 从此操作的上一次调用返回的分页标记。
- timeOrdering – 应用于返回值的顺序：ASCENDING 或 DESCENDING。
- qualities – 筛选结果的质量分为：GOOD、BAD 或 UNCERTAIN。

要查询资产属性的值历史记录（Amazon CLI），请执行以下步骤：

1. 运行以下命令以获取资产属性的值历史记录。此命令将查询特定 10 分钟间隔内的属性历史记录。*asset-id*替换为资产 ID *property-id* 和属性 ID。将日期参数替换为要查询的时间间隔。

```
aws iotsitewise get-asset-property-value-history \
  --asset-id asset-id \
  --property-id property-id \
  --start-date 1575216000 \
  --end-date 1575216600
```

该操作返回的响应包含该属 TQVs 性的历史记录，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. 如果存在更多值条目，则可以将分页令牌从`nextToken`字段传递给后续的GetAssetPropertyValueHistory操作调用。

在中查询资产属性聚合 Amazon IoT SiteWise

Amazon IoT SiteWise 自动计算聚合的资产属性值，这些值是在多个时间间隔内计算出来的一组基本指标。 Amazon IoT SiteWise 每分钟、每小时和每天计算您的资产属性的以下聚合：

- 平均值 – 属性在一个时间间隔内的平均值。
- 计数 – 属性在一个时间间隔内的数据点数。
- 最大值 – 属性在一个时间间隔内的最大值。
- 最小值 – 属性在一个时间间隔内的最小值。
- 标准偏差 – 属性值在一个时间间隔内的标准偏差。
- 总和 – 属性值在一个时间间隔内的总和。

对于非数字属性，例如字符串和布尔值，仅 Amazon IoT SiteWise 计算计数聚合。

您还可以计算资产数据的自定义指标。借助指标属性，定义特定于操作的聚合。指标属性提供了额外的聚合函数和时间间隔，这些函数和时间间隔不是为 Amazon IoT SiteWise API 预先计算的。有关更多信息，请参阅 聚合来自属性和其他资产（指标）的数据。

主题

- 资产属性的聚合（API）
- 资产属性的聚合（Amazon CLI）

资产属性的聚合（API）

使用 Amazon IoT SiteWise API 获取资产属性的聚合。

使用GetAssetPropertyAggregates操作来查询资产属性的聚合。

要标识资产属性，请指定以下项之一：

- 向其发送数据的资产属性的 `assetId` 和 `propertyId`。

- `propertyAlias`，这是一个数据流别名（例如，`/company/windfarm/3/turbine/7/temperature`）。要使用此选项，您必须首先设置资产属性的别名。要设置属性别名，请参阅管理数据流 Amazon IoT SiteWise。

您必须传递以下必需参数：

- `aggregateTypes` – 要检索的聚合的列表。您可以指定 `AVERAGE`、`COUNT`、`MAXIMUM`、`MINIMUM`、`STANDARD_DEVIATION` 和 `SUM` 中的任何一个。
- `resolution`— 检索指标的时间间隔：`1m`（1 分钟）、`15m`（15 分钟）、`1h`（1 小时）或`1d`（1 天）。
- `startDate`— 查询历史数据范围的唯一起始点，以 Unix 纪元时间（以秒为单位）表示。
- `endDate` – 查询历史数据范围的结束端，以 Unix 纪元时间（以秒为单位）表示。

还可以传递以下任何参数来优化结果：

- `maxResults` – 一个请求中要返回的最大结果数。默认为 `20` 结果。
- `nextToken` – 从此操作的上一次调用返回的分页标记。
- `timeOrdering` – 应用于返回值的顺序：`ASCENDING` 或 `DESCENDING`。
- `qualities` – 筛选结果的质量分为：`GOOD`、`BAD` 或 `UNCERTAIN`。

> **Note**
>
> 该GetAssetPropertyAggregates操作返回的 TQV 格式与本节中描述的其他操作不同。`value` 结构包含请求中每个 `aggregateTypes` 的字段。`timestamp` 包含聚合发生的时间，以 Unix 纪元时间表示（以秒为单位）。

资产属性的聚合（Amazon CLI）

要查询资产属性的聚合（Amazon CLI），请执行以下步骤：

1. 运行以下命令以获取资产属性的聚合。此命令会查询特定 1 小时间隔内的平均值和总和（采用 1 小时分辨率）。*`asset-id`*替换为资产 ID *`property-id`* 和属性 ID。将参数替换为要查询的聚合和时间间隔。

```
aws iotsitewise get-asset-property-aggregates \
  --asset-id asset-id \
  --property-id property-id \
  --start-date 1575216000 \
  --end-date 1575219600 \
  --aggregate-types AVERAGE SUM \
  --resolution 1h
```

该操作返回的响应包含以下格式 TQVs 的属性的历史记录。响应中仅包括请求的聚合。

```
{
  "aggregatedValues": [
    {
      "timestamp": Number,
      "quality": "String",
      "value": {
        "average": Number,
        "count": Number,
        "maximum": Number,
        "minimum": Number,
        "standardDeviation": Number,
        "sum": Number
      }
    }
  ],
  "nextToken": "String"
}
```

2. 如果存在更多值条目，则可以将分页令牌从`nextToken`字段传递给后续的GetAssetPropertyAggregates操作调用。

> **Note**
>
> 如果您的查询范围包含`null`值 TQVs，请参阅 AssetPropertyValueAPI。除计数之外的所有统计信息都会产生`null`响应，类似于 String 的统计信息 TQVs。如果您的查询范围包含双精`Double.NaN`度类型 TQVs，则除 count 之外的所有计算都将得出`Double.NaN`。

Amazon IoT SiteWise 查询语言

通过 Amazon IoT SiteWise 数据检索 ExecuteQueryAPI 操作，您可以从以下位置检索有关声明式结构定义以及与之关联的时间序列数据的信息：

- 模型
- 资产
- 测量值
- 指标
- 转换
- 聚合

这可以在单个 API 请求中使用类似查询语句的 SQL 来完成。

> **Note**
>
> 除了 Amazon GovCloud （美国西部）之外，所有同时提供 Amazon IoT SiteWise 和的地区 Amazon IoT TwinMaker 均提供此功能。

主题

- 先决条件
- 查询语言参考 Amazon IoT SiteWise

先决条件

Amazon IoT SiteWise 需要权限才能与之集成， Amazon IoT TwinMaker 这样它才能组织和建模工业数据。

在检索有关模型、资产、测量值、指标、转换和聚合的信息之前，请确保满足以下先决条件：

- 两者都有服务相关角色 Amazon IoT SiteWise ，并在您的 Amazon 账户中 Amazon IoT TwinMaker 设置。有关服务相关角色的更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。
- 为您的 IAM 角色启用的 Amazon IoT SiteWise 集成。有关更多信息，请参阅 整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker。

- `IoTSiteWiseDefaultWorkspace`在您所在地区的账户中带有 ID Amazon IoT TwinMaker 的工作空间。有关更多信息，请参阅 IoTSiteWiseDefaultWorkspace 用户指南中的使用Amazon IoT TwinMaker 。
- Amazon IoT TwinMaker 启用的标准或分层捆绑定价模式。有关更多信息，请参阅《Amazon IoT TwinMaker 用户指南》中的切换 Amazon IoT TwinMaker 定价模式。

查询语言参考 Amazon IoT SiteWise

Amazon IoT SiteWise 支持丰富的查询语言来处理您的数据。以下主题介绍了可用的数据类型、运算符、函数和构造。

示例查询要使用查询语言编写 Amazon IoT SiteWise 查询，请参阅。

主题

- 了解查询参考视图
- 支持的数据类型
- 使用 SELECT 语句检索数据
- 逻辑运算符
- 比较运算符
- 示例查询

了解查询参考视图

本节提供的信息可帮助您理解中的视图 Amazon IoT SiteWise，例如流程元数据和遥测数据。

下表提供了视图的名称和说明。

数据模型

视图名称	视图描述
asset	包含有关资产和模型派生的信息。
asset_property	包含有关资产属性结构的信息。
raw_time_series	包含时间序列的历史数据。

视图名称	视图描述
latest_value_time_series	包含时间序列的最新值。
precomputed_aggregates	包含自动计算的聚合资产属性值。它们是一组按多个时间间隔计算的基本指标。

以下视图列出了查询的列名称及示例数据。

视图：asset

asset_id	asset_name	asset_description	asset_model_id
88898498-0b8b-42b5-bf57-16180bc3d3a0	WindTurbine A	WindTurbine 资产 A	17847250-5bf0-4f74-b775-cc03f05e7cb8
17847250-5bf0-4f74-b775-cc03f05e7cb8	风力涡轮机资产模型	代表风电场中的涡轮机。	

视图：asset_property

property_id	asset_id	property_name	property_alias	asset_composite_model_id	
b29be434-b000-4d74-b809-75287d83bcd6	88898498-0b8b-42b5-bf57-16180bc3d3a0	电机温度	`Rochester 2/44///Line-5/Bus-2/Machine-5/Temperature`		
3b458f00-24e7-458a-b4e8-c6026eff654a	88898498-0b8b-42b5-bf57-16180bc3d3a0	风向	`/company/windfarm/3/turbine/7/winddirection`	2f458n00-56e7-458h-b4e8-c6026eff985g	

视图：raw_time_series

asset_id	property_id	property_alias	event_timestamp	quality	boolean_value	int_value	double_value	string_value
88898498-0b8b-42b5-bf57-16180bc3d3a	b29be434-b000-4d7-b809-75287d83bcd	Rochest 2/44/// Line-5/Bus-2/Machine-5/Temperature	157521960	好			115.0	
88898498-0b8b-42b5-bf57-16180bc3d3a	3b458f00-24e7-458-b4e8-c6026eff654a	/company/windfarm3/turbine/7/winddirection	157521937	好			348.75	

> **Note**
>
> 您必须在 `event_timestamp` 列中包含筛选条件子句才能查询 `raw_time_series` 视图。这是必需的筛选条件，如果没有它，查询将会失败。
>
> Example 查询
>
> ```
> SELECT event_timestamp, double_value FROM raw_time_series WHERE event_timestamp
> > 1234567890
> ```

视图：latest_value_time_series

asset_id	property_id	property_alias	event_timestamp	quality	boolean_value	int_value	double_value	string_value
88898498-0b8b-42b5-bf57-16180bc3d3a	3b458f00-24e7-458a-b4e8-c6026eff654a	/company/windfarm/3/turbine/7/winddirection	157521960	好			355.39	

视图：precomputed_aggregates

asset_i	propert id	propert alias	event_t estamp	quality	resoluti n	sum_va	count_v ue	averag alue	maximu alue	minimu alue	stdev_val ue
888984 0b8b-4 - bf57-16 80bc3d	b29be4 b000-4 - b809-7 87d83b	Roche 2/44/ Line-5/ Bus-2/ Machi -5/ Tempe ature	157521 0	好	15m	1105.4	15	73.4	80.6	68	3.64

支持的数据类型

Amazon IoT SiteWise 查询语言支持以下数据类型。

标量值

数据类型	描述
STRING	最大长度为 1024 个字节的字符串。
INTEGER	有符号的 32 位整数，其范围为 -2,147,483,648 to 2,147,483,647 。
DOUBLE	浮点数-10^100 to 10^100，范围为或精Nan度为IEEE 754双精度。
BOOLEAN	true 或 false。

Nullvalue：true表示缺少定义数据的布尔值。

> **Note**
>
> 双精度数据不准确。由于精度有限，有些值无法精确转换，也无法代表所有实数。查询中的浮点数据可能与内部表示的值不同。如果输入数字的精度过高，则该值将被四舍五入。

使用 SELECT 语句检索数据

SELECT 语句用于检索一个或多个视图中的数据。 Amazon IoT SiteWise 支持视图的隐式 JOIN。您可以列出要加入的视图（在 SELECT 语句的 FROM 子句中），以逗号进行分隔。

Example

使用以下 SELECT 语句：

```
SELECT select_expr [, ...]
[ FROM from_item [, ...] ]
[ WHERE [LIKE condition ESCAPE condition] ]
```

在前面的示例中，LIKE子句使用通配符指定搜索和筛选条件。 Amazon IoT SiteWise 支持percentage (%)作为通配符。

Example 在某种条件下使用 `%`：

```
Prefix search: String%
Infix search: %String%
Suffix search: %String
```

Example 搜索资产：

```
SELECT asset_name, asset_description FROM asset WHERE asset_name LIKE 'Wind%'
```

Example 使用 ESCAPE 条件搜索资产：

```
SELECT asset_name, asset_description FROM asset WHERE asset_name LIKE 'room\%' ESCAPE
 '\'
```

逻辑运算符

Amazon IoT SiteWise 支持以下逻辑运算符。

逻辑运算符

运算符	描述	示例
AND	如果两个值都为真，则为 TRUE	a AND b

如果 a 或 b 为 FALSE，则前一个表达式的计算结果为假。要使 AND 运算符的计算结果为真，a 和 b 都必须为真。

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
```

比较运算符

Amazon IoT SiteWise 支持以下比较运算符。所有比较操作都适用于内置数据类型，其计算结果为布尔值。

逻辑运算符

运算符	描述
<	Less than
>	Greater than
<=	小于或等于
>=	大于或等于
=	Equals
! =	Not equal

非数值的比较运算真值表

类型	键入 >= x	键入 <= x	键入 > x	键入 < x	类型 = x	键入 ！ = x
NaN	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

有些谓词的行为类似于运算符，但语法特殊。有关示例，请参见下文。

比较谓词

运算符	描述
IS NULL	测试一个值是否为NULL。
IS NOT NULL	测试一个值是否不是NULL。
IS NaN	测试一个值是否为NaN。
IS NOT NaN	测试一个值是否不是NaN。

示例查询

元数据筛选

以下示例是使用带有 Amazon IoT SiteWise 查询语言的语SELECT句进行元数据筛选：

```
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%'
```

值筛选

以下是使用带有 Amazon IoT SiteWise 查询语言的语SELECT句进行值筛选的示例：

```
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
```

与其他 Amazon 服务互动

Amazon IoT SiteWise 可以将资产数据发布到 Amazon IoT MQTT 发布-订阅消息代理，以便您可以与其他服务中的资产数据进行交互。 Amazon Amazon IoT SiteWise 为每个资产属性分配一个唯一的 MQTT 主题，您可以使用该主题使用 Amazon IoT 核心规则将资产数据路由到其他 Amazon 服务。例如，您可以将 Amazon IoT Core 规则配置为执行以下任务：

- 识别设备故障，并通过向 Amazon IoT Events 发送数据通知适当的人员。
- 在历史记录中选择要在外部软件解决方案中使用的资产数据，然后将其发送到 Amazon DynamoDB。
- 通过触发 Amazon Lambda 函数生成每周报告。

您可以按照教程进行操作，该教程逐步执行设置在 DynamoDB 中存储属性值的规则所需的步骤。有关更多信息，请参阅 将属性值更新发布到 Amazon DynamoDB。

有关如何配置规则的更多信息，请参阅 Amazon IoT 开发人员指南中的规则。

您也可以将来自其他 Amazon 服务的数据重新使用到 Amazon IoT SiteWise。要通过 Amazon IoT SiteWise 规则操作摄取数据，请参阅Amazon IoT SiteWise 使用 Amazon IoT Core 规则提取数据。

主题

- 了解 MQTT 主题中的资产属性
- 在中打开资产属性通知 Amazon IoT SiteWise
- 在中查询资产属性通知 Amazon IoT SiteWise
- 使用资产属性通知将数据导出到 Amazon S3
- Amazon IoT SiteWise 与 Grafana 集成
- 整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker
- 使用 Lookout for Equipment 检测异常

了解 MQTT 主题中的资产属性

每个资产属性都具有以下格式的唯一 MQTT 主题路径。

```
$aws/sitewise/asset-models/assetModelId/assets/assetId/properties/propertyId
```

> **Note**
>
> Amazon IoT SiteWise 不支持 C Amazon IoT ore 规则引擎中的#（多级）主题筛选通配符。您可以使用 +（单级）通配符。例如，您可以使用以下主题筛选器来匹配特定资产模型的所有更新。
>
> ```
> $aws/sitewise/asset-models/assetModelId/assets/+/properties/+
> ```
>
> 要了解有关主题筛选通配符的更多信息，请参阅 Amazon IoT Core 开发人员指南中的主题。

在中打开资产属性通知 Amazon IoT SiteWise

您可以启用属性通知以向其发布资产数据更新 Amazon IoT Core，然后对您的数据进行查询。通过资产属性通知， Amazon IoT SiteWise 提供了一个可用于将 Amazon IoT SiteWise 数据导出到 Amazon S3 的 Amazon CloudFormation 模板。

> **Note**
>
> 资产数据在 Amazon IoT Core 每次收到时都会发送到 Amazon IoT SiteWise，无论其值是否发生变化。

主题

- 启用资产属性通知（控制台）
- 启用资产属性通知（Amazon CLI）

启用资产属性通知（控制台）

默认情况下， Amazon IoT SiteWise 不发布属性值更新。您可以使用 Amazon IoT SiteWise 控制台为资产属性启用通知。

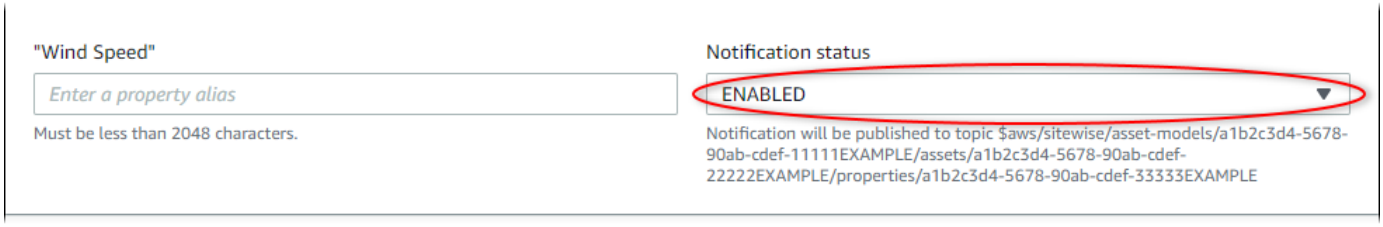
启用或禁用资产属性的通知（控制台）

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 资产。

3. 选择资产以启用属性的通知。

> **Tip**
> 您可以选择箭头图标来展开资产层次结构以查找资产。

4. 选择编辑。
5. 对于资产属性的 通知状态，选择 已启用。



   您还可以选择 已禁用 以禁用资产属性的通知。
6. 选择保存。

启用资产属性通知（Amazon CLI）

默认情况下， Amazon IoT SiteWise 不发布属性值更新。您可以使用 Amazon Command Line Interface (Amazon CLI) 来启用或禁用资产属性的通知。

您必须知道资产的 `assetId` 和属性的 `propertyId` 才能完成此过程。您还可以使用外部 ID。如果您创建了资产但不知道该资产`assetId`，请使用 ListAssetsAPI 列出特定模型的所有资产。使用该DescribeAsset操作查看资产的属性，包括财产 IDs。

使用UpdateAssetProperty操作来启用或禁用资产属性的通知。指定以下参数：

- `assetId` – 资产的 ID。
- `propertyId` – 资产属性的 ID。
- `propertyNotificationState` – 属性值通知状态：`ENABLED` 或 `DISABLED`。
- `propertyAlias` – 属性的别名。在更新通知状态时指定属性的现有别名。如果省略此参数，则将删除属性的现有别名。

启用或禁用资产属性的通知 (CLI)

1. 运行以下命令以检索资产属性的别名。*`asset-id`*替换为资产 ID *`property-id`* 和属性 ID。

```
aws iotsitewise describe-asset-property \
  --asset-id asset-id \
  --property-id property-id
```

此操作将按以下格式返回包含资产属性的详细信息的响应。属性别名位于 JSON 对象中的 `assetProperty.alias` 中。

```
{
  "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetName": "Wind Turbine 7",
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetProperty": {
    "id": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "name": "Wind Speed",
    "alias": "/company/windfarm/3/turbine/7/windspeed",
    "notification": {
      "topic": "$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE",
      "state": "DISABLED"
    },
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
      "measurement": {}
    }
  }
}
```

2. 运行以下命令以启用资产属性的通知。*property-alias*替换为上一个命令响应中的属性别名，或者省略`--property-alias`以更新不带别名的属性。

```
aws iotsitewise update-asset-property \
  --asset-id asset-id \
  --property-id property-id \
  --property-notification-state ENABLED \
  --property-alias property-alias
```

您还可以传递 `--property-notification-state DISABLED` 以禁用资产属性的通知。

在中查询资产属性通知 Amazon IoT SiteWise

要查询资产属性通知，请创建由 SQL 语句组成的 Amazon IoT Core 规则。

Amazon IoT SiteWise 以下列格式将资产属性数据更新发布到 Amazon 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`列表中的每个结构都是一个 timestamp-quality-value (TQV) 结构。

- `timestamp` 包含当前的 Unix 纪元时间（以秒为单位），偏移量为纳秒。
- `quality` 包含以下字符串之一，用于指示数据点的质量：
  - `GOOD` – 数据不受任何问题的影响。
  - `BAD` – 数据受传感器故障等问题的影响。
  - `UNCERTAIN` – 数据受传感器不准确等问题的影响。
- `value` 包含下列字段之一，具体取决于属性的类型：

- `booleanValue`
- `doubleValue`
- `integerValue`
- `stringValue`
- `nullValue`

`nullValue`— 一种结构，其中包含以下字段，表示属性值的类型，值为 Null，质量为`BAD`或`UNCERTAIN`。

- `valueType`— {“B”、“D”、“S”、“I”} 的枚举

要解析 `values` 数组中的值，您需要在规则的 SQL 语句中使用复杂的嵌套对象查询。有关更多信息，请参阅 Amazon IoT 开发人员指南中的嵌套对象查询，或参阅 将属性值更新发布到 Amazon DynamoDB 教程，了解解析资产属性通知消息的具体示例。

Example 提取值数组的示例查询

以下语句演示如何针对具有特定 double 类型属性的所有资产查询该属性的已更新属性值的数组。

```
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'
```

上一个规则查询语句采用以下格式输出数据。

```
{
  "windspeed": [
    26.32020195042838,
    26.282584572975477,
    26.352566977372508,
    26.283084346171442,
    26.571883739599322,
    26.60684140743005,
    26.628738636715045,
    26.273486932802125,
```

```
    26.436379105473964,
    26.600590095377303
  ]
}
```

Example 提取单个值的示例查询

以下语句演示如何针对具有特定 double 类型属性的所有资产查询该属性的属性值数组中的第一个值。

```
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'
```

上一个规则查询语句采用以下格式输出数据。

```
{
  "windspeed": 26.32020195042838
}
```

> **Important**
>
> 该规则查询语句忽略每个批次中的第一个值更新以外的更新。每个批次最多可以包含 10 个值。如果需要包含其余值，您必须设置更复杂的解决方案，以将资产属性值输出到其他服务。例如，您可以设置一条规则，其 Amazon Lambda 操作是将数组中的每个值重新发布到另一个主题，然后设置另一个规则来查询该主题并将每个值发布到所需的规则操作。

使用资产属性通知将数据导出到 Amazon S3

您可以将传入的数据导出 Amazon IoT SiteWise 到您账户中的 Amazon S3 存储桶。您可以按照一种格式备份数据，该格式可用于创建历史报告或使用复杂方法分析数据。

要从中导出时间序列数据 Amazon IoT SiteWise，请启用冷层功能，将数据存储在 Amazon S3 存储桶中。有关详细信息，请参阅中的 Amazon IoT SiteWise管理数据存储。

要从中导出资产模型和资产元数据 Amazon IoT SiteWise，请使用批量操作功能将元数据导出到 Amazon S3 存储桶。有关详细信息，请参阅使用资产和模型进行批量操作。

Amazon IoT SiteWise 与 Grafana 集成

Grafana 是一个数据可视化平台，用于可视化和监控仪表板中的数据。在 Grafana 版本 10.4.0 及更高版本中，使用 Amazon IoT SiteWise 该插件在 Grafana 仪表板中可视化您的 Amazon IoT SiteWise 资产数据。用户可以使用单个 Grafana 控制面板可视化来自多个 Amazon 来源（例如 Amazon IoT SiteWise Amazon Timestream 和亚马逊 CloudWatch）和其他数据源的数据。

你可以选择两个选项来使用该 Amazon IoT SiteWise 插件：

- 本地 Grafana 服务器

  你可以在自己管理的 Grafana 服务器上设置 Amazon IoT SiteWise 插件。有关如何添加和使用插件的更多信息，请参阅网站上的 Amazon IoT SiteWise Datasource 自述文件。 GitHub

- Amazon 适用于 Grafana 的托管服务

  你可以在 Grafana Amazon 托管服务 (AMG) 中使用该 Amazon IoT SiteWise 插件。AMG 为您管理 Grafana 服务器，以便您可以可视化您的数据，而无需构建、打包或部署任何硬件或任何其他 Grafana 基础设施。有关更多信息，请参阅 Amazon Managed Service for Grafana 用户指南中的以下主题：

  - 什么是 Amazon Managed Service for Grafana (AMG)？
  - 使用 Amazon IoT SiteWise 数据源

Example Grafana 控制面板示例

以下 Grafana 控制面板可视化演示风电场。您可以在 Grafana Play 网站上访问此演示控制面板。

整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker

与集成后 Amazon IoT TwinMaker ，用户可以访问中的强大功能 Amazon IoT SiteWise，例如 Amazon IoT SiteWise 控制台中的 Amazon IoT SiteWise 数据检索 `ExecuteQuery` API 和高级资产搜索。要集成服务并使用这些特征，您必须先启用集成。

主题

- 启用集成
- 整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker

启用集成

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。有关 Amazon IoT SiteWise 支持的操作的更多信息，请参阅《服务授权参考》 Amazon IoT SiteWise中定义的操作。

有关 Amazon IoT TwinMaker 服务关联角色的更多信息，请参阅《Amazon IoT TwinMaker 用户指南》 Amazon IoT TwinMaker中的服务关联角色。

在集成 Amazon IoT SiteWise 和之前 Amazon IoT TwinMaker，必须授予 Amazon IoT SiteWise 允许与 Amazon IoT TwinMaker 链接工作区集成的以下权限：

- `iotsitewise:EnableSiteWiseIntegration`— Amazon IoT SiteWise 允许与链接 Amazon IoT TwinMaker 的工作区集成。这种集成 Amazon IoT TwinMaker 允许 Amazon IoT SiteWise 通过 Amazon IoT TwinMaker 与服务相关的角色读取您的所有建模信息。要实现此权限，请为您的 IAM 角色添加以下策略：

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

整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker

要整合 Amazon IoT SiteWise 和 Amazon IoT TwinMaker，您必须具备以下条件：

- Amazon IoT SiteWise 在您的账户中设置服务相关角色
- Amazon IoT TwinMaker 在您的账户中设置服务相关角色
- Amazon IoT TwinMaker `IoTSiteWiseDefaultWorkspace`在您所在区域的账户中使用 ID 的工作空间。

使用 Amazon IoT SiteWise 控制台进行集成

当您在控制台中看到与 Amazon IoT TwinMaker集成横幅时，选择授予权限。系统会在您的账户中创建先决条件。

要通过使用进行集成 Amazon CLI

要 Amazon IoT TwinMaker 进行集成 Amazon IoT SiteWise 并使用 Amazon CLI，请输入以下命令：

1. 通过 `iotsitewise.amazonaws.com` 的 `AWSServiceName` 调用 `CreateServiceLinkedRole`。

```
aws iam create-service-linked-role --aws-service-name iotsitewise.amazonaws.com
```

2. 通过 `iottwinmaker.amazonaws.com` 的 `AWSServiceName` 调用 `CreateServiceLinkedRole`。

```
aws iam create-service-linked-role --aws-service-name iottwinmaker.amazonaws.com
```

3. 通过 `IoTSiteWiseDefaultWorkspace` 的 `ID` 调用 `CreateWorkspace`。

```
aws iottwinmaker create-workspace --workspace-id IoTSiteWiseDefaultWorkspace
```

使用 Lookout for Equipment 检测异常

> **Note**
> 异常检测仅在提供 Amazon Lookout for Equipment 的地区可用。

您可以 Amazon IoT SiteWise 与 Amazon Lookout for Equipment 集成，通过工业设备的异常检测和预测性维护，深入了解您的工业设备。Lookout for Equipment 是一项机器学习（ML）服务，用于监控工业设备，检测设备的异常行为并识别潜在故障。借助 Lookout for Equipment，您可以实施预测性维护计划并识别不太理想的设备流程。有关 Lookout for Equipment 的更多信息，请参阅《Amazon Lookout for Equipment 用户指南》中的 What is Amazon Lookout for Equipment?。

当您创建预测以训练机器学习模型以检测异常设备行为时， Amazon IoT SiteWise 会将资产属性值发送到 Lookout for Equipment，以训练机器学习模型来检测异常设备行为。要定义资产模型的预测定义，您需要指定 Lookout for Equipment 访问您的数据所需的 IAM 角色，以及要发送到 Lookout for

Equipment 并将处理过的数据发送到 Amazon S3 的属性。有关更多信息，请参阅 在中创建资产模型 Amazon IoT SiteWise。

要集成 Amazon IoT SiteWise 和 Lookout for Equipment，您需要执行以下高级步骤：

- 在资产模型上添加预测定义，概述您要跟踪的属性。预测定义是可重复使用的测量值、转换和指标集合，用于根据该资产模型对资产创建预测。
- 根据您提供的历史数据训练预测。
- 计划推断，它 Amazon IoT SiteWise 告诉运行特定预测的频率。

规划推理后，Lookout for Equipment 模型会监控它从您的设备接收的数据，并查找设备行为的异常情况。你可以使用 GET API 操作或 Lookout fo SiteWise r Equip Amazon IoT SiteWise ment 控制台在 Monitor 中查看和分析结果。您还可以使用资产模型中的警报探测器创建警报，用于提醒您注意设备的异常行为。

主题

- 添加预测定义（控制台）
- 训练预测（控制台）
- 开始或停止对预测的推理（控制台）
- 添加预测定义（CLI）
- 训练预测并开始推理（CLI）
- 训练预测（CLI）
- 开始或停止对预测的推理（CLI）

添加预测定义（控制台）

要开始将收集的数据发送 Amazon IoT SiteWise 到 Lookout for Equipment，您必须向资产模型添加 Amazon IoT SiteWise 预测定义。

向 Amazon IoT SiteWise 资产模型添加预测定义

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择模型并选择要向其添加预测定义的资产模型。
3. 选择预测。
4. 选择添加预测定义。

5. 定义有关预测定义的详细信息。

   a. 为您的预测定义输入唯一的名称和描述。请谨慎选择名称，因为创建预测定义后，您便无法再更改其名称。

   b. 创建或选择支持 Amazon IoT SiteWise 与 Amazon Lookout for Equipment 共享您的资产数据的 IAM 权限角色。该角色还应该具有以下 IAM 和信任策略。如需获取创建角色方面的帮助，请参阅使用自定义信任策略创建角色（控制台）。

      IAM 策略

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

信任策略

```
{
    "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. 选择下一步。

6. 选择要发送到 Lookout for Equipment 的数据属性（测量值、转换和指标）。

   a. （可选）选择测量值。

   b. （可选）选择转换。

   c. （可选）选择指标。

   d. 选择下一步。

7. 检视您的选择。要将预测定义添加到资产模型中，请在摘要页面上选择添加预测定义。

您也可以编辑或删除附加了活动预测的现有预测定义。

训练预测（控制台）

在向资产模型添加预测定义后，您可以对资产预测进行训练。

在中训练预测 Amazon IoT SiteWise

1. 导航至 Amazon IoT SiteWise 控制台。

2. 在导航窗格中，选择资产，然后选择您要监控的资产。
3. 选择预测。
4. 选择您要训练的预测。
5. 在操作下，选择开始训练，然后执行以下操作：
   a. 在 “预测详情” 下，选择允许 Amazon IoT SiteWise 与 Lookout for Equipment 共享您的资产数据的 IAM 权限角色。如果您需要创建一个新角色，请选择创建新角色。
   b. 在训练数据设置中，输入训练数据时间范围以选择用于训练预测的数据。
   c. （可选）选择后处理后数据的采样率。
   d. （可选）对于数据标签，请提供用于保存标注数据的 Amazon S3 存储桶和前缀。有关标记数据的更多信息，请参阅《Amazon Lookout for Equipment 用户指南》中的 Labeling your data。
   e. 选择下一步。
6. （可选）如果您希望预测在训练完成后立即变为活动状态，请在高级设置下，选择训练后自动激活预测，然后执行以下操作：
   a. 在输入数据下，对于数据上传频率，定义上传数据的频率，对于偏移延迟时间，定义要使用的缓冲区量。
   b. 选择下一步。
7. 查看预测的详细信息，然后选择保存并开始。

开始或停止对预测的推理（控制台）

> **Note**
>
> Lookout for Equipment 费用适用于在 Lookout for Equipment Amazon IoT SiteWise 之间传输数据的预定推断。有关更多信息，请参阅 Amazon Lookout for Equipment pricing。

如果您添加了预测 `lookoutequipment:CreateDataset`，但在训练后没有选择激活它，则必须将其激活才能开始监控您的资产。

要开始对预测进行推理，请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产，然后选择要添加预测的资产。

3. 选择预测。
4. 选择您要激活的预测。
5. 在操作下，选择开始推理，然后执行以下操作：
   a. 在输入数据下，对于数据上传频率，定义上传数据的频率，对于偏移延迟时间，定义要使用的缓冲区量。
   b. 选择保存并开始。

要停止对预测进行推理，请执行以下步骤：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择资产，然后选择要添加预测的资产。
3. 选择预测。
4. 选择您要停止的预测。
5. 在操作下，选择停止推理。

添加预测定义（CLI）

要在新资产模型或现有资产模型上定义预测定义，可以使用 Amazon Command Line Interface (Amazon CLI)。在资产模型上定义预测定义后，您可以使用 Lookout for Equipment 对资产的预测进行训练和计划推理， Amazon IoT SiteWise 以便进行异常检测。

先决条件

要完成这些步骤，您必须拥有一个资产模型并至少创建一个资产。有关更多信息，请参阅创建资产模型（Amazon CLI） 和创建资产（Amazon CLI）。

如果您不熟悉 Amazon IoT SiteWise，则必须调用 `CreateBulkImportJob` API 操作将资产属性值导入其中 Amazon IoT SiteWise，这将用于训练模型。有关更多信息，请参阅 创建 Amazon IoT SiteWise 批量导入任务 (Amazon CLI)。

要添加预测定义，请执行以下步骤：

1. 创建名为 `asset-model-payload.json` 的文件。按照其他部分中的步骤将资产模型的详细信息添加到文件中，但不要提交创建或更新资产模型的请求。
   - 有关如何创建资产模型的更多信息，请参阅创建资产模型（Amazon CLI）。

- 有关如何更新现有资产模型的更多信息，请参阅更新资产或组件模型（Amazon CLI）。

2. 通过添加以下代码，将 Lookout for Equipment 复合模型（`assetModelCompositeModels`）添加到资产模型中。

   - 将 *`Property`* 替换为您要包含的属性的 ID。要获得这些 IDs，请致电`DescribeAssetModel`。
   - *`RoleARN`*替换为允许 Lookout for Equipment 访问您的数据的 IAM 角色的 ARN。 Amazon IoT SiteWise

```
{
  ...
  "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. 创建资产模型或更新现有资产模型。请执行以下操作之一：

   - 要创建资源模型，请运行以下命令：

     ```
     aws iotsitewise create-asset-model --cli-input-json file://asset-model-
     payload.json
     ```

   - 要更新现有资源模型，请运行以下命令。将 *asset-model-id* 替换为您想要更新的资产模型的 ID。

     ```
     aws iotsitewise update-asset-model \
       --asset-model-id asset-model-id \
       --cli-input-json file://asset-model-payload.json
     ```

运行命令之后，请记下响应中的 `assetModelId`。

训练预测并开始推理（CLI）

现在，预测定义已经定义，您可以根据它训练资产并开始推理。如果您想训练预测但不想开始推理，请跳至训练预测（CLI）。要训练预测并开始对资产进行推理，您需要目标资源的 `assetId`。

要训练预测并开始推理，请执行以下步骤：

1. 运行以下命令以在 `assetModelCompositeModelSummaries` 下查找 `assetModelCompositeModelId`。将 *asset-model-id* 替换为您在 更新资产或组件模型（Amazon CLI）中创建的资产模型的 ID。

   ```
   aws iotsitewise describe-asset-model \
     --asset-model-id asset-model-id \
   ```

2. 运行以下命令以查找 `TrainingWithInference` 操作的 `actionDefinitionId`。将 *asset-model-id* 替换为上一步中使用的 ID，将 *asset-model-composite-model-id* 替换为上一步中返回的 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. 创建名为 `train-start-inference-prediction.json` 的文件并添加以下代码，完成以下替换：

   - 将 *asset-id* 替换为目标资产的 ID
   - *action-definition-id*带有动 TrainingWithInference 作的 ID
   - 将 *StartTime* 替换为训练数据的开头部分，以纪元秒格式提供
   - 将 *EndTime* 替换为训练数据的结尾部分，以纪元秒格式提供
   - 将 *TargetSamplingRate* 替换为 Lookout for Equipment 进行后处理后的数据采样率。支持的值为：`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. 运行以下命令以开始训练和推理：

```
aws iotsitewise execute-action --cli-input-json file://train-start-inference-
prediction.json
```

训练预测（CLI）

现在，预测定义已经定义，您可以根据它训练资产。要训练资产的预测，您需要目标资源的 `assetId`。

要训练预测，请执行以下步骤：

1. 运行以下命令以在 `assetModelCompositeModelSummaries` 下查找 `assetModelCompositeModelId`。将 *`asset-model-id`* 替换为您在 更新资产或组件模型（Amazon CLI）中创建的资产模型的 ID。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 运行以下命令以查找 `Training` 操作的 `actionDefinitionId`。将 *`asset-model-id`* 替换为上一步中使用的 ID，将 *`asset-model-composite-model-id`* 替换为上一步中返回的 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. 创建名为 `train-prediction.json` 的文件并添加以下代码，完成以下替换：

   - 将 *`asset-id`* 替换为目标资产的 ID
   - 将 *`action-definition-id`* 替换为训练动作的 ID
   - 将 *`StartTime`* 替换为训练数据的开头部分，以纪元秒格式提供
   - 将 *`EndTime`* 替换为训练数据的结尾部分，以纪元秒格式提供
   - （可选）将 *`BucketName`* 替换为存放您的标签数据的 Amazon S3 存储桶的名称
   - （可选）将 *`Prefix`* 替换为与 Amazon S3 存储桶关联的前缀。
   - 将 *`TargetSamplingRate`* 替换为 Lookout for Equipment 进行后处理后的数据采样率。支持的值为：`PT1S | PT5S | PT10S | PT15S | PT30S | PT1M | PT5M | PT10M | PT15M | PT30M | PT1H`。

   > **Note**
   >
   > 同时包括存储桶名称和前缀，或者两者都不包括在内。

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4ETraining\": {\"trainingMode\":
\"START\",\"exportDataStartTime\": StartTime, \"exportDataEndTime\": EndTime,
 \"targetSamplingRate\":\"TargetSamplingRate\"}, \"labelInputConfiguration\":
 {\"bucketName\": \"BucketName\", \"prefix\": \"Prefix\"}}}"
}
}
```

4. 运行以下命令以开始训练：

```
aws iotsitewise execute-action --cli-input-json file://train-prediction.json
```

在开始推理之前，必须先完成训练。若要检查训练的状态，请执行以下操作之一：

- 在控制台中，导航到预测所在的资产。
- 从中 Amazon CLI，`BatchGetAssetPropertyValue`使用该`propertyId``trainingStatus`属性的调用。

开始或停止对预测的推理（CLI）

训练完预测后，您即可开始推理，让 Lookout for Equipment 开始监控您的资产。要开始或停止推理，您需要目标资源的 `assetId`。

要开始推理，请执行以下步骤：

1. 运行以下命令以在 `assetModelCompositeModelSummaries` 下查找 `assetModelCompositeModelId`。将 *asset-model-id* 替换为您在 更新资产或组件模型（Amazon CLI）中创建的资产模型的 ID。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 运行以下命令以查找 Inference 操作的 actionDefinitionId。将 *asset-model-id* 替换为上一步中使用的 ID，将 *asset-model-composite-model-id* 替换为上一步中返回的 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. 创建名为 start-inference.json 的文件并添加以下代码，完成以下替换：

   - 将 *asset-id* 替换为目标资产的 ID
   - 将 *action-definition-id* 替换为开始推理操作的 ID
   - 将 *Offset* 替换为要使用的缓冲区量
   - 将 *Frequency* 替换为上传数据的频率

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4EInference\": {\"inferenceMode\":\"START
\",\"dataDelayOffsetInMinutes\": Offset, \"dataUploadFrequency\": \"Frequency\"}}"
}}
```

4. 使用以下命令以开始推理：

```
aws iotsitewise execute-action --cli-input-json file://start-inference.json
```

要停止推理，请执行以下步骤：

1. 运行以下命令以在 assetModelCompositeModelSummaries 下查找 assetModelCompositeModelId。将 *asset-model-id* 替换为您在 更新资产或组件模型（Amazon CLI）中创建的资产模型的 ID。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 运行以下命令以查找 Inference 操作的 actionDefinitionId。将 *asset-model-id* 替换为上一步中使用的 ID，将 *asset-model-composite-model-id* 替换为上一步中返回的 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. 创建名为 `stop-inference.json` 的文件并添加以下代码，完成以下替换：

   - 将 *asset-id* 替换为目标资产的 ID
   - 将 *action-definition-id* 替换为开始推理操作的 ID

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4EInference\":{\"inferenceMode\":\"STOP
\"}}"
}}
```

4. 使用以下命令以停止推理：

```
aws iotsitewise execute-action --cli-input-json file://stop-inference.json
```

管理中的数据存储 Amazon IoT SiteWise

您可以配置 Amazon IoT SiteWise 为将数据保存在以下存储层中：

热层

热存储层是 Amazon IoT SiteWise 托管的时间序列存储。Hot tier 对频繁访问的数据最有效， write-to-read延迟时间很短。存储在热层中的数据供需要快速访问设备中最新测量值的工业应用程序使用。这包括使用交互式控制面板可视化实时指标的应用程序，或监控操作并启动警报来识别性能问题的应用程序。

默认情况下，采集到的数据存储 Amazon IoT SiteWise 在热门层中。您可以为热层定义保留期，之后根据您的配置将热层中的数据 Amazon IoT SiteWise 移动到温层或冷层存储。为了获得最佳性能和成本效益，请将热层保留期设置为比经常检索数据所需的时间更长。这用于实时指标、警报和监控场景。如果未设置保留期，则数据将无限期存储在热层中。

温层

温存储层是一个 Amazon IoT SiteWise 托管层，可以有效地经济高效地存储历史数据。它最适合用于检索具有中等 write-to-read延迟特征的大量数据。使用温层来存储大型工作负载所需的历史数据。例如，它用于分析、商业智能应用程序（BI）、报告工具和机器学习（ML）模型训练的数据检索。如果您启用冷存储层，则可以定义温层保留期。保留期结束后，从温层中 Amazon IoT SiteWise 删除数据。

冷层

冷存储层使用 Amazon S3 存储桶来存储很少使用的数据。启用冷层后， Amazon IoT SiteWise 每 6 小时复制一次时间序列，包括测量值、指标、转换和聚合，以及资产模型定义。冷层用于存储可容忍历史报告和备份高读取延迟的数据。

主题

- 在中配置存储设置 Amazon IoT SiteWise
- 对存储设置进行故障排除 Amazon IoT SiteWise
- 冷层中保存的数据的文件路径和架构

在中配置存储设置 Amazon IoT SiteWise

您可以配置存储设置，以启用服务托管的温层存储，并将数据复制到冷层。要详细了解温层和热层的保留期，请参阅数据留存的影响。配置存储设置时，请执行以下操作：

- 热层保留 — 设置保留期，规定数据在删除之前在热层中存储多长时间，然后根据您的存储设置将其移至服务托管的温层存储或冷层存储。 Amazon IoT SiteWise 将删除在保留期结束之前存在的热门层中的所有数据。如果未设置保留期，则数据将无限期存储在热层中。
- 热层保留 — 设置保留期，规定您的数据在从存储中删除并移至客户管理的冷层 Amazon IoT SiteWise 存储之前在温层中存储多长时间。 Amazon IoT SiteWise 从保留期结束之前存在的暖层中删除所有数据。如果未设置保留期，则数据将无限期存储在温层中。

> **Note**
>
> 要提高查询性能，请使用温层存储设置热层保留期。

数据留存在热层和温层存储中的影响

- 缩短热层存储的保留期时，数据将从热层永久迁移到温层或冷层。缩短温层的保留期时，数据将移至冷层，并从温层中永久删除。
- 当您延长热层或温层存储的保留期时，更改会影响 Amazon IoT SiteWise 从那时起发送到的数据。Amazon IoT SiteWise 不会从温存储或冷存储中检索数据以填充热层。例如，如果热层存储的保留期最初设置为 30 天，然后增加到 60 天，则热层存储需要 30 天才能包含 60 天的数据。

主题

- 为温层配置存储设置（控制台）
- 为温层配置存储设置（Amazon CLI）
- 配置冷层的存储设置（控制台）
- 为冷层配置存储设置（Amazon CLI）

为温层配置存储设置（控制台）

以下过程向您展示如何在 Amazon IoT SiteWise 控制台中配置存储设置以将数据复制到温层。

若要配置控制台中的存储设置

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中的设置下，选择存储。
3. 在右上角，选择 编辑。
4. 在编辑存储页面中，执行以下操作：
5. 对于热层设置，执行以下操作：

   - 如果要在数据被删除并迁移至服务托管的温层存储中前，为数据在热层中的存储时间设置保留期，请选择启用保留期。
   - 要配置保留期，请输入整数并选择单位。保留期必须大于或等于 30 天。

   Amazon IoT SiteWise 删除热门层中所有早于保留期的数据。如果未设置保留期，则数据将无限期存储。
6. （推荐）对于温层设置，请执行以下操作：

   - 要选择加入温层存储，请选择我确认选择加入温层存储，选择加入温层存储。
   - （可选）要配置保留期，请输入一个整数并选择单位。保留期必须大于或等于 365 天。

   Amazon IoT SiteWise 删除在保留期之前存在的温层中的数据。如果未设置保留期，则数据将无限期存储。

   > **Note**
   >
   > - 当您选择使用温层时，配置仅显示一次。
   > - 要设置热层保留期，您必须有温层或冷层存储。为了提高成本效益和检索历史数据，Amazon IoT SiteWise 建议您将长期数据存储在温层。
   > - 要设置温层保留，您必须有冷层存储。
7. 选择保存以保存您的存储设置。

在 Amazon IoT SiteWise 存储部分中，温层存储处于以下状态之一：

- 已启用 - 如果您的数据在热层保留期之前就已存在，则 Amazon IoT SiteWise 会将数据移至温层。
- 已禁用 – 温层存储已禁用。

为温层配置存储设置（Amazon CLI）

您可以使用 Amazon CLI 和以下命令配置存储设置以将数据移动到温层。

为防止覆盖现有配置，请运行以下命令来检索当前的存储配置信息：

```
aws iotsitewise describe-storage-configuration
```

Example 无现有冷层配置的响应

```
{
          "storageType": "SITEWISE_DEFAULT_STORAGE",
          "disassociatedDataStorage": "ENABLED",
          "configurationStatus": {
              "state": "ACTIVE"
          },
          "lastUpdateDate": "2021-10-14T15:53:35-07:00",
          "warmTier": "DISABLED"
}
```

Example 包含现有冷层配置的响应

```
{
      "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"
}
```

使用配置暖层的存储设置 Amazon CLI

要配置存储设置，请运行以下命令。`file-name`替换为包含 Amazon IoT SiteWise 存储配置的文件的名称。

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example Amazon IoT SiteWise 带热层和温层的配置

```
{
             "storageType": "SITEWISE_DEFAULT_STORAGE",
             "disassociatedDataStorage": "ENABLED",
             "warmTier": "ENABLED",
             "retentionPeriod": {
                "numberOfDays": hot-tier-retention-in-days
              }

}
```

`hot-tier-retention-in-days` 必须是大于或等于 30 天的整数。

Example 响应

```
{
             "storageType": "SITEWISE_DEFAULT_STORAGE",
             "configurationStatus": {
             "state": "UPDATE_IN_PROGRESS"
             }
}
```

如果您启用了冷层存储，请参阅使用 Amazon CLI 和现有冷层配置存储设置。

使用 Amazon CLI 和现有冷层配置存储设置

使用 Amazon CLI 现有冷层存储配置存储设置

- 要配置存储设置，请运行以下命令。将 *file-name* 替换为包含 Amazon IoT SiteWise 存储配置的文件的名称。

  ```
  aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
  ```

Example Amazon IoT SiteWise 存储配置

- 将 *amzn-s3-demo-bucket* 替换为您的 Amazon S3 桶名称。
- *prefix*替换为您的 Amazon S3 前缀。
- *aws-account-id*替换为您的 Amazon 账户 ID。
- *role-name*替换为允许 Amazon IoT SiteWise 向 Amazon S3 发送数据的 Amazon S3 访问角色的名称。
- *hot-tier-retention-in-days*替换为大于或等于 30 天的整数。
- *warm-tier-retention-in-days*替换为大于或等于 365 天的整数。

> **Note**
>
> Amazon IoT SiteWise 将删除温层中所有早于冷层保留期的数据。如果未设置保留期，则数据将无限期存储。

```
{
      "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 响应

```
{
```

```
      "storageType": "MULTI_LAYER_STORAGE",
      "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
       }
}
```

配置冷层的存储设置（控制台）

以下过程向您展示如何在 Amazon IoT SiteWise 控制台中配置存储设置以将数据复制到冷层。

若要配置控制台中的存储设置

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中的设置下，选择存储。
3. 在右上角，选择 编辑。
4. 在编辑存储页面中，执行以下操作：

   a. 对于 存储设置，选择 启用冷层存储。默认情况下，冷层存储处于禁用状态。

   b. 对于 S3 存储桶位置，输入现有 Amazon S3 存储桶的名称和前缀。

   > **Note**
   >
   > - Amazon S3 使用前缀作为 Amazon S3 存储桶中的文件夹名称。前缀必须包含 1-255 个字符，并以正斜杠 (/) 结尾。您的 Amazon IoT SiteWise 数据将保存在此文件夹中。
   > - 如果您没有 Amazon S3 存储桶，请选择 查看，然后在 Amazon S3 控制台中创建一个存储桶。有关更多信息，请参阅 Amazon S3 用户指南中的创建您的第一个 S3 存储桶。

   c. 对于 S3 访问角色，执行下列操作之一：

   - 选择从 Amazon 托管模板创建角色， Amazon 自动创建允许向 Amazon S3 Amazon IoT SiteWise 发送数据的 IAM 角色。
   - 选择使用现有角色，然后从列表中选择您创建的角色。

> **Note**
>
> - 您必须对 S3 存储桶位置使用与您在上一步和 IAM Policy 中使用的相同的 Amazon S3 存储桶名称。
> - 确保您的角色具有以下示例中显示的权限。
>
> Example 权限策略：
>
> ```
> {
>  "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/*"
>  ]
>  }
>  ]
>  }
> ```
>
> 将 amzn-s3-demo-bucket 替换为您的 Amazon S3 存储桶的名称。
>
> - 如果使用客户托管的 KMS 密钥对 Amazon S3 存储桶进行加密，那么 KMS 密钥必须具有访问策略，该策略必须具有 IAM 角色`kms:Decrypt`和`kms:GenerateDataKey`操作。

d. 要设置热层，请参阅为温层配置存储设置（控制台）中的步骤 5。

e. （可选）对于 Amazon IoT Analytics 集成，请执行以下操作。

   i. 如果要使用 Amazon IoT Analytics 来查询数据，请选择已启用 Amazon IoT Analytics 数据存储。

   ii. Amazon IoT SiteWise 为您的数据存储生成一个名称，也可以输入其他名称。

Amazon IoT SiteWise 自动在中创建数据存储 Amazon IoT Analytics 以保存您的数据。要查询数据，您可以使用 Amazon IoT Analytics 创建数据集。有关更多信息，请参阅《Amazon IoT Analytics 用户指南》中的使用 Amazon IoT SiteWise 数据。

f. 选择保存。

在 Amazon IoT SiteWise 存储部分中，冷层存储可以是以下值之一：

- 已启用- Amazon IoT SiteWise 将您的数据复制到指定的 Amazon S3 存储桶。
- 启用- Amazon IoT SiteWise 正在处理您的启用冷层存储的请求。此过程可能需要几分钟才能完成。
- Enable_Failed — Amazon IoT SiteWise 无法处理您的启用冷层存储的请求。如果您启用 Amazon IoT SiteWise 向 Amazon Logs 发送 CloudWatch 日志，则可以使用这些日志来解决问题。有关更多信息，请参阅 使用 Amazon CloudWatch 日志进行监控。
- 已禁用 – 冷层存储已禁用。

为冷层配置存储设置（Amazon CLI）

以下过程演示如何配置存储设置以使用 Amazon CLI将数据复制到冷层。

要使用配置存储设置 Amazon CLI

1. 要将数据导出到您账户中的 Amazon S3 存储桶，请运行以下命令以配置存储设置。*file-name*替换为包含 Amazon IoT SiteWise 存储配置的文件的名称。

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example Amazon IoT SiteWise 存储配置

- 将 *amzn-s3-demo-bucket* 替换为您的 Amazon S3 桶名称。
- *prefix*替换为您的 Amazon S3 前缀。
- *aws-account-id*替换为您的 Amazon 账户 ID。
- *role-name*替换为允许 Amazon IoT SiteWise 向 Amazon S3 发送数据的 Amazon S3 访问角色的名称。
- *retention-in-days*替换为大于或等于 30 天的整数。

```
{
      "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

- 您必须在存储配置和 IAM 策略中使用相同的 Amazon S3 Amazon IoT SiteWise 存储桶名称。
- 确保您的角色具有以下示例中显示的权限。

  Example 权限策略：

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

```
        ]
    }
```

将 amzn-s3-demo-bucket 替换为您的 Amazon S3 存储桶的名称。

- 如果使用客户托管的 KMS 密钥对 Amazon S3 存储桶进行加密，那么 KMS 密钥必须具有访问策略，该策略必须具有 IAM 角色`kms:Decrypt`和`kms:GenerateDataKey`操作。

Example 响应

```
{
    "storageType": "MULTI_LAYER_STORAGE",
    "retentionPeriod": {
        "numberOfDays": 100,
        "unlimited": false
    },
    "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
    }
}
```

> **Note**
>
> 更新存储配置可能需要几分钟。 Amazon IoT SiteWise

2. 若要检索存储配置信息，请运行以下命令。

```
aws iotsitewise describe-storage-configuration
```

Example 响应

```
{
      "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"
    }
```

3. 要停止将数据导出到 Amazon S3 存储桶，请运行以下命令以配置存储设置。

```
aws iotsitewise put-storage-configuration --storage-type SITEWISE_DEFAULT_STORAGE
```

> **Note**
>
> 默认情况下，您的数据仅存储在热门层中 Amazon IoT SiteWise。

Example 响应

```
{
        "storageType": "SITEWISE_DEFAULT_STORAGE",
        "configurationStatus": {
            "state": "UPDATE_IN_PROGRESS"
        }
    }
```

4. 若要检索存储配置信息，请运行以下命令。

```
aws iotsitewise describe-storage-configuration
```

Example 响应

```
{
        "storageType": "SITEWISE_DEFAULT_STORAGE",
        "configurationStatus": {
            "state": "ACTIVE"
        },
        "lastUpdateDate": "2021-03-30T15:57:14-07:00"
```

```
}
```

（可选）创建 Amazon IoT Analytics 数据存储 (Amazon CLI)

Amazon IoT Analytics 数据存储是一种可扩展且可查询的存储库，用于接收和存储数据。您可以使用 Amazon IoT SiteWise 控制台或 Amazon IoT Analytics APIs 创建 Amazon IoT Analytics 数据存储来保存 Amazon IoT SiteWise 数据。要查询数据，请使用创建数据集 Amazon IoT Analytics。有关更多信息，请参阅《Amazon IoT Analytics 用户指南》中的使用 Amazon IoT SiteWise 数据。

以下步骤 Amazon CLI 用于在中创建数据存储 Amazon IoT Analytics。

若要创建数据集，运行以下命令。*file-name*替换为包含数据存储配置的文件的名称。

```
aws iotanalytics create-datastore --cli-input-json file://file-name.json
```

> **Note**
>
> - 您必须指定现有 Amazon S3 存储桶的名称。如果没有 Amazon S3 存储桶，请先创建一个。有关更多信息，请参阅 Amazon S3 用户指南中的创建您的第一个 S3 存储桶。
> - 您必须在存储配置、IAM 策略和 Amazon IoT Analytics 数据 Amazon IoT SiteWise 存储配置中使用相同的 Amazon S3 存储桶名称。

Example Amazon IoT Analytics 数据存储配置

将*data-store-name*和*amzn-s3-demo-bucket*替换为您的 Amazon IoT Analytics 数据存储名称和 Amazon S3 存储桶名称。

```
{
      "datastoreName": "data-store-name",
      "datastoreStorage": {
          "iotSiteWiseMultiLayerStorage": {
              "customerManagedS3Storage": {
                  "bucket": "amzn-s3-demo-bucket"
              }
          }
      },
      "retentionPeriod": {
```

```
            "numberOfDays": 90
        }
    }
```

Example 响应

```
{
        "datastoreName": "datastore_IoTSiteWise_demo",
        "datastoreArn": "arn:aws:iotanalytics:us-west-2:123456789012:datastore/
datastore_IoTSiteWise_demo",
        "retentionPeriod": {
            "numberOfDays": 90,
            "unlimited": false
        }
    }
```

对存储设置进行故障排除 Amazon IoT SiteWise

使用以下信息对存储配置进行故障诊断并解决问题。

事务

- 错误：存储桶不存在
- 错误：访问 Amazon S3 路径被拒
- 错误：无法代入角色 ARN
- 错误：无法访问跨区域 Amazon S3 存储桶

错误：存储桶不存在

解决方案：找 Amazon IoT SiteWise 不到您的 Amazon S3 存储桶。确保您输入当前区域中现有 Amazon S3 存储桶的名称。

错误：访问 Amazon S3 路径被拒

解决方案： Amazon IoT SiteWise 无法访问您的 Amazon S3 存储桶。执行以下操作：

- 确保您使用的 Amazon S3 存储桶与您在 IAM Policy 中指定的相同。
- 确保您的角色具有以下示例中显示的权限。

Example 权限策略

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

将 amzn-s3-demo-bucket 替换为您的 Amazon S3 存储桶的名称。

错误：无法代入角色 ARN

解决方案： Amazon IoT SiteWise 无法代表您担任 IAM 角色。确保您的角色信任以下服务：`iotsitewise.amazonaws.com`。有关更多信息，请参阅 《IAM 用户指南》中的我无法带入角色。

错误：无法访问跨区域 Amazon S3 存储桶

解决方案：您指定的 Amazon S3 存储桶位于不同的 Amazon 区域。确保您的 Amazon S3 存储桶和 Amazon IoT SiteWise 资产位于同一区域。

冷层中保存的数据的文件路径和架构

Amazon IoT SiteWise 通过复制时间序列（包括度量、指标、转换和聚合，以及资产和资产模型定义）将您的数据存储在冷层中。下面介绍发送到冷层的数据的文件路径和架构。

主题

- 设备数据（测量）
- 指标、转换和聚合
- 资产元数据
- 资产层次结构元数据
- 存储数据索引文件

设备数据（测量）

Amazon IoT SiteWise 每六小时将设备数据（测量值）导出到冷层一次。原始数据以 Apache AVRO (`.avro`) 格式保存在冷层中。

文件路径

Amazon IoT SiteWise 使用以下模板将设备数据（测量值）存储在冷层中。

```
{keyPrefix}/raw/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/raw_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Amazon S3 中原始数据的每个文件路径都包含以下组件。

路径组件	描述
`keyPrefix`	您在 Amazon IoT SiteWise 存储配置中指定的 Amazon S3 前缀。Amazon S3 使用前缀作为存储桶中的文件夹名称。
`raw`	存储设备（测量）中的时间序列数据的文件夹。`raw` 文件夹保存在前缀文件夹中。
`seriesBucket`	介于 00 和 ff 之间的十六进制数。此数字派生自 `timeSeriesId` 。此分区用于在 Amazon IoT SiteWise 写入冷层时增加吞吐量。当您使用 Amazon Athena 运行查询时，您可以使用该分区进行精细分区以提高查询性能。

路径组件	描述
	资产元数据中的 `seriesBucket` 和 `timeSeriesBucket` 是相同的数字。
`startYear`	与时间序列数据关联的独占开始时间的年份。
`startMonth`	与时间序列数据关联的独占开始时间的月份。
`startDay`	与时间序列数据关联的独占开始时间的月份中的某一天。
`fileName`	文件名使用下划线 (_) 字符作为分隔符来分隔以下内容： • `raw` 前缀。 • `timeSeriesId` 值。 • 与时间序列数据关联的独占开始时间的纪元时间戳。 • 数据质量。有效值：`GOOD`、`BAD` 和 `UNCERTAIN` 。有关更多信息，请参阅 Amazon IoT SiteWise API 参考中的 AssetPropertyValue。 该文件通过使用 Snappy 压缩以 `.avro` 格式保存。

Example 冷层中原始数据的文件路径

```
keyPrefix/raw/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
raw_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

字段

导出到冷层的原始数据的架构包含以下字段。

Amazon IoT SiteWise 建议客户在从冷层读取原始数据的系统上实现对架构演变的支持，因为将来可能会引入其他字段。

空数据表示为所有值字段均为空。但是，客户在使用查询时仍会收到正确的数据类型 Amazon IoT SiteWise APIs。

字段名称	支持的类型	默认类型	描述
`seriesId`	`string`	不适用	标识设备（度量）中的时间序列数据的 ID。您可以使用此字段在查询中联接原始数据和资产元数据。
`timeInSeconds`	`long`	不适用	采用 Unix 纪元时间格式的时间戳日期（以秒为单位）。分数纳秒数据由 `offsetInNanos` 提供。
`offsetInNanos`	`long`	不适用	与 `timeInSeconds` 的纳秒偏移量。
`quality`	`string`	不适用	时间序列值的质量。
`doubleValue`	`double` 或 `null`	`null`	双精度（浮点数）类型的时间序列数据。
`stringValue`	`string` 或 `null`	`null`	字符串类型（字符序列）的时间序列数据。
`integerValue`	`int` 或 `null`	`null`	整数类型（整数）的时间序列数据。
`booleanValue`	`boolean` 或 `null`	`null`	布尔类型的时间序列数据（true 或 false）。

字段名称	支持的类型	默认类型	描述
jsonValue	string 或 null	null	JSON 类型的时间序列数据（存储为字符串的复杂数据类型）。
recordVersion	long 或 null	null	记录的版本号。您可以使用版本号来选择最新记录。较新的记录具有较大的版本号。

Example 冷层中的原始数据


```
  {"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
  {"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
  {"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
  {"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
  {"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
```


指标、转换和聚合

Amazon IoT SiteWise 每六小时将指标、转换和聚合导出到冷层一次。指标、转换和聚合以 Apache AVRO (`.avro`) 格式保存在冷层中。

文件路径

Amazon IoT SiteWise 使用以下模板将指标、转换和聚合存储在冷层中。

```
{keyPrefix}/agg/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/agg_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Amazon S3 中指标、转换和聚合的每个文件路径都包含以下组件。

路径组件	描述
keyPrefix	您在 Amazon IoT SiteWise 存储配置中指定的 Amazon S3 前缀。Amazon S3 使用前缀作为存储桶中的文件夹名称。
agg	用于存储来自指标的时间序列数据的文件夹。agg 文件夹保存在前缀文件夹中。
seriesBucket	介于 00 和 ff 之间的十六进制数。此数字派生自 timeSeriesId 。此分区用于在 Amazon IoT SiteWise 写入冷层时增加吞吐量。当您使用 Amazon Athena 运行查询时，您可以使用该分区进行精细分区以提高查询性能。 资产元数据中的 seriesBucket 和 timeSeriesBucket 是相同的数字。
startYear	与时间序列数据关联的独占开始时间的年份。
startMonth	与时间序列数据关联的独占开始时间的月份。
startDay	与时间序列数据关联的独占开始时间的月份中的某一天。
fileName	文件名使用下划线 (_) 字符作为分隔符来分隔以下内容： • raw 前缀。 • timeSeriesId 值。 • 与时间序列数据关联的独占开始时间的纪元时间戳。

路径组件	描述
	• 数据质量。有效值：GOOD、BAD 和 UNCERTAIN 。有关更多信息，请参阅 Amazon IoT SiteWise API 参考中的 AssetPropertyValue。 该文件通过使用 Snappy 压缩以 .avro 格式保存。

Example 冷层中指标的文件路径

```
keyPrefix/agg/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
agg_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

字段

导出到冷层的指标、转换和聚合的架构包含以下字段。

字段名称	支持的类型	默认类型	描述
seriesId	string	不适用	标识来自设备、指标或转换的时序数据的 ID。您可以使用此字段在查询中联接原始数据和资产元数据。
timeInSeconds	long	不适用	采用 Unix 纪元时间格式的时间戳日期（以秒为单位）。分数纳秒数据由 offsetInNanos 提供。
offsetInNanos	long	不适用	与 timeInSeconds 的纳秒偏移量。

字段名称	支持的类型	默认类型	描述
quality	string	不适用	筛选资产数据所依据的质量。
resolution	string	不适用	聚合数据的时间间隔。
count	double 或 null	null	给定变量在当前时间间隔内的数据点总数。
average	double 或 null	null	给定变量在当前时间间隔内的平均值。
min	double 或 null	null	给定变量在当前时间间隔内的最小值。
max	boolean 或 null	null	给定变量在当前时间间隔内的最大值。
sum	string 或 null	null	给定变量在当前时间间隔内的值总和。
recordVersion	long 或 null	null	记录的版本号。您可以使用版本号来选择最新记录。较新的记录具有较大的版本号。

Example 冷层中的指标数据

```
{"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334060,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"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":"
```

```
{"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":"
{"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":"
{"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":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
```

资产元数据

首次启用 Amazon IoT SiteWise 将数据导出到冷层时，资产元数据将导出到冷层。初始配置完成后，只有在您更改资产模型定义或资产定义时，才会将资产元数据 Amazon IoT SiteWise 导出到该层。资产元数据以换行符分隔的 JSON（`.ndjson`）格式保存在冷层中。

文件路径

Amazon IoT SiteWise 使用以下模板将资产元数据存储在冷层中。

```
{keyPrefix}/asset_metadata/asset_{assetId}.ndjson
```

冷层中资产元数据的每个文件路径都包含以下组件。

路径组件	描述
`keyPrefix`	您在的存储配置中指定的 Amazon Amazon IoT SiteWise S3 前缀。Amazon S3 使用前缀作为存储桶中的文件夹名称。
`asset_metadata`	存储资产元数据的文件夹。`asset_metadata` 文件夹保存在前缀文件夹中。
`fileName`	文件名使用下划线 (_) 字符作为分隔符来分隔以下内容：

路径组件	描述
	• asset 前缀。 • assetId 值。 该文件以 .ndjson 格式保存。

Example 较冷层中资产元数据的文件路径

```
keyPrefix/asset_metadata/asset_35901915-d476-4dca-8637-d9ed4df939ed.ndjson
```

字段

导出到冷层的资产元数据的模式包含以下字段。

字段名称	描述
assetId	资产的 ID。
assetName	资产的名称。
assetExternalId	资产的外部 ID。
assetModelId	用于创建该资产的资产模型的 ID。
assetModelName	资产模型的名称。
assetModelExternalId	资产模型的外部 ID。
assetPropertyId	资产属性的 ID。
assetPropertyName	资产属性的名称。
assetPropertyExternalId	资产属性的外部 ID。
assetPropertyDataType	资产属性的数据类型。
assetPropertyUnit	资产属性的单位（例如，Newtons 和 RPM）。

字段名称	描述
assetPropertyAlias	标识资产属性的别名，例如 OPC UA 服务器数据流路径（例如，/company/windfarm/3/turbine/7/temperature ）。
timeSeriesId	标识来自设备、指标或转换的时序数据的 ID。您可以使用此字段在查询中联接原始数据和资产元数据。
timeSeriesBucket	介于 00 和 ff 之间的十六进制数。此数字派生自 timeSeriesId 。此分区用于在 Amazon IoT SiteWise 写入冷层时增加吞吐量。当您使用 Amazon Athena 运行查询时，您可以使用该分区进行精细分区以提高查询性能。 原始数据的文件路径中的 timeSeriesBucket 和 seriesBucket 是相同的数字。
assetCompositeModelId	复合模型的 ID。
assetCompositeModelExternalId	复合模型的外部 ID。
assetCompositeModelDescription	复合模型的描述。
assetCompositeModelName	复合模型的名称。
assetCompositeModelType	复合模型的类型。对于警报复合模型，这种类型是 AWS/ALARM 。
assetCreationDate	资产的创建日期，以 Unix 纪元时间为单位。
assetLastUpdateDate	资产上次更新的日期，以 Unix 纪元为单位。
assetStatusErrorCode	错误代码。
assetStatusErrorMessage	错误消息。
assetStatusState	资产的当前状态。

Example 冷层中的资产元数据


```
  {"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":"95e63da7-d34e-43e1-
bc6f-1b490154b07a","assetPropertyExternalId":null,"assetPropertyName":"Temperature","assetPrope
Washington/Seattle/WT2/temp","timeSeriesId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeSeriesBucket":"f6","assetArn":null,"assetCompositeModelDescription":nul
  {"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":"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,"ass
  {"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":nu
e6db-40fa-9845-ed0dddd4c77d_8cf1162f-dead-4fbe-b468-
c8e24cde9f50","timeSeriesBucket":"d7","assetArn":null,"assetCompositeModelDescription":null,"as

 {"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,"as
```


资产层次结构元数据

当您首次启用 Amazon IoT SiteWise 在冷层保存数据时，资产层次结构元数据将导出到冷层。初始配置完成后，只有在对资产模型或资产定义进行更改时，才会将资产层次结构元数据 Amazon IoT SiteWise 导出到冷层。资产层次结构元数据以换行符分隔的 JSON（`.ndjson`）格式保存在冷层中。

通过调用 DescribeAssetAPI 可以检索层次结构、目标资产或源资产的外部标识符。

文件路径

Amazon IoT SiteWise 使用以下模板将资产层次结构元数据存储在冷层中。

```
{keyPrefix}/asset_hierarchy_metadata/{parentAssetId}_{hierarchyId}.ndjson
```

冷层中资产层次结构元数据的每个文件路径都包含以下组件。

路径组件	描述
keyPrefix	您在 Amazon IoT SiteWise 存储配置中指定的 Amazon S3 前缀。Amazon S3 使用前缀作为存储桶中的文件夹名称。
asset_hierarchy_metadata	存储资产层次结构元数据的文件夹。asset_hierarchy_metadata 文件夹保存在前缀文件夹中。
fileName	文件名使用下划线 (_) 字符作为分隔符来分隔以下内容： • parentAssetId 值。 • hierarchyId 值。 该文件以 .ndjson 格式保存。

Example 冷层中资产层次结构元数据的文件路径

```
keyPrefix/asset_hierarchy_metadata/35901915-d476-4dca-8637-
d9ed4df939ed_c5b3ced8-589a-48c7-9998-cdccfc9747a0.ndjson
```

字段

导出到冷层的资产层次结构元数据的架构包含以下字段。

字段名称	描述
sourceAssetId	此资产关系中源资产的 ID。
targetAssetId	此资产关系中目标资产的 ID。
hierarchyId	层次结构的 ID。
associationType	此资产关系的关联类型。 值必须为 CHILD。目标资产是源资产的子资产。

Example 冷层中的资产层次结构元数据

```
{"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"}
```

若要在冷层中查看您的数据

1. 导航到 Amazon S3 控制台。
2. 在导航窗格中，选择存储桶，然后选择您的 Amazon S3 存储桶。
3. 导航到包含原始数据、资产元数据或资产层次结构元数据的文件夹。
4. 选择文件，然后从操作中选择下载。

存储数据索引文件

Amazon IoT SiteWise 使用这些文件来优化数据查询性能。它们显示在您的 Amazon S3 存储桶中，但您不需要使用它们。

文件路径

Amazon IoT SiteWise 使用以下模板将数据索引文件存储在冷层中。

```
keyPrefix/index/series=timeseriesId/startYear=startYear/startMonth=startMonth/
startDay=startDay/index_timeseriesId_startTimestamp_quality
```

Example 数据存储索引文件的文件路径

```
keyPrefix/index/series=7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-
d34e-43e1-bc6f-1b490154b07a/startYear=2022/startMonth=02/startDay=03/
index_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1643846400_GOOD
```

使用的代码示 Amazon IoT SiteWise 例 Amazon SDKs

以下代码示例说明如何 Amazon IoT SiteWise 使用 Amazon 软件开发套件 (SDK)。

基础知识是向您展示如何在服务中执行基本操作的代码示例。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

你好 Amazon IoT SiteWise

以下代码示例展示了如何开始使用 Amazon IoT SiteWise。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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();
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListAssetModels中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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();
}
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListAssetModels中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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"))
```

- 有关 API 的详细信息，请参阅适用ListAssetModels于 Python 的Amazon SDK (Boto3) API 参考。

代码示例

- 使用的基本示 Amazon IoT SiteWise 例 Amazon SDKs
  - 你好 Amazon IoT SiteWise
  - 学习 Amazon IoT SiteWise 使用 Amazon SDK 的基础知识
  - Amazon IoT SiteWise 使用的操作 Amazon SDKs
    - BatchPutAssetPropertyValue与 Amazon SDK 或 CLI 配合使用
    - CreateAsset与 Amazon SDK 或 CLI 配合使用

- CreateAssetModel与 Amazon SDK 或 CLI 配合使用
- CreateGateway与 Amazon SDK 或 CLI 配合使用
- CreatePortal与 Amazon SDK 或 CLI 配合使用
- DeleteAsset与 Amazon SDK 或 CLI 配合使用
- DeleteAssetModel与 Amazon SDK 或 CLI 配合使用
- DeleteGateway与 Amazon SDK 或 CLI 配合使用
- DeletePortal与 Amazon SDK 或 CLI 配合使用
- DescribeAssetModel与 Amazon SDK 或 CLI 配合使用
- DescribeGateway与 Amazon SDK 或 CLI 配合使用
- DescribePortal与 Amazon SDK 或 CLI 配合使用
- GetAssetPropertyValue与 Amazon SDK 或 CLI 配合使用
- ListAssetModels与 Amazon SDK 或 CLI 配合使用

使用的基本示 Amazon IoT SiteWise 例 Amazon SDKs

以下代码示例说明如何使用 with 的基础 Amazon IoT SiteWise 知识 Amazon SDKs。

示例

- 你好 Amazon IoT SiteWise
- 学习 Amazon IoT SiteWise 使用 Amazon SDK 的基础知识
- Amazon IoT SiteWise 使用的操作 Amazon SDKs
  - BatchPutAssetPropertyValue与 Amazon SDK 或 CLI 配合使用
  - CreateAsset与 Amazon SDK 或 CLI 配合使用
  - CreateAssetModel与 Amazon SDK 或 CLI 配合使用
  - CreateGateway与 Amazon SDK 或 CLI 配合使用
  - CreatePortal与 Amazon SDK 或 CLI 配合使用
  - DeleteAsset与 Amazon SDK 或 CLI 配合使用
  - DeleteAssetModel与 Amazon SDK 或 CLI 配合使用
  - DeleteGateway与 Amazon SDK 或 CLI 配合使用
  - DeletePortal与 Amazon SDK 或 CLI 配合使用
  - DescribeAssetModel与 Amazon SDK 或 CLI 配合使用

- DescribeGateway与 Amazon SDK 或 CLI 配合使用
- DescribePortal与 Amazon SDK 或 CLI 配合使用
- GetAssetPropertyValue与 Amazon SDK 或 CLI 配合使用
- ListAssetModels与 Amazon SDK 或 CLI 配合使用

你好 Amazon IoT SiteWise

以下代码示例展示了如何开始使用 Amazon IoT SiteWise。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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();
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListAssetModels中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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();
}
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListAssetModels中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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"))
```

- 有关 API 的详细信息，请参阅适用ListAssetModels于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

学习 Amazon IoT SiteWise 使用 Amazon SDK 的基础知识

以下代码示例演示了如何：

- 创建 Amazon IoT SiteWise 资产模型。
- 创建 Amazon IoT SiteWise 资产。
- 检索属性 ID 值。
- 向 Amazon IoT SiteWise 资产发送数据。
- 检索 Amazon IoT SiteWise 资产属性的值。
- 创建 Amazon IoT SiteWise 门户。
- 创建 Amazon IoT SiteWise 网关。

- 描述网 Amazon IoT SiteWise 关。
- 删除资 Amazon IoT SiteWise 产。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

运行演示 Amazon IoT SiteWise 功能的交互式场景。

```
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!");
    }
}
```

Amazon IoT SiteWise SDK 方法的包装器类。

```
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

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
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
```

TSitewise封装动作的 Io Wrapp Amazon IoT SiteWise er 类。

```
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
```

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon IoT SiteWise 使用的操作 Amazon SDKs

以下代码示例演示了如何使用执行单个 Amazon IoT SiteWise 操作 Amazon SDKs。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon IoT SiteWise API 参考。

示例

- BatchPutAssetPropertyValue与 Amazon SDK 或 CLI 配合使用
- CreateAsset与 Amazon SDK 或 CLI 配合使用
- CreateAssetModel与 Amazon SDK 或 CLI 配合使用
- CreateGateway与 Amazon SDK 或 CLI 配合使用
- CreatePortal与 Amazon SDK 或 CLI 配合使用

- DeleteAsset与 Amazon SDK 或 CLI 配合使用
- DeleteAssetModel与 Amazon SDK 或 CLI 配合使用
- DeleteGateway与 Amazon SDK 或 CLI 配合使用
- DeletePortal与 Amazon SDK 或 CLI 配合使用
- DescribeAssetModel与 Amazon SDK 或 CLI 配合使用
- DescribeGateway与 Amazon SDK 或 CLI 配合使用
- DescribePortal与 Amazon SDK 或 CLI 配合使用
- GetAssetPropertyValue与 Amazon SDK 或 CLI 配合使用
- ListAssetModels与 Amazon SDK 或 CLI 配合使用

`BatchPutAssetPropertyValue`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `BatchPutAssetPropertyValue`。

CLI

Amazon CLI

将数据发送到资产属性

以下 `batch-put-asset-property-value` 示例将功率和温度数据发送到由属性别名标识的资产属性。

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

`batch-put-asset-property-value.json` 的内容：

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

输出：

```
{
    "errorEntries": []
}
```

有关更多信息，请参阅《物联网 SiteWise 用户指南》中的使用 SiteWise I Amazon oT API 提取数据。Amazon

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考BatchPutAssetPropertyValue中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考BatchPutAssetPropertyValue中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考BatchPutAssetPropertyValue中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

一个辅助函数，用于从值列表生成条目参数。

```
    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
```

以下是传递给辅助函数的值列表的示例。

```
        values = [
            {
```

```
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
```

- 有关 API 的详细信息，请参阅适用BatchPutAssetPropertyValue于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateAsset`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateAsset`。

CLI

Amazon CLI

创建资产

以下 `create-asset` 示例根据风电涡轮机资产模型创建风电涡轮机资产。

```
aws iotsitewise create-asset \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
    --asset-name "Wind Turbine 1"
```

输出：

```
{
    "assetId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "assetArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "assetStatus": {
        "state": "CREATING"
```

```
    }
}
```

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的创建资产。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateAsset中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateAsset中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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.
```

```
      }),
    );
    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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考CreateAsset中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用CreateAsset于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateAssetModel`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateAssetModel`。

CLI

Amazon CLI

创建资产模型

以下 `create-asset-model` 示例创建一个资产模型，该模型使用下列属性定义了一个风电涡轮机：

序列号-风力涡轮机的序列号发电量-风力涡轮机产生的功率数据流温度C-风力涡轮机的温度数据流以 CelsiusTemperature F为单位-映射的温度数据指向从摄氏度到华氏度

```
aws iotsitewise create-asset-model \
    --cli-input-json file://create-wind-turbine-model.json
```

`create-wind-turbine-model.json` 的内容：

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

输出：

```
{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelStatus": {
        "state": "CREATING"
    }
}
```

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的定义资产模型。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateAssetModel中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /**
```

```
    * 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());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateAssetModel中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考CreateAssetModel中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

以下是要传递给函数的属性列表的示例。

```
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
            ]
```

- 有关 API 的详细信息，请参阅适用CreateAssetModel于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateGateway`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateGateway`。

CLI

Amazon CLI

创建网关

以下`create-gateway`示例创建了一个在 IoT Greengrass 上运行的 Amazon 网关。

```
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}
```

输出：

```
{
    "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
}
```

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的配置网关。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateGateway中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /**
     * 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();
            });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateGateway中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考CreateGateway中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用CreateGateway于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreatePortal`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreatePortal`。

CLI

Amazon CLI

创建门户

以下 `create-portal` 示例为一家风电场公司创建一个门户网站。您只能在启用 Amazon 单点登录的同一地区创建门户。

```
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
```

输出：

```
{
    "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"
}
```

有关更多信息，请参阅《物联网用户指南》中的 “Amazon 物联网 SiteWise 监控器入门” 和 “Amazon 物联网 SiteWise 用户指南” 中的 “Amazon 启用 Amazon SSO”。 SiteWise

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreatePortal中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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();
            });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreatePortal中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考CreatePortal中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用CreatePortal于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteAsset`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteAsset`。

CLI

Amazon CLI

删除资产

以下 `delete-asset` 示例删除一个风电涡轮机资产。

```
aws iotsitewise delete-asset \
    --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

输出：

```
{
    "assetStatus": {
        "state": "DELETING"
    }
}
```

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的删除资产。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteAsset中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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);
               }
           });
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeleteAsset中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DeleteAsset中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteAsset于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteAssetModel`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteAssetModel`。

CLI

Amazon CLI

删除资产模型

以下 `delete-asset-model` 示例删除了一个风电涡轮机资产模型。

```
aws iotsitewise delete-asset-model \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

输出：

```
{
    "assetModelStatus": {
        "state": "DELETING"
    }
}
```

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的删除资产模型。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteAssetModel中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeleteAssetModel中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DeleteAssetModel中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteAssetModel于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteGateway`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteGateway`。

CLI

Amazon CLI

删除网关

以下 `delete-gateway` 示例删除网关。

```
aws iotsitewise delete-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《物Amazon 联网 SiteWise 用户指南》中的使用网关摄取数据。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteGateway中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /**
     * 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());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeleteGateway中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DeleteGateway中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DeleteGateway于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeletePortal`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeletePortal`。

CLI

Amazon CLI

删除门户

以下 `delete-portal` 示例删除一家风电场公司的门户网站。

```
aws iotsitewise delete-portal \
    --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

输出：

```
{
    "portalStatus": {
        "state": "DELETING"
    }
}
```

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的删除门户。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeletePortal中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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());
               }
           });
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeletePortal中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
```

```
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DeletePortal中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DeletePortal于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeAssetModel`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeAssetModel`。

CLI

Amazon CLI

描述资产模型

以下 `describe-asset-model` 示例描述一个风电场资产模型。

```
aws iotsitewise describe-asset-model \
```

```
    --asset-model-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

输出：

```
{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "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"
    }
}
```

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的描述特定资产模型。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeAssetModel中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /**
     * 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
```

```
    *         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;
               }
           });
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DescribeAssetModel中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DescribeAssetModel中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeGateway`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeGateway`。

CLI

Amazon CLI

描述网关

以下 `describe-gateway` 示例描述网关。

```
aws iotsitewise describe-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

输出：

```
{
    "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
```

```
}
```

有关更多信息，请参阅《物Amazon 联网 SiteWise 用户指南》中的使用网关摄取数据。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeGateway中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DescribeGateway中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DescribeGateway中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DescribeGateway于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribePortal`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribePortal`。

CLI

Amazon CLI

描述门户

以下 `describe-portal` 示例描述一家风电场公司的门户网站。

```
aws iotsitewise describe-portal \
    --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

输出：

```
{
    "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"
}
```

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的管理门户。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribePortal中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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();
           });
```

```
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DescribePortal中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DescribePortal中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用DescribePortal于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

GetAssetPropertyValue与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetAssetPropertyValue。

CLI

Amazon CLI

检索资产属性的当前值

以下 get-asset-property-value 示例检索风电涡轮机资产的当前总功率。

```
aws iotsitewise get-asset-property-value \
    --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \
    --property-id a1b2c3d4-5678-90ab-cdef-66666EXAMPLE
```

输出：

```
{
    "propertyValue": {
        "value": {
            "doubleValue": 6890.8677520453875
        },
        "timestamp": {
            "timeInSeconds": 1580853000,
            "offsetInNanos": 0
        },
        "quality": "GOOD"
    }
}
```

有关更多信息，请参阅《物Amazon 联网 SiteWise 用户指南》中的查询当前资产属性值。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考GetAssetPropertyValue中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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();
                });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考GetAssetPropertyValue中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考GetAssetPropertyValue中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用GetAssetPropertyValue于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListAssetModels`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListAssetModels`。

CLI

Amazon CLI

列出所有资产模型

以下`list-asset-models`示例列出了您在当前地区的 Amazon 账户中定义的所有资产模型。

```
aws iotsitewise list-asset-models
```

输出：

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

有关更多信息，请参阅《Amazon 物联网 SiteWise 用户指南》中的列出所有资产模型。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ListAssetModels中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /**
    * 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;
                });
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListAssetModels中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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;
    }
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListAssetModels中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
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
```

- 有关 API 的详细信息，请参阅适用ListAssetModels于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

安全性 Amazon IoT SiteWise

云安全 Amazon 是重中之重。作为 Amazon 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 Amazon 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础架构。 Amazon 还为您提供可以安全使用的服务。作为的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用的合规计划 Amazon IoT SiteWise，请参阅Amazon 按合规规计划划分）。
- 云端安全-您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用时如何应用分担责任模型 Amazon IoT SiteWise。以下主题向您介绍如何进行配置 Amazon IoT SiteWise 以满足您的安全和合规性目标。您还将学习如何使用其他 Amazon 服务来帮助您监控和保护您的 Amazon IoT SiteWise 资源。

主题

- 中的数据保护 Amazon IoT SiteWise
- 中的数据加密 Amazon IoT SiteWise
- 的身份和访问管理 Amazon IoT SiteWise
- 合规性验证 Amazon IoT SiteWise
- 韧性在 Amazon IoT SiteWise
- 中的基础设施安全 Amazon IoT SiteWise
- 中的配置和漏洞分析 Amazon IoT SiteWise
- 的 VPC 终端节点 Amazon IoT SiteWise
- 以下方面的安全最佳实践 Amazon IoT SiteWise

中的数据保护 Amazon IoT SiteWise

分 Amazon 分担责任模型适用于中的数据保护 Amazon IoT SiteWise。如本模型所述 Amazon ，负责保护运行所有内容的全球基础架构 Amazon Web Services 云。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 Amazon Web Services 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon IAM Identity Center 或 Amazon Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与资源通信。 Amazon 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 Amazon CloudTrail。有关使用 CloudTrail 跟踪捕获 Amazon 活动的信息，请参阅《Amazon CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 Amazon 加密解决方案以及其中的所有默认安全控件 Amazon Web Services 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 Amazon 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API Amazon IoT SiteWise 或以其他 Amazon Web Services 服务 方式使用控制台 Amazon CLI、API 或时 Amazon SDKs。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

主题

- 网际流量隐私 Amazon IoT SiteWise
- Amazon IoT SiteWise 助理商务服务改进

网际流量隐私 Amazon IoT SiteWise

Amazon IoT SiteWise 与本地应用程序（例如 SiteWise Edge 网关）之间的连接通过传输层安全 (TLS) 连接进行保护。有关更多信息，请参阅 传输中的数据加密 Amazon IoT SiteWise。

Amazon IoT SiteWise 不支持 Amazon 区域之间的连接、区域内可用区域之间的连接或 Amazon 账户之间的连接。

Amazon IoT SiteWise 助理商务服务改进

Amazon IoT SiteWise Assistant 不会使用客户数据来改善服务或改善底层数据 LLMs。

中的数据加密 Amazon IoT SiteWise

数据加密是指保护传输中的数据（当数据在 SiteWise 边缘网关和服务器之间往返传输时）和静态数据（存储在本地设备或 Amazon 服务中时）。 Amazon IoT SiteWise您可以使用传输层安全性 (TLS) 保护传输中的数据，或使用客户端加密保护静态数据。

> **Note**
>
> Amazon IoT SiteWise 托管在 Edge 网关中并可通过本地网络访问的 SiteWise 边缘处理公开 APIs 信息。 APIs 它们通过由 Amazon IoT SiteWise Edge 连接器拥有的服务器证书支持的 TLS 连接公开。对于客户端身份验证，它们 APIs 使用访问控制密码。服务器证书私钥和访问控制密码都存储在磁盘上。 Amazon IoT SiteWise 边缘处理依赖文件系统加密来确保这些静态凭证的安全。

有关服务器端加密和客户端加密的更多信息，请查看下面列出的主题。

主题

- 静态加密 Amazon IoT SiteWise
- 传输中的数据加密 Amazon IoT SiteWise
- 中的密钥管理 Amazon IoT SiteWise

静态加密 Amazon IoT SiteWise

Amazon IoT SiteWise 将您的数据存储在 Amazon 云端和 Amazon IoT SiteWise 边缘网关上。

Amazon 云端的静态数据

Amazon IoT SiteWise 将数据存储在默认情况下对静态数据进行加密的其他 Amazon 服务中。静态加密与 Amazon Key Management Service (Amazon KMS) 集成，用于管理加密密钥，该密钥用于加密您的资产属性值和聚合中的值 Amazon IoT SiteWise。您可以选择使用客户托管密钥加密 Amazon IoT SiteWise中的资产属性值和聚合值。您可以通过 Amazon KMS创建、管理和查看您的加密密钥。

您可以选择一个 Amazon 拥有的密钥 来加密您的数据，也可以选择客户管理的密钥来加密您的资产属性值和汇总值：

工作方式

静态加密与集成， Amazon KMS 用于管理用于加密数据的加密密钥。

- Amazon 拥有的密钥 — 默认加密密钥。 Amazon IoT SiteWise 拥有这把钥匙。您无法在 Amazon 账户中查看此密钥。您也无法在 Amazon CloudTrail 日志中查看对密钥的操作。使用此密钥不会产生额外的费用。
- 客户托管密钥 - 此密钥存储在您的账户中，由您创建、拥有和管理。您对 KMS 密钥拥有完全控制权。需 Amazon KMS 支付额外费用。

Amazon 拥有的密钥

Amazon 拥有的密钥 未存储在您的账户中。它们是 KMS 密钥集合的一部分，这些密钥 Amazon 拥有并管理，可在多个 Amazon 账户中使用。 Amazon 服务可以 Amazon 拥有的密钥 用来保护您的数据。

您无法查看、管理 Amazon 拥有的密钥、使用或审核其使用情况。但是无需执行任何工作或更改任何计划即可保护用于加密数据的密钥。

如果您使用，则无需支付月费或使用费 Amazon 拥有的密钥，也不会计入您账户的 Amazon KMS 配额。

客户管理密钥

客户管理密钥是在您的账户中由您创建、拥有和管理的 KMS 密钥。您可以完全控制这些 KMS 密钥，例如：

- 制定和维护密钥策略、IAM 策略和授权
- 启用和禁用 KMS 密钥
- 轮换 KMS 密钥的加密材料
- 添加标签
- 创建引用 KMS 密钥的别名
- 安排删除 KMS 密钥

您还可以使用 CloudTrail 和 Amaz CloudWatch on Logs 来跟踪 Amazon KMS 代表您 Amazon IoT SiteWise 发送的请求。

如果您使用的是客户托管密钥，则需要授予对存储在您账户中的 KMS 密钥的 Amazon IoT SiteWise 访问权限。 Amazon IoT SiteWise 使用信封加密和密钥层次结构来加密数据。您的 Amazon KMS 加密密钥用于加密此密钥层次结构的根密钥。有关更多信息，请参阅《Amazon Key Management Service 开发人员指南》中的信封加密。

以下示例策略授予代表您创建客户托管密钥的 Amazon IoT SiteWise 权限。创建密钥时，需要允许 `kms:CreateGrant` 和 `kms:DescribeKey` 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1603902045292",
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

您创建的授权的加密上下文使用您的 `aws:iotsitewise:subscriberId` 和账户 ID。

SiteWise 边缘网关上的静态数据

Amazon IoT SiteWise 网关在本地文件系统上存储以下数据：

- OPC UA 源配置信息
- 来自已连接 OPC UA 源的 OPC UA 数据流路径集
- 当 SiteWise Edge 网关失去与互联网的连接时，工业数据会被缓存

SiteWise 边缘网关在上运行 Amazon IoT Greengrass。 Amazon IoT Greengrass 依靠 Unix 文件权限和全盘加密（如果启用）来保护核心上的静态数据。您负责确保文件系统和设备的安全性。

但是， Amazon IoT Greengrass 确实会加密从 Secrets Manager 检索到的 OPC UA 服务器密钥的本地副本。有关更多信息，请参阅 Amazon IoT Greengrass Version 1 开发人员指南中的密钥加密。

有关 Amazon IoT Greengrass 内核静态加密的更多信息，请参阅《Amazon IoT Greengrass Version 1 开发人员指南》中的静态加密。

传输中的数据加密 Amazon IoT SiteWise

Amazon IoT SiteWise 使用传输中的加密来保护您的设备、网关和 Amazon 云端之间传输的数据。与的通信使用 HT Amazon IoT SiteWise TPS 和 TLS 1.2 进行加密，从而确保您的数据保持机密并免受未经授权的访问或拦截。

有三种传输数据的通信模式：

- 通过互联网 — 本地设备（包括 SiteWise Edge 网关）之间的通信 Amazon IoT SiteWise 已加密。
- 通过本地网络 — SiteWise 应用程序网关和 SiteWise Edge 网关之间的 OpsHub 通信始终处于加密状态。浏览器中运行的 SiteWise 监控应用程序与 SiteWise Edge 网关之间的通信始终处于加密状态。可以对 SiteWise 边缘网关和 OPC UA 源之间的通信进行加密。
- SiteWise 边缘网关上的组件之间-边 SiteWise 缘网关上 Amazon IoT Greengrass 组件之间的通信未加密。

主题

- 通过 Internet 传输的数据
- 通过本地网络传输的数据
- 在 SiteWise Edge 上的本地组件之间传输的数据

通过 Internet 传输的数据

Amazon IoT SiteWise 使用传输层安全 (TLS) 加密互联网上的所有通信。发送到 Amazon 云端的所有数据均使用 MQTT 或 HTTPS 协议通过 TLS 连接发送，因此默认情况下它是安全的。 SiteWise 在上 Amazon IoT Greengrass运行的 Edge 网关和属性值通知使用 Amazon IoT 传输安全模型。有关更多信息，请参阅 Amazon IoT 开发人员指南中的传输安全。

通过本地网络传输的数据

SiteWise 边缘网关遵循 OPC UA 规范，以便与本地 OPC UA 源进行通信。您负责将源配置为使用加密传输中数据的消息安全模式。

边缘控制台应用程序与 Edg SiteWise e 网关之间的通信始终通过 TLS 加密。 SiteWise Edge 网关上的 SiteWise Edge 连接器生成并存储自签名证书，以便能够与边缘控制台建立 TLS 连接以供 Amazon IoT SiteWise 应用程序使用。在将应用程序连接到 SiteWise Edge 网关之前，您需要将此证书从 Edge 网关复制到 SiteWise Edge 控制台以供 Amazon IoT SiteWise 应用程序。这可确保 Amazon IoT SiteWise 应用程序的边缘控制台能够验证其是否已连接到您可信的 SiteWise Edge 网关。

除了用于保密和服务器真实性的 TLS 之外， SiteWise Edge 还使用 Sigv4 协议来确定 Edge 控制台应用程序的真实性。 SiteWise Edge 网关上的 SiteWise Edge 连接器接受并存储密码，以便能够验证来自边缘控制台应用程序、浏览器中运行的 SiteWise 监控应用程序以及其他基于 Amazon IoT SiteWise SDK 的客户端的传入连接。

在 SiteWise Edge 上的本地组件之间传输的数据

SiteWise 边缘网关在上运行 Amazon IoT Greengrass，它不会加密 Amazon IoT Greengrass 核心上本地交换的数据，因为数据不会离开设备。这包括 Amazon IoT SiteWise 连接器等 Amazon IoT Greengrass 组件之间的通信。想要了解更多信息，请参阅 Amazon IoT Greengrass Version 1 开发人员指南中的核心设备上的数据。

中的密钥管理 Amazon IoT SiteWise

Amazon IoT SiteWise 云密钥管理

默认情况下， Amazon IoT SiteWise Amazon 托管式密钥 用于保护您在 Amazon 云中的数据。您可以更新设置，使用客户托管密钥加密 Amazon IoT SiteWise中的某些数据。您可以通过 Amazon Key Management Service (Amazon KMS) 创建、管理和查看您的加密密钥。

Amazon IoT SiteWise 支持服务器端加密，使用存储在中的客户托管密钥 Amazon KMS 来加密以下数据：

- 资产属性值
- 汇总值

> **Note**
>
> 其他数据和资源使用默认加密进行加密，密钥由管理 Amazon IoT SiteWise。此密钥存储在 Amazon IoT SiteWise 账户中。

有关更多信息，请参阅什么是 Amazon Key Management Service？ 在《Amazon Key Management Service 开发人员指南》中。

使用由客户托管的密钥启用加密

要将客户托管密钥与一起使用 Amazon IoT SiteWise，您需要更新 Amazon IoT SiteWise 设置。

使用 KMS 密钥启用加密

1. 导航至 Amazon IoT SiteWise 控制台。
2. 选择账户设置，选择编辑 打开编辑账户设置页面。
3. 对于加密密钥类型，选择 选择其他 Amazon KMS 密钥。这对存储在 Amazon KMS中的客户托管密钥启用加密。

> **Note**
>
> 目前，您只能对资产属性值和聚合值使用客户托管密钥加密。

4. 从以下选项中选择您的 KMS 密钥：
   - 使用现有 KMS 密钥 - 从列表中选择您的 KMS 密钥别名。
   - 要创建新的 KMS 密钥，请选择创建 Amazon KMS 密钥。

> **Note**
>
> 这将打开 Amazon KMS 控制面板。有关创建 KMS 密钥的更多信息，请参阅Amazon Key Management Service 开发人员指南中的创建密钥。

5. 选择保存以保存您的设置。

SiteWise 边缘网关密钥管理

SiteWise 边缘网关在上运行 Amazon IoT Greengrass， Amazon IoT Greengrass 核心设备使用公钥和私钥向 Amazon 云端进行身份验证并加密本地密钥，例如 OPC UA 身份验证密钥。有关更多信息，请参阅Amazon IoT Greengrass Version 1 开发人员指南中的密钥管理。

的身份和访问管理 Amazon IoT SiteWise

Amazon Identity and Access Management (IAM) Amazon Web Services 服务 可帮助管理员安全地控制对 Amazon 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 Amazon IoT SiteWise 资源。您可以使用 IAM Amazon Web Services 服务 ，无需支付额外费用。

主题

- Amazon IoT SiteWise 安全受众

- 使用中的身份进行身份验证 Amazon IoT SiteWise
- 如何 Amazon IoT SiteWise 与 IAM 配合使用
- Amazon 的托管策略 Amazon IoT SiteWise
- 将服务相关角色用于 Amazon IoT SiteWise
- 在中设置事件警报的权限 Amazon IoT SiteWise
- 跨服务混淆了副手预防 Amazon IoT SiteWise
- 对 Amazon IoT SiteWise 身份和访问进行故障排除

Amazon IoT SiteWise 安全受众

您的使用方式 Amazon Identity and Access Management (IAM) 会有所不同，具体取决于您所做的工作 Amazon IoT SiteWise。

服务用户-如果您使用 Amazon IoT SiteWise 服务完成工作，则管理员会为您提供所需的凭证和权限。当你使用更多 Amazon IoT SiteWise 功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon IoT SiteWise中的特征，请参阅 对 Amazon IoT SiteWise 身份和访问进行故障排除。

服务管理员-如果您负责公司的 Amazon IoT SiteWise 资源，则可能拥有完全访问权限 Amazon IoT SiteWise。您的工作是确定您的服务用户应访问哪些 Amazon IoT SiteWise 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解您的公司如何将 IAM 与配合使用 Amazon IoT SiteWise，请参阅如何 Amazon IoT SiteWise 与 IAM 配合使用。

IAM 管理员：如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 Amazon IoT SiteWise的访问权限的详细信息。要查看您可以在 IAM 中使用的 Amazon IoT SiteWise 基于身份的策略示例，请参阅。Amazon IoT SiteWise 基于身份的策略示例

使用中的身份进行身份验证 Amazon IoT SiteWise

身份验证是您 Amazon 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 Amazon Web Services 账户根用户任 IAM 角色进行身份验证（登录 Amazon）。

如果您 Amazon 以编程方式访问，则会 Amazon 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 Amazon 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的用于签署 API 请求的Amazon 签名版本 4。

无论使用何种身份验证方法，您都可能需要提供其他安全信息。例如， Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的 IAM 中的Amazon 多重身份验证。

Amazon Web Services 账户 root 用户

创建时 Amazon Web Services 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 Amazon Web Services 服务 和资源。此身份被称为 Amazon Web Services 账户 root 用户，使用您创建帐户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

IAM 用户和群组

I AM 用户是您 Amazon Web Services 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的用例，应在需要时更新访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并向该群组授予管理 IAM 资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的 IAM 用户的使用案例。

IAM 角色

I AM 角色是您内部具有特定权限 Amazon Web Services 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。要在中临时担任 IAM 角色 Amazon Web Services Management Console，您可以从用户切换到 IAM 角色（控制台）。您可以通过调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的代入角色的方法。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问：要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM 用户指南》中的针对第三方身份提供商创建角色（联合身份验证）。

- 临时 IAM 用户权限：IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取：您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 Amazon Web Services 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南中的 IAM 中的跨账户资源访问。
- 跨服务访问 — 有些 Amazon Web Services 服务 使用其他 Amazon Web Services 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序 EC2 或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 Amazon，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限 Amazon Web Services 服务，再加上 Amazon Web Services 服务 向下游服务发出请求的请求。只有当服务收到需要与其他 Amazon Web Services 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务委派权限的角色。
  - 服务相关角色-服务相关角色是一种与服务相关联的服务角色。 Amazon Web Services 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 Amazon Web Services 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 A@@ mazon 上运行的应用程序 EC2 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时证书。这比在 EC2 实例中存储访问密钥更可取。要为 EC2 实例分配 Amazon 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含该角色，并允许在 EC2 实例上运行的程序获得临时证书。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色向在 A mazon EC2 实例上运行的应用程序授予权限。

如何 Amazon IoT SiteWise 与 IAM 配合使用

在使用 Amazon Identity and Access Management (IAM) 管理访问权限之前 Amazon IoT SiteWise，您应该了解哪些可用的 IAM 功能 Amazon IoT SiteWise。

IAM 特征	支持 Amazon IoT SiteWis
具有资源级权限的基于身份的策略	是
策略操作	是
策略资源	是
策略条件键	是
基于资源的策略	否
访问控制列表 (ACLs)	否
基于标签的授权（ABAC）	是
临时凭证	是
转发访问会话（FAS）	是
服务相关角色	是
服务角色	是

要全面了解如何 Amazon IoT SiteWise 和其他 Amazon 服务与 IAM 配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的Amazon 服务。

目录

- Amazon IoT SiteWise IAM 角色
  - 将临时证书用于 Amazon IoT SiteWise
  - 的转发访问会话 (FAS) Amazon IoT SiteWise
  - 服务相关角色
  - 服务角色
  - 在 Amazon IoT SiteWise中选择 IAM 角色

- 基于 Amazon IoT SiteWise 标签的授权
- Amazon IoT SiteWise 基于身份的策略
  - 策略操作
    - BatchPutAssetPropertyValue 授权
  - 策略资源
  - 策略条件键
  - 示例
- Amazon IoT SiteWise 基于身份的策略示例
  - 策略最佳实践
  - 使用 Amazon IoT SiteWise 控制台
  - 允许用户查看他们自己的权限
  - 支持用户将数据摄取到一个层次结构中的资产
  - 根据标签查看 Amazon IoT SiteWise 资产
- 使用中的策略管理访问权限 Amazon IoT SiteWise
  - 基于身份的策略
  - 基于资源的策略
  - 访问控制列表 (ACLs)
  - 其他策略类型
  - 多个策略类型

Amazon IoT SiteWise IAM 角色

IAM 角色是 Amazon 账户中具有特定权限的实体。

将临时证书用于 Amazon IoT SiteWise

可以使用临时凭证进行联合身份验证登录，分派 IAM 角色或分派跨账户角色。您可以调用 Amazon STS API 操作（如AssumeRole 或 GetFederationToken ）以获取临时安全凭证。

Amazon IoT SiteWise 支持使用临时证书。

SiteWise Monitor 支持联合用户访问门户。门户用户使用其 IAM 凭证进行身份验证。

> **Important**
>
> 用户或角色必须具有登录门户的 `iotsitewise:DescribePortal` 权限。

当用户登录门户时，M SiteWise onitor 会生成一个提供以下权限的会话策略：

- 对您账户 Amazon IoT SiteWise 中的资产和资产数据的只读访问权限，该门户的角色为其提供访问权限。
- 对该门户中的项目（用户对这些项目具有管理员（项目拥有者）或只读（项目查看者）访问权限）的访问权限。

有关联合身份门户用户权限的更多信息，请参阅将服务角色用于 Amazon IoT SiteWise Monitor。

的转发访问会话 (FAS) Amazon IoT SiteWise

支持转发访问会话（FAS）：是

当您使用 IAM 用户或角色在中执行操作时 Amazon，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限 Amazon Web Services 服务，再加上 Amazon Web Services 服务 向下游服务发出请求的请求。只有当服务收到需要与其他 Amazon Web Services 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

服务相关角色

服务关联角色允许 Amazon 服务访问其他服务中的资源以代表您完成操作。服务相关角色出现在您的 Amazon 账户中，归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Amazon IoT SiteWise 支持与服务相关的角色。有关创建或管理 Amazon IoT SiteWise 服务相关角色的详细信息，请参阅 将服务相关角色用于 Amazon IoT SiteWise。

服务角色

此功能允许服务代表您担任服务角色。此角色允许服务访问其他服务中的资源以代表您完成操作。服务角色显示在您的 Amazon 账户中，并归该账户所有。这意味着，IAM 管理员可以更改该角色的权限。但是，这样做可能会中断服务的功能。

Amazon IoT SiteWise 使用服务角色允许 M SiteWise onitor 门户用户代表您访问您的某些 Amazon IoT SiteWise 资源。有关更多信息，请参阅 将服务角色用于 Amazon IoT SiteWise Monitor。

在中创建 Amazon IoT Events 警报模型之前，您必须拥有所需的权限 Amazon IoT SiteWise。有关更多信息，请参阅 在中设置事件警报的权限 Amazon IoT SiteWise。

在 Amazon IoT SiteWise中选择 IAM 角色

在中创建`portal`资源时 Amazon IoT SiteWise，必须选择一个角色才能允许 Monito SiteWise r 门户的联合用户 Amazon IoT SiteWise 代表您进行访问。如果您之前创建过服务角色，则会 Amazon IoT SiteWise 为您提供可供选择的角色列表。否则，可以在创建门户时创建具有所需权限的角色。选择一个允许访问资产和资产数据的角色非常重要。有关更多信息，请参阅 将服务角色用于 Amazon IoT SiteWise Monitor。

基于 Amazon IoT SiteWise 标签的授权

您可以为 Amazon IoT SiteWise 资源附加标签或在请求中传递标签 Amazon IoT SiteWise。要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。有关标记 Amazon IoT SiteWise 资源的更多信息，请参阅标记您的 Amazon IoT SiteWise 资源。

要查看基于身份的策略（用于根据资源上的标签来限制对该资源的访问）的示例，请参阅根据标签查看 Amazon IoT SiteWise 资产。

Amazon IoT SiteWise 基于身份的策略

IAM 策略支持您在 Amazon IoT SiteWise中控制哪些人可以执行哪些操作。您可以决定允许或不允许哪些操作，并为这些操作设置具体条件。例如，您可以制定规则，规定谁可以查看或更改中的信息 Amazon IoT SiteWise。 Amazon IoT SiteWise 支持特定的操作、资源和条件键。要了解在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素参考。

策略操作

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

正在执行的策略操作在操作前 Amazon IoT SiteWise 使用以下前缀:`iotsitewise:`. 例如，要授予某人`BatchPutAssetPropertyValue`通过 API 操作将资产属性数据上传到 Amazon IoT SiteWise 的权

限，您可以将该`iotsitewise:BatchPutAssetPropertyValue`操作包含在他们的策略中。策略声明必须包含`Action`或`NotAction`元素。 Amazon IoT SiteWise 定义了它自己的一组操作，这些操作描述了您可以使用此服务执行的任务。

要在单个语句中指定多项 操作，请使用逗号将它们隔开，如下所示。

```
"Action": [
  "iotsitewise:action1",
  "iotsitewise:action2"
]
```

您也可以使用通配符 (*) 指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，请包括以下操作。

```
"Action": "iotsitewise:Describe*"
```

要查看 Amazon IoT SiteWise 操作列表，请参阅 IAM 用户指南 Amazon IoT SiteWise中定义的操作。

BatchPutAssetPropertyValue 授权

Amazon IoT SiteWise 以一种不寻常的方式授权访问BatchPutAssetPropertyValue操作。对于大多数操作而言，当您允许或拒绝访问时，如果未授予权限，则该操作将返回错误。通过`BatchPutAssetPropertyValue`，您可以在单个 API 请求中将多个数据条目发送到不同的资产和资产属性。 Amazon IoT SiteWise 会单独向每个数据条目授权。对于请求中未通过授权的任何单个条目， Amazon IoT SiteWise 请在返回的错误列表`AccessDeniedException`中包含一个。 Amazon IoT SiteWise 接收任何授权并成功的条目的数据，即使同一请求中的另一个条目失败也是如此。

> **Important**
>
> 在将数据摄取到数据流之前，请执行以下操作：
>
> - 如果您使用属性别名来标识数据流，请对 `time-series` 资源进行授权。
> - 如果您使用资产 ID 来识别包含关联资产属性的资产，请对 `asset` 资源进行授权。

策略资源

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

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

每个 IAM 政策声明都适用于您使用其指定的资源 ARNs。ARN 具有以下一般语法。

```
arn:${Partition}:${Service}:${Region}:${Account}:${ResourceType}/${ResourcePath}
```

有关格式的更多信息 ARNs，请参阅使用 Amazon 资源名称识别 Amazon 资源 (ARNs)。

例如，要在语句中指定 ID 为 a1b2c3d4-5678-90ab-cdef-22222EXAMPLE 的资产，请使用以下 ARN。

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE"
```

要指定属于特定账户的所有数据流，请使用通配符 (*)：

```
"Resource": "arn:aws:iotsitewise:region:123456789012:time-series/*"
```

要指定属于特定账户的所有资产，请使用通配符 (*)：

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/*"
```

某些 Amazon IoT SiteWise 操作（例如创建资源的操作）无法对特定资源执行。在这些情况下，您必须使用通配符 （*)。

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

要在单个语句中指定多个资源，请 ARNs 用逗号分隔。

```
"Resource": [
```

```
  "resource1",
  "resource2"
]
```

要查看 Amazon IoT SiteWise 资源类型及其列表 ARNs，请参阅 IAM 用户指南 Amazon IoT SiteWise中定义的资源类型。要了解可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon IoT SiteWise定义的操作。

策略条件键

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 Amazon 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 Amazon 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

Amazon 支持全局条件密钥和特定于服务的条件密钥。要查看所有 Amazon 全局条件键，请参阅 IAM 用户指南中的Amazon 全局条件上下文密钥。

> **Important**
>
> 许多条件键是特定于某个资源的，而某些 API 操作会使用多个资源。如果您使用条件键编写策略语句，请使用该语句的 `Resource` 元素指定要将该条件键应用于的资源。如果您没有这样做，该策略可能会完全阻止用户执行操作，因为针对未应用条件键的资源的条件检查失败。如果您不想指定资源，或者如果您已将策略的 `Action` 元素编写为包含多个 API 操作，则必须使用 `...IfExists` 条件类型以确保对不使用条件键的资源忽略条件键。有关更多信息，请参阅... IfExists IAM 用户指南中的条件。

Amazon IoT SiteWise 定义自己的条件键集，还支持使用一些全局条件键。要查看所有 Amazon 全局条件键，请参阅 IAM 用户指南中的Amazon 全局条件上下文密钥。

Amazon IoT SiteWise 条件键

条件键	描述	类型
iotsitewise:isAssociatedWithAssetProperty	数据流是否与资产属性相关联。使用此条件键根据数据流的关联资产属性的存在来定义权限。 示例值：true	字符串
iotsitewise:assetHierarchyPath	资产的层次结构路径，这是一串由正斜杠分 IDs 隔的资产。使用此条件键可根据您账户中所有资产的层次结构子集来定义权限。 示例值：/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/a1b2c3d4-5678-90ab-cdef-66666EXAMPLE	字符串
iotsitewise:propertyId	资产属性的 ID。使用此条件键可根据指定的资产模型属性来定义权限。此条件键适用于该模型的所有资产。 示例值：a1b2c3d4-5678-90ab-cdef-33333EXAMPLE	字符串
iotsitewise:childAssetId	作为子级关联到其他资产的资产 ID。使用此条件键可根据子资产来定义权限。要根据父资产定义权限，请使用策略语句的资源部分。	字符串

条件键	描述	类型
	示例值：a1b2c3d4-5678-90ab-cdef-66666EXAMPLE	
iotsitewise:iam	列出访问策略时 IAM 身份的 ARN。使用此条件键可定义 IAM 身份的访问策略权限。 示例值：arn:aws:iam::123456789012:user/JohnDoe	字符串，Null
iotsitewise:propertyAlias	标识资产属性或数据流的别名。使用此条件键可根据别名来定义权限。	字符串
iotsitewise:user	列出访问策略时的 IAM Identity Center 用户 ID。使用此条件键可定义 IAM Identity Center 用户的访问策略权限。 示例值：a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE	字符串，Null
iotsitewise:group	列出访问策略时的 IAM Identity Center 组 ID。使用此条件键可定义 IAM Identity Center 组的访问策略权限。 示例值：a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE	字符串，Null

条件键	描述	类型
iotsitewise:portal	访问策略中的门户 ID。使用此条件键可根据门户来定义访问策略权限。 示例值：a1b2c3d4-5678-90ab-cdef-77777EXAMPLE	字符串，Null
iotsitewise:project	访问策略中的项目 ID 或控制面板的项目 ID。使用此条件键可根据项目来定义控制面板或访问策略权限。 示例值：a1b2c3d4-5678-90ab-cdef-88888EXAMPLE	字符串，Null

要了解可以使用条件键的操作和资源，请参阅由定义的操作 Amazon IoT SiteWise。

示例

要查看 Amazon IoT SiteWise 基于身份的策略的示例，请参阅。Amazon IoT SiteWise 基于身份的策略示例

Amazon IoT SiteWise 基于身份的策略示例

默认情况下，实体（用户和角色）没有创建或修改 Amazon IoT SiteWise 资源的权限。他们也无法使用 Amazon Web Services Management Console、 Amazon Command Line Interface (Amazon CLI) 或 Amazon API 执行任务。要调整权限， Amazon Identity and Access Management (IAM) 管理员必须执行以下操作：

1. 创建 IAM 策略，以便为用户和角色授予权限来对所需资源执行特定的 API 操作。
2. 将这些策略附加到需要这些权限的用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅《IAM 用户指南》中的在 JSON 选项卡上创建策略。

主题

- 策略最佳实践
- 使用 Amazon IoT SiteWise 控制台
- 允许用户查看他们自己的权限
- 支持用户将数据摄取到一个层次结构中的资产
- 根据标签查看 Amazon IoT SiteWise 资产

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 Amazon IoT SiteWise 资源。这些操作可能会使 Amazon Web Services 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 Amazon 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的Amazon 托管策略。它们在你的版本中可用 Amazon Web Services 账户。我们建议您通过定义针对您的用例的 Amazon 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管式策略或工作职能的Amazon 托管式策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 Amazon Web Services 服务，例如 Amazon CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 Amazon Web Services 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Amazon IoT SiteWise 控制台

要访问 Amazon IoT SiteWise 控制台，您需要一组基本权限。这些权限允许您查看和管理有关您 Amazon 账户中 Amazon IoT SiteWise 资源的详细信息。

如果您制定的策略过于严格，那么对于使用该策略的用户或角色（实体），控制台可能无法按预期运行。为确保这些实体仍然可以使用 Amazon IoT SiteWise 控制台，请将AWSIoTSiteWiseConsoleFullAccess托管策略附加到它们或为这些实体定义等效权限。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

如果实体仅使用 Amazon Command Line Interface (CLI) 或 Amazon IoT SiteWise API，而不使用控制台，则不需要这些最低权限。在这种情况下，只需让他们能够访问其 API 任务所需的特定操作即可。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 Amazon CLI 或 Amazon API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
```

```
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

支持用户将数据摄取到一个层次结构中的资产

在此示例中，您希望授予 Amazon 账户中的用户从根资产开始向特定资产层次结构中的所有资产属性写入数据的权限`a1b2c3d4-5678-90ab-cdef-22222EXAMPLE`。此策略向用户授予 `iotsitewise:BatchPutAssetPropertyValue` 权限。此策略使用 `iotsitewise:assetHierarchyPath` 条件键来限制对其层次结构路径与资产或其后代匹配的资产的访问。

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

根据标签查看 Amazon IoT SiteWise 资产

使用基于身份的策略中的条件根据标签控制对 Amazon IoT SiteWise 资源的访问权限。此示例说明了如何创建支持查看资产的策略。不过，仅在资产标签 `Owner` 具有该用户的用户名值时授予权限。该策略还会授予用于在控制台上完成该操作的权限。

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

将此策略附加到您账户中的用户。如果名为的用户`richard-roe`尝试查看 Amazon IoT SiteWise 资产，则必须对该资产进行标记`Owner=richard-roe`或`owner=richard-roe`。否则，Richard 将被拒绝访问。条件标签密钥名称不区分大小写。因此，`Owner` 与 `Owner` 和 `owner` 均匹配。有关更多信息，请参阅 IAM 用户指南 中的 IAM JSON 策略元素：条件。

使用中的策略管理访问权限 Amazon IoT SiteWise

您可以 Amazon 通过创建策略并将其附加到 Amazon 身份或资源来控制中的访问权限。策略是其中的一个对象 Amazon ，当与身份或资源关联时，它会定义其权限。 Amazon 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以

JSON 文档的 Amazon 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 Amazon Web Services Management Console Amazon CLI、或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户托管策略定义自定义 IAM 权限。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 Amazon Web Services 账户。托管策略包括 Amazon 托管策略和客户托管策略。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 Amazon Web Services 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 Amazon 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持的服务示例 ACLs。 Amazon WAF要了解更多信息 ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

Amazon 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)- SCPs 是指定组织或组织单位 (OU) 的最大权限的 JSON 策略 Amazon Organizations。 Amazon Organizations 是一项用于对您的企业拥有的多 Amazon Web Services 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。SCP 限制成员账户中的实体（包括每个 Amazon Web Services 账户根用户实体）的权限。有关 Organization SCPs s 和的更多信息，请参阅《Amazon Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs) — RCPs 是 JSON 策略，您可以使用它来设置账户中资源的最大可用权限，而无需更新附加到您拥有的每个资源的 IAM 策略。RCP 限制成员账户中资源的权限，并可能影响身份（包括身份）的有效权限 Amazon Web Services 账户根用户，无论这些身份是否属于您的组织。有关 Organizations 的更多信息 RCPs，包括 Amazon Web Services 服务 该支持的列表 RCPs，请参阅《Amazon Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略：会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 Amazon 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon 的托管策略 Amazon IoT SiteWise

使用 Amazon 托管策略简化向用户、群组和角色添加权限的过程，而不是自己编写策略。创建为团队提供确切权限的 IAM 客户管理型策略需要时间和专业知识。为了更快地进行设置，可以考虑将我们的 Amazon 托管策略用于常见用例。在您的 Amazon 账户中查找 Amazon 托管政策。有关 Amazon 托管式策略的更多信息，请参阅《IAM 用户指南》中的 Amazon 托管式策略。

Amazon 服务负责更新和维护 Amazon 托管策略，这意味着您无法修改这些策略的权限。有时，Amazon IoT SiteWise 可能会添加权限以适应新功能，从而影响附加策略的所有身份。新服务或特征的推出往往伴随着此类更新。但是，权限永远不会被移除，这就可以确保您的设置保持不变。

此外，还 Amazon 支持跨多个服务的工作职能的托管策略。例如，ReadOnlyAccess Amazon 托管策略提供对所有 Amazon 服务和资源的只读访问权限。当服务启动一项新功能时， Amazon 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的 Amazon 托管策略。

Amazon 托管策略： AWSIoTSiteWiseReadOnlyAccess

使用`AWSIoTSiteWiseReadOnlyAccess` Amazon 托管策略允许只读访问权限 Amazon IoT SiteWise。

您可以将 `AWSIoTSiteWiseReadOnlyAccess` 策略附加到 IAM 身份。

服务级别权限

此策略提供对的只读访问权限 Amazon IoT SiteWise。此策略中不包含其他服务权限。

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

Amazon 托管策略： AWSServiceRoleForIoTSiteWise

`AWSServiceRoleForIoTSiteWise` 角色获得下列权限使用 `AWSServiceRoleForIoTSiteWise` 策略。本策略：

- Amazon IoT SiteWise 允许部署 SiteWise Edge 网关（在上运行`Amazon IoT Greengrass`）。

- Amazon IoT SiteWise 允许执行日志记录。
- Amazon IoT SiteWise 允许对 Amazon IoT TwinMaker 数据库运行元数据搜索查询。

如果您 Amazon IoT SiteWise 使用的是单个用户账户，则该`AWSServiceRoleForIoTSiteWise`角色将在您的 IAM 账户中创建`AWSServiceRoleForIoTSiteWise`策略，并将其附加到的`AWSServiceRoleForIoTSiteWise`服务相关角色。 Amazon 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-cn:logs:*:*:log-group:/aws/iotsitewise*"
  },
  {
   "Sid": "AllowSiteWiseAccessLog",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:DescribeLogStreams",
    "logs:PutLogEvents"
   ],
   "Resource": "arn:aws-cn:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
  },
```

```
  {
   "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
   "Effect": "Allow",
   "Action": [
    "iottwinmaker:GetWorkspace",
    "iottwinmaker:ExecuteQuery"
   ],
   "Resource": "arn:aws-cn:iottwinmaker:*:*:workspace/*",
   "Condition": {
    "ForAnyValue:StringEquals": {
     "iottwinmaker:linkedServices": [
      "IOTSITEWISE"
     ]
    }
   }
  }
 ]
}
```

Amazon IoT SiteWiseAmazon 托管策略的更新

您可以查看有关 Amazon 托管策略更新的详细信息 Amazon IoT SiteWise，从该服务开始跟踪更改时开始。要获得有关此页面变更的自动提醒，请订阅 “ Amazon IoT SiteWise 文档历史记录” 页面上的 RSS feed。

更改	描述	日期
AWSServiceRoleForIoTSiteWise — 更新现有政策	Amazon IoT SiteWise 现在可以对 Amazon IoT TwinMaker 数据库运行元数据搜索查询。	2023 年 11 月 6 日
AWSIoTSiteWiseReadOnlyAccess – 对现有策略的更新	Amazon IoT SiteWise 添加了新的策略前缀`BatchGet*` ，使您可以执行批量读取操作。	2022 年 9 月 16 日
AWSIoTSiteWiseReadOnlyAccess：新策略	Amazon IoT SiteWise 添加了授予只读访问权限的新策略 Amazon IoT SiteWise。	2021 年 11 月 24 日

更改	描述	日期
Amazon IoT SiteWise 开始跟踪更改	Amazon IoT SiteWise 开始跟踪其 Amazon 托管策略的更改。	2021 年 11 月 24 日

将服务相关角色用于 Amazon IoT SiteWise

Amazon IoT SiteWise 使用 Amazon Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种直接关联的 IAM 角色的独特类型 Amazon IoT SiteWise。服务相关角色由服务预定义 Amazon IoT SiteWise ，包括服务代表您调用其他 Amazon 服务所需的所有权限。

服务相关角色 Amazon IoT SiteWise 通过自动包含所有必要的权限来简化配置。 Amazon IoT SiteWise 定义其服务相关角色的权限，除非另有定义，否则 Amazon IoT SiteWise 只能担任其角色。定义的权限包括信任策略和权限策略。并且，相应权限策略不能附加到任何其他 IAM 实体。

只有在首先删除相关资源后，您才能删除服务相关角色。这样可以保护您的 Amazon IoT SiteWise 资源，因为您不会无意中删除访问资源的权限。

有关支持服务相关角色的其他服务的信息，请参阅与 IAM 配合使用的Amazon 服务，并在服务关联角色列中查找标有 “是” 的服务。选择是和链接，查看该服务的服务相关角色文档。

主题

- Amazon IoT SiteWise的服务相关角色权限
- 为 Amazon IoT SiteWise创建服务相关角色
- 为 Amazon IoT SiteWise更新服务相关角色
- 删除 Amazon IoT SiteWise的服务相关角色
- Amazon IoT SiteWise 服务相关角色支持的区域
- 将服务角色用于 Amazon IoT SiteWise Monitor

Amazon IoT SiteWise的服务相关角色权限

Amazon IoT SiteWise 使用名为 AWSServiceRoleForIoTSiteWise 的服务相关角色。 Amazon IoT SiteWise 使用此服务相关角色部署 SiteWise Edge 网关（在上运行 Amazon IoT Greengrass）和执行日志记录。

`AWSServiceRoleForIoTSiteWise` 服务相关角色使用具有以下权限的 `AWSServiceRoleForIoTSiteWise` 策略。本策略：

- Amazon IoT SiteWise 允许部署 SiteWise Edge 网关（在上运行`Amazon IoT Greengrass`）。
- Amazon IoT SiteWise 允许执行日志记录。
- Amazon IoT SiteWise 允许对 Amazon IoT TwinMaker 数据库运行元数据搜索查询。

有关 `AWSServiceRoleForIoTSiteWise` 中允许执行的操作的更多信息，请参阅 Amazon managed policies for Amazon 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-cn:logs:*:*:log-group:/aws/iotsitewise*"
  },
  {
   "Sid": "AllowSiteWiseAccessLog",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:DescribeLogStreams",
```

```
      "logs:PutLogEvents"
     ],
     "Resource": "arn:aws-cn:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
    },
    {
     "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
     "Effect": "Allow",
     "Action": [
      "iottwinmaker:GetWorkspace",
      "iottwinmaker:ExecuteQuery"
     ],
     "Resource": "arn:aws-cn:iottwinmaker:*:*:workspace/*",
     "Condition": {
      "ForAnyValue:StringEquals": {
       "iottwinmaker:linkedServices": [
        "IOTSITEWISE"
       ]
      }
     }
    }
   ]
}
```

您可以使用日志来监控 SiteWise Edge 网关并对其进行故障排除。有关更多信息，请参阅 监控 SiteWise 边缘网关日志。

为允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色，首先配置权限。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为 Amazon IoT SiteWise创建服务相关角色

Amazon IoT SiteWise 需要服务相关角色才能代表您执行某些操作和访问资源。服务相关角色是一种与其直接关联的独特 Amazon 身份和访问管理 (IAM) 角色。 Amazon IoT SiteWise创建此角色 Amazon IoT SiteWise 即表示您授予访问其操作所需的其他 Amazon 服务和资源的必要权限，例如用于数据存储或 Amazon IoT 设备通信的 Amazon S3。

您无需手动创建服务相关角色。当您在 Amazon IoT SiteWise 控制台中执行以下操作时， Amazon IoT SiteWise 会为您创建服务相关角色。

- 创建 Greengrass V1 网关。
- 配置日志记录选项。
- 在执行查询横幅中选择“选择加入”按钮。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您在 Amazon IoT SiteWise 控制台中执行任何操作时， Amazon IoT SiteWise 会再次为您创建服务相关角色。

您也可以使用 IAM 控制台或 API 为 Amazon IoT SiteWise创建服务相关角色。

- 要在 IAM 控制台中执行此操作，请使用 AWSServiceRoleForIoTSiteWise 策略创建一个角色并与建立信任关系`iotsitewise.amazonaws.com`。
- 要使用 Amazon CLI 或 IAM API 执行此操作，请使用`arn:aws:iam::aws:policy/aws-service-role/AWSServiceRoleForIoTSiteWise`策略创建一个角色并与建立信任关系`iotsitewise.amazonaws.com`。

有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。

如果您删除了此服务相关角色，可以使用同样的过程再次创建角色。

为 Amazon IoT SiteWise更新服务相关角色

Amazon IoT SiteWise 不允许您编辑 AWSService RoleForIo TSite Wise 服务相关角色。在创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。不过，您可以使用 IAM 编辑角色的说明。有关更多信息，请参阅《IAM 用户指南》中的更新服务相关角色。

删除 Amazon IoT SiteWise的服务相关角色

如果不再使用某个需要服务相关角色的特征或服务，建议删除关联的角色。这是为了避免出现未被监控或维护的非活动实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。

> **Note**
>
> 如果您尝试删除资源时 Amazon IoT SiteWise 服务正在使用该角色，则删除可能会失败。如果发生这种情况，请等待几分钟，然后重试。

删除 AWSServiceRoleForIoTSite智者使用的 Amazon IoT SiteWise 资源

1. 禁用登录功能 Amazon IoT SiteWise。有关更多信息，请参阅 更改日志记录级别。
2. 删除所有活动的 SiteWise Edge 网关。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台 Amazon CLI、或 Amazon API 删除 AWSService RoleForIo TSite Wise 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的删除角色或实例配置文件。

Amazon IoT SiteWise 服务相关角色支持的区域

Amazon IoT SiteWise 支持在提供服务的所有区域中使用服务相关角色。有关更多信息，请参阅 Amazon IoT SiteWise 终端节点和配额。

将服务角色用于 Amazon IoT SiteWise Monitor

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务委派权限的角色。

要允许联合的 M SiteWise onitor 门户用户访问您的 Amazon IoT SiteWise资源，您必须将服务角色附加到您创建的每个门户。服务角色必须将 M SiteWise onitor 指定为可信实体，并包含AWSIoTSiteWiseMonitorPortalAccess托管策略或定义等效权限。此策略由 M SiteWise onitor 维护 Amazon 并定义了用于访问您 Amazon IoT SiteWise 资源的一组权限。

创建 M SiteWise onitor 门户时，必须选择允许该门户的用户访问您 Amazon IoT SiteWise 资源的角色。 Amazon IoT SiteWise 控制台可以为您创建和配置角色。您可以稍后在 IAM 中编辑该角色。如果您从角色中移除所需的权限或删除该角色，您的门户用户在使用其 SiteWise 监控器门户时会遇到问题。

> **Note**
>
> 2020 年 4 月 29 日之前创建的门户不需要服务角色。如果您是在此日期之后创建的门户，则必须附加服务角色才能继续使用。为此，请导航到 Amazon IoT SiteWise 控制台的门户页面，再选择迁移所有门户以使用 IAM 角色。

以下各节介绍如何在 Amazon Web Services Management Console 或中创建和管理 M SiteWise onitor 服务角色 Amazon Command Line Interface。

目录

- SiteWise 监控器（经典）的服务角色权限
- SiteWise 监控器的服务角色权限（AI 感知）
- 管理 SiteWise 监控服务角色（控制台）
  - 查找门户的服务角色（控制台）

- 创建 SiteWise 监控服务角色（Amazon IoT SiteWise 控制台）
- 创建 SiteWise 监控服务角色（IAM 控制台）
- 更改门户的服务角色（控制台）
- 管理 SiteWise 监控服务角色 (CLI)
  - 查找门户的服务角色（CLI）
  - 创建 SiteWise 监控服务角色 (CLI)
- SiteWise 监视更新 AWSIo TSite WiseMonitorServiceRole

SiteWise 监控器（经典）的服务角色权限

创建门户时， Amazon IoT SiteWise 允许您创建名称以开头的角色AWSIoTSiteWiseMonitorServiceRole。此角色允许联合 SiteWise 监视器用户访问您的门户配置、资产、和资产数据。

该角色信任以下服务来代入角色：

- `monitor.iotsitewise.amazonaws.com`

该角色使用以下权限策略（开头为）AWSIoTSiteWiseMonitorServicePortalPolicy，以允许 M SiteWise onitor 用户对您账户中的资源完成操作。AWSIoTSiteWiseMonitorPortalAccess 托管策略定义等效权限。

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

有关警报所需权限的更多信息，请参阅 在中设置事件警报的权限 Amazon IoT SiteWise。

当门户用户登录时，M SiteWise onitor 会根据服务角色与该用户的访问策略的交叉点创建会话策略。访问策略定义 身份对门户和项目的访问级别。有关门户权限和访问策略的更多信息，请参阅管理您的 SiteWise 监控器门户和CreateAccessPolicy。

SiteWise 监控器的服务角色权限（AI 感知）

创建门户时， Amazon IoT SiteWise 允许您创建一个名称以 Io 开头的角色TSiteWisePortalRole。此角色允许联合 SiteWise 监视器用户访问您的门户配置、资产、和资产数据。

> **Warning**
>
> SiteWise 监视器（AI 感知）不支持@@ 项目所有者和项目查看者角色。

该角色信任以下服务来代入角色：

- `monitor.iotsitewise.amazonaws.com`

该角色使用以下以 Io TSite Wise 开头的权限策略 AIPortalAccessPolicy，以允许 M SiteWise onitor 用户完成对您账户中的资源的操作。

当门户用户登录时，M SiteWise onitor 会根据服务角色与该用户的访问策略的交叉点创建会话策略。

管理 SiteWise 监控服务角色（控制台）

Amazon IoT SiteWise 控制台 便于管理门户的 SiteWise 监控服务角色。创建门户后，控制台会检查是否存在适合附加到该门户的现有角色。如果没有可用的角色，则控制台可以为您创建和配置服务角色。有关更多信息，请参阅 在 Monito SiteWise r 中创建门户。

主题

- 查找门户的服务角色（控制台）
- 创建 SiteWise 监控服务角色（Amazon IoT SiteWise 控制台）
- 创建 SiteWise 监控服务角色（IAM 控制台）
- 更改门户的服务角色（控制台）

查找门户的服务角色（控制台）

使用以下步骤查找附加到 SiteWise 监控器门户的服务角色。

查找门户的服务角色

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择 门户。
3. 选择要查找其服务角色的门户。

   附加到门户的角色显示在权限、服务角色 下。

创建 SiteWise 监控服务角色（Amazon IoT SiteWise 控制台）

创建 M SiteWise onitor 门户时，可以为门户创建服务角色。有关更多信息，请参阅 在 Monito SiteWise r 中创建门户。

您也可以在 Amazon IoT SiteWise 控制台中为现有门户创建服务角色。这将取代门户的现有服务角色。

为现有门户创建服务角色

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 门户。

3. 选择要为其创建新服务角色的门户。
4. 在门户详细信息下，选择编辑。
5. 在权限下，从列表中选择创建和使用新的服务角色。
6. 输入新角色的名称。
7. 选择保存。

创建 SiteWise 监控服务角色（IAM 控制台）

您可以在 IAM 控制台中从服务角色模板创建服务角色。此角色模板包括AWSIoTSiteWiseMonitorPortalAccess托管策略，并将 Monito SiteWise r 指定为可信实体。

通过门户服务角色模板创建服务角色

1. 导航到 IAM 控制台。
2. 在导航窗格中，选择 角色。
3. 选择创建角色。
4. 在选择用例中，选择物联网 SiteWise。
5. 在选择您的使用案例中，选择 IoT SiteWise Monitor - Portal。
6. 选择下一步: 权限。
7. 选择下一步: 标签。
8. 选择下一步：审核。
9. 在角色名称中，输入新服务角色名称。
10. 选择创建角色。

更改门户的服务角色（控制台）

使用以下步骤为门户选择不同的 SiteWise 监控服务角色。

更改门户的服务角色

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 门户。
3. 选择要更改其服务角色的门户。
4. 在门户详细信息下，选择编辑。

5. 在权限下，选择使用现有角色。
6. 选择要附加到此门户的现有角色。
7. 选择保存。

管理 SiteWise 监控服务角色 (CLI)

您可以将 Amazon CLI 用于以下门户服务角色管理任务：

主题

- 查找门户的服务角色（CLI）
- 创建 SiteWise 监控服务角色 (CLI)

查找门户的服务角色（CLI）

要查找附加到 SiteWise 监控器门户的服务角色，请运行以下命令列出当前区域中的所有门户。

```
aws iotsitewise list-portals
```

此操作将按照以下格式返回包含门户摘要的响应。

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

如果您知道门户的 ID，也可以使用该DescribePortal操作来查找门户的角色。

创建 SiteWise 监控服务角色 (CLI)

使用以下步骤创建新的 SiteWise 监控器服务角色。

创建 SiteWise 监控服务角色

1. 使用允许 Monito SiteWise r 担任该角色的信任策略创建角色。此示例根据存储在 JSON 字符串中的信任策略创建名为 **`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. 从输出中的角色元数据复制角色 ARN。创建门户时，您可以使用此 ARN 将角色与门户关联。有关创建门户的更多信息，请参阅 Amazon IoT SiteWise API 参考CreatePortal中的。

3. a. 对于 SiteWise 监控器（经典）-将`AWSIoTSiteWiseMonitorPortalAccess`策略附加到角色，或附加定义等效权限的策略。

   ```
   aws iam attach-role-policy --role-name MySiteWiseMonitorPortalRole --policy-arn
    arn:aws:iam::aws:policy/service-role/AWSIoTSiteWiseMonitorPortalAccess
   ```

   b. 对于 SiteWise 监控器（AI 感知）-将`IoTSiteWiseAIPortalAccessPolicy`策略附加到角色，或附加定义等效权限的策略。例如，创建具有门户访问权限的策略。以下示例创建了一个名为的策略`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": "*"
        }
    ]
}'
```

将服务角色附加到现有门户

1. 要检索门户的现有详细信息，请运行以下命令。*portal-id*替换为门户的 ID。

```
aws iotsitewise describe-portal --portal-id portal-id
```

该操作返回一个响应，其中包含如下格式的门户详细信息。

```
{
    "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"
}
```

2. 要将服务角色附加到门户，请运行以下命令。*role-arn*替换为服务角色 ARN，并将剩余参数替换为门户的现有值。

```
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 监视更新 AWSIo TSite WiseMonitorServiceRole

您可以查看有关 for M SiteWise onit AWSIoTSiteWiseMonitorServiceRoleor 更新的详细信息，从该服务开始跟踪更改时开始。要获得有关此页面变更的自动提醒，请订阅 “ Amazon IoT SiteWise 文档历史记录” 页面上的 RSS feed。

更改	描述	日期
AWSIoTSiteWiseMonitorPortalAccess - 更新的策略	Amazon IoT SiteWise 更新了警报功能的AWSIoTSiteWiseMonitorPortalAccess托管策略。	2021 年 5 月 27 日
Amazon IoT SiteWise 开始跟踪更改	Amazon IoT SiteWise 开始跟踪其服务角色的变化。	2020 年 12 月 15 日

在中设置事件警报的权限 Amazon IoT SiteWise

当您使用 Amazon IoT Events 警报模型监控 Amazon IoT SiteWise 资产属性时，您必须拥有以下 IAM 权限：

- 允许 Amazon IoT Events 向发送数据的 Amazon IoT Events 服务角色 Amazon IoT SiteWise。有关更多信息，请参阅 Amazon IoT Events 开发人员指南中的适用于 Amazon IoT Events的 身份和访问管理。
- 您必须拥有以下 Amazon IoT SiteWise 操作权限：`iotsitewise:DescribeAssetModel`和`iotsitewise:UpdateAssetModelPropertyRouting`。这些权限允许 Amazon IoT SiteWise 向 Amazon IoT Events 警报模型发送资产属性值。

有关更多信息，请参阅 IAM 用户指南的基于资源的策略。

所需的操作权限

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。

在定义 Amazon IoT Events 警报模型之前，必须授予以下权限，允许 Amazon IoT SiteWise 向警报模型发送资产属性值。

- `iotsitewise:DescribeAssetModel`，`iotsitewise:ListAssetModels`— Amazon IoT Events 允许检查资产属性是否存在。
- `iotsitewise:UpdateAssetModelPropertyRouting`— Amazon IoT SiteWise 允许自动创建允许 Amazon IoT SiteWise 向其发送数据的订阅 Amazon IoT Events。

有关 Amazon IoT SiteWise 支持的操作的更多信息，请参阅《服务授权参考》 Amazon IoT SiteWise 中定义的操作。

Example 权限策略示例 1

以下策略允许 Amazon IoT SiteWise 向任何 Amazon IoT Events 警报模型发送资产属性值。

```
{
    "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 权限策略示例 2

以下策略 Amazon IoT SiteWise 允许将指定资产属性的值发送到指定的 Amazon IoT Events 警报模型。

```
{
    "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"
                }
```

```
            }
        }
    ]
}
```

（可选）ListInputRoutings 权限

更新或删除资产模型时，Amazon IoT SiteWise 可以检查中的警报模型是否 Amazon IoT Events 正在监视与该资产模型关联的资产属性。这可以防止您删除 Amazon IoT Events 警报当前正在使用的资产属性。要在中启用此功能 Amazon IoT SiteWise，您必须拥有`iotevents:ListInputRoutings`权限。此权限 Amazon IoT SiteWise 允许调用所支持ListInputRoutings的 API 操作 Amazon IoT Events。

> **Note**
>
> 强烈建议您添加 `ListInputRoutings` 权限。

Example 权限策略示例

以下政策允许您更新和删除资产模型，并使用中的 `ListInputRoutings` API Amazon 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 监控器所需的权限

如果要在 SiteWise 监控门户中使用警报功能，则必须使用以下策略更新SiteWise 监控服务角色：

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

跨服务混淆了副手预防 Amazon IoT SiteWise

混淆代理问题是一个安全性问题，即不具有某操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 Amazon，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务) 调用另一项服务（所谓的服务)时，可能会发生跨服务模拟。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况， Amazon 提供可帮助您保护所有服务的服务委托人数据的工具，这些服务委托人有权限访问账户中的资源。

我们建议在资源策略中使用`aws:SourceArn`和`aws:SourceAccount`全局条件上下文密钥来限制为资源 Amazon IoT SiteWise 提供其他服务的权限。如果 `aws:SourceArn` 值不包含账户 ID，例如 Amazon S3 存储桶 Amazon 资源名称（ARN），您必须使用两个全局条件上下文键来限制权限。如果同时使用全局条件上下文密钥和包含账户 ID 的 `aws:SourceArn` 值，则 `aws:SourceAccount` 值和 `aws:SourceArn` 值中的账户在同一策略语句中使用时，必须使用相同的账户 ID。

- 如果您只希望将一个资源与跨服务访问相关联，请使用 `aws:SourceArn`。
- 如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 `aws:SourceAccount`。

的值`aws:SourceArn`必须是与`sts:AssumeRole`请求关联的 Amazon IoT SiteWise 客户资源。

防范混淆代理问题最有效的方法是使用 `aws:SourceArn` 全局条件上下文键和资源的完整 ARN。如果您不知道资源的完整 ARN，或正在指定多个资源，请针对 ARN 未知部分使用带有通配符 (*) 的 `aws:SourceArn` 全局上下文条件键。例如 `arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`。

Example - 混淆代理问题防范

以下示例显示了如何在中使用`aws:SourceArn`和`aws:SourceAccount`全局条件上下文键 Amazon IoT SiteWise 来防止出现混淆的副手问题。

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

对 Amazon IoT SiteWise 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复使用 Amazon IoT SiteWise 和 Amazon Identity and Access Management (IAM) 时可能遇到的常见问题。

主题

- 我无权在以下位置执行操作 Amazon IoT SiteWise
- 我无权执行 iam:PassRole
- 我想允许 Amazon 账户之外的人访问我的 Amazon IoT SiteWise 资源

我无权在以下位置执行操作 Amazon IoT SiteWise

如果 Amazon Web Services Management Console 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是指提供用户名和密码的人员。

当 `mateojackson` IAM 用户尝试使用控制台查看有关资产的详细信息，但不具有 `iotsitewise:DescribeAsset` 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotsitewise:DescribeAsset on resource: a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

在这种情况下，Mateo 请求管理员更新其策略，以允许他使用 `iotsitewise:DescribeAsset` 操作访问 ID 为 `a1b2c3d4-5678-90ab-cdef-22222EXAMPLE` 的资产资源。

我无权执行 `iam:PassRole`

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Amazon IoT SiteWise。

有些 Amazon Web Services 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amazon IoT SiteWise中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 Amazon 管理员。您的管理员是提供登录凭证的人。

我想允许 Amazon 账户之外的人访问我的 Amazon IoT SiteWise 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解是否 Amazon IoT SiteWise 支持这些功能，请参阅如何 Amazon IoT SiteWise 与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 Amazon Web Services 账户 ，请参阅 IAM 用户指南中的向您拥有 Amazon Web Services 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 Amazon Web Services 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 Amazon Web Services 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

合规性验证 Amazon IoT SiteWise

Amazon IoT SiteWise 不在任何 Amazon 合规计划的范围内。

有关特定合规计划范围内的 Amazon 服务列表，请参阅按合规计划划分的划分的范围内的服务。有关一般信息，请参阅合规计划。

您可以使用下载第三方审计报告 Amazon Artifact。有关更多信息，请参阅中的 “下载报告” Amazon Artifact。

您在使用 Amazon IoT SiteWise 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 Amazon 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在上部署以安全性和合规性为重点的基准环境的步骤。 Amazon
- HIPAA 安全与合规架构白皮书 — 本白皮书描述了公司如何使用来 Amazon 创建符合 HIPAA 标准的应用程序。
- 合规资源 — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- 使用Amazon Config 开发人员指南中的规则评估资源 — 该 Amazon Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- Amazon Security Hub— 此 Amazon 服务可全面了解您的安全状态 Amazon ，帮助您检查是否符合安全行业标准和最佳实践。
- 工业物联网解决方案的十大安全黄金法则 — 这篇博客文章介绍了有助于保护工业控制系统 (ICS)、工业物联网 (IIoT) 和云环境的十条黄金法则。

- 制造 OT 的安全最佳实践 — 本白皮书描述了为 Amazon 云设计、部署和架构这些本地混合制造工作负载的安全最佳实践。

韧性在 Amazon IoT SiteWise

Amazon 全球基础设施是围绕 Amazon 区域和可用区构建的。 Amazon 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础结构相比，可用区具有更高的可用性、容错性和可扩展性。

Amazon IoT SiteWise 完全托管，使用高度可用和耐用的 Amazon 服务，例如 Amazon S3 和 Amazon EC2。为了确保可用区中断时的可用性，需要跨多个可用区 Amazon IoT SiteWise 运行。

有关 Amazon 区域和可用区的更多信息，请参阅Amazon 全球基础设施。

除了 Amazon 全球基础架构外，还 Amazon IoT SiteWise 提供多项功能来帮助支持您的数据弹性和备份需求：

- 您可以 Amazon IoT Core 通过 MQTT 消息将属性值更新发布到，然后配置规则以对这些数据进行操作。使用此功能，您可以备份其他 Amazon 服务中的数据，例如 Amazon S3 和 Amazon DynamoDB。有关更多信息，请参阅与其他 Amazon 服务互动和使用资产属性通知将数据导出到 Amazon S3。
- 您可以使用 Amazon IoT SiteWise `Get*` APIs 来检索和备份历史资产属性数据。有关更多信息，请参阅 在中查询历史资产属性值 Amazon IoT SiteWise。
- 您可以使用 Amazon IoT SiteWise `Describe*` APIs 来检索资源的定义，例如资产和模型。可以对这些定义进行备份，并稍后使用它们重新创建资源。有关更多信息，请参阅 Amazon IoT SiteWise API 参考。

中的基础设施安全 Amazon IoT SiteWise

作为一项托管服务 Amazon IoT SiteWise ，受 Amazon 全球网络安全的保护。有关 Amazon 安全服务以及如何 Amazon 保护基础设施的信息，请参阅Amazon 云安全。要使用基础设施安全的最佳实践来设计您的 Amazon 环境，请参阅 S Amazon ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 Amazon 已发布的 API 调用 Amazon IoT SiteWise 通过网络进行访问。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service（Amazon STS）生成临时安全凭证来对请求进行签名。

SiteWise 在上 Amazon IoT Greengrass运行的 Edge 网关使用 X.509 证书和加密密钥来连接云并进行身份验证。 Amazon 有关更多信息，请参阅Amazon IoT Greengrass Version 1 开发人员指南中的Amazon IoT Greengrass设备身份验证和授权。

中的配置和漏洞分析 Amazon IoT SiteWise

IoT 实例集可能由大量具有不同功能、长期存在且地理位置分散的设备组成。这些特性导致实例集设置复杂且容易出错。由于设备的处理能力、内存和存储空间通常有限，因此它们并不总是支持加密和其他安全措施。此外，设备经常使用具有已知漏洞的软件。这些因素不仅令 IoT 实例集成为吸引黑客的目标，而且导致难以持续保护设备实例集安全。

Amazon IoT Device Defender 通过提供工具来识别安全问题和与最佳实践的偏差，从而应对这些挑战。 Amazon IoT Device Defender 用于分析、审计和监控连接的设备，以检测异常行为并降低安全风险。 Amazon IoT Device Defender 可以对设备队列进行审计，以确保它们遵守安全最佳实践并检测设备上的异常行为。这使得在您的 Amazon IoT 设备群中实施一致的安全策略成为可能，并在设备遭到入侵时快速做出响应。有关更多信息，请参阅《Amazon IoT Device Defender 开发人员指南》中的什么是 Amazon IoT Device Defender。

如果您使用 SiteWise Edge 网关向服务采集数据，则需要您负责配置和维护 SiteWise Edge 网关的环境。此职责包括升级到最新版本的 SiteWise Edge 网关的系统 Amazon IoT Greengrass 软件、软件和 Amazon IoT SiteWise 连接器。有关更多信息，请参阅《Amazon IoT Greengrass Version 1 开发人员指南》中的配置内 Amazon IoT Greengrass 核和管理 SiteWise Edge 网关。

的 VPC 终端节点 Amazon IoT SiteWise

接口 VPC 终端节点在您的虚拟私有云 (VPC) 和之间建立私有连接 Amazon IoT SiteWise。 Amazon PrivateLink为接口端点提供支持，允许对 Amazon IoT SiteWise API 操作进行私有访问。您可以绕过对互联网网关、NAT 设备、VPN 连接或 Amazon Direct Connect的需求。VPC 中的实例即使没有公有 IP 地址也可与 Amazon IoT SiteWise API 操作进行通信。您的 VPC 和 VPC 之间的流量 Amazon IoT SiteWise 不会离开 Amazon 网络。

每个接口端点均由子网中的一个或多个弹性网络接口表示。

在为设置接口 VPC 终端节点之前 Amazon IoT SiteWise，请查看Amazon PrivateLink 指南中的使用接口 VPC 终端节点访问 Amazon 服务。

中针对 VPC 终端节点的 API 操作 Amazon IoT SiteWise

Amazon IoT SiteWise 支持从您的 VPC 调用以下 Amazon IoT SiteWise API 操作：

- 对于所有数据面板 API 操作，请使用以下端点：

  ```
  data.iotsitewise.cn-north-1.amazonaws.com.cn
  ```

  数据平面 API 操作包括以下内容：

  - BatchGetAssetPropertyValue
  - BatchGetAssetPropertyValueHistory
  - BatchPutAssetPropertyValue
  - GetAssetPropertyAggregates
  - GetAssetPropertyValue
  - GetAssetPropertyValueHistory
  - GetInterpolatedAssetPropertyValues
- 对于用于管理资产模型、资产、 SiteWise 边缘网关、标签和账户配置的控制平面 API 操作，请使用以下端点。

  ```
  api.iotsitewise.cn-north-1.amazonaws.com.cn
  ```

  支持的控制平面 API 操作包含以下内容：

  - 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
-
- ListGateways
- ListTagsForResource
- UpdateGateway
-
- PutDefaultEncryptionConfiguration
- PutStorageConfiguration
- TagResource
- UntagResource

> **Note**
>
> 控制平面 API 操作的接口 VPC 终端节点目前不支持调用以下 SiteWise 监控 API 操作：
>
> - BatchAssociateProjectAssets

- BatchDisassociateProjectAssets
- CreateAccessPolicy
- CreateDashboard
- CreatePortal
- CreateProject
- DeleteAccessPolicy
- DeletePortal
- DeleteProject
- DescribeAccessPolicy
- DescribePortal
- DescribeProject
- ListAccessPolicies
- ListDashboards
- ListPortals
- ListProjects
- ListProjectAssets
- UpdateAccessPolicy
- UpdateDashboard
- UpdatePortal
- UpdateProject

为 Amazon IoT SiteWise创建接口 VPC 终端节点

要为 Amazon IoT SiteWise 服务创建 VPC 终端节点，请使用 Amazon VPC 控制台或 Amazon Command Line Interface (Amazon CLI)。有关更多信息，请参阅Amazon PrivateLink 指南中的使用接口 VPC 终端节点访问 Amazon 服务。

使用以下服务名称之一 Amazon IoT SiteWise 为创建 VPC 终端节点：

- 对于数据面板 API 操作，请使用以下服务名称：

```
com.cn.amazonaws.cn-north-1.iotsitewise.data
```

- 对于控制面板 API 操作，请使用以下服务名称：

```
com.cn.amazonaws.cn-north-1.iotsitewise.api
```

Amazon IoT SiteWise 通过接口 VPC 终端节点进行访问

当您创建接口终端节点时，我们会生成特定于终端节点的 DNS 主机名，您可以用来与之通信。Amazon IoT SiteWise默认情况下启用私有 DNS。有关更多信息，请参阅 Amazon VPC 用户指南中的使用私有托管区域。

如果您为终端节点启用私有 DNS，则可以通过以下 VPC 终端节点之一向 Amazon IoT SiteWise 发出 API 请求。

- 对于数据层面 API 操作，请使用以下端点：

```
data.iotsitewise.cn-north-1.amazonaws.com.cn
```

- 对于控制平面 API 操作，请使用以下端点：

```
data.iotsitewise.cn-north-1.amazonaws.com.cn
```

如果您为终端节点禁用私有 DNS，则必须执行以下操作才能 Amazon IoT SiteWise 通过终端节点进行访问：

1. 在 API 请求中指定 VPC 端点 url。
   - 对于数据面板 API 操作，请使用以下端点 url。将*vpc-endpoint-id*替换为您的 VPC 终端节点 ID 。

     ```
     vpc-endpoint-id.data.iotsitewise.cn-north-1.vpce.amazonaws.com.cn
     ```

   - 对于控制面板 API 操作，请使用以下端点 url。将*vpc-endpoint-id*替换为您的 VPC 终端节点 ID 。

     ```
     vpc-endpoint-id.api.iotsitewise.cn-north-1.vpce.amazonaws.com.cn
     ```

2. 禁用主机前缀注入功能。在 Amazon CLI 调用每个 Amazon SDKs API 操作时，在服务端点前面加上各种主机前缀。当您指定 VPC 终端节点 Amazon IoT SiteWise 时 URLs ，此功能会导致 Amazon CLI 和 Amazon SDKs 生成无效的结果。

> **Important**
>
> 您无法在 Amazon CLI 或中禁用主机前缀注入 Amazon Tools for PowerShell。这意味着，如果您禁用私有 DNS，则无法使用这些工具 Amazon IoT SiteWise 通过 VPC 终端节点进行访问。允许私有 DNS 使用 Amazon CLI 或 Amazon IoT SiteWise 通过终端节点 Amazon Tools for PowerShell 进行访问。

有关如何在中禁用主机前缀注入的更多信息 Amazon SDKs，请参阅每个 SDK 的以下文档部分：

- 适用于 C++ 的 Amazon SDK
- 适用于 Go 的 Amazon SDK
- 适用于 Go 的 Amazon SDK v2
- 适用于 Java 的 Amazon SDK
- Amazon SDK for Java 2.x
- 适用于 JavaScript 的 Amazon SDK
- 适用于 .NET 的 Amazon SDK
- 适用于 PHP 的 Amazon SDK
- 适用于 Python (Boto3) 的 Amazon SDK
- 适用于 Ruby 的 Amazon SDK

有关更多信息，请参阅Amazon PrivateLink 指南中的使用接口 VPC 终端节点访问 Amazon 服务。

为创建 VPC 终端节点策略 Amazon IoT SiteWise

您可以为 VPC 端点附加控制对 Amazon IoT SiteWise的访问的端点策略。该策略指定以下信息：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用端点策略控制对 VPC 端点的访问。

示例：用于 Amazon IoT SiteWise 操作的 VPC 终端节点策略

以下是的终端节点策略示例 Amazon IoT SiteWise。当连接到终端节点时，此策略将授予 Amazon 账户*iotsitewiseadmin123456789012*中的用户对指定资产 Amazon IoT SiteWise 执行列出的操作的访问权限。

```
{
    "Statement": [
        {
            "Action": [
                "iotsitewise:CreateAsset",
                "iotsitewise:ListGateways",
                "iotsitewise:ListTagsForResource"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iotsitewise:cn-north-1:123456789012:asset/
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
            "Principal": {
                "AWS": [
                    "123456789012:user/iotsitewiseadmin"
                ]
            }
        }
    ]
}
```

以下方面的安全最佳实践 Amazon IoT SiteWise

本主题包含的安全最佳实践 Amazon IoT SiteWise。

保持组件更新

如果您使用 SiteWise Edge 网关向服务采集数据，则您有责任配置和维护 SiteWise Edge 网关的环境。此责任包括升级到网关的系统软件、 Amazon IoT Greengrass 软件和连接器的最新版本。

> **Note**
>
> Amazon IoT SiteWise Edge 连接器在您的文件系统上存储机密。这些密钥控制谁可以查看您的 SiteWise Edge 网关中缓存的数据。强烈建议您为运行 E SiteWise dge 网关的系统开启磁盘或文件系统加密。

有关如何在 Amazon IoT SiteWise 控制台中升级组件的信息，请参阅更改 SiteWise Edge 网关组件包的版本。

加密您的 SiteWise Edge 网关的文件系统

加密和保护您的 SiteWise Edge 网关，因此您的工业数据在通过 SiteWise Edge 网关时是安全的。如果您的 SiteWise Edge 网关具有硬件安全模块，则可以进行配置 Amazon IoT Greengrass 以保护您的 SiteWise Edge 网关。有关更多信息，请参阅 Amazon IoT Greengrass Version 1 开发人员指南中的硬件安全性集成。否则，请参阅适用于您的操作系统的文档，了解如何加密和保护文件系统。

安全访问您的边缘配置

不要共享您的 Edge 控制台应用程序密码或 Mon SiteWise itor 应用程序密码。请勿将此密码放在任何人都能看到的地方。为密码配置适当的过期时间，实施健康的密码轮换策略。

在 Siemens Industrial Edge Management 上保护数据安全

您选择与 Amazon IoT SiteWise Edge 共享的设备数据将在您的Siemens IEM Databus配置主题中确定。通过选择要与 SiteWise Edge 共享的主题，您就是在与之共享主题级数据。 Amazon IoT SiteWiseSiemens Industrial Edge Marketplace是一个独立的市场，与 Amazon。为了保护您的共享数据，除非您使用 SiteWise Edge 应用程序，否则不会运行Siemens Secured Storage。有关更多信息，请参阅 Siemens 文档中的安全存储。

向 SiteWise 监控器用户授予可能的最低权限

通过对门户用户使用最小访问策略权限集，遵循最低权限原则。

- 创建门户时，请定义一个角色，允许该门户所需的最小资源集。有关更多信息，请参阅 将服务角色用于 Amazon IoT SiteWise Monitor。
- 当您和门户管理员创建并共享项目时，请使用该项目所需的最小资源集。
- 当身份不再需要访问门户或项目时，请将其从该资源中删除。如果该身份不再适用于您的组织，请从您的身份存储中删除该身份。

最低权限原则最佳实践也适用于 IAM 角色。有关更多信息，请参阅 策略最佳实践。

不要暴露敏感信息

您应该禁止记录凭证和其他敏感信息，例如个人身份信息 (PII)。尽管访问 SiteWise Edge 网关上的本地日志需要根权限，而访问 CloudWatch 日志需要 IAM 权限，但我们仍建议您实施以下保护措施。

- 请勿在资产或模型的名称、描述或属性中使用敏感信息。
- 请勿在 SiteWise Edge 网关或源名称中使用敏感信息。
- 请勿在门户、项目或控制面板的名称或描述中使用敏感信息。

遵循 Amazon IoT Greengrass 安全最佳实践

遵循 SiteWise Edge 网关 Amazon IoT Greengrass 的安全最佳实践。有关更多信息，请参阅Amazon IoT Greengrass Version 1 开发人员指南中的安全最佳实践。

另请参阅

- Amazon IoT 开发人员指南中的安全最佳实践
- 工业 IoT 解决方案的十大安全黄金法则

登录并监控 Amazon IoT SiteWise

监控是维护和其他 Amazon 解决方案的可靠性、可用性和性能的重要组成部分。 Amazon IoT SiteWise Amazon IoT SiteWise 支持以下监控工具来监视服务，在出现问题时进行报告，并在适当时自动采取措施：

- Amazon 会实时 CloudWatch监控您的 Amazon 资源和您运行 Amazon 的应用程序。收集和跟踪指标，创建自定义的控制面板，以及设置警报以在指定的指标达到特定阈值时通知您或采取措施。例如，您可以 CloudWatch 跟踪您的 Amazon EC2 实例的 CPU 使用率或其他指标，并在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon CloudWatch Logs 监控、存储和访问来自 SiteWise 边缘网关和其他来源的日志文件。CloudTrail CloudWatch 日志可以监视日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南。
- Amazon CloudTrail捕获由您的账户或代表您的 Amazon 账户进行的 API 调用和相关事件。然后，将日志文件 CloudTrail 传送到您指定的 Amazon S3 存储桶。您可以识别哪些用户和帐户拨打了电话 Amazon、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅 Amazon CloudTrail 用户指南。

主题

- 使用 Amazon CloudWatch 日志进行监控
- 监控 SiteWise 边缘网关日志
- Amazon IoT SiteWise 使用 Amazon CloudWatch 指标进行监控
- 使用记录 Amazon IoT SiteWise API 调用 Amazon CloudTrail

使用 Amazon CloudWatch 日志进行监控

配置 Amazon IoT SiteWise 为将信息记录到 CloudWatch 日志，以监控服务并对其进行故障排除。

使用 Amazon IoT SiteWise 控制台时， Amazon IoT SiteWise 会创建一个服务相关角色，允许服务代表您记录信息。如果您不使用 Amazon IoT SiteWise 控制台，则必须手动创建服务相关角色才能接收日志。有关更多信息，请参阅 为 Amazon IoT SiteWise创建服务相关角色。

您必须有一个允许将日志事件放 Amazon IoT SiteWise 入 CloudWatch 流中的资源策略。要创建和更新 CloudWatch 日志的资源策略，请运行以下命令。*logging-policy-name*替换为要创建的策略的名称。

```
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 日志还支持 a ws: SourceArn 和 a ws: SourceAccount 条件上下文密钥。这些条件上下文键是可选的。

要创建或更新仅允许 Amazon IoT SiteWise 将与指定资源关联的日志放入 CloudWatch 流的 Amazon IoT SiteWise 资源策略，请运行命令并执行以下操作：

- *logging-policy-name*替换为要创建的策略的名称。
- *source-ARN*替换为 Amazon IoT SiteWise 资源的 ARN，例如资产模型或资产。要查找每种 Amazon IoT SiteWise 资源类型的 ARN，请参阅《服务授权参考》 Amazon IoT SiteWise中定义的资源类型。
- 替换为*account-ID*与指定 Amazon IoT SiteWise 资源关联的 Amazon 账户 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\"]}}}]}"
```

默认情况下， Amazon IoT SiteWise 不将信息记录到 CloudWatch 日志。要激活日志记录，请选择 “禁用” (OFF) 以外的日志级别。 Amazon IoT SiteWise 支持以下日志级别：

- OFF – 日志记录处于关闭状态。
- ERROR – 记录错误。
- INFO – 记录错误和信息性消息。

您可以将 SiteWise Edge 网关配置为通过 Lo CloudWatch gs 记录信息 Amazon IoT Greengrass。有关更多信息，请参阅 监控 SiteWise 边缘网关日志。

如果 Amazon IoT Core 要对 Amazon IoT SiteWise 规则操作进行故障排除，也可以配置为将信息 CloudWatch 记录到日志。有关更多信息，请参阅 对 Amazon IoT SiteWise 规则操作进行故障排除。

目录

- 管理登录 Amazon IoT SiteWise
  - 查找日志记录级别
  - 更改日志记录级别
- 示例： Amazon IoT SiteWise 日志文件条目

管理登录 Amazon IoT SiteWise

使用 Amazon IoT SiteWise 控制台或 Amazon CLI 执行以下日志配置任务。

查找日志记录级别

Console

使用以下过程可在 Amazon IoT SiteWise 控制台中查找当前的日志记录级别。

要查找您当前的 Amazon IoT SiteWise 日志级别

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择日志记录选项。

   当前日志记录状态显示在日志记录状态下。如果已激活日志记录，则当前日志记录级别将显示在详细程度级别下。

Amazon CLI

运行以下命令以查找您当前的 Amazon IoT SiteWise 日志记录级别 Amazon CLI。

```
aws iotsitewise describe-logging-options
```

此操作将采用以下格式返回包含日志记录级别的响应。

```
{
  "loggingOptions": {
    "level": "String"
  }
}
```

更改日志记录级别

使用以下步骤在 Amazon IoT SiteWise 控制台或使用中更改日志记录级别 Amazon CLI。

Console

更改您的 Amazon IoT SiteWise 日志级别

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在左侧导航窗格中，选择日志记录选项。
3. 选择编辑。
4. 选择要激活的详细程度级别。
5. 选择保存。

Amazon CLI

运行以下 Amazon CLI 命令来更改您的 Amazon IoT SiteWise 日志级别。*logging-level*替换为所需的日志级别。

```
aws iotsitewise put-logging-options --logging-options level=logging-level
```

示例： Amazon IoT SiteWise 日志文件条目

每个 Amazon IoT SiteWise 日志条目都包含事件信息和该事件的相关资源，因此您可以了解和分析日志数据。

以下示例显示了成功创建资产模型时 Amazon IoT SiteWise 记录的 CloudWatch 日志条目。

```
{
  "eventTime": "2020-05-05T00:10:22.902Z",
  "logLevel": "INFO",
  "eventType": "AssetModelCreationSuccess",
  "message": "Successfully created asset model.",
  "resources": {
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
  }
}
```

监控 SiteWise 边缘网关日志

您可以将 Amazon IoT SiteWise Edge 网关配置为将信息记录到 Amazon Lo CloudWatch gs 或本地文件系统。

主题

- 使用 Amazon CloudWatch 日志
- 使用服务登录 Amazon IoT SiteWise
- 使用事件日志

使用 Amazon CloudWatch 日志

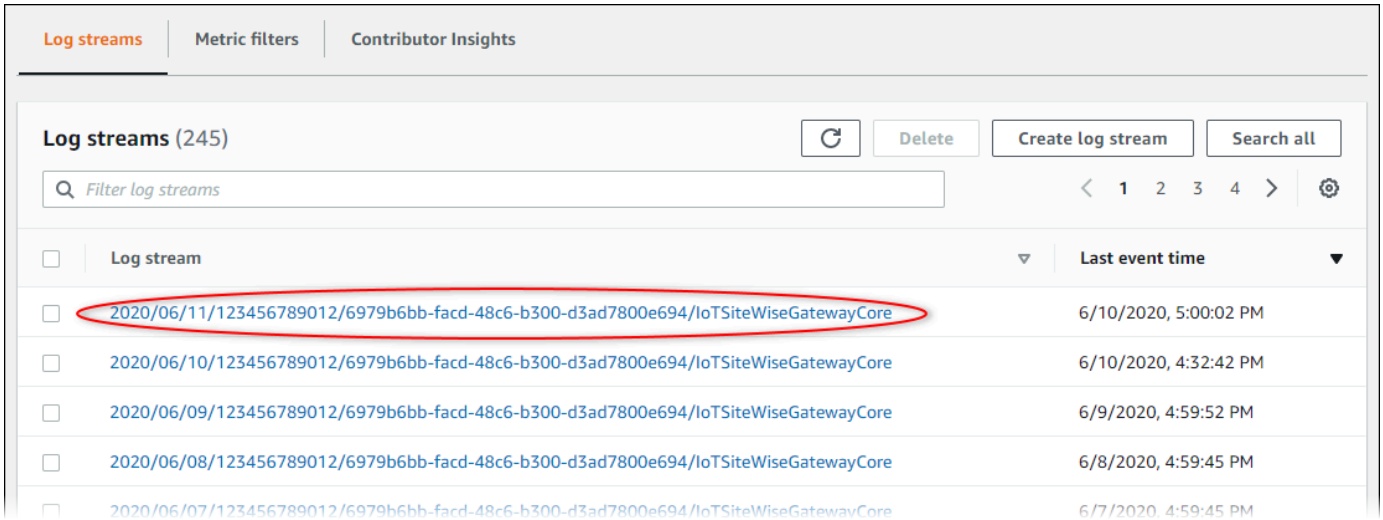
您可以将 SiteWise Edge 网关配置为向日志发送 CloudWatch 日志。有关更多信息，请参阅《Amazon IoT Greengrass Version 2 开发人员指南》中的启用 CloudWatch 日志记录。

配置和访问 CloudWatch 日志（控制台）

1. 导航至 CloudWatch 控制台。
2. 在导航窗格中，选择 日志组。
3. 您可以在以下日志 Amazon IoT SiteWise 组中找到组件日志：

   - `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgeCollectorOpcua`— 边 SiteWise 缘网关组件的日志，该组件从 SiteWise 边缘网关的 OPC UA 源收集数据。
   - `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgePublisher`— 将 OPC UA 数据流发布到 Amazon IoT SiteWise的 SiteWise Edge 网关组件的日志。

   选择要调试的函数的日志组。
4. 选择名称以您的 Amazon IoT Greengrass 群组名称结尾的日志流。默认情况下，首先 CloudWatch 显示最新的日志流。



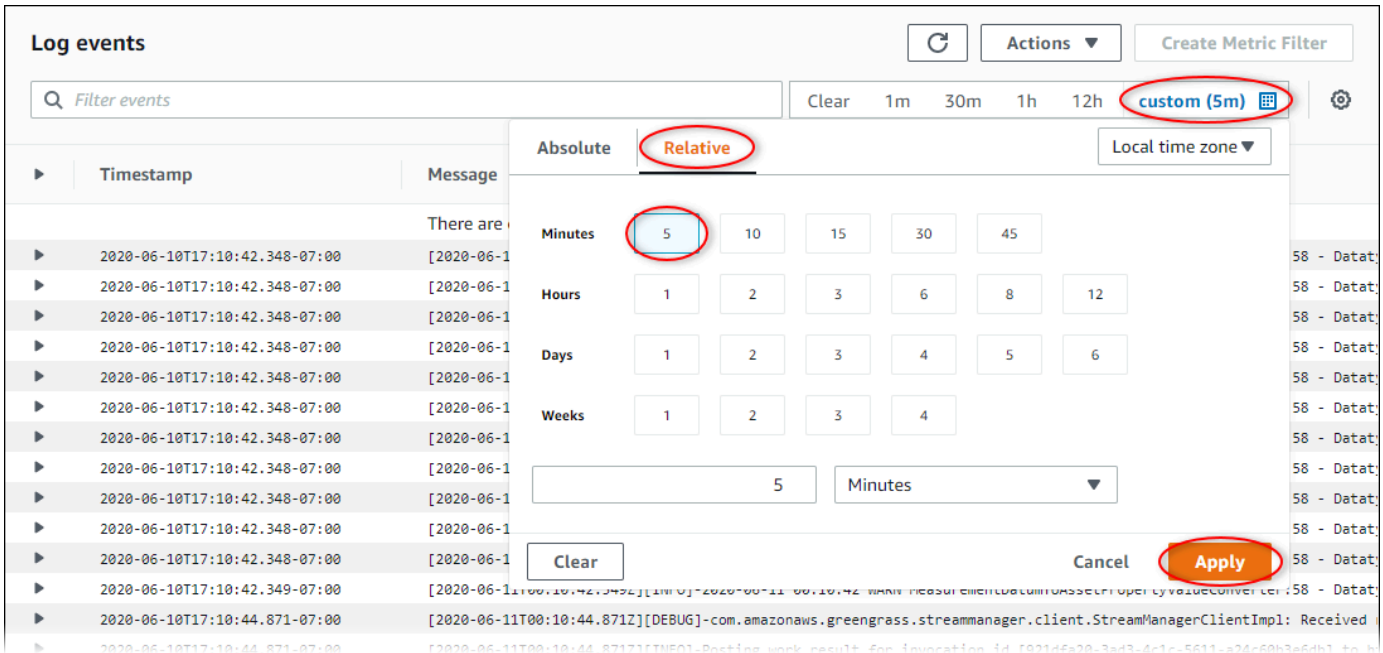
5. 要显示最近 5 分钟的日志，请执行以下操作：

   a. 选择右上角的 自定义。

   b. 选择 相对。

   c. 选择 5 分钟。

   d. 选择 应用。



6. （可选）要查看更少的日志，可以从右上角选择 1 分钟。

7. 滚动到日志条目底部可以显示最近的日志。

使用服务登录 Amazon IoT SiteWise

SiteWise Edge 网关设备包含服务日志文件以帮助调试问题。以下各节将帮助您查找和使用 Amazon IoT SiteWise OPC UA Collector 和 P Amazon IoT SiteWise ublisher 组件的服务日志文件。

Amazon IoT SiteWise OPC UA 收集器服务日志文件

Amazon IoT SiteWise OPC UA 收集器组件使用以下日志文件。

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log
```

查看此组件的日志

- 在核心设备上运行以下命令可实时查看此组件的日志文件。将*/greengrass/v2*或*C:\greengrass\v2*替换为 Amazon IoT Greengrass 根文件夹的路径。

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log -Tail
   10 -Wait
  ```

Amazon IoT SiteWise 发布者服务日志文件

P Amazon IoT SiteWise ublisher 组件使用以下日志文件。

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log
```

查看此组件的日志

- 在核心设备上运行以下命令可实时查看此组件的日志文件。将*/greengrass/v2*或*C:\greengrass\v2*替换为 Amazon IoT Greengrass 根文件夹的路径。

  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
  ```

使用事件日志

SiteWise Edge 网关设备包含事件日志文件，以帮助调试问题。以下各节将帮助您查找和使用 Amazon IoT SiteWise OPC UA Collector 和 P Amazon IoT SiteWise ublisher 组件的事件日志文件。

Amazon IoT SiteWise OPC UA 收集器事件日志

Amazon IoT SiteWise OPC UA Collector 组件包括事件日志，可帮助客户识别和修复问题。日志文件与本地日志文件是分开的，位于以下位置。将*/greengrass/v2*或*C:\greengrass\v2*替换为 Amazon IoT Greengrass 根文件夹的路径。

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua/logs/
IotSiteWiseOpcUaCollectorEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeCollectorOpcua\logs
\IotSiteWiseOpcUaCollectorEvents.log
```

此日志包含详细信息和问题排查说明。问题排查信息与诊断信息一起提供，并附有如何解决问题的说明，有时还会提供指向更多信息的链接。诊断信息包含以下内容：

- 严重性级别
- Timestamp
- 其他事件特定信息

Example 示例日志

```
dataSourceConnectionSuccess:
  Summary: Successfully connected to OpcUa server
  Level: INFO
  Timestamp: '2023-06-15T21:04:16.303Z'
  Description: Successfully connected to the data source.
  AssociatedMetrics:
  - Name: FetchedDataStreams
    Description: The number of fetched data streams for this data source
    Value: 1.0
    Namespace: IoTSiteWise
    Dimensions:
    - Name: SourceName
      Value: SourceName{value=OPC-UA Server}
    - Name: ThingName
      Value: test-core
  AssociatedData:
  - Name: DataSourceTrace
    Description: Name of the data source
    Data:
    - OPC-UA Server
  - Name: EndpointUri
    Description: The endpoint to which the connection was attempted.
    Data:
    - '"opc.tcp://10.0.0.1:1234"'
```

Amazon IoT SiteWise 发布者事件日志

Amazon IoT SiteWise Publisher 组件包括事件日志，可帮助客户识别和修复问题。日志文件与本地日志文件是分开的，位于以下位置。将*/greengrass/v2*或*C:\greengrass\v2*替换为 Amazon IoT Greengrass 根文件夹的路径。

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/logs/
IotSiteWisePublisherEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgePublisher\logs
\IotSiteWisePublisherEvents.log
```

此日志包含详细信息和问题排查说明。问题排查信息与诊断信息一起提供，并附有如何解决问题的说明，有时还会提供指向更多信息的链接。诊断信息包含以下内容：

- 严重性级别
- Timestamp
- 其他事件特定信息

Amazon IoT SiteWise 使用 Amazon CloudWatch 指标进行监控

您可以使用 Amazon IoT SiteWise 进行监控 CloudWatch，它收集原始数据并将其处理为可读的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

Amazon IoT SiteWise 将以下各节中列出的指标和维度发布到`AWS/IoTSiteWise`命名空间。

> **Tip**
>
> Amazon IoT SiteWise 每隔一分钟发布指标。当您在 CloudWatch 控制台的图表中查看这些指标时，我们建议您选择 1 分钟的时间段。这样，您便能查看指标数据的最高可用解析。

主题

- Amazon IoT Greengrass Version 2 网关指标

Amazon IoT Greengrass Version 2 网关指标

Amazon IoT SiteWise 发布经典流、V2 网关和支持 MQTT 的 V3 网关的网关指标。除非另有说明，否则每个指标都适用于两个自托管网关版本。所有 SiteWise Edge 网关指标每隔一分钟发布一次。

SiteWise 边缘网关指标

指标	描述
`Gateway.AvailableMemory`	SiteWise 边缘网关的可用内存。 单位：字节 维度：无
`Gateway.AvailableDiskSpace`	SiteWise Edge 网关的可用磁盘空间。 单位：字节 维度：无
`Gateway.CloudConnectivity`	SiteWise 边缘网关的云连接状态。 单位：无 维度： GatewayId
`Gateway.CpuUsage`	SiteWise 边缘网关的 CPU 使用率。 单位：百分比 维度：无
`Gateway.TotalDiskSpace`	SiteWise Edge 网关的总磁盘空间。 单位：字节 维度：无
`Gateway.TotalMemory`	SiteWise 边缘网关的总内存。 单位：字节

指标	描述
	维度：无
`Gateway.UsedDiskSpace`	SiteWise Edge 网关的已用磁盘空间。 单位：字节 维度：无
`Gateway.UsedMemory`	SiteWise 边缘网关的已用内存。 单位：字节 维度：无
`Gateway.UsedPercentageDiskSpace`	SiteWise Edge 网关磁盘空间的已用百分比。 单位：字节 维度：无
`Gateway.UsedPercentageMemory`	SiteWise Edge 网关的已用内存百分比。 单位：字节 维度：无

Amazon IoT SiteWise 发布者指标

指标	描述
`IoTSiteWisePublisher.ComponentBuildVersion`	该指标表示网关上运行的 IoT SiteWise 发布者组件的构建版本。值为 1 表示网关正在运行与该`ComponentBuildVersion` 维度相对应的发布者版本。 单位：1 尺寸： GatewayId， ComponentBuildVersion

指标	描述
IoTSiteWisePublisher.DroppedCount	由 SiteWise Edge 网关 (GatewayId) 丢弃但未发布到云端的数据点数量，每分钟生成一次。 单位：计数 尺寸： GatewayId
IoTSiteWisePublisher.Heartbeat	由发布者在 SiteWise Edge 网关中每分钟生成一次。 单位：1 (1 表示发布者正在运行但缺少表示发布者未运行的数据点。) 尺寸： GatewayId
IoTSiteWisePublisher.IsConnectedToMqttBroker	由发布者在 SiteWise Edge 网关中每分钟生成一次。 单位：1 (1 表示发布者已连接到 MQTT 代理。) 尺寸： GatewayId
IoTSiteWisePublisher.MessageCheckpointPersistenceErrorCount	该指标表明网关检测到用于跟踪发布者处理的数据的检查点文件存在问题。值为 1 表示已发生故障。 单位：无 尺寸： AccountId， GatewayId
IoTSiteWisePublisher.MqttMessageReceivedSuccessCount	发布者成功接收来自 MQTT 代理的消息数量，每分钟生成一次。 单位：计数 尺寸： GatewayId

指标	描述
IoTSiteWisePublisher.MqttReceivedSuccessBytes	发布者成功接收来自 MQTT 代理的消息数据的字节数，每分钟生成一次。 单位：计数 尺寸： GatewayId
IoTSiteWisePublisher.NumberOfSubscriptionsToMqttBroker	发布者订阅 MQTT 代理的主题数量，每分钟生成一次。多级通配符主题计为 1。 单位：计数 尺寸： GatewayId
IoTSiteWisePublisher.NumberOfUniqueMqttTopicsReceived	发布者从 MQTT 代理收到的唯一主题的数量，每分钟生成一次。 单位：计数 尺寸： GatewayId
IoTSiteWisePublisher.PublishFailureCount	SiteWise 边缘网关 (GatewayId) 发布失败的数据点数量，每分钟生成。 单位：计数 尺寸： GatewayId
IoTSiteWisePublisher.PublishRejectedCount	SiteWise 边缘网关 (GatewayId) 每分钟从云端拒绝的数据点数量。 单位：计数 尺寸： GatewayId

指标	描述
`IoTSiteWisePublisher.PublishSuccessCount`	SiteWise 边缘网关 (`GatewayId`) 每分钟成功发布到云端的数据点数量。 单位：计数 尺寸： GatewayId
`IoTSiteWisePublisher.PublishToS3FailureCount`	网关（`GatewayId` ）未能发布到 Amazon S3 存储桶的数据点的数量。 单位：计数 尺寸： GatewayId
`IoTSiteWisePublisher.PublishToS3SuccessCount`	网关（`GatewayId` ）成功发布到 Amazon S3 存储桶的数据点的数量。 单位：计数 尺寸： GatewayId

OPC UA 收集器指标

指标	描述
`OpcUaCollector.ActiveDataStreamCount`	SiteWise 边缘网关 (`gatewayId`) 为 OPC UA 源 (`sourceName`) 订阅的数据流数量。 单位：计数 尺寸: GatewayId, SourceName, PropertyGroup
`OpcUaCollector.ComponentBuildVersion` （不适用于经典直播、V2 网关）	该指标表示网关上运行的 IoT SiteWise OPC UA 收集器组件的构建版本。值为 1 表示网关正在运行与该`ComponentBuildVersion` 维度相对应的收集器版本。 单位：1

指标	描述
	尺寸： GatewayId， ComponentBuildVersion
`OpcUaCollector.ConversionErrors`	SiteWise 边缘网关 (`gatewayId`) 接收到的 OPC UA 源 (`sourceName`) 的数据点数量，这些数据点在将数据发送到 Amazon IoT SiteWise 时导致转换错误。OPC UA 收集器不会摄取这些数据点。 单位：计数 尺寸： GatewayId， SourceName
`OpcUaCollector.Heartbeat`	每分钟为连接到 SiteWise Edge 网关 (`sourceName`) 的每个 OPC UA 源 (`gatewayId`) 生成一次。 单位：计数（1 表示源已连接，0 表示源已断开连接。） 尺寸： GatewayId， SourceName
`OpcUaCollector.IncomingValuesCount`	SiteWise 边缘网关 (`gatewayId`) 每分钟为 OPC UA 源 (`sourceName`) 接收的数据点数量。 单位：计数 尺寸: GatewayId, SourceName, PropertyGroup
`OpcUaCollector.IncomingValueErrors`	SiteWise 边缘网关 (`gatewayId`) 从 OPC UA 源 (`sourceName`) 接收的无效值的数据点数量。OPC UA 收集器不会摄取这些数据点，而是每分钟生成一次。 单位：计数 尺寸: GatewayId, SourceName, PropertyGroup

指标	描述
`OpcUaCollector.IsConnectedToMqttBroker` （不适用于经典直播、V2 网关）	由 SiteWise Edge 网关中的 IoT SiteWise OPC UA 收集器组件每分钟生成一次。 单位：1（1 表示物联网 SiteWise OPC UA 收集器组件连接到 MQTT 代理） 尺寸： GatewayId
`OpcUaCollector.MqttMessagesDroppedCount` （不适用于经典直播、V2 网关）	物联网 SiteWise OPC UA 收集器组件丢弃的 MQTT 消息数量。 单位：计数 尺寸： GatewayId， SourceName
`OpcUaCollector.MqttMessagesPublishedBytes` （不适用于经典直播、V2 网关）	物联网 SiteWise OPC UA 收集器组件成功发布到 MQTT 代理的 MQTT 消息数据的字节数。 单位：计数 尺寸： GatewayId， SourceName
`OpcUaCollector.MqttMessagesPublishedCount` （不适用于经典直播、V2 网关）	物联网 SiteWise OPC UA 收集器组件成功向 MQTT 代理发布的 MQTT 消息的数量。 单位：计数 尺寸： GatewayId， SourceName
`OpcUaCollector.NullValueCount` （不适用于经典直播、V2 网关）	物联网 SiteWise OPC UA 收集器组件从 OPC UA 服务器接收的空值的数量。 单位：计数 尺寸: GatewayId, SourceName, PropertyGroup

指标	描述
`OpcUaCollector.NumberOfUniqueMqttTopicsPublished` （不适用于经典直播、V2 网关）	物联网 SiteWise OPC UA 收集器向 MQTT 代理发布的唯一 MQTT 主题的数量。 单位：计数 尺寸： GatewayId， SourceName

Amazon IoT SiteWise 处理器指标

指标	描述
`Gateway.DataProcessor.IngestionThrottled` （在支持 MQTT 的 V3 网关上不可用）	受限数据点的数量，每分钟生成一次。 单位：计数 尺寸： ThrottledAt
`Gateway.DataProcessor.MeasurementRejected` （在支持 MQTT 的 V3 网关上不可用）	被拒绝的测量值的数量，每分钟生成一次。 单位：计数 维度：原因
`Gateway.DataProcessor.MessagesRemaining` （在支持 MQTT 的 V3 网关上不可用）	流中剩余的消息数量，每分钟生成一次。 单位：计数 尺寸： StreamName
`Gateway.DataProcessor.ProcessingError` （在支持 MQTT 的 V3 网关上不可用）	处理错误的数量，每分钟生成一次。 单位：计数 维度：原因

使用记录 Amazon IoT SiteWise API 调用 Amazon CloudTrail

Amazon IoT SiteWise 与 Amazon CloudTrail一项服务集成，该服务提供用户、角色或 Amazon 服务在中执行的操作的记录 Amazon IoT SiteWise。 CloudTrail 将发出的 API 调用捕获 Amazon IoT SiteWise 为事件。捕获的调用包括来自 Amazon IoT SiteWise 控制台的调用和对 Amazon IoT SiteWise API 操作的代码调用。如果您创建跟踪，则可以激活向 Amazon S3 存储桶持续传输事件，包括的事件 Amazon IoT SiteWise。 CloudTrail 如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 Amazon IoT SiteWise、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

有关的更多信息 CloudTrail，请参阅《Amazon CloudTrail 用户指南》。

Amazon IoT SiteWise 信息在 CloudTrail

CloudTrail 在您创建 Amazon 账户时已在您的账户上激活。当支持的事件活动发生在中时 Amazon IoT SiteWise，该活动会与其他 Amazon 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 Amazon 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史查看 CloudTrail 事件。

要持续记录您 Amazon 账户中的事件，包括的事件 Amazon IoT SiteWise，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 Amazon 区域。跟踪记录 Amazon 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 Amazon 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail日志文件

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根证书还是 Amazon Identity and Access Management (IAM) 用户凭证发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

Amazon IoT SiteWise 中的数据事件 CloudTrail

数据事件可提供对资源或在资源中所执行资源操作（例如，读取或写入 Amazon S3 对象）的相关信息。这些也称为数据层面操作。数据事件通常是高容量活动。默认情况下， CloudTrail 不记录数据事件。 CloudTrail 事件历史记录不记录数据事件。

记录数据事件将收取额外费用。有关 CloudTrail 定价的更多信息，请参阅Amazon CloudTrail 定价。

您可以使用 CloudTrail 控制台或 CloudTrail API 操作记录 Amazon IoT SiteWise 资源类型的数据事件。 Amazon CLI本节中的表格显示了可用的资源类型 Amazon IoT SiteWise。

- 要使用 CloudTrail 控制台记录数据事件，请创建跟踪或事件数据存储以记录数据事件，或者更新现有的跟踪或事件数据存储以记录数据事件。
  1. 选择数据事件来记录数据事件。
  2. 在数据事件类型列表中，选择要对其记录数据事件的资源类型。
  3. 选择要使用的记录选择器模板。可以记录资源类型的所有数据事件、记录所有 `readOnly` 事件、记录所有 `writeOnly` 事件，或者创建自定义记录选择器模板来根据 `readOnly`、`eventName` 和 `resources.ARN` 字段进行筛选。
- 要使用记录数据事件 Amazon CLI，请将`--advanced-event-selectors`参数配置为将`eventCategory`字段设置为等于`Data`并将该`resources.type`字段设置为资源类型值（参见表）。可以添加条件来根据 `readOnly`、`eventName` 和 `resources.ARN` 字段的值进行筛选。
  - 要配置记录数据事件的跟踪，请运行Amazon CloudTrail put-event-selectors命令。有关更多信息，请参阅使用 Amazon CLI记录跟踪的数据事件。
  - 要将事件数据存储配置为记录数据事件，请运行Amazon CloudTrail create-event-data-store命令创建新的事件数据存储以记录数据事件，或者运行Amazon CloudTrail update-event-data-store命令来更新现有的事件数据存储。有关更多信息，请参阅 Logging data events for event data stores with the Amazon CLI。

下表列出了 Amazon IoT SiteWise 资源类型。数据事件类型（控制台）列显示要从控制 CloudTrail 台上的数据事件类型列表中选择的值。res ources.type 值列显示该`resources.type`值，您将在使用或配置高级事件选择器时指定该值。 Amazon CLI CloudTrail APIs“ APIs 记录到的数据 CloudTrail” 列显示了 CloudTrail针对该资源类型记录的 API 调用。

数据事件类型（控制台）	resources.type 值	数据 APIs 已记录到 CloudTrail *
Amazon IoT SiteWise 资产	`Amazon::IoTSiteWise::Asset`	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory • BatchGetAssetPropertyAggregates
Amazon IoT SiteWise 时间序列	`Amazon::IoTSiteWise::TimeSeries`	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory

数据事件类型（控制台）	resources.type 值	数据 APIs 已记录到 CloudTrail *
		• BatchGetAssetPropertyAggregates
Amazon IoT SiteWise Assistant	`Amazon::SitewiseAssistant::Conversation`	• InvokeAssistant

> **Note**
>
> Cloudtrail 事件中记录的 resources.type 取决于 API 请求中使用的标识符。如果在请求中指定了资产 ID，则会记录资产 resources.type，否则会记录 resourc TimeSeries es.type。

*可以将高级事件选择器配置为根据 `eventName`、`readOnly` 和 `resources.ARN` 字段进行筛选，从而仅记录那些对您很重要的事件。有关这些字段的更多信息，请参阅 AdvancedFieldSelector.

Amazon IoT SiteWise 中的管理事件 CloudTrail

日志管理事件提供有关对您 Amazon 账户中的资源执行的管理操作的信息。这些也称为控制面板操作。默认情况下， CloudTrail 记录管理事件。

Amazon IoT SiteWise 将所有 Amazon IoT SiteWise 控制平面操作记录为管理事件。有关 Amazon IoT SiteWise 记录到的 Amazon IoT SiteWise 控制平面操作的列表 CloudTrail，请参阅 Amazon IoT SiteWise API 参考。

示例： Amazon IoT SiteWise 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关所请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了演示该`CreateAsset`操作的 CloudTrail 日志条目。

```
{
```

```
  "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"
}
```

标记您的 Amazon IoT SiteWise 资源

为 Amazon IoT SiteWise 资源添加标签提供了一种高效地分类、管理和检索组织资产的强大方法。通过分配由键值对组成的标签，您可以将描述性元数据附加到您的资源中。标签中的元数据可用于简化操作。例如，在风电场场景中，通过标签，您可以使用位置、容量和运行状态等特定属性标记涡轮机，实现在 Amazon IoT SiteWise中进行快速识别和管理。

将标签与 Amazon Identity and Access Management (IAM) 策略集成可通过定义条件访问规则来增强安全性和操作控制。这意味着您可以仅指定具有特定标签的用户。例如，只有那些被标记为特定角色或部门的人才能访问或修改特定的资源。

在中使用标签 Amazon IoT SiteWise

使用标签按用途、所有者、环境或用例的任何其他分类对 Amazon IoT SiteWise 资源进行分类。在您具有同一类型的许多资源时，可以根据特定资源的标签快速识别该资源。

每个标签都由您指定的一个键和一个可选值组成。例如，您可以为资产模型建立一系列标签，以便根据它们支持的工业流程对其进行跟踪。建议为您管理的每种资源开发一组量身定制的标签键。使用一组连续的标签键，管理资源时会更加轻松。

用 tag Amazon Web Services Management Console

中的标签编辑器 Amazon Web Services Management Console 提供了一种集中、统一的方式，供您为来自所有 Amazon 服务的资源创建和管理标签。有关更多信息，请参阅《标签 Amazon 资源和标签编辑器用户指南》中的标签编辑器入门。

使用 Amazon IoT SiteWise API 进行标记

Amazon IoT SiteWise API 还使用标签。在创建标签之前，请注意标签限制。有关更多信息，请参阅 Amazon Web Services 一般参考中的标签命名和使用惯例。

- 要在创建资源时添加标签，请在资源的 `tags` 属性中定义这些标签。
- 要向现有资源添加标签或更新标签值，请使用TagResource操作。
- 要从资源中移除标签，请使用UntagResource操作。
- 要检索与资源关联的标签，请使用ListTagsForResource操作，或者描述资源并检查其`tags`属性。

下表列出了您可以使用 Amazon IoT SiteWise API 标记的资源及其对应的`Create`和`Describe`操作。

可标记的资源 Amazon IoT SiteWise

资源	创建操作	描述操作
资产模型或组件模型	CreateAssetModel	DescribeAssetModel
资产	CreateAsset	DescribeAsset
SiteWise 边缘网关	CreateGateway	DescribeGateway
门户	CreatePortal	DescribePortal
Project	CreateProject	DescribeProject
控制面板	CreateDashboard	DescribeDashboard
访问策略	CreateAccessPolicy	DescribeAccessPolicy
时间序列	BatchPutAssetPropertyValue	DescribeTimeSeries

对于`BatchPutAssetPropertyValue`，在创建资产模型和资产 Amazon IoT SiteWise 之前，您可以配置要将工业数据发送到的数据源。 Amazon IoT SiteWise 自动创建数据流以接收来自您的设备的原始数据流。有关更多信息，请参阅管理数据摄取。

使用以下操作可查看和管理支持标记的资源的标签：

- TagResource— 向资源添加标签，或更新现有标签的值。
- ListTagsForResource— 列出资源的标签。
- UntagResource— 从资源中移除标签。

随时在资源中添加或删除标签。要更新现有标签键的值，请将具有相同键和所需的新值的新标签添加到该资源。此操作会将旧值替换为新值。虽然可以将空字符串指定为标签值，但无法分配空值。

删除资源也会移除与其关联的所有标签。

将标签与 IAM 策略结合使用

在您的 IAM 策略中，使用资源标签来控制用户访问和权限。例如，策略可以支持用户仅创建带有特定标签的资源。策略还可以限制用户创建或修改具有特定标签的资源。

> **Note**
>
> 如果您使用标签来允许或拒绝用户对资源的访问，则应拒绝用户对相同资源添加或删除这些标签的能力。否则，用户会通过修改资源标签来绕过您的限制并获得资源的访问权限。

可以在策略语句的 `Condition` 元素（也称为 `Condition` 块）中使用以下条件上下文键和值。

`aws:ResourceTag/`*`tag-key`*`: `*`tag-value`*

允许或拒绝带特定标签的资源上的操作。

`aws:RequestTag/`*`tag-key`*`: `*`tag-value`*

要求在创建或修改可标记的资源时使用（或不使用）特定标签。

`aws:TagKeys: [`*`tag-key`*`, ...]`

要求在创建或修改可标记的资源时使用（或不使用）一组特定的标签键。

> **Note**
>
> IAM policy 中的条件上下文键和值仅适用于将可标记的资源作为必需参数的操作。例如，您可以为设置基于标签的条件访问权限。ListAssets您无法将基于标签的条件访问设置为开启，PutLoggingOptions因为请求中未引用任何可标记的资源。

有关更多信息，请参阅 IAM 用户指南中的使用 Amazon 资源标签控制资源访问权限和 IAM JSON 策略参考。

使用标签的 IAM 策略示例

- 根据标签查看 Amazon IoT SiteWise 资产

故障排除 Amazon IoT SiteWise

使用这些部分中的信息来排除和解决问题 Amazon IoT SiteWise。

主题

- 批量导入和导出操作故障排除
- 对 Amazon IoT SiteWise 门户进行故障排除
- 对 SiteWise 边缘网关进行故障排除
- 对 Amazon IoT SiteWise 规则操作进行故障排除

批量导入和导出操作故障排除

要处理和诊断传输任务期间产生的错误，请参阅 Amazon IoT TwinMaker GetMetadataTransferJobAPI：

1. 创建并运行传输任务后，调用 GetMetadataTransferJobAPI：

```
aws iottwinmaker get-metadata-transfer-job \
--metadata-transfer-job-id your_metadata_transfer_job_id \
--region us-east-1
```

2. 任务状态会变为以下状态之一：
   - COMPLETED
   - CANCELLED
   - ERROR
3. GetMetadataTransferJobAPI 返回一个 MetadataTransferJobProgress对象。
4. 该MetadataTransferJobProgress对象包含以下参数：
   - failedCount：表示在传输过程中失败的资产数量。
   - skippedCount：表示在传输过程中跳过的资产数量。
   - succeededCount：表示在传输过程中成功的资产数量。
   - totalCount：表示传输过程中涉及的资产总数。

5. 此外，该 API 调用还会返回一个 reportUrl 元素，其中包含一个预签名 URL。如果您的传输任务出现错误并需要调查，则您可以通过此 URL 下载完整的错误报告。

对 Amazon IoT SiteWise 门户进行故障排除

对 Amazon IoT SiteWise 门户网站的常见问题进行故障排除。

用户和管理员无法访问 Amazon IoT SiteWise 门户

如果用户或管理员无法访问您的 Amazon IoT SiteWise 门户，则您的附加的 Amazon Identity and Access Management (IAM) 策略可能会限制您的权限，从而阻止成功登录。

请参阅以下会导致登录失败的 IAM 策略示例：

> **Note**
>
> 任何包含 "Condition" 元素的附加 IAM 策略都会导致登录失败。

示例 1：此处的条件是 IP 受限，这将导致登录失败。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribePortal"
            ],
            "Resource": "*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "REPLACESAMPLEIP"
                    ]
                }
            }
        }
    ]
}
```

示例 2：此处的条件是包含标签，这将导致登录失败。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribePortal"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/project": "*"
                }
            }
        }
    ]
}
```

向门户添加用户或管理员时，请避免创建限制用户权限（例如受限 IP）的 IAM policy。任何具有受限权限的附加策略都将无法连接到 Amazon IoT SiteWise 门户。

对 SiteWise 边缘网关进行故障排除

通过浏览相关主题来解决常见的 Amazon IoT SiteWise Edge 网关问题。

您还可以查看 SiteWise Edge 网关报告的 CloudWatch 指标，以解决连接或数据流问题。有关更多信息，请参阅 Amazon IoT SiteWise 使用 Amazon CloudWatch 指标进行监控。

主题

- 配置和访问 SiteWise Edge 网关日志
- 排除 SiteWise Edge 网关问题
- 对 Amazon IoT SiteWise Edge 应用程序进行故障排除 Siemens Industrial Edge
- 疑难解答 Amazon IoT Greengrass 问题

配置和访问 SiteWise Edge 网关日志

在查看 SiteWise Edge 网关日志之前，必须将 SiteWise Edge 网关配置为将日志发送到 Amazon CloudWatch Logs 或将日志存储在本地文件系统上。

- 如果您想使用 CloudWatch Amazon Web Services Management Console 来查看 SiteWise Edge 网关的日志文件，请使用日志。有关更多信息，请参阅 使用 Amazon CloudWatch 日志。
- 如果您想使用命令行或本地软件查看 SiteWise Edge 网关的日志文件，请使用本地文件系统日志。有关更多信息，请参阅 使用服务登录 Amazon IoT SiteWise。

排除 SiteWise Edge 网关问题

使用以下信息对 SiteWise Edge 网关问题进行故障排除。

事务

- 无法将包部署到 SiteWise Edge 网关
- Amazon IoT SiteWise 不接收来自 OPC UA 服务器的数据
- 控制面板中未显示任何数据
- aws.iot 中显示 “找不到或加载主类”。 SiteWiseEdgePublisher 日志at /greengrass/v2/logs错误
- 我看见了 “SESSION_TAKEN_OVER” 或 “com.aws.greengrass.mqttclient”。 MqttClient: 无法通过 Spooler 发布消息，将重试。' ”
- 我看到了 'com.aws.greengrass.deployment。 IotJobsHelper: 未找到部署作业。' 或“Deployment result already reported. ”
- 尝试在 OPC UA 数据源的属性组中配置时间戳设置时，我看到 “SYNC_FAILED” 状态
- 不包括转换后的数据类型
- 信任商店问题
- 支持代理的安装问题

无法将包部署到 SiteWise Edge 网关

如果 n Amazon IoT Greengrass ucleus 组件 (`aws.greengrass.Nucleus`) 已过期，则可能无法将包部署到您的 SiteWise Edge 网关。您可以使用 Amazon IoT Greengrass V2 控制台升级 n Amazon IoT Greengrass ucleus 组件。

升级 n Amazon IoT Greengrass ucleus 组件（控制台）

1. 导航至 Amazon IoT Greengrass 控制台。
2. 在导航窗格的 Amazon IoT Greengrass 下，选择部署。
3. 在部署列表中，选择要修改的部署。
4. 选择修订。
5. 在指定目标页面，选择下一步。
6. 在选择组件页面上公共组件下的搜索框中，输入 **`aws.greengrass.Nucleus`**，然后选择 aws.greengrass.Nucleus。
7. 选择下一步。
8. 在配置组件页面上，选择下一步。
9. 在配置高级设置页面上，选择下一步。
10. 在 检查 页上，选择 部署。

Amazon IoT SiteWise 不接收来自 OPC UA 服务器的数据

如果您的 Amazon IoT SiteWise 资产未接收 OPC UA 服务器发送的数据，则可以搜索 SiteWise Edge 网关的日志来解决问题。查找包含以下消息的信息级别 `swPublisher` 日志。

```
Emitting diagnostic name=PublishError.SomeException
```

根据日志*`SomeException`*中的类型，使用以下异常类型和相应的问题对您的 SiteWise Edge 网关进行故障排除：

- ResourceNotFoundException— 您的 OPC UA 服务器发送的数据与任何资产的属性别名都不匹配。在以下两种情况下，可能会发生此异常：
  - 您的属性别名与 OPC UA 变量不完全匹配，包括您定义的任何源前缀。检查属性别名和源前缀是否正确。
  - 您尚未将 OPC UA 变量映射到资产属性。有关更多信息，请参阅 管理数据流 Amazon IoT SiteWise。
- InvalidRequestException— 您的 OPC UA 变量数据类型与您的资产属性数据类型不匹配。例如，如果 OPC UA 变量具有整数数据类型，相应的资产属性则必须是整数数据类型。双精度类型的资产属性不能接收 OPC UA 整数值。要解决此问题，请使用正确的数据类型定义新属性。
- TimestampOutOfRangeException— 您的 SiteWise Edge 网关正在发送超出可 Amazon IoT SiteWise 接受范围的数据。 Amazon IoT SiteWise 拒绝过去时间戳早于 7 天或将来 5 分钟以上

的任何数据点。如果您的 SiteWise Edge 网关断电或与 Amazon 云的连接中断，则可能需要清除 SiteWise Edge 网关的缓存。

- ThrottlingException或者 LimitExceededException— 您的请求超出了 Amazon IoT SiteWise 服务配额，例如提取的数据点速率或资产属性数据 API 操作的请求速率。检查您的配置是否未超过 Amazon IoT SiteWise 配额。

控制面板中未显示任何数据

如果您的控制面板中没有显示任何数据，则 SiteWise Edge 网关的发布器配置和 Dat a Sour ce 可能不同步。如果二者不同步，更新数据来源名称可能会加快从云端到边缘的同步，从而修复不同步错误。

要更新数据来源的名称，请按以下步骤操作：

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择连接到仪表板的 SiteWise Edge 网关。
4. 在数据来源下，选择编辑。
5. 选择一个新的源名称，然后选择保存以确认您的更改。
6. 确认数据来源表中的数据来源名称已更新，以验证您的更改。

aws.iot 中显示 “找不到或加载主类”。 SiteWiseEdgePublisher 日志at /greengrass/v2/logs错误

如果您看到此错误，则可能需要更新 SiteWise Edge 网关的 Java 版本。

- 从终端设备运行以下命令：

```
java -version
```

  您的 SiteWise Edge 网关所运行的 java 版本将显示在下面`OpenJDK Runtime Environment`。您会看到如下响应：

```
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 node)
```

如果你运行的是 Java 版本 11.0.20.8.1，则必须将 IoT P SiteWise ublisher 包更新到 2.4.1 或更高版本。只有 java 版本 11.0.20.8.1 受到影响，具有其他 Java 版本的环境可以继续使用旧版本的 IoT SiteWise Publisher 组件。有关更新组件包的更多信息，请参阅更改 SiteWise Edge 网关组件包的版本。

我看见了 “SESSION_TAKEN_OVER” 或 “com.aws.greengrass.mqttclient”。MqttClient: 无法通过 Spooler 发布消息，将重试。' ”

如果您在的日志`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`或包含错误的信息`/greengrass/v2/logs/greengrass.log`，则可能是您正在尝试对多台设备上的多个 SiteWise Edge 网关使用相同的配置文件。每个 SiteWise Edge 网关都需要一个唯一的配置文件才能连接到您的 Amazon 账户。

我看到了 'com.aws.greengrass.deployment。 IotJobsHelper: 未找到部署作业。' 或“Deployment result already reported. ”

如果您在 `/greengrass/v2/logs/greengrass.log` 的日志中看到 `com.aws.greengrass.deployment.IotJobsHelper: No deployment job found.` 或 `Deployment result already reported.`，则您可能正在尝试重复使用相同的配置文件。

这种情况有多种解决方案：

- 要重复使用此配置文件，请执行以下操作：
  1. 导航至 Amazon IoT SiteWise 控制台。
  2. 在导航窗格中，选择 Edge 网关。
  3. 选择要重复使用的 SiteWise Edge 网关。
  4. 选择更新选项卡。
  5. 选择其他发布者版本，然后选择部署。

按照 为 Siemens Industrial Edge 创建网关 中的步骤创建新的配置文件。

尝试在 OPC UA 数据源的属性组中配置时间戳设置时，我看到 “SYNC_FAILED” 状态

在 Amazon IoT SiteWise 更新 2.5.0 版本的 OPC UA 收集器组件时，我们引入了一个新的时间戳配置选项。 Amazon IoT Greengrass 您可以使用设备上的时间戳，也可以使用服务器上的时间戳。旧版本的 OPC UA 收集器组件不支持此选项，因此无法同步。

有两种方法可以解决数据源同步失败的状态。推荐的方法是将 IoT SiteWise OPC UA 收集器组件升级到 2.5.0 或更高版本。或者，如果您将时间戳设置为，则可以继续使用较旧的 OPC UA 收集器组件版本。`Source`要了解如何升级 IoT SiteWise OPC UA 收集器组件，请参阅更新 Amazon IoT SiteWise 组件的版本。我们建议使用所有组件的最新版本。

> **Note**
>
> 当数据源同步状态失败时，不会出现数据中断。源数据继续流入 Amazon IoT SiteWise。配置根本没有与部署中的 IoT SiteWise OPC UA 收集器组件同步。 Amazon IoT Greengrass V2

更改属性组的时间戳配置

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要编辑的网关。
4. 在数据源部分，选择同步失败状态的数据源，然后选择编辑。
5. 展开高级配置，然后展开群组设置。
6. 在时间戳中，选择来源。选择 S ourc e 会将该`timestampToReturn`属性从配置中删除。默认情况下，此设置允许从您的设备收集数据源时间戳，从而允许数据源与 IoT SiteWise OPC UA 收集器组件同步。
7. 选择保存。

不包括转换后的数据类型

如果您在将不支持的 OPC UA 数据类型转换为字符串时看到错误 Amazon IoT SiteWise，可能有以下几个原因：

- 您尝试转换的数据类型是复杂数据类型。不支持复杂数据类型。
- 将目标用于使用 Amazon S3 进行Amazon IoT SiteWise 缓冲时，完整字符串值会保留在推送到 Amazon S3 存储桶的文件中。稍后向中提取数据时 Amazon IoT SiteWise，长度超过 1024 字节的完整字符串值将被拒绝。

信任商店问题

如果您在 SiteWise Edge 中遇到与信任存储相关的问题，请考虑以下故障排除步骤：

- 验证 Amazon IoT Greengrass 根 CA 证书在相应的信任存储中是否存在且格式正确
- 确保 Java KeyStore 密码设置正确，并且 SiteWise Edge 组件可以访问
- 检查所有自定义证书（例如 HTTPS 代理）的格式是否正确（通常为 PEM）并正确导入到信任存储库中
- 确认信任存储库具有正确的文件权限并且 SiteWise Edge 进程可以访问
- 查看 SiteWise Edge 日志中是否存在任何与 SSL/TLS 相关的错误，这些错误可能表明存在信任存储问题
- 使用诸如验证信任存储库功能之类`openssl`的工具独立测试 SSL/TLS 连接

支持代理的安装问题

如果您在代理配置过程中遇到问题，请考虑以下故障排除步骤：

- 验证代理 URL 的格式是否正确且包含正确的架构（`http://`或`https://`）
- 如果任何代理凭证包含特殊字符，请确保它们都经过网址编码
- 确认无代理列表包含所有必需的本地地址和 Amazon 服务端点
- 对于 HTTPS 代理，请验证所提供的 CA 证书是否为 PEM 格式
- 查看安装日志，了解可能表明问题根源的特定错误消息
- 独立测试代理连接以确保其正常运行

对 Amazon IoT SiteWise Edge 应用程序进行故障排除 Siemens Industrial Edge

要对Siemens Industrial Edge设备上的 Amazon IoT SiteWise Edge 应用程序进行故障排除，您可以通过Siemens Industrial Edge Management或Siemens Industrial Edge设备 (IED) 门户访问该应用程序的日志。有关更多信息，请参阅 Siemens 文档中的下载日志。

我的数据未显示在 Amazon IoT SiteWise

- 确保您的 Databus 用户没有任何问题，并且 Databus_Configuration 的对勾图标为绿色，而非灰色。
- 您当前运行的 Siemens Industrial Edge Management 版本可能不包含 Secure Storage 功能。请升级您的 Siemens 操作系统版本。有关更多信息，请参阅 Siemens Secure Storage和 Amazon IoT SiteWise Edge 应用程序。

我在日志中看到 “缺少配置文件 AWS_REGION”。

如果您在 Siemens 日志中看到 `Config file missing AWS_REGION`，则该配置文件的 JSON 已损坏。您将需要创建一个新的配置文件。按照 为 Siemens Industrial Edge 创建网关 中的步骤创建新的配置文件。

我在 Edge 网关配置中看到 “不同步” 错误消息

部署完成后，如果您在Siemens Industrial Edge网关上看到`Out of sync`错误消息，则表示物联网 SiteWise 发布者组件与您的网关不同步。物联网 SiteWise 发布者组件在Siemens Industrial Edge网关的后台运行，以提供 MQTT 主题功能。我们将Siemens Industrial Edge网关升级为使用功能命名空间，`iotsitewise:publisher:3`而不是`iotsitewise:publisher:2`。您可以更新到最新版本的发布者来解决此问题。

升级到最新版本的 IoT SiteWise 发行商

1. 导航至 Amazon IoT SiteWise 控制台。
2. 在导航窗格中，选择 Edge 网关。
3. 选择要编辑的Siemens Industrial Edge网关。
4. 在 Edge 功能部分中，选择 View 软件版本。
5. 在 “发布者” 下拉菜单下选择最新版本的 IoT SiteWise 发行商。
6. 选择完成。

疑难解答 Amazon IoT Greengrass 问题

要查找在上配置或部署 SiteWise Edge 网关的许多问题的解决方案 Amazon IoT Greengrass，请参阅Amazon IoT Greengrass 开发人员指南 Amazon IoT Greengrass中的疑难解答。

对 Amazon IoT SiteWise 规则操作进行故障排除

要对中的 Amazon IoT SiteWise 规则操作进行故障排除 Amazon IoT Core，可以执行以下步骤之一：

- 配置 Amazon CloudWatch 日志
- 为规则配置重新发布错误操作

然后，将错误消息与本主题中的错误进行比较，以排查问题。

主题

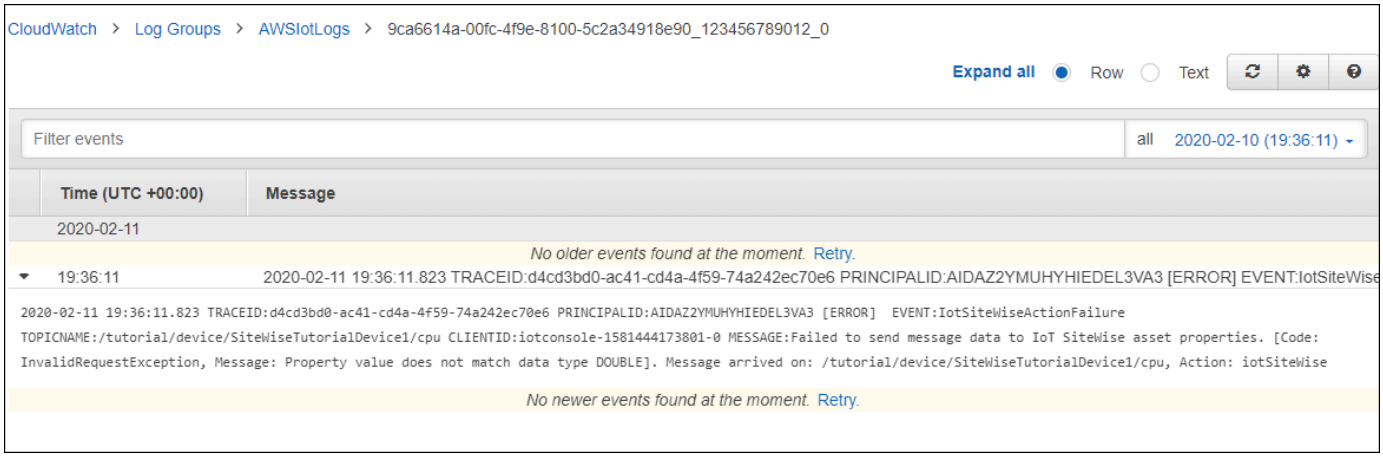
- 配置 Amazon IoT Core 日志
- 配置重新发布错误操作
- 排查规则问题
- 排查规则问题（Amazon IoT SiteWise）
- 对规则进行故障排除（DynamoDB）

配置 Amazon IoT Core 日志

您可以配置为 Amazon IoT 将不同级别的信息 CloudWatch 记录到 Logs 中。

配置和访问 CloudWatch 日志

1. 要为配置日志记录 Amazon IoT Core，请参阅《Amazon IoT 开发人员指南》中的使用 CloudWatch 日志进行监控。
2. 导航至 CloudWatch 控制台。
3. 在导航窗格中，选择 日志组。
4. 选择AWSIot日志组。
5. 选择最近的日志流。默认情况下，首先 CloudWatch 显示最新的日志流。
6. 选择日志条目以展开日志消息。您的日志条目可能类似于以下屏幕截图。



7. 将错误消息与本主题中的错误进行比较，以排查问题。

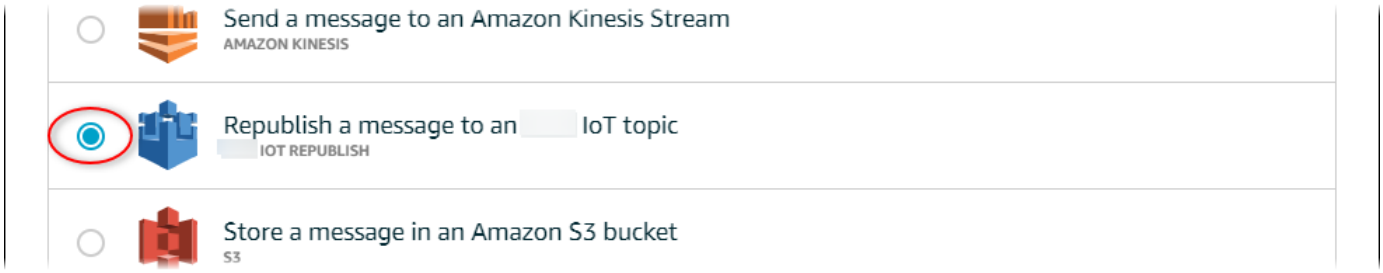
配置重新发布错误操作

您可以对规则的错误操作进行配置以处理错误消息。在该过程中，您将重新发布规则操作配置为错误操作，以在 MQTT 测试客户端中查看错误消息。

> **Note**
>
> 重新发布错误操作仅输出 ERROR 级别日志的等效内容。如果需要更多详细日志，则必须配置 CloudWatch 日志。

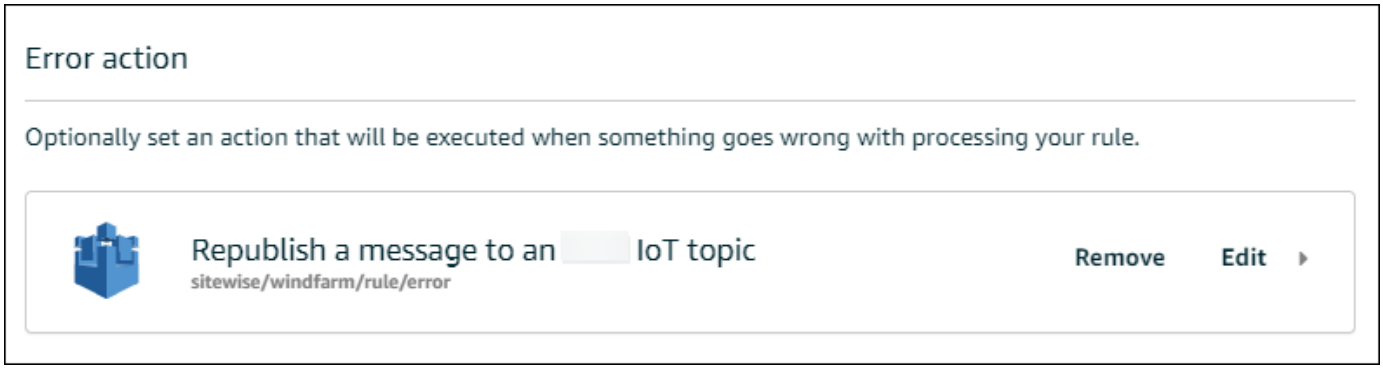
在规则中添加重新发布错误操作

1. 导航至 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择行动，然后选择规则。
3. 选择您的规则。
4. 在 错误操作 下面，选择 添加操作。
5. 选择 “向 Amazon IoT 主题重新发布消息”。



6. 选择页面底部的 配置操作。
7. 在主题中，输入唯一的主题（例如，**`sitewise/windfarm/rule/error`**）。 Amazon IoT Core 将重新发布此主题的错误消息。
8. 选择 “选择” 以授予执行错误操作的 Amazon IoT Core 访问权限。
9. 选择为规则创建的角色旁边的 选择。
10. 选择 更新角色 以在角色中添加其他权限。
11. 选择添加操作。

    您的规则的错误操作应类似于以下屏幕截图。



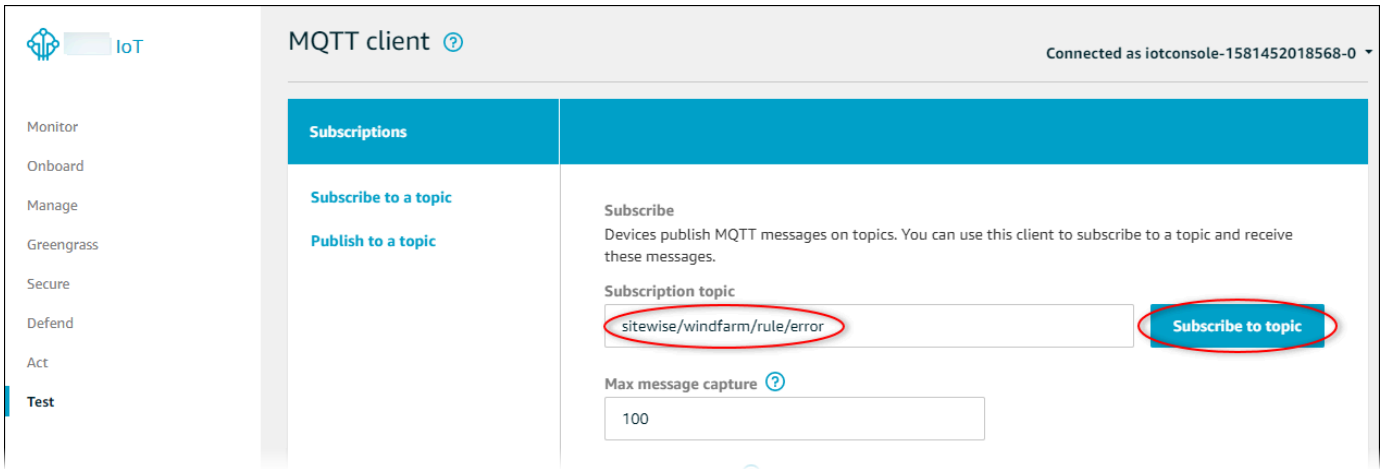
12. 选择主机左上角的返回箭头返回 Amazon IoT 主机主页。

在设置重新发布错误操作后，您可以在 Amazon IoT Core上的 MQTT 测试客户端中查看错误消息。

在以下过程中，您在 MQTT 测试客户端中订阅错误主题。在 MQTT 测试客户端中，您可以接收规则的错误消息以排查问题。

订阅错误操作主题

1. 导航至 Amazon IoT 控制台。
2. 在左侧导航页面中，选择 测试 以打开 MQTT 测试客户端。
3. 在 订阅主题 字段中，输入您之前配置的错误主题（例如，**`sitewise/windfarm/rule/error`**），然后选择 订阅主题。



4. 等待错误消息出现，然后在任何错误消息中展开 `failures` 数组。

   接下来，将错误消息与本主题中的错误进行比较，以排查问题。

排查规则问题

使用以下信息排查规则问题。

事务

- 错误：成员必须在当前时间戳之前 604800 秒和之后 300 秒范围内
- 错误：Property value does not match data type <type> (属性值与数据类型 <type> 不匹配)
- 错误：用户：<role-arn>无权在资源上执行：iotsitewise：BatchPutAssetPropertyValue
- 错误：iot.amazonaws.com 无法在资源上执行：sts：AssumeRole <role-arn>
- 信息：未发送任何请求。 PutAssetPropertyValueEntries 执行替换模板后为空。

错误：成员必须在当前时间戳之前 604800 秒和之后 300 秒范围内

与当前 Unix 纪元时间相比，您的时间戳早了超过 7 天或晚了超过 5 分钟。尝试以下操作：

- 检查您的时间戳是否采用 Unix 纪元 (UTC) 时间。如果您提供具有不同时区的时间戳，则会收到此错误。
- 检查您的时间戳是否以秒为单位。 Amazon IoT SiteWise 期望时间戳分为以秒为单位的时间（以 Unix 纪元时间为单位）和以纳秒为单位的偏移量。
- 检查您正在上传的数据的时间戳是否是在过去 7 天之前。

错误：Property value does not match data type <type> (属性值与数据类型 <type> 不匹配)

规则操作中的条目与目标资产属性具有不同的数据类型。例如，您的目标资产属性为 `DOUBLE`，但您选择的数据类型为 整数，或者您用 `integerValue` 传递了值。尝试以下操作：

- 如果您从 Amazon IoT 控制台配置规则，请检查是否为每个条目选择了正确的数据类型。
- 如果您通过 API 或 Amazon Command Line Interface (Amazon CLI) 配置规则，请检查您的`value`对象是否使用了正确的类型字段（`doubleValue`例如，`DOUBLE`属性字段）。

错误：用户：<role-arn>无权在资源上执行：iotsitewise：BatchPutAssetPropertyValue

未授权您的规则访问目标资产属性，或者目标资产属性不存在。尝试以下操作：

- 检查您的属性别名是否正确，以及您是否有具有给定属性别名的资产属性。有关更多信息，请参阅管理数据流 Amazon IoT SiteWise。
- 检查您的规则是否具有角色以及该角色是否允许目标资产属性具有 `iotsitewise:BatchPutAssetPropertyValue` 权限，例如通过目标资产的层次结构。有关更多信息，请参阅 授 Amazon IoT 予所需的访问权限。

错误：iot.amazonaws.com 无法在资源上执行：sts：AssumeRole <role-arn>

您的用户无权代入您在 Amazon Identity and Access Management (IAM) 中的规则中的角色。

检查是否对您的用户授予规则上角色的 `iam:PassRole` 权限。有关更多信息，请参阅 Amazon IoT 开发人员指南中的 传递角色权限。

信息：未发送任何请求。 PutAssetPropertyValueEntries 执行替换模板后为空。

> **Note**
>
> 此消息是 `INFO` 级别日志。

您的请求必须至少具有一个包含所有必需参数的条目。

检查规则的参数（包括替换模板）是否会导致非空值。替换模板无法访问规则查询语句中的 `AS` 子句定义的值。有关更多信息，请参阅 Amazon IoT 开发人员指南中的替换模板。

排查规则问题（Amazon IoT SiteWise）

如果 CPU 和内存使用率数据未按预期显示，请按照此过程中的 Amazon IoT SiteWise 步骤对规则进行故障排除。在该过程中，您将重新发布规则操作配置为错误操作，以在 MQTT 测试客户端中查看错误消息。您也可以配置日志记录以 CloudWatch 进行故障排除。有关更多信息，请参阅 对 Amazon IoT SiteWise 规则操作进行故障排除。

在规则中添加重新发布错误操作

1. 导航至 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择消息路由，然后选择规则。
3. 选择您以前创建的规则，然后选择编辑。

4. 在错误操作 -可选下，选择添加错误操作。
5. 选择 “向 Amazon IoT 主题重新发布消息”。
6. 在主题中，输入错误路径（例如**sitewise/rule/tutorial/error**）。 Amazon IoT Core 将重新发布此主题的错误消息。
7. 选择您之前创建的角色（例如，SiteWiseTutorialDeviceRuleRole）。
8. 选择更新。

在设置重新发布错误操作后，您可以在 Amazon IoT Core上的 MQTT 测试客户端中查看错误消息。

在以下过程中，您在 MQTT 测试客户端中订阅错误主题。

订阅错误操作主题

1. 导航至 Amazon IoT 控制台。
2. 在左侧导航页面中，选择 MQTT 测试客户端以打开 MQTT 测试客户端。
3. 在主题筛选器字段中，输入 **sitewise/rule/tutorial/error** 并选择订阅。

出现错误消息时，查看任何错误消息中的 `failures` 阵列以诊断问题。有关可能的问题以及如何解决这些问题的更多信息，请参阅对 Amazon IoT SiteWise 规则操作进行故障排除。

如果未显示错误，请检查是否启用了您的规则，以及您是否订阅了在重新发布错误操作中配置的相同主题。如果在执行此操作后仍未出现错误，请检查设备脚本是否正在运行并成功更新设备的影子。

> **Note**
>
> 您还可以订阅设备的影子更新主题，以查看您的 Amazon IoT SiteWise 操作解析的有效负载。为此，请订阅以下主题。
>
> ```
> $aws/things/+/shadow/update/accepted
> ```

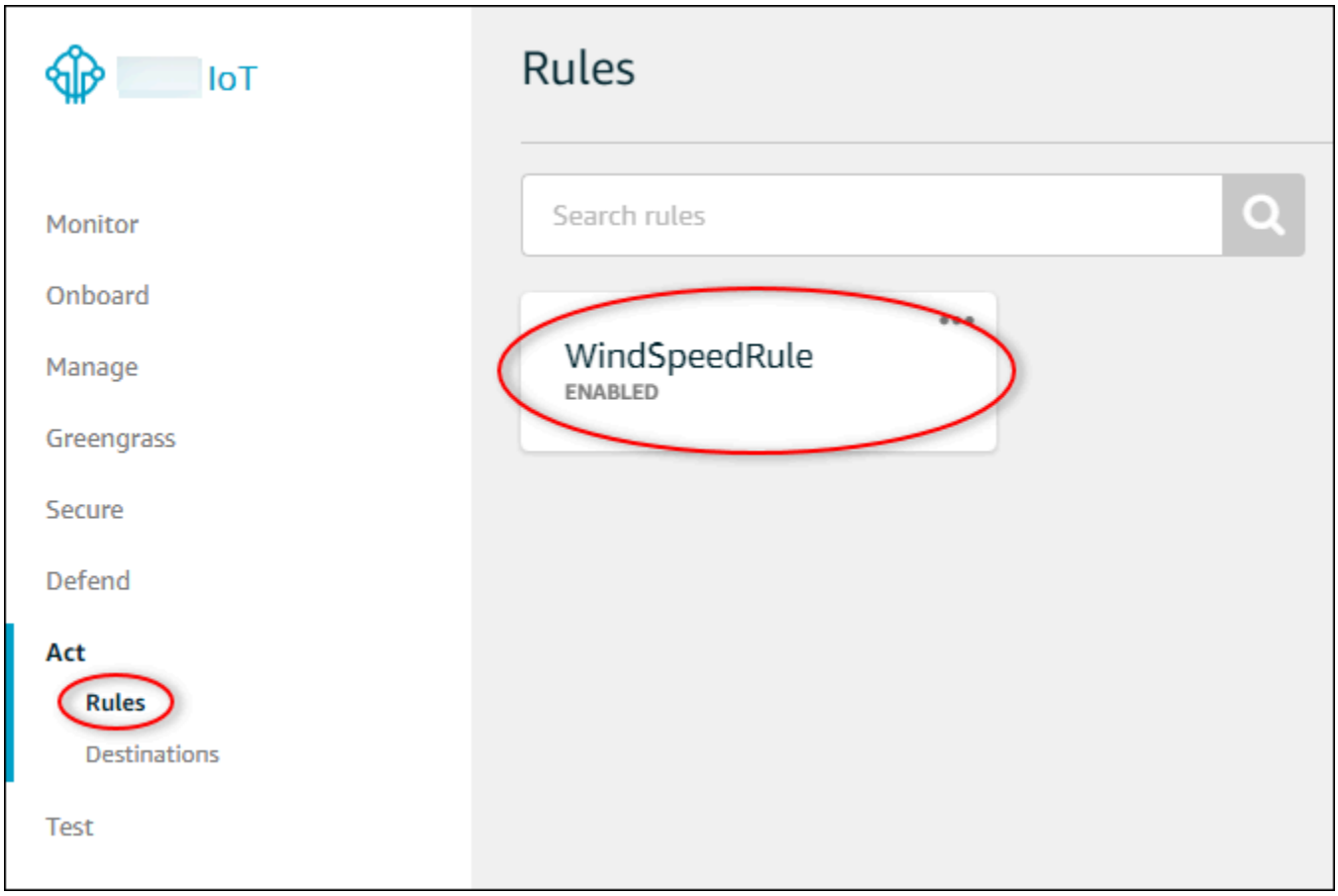
对规则进行故障排除（DynamoDB）

如果演示资产数据未按预期方式显示在 DynamoDB 表中，请按照该过程中的步骤排除规则故障。在该过程中，您将重新发布规则操作配置为错误操作，以在 MQTT 测试客户端中查看错误消息。您也可以

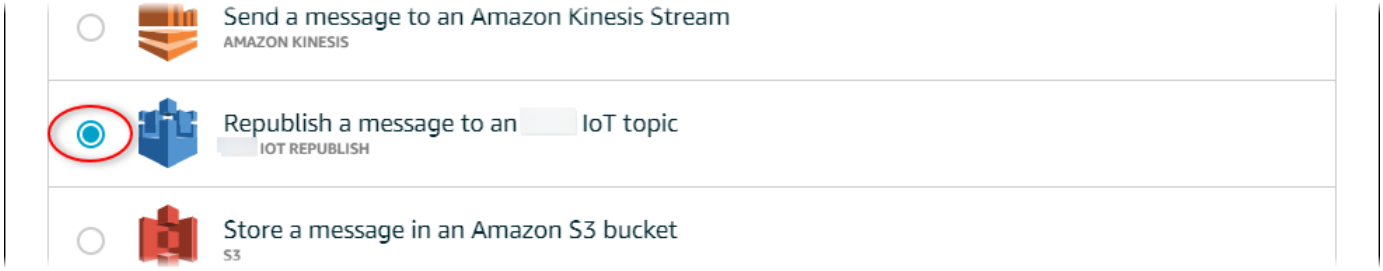
配置日志记录以 CloudWatch 进行故障排除。有关更多信息，请参阅《Amazon IoT 开发人员指南》中的使用 CloudWatch 日志进行监控。

在规则中添加重新发布错误操作

1. 导航至 Amazon IoT 控制台。
2. 在左侧导航窗格中，选择行动，然后选择规则。
3. 选择您以前创建的规则。

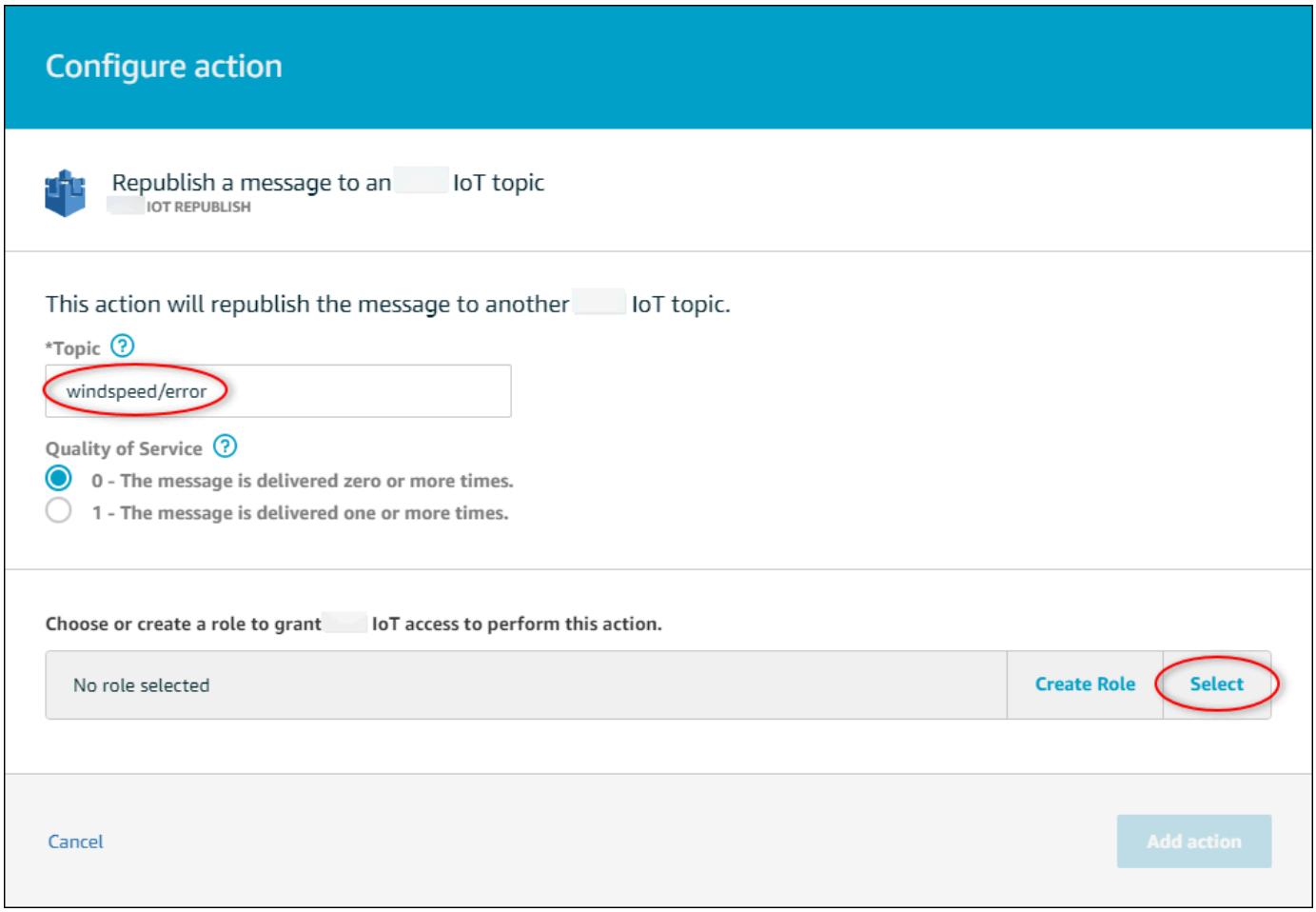


4. 在 错误操作 下面，选择 添加操作。
5. 选择 “向 Amazon IoT 主题重新发布消息”。

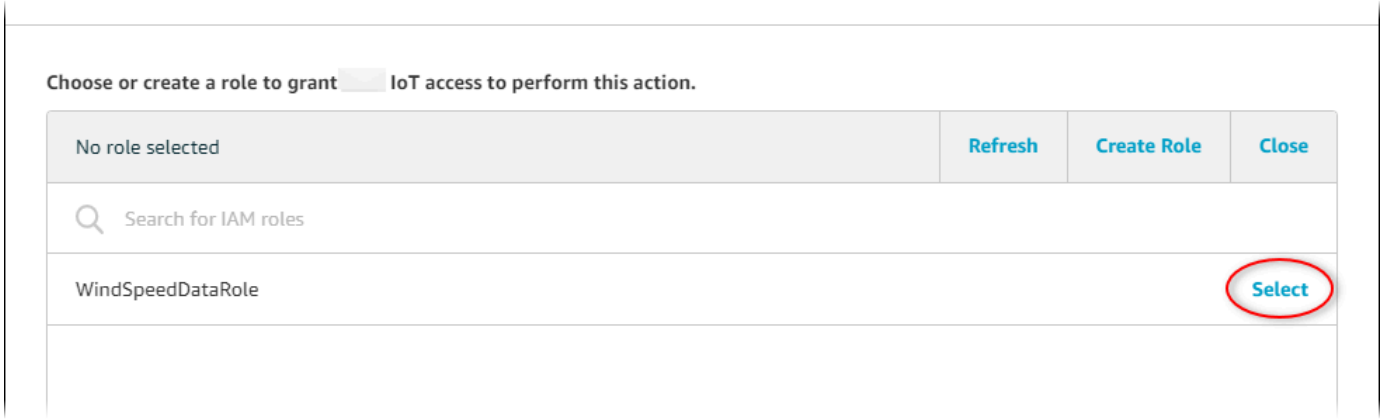


6. 选择页面底部的 配置操作。

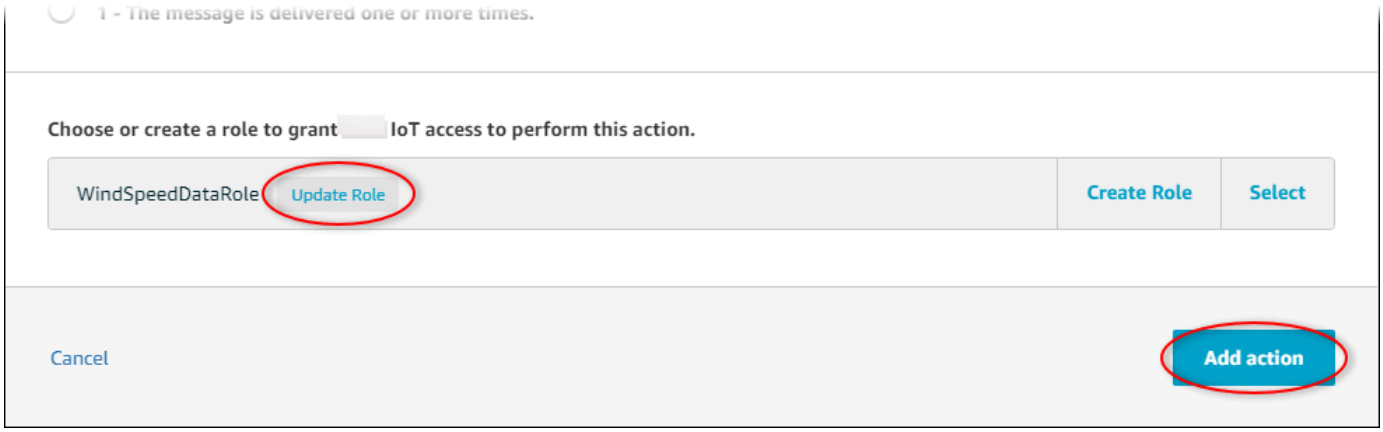
7. 在 Topic（主题）中输入 **`windspeed/error`**。 Amazon IoT Core 将重新发布此主题的错误消息。



8. 选择 Select 以授予使用您之前创建的角色执行错误操作的 C Amazon IoT ore 访问权限。
9. 选择您的角色旁边的 选择。



10. 选择 更新角色 以在角色中添加其他权限。



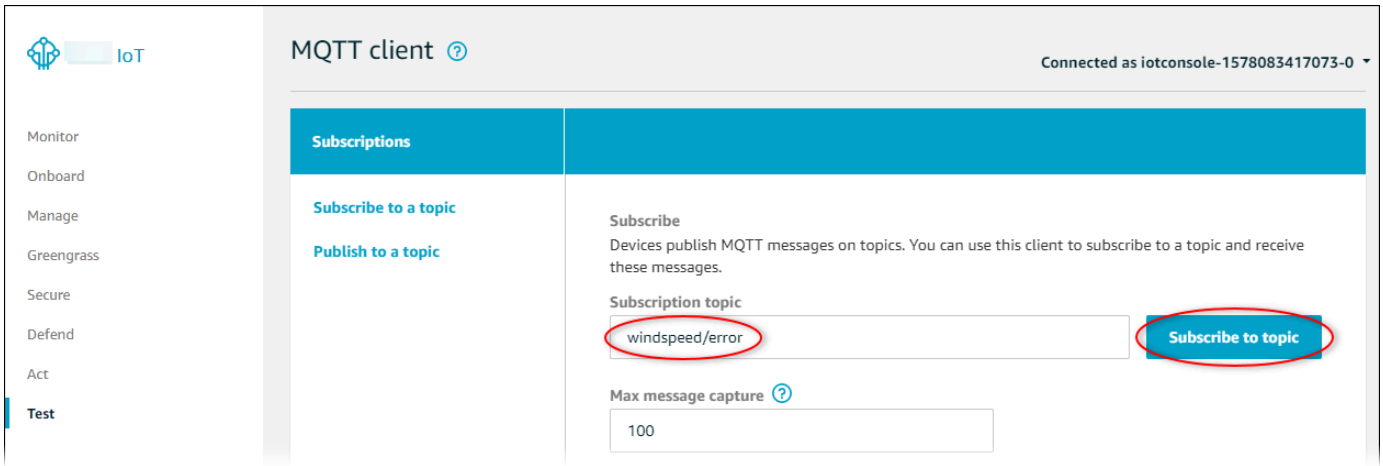
11. 选择 添加操作 以完成添加错误操作的过程。
12. 选择主机左上角的后退箭头返回 Amazon IoT 酷睿主机主页。

设置重新发布错误操作后，您可以在 Core 的 MQTT 测试客户端中 Amazon IoT 查看错误消息。

在以下过程中，您在 MQTT 测试客户端中订阅错误主题。

订阅错误操作主题

1. 在 Amazon IoT Core 控制台的左侧导航页中，选择 “测试”。
2. 在 订阅主题 字段中，输入 **`windspeed/error`** 并选择 订阅主题。



3. 观察显示的错误消息，并在错误消息中浏览 `failures` 数组以诊断以下常见问题：
   - 规则查询语句中的错别字
   - 角色权限不足

如果未显示错误，请检查是否启用了您的规则，以及您是否订阅了在重新发布错误操作中配置的相同主题。如果仍未显示错误，请检查您的演示风电场资产是否仍然存在，以及是否在风速属性上启用了通知。如果您的演示资产过期并从中消失 Amazon IoT SiteWise，则可以创建一个新的演示并更新规则查询语句以反映更新的资产模型和属性 IDs。

Amazon IoT SiteWise 终端节点和配额

以下各节描述了的终端节点和配额 Amazon IoT SiteWise。

主题

- Amazon IoT SiteWise 端点
- Amazon IoT SiteWise 配额

Amazon IoT SiteWise 端点

该 Amazon Web Services 一般参考 指南列出了的 Amazon IoT SiteWise 终端节点 Amazon Web Services 账户。有关更多信息，请参阅《Amazon Web Services 一般参考 指南》中的 Amazon IoT SiteWise endpoints and quotas。

Amazon IoT SiteWise 配额

下表描述了中的配额 Amazon IoT SiteWise。有关配额以及如何请求增加配额的更多信息，请参阅 Amazon Web Services 一般参考 中的 Amazon 服务限额。有关 Amazon IoT SiteWise 配额的更多信息，请参阅中的Amazon IoT SiteWise 服务配额Amazon Web Services 一般参考。

Amazon IoT SiteWise 资产和资产模型的配额

资源	描述	配额	可调整
每种资产模型 Amazon Web Services 区域的数量 Amazon Web Services 账户	您可以在 for 中创建的最大资产模型数量 Amazon Web Services 账户。 Amazon Web Services 区域	1000	是
每种资产模型中的资产数量	您可以为每种资产模型创建的最大资源数量。	10000	是

资源	描述	配额	可调整
每项父资产中的子资产数量	您可以与父项资产关联的最大子项资产数量。	2000	是
资产模型层次结构树的深度	资产模型的最大资产层次结构树深度。	30	是
每个资产模型中层次结构定义的数量	一个资产模型中可以拥有的最大层次结构定义数。	10	否
每个资产模型中根级别的属性数量	每种资产模型的最大数量。`assetModelProperties` 此计数不包括 `compositeModelProperties` 。此配额也适用于根据此资产模型创建的任何独有资产。	500	是
资产模型中的属性数量	`ASSET_MODEL` 或 `COMPONENT_MODEL` 类资产模型的最大属性数量。这个数字是通过组合根资产模型和任何包含的 component-model-based或内联的复合模型的属性来确定的。此配额也适用于根据此资产模型创建的任何独有资产。	5000	是

资源	描述	配额	可调整
每个复合模型中的属性数量	复合模型支持的最大属性数量。也是 COMPONENT_MODEL 类资产模型支持的最大属性数量。	100	是
资产模型中属性树的深度	例如，一个模型具有转换属性 C，该属性使用转换属性 B，后者又使用测量属性 A，模型的深度为 3。	10	否
每个层次结构树中的资产模型数量	您可以在单个层次结构树中包含的最大资产模型数量。	20	是
资产模型的直接依赖属性的数量	此配额限制了可直接依赖于单个属性的属性数量，如属性公式表达式中所定义。从属属属性的数量必须大于资产模型的直接依赖属性的数量。如果资产模型的直接依赖属性多于从属属性，则请求增加这两个配额。	20	是
资产模型中依赖属性的数量	此配额限制了可以直接或间接依赖于单个属性的属性数量，如属性公式表达式中所定义。	30	否

资源	描述	配额	可调整
资产模型中的复合模型数量	单个资产模型上可以拥有的最大复合模型数量。	50	可以
复合模型深度	每个资产模型（包括内联模型和复合模型）中 component-model-based复合模型树的最大深度。	2	可以
使用相同组件模型的独有资产模型的数量	具有至少一个直接引用 COMPONENT_MODEL 类型的特定资产模型的 component-model-based复合模型的唯一资产模型的最大数量。	20	是
属性公式表达式中属性变量的数量	例如，表达式 `avg(power) + max(temp)` 中有两个属性变量 `power` 和 `temp`。这也适用于转换计算结果。	10	否
属性公式表达式中的函数数量	例如，表达式 `avg(power) + max(temp)` 中有两个函数 `avg` 和 `max`。	10	否

Amazon IoT SiteWise 资产属性数据的配额

资源	描述	配额	可调整
资产属性数据 API 操作的请求速率	每个请求中每秒可以执行的最大	1000	是

资源	描述	配额	可调整
	资产属性数据 API 请求数 Amazon Web Services 账户。 Amazon Web Services 区域 此配额应用于 API 操作，例如 `GetAssetPropertyValue` 和 `BatchPutAssetPropertyValue` 。		
每个资产属性的每种数据质量每秒的数据点数	此配额适用于每个资产属性的每种数据质量具有相同时间戳 timestamp-quality-value（以秒为单位）的最大数量 (TQV) 数据点。对于每个资产属性，在任何给定的秒数内，最多可以存储此数量的高质量、不确定质量和低质量数据点。	10	否
每秒在每个资产属性中摄取的`BatchPutAssetPropertyValue` 条目数。 Amazon Web Services 区域 Amazon Web Services 账户	每个资产属性中`BatchPutAssetPropertyValue` 来自所有来源（包括 SiteWise Edge 网关、 Amazon IoT Core 规则和 API 调用）的最大条目数。	10	否

资源	描述	配额	可调整
提取的数据点的速率	每个数据点中每秒摄入的最大 timestamp-quality-value (TQV) 数据点数 Amazon Web Services 区域 。 Amazon Web Services 账户	5000	是
`BatchGetAssetPropertyAggregates` 的请求速率	每秒可以执行的最大`BatchGetAssetPropertyAggregates` 请求数 Amazon Web Services 账户。 Amazon Web Services 区域	200	是
`BatchGetAssetPropertyValue` 的请求速率	每秒可以执行的最大`BatchGetAssetPropertyValue` 请求数 Amazon Web Services 账户。 Amazon Web Services 区域	500	是
`BatchGetAssetPropertyValueHistory` 的请求速率	您每秒可以执行的最大`BatchGetAssetPropertyValueHistory` 请求数。	200	是

资源	描述	配额	可调整
每秒为每个资产属性摄取的BatchPutAssetPropertyValue 条目数。 Amazon Web Services 区域 Amazon Web Services 账户	此配额适用于所有BatchPutAssetPropertyValue 来源的每个资产属性中的条目，包括 SiteWise Edge 网关、 Amazon IoT Core 规则和 API 调用。	10	否
每个资产属性的GetAssetPropertyAggregates 请求率和BatchGetAssetPropertyAggregates 输入查询率	每个资产属性中每秒GetAssetPropertyAggregates 请求和BatchGetAssetPropertyAggregates 输入的最大总 Amazon Web Services 区域 次数 Amazon Web Services 账户。	50	不可以
每个资产属性的GetAssetPropertyValue 请求率和BatchGetAssetPropertyValue 输入查询率	每个资产属性中每秒GetAssetPropertyValue 请求和BatchGetAssetPropertyValue 输入的最大总数 Amazon Web Services 账户。 Amazon Web Services 区域	500	否

资源	描述	配额	可调整
每个资产属性的GetAssetPropertyValueHistory 请求率和BatchGetAssetPropertyValueHistory 输入查询率	每个资产属性中每秒GetAssetPropertyValueHistory 请求和BatchGetAssetPropertyValueHistory 输入的最大总数 Amazon Web Services 账户。 Amazon Web Services 区域	30	否
GetInterpolatedAssetPropertyValues 请求速率	每秒可以执行的最大GetInterpolatedAssetPropertyValues 请求数 Amazon Web Services 账户。 Amazon Web Services 区域	500	是
每个GetInterpolatedAssetPropertyValues 请求中的结果数	分页GetInterpolatedAssetPropertyValues 请求返回的最大结果数。	10	是

资源	描述	配额	可调整
从和中检索的数据点`GetAssetPropertyValueHistory` 的速率 `BatchGetAssetPropertyValueHistory`	跨和中每秒检索的数据点的最大字节速率 (MB/秒)。 Amazon Web Services 区域 Amazon Web Services 账户 `GetAssetPropertyValueHistory` `BatchGetAssetPropertyValueHistory` 为此配额评估的响应负载使用每个数据点的 Timestamp-Quality-Value (TQV) 字段，并将每个 API 请求的字节大小四舍五入到下一个 4KB 增量。 Timestamp-quality-value (TQV) 每秒检索的数据点因每种数据类型而异： • 整数 – 每秒高达 500 万 TQV • 双倍 – 每秒高达 400 万 TQV • 布尔值 – 每秒高达 600 万 TQV • 字符串 – 因每个字符串值大小而异。	100	是

SiteWise 边缘网关的配额

资源	描述	配额	可调整
每个 SiteWise Edge 网关 Amazon Web Services 区域 的数量 Amazon Web Services 账户	您可以在 for 中创建的最大 SiteWise Edge 网关数量 Amazon Web Services 账户。 Amazon Web Services 区域	100	是
SiteWise 边缘网关中的 OPC UA 源数量	您可以在 SiteWise Edge 网关中配置的最大 OPC UA 源数量。	100	否
SiteWise Edge 网关中的目的地总数	您可以在 SiteWise Edge 网关中配置的最大目的地数量。	100	否

的配额 Amazon IoT SiteWise Monitor

资源	描述	配额	可调整
每个传送门 Amazon Web Services 区域的数量 Amazon Web Services 账户	您可以在 for 中创建的最大 SiteWise 监控器入 Amazon Web Services 区域口数量 Amazon Web Services 账户。	100	是
门户网站中的项目数量	您可以在 Monito SiteWise r 门户中创建的最大项目数。	100	是
项目中的仪表板数量	您可以在 Monito SiteWise r 中在一个项目中创建的最大仪表板数量。	100	是

资源	描述	配额	可调整
项目中根资源的数量	您可以在 Monito SiteWise r 中向项目添加的最大顶级资源数量。	1	否
仪表板中的可视化次数	您可以在 M SiteWise onitor 中向仪表板添加的最大可视元素（例如图表、图形或表格）数量。	10	是
每个仪表板可视化中的指标数量	在 M SiteWise onitor 中仪表板上的单个可视化中可以显示的最大指标或数据点数。	5	是
每个仪表板可视化的阈值数量	您可以在 M SiteWise onitor 中为仪表板上的每个可视化设置的最大阈值级别数。	12	否

Amazon IoT SiteWise 批量导入和导出元数据的配额

资源	描述	配额	可调整
队列中元数据传输作业的数量	队列中 `PENDING` 元数据传输作业的最大数量。	10	是
元数据传输作业导入文件的大小	导入文件的最大大小（以 MB 表示）。	100	是
任务中的 Amazon IoT SiteWise 导入资源数量	单个任务中 Amazon IoT SiteWise 导入资源的最大数量。资源包括资产和资产模型。	5000	是

资源	描述	配额	可调整
任务中的 Amazon IoT SiteWise 导出资源数量	单个任务中 Amazon IoT SiteWise 导出资源的最大数量。资源包括资产和资产模型。	5000	是

Amazon IoT SiteWise 批量导入数据的配额

资源	描述	配额	可调整
正在运行的批量导入作业数	可以同时运行的批量导入任务的最大数量。	100	否
CSV 文件的大小	批量导入任务中的最大 CSV 文件大小（以 GB 为单位）。	10	否
未压缩的 Parquet 文件的大小	批量导入任务中未压缩的 parquet 文件的最大文件大小（以 MB 为单位）。	256 MB	否
用于缓冲摄取的 CSV 文件的大小	在批量导入任务中使用缓冲提取时的最大 CSV 文件大小（以 MB 为单位）。	256 MB	否
未压缩的 Parquet 行组的大小	未压缩的拼花地板行组的最大尺寸。	64 MB	否
镶木地板文件中唯一测量值的数量	镶木地板文件中唯一测量值的最大数量。	10000	否
用于缓冲摄取的过去时间戳和今天之间的天数	使用缓冲摄取时，从过去的时间戳到今天	30	是

资源	描述	配额	可调整
	的日期之间的最大天数。		
每`CreateBulkImportJobs` Amazon Web Services 区域 种请求费率 Amazon Web Services 账户		10	是
每 Amazon Web Services 区域 人对应`ListBulkImportJobs` 的请求费率 Amazon Web Services 账户		50	可以
每 Amazon Web Services 区域 人对应`DescribeBulkImportJobs` 的请求费率 Amazon Web Services 账户		50	可以

Amazon IoT SiteWise 助手 API 限流配额

Amazon IoT SiteWise 助手 API 限流限制的配额

资源	描述	配额	可调整
`InvokeAssistant` 操作请求速率	每分钟可以向 Amazon IoT SiteWise `InvokeAssistant` API 进行的最大交易数 (TPM)。 Amazon	10	否

资源	描述	配额	可调整
	Web Services 账户 TPM 限制适用于所有支持的区域，并且在某些地区可以调整。		

异常检测的配额

异常检测的配额由 Amazon IoT SiteWise 和 Amazon Lookout for Equipment 共享。有关更多信息，请参阅 Quotas for using Lookout for Equipment。

《Amazon IoT SiteWise 用户指南》的文档历史记录

下表描述了此版本的文档 Amazon IoT SiteWise。

- API 版本：2019-12-02

变更	说明	日期
使用用户名和密码将外部应用程序连接到 EMQX 代理	增加了使用用户名和密码将外部应用程序连接到 Amazon IoT Greengrass EMQX 代理的支持，该代理在创建支持 MQTT 的新的 V3 网关时默认部署。	2025 年 5 月 1 日
Support 在 Edge 上支持启用 MQTT 的 V3 网关 SiteWise	添加了新功能并删除了已弃用的内容 • 增加了对启用 MQTT 的 V3 网关的支持。使用路径筛选器订阅 MQTT 主题增强了目标配置，包括直接向 Amazon S3 提取实时数据 Amazon IoT SiteWise 或使用 Amazon S3 进行缓冲数据摄取。 • 发布了 IoT SiteWise OPC UA 收集器的 3.0.0 版和适用的 IoT SiteWise 发布者组件 4.0.0 版。 Amazon IoT Greengrass V2 • 将先前版本的自托管 SiteWise Edge 网关重命名为 Classic Streams，即 V2 网关。 • 删除了 SiteWise Edge 文档 Amazon IoT Greengrass V1	2025年2月26日

	中对的引用，因为不再支持将其与一起使用 Amazon IoT SiteWise。	
Support Amazon IoT SiteWise Assistant	增加了对 Amazon IoT SiteWise 助手的支持，这是一款由人工智能驱动的生成式助手。	2024 年 11 月 18 日
为 Edge 添加了可配置的 SiteWise 会话超时 APIs	添加了可配置的会话超时设置，以管理 Amazon OpsHub 和 SiteWise Edge APIs 的闲置时间。	2024 年 10 月 31 日
为 SiteWise Edge 添加了可配置的代理设置 APIs	添加了管理信任存储库信息以启用 SiteWise Edge 网关的 HTTPS 代理支持。	2024 年 10 月 31 日
在 Edge 上启用 CORS SiteWise APIs	为 SiteWise Edge 添加了 CORS 支持 APIs ，以实现安全的跨域 Web 应用程序访问。	2024 年 9 月 30 日
Support CloudRail 以及 Litmus Edge 合作伙伴数据源	增加了对两者的支持 CloudRail 以及 Litmus Edge 作为合作伙伴数据源。	2024 年 9 月 5 日
在西门子工业 SiteWise 边缘上运行 Edge 的正式上市	Amazon IoT SiteWise 现在支持在西门子工业 SiteWise 边缘设备上运行 Edge 的正式上市。	2024 年 7 月 24 日
增加了对 OPC UA 数据来源上的时间戳配置的支持	Amazon IoT SiteWise 现在支持 OPC UA 数据源的时间戳配置。	2024 年 7 月 24 日
增加了对 OPC UA 数据来源上的数据类型转换的支持	Amazon IoT SiteWise 现在支持对不支持的 OPC UA 数据类型进行数据类型转换。	2024 年 7 月 24 日

增加了对在西门子工业 SiteWise 边缘上运行 Edge 预览版的支持	Amazon IoT SiteWise 现在支持在西门子工业 SiteWise 边缘设备上运行 Edge 预览。	2023 年 11 月 26 日
增加了对温层存储的支持	Amazon IoT SiteWise 现在支持温存储，这是一种完全托管的存储层，可让客户轻松安全地存储和访问工业数据。	2023 年 11 月 15 日
增加了对用户定义的唯一标识符的支持	Amazon IoT SiteWise 现在支持对资产、资产模型、属性和层次结构使用用户定义的唯一标识符。	2023 年 11 月 15 日
增加了对工业资产多变量异常检测的支持	Amazon IoT SiteWise 现在，通过将历史和实时设备数据与 Amazon Lookout for Equipment 集成，支持对工业资产进行多变异常检测。	2023 年 11 月 15 日
增加了在 Amazon IoT SiteWise中对时间序列数据进行经济高效且可扩展的摄取的支持	Amazon IoT SiteWise 现在支持以经济高效且可扩展的方式摄取分析用例所需的时间序列数据。	2023 年 11 月 15 日
增加了对批量导入、导出和更新的支持	Amazon IoT SiteWise 现在支持批量导入、导出和更新工业设备元数据。	2023 年 11 月 15 日
增加了对资产模型组件的支持	Amazon IoT SiteWise 现在支持资产模型组件，以帮助工业客户创建可重复使用的组件。	2023 年 11 月 15 日
增加了对 IoT 控制面板应用程序的支持	Amazon IoT SiteWise 现在支持开源仪表板应用程序，您可以在其中可视化操作数据并与之交互。	2023 年 11 月 15 日

更新了的服务相关角色 Amazon IoT SiteWise	Amazon IoT SiteWise 具有新的服务相关角色，可以对 Amazon IoT TwinMaker 数据库运行元数据搜索查询。	2023 年 11 月 6 日
更新了 Amazon IoT SiteWise 数据流资源的标记	添加了对标记数据流资源的支持。	2022 年 8 月 18 日
更新了 SiteWise 边缘网关	现在，您可以配置发布者，以控制从边缘发送到云的数据以及发送到云的顺序。	2022 年 1 月 12 日
更新了 Amazon IoT SiteWise 演示	现在，您可以使用该演示来创建 SiteWise 监控器门户。	2022 年 1 月 10 日
更新了存储管理	现在可以定义保留期，以控制数据在热层中的保留时间。	2021 年 11 月 29 日
添加了对数据流管理的支持	现在，您可以在创建资产模型和资产 Amazon IoT SiteWise 之前提取数据。	2021 年 11 月 24 日
区域启动	Amazon IoT SiteWise 在 Amazon GovCloud（美国西部）推出。	2021 年 9 月 29 日
自定义指标时间间隔	在指标中添加了对自定义时间间隔和偏移量的支持。	2021 年 8 月 3 日
Amazon IoT SiteWise 在边缘使用	边缘处理功能现已正式发布。	2021 年 7 月 29 日
将数据导出到 Amazon S3	Amazon IoT SiteWise 现在可以将数据导出到 Amazon S3。	2021 年 7 月 27 日
VPC 端点 (Amazon PrivateLink)	用于控制面板 API 操作的接口 VPC 端点现已正式发布。	2021 年 7 月 15 日

转换	转换现在可以输入多个资产属性变量。	2021 年 7 月 8 日
更新了 timestamp() 函数	在转换中，现在可以提供变量作为 `timestamp()` 函数的参数。	2021 年 6 月 16 日
警报正式发布	警报功能现已正式发布。	2021 年 5 月 27 日
Modbus-TCP 协议适配器版本 2 发布	Modbus-TCP 协议适配器连接器版本 2 现已推出。此版本增加了对 ASCII 和 ISO8859 编码源字符串的支持。 UTF8	2021 年 5 月 24 日
更新了服务限额	为 GetInterpolatedAssetPropertyValuesAPI 添加了以下配额：`GetInterpolatedAssetPropertyValues` 请求率、每个`GetInterpolatedAssetPropertyValues` 请求的结果数以及从过去开始日期到今天之间的天数`GetInterpolatedAssetPropertyValues` 。	2021 年 4 月 29 日
更新了公式表达式	添加了以下运算符和函数： • 添加了以下运算符：`<`、`>`、`<=`、`>=`、`==`、`!=`、`!`、`and`、`or` 和 `not`。 • 添加了以下比较功能：`neq(x, y)`。 • 添加了以下字符串函数：`join()`、`format()` 和 `f''`。	2021 年 4 月 22 日

VPC 端点 (Amazon PrivateLink)	添加了有关如何 APIs 通过创建接口 VPC 终端节点在虚拟私有云 (VPC) 和 Amazon IoT SiteWise 控制平面之间建立私有连接的信息。	2021 年 3 月 16 日
IAM 联合身份验证	您的 M SiteWise onitor 门户管理员和用户现在可以使用 IAM 证书登录分配给他们的门户。	2021 年 3 月 16 日
区域启动	Amazon IoT SiteWise 在中国（北京）推出。	2021 年 2 月 3 日
物联网 SiteWise 连接器版本 10 已发布	物联网 SiteWise 连接器的版本 10 已推出。此版本对 `StreamManager` 进行了配置，以改进源连接丢失并重新建立时的处理。当没有 `SourceTimestamp` 可用时，此版本还接受带有 `ServerTimestamp` 的 OPC UA 值。	2021 年 1 月 22 日
日期和时间函数	Amazon IoT SiteWise 现在支持日期和时间功能。	2021 年 1 月 21 日
函数语法	您现在可以对函数使用统一函数调用语法 (UFCS)。 Amazon IoT SiteWise	2021 年 1 月 11 日
与 Grafana 集成	添加了有关如何在 Grafana 仪 Amazon IoT SiteWise 表板中可视化数据的信息。	2020 年 12 月 15 日
Amazon IoT SiteWise 现在支持客户管理 CMKs。	Amazon IoT SiteWise 现在支持通过客户管理进行加密 CMKs。	2020 年 11 月 24 日

物联网 SiteWise 连接器版本 8 已发布	物联网 SiteWise 连接器的版本 8 现已推出。此版本改进了连接器遇到间歇性网络连接时的稳定性。	2020 年 11 月 19 日
在公式表达式中使用字符串和条件语句	添加了有关如何在转换和指标的公式表达式中使用字符串和条件函数的信息。	2020 年 11 月 16 日
使用 Amazon IoT Greengrass 流管理器摄取数据	添加了有关如何使用 Amazon IoT Greengrass 边缘设备从本地数据源提取大量 IoT 数据的信息。	2020 年 9 月 16 日
VPC 端点 (Amazon PrivateLink)	添加了有关如何 APIs 通过创建接口 VPC 终端节点在您的虚拟私有云 (VPC) 和 Amazon IoT SiteWise 数据之间建立私有连接的信息。	2020 年 9 月 4 日
物联网 SiteWise 连接器版本 7 已发布	物联网 SiteWise 连接器版本 7现已推出。此版本修复了 E SiteWise dge 网关指标的问题。	2020 年 8 月 14 日
通过 Amazon IoT SiteWise 控制台创建 IAM 身份中心用户	添加了有关如何在 Amazon IoT SiteWise 控制台中创建 IAM 身份中心用户的信息。现在，您可以在将用户分配到新门户或现有门户时创建 IAM Identity Center 用户。更新了可视化和共享风电场数据教程以使用此功能。此更改减少了教程中的步骤数。	2020 年 8 月 4 日

改进了 SiteWise Edge 网关故障	添加了有关如何对 SiteWise Edge 网关进行故障排除以及如何导出源的 OPC UA 客户端证书的其他信息。	2020 年 6 月 18 日
控制台任务文档	添加了用于工业资产建模、查询资产属性数据和与其他服务交互的控制台任务文档。您可以按照以下说明完成 Amazon IoT SiteWise 控制台中的任务。	2020 年 6 月 11 日
“分析导出的数据”教程	添加了一个教程，您可以按照该教程学习如何使用 Amazon Athena 分析使用导出功能模板导出到 Amazon S3 的资产数据。 Amazon CloudFormation	2020 年 5 月 27 日
改进了对公式表达式的使用	添加了有关 Amazon IoT SiteWise 公式属性行为的详细信息，并添加了如何计算筛选数据点的示例。	2020 年 5 月 18 日
物联网 SiteWise 连接器版本 6 已发布	物联网 SiteWise 连接器的版本 6 已推出。此版本增加了对 CloudWatch 指标和自动发现新 OPC UA 标签的支持。这意味着，当 OPC UA 来源的标签发生变化时，您无需重新启动 SiteWise Edge 网关。此版本的连接器需要流管理器和 Amazon IoT Greengrass 核心软件 v1.10.0 或更高版本。	2020 年 4 月 29 日

Amazon IoT SiteWise 功能发布	Amazon IoT SiteWise 功能发布。现在，您可以使用 API 管理 SiteWise Edge 网关、向门户添加徽标、查看 SiteWise Edge 网关指标等。 • 添加了将数据导出到 Amazon S3 部分，其中包含可用于将新数据值导出到 Amazon S3 存储桶的 Amazon CloudFormation 模板。 • 添加了 SiteWise Edge 网关指标部分，描述了 SiteWise Edge 网关发布的 CloudWatch 指标。 • 添加了在 Amazon 上配置 SiteWise Edge 网关 EC2 部分，其中包含可用于快速配置 Amazon EC2 实例上的 SiteWise Edge 网关依赖项的 Amazon CloudFormation 模板。 • 添加了门户服务角色部分，该部分描述了 SiteWise 监控器门户的新权限功能。 • 针对门户服务角色和门户徽标更新了门户文档。 • 添加了为Amazon IoT SiteWise 资源添加标签部分。 • 针对新的控制面板定义结构更新了创建控制面板（CLI）部分。 • 增加了安全性部分。	2020 年 4 月 29 日

正在从中提取数据 Amazon IoT Events	添加了有关如何从事件发生 Amazon IoT Events 时提取数据的信息。	2020 年 4 月 20 日
在 Monitor 教程中可 SiteWise 视化和共享风电场数据	添加了一个教程，您可以按照该教程学习如何使用该教程 Amazon IoT SiteWise Monitor 来可视化和共享资产数据。	2020 年 3 月 12 日
Amazon IoT SiteWise 概念	添加了 Amazon IoT SiteWise 概念词汇表，可用于了解该服务及其常用术语。	2020 年 3 月 5 日
已删除 Amazon IoT Greengrass 安装说明	已从《 Amazon IoT SiteWise 用户指南》中删除 Amazon IoT Greengrass 核心软件安装说明。《Amazon IoT Greengrass 开发者指南》提供了设备设置脚本以及在 Amazon EC2 和 Docker 等其他平台 Amazon IoT Greengrass 上进行设置的说明。	2020 年 2 月 14 日
使用规则改进了数据采集功能 Amazon IoT Core	添加了有关如何使用规则操作以及如何对 Amazon IoT SiteWise 规则操作进行故障排除的详细信息，您可以使用该操作从 MQTT 消息中提取数据。 Amazon IoT Core	2020 年 2 月 14 日
物联网 SiteWise 连接器版本 5 已发布	物联网 SiteWise 连接器的版本 5 现已推出。此版本修复了与 Amazon IoT Greengrass 核心软件 v1.9.4 的兼容性问题。	2020 年 2 月 12 日

物联网 SiteWise 连接器第 4 版发布	物联网 SiteWise 连接器的版本 4 现已推出。此版本修复了 OPC UA 服务器重新连接的问题。	2020 年 2 月 7 日
新编了工业资产建模	将“更新资产和模型”部分重新组合为“工业资产建模”中的多个主题。 • 资产和模型状态 • 管理数据流 Amazon IoT SiteWise • 更新属性值 • 关联资产和取消关联资产 • 更新资产和模型 • 删除中的资源和模型 Amazon IoT SiteWise	2020 年 2 月 4 日
从事物中提取数据教程 Amazon IoT	添加了一个教程，您可以按照该教程来学习如何配置 Amazon IoT SiteWise 规则操作以从新的或现有的设备队列中提取数据 Amazon IoT 。	2020 年 2 月 4 日
重构了从中检索数据的结构 Amazon IoT SiteWise	将 “检索数据” 部分重组为两个顶级部分：查询资产属性值和聚合以及与其他服务交互。Amazon	2020 年 1 月 21 日
将属性值更新发布到 Amazon DynamoDB 教程	添加了一个教程，您可以遵循该教程了解如何使用属性值通知在 DynamoDB 中存储资产数据。	2020 年 1 月 8 日

使用公式表达式	添加了公式表达式引用，以整理可用于转换属性和指标属性中的常量和函数。将资产属性重新组合为每种属性类型的单独主题。	2020 年 1 月 7 日
使用 OPC UA 节点筛选条件	添加了有关在添加 Edge 网关源时如何使用 OPC UA 节点筛选器来提高 SiteWise E SiteWise dge 网关性能的信息。	2020 年 1 月 3 日
升级连接器	添加了有关在新连接器版本发布时如何升级 SiteWise Edge 网关的信息。	2019 年 12 月 30 日
物联网 SiteWise 连接器第 3 版发布	物联网 SiteWise 连接器的版本 3 现已推出。此版本消除了 iot:* 权限要求。	2019 年 12 月 17 日
物联网 SiteWise 连接器第 2 版发布	物联网 SiteWise 连接器的版本 2 现已推出。此版本增加了对多个 OPC UA 秘密资源的支持。	2019 年 12 月 10 日
创建控制面板 (Amazon CLI)	在 Amazon IoT SiteWise Monitor 使用中添加了有关如何创建仪表板的信息 Amazon CLI。	2019 年 12 月 6 日

Amazon IoT SiteWise 版本 2 已发布	已发布版本 2 的预览版 Amazon IoT SiteWise。现在，您可以通过 OPC UA、MQTT 和 HTTP 提取数据，在资产层次结构中对数据进行建模，并使用 Monitor 对数据进行可视化。 SiteWise • 针对资产、资产模型和资产层次结构的更改重新编写了资产建模部分。 • 更新了数据提取部分，增加了 Amazon IoT Greengrass 连接器步骤和非网关数据摄取部分。 • 添加了该Amazon IoT SiteWise Monitor部分和单独的应用程序指南，其中显示了如何使用 Monito SiteWise r Web 应用程序。 • 添加了从中查询数据 Amazon IoT SiteWise和与其他 Amazon 服务互动部分。 • 重新编写了入门部分，以便与更新的演示体验相符。	2019 年 12 月 2 日
Amazon IoT SiteWise 版本 1 已发布	已发布版本 1 的初始预览版 Amazon IoT SiteWise。	2019 年 2 月 25 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。