



사용자 가이드

AWS IoT SiteWise



AWS IoT SiteWise: 사용자 가이드

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

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

AWS IoT SiteWise란 무엇인가요?	1
AWS IoT SiteWise 작동 방식	2
산업 데이터 수집	2
수집된 데이터를 컨텍스트화하기 위한 자산 모델링	3
쿼리, 경보 및 예측을 사용하여 분석	4
작업 시각화	4
데이터 저장	4
다른 서비스와 통합	5
에 대한 사용 사례 AWS IoT SiteWise	5
제조	5
식음료	5
에너지/공공 설비	6
AWS SDKs 작업	6
개념	7
시작	13
요구 사항	13
AWS 계정 설정	14
에 가입 AWS 계정	14
관리자 액세스 권한이 있는 사용자 생성	14
빠른 시작 데모 사용	16
AWS IoT SiteWise 데모 생성	16
AWS IoT SiteWise 데모 삭제	18
자습서	19
OEE 계산	19
사전 조건	19
OEE 계산 방법	20
데이터 수집	22
사전 조건	23
1단계: AWS IoT 정책 생성	23
2단계: AWS IoT 사물 생성	26
3단계: 디바이스 자산 모델 생성	27
4단계: 디바이스 플릿 자산 모델 생성	29
5단계: 디바이스 자산 생성 및 구성	30
6단계: 디바이스 플릿 자산 생성 및 구성	31

7단계: AWS IoT 코어에서 디바이스 자산으로 데이터를 전송하는 규칙 생성	32
8단계: 디바이스 클라이언트 스크립트 실행	35
9단계: 자습서 완료 후 리소스 정리	42
SiteWise Monitor에서 데이터 시각화 및 공유	43
사전 조건	44
1단계: SiteWise Monitor에서 포털 생성	45
2단계: 포털에 로그인	49
3단계: 풍력 발전소 프로젝트 생성	50
4단계: 풍력 발전소 데이터를 시각화하는 대시보드 생성	53
5단계: 포털 살펴보기	60
6단계: 자습서 완료 후 리소스 정리	61
Amazon DynamoDB에 게시	63
사전 조건	64
1단계: 속성 값 업데이트를 게시 AWS IoT SiteWise 하도록 구성	64
2단계: AWS IoT Core에서 규칙 생성	67
3단계: DynamoDB 테이블 생성	69
4단계: DynamoDB 규칙 작업 구성	71
5단계: DynamoDB에서 데이터 탐색	72
6단계: 자습서 완료 후 리소스 정리	73
에 데이터 수집 AWS IoT SiteWise	77
데이터 스트림 관리	77
권한 및 설정 구성	78
데이터 스트림을 자산 속성에 연결	80
자산 속성에서 데이터 스트림 연결 해제	81
데이터 스트림 삭제	82
자산 속성 별칭 업데이트	83
일반적인 시나리오	84
AWS IoT SiteWise APIs 사용하여 데이터 수집	86
BatchPutAssetPropertyValue API	86
CreateBulkImportJob API	89
AWS IoT Core 규칙 사용	97
필요한 액세스 권한 부여	97
규칙 작업 구성	98
의 Basic Ingest를 통한 비용 절감	106
AWS IoT Events 작업 사용	107
AWS IoT Greengrass 스트림 관리자 사용	108

SiteWise Edge 게이트웨이 사용	109
게이트웨이 주요 개념	109
SiteWise Edge 구현의 이점	110
게이트웨이 자체 호스팅	110
요구 사항	112
게이트웨이 생성	116
게이트웨이 소프트웨어 설치	119
MQTT 지원, V3 게이트웨이	122
클래식 스트림, V2 게이트웨이	176
데이터 소스 추가	189
SiteWise Edge 구성 요소	229
자산 필터링	231
프록시 지원 및 트러스트 스토어	232
API 사용	238
Siemens Industrial Edge에서 게이트웨이 호스팅	255
보안	256
Siemens Secure Storage 및 AWS IoT SiteWise Edge 애플리케이션	256
Siemens Industrial Edge 디바이스의 대상	257
미리 보기 애플리케이션에서 마이그레이션	258
문제 해결	259
AWS IoT SiteWise 엣지 애플리케이션 변경 로그	259
요구 사항	260
게이트웨이 생성	261
Siemens Databus user 생성	262
애플리케이션에 액세스	264
애플리케이션 설치	264
설치된 애플리케이션 구성 업데이트	266
대상 및 경로 필터	267
대상 이해	268
경로 필터 이해	271
실시간 대상 추가	274
Amazon S3를 사용하여 버퍼링된 대상 추가	282
경로 필터 추가	289
대상 관리	292
게이트웨이 관리	298
AWS IoT SiteWise 콘솔을 사용하여 SiteWise Edge 게이트웨이 관리	299

용를 사용하여 SiteWise Edge 게이트웨이 관리 AWS OpsHubAWS IoT SiteWise	299
로컬 운영 체제 자격 증명을 사용하여 SiteWise Edge 게이트웨이에 액세스	301
SiteWise Edge 게이트웨이 인증서 관리	303
SiteWise Edge 게이트웨이 구성 요소 팩 버전 변경	304
SiteWise Edge 게이트웨이 나열	305
SiteWise Edge 게이트웨이 설명	305
SiteWise Edge 게이트웨이 생성	306
SiteWise Edge 게이트웨이 업데이트	307
게이트웨이 기능 구성 업데이트	308
게이트웨이 리소스에 태그 지정	309
게이트웨이에 대한 태그 나열	310
게이트웨이에서 태그 제거	311
구성 요소의 AWS IoT SiteWise 버전 업데이트	312
SiteWise Edge 게이트웨이 삭제	313
게이트웨이 백업 및 복원	314
지표 데이터의 일일 백업	314
SiteWise Edge 게이트웨이 복원	315
AWS IoT SiteWise 데이터 복원	316
백업 및 복원의 성공 여부 확인	317
레거시 게이트웨이(AWS IoT Greengrass Version 1)	317
산업 자산 모델링	319
자산 및 모델 상태	320
자산의 상태 확인	321
자산 또는 구성 요소 모델의 상태 확인	323
자산 모델 버전	325
자산 모델 또는 구성 요소 모델의 활성 버전 검색(콘솔)	326
자산 모델 또는 구성 요소 모델의 활성 버전 검색(AWS CLI)	326
사용자 지정 복합 모델(구성 요소)	327
인라인 사용자 지정 복합 모델	328
구성 요소 모델 기반 사용자 지정 복합 모델	329
경로를 사용하여 사용자 지정 복합 모델 속성 참조	331
객체 ID 설정	333
객체 UUID 작업	333
외부 ID 재사용	333
모델 생성	335
에서 자산 모델 생성 AWS IoT SiteWise	336

구성 요소 모델 생성	351
데이터 속성 정의	354
사용자 지정 복합 모델(구성 요소) 생성	431
자산 생성	435
자산 생성(콘솔)	435
자산 생성(AWS CLI)	436
새 자산 구성	438
자산 검색	438
사전 조건	438
에 대한 고급 검색 AWS IoT SiteWise Console	438
속성 값 업데이트	442
자산 연결 및 연결 해제	444
자산 연결 및 연결 해제(콘솔)	445
자산 연결 및 연결 해제(AWS CLI)	446
자산 및 모델 업데이트	448
에서 자산 업데이트 AWS IoT SiteWise	448
자산 모델 및 구성 요소 모델 업데이트	450
사용자 지정 복합 모델 업데이트(구성 요소)	455
자산 모델 쓰기를 위한 낙관적 잠금	458
에서 자산 및 모델 삭제 AWS IoT SiteWise	462
자산 삭제	462
자산 모델 삭제	464
자산 및 모델을 사용한 일괄 작업	466
주요 개념 및 용어	467
지원되는 기능	468
일괄 작업 사전 조건	468
일괄 가져오기 작업 실행	471
일괄 내보내기 작업 실행	473
작업 진행 상황 추적 및 오류 처리	476
메타데이터 가져오기 예시	481
메타데이터 예시 내보내기	496
메타데이터 전송 작업 스키마	499
경보로 데이터 모니터링	518
경보 유형	518
경보 상태	519
경보 상태 속성	520

자산 모델에 대한 경보 정의	523
경보 알림 요구 사항	526
AWS IoT Events 경보 정의	527
외부 경보 정의	560
자산에 대한 경보 구성	562
임계값 구성(콘솔)	563
임계값 구성(AWS CLI)	563
알림 설정 구성	565
경보에 대응	568
경보에 대응(콘솔)	568
경보에 응답(API)	571
외부 경보 상태 수집	572
외부 경보 상태 스트림 매핑	572
경보 상태 데이터 수집	574
AWS IoT SiteWise 어시스턴트	576
AWS IoT SiteWise 도우미 구성	576
데이터세트 생성	578
데이터 세트 편집	583
데이터 세트 삭제	585
AWS IoT SiteWise 어시스턴트 질문	586
를 사용하여 데이터 모니터링 AWS IoT SiteWise Monitor	587
SiteWise Monitor 역할	588
SAML 페더레이션	589
SiteWise Monitor 개념	590
시작하기 AWS IoT SiteWise Monitor (클래식)	592
포털 생성	593
포털 구성	594
관리자 초대	598
포털 사용자 추가	600
대시보드 생성(CLI)	605
포털에 대한 경보 켜기	610
엣지에서 포털 활성화	614
포털 관리	614
시작하기 AWS IoT SiteWise Monitor (AI 인식)	624
포털 생성	625
포털 구성	626

포털 관리	628
포털 삭제	632
를 사용하여 대시보드 생성 AWS CLI	633
포털 로그인	638
프로젝트 생성	638
프로젝트 업데이트	639
프로젝트 삭제	639
대시보드 생성	640
대시보드 업데이트	641
대시보드 삭제	642
대시보드 구성	643
에서 데이터 쿼리 AWS IoT SiteWise	664
현재 자산 값 쿼리	665
자산 속성의 현재 값 쿼리(콘솔)	665
자산 속성의 현재 값 쿼리(AWS CLI)	665
기록 자산 속성 값 쿼리	666
자산 속성 집계 쿼리	668
자산 속성에 대한 집계(API)	669
자산 속성에 대한 집계를 쿼리하려면(AWS CLI) 다음을 수행하세요.	670
AWS IoT SiteWise 쿼리 언어	671
사전 조건	671
쿼리 언어 참조	672
다른 서비스와 상호 작용	680
MQTT 주제의 자산 속성 이해	680
알림 사용	681
자산 속성 알림 켜기(콘솔)	681
자산 속성 알림 켜기(AWS CLI)	682
쿼리 알림	684
Amazon S3로 데이터 내보내기	687
Grafana 통합	687
와 통합 AWS IoT TwinMaker	688
통합 활성화	689
AWS IoT SiteWise 및 통합 AWS IoT TwinMaker	689
장비 이상 감지	690
예측 정의 추가(콘솔)	692
예측 훈련(콘솔)	695

예측에 대한 추론 시작 또는 중지(콘솔)	696
예측 정의 추가(CLI)	697
예측 훈련 및 추론 시작(CLI)	700
예측 훈련(CLI)	701
예측에 대한 추론 시작 또는 중지(CLI)	703
데이터 스토리지 관리	706
스토리지 설정 구성	707
데이터 보존 영향	707
웜 티어 구성(콘솔)	708
웜 티어에 대한 구성(AWS CLI)	709
콜드 티어에 대한 구성(콘솔)	712
콜드 티어 구성(AWS CLI)	715
스토리지 설정 문제 해결	720
오류: 버킷이 존재하지 않습니다	720
오류: Amazon S3 경로에 대한 액세스 거부됨	720
오류: 역할 ARN을 맡을 수 없습니다	721
오류: 리전 간 S3 버킷에 액세스하지 못함	721
콜드 티어에 저장된 데이터의 파일 경로 및 스키마	721
장비 데이터(측정값)	722
지표, 변환, 집계	725
자산 메타데이터	729
자산 계층 구조 메타데이터	733
스토리지 데이터 인덱스 파일	735
코드 예제	736
기본 사항	740
안녕하세요 AWS IoT SiteWise	741
기본 사항 알아보기	744
작업	808
보안	881
데이터 보호	882
인터넷워크 트래픽 개인 정보	883
AWS IoT SiteWise 어시스턴트 비즈니스 서비스 개선	883
데이터 암호화	883
저장 시 암호화	884
전송 중 암호화	886
키 관리	888

자격 증명 및 액세스 관리	889
대상	890
ID로 인증	890
AWS IoT SiteWise 에서 IAM을 사용하는 방법	893
관리형 정책	911
서비스 연결 역할	915
경보에 대한 권한 설정	932
의 교차 서비스 혼동된 대리자 방지 AWS IoT SiteWise	938
자격 증명 및 액세스 문제 해결	939
규정 준수 확인	941
복원성	942
인프라 보안	942
구성 및 취약성 분석	943
VPC 엔드포인트	944
지원되는 API 작업	944
인터페이스 VPC 엔드포인트 생성	947
인터페이스 VPC 엔드포인트를 AWS IoT SiteWise 통한 액세스	947
VPC 엔드포인트 정책 생성	949
보안 모범 사례	950
OPC UA 서버에서 인증 자격 증명 사용	950
OPC UA 서버에 암호화된 통신 모드 사용	950
구성 요소를 최신 상태로 유지	950
SiteWise Edge 게이트웨이의 파일 시스템 암호화	951
엣지 구성에 대한 보안 액세스	951
Siemens Industrial Edge Management의 데이터 보안	951
SiteWise Monitor 사용자에게 가능한 최소 권한 부여	951
민감한 정보 노출 금지	952
AWS IoT Greengrass 보안 모범 사례 준수	952
다음 사항도 참조하세요.	952
로깅 및 모니터링	953
서비스 로그 모니터링	953
로그인 관리 AWS IoT SiteWise	955
예: AWS IoT SiteWise 로그 파일 항목	956
SiteWise Edge 게이트웨이 로그 모니터링	957
Amazon CloudWatch Logs 사용	957
서비스 로그 사용	959

Amazon CloudWatch 지표를 사용한 모니터링	960
AWS IoT Greengrass Version 2 게이트웨이 지표	961
를 사용하여 API 호출 로깅 AWS CloudTrail	969
AWS IoT SiteWise CloudTrail의 정보	970
AWS IoT SiteWise CloudTrail의 데이터 이벤트	970
AWS IoT SiteWise CloudTrail의 관리 이벤트	973
예: AWS IoT SiteWise 로그 파일 항목	973
리소스에 태그 지정	975
에서 태그 사용 AWS IoT SiteWise	975
로 태그 지정 AWS Management Console	975
AWS IoT SiteWise API로 태그 지정	975
IAM 정책에 태그 사용	977
문제 해결	979
게이트웨이의 문제 해결	979
SiteWise Edge 게이트웨이 로그 구성 및 액세스	979
SiteWise Edge 게이트웨이 문제 해결	980
에서 AWS IoT SiteWise Edge 애플리케이션 문제 해결 Siemens Industrial Edge	986
Edge에서 오픈 소스 통합 문제 해결	987
AWS IoT Greengrass 문제 해결	989
포털 문제 해결	989
사용자와 관리자가 AWS IoT SiteWise 포털에 접근할 수 없음	989
AWS IoT SiteWise 규칙 작업 문제 해결	991
AWS IoT Core 로그 구성	991
재게시 오류 작업 구성	992
규칙 문제 해결	994
규칙 문제 해결(AWS IoT SiteWise)	996
규칙 문제 해결(DynamoDB)	997
일괄 가져오기 및 내보내기 문제 해결	1001
엔드포인트 및 할당량	1002
엔드포인트	1002
할당량	1002
AWS IoT SiteWise 어시스턴트 API 제한 한도	1016
이상 감지 할당량	1016
문서 기록	1017
.....	mxxxv

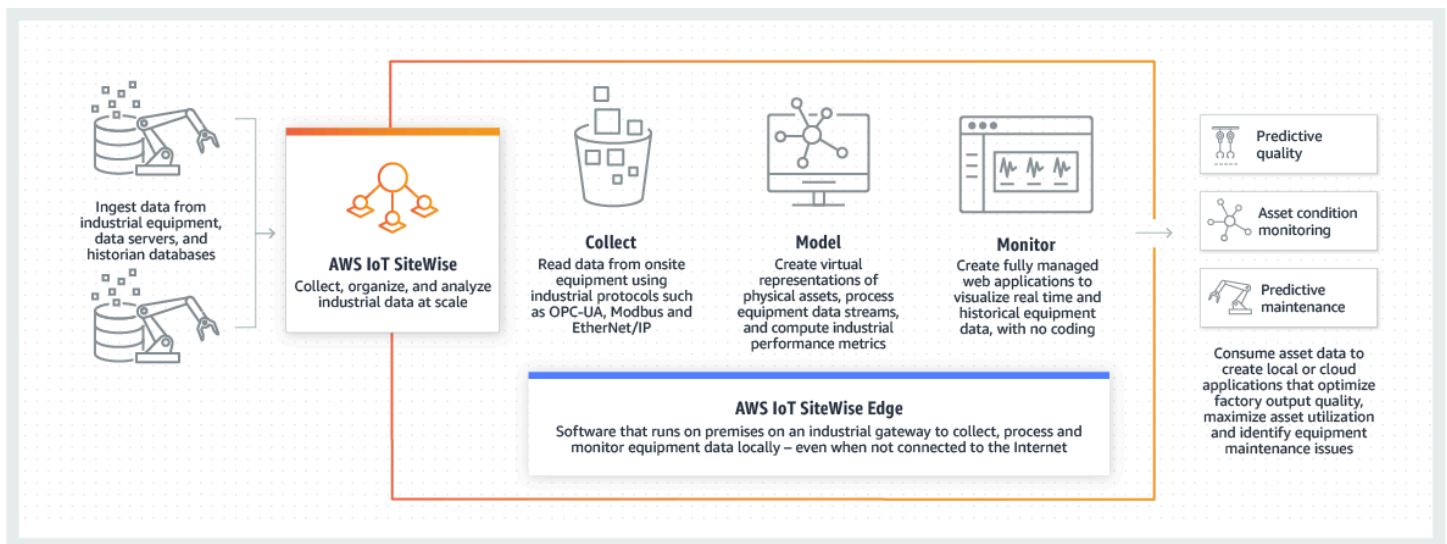
AWS IoT SiteWise란 무엇인가요?

AWS IoT SiteWise 는 대규모 산업 장비에서 데이터를 수집, 저장, 구성 및 모니터링하여 더 나은 데이터 기반 결정을 내리는 데 도움이 되는 관리형 서비스입니다. AWS IoT SiteWise 를 사용하여 시설 전반의 운영을 모니터링하고, 일반적인 산업 성능 지표를 빠르게 계산하고, 산업 장비 데이터를 분석하여 비용이 많이 드는 장비 문제를 방지하고, 프로덕션 격차를 줄이는 애플리케이션을 만들 수 있습니다.

AWS IoT SiteWise Monitor를 사용하면 운영 사용자가 웹 애플리케이션을 생성하여 산업 데이터를 실시간으로 보고 분석할 수 있습니다. 평균 고장 시간 및 전반적인 장비 효율성(OEE) 등의 지표를 구성하고 모니터링하여 산업 조직에 대한 통찰을 얻을 수 있습니다.

AWS IoT SiteWise Edge는 로컬 디바이스에서 데이터를 수집, 저장 및 처리할 수 있는 AWS IoT SiteWise 있는의 구성 요소입니다. 인터넷에 대한 액세스가 제한되거나 데이터를 비공개로 유지해야 하는 경우에 유용합니다.

다음 다이어그램은의 기본 아키텍처를 보여줍니다 AWS IoT SiteWise.



주제

- [AWS IoT SiteWise 작동 방식](#)
- [에 대한 사용 사례 AWS IoT SiteWise](#)
- [AWS SDK에서이 서비스 사용](#)
- [AWS IoT SiteWise 개념](#)

AWS IoT SiteWise 작동 방식

AWS IoT SiteWise 는 산업 디바이스, 프로세스 및 시설의 표현을 생성하는 데 사용할 수 있는 리소스 모델링 프레임워크를 제공합니다. 장비 및 프로세스의 표현을에서 자산 모델이라고 합니다 AWS IoT SiteWise. 자산 모델을 사용하여 사용할 원시 데이터, 원시 데이터를 유용한 지표로 처리하는 방식을 정의합니다. [AWS IoT SiteWise 콘솔](#)에서 산업 조직의 자산 및 모델을 구축하고 시각화할 수 있습니다. 엣지 또는 AWS 클라우드에서 데이터를 수집하고 처리하도록 자산 모델을 구성할 수도 있습니다.

주제

- [산업 데이터 수집](#)
- [수집된 데이터를 컨텍스트화하기 위한 자산 모델링](#)
- [쿼리, 경보 및 예측을 사용하여 분석](#)
- [작업 시각화](#)
- [데이터 저장](#)
- [다른 서비스와 통합](#)

산업 데이터 수집

산업 데이터를 수집 AWS IoT SiteWise 하여 사용을 시작합니다. 데이터 수집은 다음 방법 중 하나로 이루어집니다.

- 현장 서버에서 직접 수집: OPC UA와 같은 프로토콜을 사용하여 현장 디바이스에서 직접 데이터를 읽습니다. 공통 산업 게이트웨이 또는 가상 서버와 같은 AWS IoT Greengrass V2 다양한 플랫폼에와 호환되는 SiteWise Edge 게이트웨이 소프트웨어를 배포합니다. 단일 AWS IoT SiteWise 게이트웨이에 최대 100개의 OPC UA 서버를 연결할 수 있습니다. 자세한 내용은 [AWS IoT SiteWise 엣지 자체 호스팅 게이트웨이 요구 사항](#) 단원을 참조하십시오.

Modbus TCP 및 이더넷/IP(EIP)와 같은 프로토콜은 컨텍스트Domatica에서와 파트너십을 통해 지원됩니다 AWS IoT Greengrass V2.

- 팩을 사용한 엣지 데이터 처리: 포괄적인 엣지 기능을 활성화하기 위해 팩을 추가하여 SiteWise Edge 게이트웨이를 개선합니다. 에서 사용할 수 있는 SiteWise Edge를 사용하면 AWS IoT Greengrass 스트림을 사용하여 AWS 클라우드로 안전하게 전송되기 전에 AWS IoT Greengrass V2 데이터 처리가 현장에서 직접 실행됩니다. 자세한 내용은 [SiteWise Edge에서 OPC UA 소스 설정](#) 단원을 참조하십시오.

- 일괄 작업으로 Amazon S3를 통한 적응형 수집: 대량의 자산 또는 자산 모델로 작업할 때는 일괄 작업을 사용하여 Amazon S3 버킷에서 리소스를 일괄적으로 가져오고 내보냅니다. 자세한 내용은 [자산 및 모델을 사용한 일괄 작업](#) 단원을 참조하십시오.
- AWS IoT 코어 규칙이 포함된 MQTT 메시지: AWS IoT 코어 전송 MQTT 메시지에 연결된 디바이스의 경우 AWS IoT 코어 규칙 엔진을 사용하여 해당 메시지를 로 전달합니다 AWS IoT SiteWise. AWS IoT 코어 전송 [MQTT](#) 메시지에 연결된 디바이스가 있는 경우 AWS IoT 코어 규칙 엔진을 사용하여 해당 메시지를 로 라우팅합니다 AWS IoT SiteWise. 자세한 내용은 [AWS IoT Core 규칙을 AWS IoT SiteWise 사용하여 데이터 수집](#) 단원을 참조하십시오.
- 이벤트 트리거 데이터 수집: 작업을 AWS IoT Events 사용하여 이벤트가 발생할 AWS IoT SiteWise 때 로 데이터를 전송 AWS IoT Events 하도록의 IoT SiteWise 작업을 구성합니다. 자세한 내용은 [AWS IoT SiteWise 에서 로 데이터 수집 AWS IoT Events](#) 단원을 참조하십시오.
- AWS IoT SiteWise API: Edge 또는 클라우드의 애플리케이션이 데이터를 로 직접 전송할 수 있습니다 AWS IoT SiteWise. 자세한 내용은 [AWS IoT SiteWise APIs 사용하여 데이터 수집](#) 단원을 참조하십시오.

수집된 데이터를 컨텍스트화하기 위한 자산 모델링

데이터를 수집한 후 물리적 운영의 모델을 구축함으로써 데이터를 사용하여 자산, 프로세스 및 시설을 가상으로 표현할 수 있습니다. 디바이스 또는 프로세스를 나타내는 자산은 데이터 스트림을 AWS 클라우드로 전송합니다. 자산은 논리적 디바이스 그룹화를 나타낼 수도 있습니다. 계층 구조는 자산을 연결하여 복잡한 작업을 미러링하는 방법으로 형성됩니다. 이러한 계층 구조를 통해 자산은 연결된 하위 자산의 데이터에 액세스할 수 있습니다. 자산은 자산 모델로부터 만들어집니다. 자산 모델은 자산의 형식을 표준화하는 선언형 구조입니다. 모델의 구성 및 유지 관리 용이성을 위해 자산의 구성 요소를 재사용합니다. 자세한 내용은 [산업 자산 모델링](#) 단원을 참조하십시오.

를 사용하면 수신 데이터를 컨텍스트 지표 및 변환으로 변환하도록 자산을 구성할 AWS IoT SiteWise 수 있습니다.

- 장비 데이터를 수신할 때 변환이 수행됩니다.
- 지표는 정의한 간격으로 계산됩니다.

지표 및 변환은 개별 자산 또는 여러 자산 모두에 적용됩니다. AWS IoT SiteWise 는 장비 데이터, 지표 및 변환과 관련된 다양한 기간에 걸쳐 평균, 합계 및 수와 같이 일반적으로 사용되는 통계 집계를 자동으로 계산합니다.

를 사용하여 자산을 동기화할 수 있습니다 AWS IoT TwinMaker. 자세한 내용은 [AWS IoT SiteWise 및 통합 AWS IoT TwinMaker](#) 단원을 참조하십시오.

쿼리, 경고 및 예측을 사용하여 분석

쿼리를 실행하고 경보를 설정 AWS IoT SiteWise 하여 로 수집된 날짜를 분석합니다. Amazon Lookout 을 사용하여 지표 내의 이상을 자동으로 감지하고 근본 원인을 식별할 수도 있습니다.

- 특정 경보를 설정하여 장비 또는 프로세스가 최적의 성능에서 벗어나면 팀에게 알림으로써 신속한 문제 식별 및 해결을 보장합니다. 자세한 내용은 [에서 경보를 사용하여 데이터 모니터링 AWS IoT SiteWise](#) 단원을 참조하십시오.
- AWS IoT SiteWise API 작업을 사용하여 특정 시간 간격 동안 자산 속성의 현재 값, 기록 값 및 집계 를 쿼리합니다. 자세한 내용은 [에서 데이터 쿼리 AWS IoT SiteWise](#) 단원을 참조하십시오.
- Amazon Lookout for Equipment에서 이상 감지를 사용하여 장비 또는 작동 조건의 변화를 식별하고 시각화합니다. 이상 감지를 사용하면 작업에 대한 예방적 유지 관리 조치를 결정할 수 있습니다. 이 통합을 통해 고객은 AWS IoT SiteWise 및 Amazon Lookout for Equipment 간에 데이터를 동기화할 수 있습니다. 자세한 내용은 [Lookout for Equipment로 이상 감지](#) 단원을 참조하십시오.

작업 시각화

SiteWise Monitor를 설정하여 운영 직원을 위한 웹 애플리케이션을 만듭니다. 웹 애플리케이션은 직원 이 작업을 시각화하는 데 도움이 됩니다. IAM Identity Center 또는 IAM을 사용하여 직원의 다양한 액세스 수준을 처리합니다. 전체 산업 운영의 특정 하위 집합을 볼 수 있도록 각 직원의 고유한 로그인 및 권한을 구성합니다. 이러한 직원이 SiteWise Monitor를 사용하는 방법을 배울 수 있도록 [애플리케이션 가이드](#)를 AWS IoT SiteWise 제공합니다.

작업 시각화에 대한 자세한 내용은 [를 사용하여 데이터 모니터링 AWS IoT SiteWise Monitor](#) 섹션을 참조하세요.

데이터 저장

시계열 스토리지를 산업 데이터 lake와 통합할 수 있습니다. AWS IoT SiteWise 에는 산업 데이터에 대한 세 가지 스토리지 계층이 있습니다.

- 실시간 애플리케이션에 최적화된 핫 스토리지 계층입니다.
- 분석 워크로드에 최적화된 웜 스토리지 계층입니다.
- 자연 시간이 긴 운영 데이터 애플리케이션에 Amazon S3를 사용하는 고객 관리형 콜드 스토리지 계층입니다.

AWS IoT SiteWise 는 최신 데이터를 핫 스토리지 계층에 유지하여 스토리지 비용을 관리하는 데 도움이 됩니다. 그런 다음 데이터 보존 정책을 정의하여 기록 데이터를 웜 또는 콜드 티어 스토리지로 이동합니다. 자세한 내용은 [에서 데이터 스토리지 관리 AWS IoT SiteWise](#) 단원을 참조하십시오.

자산 메타데이터를 가져오고 내보낼 수도 있습니다. 자세한 내용은 [자산 메타데이터](#)를 참조하세요.

다른 서비스와 통합

AWS IoT SiteWise 는 여러 AWS 서비스와 통합되어 AWS 클라우드에서 완전한 AWS IoT 솔루션을 개발합니다. 자세한 내용은 [다른 AWS 서비스와 상호 작용](#) 단원을 참조하십시오.

에 대한 사용 사례 AWS IoT SiteWise

AWS IoT SiteWise 는 다양한 산업에서 많은 산업 데이터 수집 및 분석 애플리케이션에 사용됩니다.

문제를 신속하게 해결하는 데 도움이 되도록 모든 소스에서 일관되게 데이터를 수집합니다. AWS IoT SiteWise 는 원격 모니터링을 제공하여 현장에서 직접 데이터를 수집하거나 여러 시설의 여러 소스에서 데이터를 수집합니다.는 산업 IoT 데이터 솔루션에 필요한 유연성을 AWS IoT SiteWise 제공합니다.

제조

AWS IoT SiteWise 는 장비에서 데이터를 수집하고 활용하여 비효율성을 정확히 찾아내고 최소화하는 프로세스를 간소화하여 산업 운영을 개선할 수 있습니다.는 제조 라인 및 장비에서 데이터를 수집하는데 AWS IoT SiteWise 도움이 됩니다. 를 사용하면 데이터를 AWS 클라우드로 전송하고 특정 장비 및 프로세스에 대한 성능 지표를 구축할 AWS IoT SiteWise수 있습니다. 이 지표를 사용해 전반적인 운영 효과를 파악하고 혁신 및 개선이 필요한 영역을 파악할 수 있습니다. 또한 제조 프로세스를 보고 장비 및 프로세스 결함, 생산 격차 또는 제품 결함을 식별할 수 있습니다.

식음료

식음료 산업 시설에서는 곡물 제분, 도축/육류 포장, 간편 조리 식품의 조합/조리/냉동을 비롯하여 다양한 식품 처리를 진행합니다. 식품 처리 공장은 여러 위치에 분포되어 있을 때가 많으며, 공장과 장비 운영자는 중앙 위치에서 프로세스와 장비를 모니터링합니다. 예를 들어, 냉장 유닛은 재료 취급 및 유통기한 만료를 평가합니다. 시설 전반의 폐기물 발생을 모니터링하여 운영 효율성을 보장합니다. AWS IoT SiteWise를 사용하면 여러 위치의 센서 데이터 스트림을 프로덕션 라인 및 시설별로 그룹화하여 프로세스 엔지니어가 시설 전체에서 더 잘 이해하고 개선할 수 있습니다.

에너지/공공 설비

AWS IoT SiteWise를 사용하면 장비 문제를 더 쉽고 효율적으로 해결할 수 있습니다. 자산 성능을 원격으로 그리고 실시간으로 모니터링할 수 있습니다. 어디에서나 기록 장비 데이터에 액세스하여 잠재적 문제를 정확히 파악하고, 정확한 리소스를 배포하고, 문제를 더 빠르게 방지 및 해결합니다.

AWS SDK에서이 서비스 사용

AWS 소프트웨어 개발 키트(SDKs)는 널리 사용되는 많은 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예제
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	AWS Tools for PowerShell 코드 예제
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

예제 사용 가능 여부

필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

AWS IoT SiteWise 개념

다음은의 핵심 개념입니다 AWS IoT SiteWise.

Aggregate

집계는가 모든 시계열 데이터에 대해 AWS IoT SiteWise 자동으로 계산하는 기본 지표 또는 측정값입니다. 자세한 내용은 [에서 자산 속성 집계 쿼리 AWS IoT SiteWise](#) 단원을 참조하십시오.

자산

산업 장비 AWS IoT SiteWise 에서에 데이터를 입력하거나 수집하면 디바이스, 장비 및 프로세스가 각각 자산으로 표시됩니다. 각 자산에는 연결된 데이터가 있습니다. 예를 들어 장비에는 일련 번호, 위치, 제조업체 및 모델, 설치 날짜가 있을 수 있습니다. 또한 가용성, 성능, 품질, 온도, 압력 등에 대한 시계열 값도 있을 수 있습니다. 자산을 계층으로 그룹화하여 자산이 하위 자산에 저장된 데이터에 액세스할 수 있도록 합니다. 자세한 내용은 [산업 자산 모델링](#) 단원을 참조하십시오.

자산 계층 구조

자산 계층 구조를 설정하여 산업 조직의 논리적 표현을 만들 수 있습니다. 이렇게 하려면 자산 모델에서 계층을 정의하고 해당 모델에서 생성된 자산을 지정된 계층과 연결합니다. 상위 자산의 지표는 하위 자산 속성의 데이터를 결합하여 전체 작업 또는 특정 부분에 대한 인사이트를 제공하는 지표를 계산할 수 있습니다. 자세한 내용은 [자산 모델 계층 정의](#) 단원을 참조하십시오.

자산 모델

모든 자산은 자산 모델을 사용하여 생성됩니다. 자산 모델은 자산의 형식을 정의 및 표준화하는 구조입니다. 자산 모델은 동일한 유형의 여러 자산에 걸쳐 일관된 정보를 적용하므로 디바이스 그룹을 나타내는 자산의 데이터를 처리할 수 있습니다. 각 자산 모델에서 [특성](#), 시계열 입력([측정](#)), 시계열 변환([변환](#)), 시계열 집계([지표](#)) 및 [자산 계층 구조](#)를 정의할 수 있습니다. 자세한 내용은 [산업 자산 모델링](#) 단원을 참조하십시오.

엣지용으로 자산 모델을 구성하여 자산 모델의 속성이 처리되는 위치를 결정합니다. 이 기능을 사용하여 로컬 장치의 자산 데이터를 처리하고 모니터링합니다.

자산 속성

자산 속성은 산업 데이터를 포함하는 각 자산 내부의 구조입니다. 각 속성에는 데이터 유형이 있으며 단위 또한 있을 수 있습니다. 속성은 [특성](#), [측정치](#), [변환](#) 또는 [지표](#)가 될 수 있습니다. 자세한 내용은 [데이터 속성 정의](#) 단원을 참조하십시오.

엣지에서 계산하도록 자산 속성을 구성합니다. 엣지에서 데이터 프로세싱에 대한 자세한 내용은 [SiteWise Edge에서 OPC UA 소스 설정](#) 단원을 참조하세요.

특성

특성은 디바이스 제조업체 또는 디바이스 위치와 같이 일반적으로 일정하게 유지되는 자산의 속성입니다. 특성에는 사전 설정된 값이 있을 수 있습니다. 자산 모델에서 만들어진 각 자산은 해당 모델에서 정의된 특성의 기본값을 포함합니다. 자세한 내용은 [정적 데이터\(속성\) 정의](#) 단원을 참조하십시오.

대시보드

각 프로젝트에는 대시보드 집합이 포함되어 있습니다. 대시보드는 자산 집합의 값에 대한 시각화 집합을 제공합니다. 프로젝트 소유자는 대시보드와 대시보드에 포함된 시각화를 생성합니다. 프로젝트 소유자가 대시보드 집합을 공유할 준비가 되면 소유자는 프로젝트에 뷰어를 초대하여 프로젝트의 모든 대시보드에 대한 액세스 권한을 부여할 수 있습니다. 다른 대시보드에 대해 다른 뷰어 집합을 사용하려면 프로젝트 간에 대시보드를 나누어야 합니다. 뷰어는 대시보드를 볼 때 특정 데이터를 확인할 수 있도록 시간 범위를 사용자 지정할 수 있습니다.

데이터 세트

데이터 세트는 시계열 데이터, 시계열non-time-series 데이터, 교대 근무 일정, 유지 관리 레코드, 직원 데이터베이스와 같은 비 장비 데이터를 나타내는 데이터 모음입니다. 외부 데이터를 지원하고 AWS IoT SiteWise 분석 기능을 사용합니다. 여기에는 데이터 세트 소스, 데이터 세트 스키마 및 데이터 세트 파라미터가 포함됩니다. AWS IoT SiteWise Assistant는 Amazon Kendra 인덱스를 사용하는 데이터 세트를 사용합니다.

데이터 스트림

자산 모델 및 자산을 생성하기 전에 AWS IoT SiteWise 산업 데이터스에 입력하거나 수집합니다. AWS IoT SiteWise 자동으로 데이터 스트림을 생성하여 장비에서 원시 데이터 스트림을 수집합니다.

데이터 스트림 별칭

데이터 스트림 별칭을 사용하면 데이터 스트림을 쉽게 식별할 수 있습니다. 예를 들어 server1-windfarm/3/turbine/7/temperature 별칭은 풍력 발전소 #3의 터빈 #7에서 나오는 온도 값

을 나타냅니다. server1 용어는 OPC UA 서버를 식별하는 데 도움이 되는 데이터 소스 이름이며, server1-은 이 OPC UA 서버에서 보고된 모든 데이터 스트림에 연결된 접두사입니다.

데이터 스트림 연결

자산 모델 및 자산을 생성한 후 데이터 스트림을 자산에 정의된 자산 속성과 연결하여 데이터를 구조화합니다. 그런 다음 자산 모델 및 자산을 사용하여 데이터 스트림에서 들어오는 데이터를 처리할 AWS IoT SiteWise 수 있습니다. 자산 속성에서 데이터 스트림을 연결 해제할 수도 있습니다. 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 단원을 참조하십시오.

대상

SiteWise Edge의 대상은 원격 측정 또는 처리된 데이터를 전송하려는 엔드포인트를 나타냅니다. SiteWise Edge는 AWS IoT SiteWise 핫 티어, 버퍼링된 수집 또는 Amazon S3 버킷을 대상으로 지원합니다. 경로 필터를 사용하여 특정 MQTT 주제를 구독하도록 대상을 구성할 수 있습니다. 자세한 내용은 [AWS IoT SiteWise 엣지 대상 이해](#) 단원을 참조하십시오.

공식

모든 [변환](#) 및 [지표](#) 속성에는 해당 속성이 데이터를 변환하거나 집계하는 방법을 정의하는 공식이 있습니다. 이러한 공식은 AWS IoT SiteWise에서 제공하는 속성 입력, 연산자 및 함수로 구성됩니다. 자세한 내용은 [수식 표현식 사용](#) 단원을 참조하십시오.

측정

측정은 장치 또는 장비의 원시 센서 시계열 데이터 스트림을 나타내는 자산 속성입니다. 자세한 내용은 [장비의 데이터 스트림 정의\(측정값\)](#) 단원을 참조하십시오.

지표

지표는 집계된 시계열 데이터를 나타내는 자산의 속성입니다. 각 지표에는 데이터 포인트를 집계하는 방법과 해당 집계를 계산하기 위한 시간 간격을 설명하는 수학적 표현식([공식](#))이 함께 제공됩니다. 지표는 지정된 각 시간 간격에 대해 단일 데이터 포인트를 생성합니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하십시오.

MQTT

MQTT(Message Queuing Telemetry Transport)는 센서 및 디바이스를 위한 경량 메시징 프로토콜입니다.

팩

SiteWise Edge 게이트웨이는 팩을 사용하여 데이터를 수집, 처리 및 라우팅하는 방법을 결정합니다. SiteWise Edge 게이트웨이에서 사용 가능한 팩에 대한 자세한 내용은 [the section called “팩 사용”](#) 섹션을 참조하세요.

데이터 수집 팩

SiteWise Edge 게이트웨이가 산업 데이터를 수집하여 원하는 AWS 대상으로 라우팅할 수 있도록 데이터 수집 팩을 사용합니다.

데이터 처리 팩

데이터 처리 팩을 사용하여 최대 30일 동안 엣지에서 데이터를 처리, 저장 및 검색할 수 있습니다. SiteWise Edge APIs.

OPC UA

OPC UA(Open Platform Communications Unified Architecture)는 산업 자동화를 위한 통신 프로토콜입니다.

경로 필터

게이트웨이 내의 경로 필터를 사용하여 MQTT 주제를 구독하고 지원되는 대상에 AWS IoT SiteWise 게시합니다. MQTT 기반 소스, 데이터 처리 파이프라인 및 대상은 모두 자체 호스팅 MQTT 지원 V3 게이트웨이에서 MQTT 주제를 사용하여 데이터를 교환합니다. 주제 필터를 정의하여 수집하거나 다른 대상으로 라우팅할 데이터를 지정할 수 있습니다.

Portal

AWS IoT SiteWise Monitor 포털은 AWS IoT SiteWise 데이터를 시각화하고 공유하는 데 사용할 수 있는 웹 애플리케이션입니다. 포털에는 관리자가 한 명 이상 있으며 프로젝트가 0개 이상 포함되어 있습니다.

포털 관리자

각 SiteWise Monitor 포털에는 한 명 이상의 포털 관리자가 있습니다. 포털 관리자는 포털을 사용하여 자산 및 대시보드 컬렉션이 포함된 프로젝트를 생성합니다. 그런 다음 포털 관리자가 각 프로젝트에 자산과 소유자를 할당합니다. 프로젝트에 대한 액세스를 제어하면 포털 관리자가 프로젝트 소유자와 뷰어가 볼 수 있는 자산을 제한할 수 있습니다.

Project

각 SiteWise Monitor 포털에는 일련의 프로젝트가 포함되어 있습니다. 각 프로젝트에는 연결된 AWS IoT SiteWise 자산의 하위 집합이 있습니다. 프로젝트 소유자는 하나 이상의 대시보드를 생성하여 해당 자산과 연결된 데이터를 일관된 방식으로 볼 수 있습니다. 프로젝트 소유자는 프로젝트에 뷰어를 초대하여 프로젝트의 자산 및 대시보드를 보도록 허용할 수 있습니다. 이 프로젝트는 SiteWise Monitor 내 공유의 기본 단위입니다. 프로젝트 소유자는 AWS 관리자가 포털에 대한 액세스 권한을 부여한 사용자를 초대할 수 있습니다. 해당 포털에 있는 프로젝트를 해당 사용자와 공유하려면 먼저 사용자에게 포털에 대한 액세스 권한이 있어야 합니다.

프로젝트 소유자

각 SiteWise Monitor 프로젝트에는 소유자가 있습니다. 프로젝트 소유자는 운영 데이터를 일관된 방식으로 나타내기 위해 대시보드 형태로 시각화를 생성합니다. 대시보드를 공유할 준비가 되면 프로젝트 소유자가 프로젝트에 뷰어를 초대할 수 있습니다. 프로젝트 소유자는 프로젝트에 다른 소유자를 할당할 수도 있습니다. 프로젝트 소유자는 경보에 대한 임계값 및 알림 설정을 구성할 수 있습니다.

프로젝트 뷰어

각 SiteWise Monitor 프로젝트에는 뷰어가 있습니다. 프로젝트 뷰어는 포털에 연결하여 프로젝트 소유자가 생성한 대시보드를 볼 수 있습니다. 각 대시보드에서 프로젝트 뷰어는 시간 범위를 조정하여 운영 데이터를 더 잘 이해할 수 있습니다. 프로젝트 뷰어는 액세스 권한이 있는 프로젝트의 대시보드만 볼 수 있습니다. 프로젝트 뷰어는 경보를 확인하고 일시 중지할 수 있습니다.

속성 별칭

OPC UA 서버 데이터 스트림 경로(예: /company/windfarm/3/turbine/7/temperature)와 같은 자산 속성에 별칭을 만들어 자산 데이터를 수집하거나 검색하는 동안 자산 속성을 쉽게 식별할 수 있습니다. [SiteWise Edge 게이트웨이](#)를 사용하여 서버에서 데이터를 수집하는 경우 속성 별칭이 원시 데이터 스트림의 경로와 일치해야 합니다. 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 단원을 참조하십시오.

속성 알림

자산 속성에 대한 속성 알림을 활성화하면 속성이 새 값을 수신할 AWS IoT Core 때마다 MQTT 메시지를 AWS IoT SiteWise 게시합니다. 메시지 페이로드에는 해당 속성 값의 업데이트에 대한 세부 정보가 포함됩니다. 속성 값 알림을 사용하여의 산업 데이터를 AWS IoT SiteWise 다른 AWS 서비스와 연결하는 솔루션을 생성합니다. 자세한 내용은 [다른 AWS 서비스와 상호 작용](#) 단원을 참조하십시오.

SiteWise Edge 게이트웨이

SiteWise Edge 게이트웨이는 데이터를 수집, 처리 및 전송하기 위해 고객의 온프레미스에 설치됩니다. SiteWise Edge 게이트웨이는 다양한 프로토콜을 통해 산업 데이터 소스에 연결하여 데이터를 수집하고 처리하여 AWS 클라우드로 전송합니다. SiteWise Edge 게이트웨이는 파트너 데이터 소스에도 연결할 수 있습니다. 자세한 내용은 [AWS IoT SiteWise Edge 게이트웨이 사용](#) 단원을 참조하십시오.

변환

변환은 변환된 시계열 데이터를 나타내는 자산의 속성입니다. 모든 변환에는 데이터 포인트를 한 형식에서 다른 형식으로 변환하는 방법을 정의하는 수학적 표현식([공식](#))이 있습니다. 변형된 데이터

포인트는 입력 데이터 포인트와 일대일 관계를 유지합니다. 자세한 내용은 [데이터 변환\(변환\)](#) 단원을 참조하세요.

시각화

각 대시보드에서 프로젝트 소유자는 프로젝트와 연결된 속성과 경보를 표시하는 방법을 결정합니다. 가용성은 꺾은선형 차트로 가장 잘 표현되고 다른 값은 막대 차트나 핵심 성능 지표(KPI)로 표시될 수 있습니다. 경보는 상태 그리드 및 상태 타임라인으로 표시하는 것이 가장 좋습니다. 프로젝트 소유자는 각 시각화를 사용자 지정하여 해당 자산의 데이터를 가장 잘 이해할 수 있도록 합니다.

시작하기 AWS IoT SiteWise

AWS IoT SiteWise를 사용하면 데이터를 수집, 구성, 분석 및 시각화할 수 있습니다.

AWS IoT SiteWise 는 실제 데이터 소스를 구성하지 않고도 서비스를 탐색하는 데 사용할 수 있는 데모를 제공합니다. 자세한 내용은 [AWS IoT SiteWise 데모 사용](#) 단원을 참조하십시오.

다음 자습서를 완료하여 AWS IoT SiteWise의 특정 기능을 살펴볼 수 있습니다.

- [AWS IoT 사물 AWS IoT SiteWise 에서 로 데이터 수집](#)
- [SiteWise Monitor에서 풍력 발전소 데이터 시각화 및 공유](#)
- [Amazon DynamoDB에 속성 값 업데이트 게시](#)

자세한 내용은 AWS IoT SiteWise다음 주제를 참조하세요.

- [예 데이터 수집 AWS IoT SiteWise](#)
- [산업 자산 모델링](#)
- [Edge에서 AWS IoT SiteWise 엣지 기능 구성](#)
- [를 사용하여 데이터 모니터링 AWS IoT SiteWise Monitor](#)
- [에서 데이터 쿼리 AWS IoT SiteWise](#)
- [다른 AWS 서비스와 상호 작용](#)

주제

- [요구 사항](#)
- [AWS 계정 설정](#)

요구 사항

시작하려면 AWS 계정이 있어야 합니다 AWS IoT SiteWise. 없는 경우 다음 섹션에서 계정을 설정하는 방법을 알아봅니다.

를 사용할 수 있는 리전 AWS IoT SiteWise 을 사용합니다. 자세한 내용은 [AWS IoT SiteWise 엔드포인트 및 할당량](#)을 참조하세요. 의 리전 선택기 AWS Management Console 를 사용하여 이러한 리전 중 하나로 전환할 수 있습니다.

AWS 계정 설정

주제

- [에 가입 AWS 계정](#)
- [관리자 액세스 권한이 있는 사용자 생성](#)

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. <https://portal.aws.amazon.com/billing/signup>을 엽니다.
2. 온라인 지시 사항을 따릅니다.

등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

에 가입하면 AWS 계정AWS 계정 루트 사용자의 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 [루트 사용자 액세스 권한이 필요한 작업](#)을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 <https://aws.amazon.com/>으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자[AWS Management Console](#)로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 [루트 사용자 로 로그인](#)을 참조하세요.

2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

지침은 IAM 사용 설명서의 [AWS 계정 루트 사용자\(콘솔\)에 대한 가상 MFA 디바이스 활성화를 참조하세요.](#)

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

지침은 AWS IAM Identity Center 사용 설명서의 [AWS IAM Identity Center 설정](#)을 참조하세요.

2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

를 자격 증명 소스 IAM Identity Center 디렉터리로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 [기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리](#) 참조하세요.

관리 액세스 권한이 있는 사용자 로그인

- IAM Identity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 [AWS 액세스 포털에 로그인](#)을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

지침은 AWS IAM Identity Center 사용 설명서의 [Create a permission set](#)을 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

지침은 AWS IAM Identity Center 사용 설명서의 [Add groups](#)을 참조하세요.

AWS IoT SiteWise 데모 사용

AWS IoT SiteWise demo. AWS IoT SiteWise provides를 AWS IoT SiteWise 사용하여 쉽게 탐색할 수 있습니다. 데모를 AWS CloudFormation 템플릿으로 제공하여 자산 모델, 자산 및 SiteWise Monitor 포털을 생성하고 최대 1주일 동안 샘플 데이터를 생성할 수 있습니다.

Important

데모를 만들면 이 데모가 만들고 사용하는 리소스에 대한 요금이 청구됩니다.

주제

- [AWS IoT SiteWise 데모 생성](#)
- [AWS IoT SiteWise 데모 삭제](#)

AWS IoT SiteWise 데모 생성

AWS IoT SiteWise 콘솔에서 AWS IoT SiteWise 데모를 생성할 수 있습니다.

Note

데모는 Lambda 함수, CloudWatch Events 규칙 1개, 데모에 필요한 AWS Identity and Access Management (IAM) 역할을 생성합니다. AWS 계정에 이러한 리소스가 표시될 수 있습니다. 데모를 완료할 때까지 이러한 리소스를 보관하는 것이 좋습니다. 이러한 리소스를 삭제하면 데모가 제대로 작동하지 않을 수 있습니다.

AWS IoT SiteWise 콘솔에서 데모를 생성하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동하여 페이지의 오른쪽 상단에 있는 SiteWise 데모를 찾습니다.
2. (선택 사항) SiteWise 데모에서 Days to keep demo assets(데모 자산 보관 일수) 필드를 데모 삭제 전 보관 일수로 변경합니다.
3. (선택 사항) 샘플 데이터를 모니터링할 SiteWise 모니터 포털을 만들려면 다음을 수행하세요.

Note

이 데모가 생성하고 사용하는 SiteWise Monitor 리소스에 대한 요금이 청구됩니다. 자세한 내용은 AWS IoT SiteWise 요금의 [SiteWise 모니터](#)를 참조하세요.

- 모니터 리소스를 선택합니다.
- 권한을 선택합니다.
- 페더레이션된 IAM 사용자에게 포털 액세스 권한을 부여하는 기존 IAM 역할을 선택합니다.

Important

IAM 역할에 다음 권한이 있어야 합니다.

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

SiteWise Monitor를 사용하는 방법에 대한 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [What is AWS IoT SiteWise Monitor?](#)를 참조하세요.

4. Create demo(데모 생성)을 선택합니다.

데모를 만드는 데 약 3분이 걸립니다. 데모를 만들지 못하는 경우, 계정에 사용 권한이 부족하기 때문일 수 있습니다. 관리 권한이 있는 계정으로 전환하거나 다음 단계에 따라 데모를 삭제하고 다시 시도합니다.

a. Delete demo(데모 삭제)를 선택합니다.

데모를 삭제하는 데 약 15분이 걸립니다.

b. 데모가 삭제되지 않으면 [AWS CloudFormation 콘솔](#)을 열고 IoTSiteWiseDemoAssets라는 스택을 선택한 다음 오른쪽 상단에서 삭제를 선택합니다.

c. 데모가 다시 삭제되지 않는 경우 AWS CloudFormation 콘솔의 단계에 따라 삭제에 실패한 리소스를 건너뛰고 다시 시도하세요.

5. 데모가 성공적으로 생성되면 [AWS IoT SiteWise 콘솔](#)에서 데모 자산 및 데이터를 탐색할 수 있습니다.

AWS IoT SiteWise 데모 삭제

AWS IoT SiteWise 데모는 일주일 후 또는 AWS CloudFormation 콘솔에서 데모 스택을 생성한 경우 선택한 일수를 삭제합니다. 데모 리소스의 사용을 마쳤으면 그 전에 데모를 삭제할 수 있습니다. 데모를 만드는 데 실패한 경우에도 데모를 삭제할 수 있습니다. 데모를 수동으로 삭제하려면 다음 단계를 수행하세요.

AWS IoT SiteWise 데모를 삭제하려면

1. [AWS CloudFormation 콘솔](#)로 이동합니다.
2. Stacks 목록에서 IoTSiteWiseDemoAssets를 선택합니다.
3. Delete(삭제)를 선택합니다.

스택을 삭제하면 데모용으로 생성된 모든 리소스가 삭제됩니다.

4. 확인 대화 상자에서 스택 삭제를 선택합니다.

스택을 삭제하는 데 약 15분이 걸립니다. 데모가 삭제되지 않으면 오른쪽 상단에서 삭제를 다시 선택합니다. 데모가 다시 삭제되지 않는 경우 AWS CloudFormation 콘솔의 단계에 따라 삭제에 실패한 리소스를 건너뛰고 다시 시도하세요.

AWS IoT SiteWise 자습서

자 AWS IoT SiteWise 습서 페이지에 오신 것을 환영합니다. 점점 규모가 커지고 있는 이 자습서 컬렉션은 AWS IoT SiteWise의 복잡성을 탐색하는 데 필요한 지식과 기술을 제공합니다. 이러한 자습서에서는 필요에 맞는 다양한 기본 주제를 제공합니다. 자습서를 살펴보면서의 다양한 측면에 대한 귀중한 인사이트를 찾아보세요 AWS IoT SiteWise.

각 자습서에서는 특정 장비 예시를 사용합니다. 이러한 자습서는 테스트 환경을 위한 것이며 가상의 회사 이름, 모델, 자산, 속성 등을 사용합니다. 그 목적은 일반적인 지침을 제공하는 것입니다. 이 자습서는 조직의 고유한 요구 사항에 맞도록 주의 깊은 검토 및 조정 없이 프로덕션 환경에 바로 사용할 수 있도록 고안된 것이 아닙니다.

주제

- [에서 OEE 계산 AWS IoT SiteWise](#)
- [AWS IoT 사물 AWS IoT SiteWise 에서 로 데이터 수집](#)
- [SiteWise Monitor에서 풍력 발전소 데이터 시각화 및 공유](#)
- [Amazon DynamoDB에 속성 값 업데이트 게시](#)

에서 OEE 계산 AWS IoT SiteWise

이 자습서에서는 제조 프로세스에 대한 전반적인 장비 효율성(OEE)을 계산하는 방법에 대한 예가 수록되어 있습니다. 결과적으로, OEE 계산 또는 공식이 여기에 표시된 것과 다를 수도 있습니다. 일반적으로 OEE는 Availability * Quality * Performance로 정의됩니다. OEE 계산에 대한 자세한 내용은 Wikipedia의 [전체 장비 효율성](#)을 참조하세요.

사전 조건

이 자습서를 완료하려면 다음과 같은 세 가지 데이터 스트림이 있는 디바이스에 대한 데이터 수집을 구성해야 합니다.

- Equipment_State – 유휴, 결함, 계획된 중지 또는 정상 작동과 같이 시스템 상태를 나타내는 숫자 코드입니다.
- Good_Count – 각 데이터 포인트가 마지막 데이터 포인트 이후 성공한 작업 수를 포함하는 데이터 스트림입니다.
- Bad_Count – 각 데이터 포인트가 마지막 데이터 포인트 이후 실패한 작업 수를 포함하는 데이터 스트림입니다.

데이터 수집을 구성하려면 [예 데이터 수집 AWS IoT SiteWise](#) 단원을 참조하세요. 사용 가능한 산업 조직이 없는 경우 AWS IoT SiteWise API를 통해 샘플 데이터를 생성하고 업로드하는 스크립트를 작성할 수 있습니다.

OEE 계산 방법

이 자습서에서는 Equipment_State, Good_Count 및 Bad_Count 등, 세 가지 데이터 입력 스트림에서 OEE를 계산하는 자산 모델을 생성합니다. 이 예에서는 설탕, 감자 칩 또는 페인트 포장에 사용되는 일반 포장기를 살펴보고겠습니다. [AWS IoT SiteWise 콘솔](#)에서 다음 측정값, 변환 및 지표를 사용하여 AWS IoT SiteWise 자산 모델을 생성합니다. 그런 다음 패키징 머신을 나타내는 자산을 생성하고 OEE를 AWS IoT SiteWise 계산하는 방법을 관찰할 수 있습니다.

포장기의 원시 데이터 스트림을 나타내는 다음 [측정](#)을 정의합니다.

측정

- Equipment_State – 포장기의 현재 상태를 숫자 코드로 나타내는 데이터 스트림(또는 측정)입니다.
 - 1024 – 머신이 유휴 상태입니다.
 - 1020 – 오류나 지연 같은 결함입니다.
 - 1000 – 계획된 중지입니다.
 - 1111 – 정상적인 작업입니다.
- Good_Count – 각 데이터 포인트가 마지막 데이터 포인트 이후 성공한 작업 수를 포함하는 데이터 스트림입니다.
- Bad_Count – 각 데이터 포인트가 마지막 데이터 포인트 이후 실패한 작업 수를 포함하는 데이터 스트림입니다.

Equipment_State 측정 데이터 스트림 및 포함된 코드를 사용하여 다음 [변형](#)(또는 파생 측정)을 정의합니다. 변형은 원시 측정과 일대일 관계에 있습니다.

변형

- Idle = eq(Equipment_State, 1024) – 머신의 유휴 상태를 포함하는 변형된 데이터 스트림입니다.
- Fault = eq(Equipment_State, 1020) – 머신의 결함 상태를 포함하는 변형된 데이터 스트림입니다.

- $\text{Stop} = \text{eq}(\text{Equipment\_State}, 1000)$ – 머신의 계획된 중지 상태를 포함하는 변형된 데이터 스트림입니다.
- $\text{Running} = \text{eq}(\text{Equipment\_State}, 1111)$ – 머신의 정상 작동 상태를 포함하는 변형된 데이터 스트림입니다.

원시 측정과 변형된 측정을 사용하여 지정된 시간 간격 동안 머신 데이터를 집계하는, 다음과 같은 [지표](#)를 정의합니다. 이 섹션에서 지표를 정의할 때 각 지표에 동일한 시간 간격을 선택합니다.

Metrics

- $\text{Successes} = \text{sum}(\text{Good\_Count})$ – 지정된 시간 간격 동안 성공적으로 채워진 패키지 수입니다.
- $\text{Failures} = \text{sum}(\text{Bad\_Count})$ – 지정된 시간 간격 동안 성공적으로 채워지지 않은 패키지 수입니다.
- $\text{Idle\_Time} = \text{statetime}(\text{Idle})$ – 지정된 시간 간격당 머신의 총 유휴 시간(초)입니다.
- $\text{Fault\_Time} = \text{statetime}(\text{Fault})$ – 지정된 시간 간격당 머신의 총 결함 시간(초)입니다.
- $\text{Stop\_Time} = \text{statetime}(\text{Stop})$ – 지정된 시간 간격당 머신의 총 계획된 중지 시간(초)입니다.
- $\text{Run\_Time} = \text{statetime}(\text{Running})$ – 지정된 시간 간격당 문제 없이 실행되는 머신의 총 시간(초)입니다.
- $\text{Down\_Time} = \text{Idle\_Time} + \text{Fault\_Time} + \text{Stop\_Time}$ – 지정된 시간 간격 동안 머신의 총 가동 중지 시간(초)으로, 이외의 머신 상태의 합계로 계산됩니다. Run_Time .
- $\text{Availability} = \text{Run\_Time} / (\text{Run\_Time} + \text{Down\_Time})$ – 지정된 시간 간격 동안 머신의 가동 시간 또는 머신을 작동에 사용할 수 있는 예약 시간의 백분율입니다.
- $\text{Quality} = \text{Successes} / (\text{Successes} + \text{Failures})$ – 지정된 시간 간격 동안 머신의 성공적으로 채워진 패키지 백분율입니다.
- $\text{Performance} = ((\text{Successes} + \text{Failures}) / \text{Run\_Time}) / \text{Ideal\_Run\_Rate}$ – 지정된 시간 간격 동안 프로세스의 이상적인 실행 속도(초) 대비 백분율로 나타낸 머신 성능입니다.

예를 들어, Ideal_Run_Rate 은 분당 60개의 패키지(초당 1개 패키지)가 될 수 있습니다. 분당 또는 시간당 Ideal_Run_Rate 인 경우 Run_Time 은 초 단위이므로 적절한 단위 변환 인수로 나누어야 합니다.

- $\text{OEE} = \text{Availability} * \text{Quality} * \text{Performance}$ – 지정된 시간 간격 동안 머신의 전반적인 장비 효율성. 이 공식은 OEE를 1의 분수로 계산합니다.

Note

OEE가 변환으로 정의된 경우 출력 값은 각 입력 값에 대해 계산됩니다. 변환 평가는 공식의 모든 기여 속성에 대해 사용 가능한 최신 값을 고려하므로 예상치 못한 값을 생성할 가능성이 있습니다. 타임스탬프가 동일한 속성 업데이트의 경우 다른 수신 속성의 업데이트로 출력 값을 덮어쓸 수 있습니다. 예를 들어 가용성, 품질 및 성능이 계산되면 OEE는 다른 두 속성에 대해 마지막으로 사용 가능한 데이터 포인트로 계산됩니다. 이러한 기여 값은 타임스탬프를 공유하며 OEE의 잘못된 출력 값을 유발합니다. 변환 계산에는 순서가 보장되지 않습니다.

AWS IoT 사물 AWS IoT SiteWise 에서 로 데이터 수집

이 자습서에서는 디바이스 새도우를 사용하여 AWS IoT 사물 플릿 AWS IoT SiteWise 에서 로 데이터를 수집하는 방법을 알아봅니다. 디바이스 새도우는 AWS IoT 디바이스의 현재 상태 정보를 저장하는 JSON 객체입니다. 자세한 내용은 AWS IoT 개발자 가이드의 [디바이스 새도우 서비스](#) 단원을 참조하세요.

이 자습서를 완료한 후 사물을 AWS IoT SiteWise 기반으로 AWS IoT 에서 작업을 설정할 수 있습니다. AWS IoT 사물을 사용하면 작업을의 다른 유용한 기능과 통합할 수 있습니다 AWS IoT. 예를 들어 다음 작업을 수행하도록 AWS IoT 기능을 구성할 수 있습니다.

- [AWS IoT Events](#), [Amazon DynamoDB](#) 및 기타 AWS 서비스로 데이터를 스트리밍하는 추가 규칙을 구성합니다. 자세한 내용은 AWS IoT 개발자 가이드의 [규칙](#)을 참조하세요.
- AWS IoT 플릿 인덱싱 서비스를 사용하여 디바이스 데이터를 인덱싱, 검색 및 집계합니다. 자세한 내용은 AWS IoT 개발자 가이드의 [플릿 인덱싱](#)을 참조하세요.
- 를 사용하여 디바이스를 감사하고 보호합니다 AWS IoT Device Defender. 자세한 내용은 AWS IoT 개발자 안내서의 [AWS IoT Device Defender](#) 섹션을 참조하세요.

이 자습서에서는 AWS IoT 사물의 디바이스 새도우에서 자산으로 데이터를 수집하는 방법을 알아봅니다 AWS IoT SiteWise. 이렇게 하려면 하나 이상의 AWS IoT 사물을 생성하고 각 사물의 디바이스 새도우를 CPU 및 메모리 사용량 데이터로 업데이트하는 스크립트를 실행합니다. 이 자습서에서는 CPU 및 메모리 사용량 데이터를 사용하여 실제 센서 데이터를 모방합니다. 그런 다음 사물의 디바이스 새도우가 업데이트 AWS IoT SiteWise 될 때마다 데이터들의 자산으로 보내는 AWS IoT SiteWise 작업이 포함된 규칙을 생성합니다. 자세한 내용은 [AWS IoT Core 규칙을 AWS IoT SiteWise 사용하여 데이터 수집](#) 단원을 참조하십시오.

주제

- [사전 조건](#)
- [1단계: AWS IoT 정책 생성](#)
- [2단계: AWS IoT 사물 생성 및 구성](#)
- [3단계: 디바이스 자산 모델 생성](#)
- [4단계: 디바이스 플릿 자산 모델 생성](#)
- [5단계: 디바이스 자산 생성 및 구성](#)
- [6단계: 디바이스 플릿 자산 생성 및 구성](#)
- [7단계: AWS IoT 코어에서 디바이스 자산으로 데이터를 전송하는 규칙 생성](#)
- [8단계: 디바이스 클라이언트 스크립트 실행](#)
- [9단계: 자습서 완료 후 리소스 정리](#)

사전 조건

이 자습서를 완료하려면 다음이 필요합니다.

- AWS 계정. 없으면 [AWS 계정 설정](#) 단원을 참조하세요.
- Windows, Linux, 또는 macOS를 실행하여 macOS에 액세스하는 개발 컴퓨터입니다 AWS Management Console. 자세한 내용은 [AWS Management Console 시작하기](#)를 참조하세요.
- 관리자 권한이 있는 AWS Identity and Access Management (IAM) 사용자.
- Python 3은 개발 컴퓨터에 설치되거나 AWS IoT 사물로 등록하려는 디바이스에 설치됩니다.

1단계: AWS IoT 정책 생성

이 절차에서는 AWS IoT 사물이 이 자습서에 사용된 리소스에 액세스할 수 있도록 허용하는 AWS IoT 정책을 생성합니다.

AWS IoT 정책을 생성하려면

1. [AWS Management Console](#)에 로그인합니다.
2. 가 AWS IoT SiteWise 지원되는 [AWS 리전](#)을 검토합니다. 필요한 경우 지원되는 리전 중 하나로 전환합니다.
3. [AWS IoT 콘솔](#)로 이동합니다. 디바이스 연결 버튼이 나타나면 선택합니다.
4. 왼쪽 탐색 창에서 보안을 선택하고 정책을 선택합니다.
5. 생성(Create)을 선택합니다.

6. AWS IoT 정책의 이름을 입력합니다(예: **SiteWiseTutorialDevicePolicy**).
7. 정책 문서에서 JSON을 선택하여 JSON 형식으로 다음 정책을 입력합니다. **##** 및 **account-id**를 사용자 리전 및 계정 ID로 바꿉니다(예: **us-east-1** 및 **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*"
    }
  ]
}

```

이 정책을 통해 AWS IoT 디바이스는 MQTT 메시지를 사용하여 연결을 설정하고 디바이스 새도우와 통신할 수 있습니다. MQTT 메시지에 대한 자세한 내용은 [MQTT란 무엇인가요?](#)를 참조하세요. 디바이스 새도우와 상호 작용하기 위해 AWS IoT 사물은 로 시작하는 주제에 대해 MQTT 메시지를 게시하고 수신합니다 \$aws/things/**thing-name**/shadow/. 이 정책은 \${iot:Connection.Thing.ThingName}이라는 사물 정책 변수를 통합합니다. 이 변수는 각 주제에서 연결된 사물의 이름을 대체합니다. iot:Connect 문은 연결할 수 있는 디바이스를 제한하므로 사물 정책 변수는 SiteWiseTutorialDevice로 시작하는 이름만 대체할 수 있습니다.

자세한 내용은 AWS IoT 개발자 가이드의 [사물 정책 변수](#)를 참조하세요.

Note

이 정책은 이름이 SiteWiseTutorialDevice로 시작하는 사물에 적용됩니다. 사물에 다른 이름을 사용하려면 그에 따라 정책을 업데이트해야 합니다.

8. 생성(Create)을 선택합니다.

2단계: AWS IoT 사물 생성 및 구성

이 절차에서는 AWS IoT 사물을 생성하고 구성합니다. 개발 컴퓨터를 AWS IoT 사물로 지정할 수 있습니다. 진행하면서 여기서 학습하는 원칙을 실제 프로젝트에 적용할 수 있다는 점을 기억하세요. AWS IoT Greengrass 및 FreeRTOS를 포함하여 AWS IoT SDK를 실행할 수 있는 모든 디바이스에서 AWS IoT 사물을 유연하게 만들고 설정할 수 있습니다. 자세한 내용은 AWS IoT 개발자 가이드의 [AWS IoT SDK](#) 단원을 참조하세요.

AWS IoT 사물을 생성하고 구성하려면

1. 명령줄을 열고 다음 명령을 실행하여 이 자습서의 디렉터리를 생성합니다.

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

2. 다음 명령을 실행하여 사물의 인증서 디렉터를 생성합니다.

```
mkdir device1
```

추가 사물을 생성하는 경우 그에 따라 디렉터리 이름의 번호를 늘려 어떤 인증서가 어떤 사물에 속하는지 추적하세요.

3. [AWS IoT 콘솔](#)로 이동합니다.
4. 왼쪽 탐색 창에서 관리 섹션의 모든 디바이스를 선택합니다. 그런 다음 사물을 선택합니다.
5. 아직 사물이 없습니다 대화 상자가 나타나면 사물 생성을 선택합니다. 또는 사물 생성을 선택합니다.
6. 사물 생성 페이지에서 단일 사물 생성을 선택한 후 다음을 선택합니다.
7. 사물 속성 지정 페이지에서 AWS IoT 사물의 이름(예: **SiteWiseTutorialDevice1**)을 입력하고 다음을 선택합니다. 추가 사물을 생성하는 경우 그에 따라 사물 이름의 번호를 늘립니다.

Important

사물 이름은 1단계: 정책 생성에서 생성한 AWS IoT 정책에 사용된 이름과 일치해야 합니다. 그렇지 않으면 디바이스를 연결할 수 없습니다 AWS IoT.

8. 디바이스 인증서 구성 - 선택 사항 페이지에서 새 인증서 자동 생성(권장)을 선택한 후 다음을 선택합니다. 인증서를 사용하면 디바이스 AWS IoT 를 안전하게 식별할 수 있습니다.

9. 인증서에 정책 연결 - 선택 사항 페이지에서 1단계: 정책 생성에서 생성한 AWS IoT 정책을 선택하고 사물 생성을 선택합니다.
10. 인증서 및 키 다운로드 대화 상자에서 다음 작업을 수행합니다.

- a. 다운로드 링크를 선택하여 사물의 인증서, 퍼블릭 키 및 프라이빗 키를 다운로드합니다. 사물의 인증서(예: `iot-sitewise-rule-tutorial/device1`)에 대해 생성한 디렉터리에 세 개 파일을 모두 저장합니다.

Important

디바이스를 AWS IoT에 성공적으로 연결하는 데 필요한 사물의 인증서와 키를 다운로드할 수 있는 유일한 시간입니다.

- b. 다운로드 링크를 선택하여 루트 CA 인증서를 다운로드합니다. 루트 CA 인증서를 `iot-sitewise-rule-tutorial`에 저장합니다. Amazon Root CA 1을 다운로드하는 것이 좋습니다.

11. 완료를 선택합니다.

이제 컴퓨터에 AWS IoT 사물을 등록했습니다. 다음 단계 중 하나를 수행합니다.

- 3단계: 추가 사물을 생성하지 않고 디바이스 자산 모델 생성을 계속합니다. AWS IoT 하나의 사물만으로 이 자습서를 완료할 수 있습니다.
- 다른 컴퓨터 또는 디바이스에서 이 섹션의 단계를 반복하여 더 많은 AWS IoT 사물을 생성합니다. 이 자습서에서는 여러 디바이스에서 고유한 CPU 및 메모리 사용량 데이터를 수집할 수 있도록 이 옵션을 따르는 것이 좋습니다.
- 동일한 디바이스(컴퓨터)에서 이 섹션의 단계를 반복하여 더 많은 AWS IoT 사물을 생성합니다. 각 AWS IoT 사물은 컴퓨터에서 유사한 CPU 및 메모리 사용량 데이터를 수신하므로 이 접근 방식을 사용하여 여러 디바이스에서 고유하지 않은 데이터를 수집하는 방법을 보여줍니다.

3단계: 디바이스 자산 모델 생성

이 절차에서는에서 자산 모델을 생성 AWS IoT SiteWise 하여 CPU 및 메모리 사용량 데이터를 스트리밍하는 디바이스를 나타냅니다. 디바이스 그룹을 나타내는 자산의 데이터를 처리하기 위해 자산 모델은 동일한 유형의 여러 자산에 걸쳐 일관된 정보를 적용합니다. 자세한 내용은 [산업 자산 모델링](#) 단원을 참조하십시오.

디바이스를 나타내는 자산 모델 생성

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 모델을 선택합니다.
3. Create model(모델 생성)을 선택합니다.
4. 모델 세부 정보에서 모델 이름을 입력합니다. 예를 들어 **SiteWise Tutorial Device Model**입니다.
5. 측정 정의에서 다음을 수행합니다.
 - a. 이름에 **CPU Usage**를 입력합니다.
 - b. 단위에 %를 입력합니다.
 - c. 데이터 유형을 Double로 둡니다.

측정 속성은 디바이스의 원시 데이터 스트림을 나타냅니다. 자세한 내용은 [장비의 데이터 스트림 정의\(측정값\)](#) 단원을 참조하세요.

6. 새 측정 추가를 선택하여 두 번째 측정 속성을 추가합니다.
7. 측정 정의 아래의 두 번째 행에서 다음을 수행합니다.
 - a. 이름에 **Memory Usage**를 입력합니다.
 - b. 단위에 %를 입력합니다.
 - c. 데이터 유형을 Double로 둡니다.
8. 지표 정의에서 다음을 수행합니다.
 - a. 이름에 **Average CPU Usage**를 입력합니다.
 - b. 공식에 **avg(CPU Usage)**를 입력합니다. 자동 완성 목록에서 CPU Usage가 나타나면 선택합니다.
 - c. 시간 간격에 **5 minutes**를 입력합니다.

지표 속성은 한 간격 동안 모든 입력 데이터 포인트를 처리하고 간격당 단일 데이터 포인트를 출력하는 집계 계산을 정의합니다. 이 지표 속성은 각 디바이스의 평균 CPU 사용량을 5분마다 계산합니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하세요.

9. 새 지표 추가를 선택하여 두 번째 지표 속성을 추가합니다.
10. 지표 정의 아래의 두 번째 행에서 다음을 수행합니다.

- a. 이름에 **Average Memory Usage**를 입력합니다.
- b. 공식에 **avg(Memory Usage)**를 입력합니다. 자동 완성 목록에서 Memory Usage가 나타나면 선택합니다.
- c. 시간 간격에 **5 minutes**를 입력합니다.

이 지표 속성은 각 디바이스의 평균 메모리 사용량을 5분마다 계산합니다.

11. (선택 사항) 디바이스별로 계산하려는 다른 지표를 추가합니다. 몇 가지 흥미로운 함수는 min 및 max를 포함합니다. 자세한 내용은 [수식 표현식 사용](#) 단원을 참조하세요. 단계 4: 디바이스 플릿 자산 모델 생성에서는 전체 디바이스 플릿의 데이터를 사용하여 지표를 계산할 수 있는 상위 자산을 생성합니다.
12. Create model(모델 생성)을 선택합니다.

4단계: 디바이스 플릿 자산 모델 생성

이 절차에서는에서 자산 모델을 만들어 디바이스 컬렉션을 AWS IoT SiteWise 기호화합니다. 이 자산 모델 내에서 여러 디바이스 자산을 하나의 전반적인 플릿 자산에 연결할 수 있는 구조를 설정합니다. 그런 다음 플릿 자산 모델의 지표를 요약하여 연결된 모든 디바이스 자산의 데이터를 통합합니다. 이 접근 방식은 전체 플릿의 집합적 성능에 대한 포괄적인 인사이트를 제공합니다.

디바이스 플릿을 나타내는 자산 모델 생성

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 모델을 선택합니다.
3. Create model(모델 생성)을 선택합니다.
4. 모델 세부 정보에서 모델 이름을 입력합니다. 예를 들어 **SiteWise Tutorial Device Fleet Model**입니다.
5. 계층 구조 정의에서 다음을 수행하세요.
 - a. 계층 구조 이름에 **Device**를 입력합니다.
 - b. 계층 구조 모델에서 디바이스 자산 모델(**SiteWise Tutorial Device Model**)을 선택합니다.

계층 구조는 상위(플릿) 자산 모델과 하위(디바이스) 자산 모델 간의 관계를 정의합니다. 상위 자산은 하위 자산의 속성 데이터에 액세스할 수 있습니다. 나중에 자산을 생성할 때는 상위 자산 모델

의 계층 구조 정의에 따라 하위 자산을 상위 자산에 연결해야 합니다. 자세한 내용은 [자산 모델 계층 정의](#) 단원을 참조하세요.

6. 지표 정의에서 다음을 수행합니다.

- a. 이름에 **Average CPU Usage**를 입력합니다.
- b. 공식에 **avg(Device | Average CPU Usage)**를 입력합니다. 자동 완성 목록이 나타나면 Device를 선택하여 계층 구조를 선택한 다음 Average CPU Usage를 선택하여 이전에 생성한 디바이스 자산에서 지표를 선택합니다.
- c. 시간 간격에 **5 minutes**를 입력합니다.

이 지표 속성은 **Device** 계층 구조를 통해 플릿 자산과 연결된 모든 디바이스 자산의 평균 CPU 사용량을 계산합니다.

7. 새 지표 추가를 선택하여 두 번째 지표 속성을 추가합니다.

8. 지표 정의 아래의 두 번째 행에서 다음을 수행합니다.

- a. 이름에 **Average Memory Usage**를 입력합니다.
- b. 공식에 **avg(Device | Average Memory Usage)**를 입력합니다. 자동 완성 목록이 나타나면 Device를 선택하여 계층 구조를 선택한 다음 Average Memory Usage를 선택하여 이전에 생성한 디바이스 자산에서 지표를 선택합니다.
- c. 시간 간격에 **5 minutes**를 입력합니다.

이 지표 속성은 **Device** 계층 구조를 통해 플릿 자산과 연결된 모든 디바이스 자산의 평균 메모리 사용량을 계산합니다.

9. (선택 사항) 디바이스 플릿에서 계산하려는 다른 지표를 추가합니다.

10. Create model(모델 생성)을 선택합니다.

5단계: 디바이스 자산 생성 및 구성

이 절차에서는 디바이스 자산 모델을 기반으로 디바이스 자산을 생성합니다. 그런 다음 각 측정 속성에 대한 속성 별칭을 정의합니다. 속성 별칭은 자산 속성을 식별하는 고유 문자열입니다. 나중에 자산 ID 및 속성 ID 대신 별칭을 사용하여 데이터 업로드를 위한 속성을 식별할 수 있습니다. 자세한 내용은 [예 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 단원을 참조하십시오.

디바이스 자산을 생성하고 속성 별칭 정의

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 자산을 선택합니다.
3. 자산 생성을 선택합니다.
4. 모델 정보에서 디바이스 자산 모델인 **SiteWise Tutorial Device Model**을 선택합니다.
5. 자산 정보에서 자산 이름을 입력합니다. 예를 들어 **SiteWise Tutorial Device 1**입니다.
6. 자산 생성을 선택합니다.
7. 새 디바이스 자산에 대해 편집을 선택합니다.
8. CPU Usage에서 속성 별칭으로 **/tutorial/device/SiteWiseTutorialDevice1/cpu**를 입력합니다. 속성 별칭에 AWS IoT 사물 이름을 포함하면 단일 AWS IoT 규칙을 사용하여 모든 디바이스에서 데이터를 수집할 수 있습니다.
9. Memory Usage에서 속성 별칭으로 **/tutorial/device/SiteWiseTutorialDevice1/memory**를 입력합니다.
10. 저장을 선택합니다.

이전에 여러 AWS IoT 사물을 생성한 경우 각 디바이스에 대해 3~10단계를 반복하고 그에 따라 자산 이름 및 속성 별칭의 숫자를 증가시킵니다. 예를 들어 두 번째 디바이스 자산의 이름은 **SiteWise Tutorial Device 2**이고 속성 별칭은 **/tutorial/device/SiteWiseTutorialDevice2/cpu** 및 **/tutorial/device/SiteWiseTutorialDevice2/memory**여야 합니다.

6단계: 디바이스 플릿 자산 생성 및 구성

이 절차에서는 디바이스 플릿 자산 모델에서 디바이스 플릿 자산을 생성합니다. 그런 다음 개별 디바이스 자산을 플릿 자산에 연결합니다. 이 연결을 통해 플릿 자산의 지표 속성이 여러 디바이스의 데이터를 컴파일하고 분석할 수 있습니다. 이 데이터는 전체 플릿의 집합적 성능에 대한 통합 보기를 제공합니다.

디바이스 플릿 자산을 생성하고 디바이스 플릿 연결

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 자산을 선택합니다.
3. 자산 생성을 선택합니다.
4. 모델 정보에서 디바이스 플릿 자산 모델인 **SiteWise Tutorial Device Fleet Model**을 선택합니다.

5. 자산 정보에서 자산 이름을 입력합니다. 예를 들어 **SiteWise Tutorial Device Fleet 1**입니다.
6. 자산 생성을 선택합니다.
7. 새 디바이스 플릿 자산에 대해 편집을 선택합니다.
8. 이 자산에 연결된 자산에서 연결된 자산 추가를 선택하고 다음을 수행합니다.
 - a. 계층 구조에서 Device를 선택합니다. 이 계층 구조는 디바이스 자산과 디바이스 플릿 자산 간의 계층적 관계를 식별합니다. 이 자습서의 앞부분에서 디바이스 플릿 자산 모델에서 이 계층 구조를 정의했습니다.
 - b. 자산에서 디바이스 자산 SiteWise Tutorial Device 1을 선택합니다.
9. (선택 사항) 이전에 여러 디바이스 자산을 생성한 경우 생성한 각 디바이스 자산에 대해 8~10단계를 반복합니다.
10. 저장을 선택합니다.

이제 디바이스 자산이 계층 구조에 구성되어 있어야 합니다.

7단계: AWS IoT 코어에서 디바이스 자산으로 데이터를 전송하는 규칙 생성

이 절차에서는 AWS IoT Core에서 규칙을 설정합니다. 이 규칙은 디바이스 새도우의 알림 메시지를 해석하고 디바이스 자산으로 데이터를 전송하도록 설계되었습니다 AWS IoT SiteWise. 디바이스의 새도우가 업데이트될 때마다 MQTT 메시지를 AWS IoT 보냅니다. MQTT 메시지를 기반으로 디바이스 새도우가 변경될 때 조치를 취하는 규칙을 생성할 수 있습니다. 이 경우 목표는 업데이트 메시지를 처리하고 속성 값을 추출하여 AWS IoT SiteWise의 디바이스 자산으로 전송하는 것입니다.

AWS IoT SiteWise 작업을 사용하여 규칙을 생성하려면

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 메시지 라우팅을 선택한 다음 규칙을 선택합니다.
3. 규칙 생성을 선택합니다.
4. 규칙의 이름과 설명을 입력하고 다음을 선택합니다.
5. 다음 SQL 문을 입력하고 다음을 선택합니다.

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



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

이 규칙 쿼리 문은 디바이스 새도우 서비스가 \$aws/things/*thingName*/shadow/update/accepted에 새도우 업데이트를 게시하기 때문에 작동합니다. 디바이스 새도우에 대한 자세한 내용은 AWS IoT 개발자 가이드의 [디바이스 새도우 서비스](#)를 참조하세요.

WHERE 절에서 이 규칙 쿼리 문은 topic(3) 함수를 사용하여 주제의 세 번째 세그먼트에서 사물 이름을 가져옵니다. 그런 다음 명령문은 자습서 디바이스의 이름과 일치하지 않는 이름을 가진 디바이스를 필터링합니다. AWS IoT SQL에 대한 자세한 내용은 AWS IoT 개발자 안내서의 [AWS IoT SQL 참조](#)를 참조하세요.

6. 규칙 작업에서 AWS IoT SiteWise의 자산 속성에 메시지 데이터 보내기를 선택하고 다음을 수행하세요.

- a. 속성 별칭 기준을 선택합니다.
- b. 속성 별칭에 **/tutorial/device/\${topic(3)}/cpu**를 입력합니다.

구문은 대체 템플릿입니다. \${...}는 중괄호 내의 내용을 AWS IoT 평가합니다. 이 대체 템플릿은 주제에서 사물 이름을 가져와서 각 사물에 고유한 별칭을 생성합니다. 자세한 내용은 AWS IoT 개발자 가이드의 [대체 템플릿](#)을 참조하세요.

Note

대체 템플릿의 표현식은 SELECT 문과 별도로 평가되므로 대체 템플릿을 사용하면 AS 절을 사용하여 생성한 별칭은 참조할 수 없습니다. 지원되는 함수와 연산자 외에 원래 페이로드에 있는 정보만 참조할 수 있습니다.

- c. 항목 ID - 선택 사항에 **\${concat(topic(3), "-cpu-", floor(state.reported.timestamp))}**를 입력합니다.

항목 ID는 각 값 입력 시도를 고유하게 식별합니다. 항목이 오류를 반환하는 경우 오류 출력에서 항목 ID를 찾아 문제를 해결할 수 있습니다. 이 항목 ID의 대체 템플릿은 사물 이름과 디바이스의 보고된 타임스탬프를 결합합니다. 예를 들어 결과 항목 ID는 SiteWiseTutorialDevice1-cpu-1579808494와 같을 수 있습니다.

- d. 초 단위 시간에 **\${floor(state.reported.timestamp)}**를 입력합니다.

이 대체 템플릿은 디바이스의 보고된 타임스탬프에서 시간(초)을 계산합니다. 이 자습서에서 디바이스가 Unix 에포크 시간의 타임스탬프(초)를 부동 소수점 숫자로 보고합니다.

- e. 나노초 단위 오프셋 - 선택 사항에 **`${floor((state.reported.timestamp % 1) * 1E9)}`**를 입력합니다.

이 대체 템플릿은 디바이스의 보고된 타임스탬프의 소수 부분을 변환하여 초 단위 시간에서 나노초 단위 오프셋을 계산합니다.

Note

AWS IoT SiteWise에서는 데이터의 현재 타임스탬프가 Unix epoch 시간이어야 합니다. 디바이스가 시간을 정확하게 보고하지 않는 경우 [timestamp\(\)](#)를 사용하여 AWS IoT 규칙 엔진에서 현재 시간을 가져올 수 있습니다. 이 함수는 시간을 밀리초 단위로 보고하므로 규칙 작업의 시간 파라미터를 다음 값으로 업데이트해야 합니다.

- 초 단위 시간에 **`${floor(timestamp() / 1E3)}`**를 입력합니다.
- 나노초 단위 오프셋에 **`${(timestamp() % 1E3) * 1E6}`**를 입력합니다.

- f. 데이터 유형에서 Double을 선택합니다.

이 데이터 유형은 자산 모델에서 정의한 자산 속성의 데이터 유형과 일치해야 합니다.

- g. 값에는 **`${state.reported.cpu}`**를 입력합니다. 대체 템플릿에서는 . 연산자를 사용하여 JSON 구조 내에서 값을 검색합니다.
- h. 항목 추가를 선택하여 메모리 사용량 속성에 대한 새 항목을 추가하고 해당 속성에 대해 다음 단계를 다시 완료합니다.

- i. 속성 별칭 기준을 선택합니다.

- ii. 속성 별칭에 **`/tutorial/device/${topic(3)}/memory`**를 입력합니다.

- iii. 항목 ID - 선택 사항에 **`${concat(topic(3), "-memory-", floor(state.reported.timestamp))}`**를 입력합니다.

- iv. 초 단위 시간에 **`${floor(state.reported.timestamp)}`**를 입력합니다.

- v. 나노초 단위 오프셋 - 선택 사항에 **`${floor((state.reported.timestamp % 1) * 1E9)}`**를 입력합니다.

- vi. 데이터 유형에서 Double을 선택합니다.

- vii. 값에는 **`${state.reported.memory}`**를 입력합니다.

- i. IAM 역할에서 새 역할 생성을 선택하여 이 규칙 작업에 대한 IAM 역할을 생성합니다. 이 역할을 사용하면 AWS IoT가 디바이스 플릿 자산 및 해당 자산 계층 구조의 속성으로 데이터를 푸시할 수 있습니다.

- j. 역할 이름을 입력하고 생성을 선택합니다.
7. (선택 사항) 규칙 문제를 해결하는 데 사용할 수 있는 오류 작업을 구성합니다. 자세한 내용은 [규칙 문제 해결\(AWS IoT SiteWise\)](#) 단원을 참조하세요.
8. 다음을 선택합니다.
9. 설정을 검토하고 생성을 선택하여 규칙을 생성합니다.

8단계: 디바이스 클라이언트 스크립트 실행

이 자습서에서는 실제 디바이스를 사용하여 데이터를 보고하지 않습니다. 대신 스크립트를 실행하여 실제 센서 데이터를 모방하기 위해 CPU 및 메모리 사용량으로 AWS IoT 사물의 디바이스 새도를 업데이트합니다. 스크립트를 실행하려면 먼저 필요한 Python 패키지를 설치해야 합니다. 이 절차에서는 필요한 Python 패키지를 설치한 다음 디바이스 클라이언트 스크립트를 실행합니다.

디바이스 클라이언트 스크립트 구성 및 실행

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창 하단에서 설정을 선택합니다.
3. 디바이스 클라이언트 스크립트와 함께 사용할 사용자 지정 엔드포인트를 저장합니다. 이 엔드포인트를 사용하여 사물의 새도우와 상호 작용할 수 있습니다. 이 엔드포인트는 현재 리전의 계정에 고유합니다.

사용자 지정 엔드포인트는 다음 예제와 같아야 합니다.

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

4. 명령줄을 열고 다음 명령을 실행하여 이전에 생성한 자습서 디렉터리로 이동합니다.

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

5. 다음 명령을 실행하여 AWS IoT Device SDK for Python를 설치합니다.

```
pip3 install AWSIoTPythonSDK
```

자세한 내용은 AWS IoT 개발자 가이드의 [AWS IoT Device SDK for Python](#) 단원을 참조하세요.

6. 다음 명령을 실행하여 교차 플랫폼 프로세스 및 시스템 유틸리티 라이브러리인 psutil을 설치합니다.

```
pip3 install psutil
```

자세한 내용은 Python 패키지 인덱스의 [psutil](#)을 참조하세요.

7. `iot-sitewise-rule-tutorial` 디렉터리에서 `thing_performance.py`이라는 파일을 생성한 후 다음 Python 코드를 파일에 복사합니다.

```
import AWSIoTPythonSDK.MQTTLib as AWSIoTPyMQTT

import json
import psutil
import argparse
import logging
import time

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

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

```
# An MQTT shadow client that uploads device performance data to AWS IoT at a
regular interval.
class PerformanceShadowClient:
    def __init__(
        self,
        thingName,
        host,
        port,
        rootCAPath,
        privateKeyPath,
        certificatePath,
        requestDelay,
    ):
        self.thingName = thingName
        self.host = host
        self.port = port
        self.rootCAPath = rootCAPath
        self.privateKeyPath = privateKeyPath
        self.certificatePath = certificatePath
        self.requestDelay = requestDelay

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

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

```

        time.sleep(args.requestDelay)

# Configures the MQTT shadow client for this thing.
def configureMQTTClient(self):
    mqttClient = AWSIoTPyMQTT.AWSIoTMQTTShadowClient(self.thingName)
    mqttClient.configureEndpoint(self.host, self.port)
    mqttClient.configureCredentials(
        self.rootCAPath, self.privateKeyPath, self.certificatePath
    )
    mqttClient.configureAutoReconnectBackoffTime(1, 32, 20)
    mqttClient.configureConnectDisconnectTimeout(10)
    mqttClient.configureMQTTOperationTimeout(5)
    return mqttClient

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

# Prints the result of a shadow update call.
def shadowUpdateCallback(self, payload, responseStatus, token):
    print("{} {}".format(self.thingName, token))
    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. 다음 파라미터가 있는 명령줄에서 `thing_performance.py`를 실행합니다.

- `-n, --thingName` - 사물 이름(예: **SiteWiseTutorialDevice1**)입니다.
- `-e, --endpoint`-이 절차의 앞부분에서 저장한 사용자 지정 AWS IoT 엔드포인트입니다.
- `-r, --rootCA`- AWS IoT 루트 CA 인증서의 경로입니다.
- `-c, --cert`- AWS IoT 사물 인증서의 경로입니다.
- `-k, --key`- AWS IoT 사물 인증서 프라이빗 키의 경로입니다.
- `-d, --requestDelay` - (선택 사항) 각 디바이스 새도우 업데이트 사이에 대기할 시간(초)입니다. 기본값은 1초입니다.
- `-v, --enableLogging` - (선택 사항) 이 파라미터가 있으면 스크립트는 AWS IoT Device SDK for Python에서 디버그 메시지를 인쇄합니다.

명령은 다음 예제와 비슷해야 합니다.

```

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

```

추가 AWS IoT 사물에 대해 스크립트를 실행하는 경우 사물 이름과 인증서 디렉토리를 적절히 업데이트합니다.

9. 디바이스에서 프로그램을 열고 닫으면 CPU 및 메모리 사용량이 어떻게 변하는지 확인할 수 있습니다. 스크립트는 각 CPU 및 메모리 사용량 읽기를 인쇄합니다. 스크립트가 디바이스 새도우 서비스에 데이터를 성공적으로 업로드하면 스크립트의 출력은 다음 예제와 같아야 합니다.


```
[SiteWiseTutorialDevice1]
```

```
CPU:      24.6%
```

```
Memory: 85.2%
```

```
[SiteWiseTutorialDevice1]
```

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

10. 스크립트가 디바이스 새도우를 업데이트하는지 확인하려면 다음 단계를 따르세요.

- [AWS IoT 콘솔](#)로 이동합니다.
- 왼쪽 탐색 창에서 모든 디바이스스를 선택한 후 사물을 선택합니다.
- SiteWiseTutorialDevice를 선택하세요.
- 디바이스 새도우 탭을 선택하고 클래식 새도우를 선택한 다음 새도우 상태가 다음 예와 같은지 확인합니다.

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

사물의 새도우 상태가 비어 있거나 이전 예제처럼 보이지 않는 경우 스크립트가 실행 중이고 성공적으로 연결되어 있는지 확인합니다 AWS IoT. 연결할 때 스크립트가 계속 시간 초과되면 [사물 정책](#)이이 자습서에 따라 구성되어 있는지 AWS IoT 확인합니다.

11. 규칙 작업에서 데이터를 AWS IoT SiteWise에 전송하는지 확인하려면 다음 단계를 따르세요.

- [AWS IoT SiteWise 콘솔](#)로 이동합니다.
- 왼쪽 탐색 창에서 자산을 선택합니다.
- 디바이스 플릿 자산(SiteWise Tutorial Device Fleet 1 1) 옆에 있는 화살표를 선택하여 자산 계층 구조를 확장한 다음 디바이스 자산(SiteWise Tutorial Device 1)을 선택합니다.
- 측정을 선택합니다.
- 최신 값 셀에 CPU Usage 및 Memory Usage 속성에 대한 값이 있는지 확인합니다.

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. CPU Usage 및 Memory Usage 속성에 최신 값이 없으면 페이지를 새로 고칩니다. 몇 분 후에도 값이 나타나지 않는 경우 [규칙 문제 해결\(AWS IoT SiteWise\)](#) 단원을 참조하세요.

이 자습서를 완료했습니다. 데이터의 실시간 시각화를 탐색하려는 경우 AWS IoT SiteWise Monitor에서 포털을 구성할 수 있습니다. 자세한 내용은 [블 사용하여 데이터 모니터링 AWS IoT SiteWise Monitor](#) 단원을 참조하십시오. 그렇지 않으면 명령 프롬프트에서 CTRL+C를 눌러 디바이스 클라이언트 스크립트를 중지할 수 있습니다. Python 프로그램으로 전송되는 메시지로 요금이 발생할 가능성은 적지만 사용 후에는 프로그램을 중지시키는 것이 좋습니다.

9단계: 자습서 완료 후 리소스 정리

AWS IoT 사물에서 데이터를 수집하는 방법에 대한 자습서를 완료한 후 추가 요금이 발생하지 않도록 리소스를 정리합니다.

에서 계층적 자산을 삭제하려면 AWS IoT SiteWise

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 자산을 선택합니다.
3. 에서 자산을 삭제할 때는 먼저 자산의 연결을 해제해야 AWS IoT SiteWise합니다.

디바이스 자산을 디바이스 플릿 자산에서 분리하려면 다음 단계를 완료하세요.

- a. 디바이스 플릿 자산(SiteWise Tutorial Device Fleet 1)을 선택합니다.
- b. 편집을 선택합니다.
- c. 이 자산에 연결된 자산 아래에서 이 디바이스 플릿 자산과 연결된 각 디바이스 자산에 대해 연결 해제를 선택합니다.
- d. 저장을 선택합니다.

이제 디바이스 자산이 더 이상 계층 구조로 구성되어 있지 않음을 확인할 수 있습니다.

4. 디바이스 자산(SiteWise Tutorial Device 1)을 선택합니다.
5. 삭제를 선택합니다.

6. 확인 대화 상자에 **Delete**를 입력한 다음 삭제를 선택합니다.
7. 각 디바이스 자산 및 디바이스 플릿 자산(SiteWise Tutorial Device Fleet 1)에 대해 단계 4~6을 반복합니다.

에서 계층적 자산 모델을 삭제하려면 AWS IoT SiteWise

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 아직 디바이스 자산 및 디바이스 플릿 자산을 삭제하지 않았다면 삭제하세요. 자세한 내용은 [이전 절차](#)를 참조하세요. 해당 모델에서 생성된 자산이 있으면 모델을 삭제할 수 없습니다.
3. 왼쪽 탐색 창에서 모델을 선택합니다.
4. 디바이스 플릿 자산 모델(SiteWise Tutorial Device Fleet Model)을 선택합니다.

계층적 자산 모델을 삭제할 때는 상위 자산 모델을 삭제하는 것으로 시작합니다.

5. 삭제를 선택합니다.
6. 확인 대화 상자에 **Delete**를 입력한 다음 삭제를 선택합니다.
7. 디바이스 자산 모델(SiteWise Tutorial Device Model)에 대해 단계 4~6을 반복합니다.

에서 규칙을 비활성화하거나 삭제하려면 AWS IoT Core

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 메시지 라우팅을 선택한 다음 규칙을 선택합니다.
3. 규칙을 선택하고 삭제를 선택합니다.
4. 확인 대화 상자에서 규칙 이름을 입력한 다음 삭제를 선택합니다.

SiteWise Monitor에서 전력 발전소 데이터 시각화 및 공유

이 자습서에서는 AWS IoT SiteWise Monitor 사용하여 포털이라고 하는 관리형 웹 애플리케이션을 통해 산업 데이터를 시각화하고 공유하는 방법을 설명합니다. 각 포털은 프로젝트를 포괄하므로 각 프로젝트 내에서 액세스할 수 있는 데이터를 유연하게 선택할 수 있습니다. 그런 다음 각 포털에 액세스할 수 있는 조직 내 사용자를 지정합니다. 사용자는 AWS IAM Identity Center 계정을 사용하여 포털에 로그인하므로 기존 ID 저장소 또는 AWS가 관리하는 저장소를 사용할 수 있습니다.

관리자와 충분한 권한이 있는 사용자는 각 프로젝트에서 대시보드를 생성하여 산업 데이터를 의미 있는 방식으로 시각화할 수 있습니다. 그런 다음 사용자가 이러한 대시보드를 통해 데이터에 대한 통찰력을 빠르게 확보하고 운영을 모니터링할 수 있습니다. 회사의 모든 사용자에게 각 프로젝트에 대한 관리

권한이나 읽기 전용 권한을 구성할 수 있습니다. 자세한 내용은 [를 사용하여 데이터 모니터링 AWS IoT SiteWise Monitor](#) 단원을 참조하십시오.

이 자습서에서는 AWS IoT SiteWise 데모를 개선하여 풍력 발전 단지용 샘플 데이터 세트를 제공합니다. SiteWise Monitor에서 포털을 구성하고 프로젝트 및 대시보드를 생성하여 풍력 발전단지 데이터를 시각화합니다. 자습서에서는 프로젝트 및 관련 대시보드를 소유하거나 볼 수 있는 권한 할당과 함께 추가 사용자 생성도 다룹니다.

Note

SiteWise Monitor를 사용하는 경우 포털에 로그인하는 사용자당(월별) 요금이 청구됩니다. 이 자습서에서는 3명의 사용자를 생성하지만 한 명의 사용자만 로그인하면 됩니다. 이 자습서를 완료하면 한 명의 사용자에 대한 요금이 청구됩니다. 자세한 내용은 [AWS IoT SiteWise 요금](#)을 참조하세요.

주제

- [사전 조건](#)
- [1단계: SiteWise Monitor에서 포털 생성](#)
- [2단계: 포털에 로그인](#)
- [3단계: 풍력 발전소 프로젝트 생성](#)
- [4단계: 풍력 발전소 데이터를 시각화하는 대시보드 생성](#)
- [5단계: 포털 살펴보기](#)
- [6단계: 자습서 완료 후 리소스 정리](#)

사전 조건

이 자습서를 완료하려면 다음이 필요합니다.

- AWS 계정. 없으면 [AWS 계정 설정](#) 단원을 참조하세요.
- Windows, Linux, 또는를 실행하여 macOS에 액세스하는 개발 컴퓨터입니다 AWS Management Console. 자세한 내용은 [AWS Management Console이란 무엇인가요?](#)를 참조하세요.
- 관리자 권한이 있는 AWS Identity and Access Management (IAM) 사용자.
- 실행 중인 AWS IoT SiteWise 풍력 발전소 데모입니다. 데모를 설정하면서 모델과 자산을 정의하고 풍력 발전소를 나타내기 위해 해당 모델 AWS IoT SiteWise 과 자산으로 데이터를 스트리밍합니다. 자세한 내용은 [AWS IoT SiteWise 데모 사용](#) 단원을 참조하십시오.

- 계정에서 IAM Identity Center를 활성화한 경우 AWS Organizations 관리 계정에 로그인합니다. 자세한 내용을 알아보려면 [AWS Organizations 용어 및 개념](#)을 참조하세요. IAM Identity Center를 활성화하지 않은 경우 이 자습서에서 활성화하고 계정을 관리 계정으로 설정해야 합니다.

AWS Organizations 관리 계정에 로그인할 수 없는 경우 조직에 IAM Identity Center 사용자가 있는 자습서를 부분적으로 완료할 수 있습니다. 이 경우 포털과 대시보드를 생성할 수 있지만 프로젝트에 할당할 새 IAM Identity Center 사용자를 생성할 수는 없습니다.

1단계: SiteWise Monitor에서 포털 생성

이 절차에서는 AWS IoT SiteWise Monitor에서 포털을 생성합니다. 각 포털은 관리되는 웹 애플리케이션으로, 사용자가 AWS IAM Identity Center 계정으로 로그인할 수 있습니다. IAM Identity Center를 사용하면 회사의 기존 ID 저장소를 사용하거나 AWS에서 관리하는 ID 저장소를 만들 수 있습니다. 회사 직원은 별도의 AWS 계정을 생성하지 않고도 로그인할 수 있습니다.

포털 생성

1. [AWS IoT SiteWise 콘솔](#)에 로그인합니다.
2. 가 AWS IoT SiteWise 지원되는 [AWS IoT SiteWise 엔드포인트 및 할당량](#)을 검토하고 필요한 경우 리전을 전환합니다. 동일한 리전에서 AWS IoT SiteWise 데모를 실행해야 합니다.
3. 왼쪽 탐색 창에서 포털을 선택합니다.
4. 포털 생성을 선택합니다.
5. IAM Identity Center를 이미 활성화한 경우 6단계로 건너뛰세요. 그렇지 않은 경우 다음 단계를 완료하여 IAM Identity Center를 활성화합니다.
 - a. 활성화 AWS IAM Identity Center (SSO) 페이지에서 이메일 주소, 이름 및 성을 입력하여 포털 관리자가 될 IAM Identity Center 사용자를 생성합니다. 새 IAM Identity Center 사용자의 암호를 설정할 때 이메일을 받을 수 있도록 액세스할 수 있는 이메일 주소를 사용합니다.

포털에서 포털 관리자는 프로젝트를 생성하고 사용자를 프로젝트에 할당합니다. 나중에 더 많은 사용자를 만들 수 있습니다.

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Enable SSO

Step 2
Portal configuration

Step 3
Invite administrators

Step 4
Assign users

Enable AWS Single Sign-On (SSO)

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

Create a user

Email address
john.doe@example.com

First name
John

Last name
Doe

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

Cancel **Create user**

- b. 사용자 생성을 선택합니다.
6. Portal configuration(포털 구성) 페이지에서 다음 단계를 완료합니다.
 - a. 포털의 이름(예: **WindFarmPortal**)을 입력합니다.
 - b. (선택 사항) 포털에 대한 설명을 입력합니다. 포털이 여러 개인 경우 의미 있는 설명을 사용하여 각 포털에 포함된 내용을 추적합니다.
 - c. (선택 사항) 이미지를 업로드하여 포털에 표시합니다.
 - d. 포털에 문제가 있을 때 포털 사용자가 연락할 수 있는 이메일 주소를 입력하고 문제를 해결하기 위해 회사 AWS 관리자의 도움이 필요합니다.
 - e. 포털 생성을 선택합니다.
 7. 관리자 초대 페이지에서 IAM Identity Center 사용자를 포털에 관리자로 할당할 수 있습니다. 포털 관리자는 포털 내에서 권한과 프로젝트를 관리합니다. 이 페이지에서 다음을 수행합니다.
 - a. 포털 관리자가 될 사용자를 선택합니다. 이 자습서의 앞 부분에서 IAM Identity Center를 활성화한 경우 생성한 사용자를 선택하세요.

- b. (선택 사항) 선택한 사용자에게 초대장 보내기를 선택합니다. 이메일 클라이언트가 열리고 메시지 본문에 초대장이 표시됩니다. 이메일을 포털 관리자에게 보내기 전에 사용자 지정할 수 있습니다. 나중에 포털 관리자에게 이메일을 보낼 수도 있습니다. SiteWise Monitor를 처음 사용하고, 포털 관리자가 될 예정이라면 본인에게는 이메일을 보내지 않아도 됩니다.
 - c. Next(다음)를 선택합니다.
8. 사용자 할당 페이지에서 IAM Identity Center 사용자를 포털에 할당할 수 있습니다. 포털 관리자는 나중에 포털 사용자를 프로젝트 소유자 또는 뷰어로 할당할 수 있습니다. 프로젝트 소유자는 프로젝트에 대시보드를 생성할 수 있습니다. 프로젝트 뷰어는 자신에게 배정된 프로젝트에 대한 읽기 전용 액세스 권한을 가집니다. 이 페이지에서 IAM Identity Center 사용자를 생성하여 포털에 추가할 수 있습니다.

Note

AWS Organizations 관리 계정에 로그인하지 않은 경우 IAM Identity Center 사용자를 생성할 수 없습니다. 사용자 할당을 선택하여 포털 사용자 없이 포털을 생성한 다음 이 단계를 건너뛰세요.

이 페이지에서 다음을 수행합니다.

- a. 다음 단계를 두 번 완료하여 두 개의 IAM Identity Center 사용자를 생성합니다.

- i. 사용자 생성을 선택하여 새 사용자에게 대한 세부 정보를 입력할 수 있는 대화 상자를 엽니다.
- ii. 새 사용자의 이메일 주소, 이름, 성을 입력합니다. IAM Identity Center는 사용자에게 암호 설정을 위한 이메일을 보냅니다. 이러한 사용자로 포털에 로그인하려면 액세스할 수 있는 이메일 주소를 선택하세요. 각 이메일 주소는 고유해야 합니다. 사용자는 이메일 주소를 사용자 이름으로 사용하여 포털에 로그인합니다.

Create user [X]

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

Email address
mary.major@example.com

First name
Mary

Last name
Major

Cancel Create user

- iii. 사용자 생성을 선택합니다.
- b. 이전 단계에서 생성한 두 명의 IAM Identity Center 사용자를 선택합니다.

AWS IoT SiteWise > Monitor > Portals > WindFarmPortal > Assign users

Assign users

Users (3) [Create user]

Find resources

	Display name	Email
☐	John Doe	john.doe@example.com
☒	Mary Major	mary.major@example.com
☒	Mateo Jackson	mateo.jackson@example.com

► Selected users (2)

Cancel Assign users

- c. 사용자 할당을 선택하여 해당 사용자를 포털에 추가합니다.

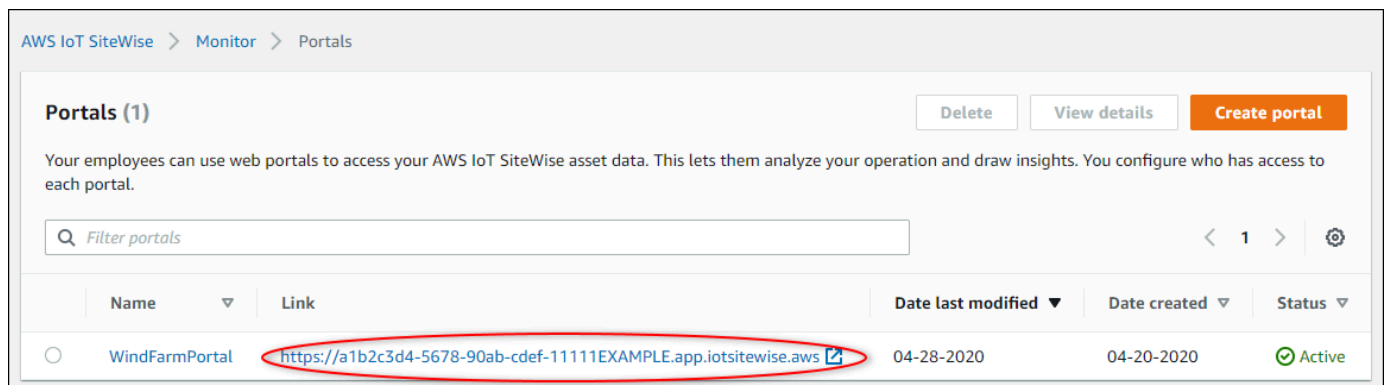
포털 페이지가 열리고 새 포털이 나열됩니다.

2단계: 포털에 로그인

이 절차에서는 포털에 추가한 AWS IAM Identity Center 사용자를 사용하여 새 포털에 로그인합니다.

포털 로그인

1. 포털 페이지에서 새 포털의 링크를 선택하여 새 탭에서 포털을 엽니다.



2. 자습서의 앞부분에서 첫 번째 IAM Identity Center 사용자를 생성한 경우 다음 단계를 따라 사용자의 암호를 생성합니다.
 - a. 이메일을 확인해 제목이 Invitation to join AWS IAM Identity Center인 이메일을 찾습니다.
 - b. 해당 초대 이메일을 열고 Accept invitation을 선택합니다.
 - c. 새 창에서 IAM Identity Center 사용자의 암호를 설정합니다.

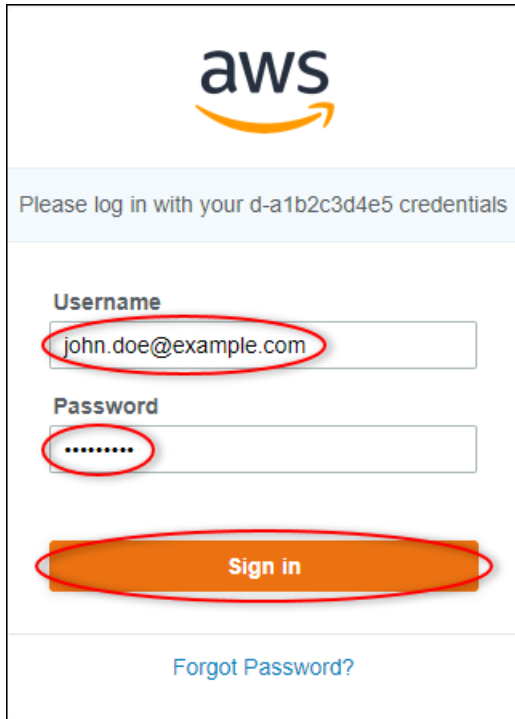
이전에 생성한 두 번째 및 세 번째 IAM Identity Center 사용자로 나중에 포털에 로그인하려는 경우 다음 단계를 완료하여 해당 사용자의 암호를 설정할 수도 있습니다.

Note

이메일을 받지 못한 경우 IAM Identity Center 콘솔에서 사용자의 암호를 생성할 수 있습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [최종 사용자의 IAM Identity Center 사용자 암호 재설정](#)을 참조하세요.

3. IAM Identity Center Username 및 Password를 입력합니다. 이 자습서의 앞 부분에서 IAM Identity Center 사용자를 생성한 경우 Username은 생성한 포털 관리자 사용자의 이메일 주소입니다.

포털 관리자를 포함한 모든 포털 사용자는 자신의 IAM Identity Center 사용자 자격 증명으로 로그인해야 합니다. 이러한 자격 증명은 일반적으로 AWS Management Console에 로그인하는 데 사용하는 자격 증명과 다릅니다.

The image shows the AWS IAM Identity Center login interface. At the top is the AWS logo. Below it, a message says "Please log in with your d-a1b2c3d4e5 credentials". There are two input fields: "Username" with the value "john.doe@example.com" and "Password" with masked characters "*****". Both fields are circled in red. Below the fields is an orange "Sign in" button, also circled in red. At the bottom, there is a blue link that says "Forgot Password?".

4. Sign in을 선택합니다.

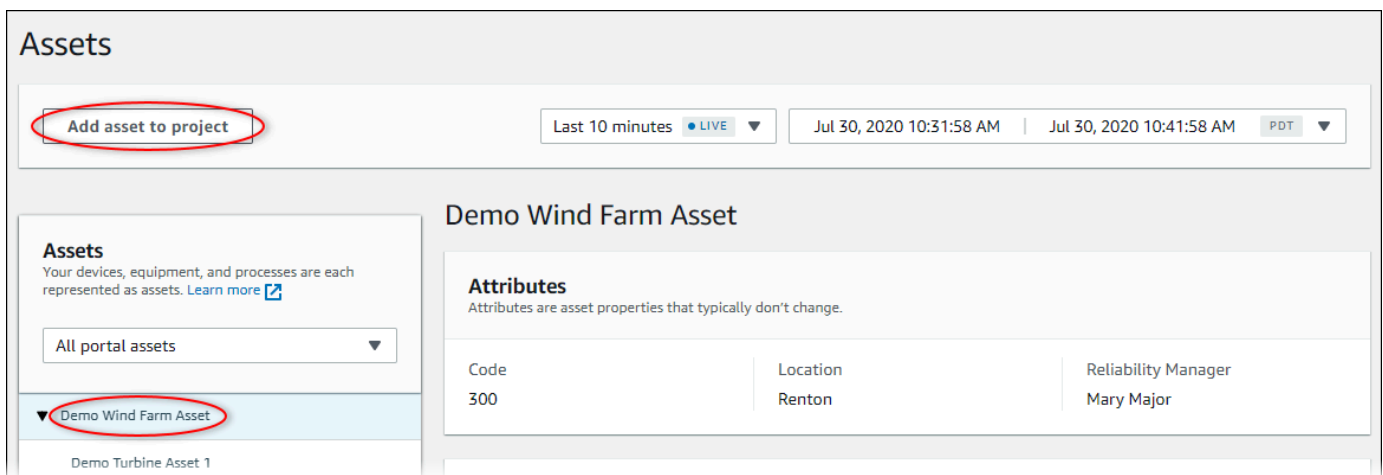
포털이 열립니다.

3단계: 풍력 발전소 프로젝트 생성

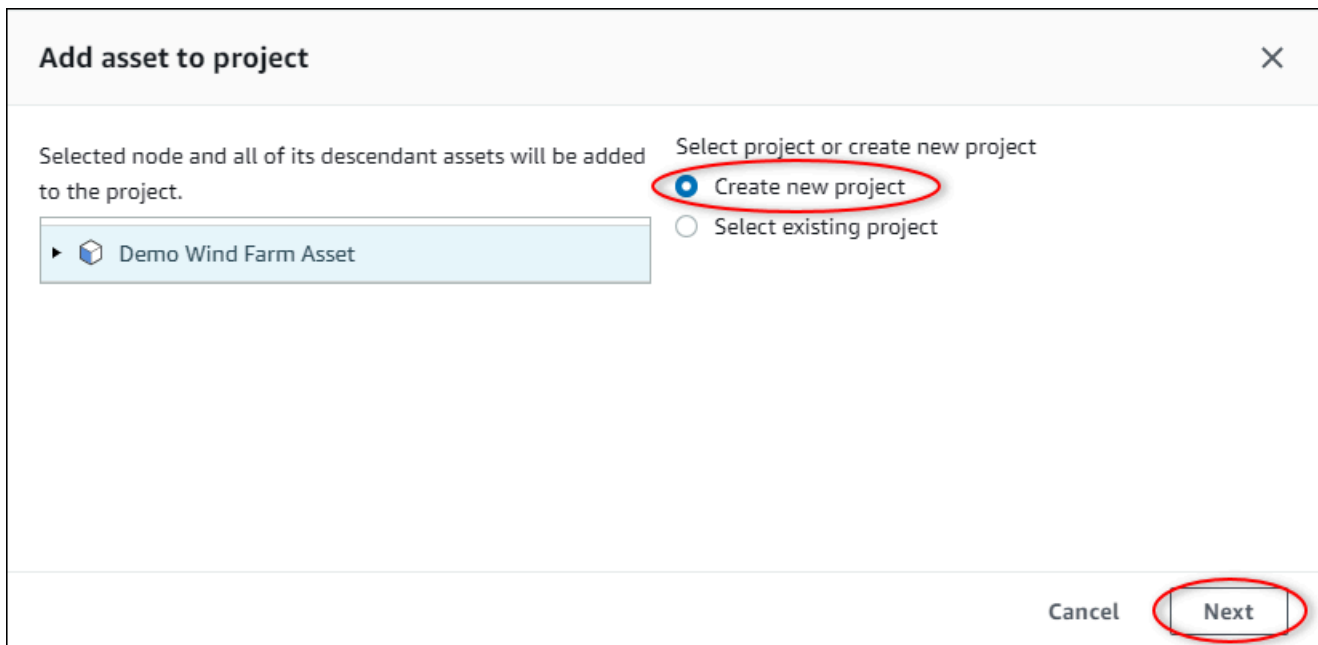
이 절차에서는 포털에 프로젝트를 생성합니다. 프로젝트는 권한, 자산 및 대시보드 집합을 정의한 리소스로, 프로젝트를 구성하여 해당 프로젝트에서 자산 데이터를 시각화할 수 있습니다. 프로젝트를 통해 각 운영 하위 집합에 액세스할 수 있는 사용자와 해당 운영 하위 집합의 데이터를 시각화하는 방법을 정의합니다. 포털 사용자를 각 프로젝트의 소유자나 뷰어로 할당할 수 있습니다. 프로젝트 소유자는 대시보드를 생성하여 데이터를 시각화하고 다른 사용자와 프로젝트를 공유할 수 있습니다. 프로젝트 뷰어는 대시보드를 볼 수 있지만 편집할 수는 없습니다. SiteWise Monitor의 역할에 대한 자세한 내용은 [SiteWise Monitor 역할](#) 단원을 참조하세요.

풍력 발전소 프로젝트 생성

1. 포털의 왼쪽 탐색 창에서 자산 탭을 선택합니다. 자산 페이지에서 포털에서 사용 가능한 모든 자산을 탐색하고 프로젝트에 자산을 추가할 수 있습니다.
2. 자산 브라우저에서 Demo Wind Farm Asset을 선택합니다. 자산을 선택하면 해당 자산의 실시간 데이터와 기록 데이터를 탐색할 수 있습니다. Shift 키를 눌러 여러 자산을 선택하고 해당 데이터를 나란히 비교할 수도 있습니다.
3. 왼쪽 상단에서 프로젝트에 자산 추가를 선택합니다. 프로젝트에는 포털 사용자가 데이터를 탐색하기 위해 볼 수 있는 대시보드가 포함되어 있습니다. 각 프로젝트에는 있는 자산의 하위 집합에 액세스할 수 있습니다 AWS IoT SiteWise. 프로젝트에 자산을 추가하면 해당 프로젝트에 대한 액세스 권한이 있는 모든 사용자가 해당 자산과 하위 자산에 대한 데이터에도 액세스할 수 있습니다.



4. 프로젝트에 자산 추가 대화 상자에서 새 프로젝트 생성을 선택하고 다음을 선택합니다.



5. 새 프로젝트 생성 대화 상자에서 프로젝트의 프로젝트 이름과 프로젝트 설명을 입력한 다음 프로젝트에 자산 추가를 선택합니다.

Create new project [X]

Project name

The project name can have up to 256 characters.

Project description

The project description can have up to 2048 characters.

Cancel Previous **Add asset to project**

새 프로젝트 페이지가 열립니다.

6. 프로젝트 페이지에서 포털 사용자를 이 프로젝트의 소유자 또는 뷰어로 추가할 수 있습니다.

Note

AWS Organizations 관리 계정에 로그인하지 않은 경우 이 프로젝트에 할당할 포털 사용자가 없을 수 있으므로 이 단계를 건너뛸 수 있습니다.

이 페이지에서 다음을 수행합니다.

- a. 프로젝트 소유자에서 소유자 추가 또는 사용자 편집을 선택합니다.

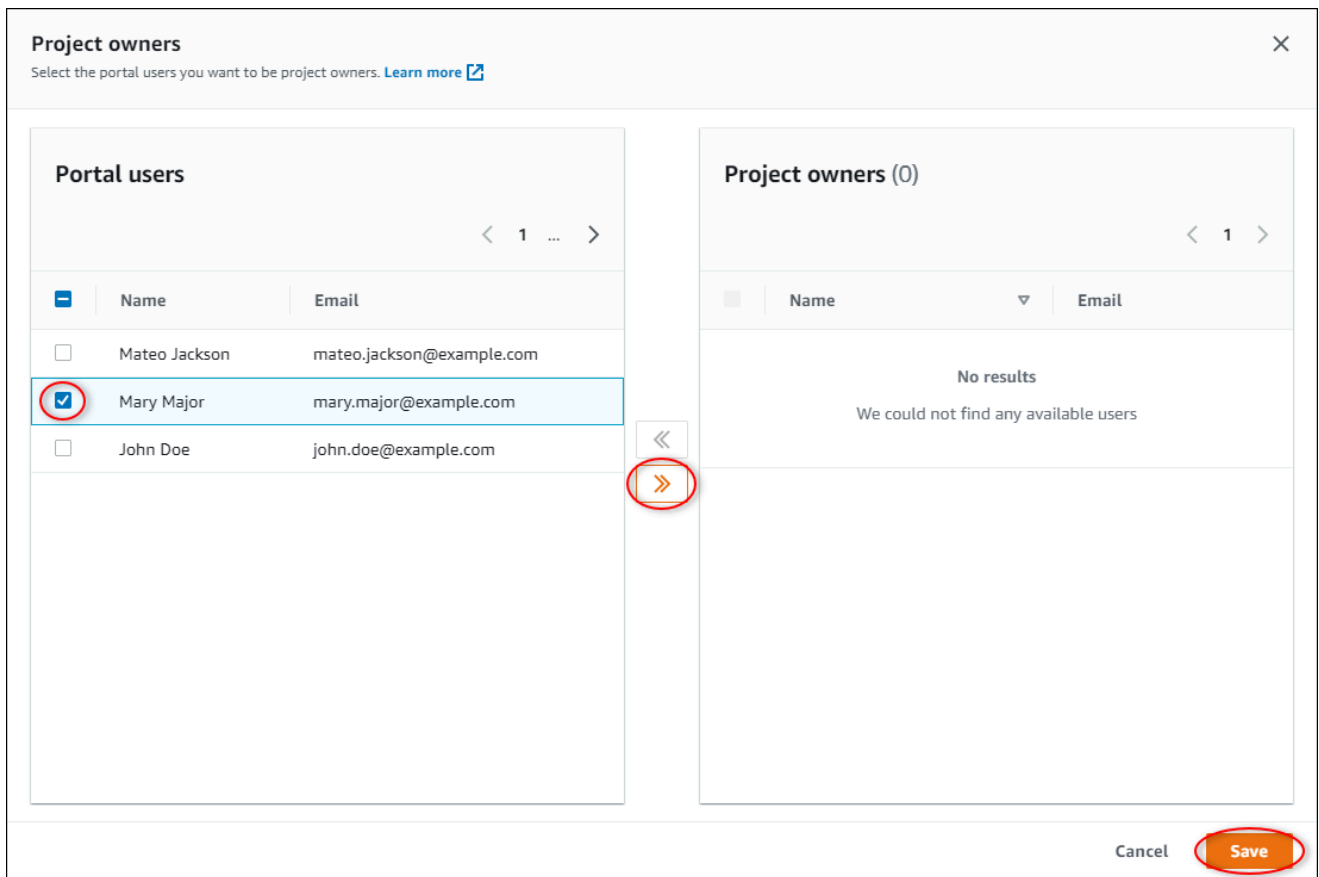
Project owners [Send invitations] [Remove owners] **Edit owners**

Project owners can create dashboards, view asset data, and invite other users to this project as owners or viewers.

< 1 >

Name	Email
You have not invited any other portal users to own this project. Project owners can modify and update dashboards and project viewers. Learn more Add owners	

- b. 프로젝트 소유자로 추가할 사용자(예: Mary Major)를 선택한 다음 >> 아이콘을 선택합니다.



- c. 저장(Save)을 선택합니다.

IAM Identity Center 사용자 Mary Major는 이 포털에 로그인하여 이 프로젝트의 대시보드를 편집하고 이 프로젝트를 이 포털의 다른 사용자와 공유할 수 있습니다.

- d. 프로젝트 뷰어에서 뷰어 추가 또는 사용자 편집을 선택합니다.
- e. 프로젝트 뷰어로 추가할 사용자(예: Mateo Jackson)를 선택한 다음 >> 아이콘을 선택합니다.
- f. 저장(Save)을 선택합니다.

IAM Identity Center 사용자 Mateo Jackson은 이 포털에 로그인하여 풍력 발전소 프로젝트의 대시보드를 볼 수 있지만 편집할 수는 없습니다.

4단계: 풍력 발전소 데이터를 시각화하는 대시보드 생성

이 절차에서는 데모 풍력 발전소 데이터를 시각화하는 대시보드를 생성합니다. 대시보드에는 프로젝트 자산 데이터의 사용자 지정 가능한 시각화가 포함되어 있습니다. 각 시각화는 선 차트, 막대 차트 또는 핵심 성과 지표(KPI) 표시와 같이 다양한 유형을 가질 수 있습니다. 데이터에 가장 적합한 시각화 유

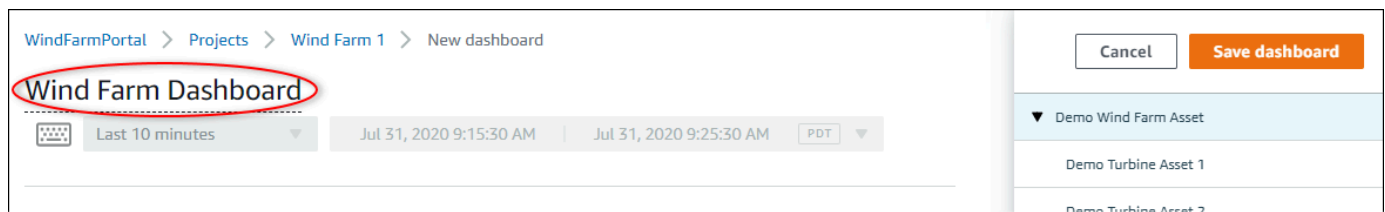
형을 선택할 수 있습니다. 프로젝트 소유자는 대시보드를 편집할 수 있고, 프로젝트 뷰어는 인사이트를 얻기 위해 대시보드를 볼 수만 있습니다.

시각화가 포함된 대시보드 생성

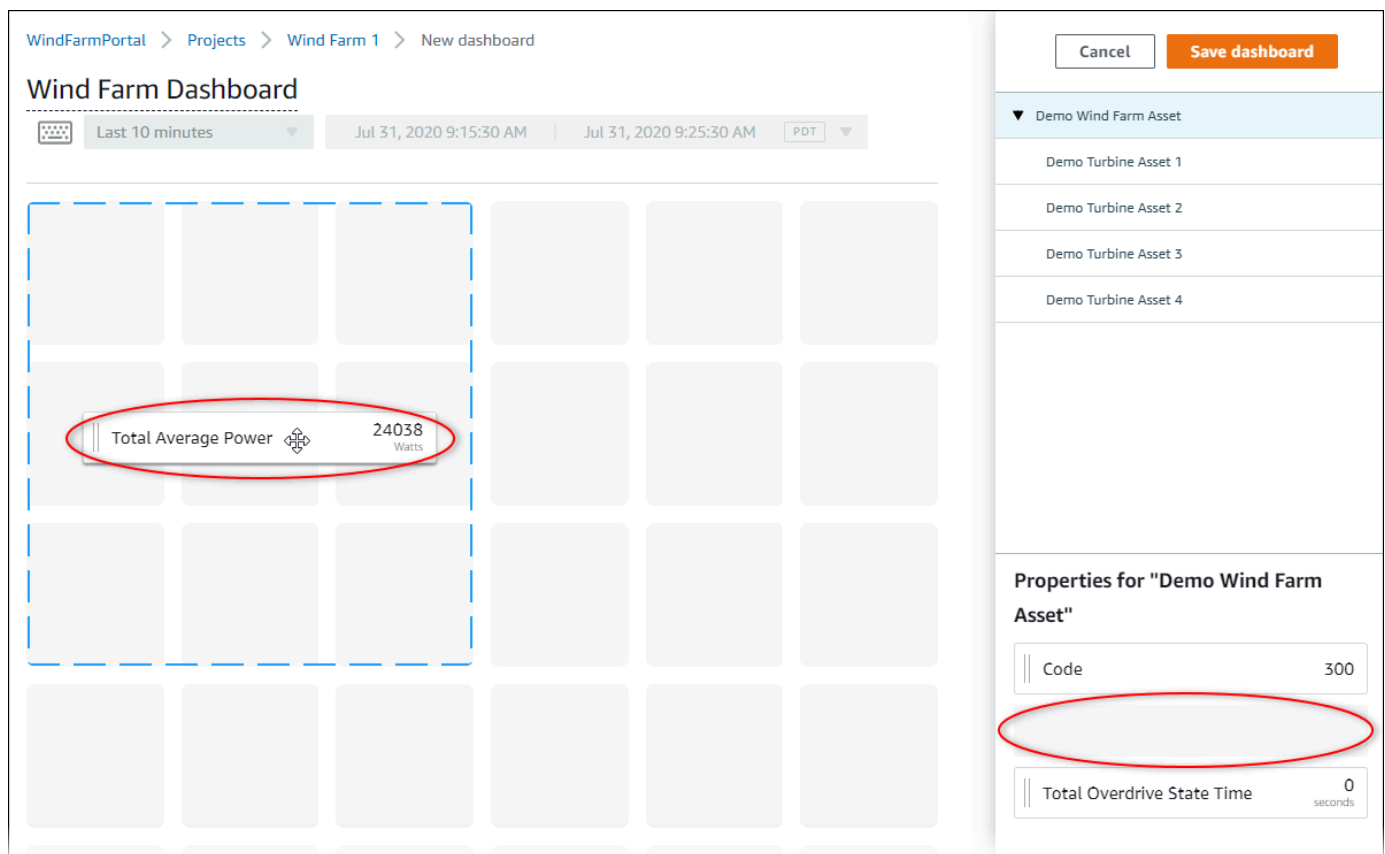
1. 새 프로젝트 페이지에서 대시보드 생성을 선택하여 대시보드를 만들고 편집 페이지를 엽니다.

대시보드의 편집 페이지에서 자산 계층 구조의 자산 속성을 대시보드로 끌어서 시각화를 만들 수 있습니다. 그런 다음 대시보드에서 각 시각화의 제목, 범례 제목, 유형, 크기 및 위치를 편집할 수 있습니다.

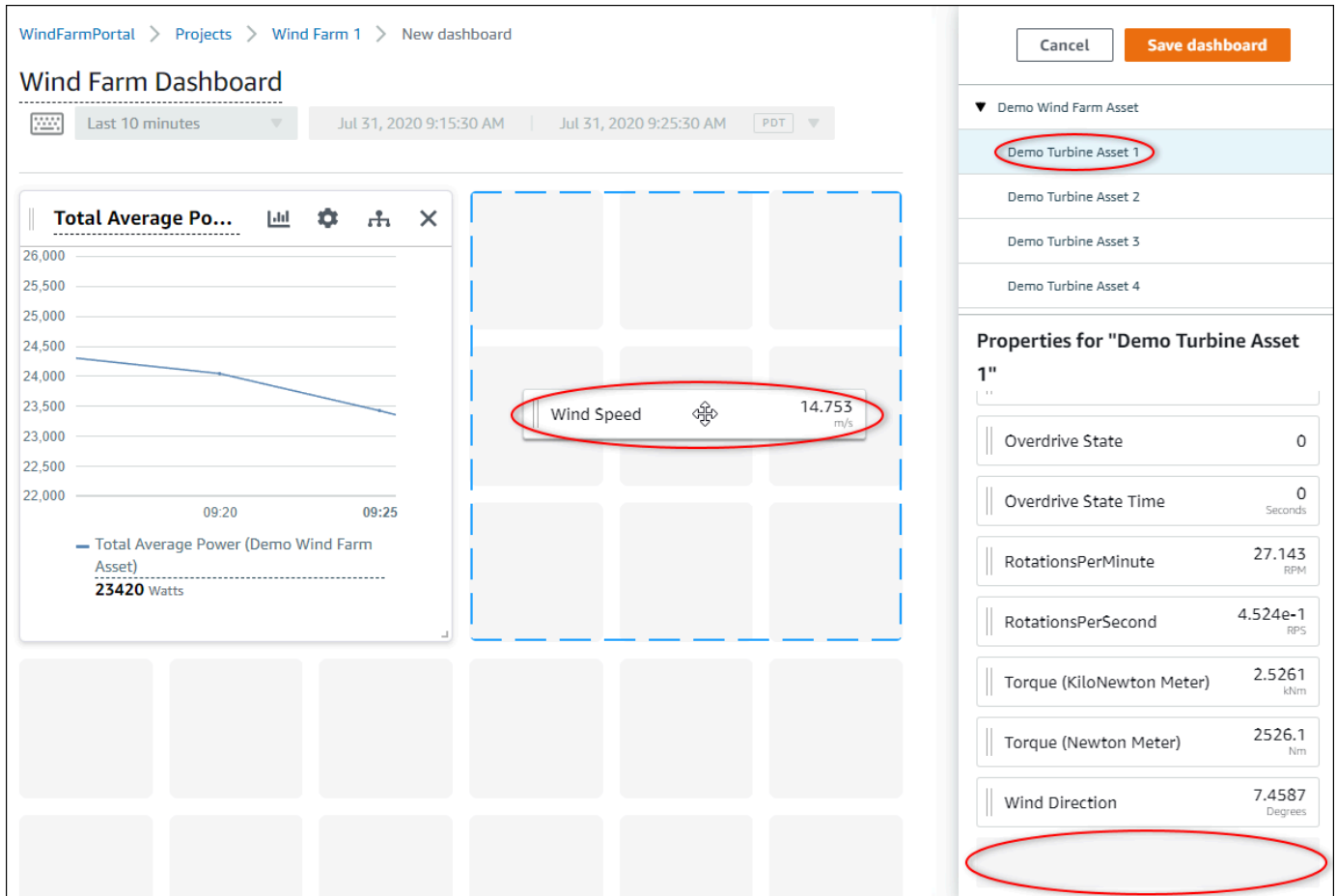
2. 대시보드 이름을 입력합니다.



3. Demo Wind Farm Asset에서 Total Average Power를 대시보드로 드래그하여 시각화를 생성합니다.

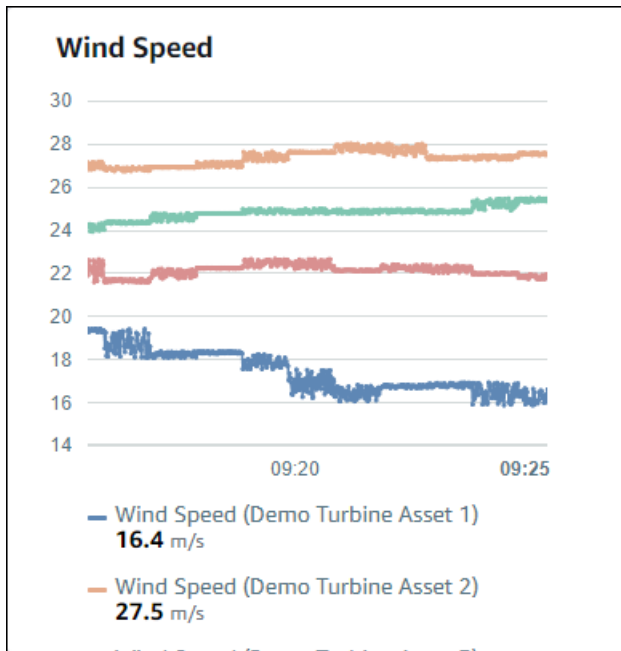


4. 해당 자산의 속성을 표시하도록 Demo Turbine Asset 1을 선택한 다음 대시보드로 Wind Speed를 드래그하여 풍속을 시각화할 수 있습니다.

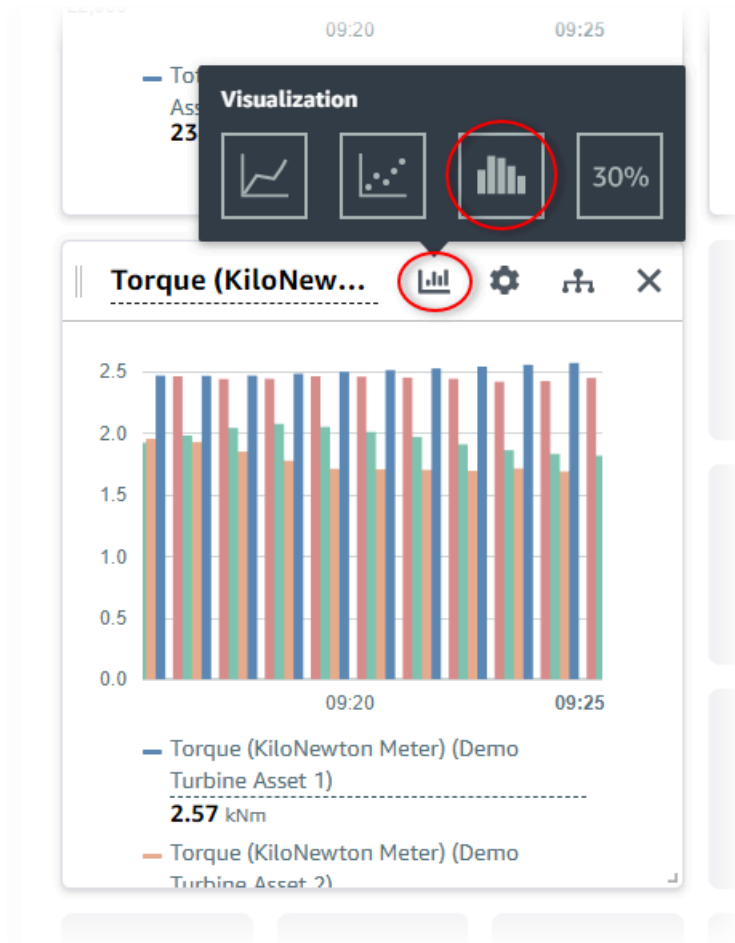


5. 각 Demo Turbine Asset 2, 3, 4(순서대로)에 대한 새 풍속 시각화에 Wind Speed를 추가합니다.

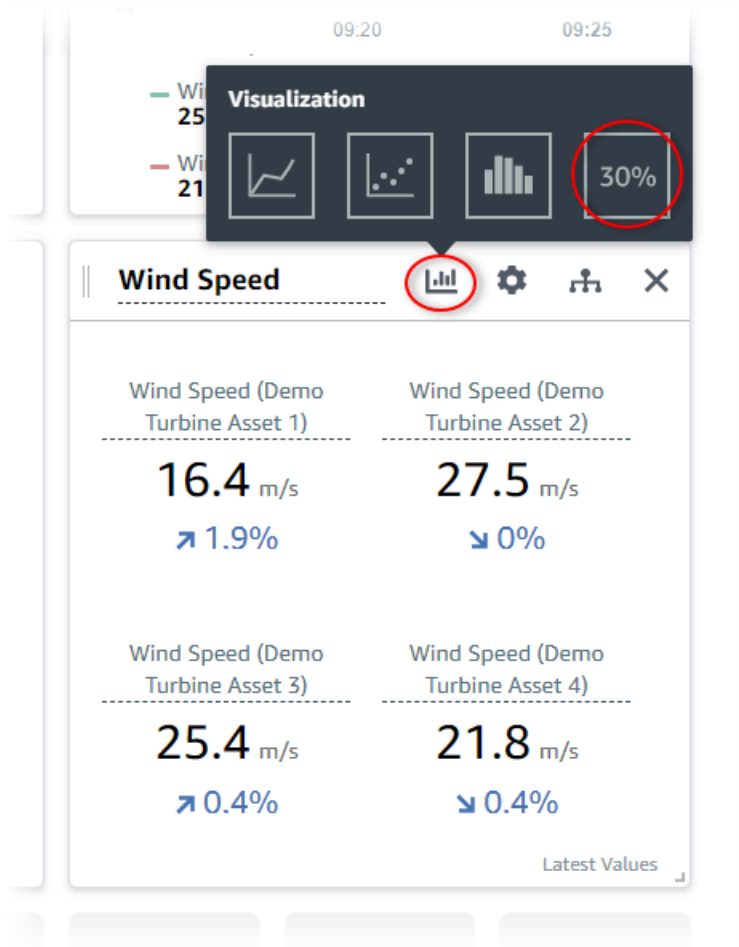
Wind Speed 시각화는 다음 스크린샷과 비슷한 모습이 되어야 합니다.



6. 풍력 터빈의 Torque (KiloNewton Meter) 속성에 대해 단계 4와 단계 5를 반복하여 풍력 터빈 토크에 대한 시각화를 만듭니다.
7. Torque (KiloNewton Meter) 시각화에 대한 시각화 유형 아이콘을 선택한 다음 막대 차트 아이콘을 선택합니다.

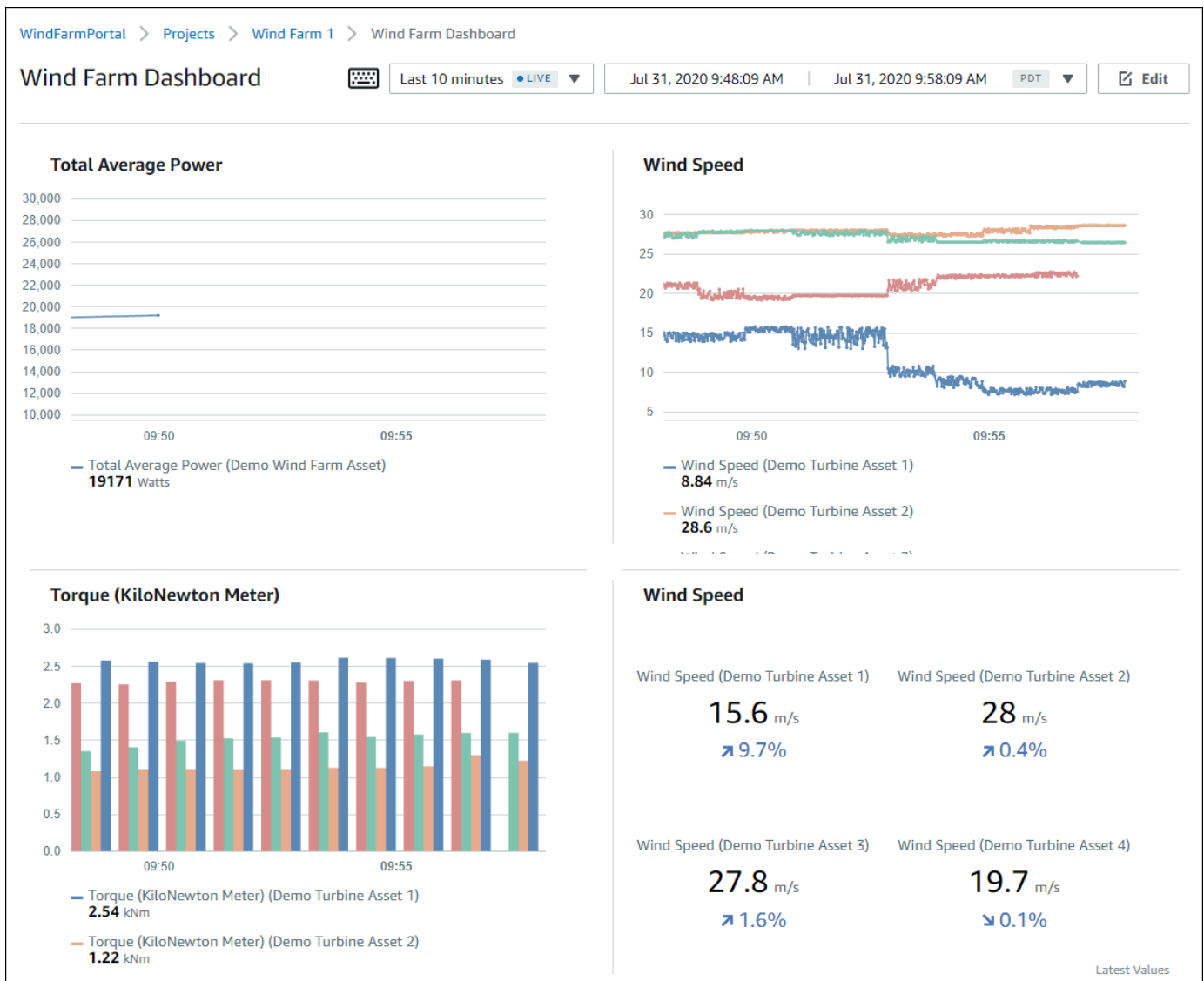


8. 풍력 터빈의 Wind Direction 속성에 대해 단계 4와 단계 5를 반복하여 풍향에 대한 시각화를 생성합니다.
9. Wind Direction 시각화에 대한 시각화 유형 아이콘을 선택한 다음 KPI 차트 아이콘(30%)을 선택합니다.



10. (선택 사항) 필요에 따라 각 시각화의 제목, 범례 제목, 유형, 크기 및 위치를 변경합니다.
11. 오른쪽 상단에 있는 대시보드 저장을 선택하여 대시보드를 저장합니다.

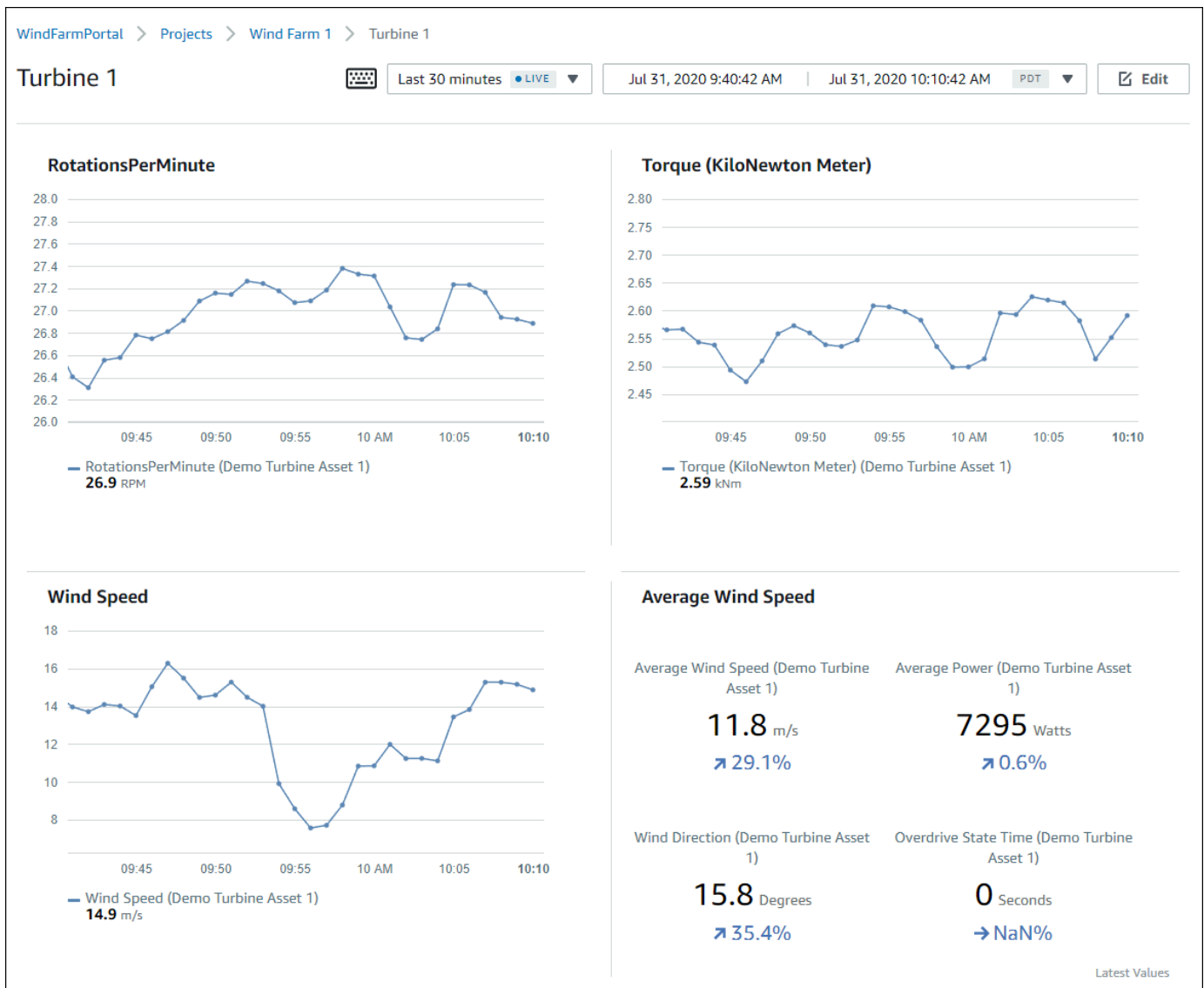
대시보드는 다음 스크린샷과 비슷한 모습이 되어야 합니다.



12. (선택 사항) 각 풍력 터빈 자산에 대해 추가 대시보드를 만듭니다.

프로젝트 뷰어가 각 개별 자산의 문제를 조사할 수 있도록 각 자산에 대한 대시보드를 만드는 것이 좋습니다. 각 시각화에는 최대 5개의 자산만 추가할 수 있으므로 다양한 시나리오에서 계층적 자산에 대한 여러 대시보드를 만들어야 합니다.

데모 풍력 터빈의 대시보드는 다음 스크린샷과 유사하게 보일 수 있습니다.



13. (선택 사항) 타임라인을 변경하거나 시각화에서 데이터 포인트를 선택하여 대시보드의 데이터를 탐색합니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 가이드의 [대시보드 보기](#)를 참조하세요.

5단계: 포털 살펴보기

이 절차에서 AWS IoT SiteWise 포털 관리자보다 권한이 적은 사용자로 포털을 탐색할 수 있습니다.

포털 탐색 및 자습서 완료

- (선택 사항) 다른 사용자를 프로젝트에 소유자 또는 뷰어로 추가한 경우 해당 사용자로 포털에 로그인할 수 있습니다. 이렇게 하면 포털 관리자보다 권한이 적은 사용자로 포털을 탐색할 수 있습니다.

⚠ Important

포털에 로그인하는 각 사용자에게 대해 요금이 부과됩니다. 자세한 내용은 [AWS IoT SiteWise 요금](#)을 참조하세요.

다른 사용자와 마찬가지로 포털을 탐색하려면 다음을 수행합니다.

- a. 포털 왼쪽 하단에서 로그아웃을 선택하여 웹 애플리케이션을 종료합니다.
- b. IAM Identity Center 애플리케이션 포털의 오른쪽 상단에서 로그아웃을 선택하여 IAM Identity Center 사용자에서 로그아웃합니다.
- c. 프로젝트 소유자 또는 프로젝트 뷰어로 할당한 IAM Identity Center 사용자로 포털에 로그인합니다. 자세한 내용은 [2단계: 포털에 로그인](#) 단원을 참조하세요.

이 자습서를 완료했습니다. SiteWise Monitor에서 데모 풍력 발전 발전소 탐색을 마치면 다음 절차에 따라 리소스를 정리하세요.

6단계: 자습서 완료 후 리소스 정리

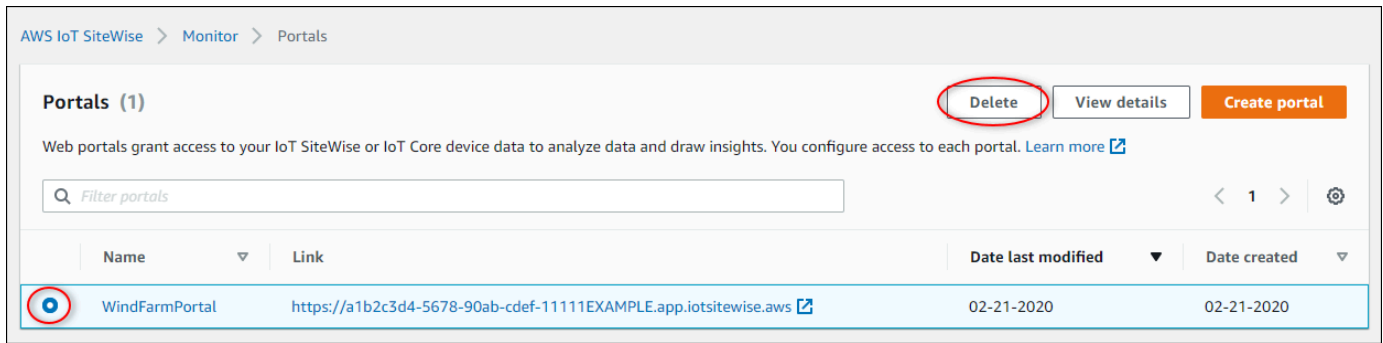
자습서를 완료한 후 리소스를 정리할 수 있습니다. 사용자가 포털에 로그인하지 않으면 AWS IoT SiteWise 에 대한 요금이 청구되지 않지만 포털과 AWS IAM Identity Center 디렉터리 사용자를 삭제할 수 있습니다. 데모 풍력 발전소 자산은 데모를 생성할 때 선택한 기간이 끝날 때 삭제되며, 데모를 수동으로 삭제할 수도 있습니다. 자세한 내용은 [AWS IoT SiteWise 데모 삭제](#) 단원을 참조하세요.

포털 및 IAM Identity Center 사용자를 삭제하려면 다음 절차를 수행하세요.

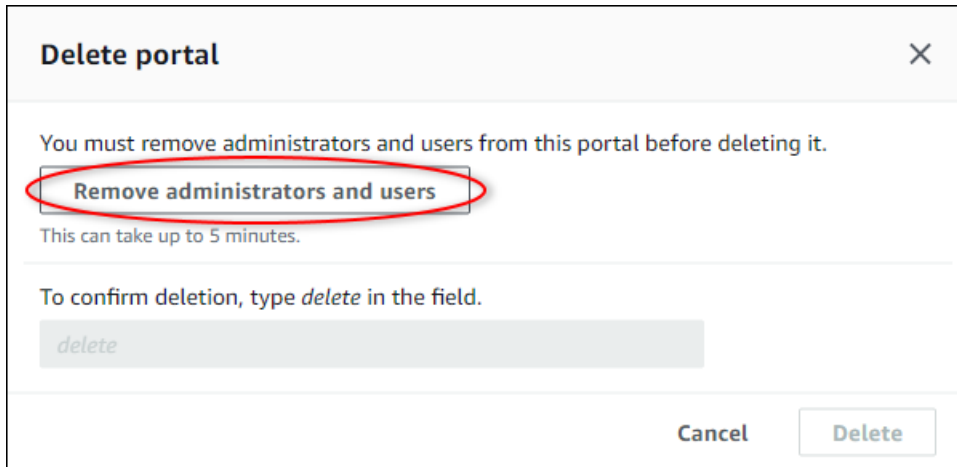
포털 삭제

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 포털을 선택합니다.
3. WindFarmPortal 포털을 선택한 다음 삭제를 선택합니다.

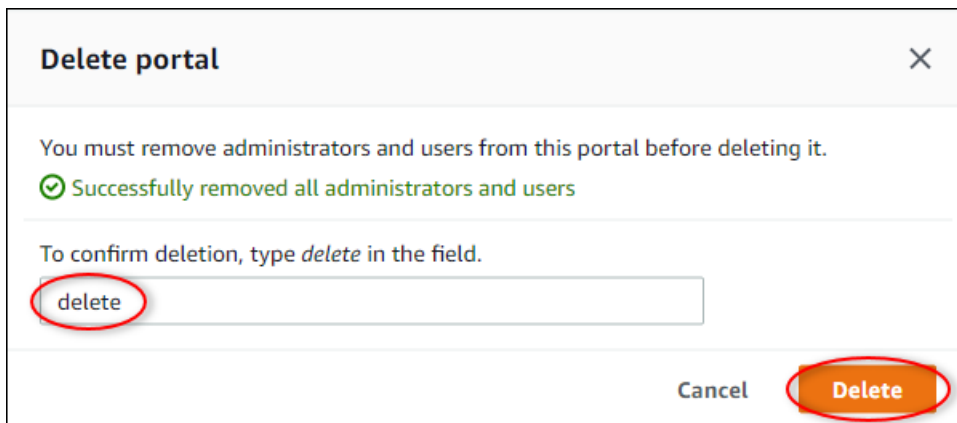
포털이나 프로젝트를 삭제할 때 삭제된 프로젝트와 연결된 자산은 영향을 받지 않습니다.



4. 포털 삭제 대화 상자에서 관리자 및 사용자 제거를 선택합니다.

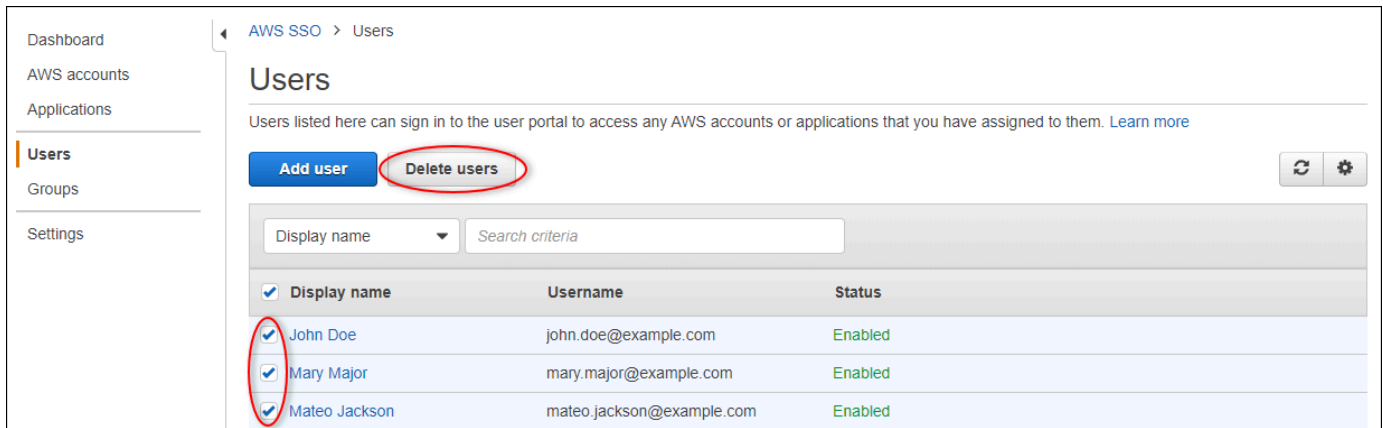


5. **delete**를 입력하여 삭제를 확인한 다음 삭제를 선택합니다.

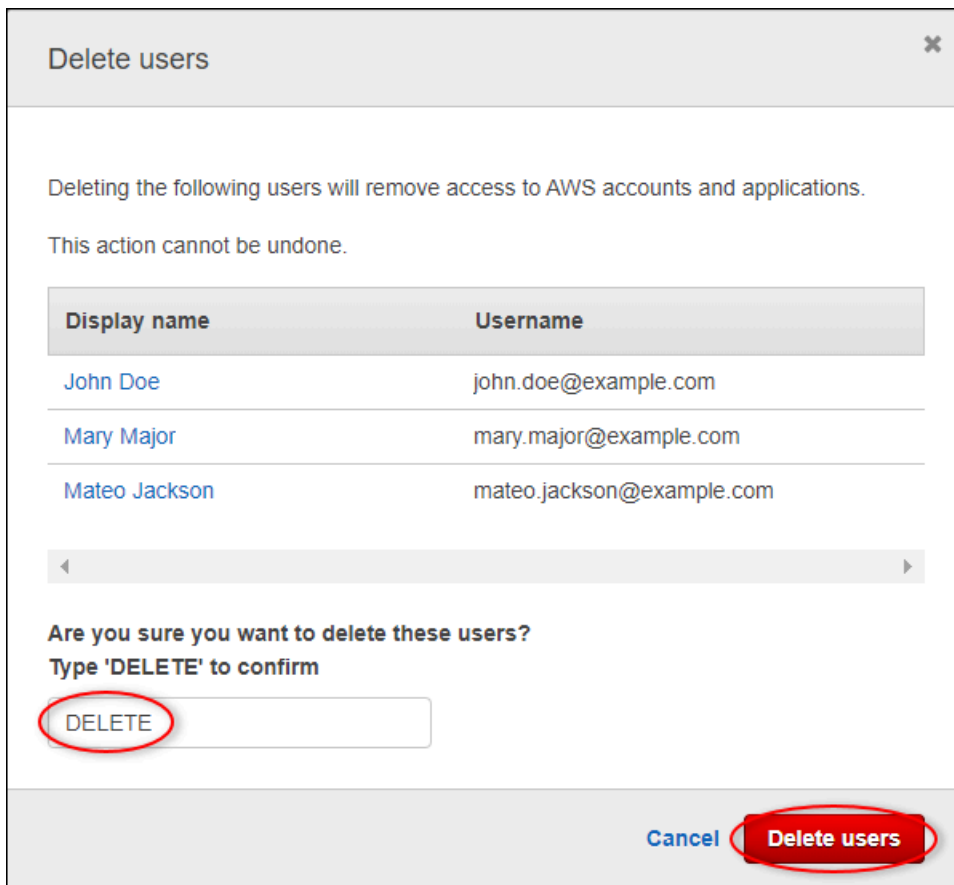


IAM Identity Center 사용자 삭제

1. [IAM Identity Center 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 사용자를 선택합니다.
3. 삭제할 각 사용자의 확인란을 선택한 다음 사용자 삭제를 선택합니다.



4. 사용자 삭제 대화 상자에 **DELETE**를 입력한 후 사용자 삭제를 선택합니다.



Amazon DynamoDB에 속성 값 업데이트 게시

이 자습서에서는 [Amazon DynamoDB](#)를 사용하여 데이터를 저장하는 편리한 방법을 소개하므로 AWS IoT SiteWise API를 반복적으로 쿼리하지 않고도 과거 자산 데이터에 더 쉽게 액세스할 수 있습니다. 이 자습서를 완료하면 전체 풍력 발전소의 풍속 및 방향에 대한 라이브 맵과 같은 자산 데이터를 사용하는 사용자 지정 소프트웨어를 생성할 수 있습니다. 사용자 지정 솔루션을 구현하지 않고 데이터를 모

니터링 및 시각화하려는 경우 [를 사용하여 데이터 모니터링 AWS IoT SiteWise Monitor](#) 섹션을 참조하세요.

이 자습서에서는 풍력 발전소에 대한 샘플 데이터 세트를 제공하는 AWS IoT SiteWise 데모를 기반으로 합니다. AWS IoT Core 규칙을 통해 데이터를 생성한 DynamoDB 테이블로 보내도록 풍력 발전소 데모에서 속성 값 업데이트를 구성합니다. 속성 값 업데이트를 활성화하면 AWS IoT SiteWise MQTT 메시지 AWS IoT Core 의 로 데이터를 전송합니다. 그런 다음 해당 메시지의 내용에 따라 DynamoDB 작업과 같은 작업을 수행하는 AWS IoT 코어 규칙을 정의합니다. 자세한 내용은 [다른 AWS 서비스와 상호 작용](#) 단원을 참조하십시오.

주제

- [사전 조건](#)
- [1단계: 속성 값 업데이트를 게시 AWS IoT SiteWise 하도록 구성](#)
- [2단계: AWS IoT Core에서 규칙 생성](#)
- [3단계: DynamoDB 테이블 생성](#)
- [4단계: DynamoDB 규칙 작업 구성](#)
- [5단계: DynamoDB에서 데이터 탐색](#)
- [6단계: 자습서 완료 후 리소스 정리](#)

사전 조건

이 자습서를 완료하려면 다음이 필요합니다.

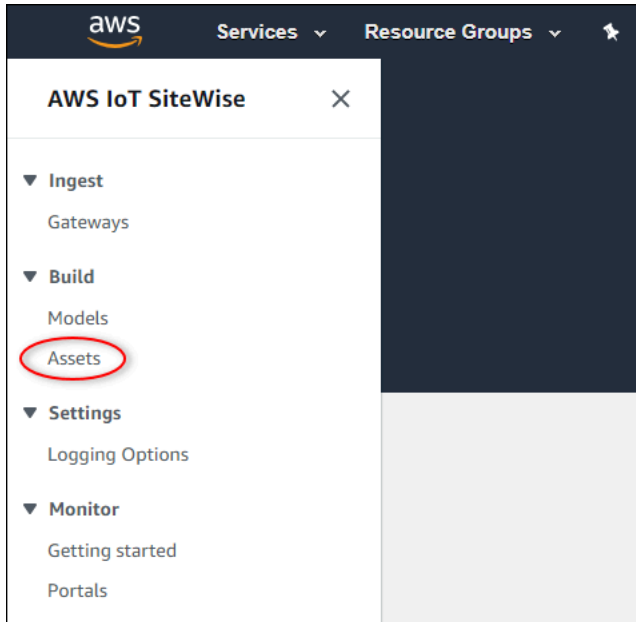
- AWS 계정. 없으면 [AWS 계정 설정](#) 단원을 참조하세요.
- 에 액세스하기 위해 Windows, macOS, Linux 또는 Unix를 실행하는 개발 컴퓨터입니다 AWS Management Console. 자세한 내용은 [AWS Management Console이란 무엇인가요?](#)를 참조하세요.
- 관리자 권한이 있는 IAM 사용자.
- 실행 중인 AWS IoT SiteWise 풍력 발전소 데모입니다. 데모를 설정하면에서 모델과 자산을 정의하고 풍력 발전소를 나타내기 위해 해당 모델 AWS IoT SiteWise 과 자산으로 데이터를 스트리밍합니다. 자세한 내용은 [AWS IoT SiteWise 데모 사용](#) 단원을 참조하십시오.

1단계: 속성 값 업데이트를 게시 AWS IoT SiteWise 하도록 구성

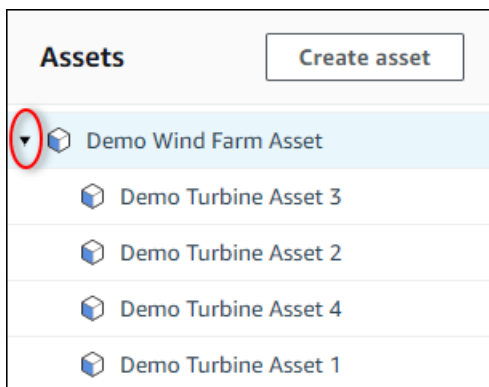
이 절차에서는 데모 터빈 자산의 Wind Speed 속성에 대한 속성 값 알림을 활성화합니다. 속성 값 알림을 활성화하면 MQTT 메시지의 각 값 업데이트를 AWS IoT 코어에 AWS IoT SiteWise 게시합니다.

자산 속성에 대한 속성 값 업데이트 알림 활성화

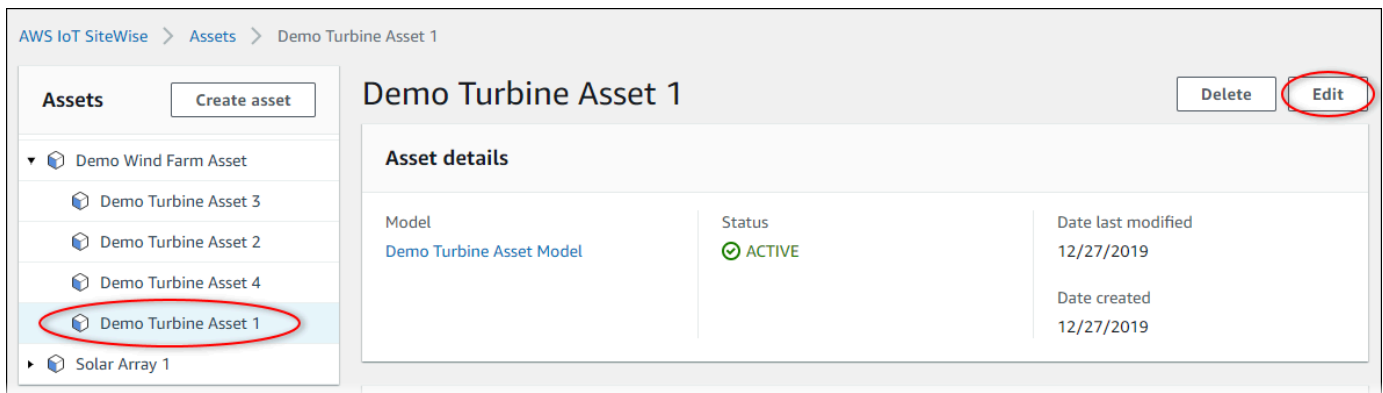
1. [AWS IoT SiteWise 콘솔](#)에 로그인합니다.
2. 가 AWS IoT SiteWise 지원되는 [AWS IoT SiteWise 엔드포인트 및 할당량을](#) 검토하고 필요한 경우 AWS 리전을 전환합니다. AWS IoT SiteWise 데모를 실행하는 리전으로 전환합니다.
3. 왼쪽 탐색 창에서 자산을 선택합니다.



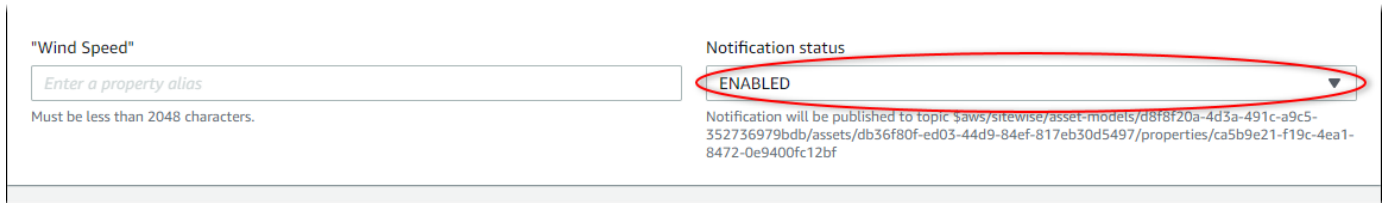
4. Demo Wind Farm Asset 옆에 있는 화살표를 선택하여 풍력 발전소 자산의 계층 구조를 확장합니다.



5. 데모 터빈을 선택하고 편집을 선택합니다.



6. Wind Speed 속성의 알림 상태를 활성화로 업데이트합니다.



7. 페이지 하단에서 자산 저장을 선택합니다.

8. 각 데모 터빈 자산에 대해 단계 5~7을 반복합니다.

9. 데모 터빈(예: Demo Turbine Asset 1)을 선택합니다.

10. 측정을 선택합니다.

11. Wind Speed 속성 옆에 있는 복사 아이콘을 선택하여 알림 주제를 클립보드로 복사합니다. 이 자습서의 뒷부분에서 사용하도록 알림 주제를 저장합니다. 하나의 터빈에서만 알림 주제를 기록하면 됩니다.

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

알림 주제는 다음 예시와 같은 형식이어야 합니다.

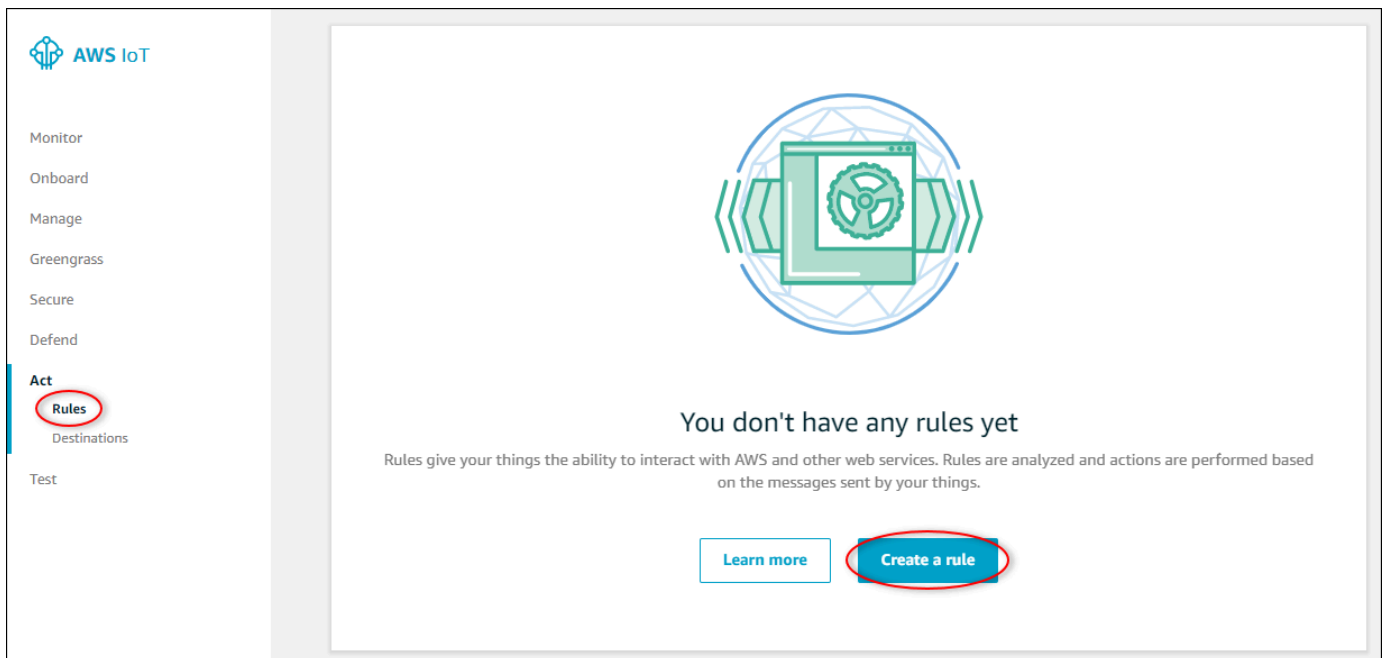
```
$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE
```

2단계: AWS IoT Core에서 규칙 생성

이 절차에서는 속성 값 알림 메시지를 구문 분석하고 Amazon DynamoDB 테이블에 데이터를 삽입하는 규칙을 AWS IoT 코어에서 생성합니다. AWS IoT 코어 규칙은 MQTT 메시지를 구문 분석하고 각 메시지의 내용과 주제를 기반으로 작업을 수행합니다. 그리고 나서 이 자습서의 일부로 생성한 DynamoDB 테이블에 데이터를 삽입하는 DynamoDB 작업을 사용하여 규칙을 생성합니다.

DynamoDB 작업을 사용하여 규칙 생성

1. [AWS IoT 콘솔](#)로 이동합니다. 시작하기 버튼이 나타나면 선택합니다.
2. 왼쪽 탐색 창에서 작업을 선택한 다음 규칙을 선택합니다.



3. 아직 규칙이 없습니다 대화 상자가 나타나면 규칙 생성을 선택합니다. 그렇지 않은 경우 생성을 선택합니다.
4. 규칙에 대해 이름과 설명을 입력하세요.

Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name
WindSpeedRule

Description
A DynamoDBv2 rule that records wind data from wind turbine assets in AWS IoT SiteWise.

5. 이 자습서의 앞부분에서 저장한 알림 주제를 찾습니다.

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

주제의 자산 ID(assets/ 뒤의 ID)를 + 기호로 바꿉니다. 이렇게 하면 모든 데모 풍력 터빈 자산의 풍속 속성이 선택됩니다. + 주제 필터는 해당 주제에 포함된 단일 수준의 모든 노드를 수용합니다. 주제는 다음 예시와 같은 형식이어야 합니다.

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

6. 다음 규칙 쿼리 문을 입력합니다. FROM 섹션의 주제를 해당 알림 주제로 바꿉니다.

```
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. 하나 이상의 작업 설정에서 작업 추가를 선택합니다.

8 `type = "PropertyValueUpdate"`

Set one or more actions

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

Add action

Error action

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

8. 작업 선택 페이지에서 메시지를 DynamoDB 테이블(DynamoDBv2)의 여러 열로 분할을 선택합니다.

Select an action

Select an action.

- ☐  Insert a message into a DynamoDB table
DYNAMODB
- ☒  Split message into multiple columns of a DynamoDB table (DynamoDBv2)
DYNAMODBv2
- ☐  Send a message to a Lambda function
LAMBDA

9. 페이지 하단에서 작업 구성을 선택합니다.
10. 작업 구성 페이지에서 [새 리소스 생성]을 선택합니다.

DynamoDB 콘솔이 새 탭에서 열립니다. 다음 절차를 완료하는 동안 규칙 작업 탭을 열어 둡니다.

3단계: DynamoDB 테이블 생성

이 절차에서는 규칙 작업에서 풍속 데이터를 받을 Amazon DynamoDB 테이블을 만듭니다.

DynamoDB 테이블 생성

1. DynamoDB 콘솔 대시보드에서 테이블 생성을 선택합니다.
2. 테이블의 이름을 입력합니다.

Create DynamoDB table

Tutorial ?

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

Table name*

Primary key* Partition key

☒ Add sort key

Table settings

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

☒ Use default settings

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

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

+ Add tags **NEW!**

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

Cancel

3. 기본 키에서 다음 작업을 수행합니다.

- 파티션 키로 **timestamp**를 입력합니다.
- 번호 유형을 선택합니다.
- 정렬 키 추가 확인란을 선택합니다.
- 정렬 키로 **asset**을 입력하고 기본 정렬 키 유형인 문자열을 그대로 둡니다.

4. 생성(Create)을 선택합니다.

테이블 생성 중 알림이 사라지면 테이블이 준비된 것입니다.

5. 작업 구성 페이지가 있는 탭으로 돌아갑니다. 다음 절차를 완료하는 동안 DynamoDB 탭을 열어 둡니다.

4단계: DynamoDB 규칙 작업 구성

이 절차에서는 속성 값 업데이트의 데이터를 새 DynamoDB 테이블에 삽입하도록 Amazon DynamoDB 규칙 작업을 구성합니다.

DynamoDB 규칙 작업 구성

1. 작업 구성 페이지에서 테이블 이름 목록을 새로 고치고 새 DynamoDB 테이블을 선택합니다.

2. 역할 생성을 선택하여 규칙 작업을 수행할 수 있는 AWS IoT 코어 액세스 권한을 부여하는 IAM 역할을 생성합니다.
3. 역할 이름을 입력하고 역할 생성을 선택합니다.

4. 작업 추가를 선택합니다.
5. 페이지 하단에서 규칙 생성을 선택하여 규칙 생성을 완료합니다.

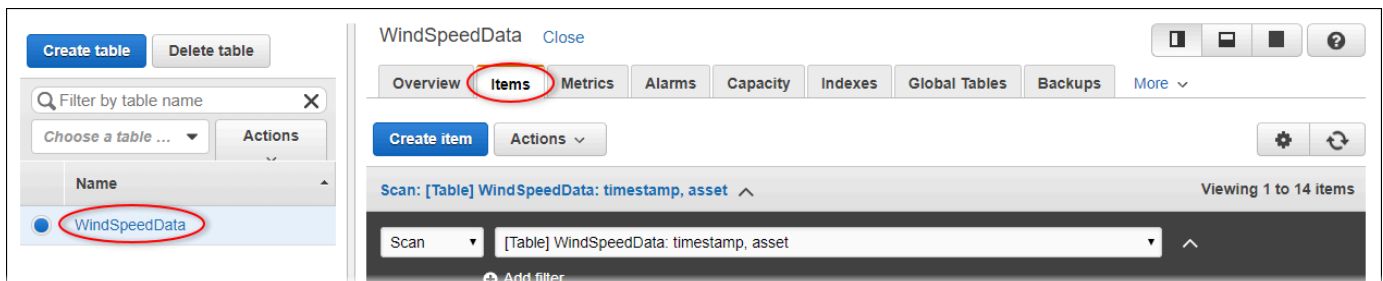
데모 자산 데이터가 DynamoDB 테이블에 나타나기 시작합니다.

5단계: DynamoDB에서 데이터 탐색

이 절차에서는 새 Amazon DynamoDB 테이블에서 데모 자산의 풍속 데이터를 탐색합니다.

DynamoDB에서 자산 데이터 탐색

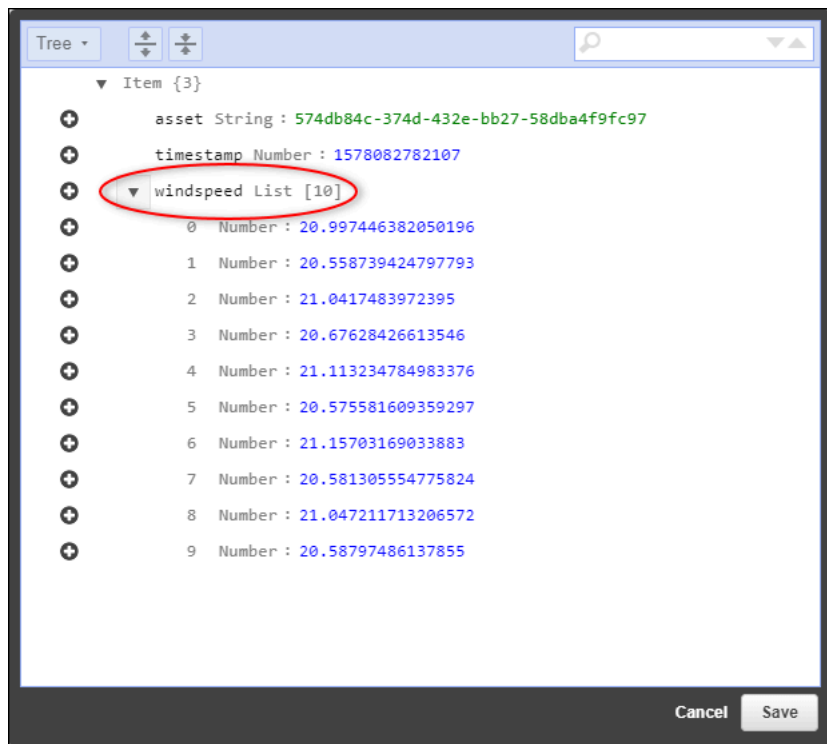
1. DynamoDB 테이블이 열려 있는 탭으로 돌아갑니다.
2. 앞에서 만든 테이블에서 항목 탭을 선택하여 테이블의 데이터를 봅니다. 테이블에 행이 표시되지 않으면 페이지를 새로 고칩니다. 몇 분 후에도 행이 나타나지 않는 경우 [규칙 문제 해결 \(DynamoDB\)](#) 단원을 참조하세요.



3. 테이블의 행에서 편집 아이콘을 선택하여 데이터를 확장합니다.

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

4. windspeed 구조 옆의 화살표를 선택하여 풍속 데이터 포인트 목록을 확장합니다. 각 목록에는 풍력 발전소 데모에서 로 전송 AWS IoT SiteWise 한 바람 속도 데이터 포인트 배치가 반영됩니다. 고유한 용도의 규칙 작업을 설정하는 경우 다른 데이터 형식을 사용할 수 있습니다. 자세한 내용은 [에서 자산 속성 알림 쿼리 AWS IoT SiteWise](#) 단원을 참조하세요.



자습서를 완료했으므로 추가 요금이 발생하지 않도록 규칙을 비활성화하거나 삭제하고 DynamoDB 테이블을 삭제할 수 있습니다. 리소스를 정리하려면 [6단계: 자습서 완료 후 리소스 정리](#) 섹션을 참조하세요.

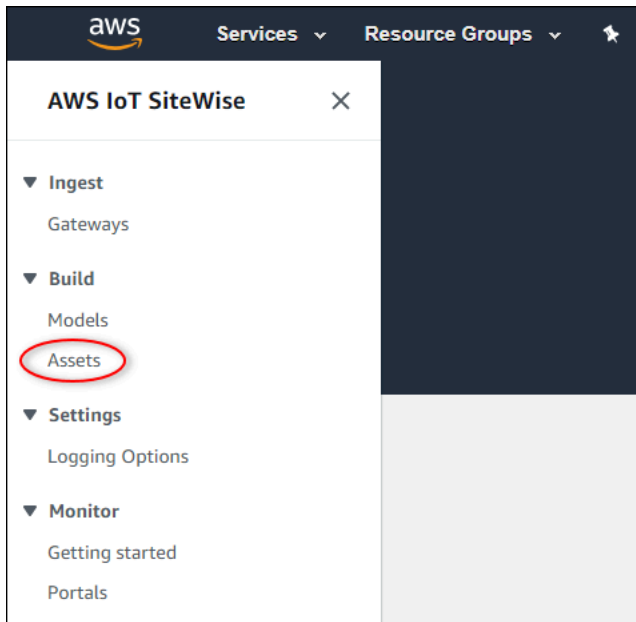
6단계: 자습서 완료 후 리소스 정리

자습서를 완료한 후 추가 요금이 발생하지 않도록 리소스를 정리합니다. 데모 풍력 발전소 자산은 데모를 생성할 때 선택한 기간이 끝날 때 삭제됩니다. 데모를 수동으로 삭제할 수도 있습니다. 자세한 내용은 [AWS IoT SiteWise 데모 삭제](#) 단원을 참조하십시오.

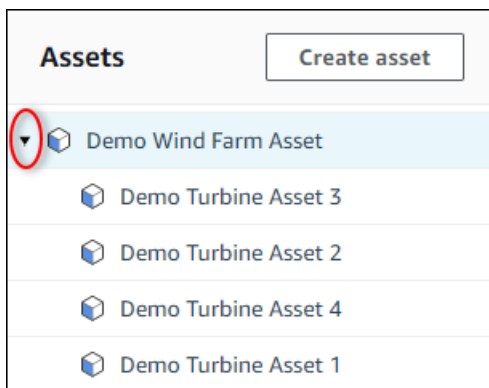
다음 절차에 따라 속성 값 업데이트 알림을 비활성화하고(데모를 삭제하지 않은 경우), AWS IoT 규칙을 비활성화 또는 삭제하고, DynamoDB 테이블을 삭제합니다.

자산 속성에 대한 속성 값 업데이트 알림 비활성화

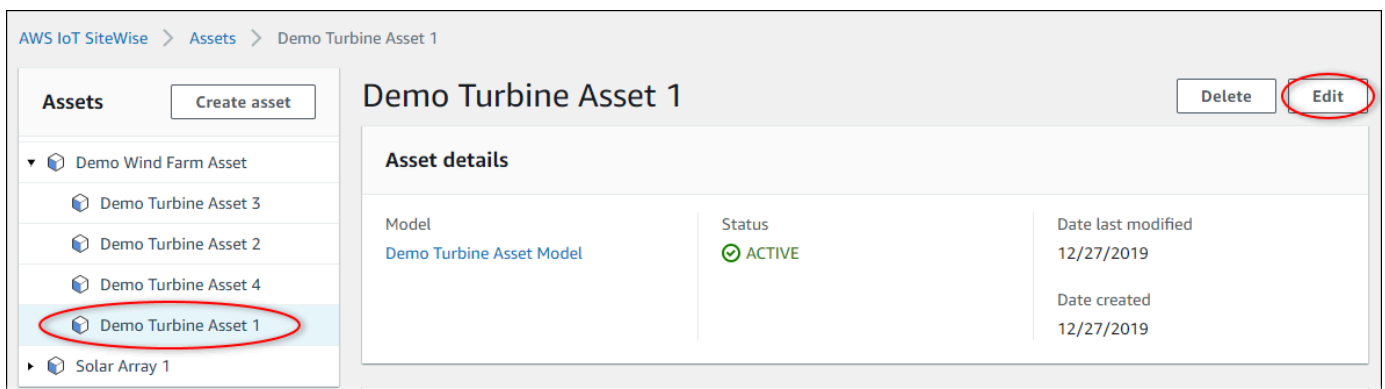
1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 자산을 선택합니다.



3. Demo Wind Farm Asset 옆에 있는 화살표를 선택하여 풍력 발전소 자산의 계층 구조를 확장합니다.



4. 데모 터빈을 선택하고 편집을 선택합니다.



5. Wind Speed 속성의 알림 상태를 비활성으로 업데이트합니다.

"Wind Speed"

Enter a property alias

Must be less than 2048 characters.

Notification status

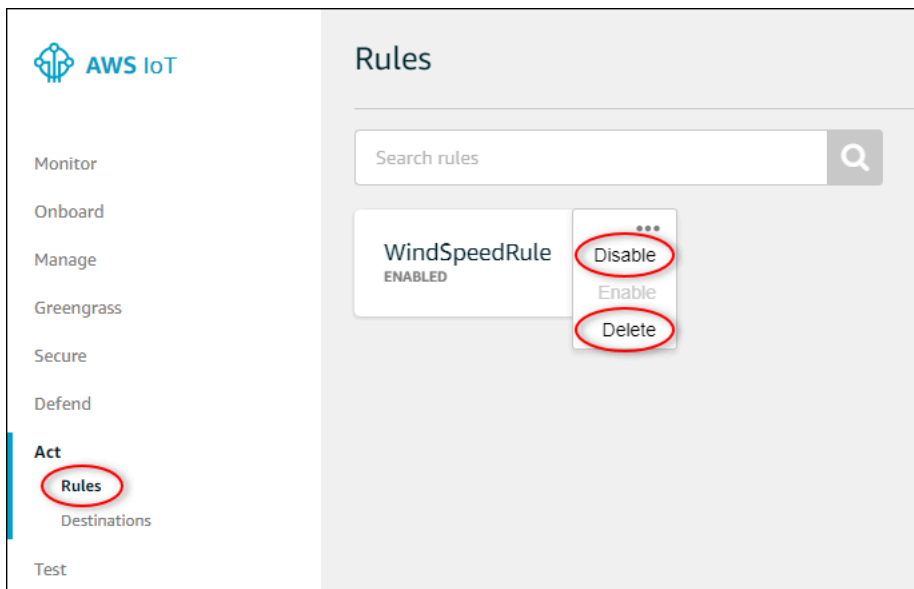
DISABLED

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

6. 페이지 하단에서 자산 저장을 선택합니다.
7. 각 데모 터빈 자산에 대해 4~6단계를 반복합니다.

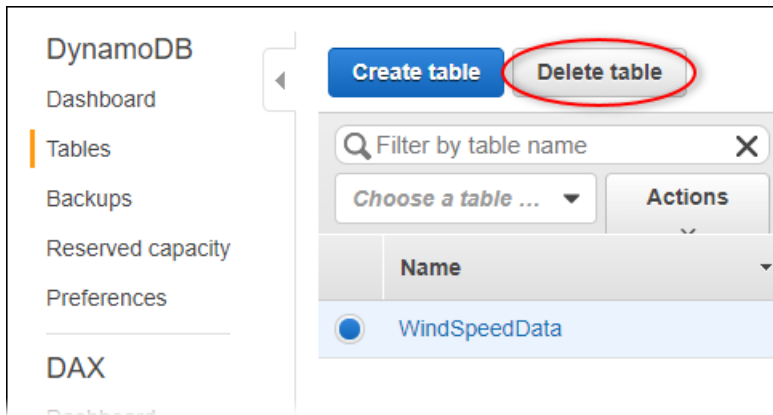
에서 규칙을 비활성화하거나 삭제하려면 AWS IoT Core

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 작업을 선택한 다음 규칙을 선택합니다.
3. 규칙에서 메뉴를 선택하고 비활성화 또는 삭제를 선택합니다.

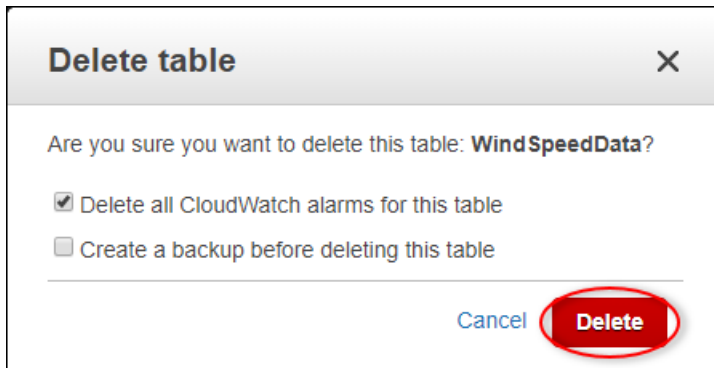


DynamoDB 테이블 삭제

1. [DynamoDB 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 테이블을 선택합니다.
3. 앞에서 생성한 테이블인 WindSpeedData를 선택합니다.
4. 테이블 삭제를 선택합니다.



5. 테이블 삭제 대화 상자에서 삭제를 선택합니다.



에 데이터 수집 AWS IoT SiteWise

AWS IoT SiteWise 는 산업 데이터를 효율적으로 수집하고 해당 자산과 상호 연관시켜 산업 운영의 다양한 측면을 나타내도록 설계되었습니다. 이 설명서는 데이터에 수집하는 실제 측면에 중점을 두고 다양한 산업 사용 사례에 맞는 여러 가지 방법을 AWS IoT SiteWise 제공합니다. 가상 산업 조직 구축에 대한 지침은 [산업 자산 모델링](#) 단원을 참조하세요.

다음 옵션 중 하나를 AWS IoT SiteWise 사용하여 산업 데이터를 로 전송할 수 있습니다.

- AWS IoT SiteWise Edge - [SiteWise Edge 게이트웨이](#)를 AWS IoT SiteWise 와 데이터 서버 간의 중개자로 사용합니다. AWS IoT SiteWise 는 SiteWise Edge 게이트웨이를 설정하기 AWS IoT Greengrass 위해 실행할 수 있는 모든 플랫폼에 배포할 수 있는 AWS IoT Greengrass 구성 요소를 제공합니다. 이 옵션은 [OPC UA](#) 서버 프로토콜과의 연결을 지원합니다.
- AWS IoT SiteWise API - [AWS IoT SiteWise API](#)를 사용하여 다른 소스에서 데이터를 업로드합니다. 스트리밍 [BatchPutAssetPropertyValue](#) API를 사용하여 몇 초 내에 수집하거나 배치 지향 [CreateBulkImportJob](#) API를 사용하여 비용 효율적인 대규모 배치로 수집할 수 있습니다.
- AWS IoT 코어 규칙 - [AWS IoT 코어 규칙](#)을 사용하여 AWS IoT 사물 또는 다른 AWS 서비스에서 게시한 MQTT 메시지에서 데이터를 업로드합니다.
- AWS IoT Events 작업 -의 특정 이벤트에 의해 트리거되는 [AWS IoT Events 작업을](#) 사용합니다 AWS IoT Events. 이 방법은 데이터 업로드가 이벤트 발생과 연결된 시나리오에 적합합니다.
- AWS IoT Greengrass 스트림 관리자 - [AWS IoT Greengrass 스트림 관리자](#)를 사용하여 엣지 디바이스를 사용하여 로컬 데이터 소스에서 데이터를 업로드합니다. 이 옵션은 데이터가 온프레미스 또는 엣지 로케이션에서 발생하는 상황에 적합합니다.

이러한 방법은 다양한 소스의 데이터를 관리하기 위한 다양한 솔루션을 제공합니다. 각 옵션의 세부 정보를 살펴보고에서 AWS IoT SiteWise 제공하는 데이터 수집 기능을 포괄적으로 이해합니다.

에 대한 데이터 스트림 관리 AWS IoT SiteWise

데이터 스트림은 과거 시계열 데이터를 포함하는 리소스입니다. 각 데이터 스트림은 고유한 별칭으로 식별되므로 각 데이터의 오리진을 더 쉽게 추적할 수 있습니다. 첫 번째 시계열 데이터가 수신 AWS IoT SiteWise 되면서 데이터 스트림이 자동으로 생성됩니다. 첫 번째 시계열 데이터가 별칭으로 식별되는 경우는 해당 별칭에 자산 속성이 아직 할당되지 않은 경우 해당 별칭이 있는 새 데이터 스트림을 AWS IoT SiteWise 생성합니다. 또는 첫 번째 시계열 데이터가 자산 ID 및 속성 ID로 식별되면 새 데이터 스트림을 AWS IoT SiteWise 생성하고 해당 데이터 스트림을 자산 속성과 연결합니다.

자산 속성에 별칭을 할당하는 방법에는 두 가지가 있습니다. 사용되는 방법은 데이터가 AWS IoT SiteWise 먼저 로 전송되는지 또는 자산이 먼저 생성되는지에 따라 달라집니다.

- 데이터가 AWS IoT SiteWise 먼저 로 전송되는 경우 할당된 별칭이 있는 데이터 스트림이 자동으로 생성됩니다. 나중에 자산을 생성할 때 [AssociateTimeSeriesToAssetProperty](#) API를 사용하여 데이터 스트림과 해당 별칭을 자산 속성에 연결합니다.
- 자산이 먼저 생성된 경우 [UpdateAssetProperty](#) API를 사용하여 자산 속성에 별칭을 할당합니다. 나중에 데이터가 로 전송되면 AWS IoT SiteWise 데이터 스트림이 자동으로 생성되고 자산 속성과 연결됩니다.

현재는 데이터 스트림을 측정값에만 연결할 수 있습니다. 측정값은 타임스탬프가 지정된 온도 값 또는 타임스탬프가 지정된 분당 회전 수(RPM) 값 같은, 디바이스의 원시 센서 데이터 스트림을 나타내는 자산 속성 유형입니다.

이러한 측정값이 지표 또는 변환을 정의하면 수신 데이터가 특정 계산을 트리거합니다. 자산 속성은 한번에 하나의 데이터 스트림에만 연결할 수 있다는 점에 유의해야 합니다.

AWS IoT SiteWise 는 Amazon 리소스 이름(ARN) 리소스TimeSeries에를 사용하여 스토리지 요금을 결정합니다. 자세한 내용은 [AWS IoT SiteWise 요금](#)을 참조하세요.

다음 섹션에서는 AWS IoT SiteWise 콘솔 또는 API를 사용하여 데이터 스트림을 관리하는 방법을 보여줍니다.

주제

- [권한 및 설정 구성](#)
- [데이터 스트림을 자산 속성에 연결](#)
- [자산 속성에서 데이터 스트림 연결 해제](#)
- [데이터 스트림 삭제](#)
- [자산 속성 별칭 업데이트](#)
- [일반적인 시나리오](#)

권한 및 설정 구성

첫 번째 시계열 데이터가 수신 AWS IoT SiteWise 되면서 데이터 스트림이 자동으로 생성됩니다. 수집된 데이터가 자산 속성과 연결되지 않은 경우는 자산 속성과 연결되도록 구성할 수 있는 연결 해제된 새 데이터 스트림을 AWS IoT SiteWise 생성합니다. IAM 정책을 AWS IoT SiteWise 사용하여 수집할 데이터 유형을 지정하여 데이터를 전송하는 게이트웨이의 액세스 제어를 구성합니다.

다음 IAM 정책은 게이트웨이에서 연결 해제된 데이터 수집을 비활성화하는 동시에 자산 속성과 연결된 데이터 스트림에 대한 데이터 수집을 허용합니다.

Example 게이트웨이에서 연결 해제된 데이터 수집을 비활성화하는 IAM 사용자 정책

```
{
  "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 게이트웨이에서 모든 데이터 수집을 비활성화하는 IAM 사용자 정책

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

```

"Statement": [
  {
    "Sid": "DenyPutAssetPropertyValues",
    "Effect": "Deny",
    "Action": "iotsitewise:BatchPutAssetPropertyValue",
    "Resource": {
      "arn:aws:iotsitewise:*:*:asset/*",
      "arn:aws:iotsitewise:*:*:time-series/*"
    }
  }
]
}

```

데이터 스트림을 자산 속성에 연결

AWS IoT SiteWise Console 또는를 사용하여 데이터 스트림을 관리합니다 AWS CLI.

Console

AWS IoT SiteWise 콘솔을 사용하여 데이터 스트림을 관리합니다.

데이터 스트림을 관리하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 패널에서 데이터 스트림을 선택합니다.
3. 데이터 스트림 별칭을 필터링하거나 필터 드롭다운 메뉴에서 연결 해제된 데이터 스트림을 선택하여 데이터 스트림을 선택합니다.
4. 업데이트할 데이터 스트림을 선택합니다. 여러 데이터 스트림을 선택할 수 있습니다. 오른쪽 상단에서 데이터 스트림 관리를 클릭합니다.
5. 데이터 스트림 연결 업데이트에서 연결할 데이터 스트림을 선택하고 측정 선택 버튼을 클릭합니다.
6. 측정 선택 섹션에서 해당 자산 측정 속성을 찾습니다. 측정값을 선택한 다음 선택을 클릭합니다.
7. 3단계에서 선택한 다른 데이터 스트림에 대해 4단계와 5단계를 수행합니다. 모든 데이터 스트림에 자산 속성을 할당합니다.
8. 업데이트를 선택하여 변경 사항을 커밋합니다. 업데이트를 확인하는 성공 확인 배너가 표시됩니다.

AWS CLI

데이터 스트림(별칭으로 식별됨)을 자산 속성(IDs로 식별됨)에 연결하려면 다음 명령을 실행합니다.

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

자산 속성에서 데이터 스트림 연결 해제

Console

AWS IoT SiteWise 콘솔을 사용하여 자산 속성에서 데이터 스트림의 연결을 해제합니다.

자산 속성에서 데이터 스트림의 연결을 해제하려면(콘솔)

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 패널에서 데이터 스트림을 선택합니다.
3. 데이터 스트림 별칭을 필터링하거나 필터 드롭다운 메뉴에서 연결된 데이터 스트림을 선택하여 데이터 스트림을 선택합니다.
4. 연결 해제할 데이터 스트림을 선택합니다. 데이터 스트림 별칭 옆에는 별칭이 포함되어야 합니다. 자산 이름 및 자산 속성 이름 옆에는 데이터 스트림이 연결된 자산 속성의 값이 포함되어야 합니다. 여러 데이터 스트림을 선택할 수 있습니다.
5. 오른쪽 상단에서 데이터 스트림 관리를 클릭합니다.
6. 데이터 스트림 연결 업데이트 섹션의 측정 이름 열에서 X를 클릭합니다. submitted 상태가 상태 열에 나타나야 합니다.
7. 업데이트를 선택하여 변경 사항을 커밋합니다. 이제 데이터 스트림이 자산 속성에서 연결 해제되고 별칭이 데이터 스트림을 식별하는 데 사용됩니다.

AWS CLI

자산 속성(ID 및 별칭으로 식별됨)에서 데이터 스트림의 연결을 해제하려면 다음 명령을 실행합니다.

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

이제 데이터 스트림이 자산 속성에서 연결 해제되고 별칭이 데이터 스트림을 식별하는 데 사용됩니다. 별칭은 이제 데이터 스트림과 연결되므로 자산 속성과 더 이상 연결되지 않습니다.

데이터 스트림 삭제

자산 모델에서 속성이 제거되면 자산 모델에서 관리하는 모든 자산에서 속성과 해당 데이터 스트림을 AWS IoT SiteWise 삭제합니다. 또한 자산이 삭제될 때 자산의 모든 속성과 해당 데이터 스트림을 삭제합니다. 데이터 스트림 데이터를 보존해야 하는 경우 삭제하기 전에 자산 속성과 연결을 해제해야 합니다.

Warning

자산에서 속성을 삭제하면 연결된 데이터 스트림도 삭제됩니다. 데이터 스트림을 보존하려면 먼저 자산 모델에서 속성을 삭제하거나 자산을 삭제하기 전에 자산 속성에서 연결을 해제합니다.

Console

AWS IoT SiteWise 콘솔을 사용하여 자산 속성에서 데이터 스트림의 연결을 해제합니다.

데이터 스트림을 삭제하려면(콘솔)

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 패널에서 데이터 스트림을 선택합니다.
3. 데이터 스트림 별칭을 필터링하여 데이터 스트림을 선택합니다.
4. 삭제할 데이터 스트림을 선택합니다. 여러 데이터 스트림을 선택할 수 있습니다.
5. 삭제 버튼을 선택하여 데이터 스트림을 삭제합니다.

AWS CLI

[DeleteTimeSeries](#) API를 사용하여 별칭으로 특정 데이터 스트림을 삭제합니다.

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

자산 속성 별칭 업데이트

별칭은 AWS 리전 내에서 고유해야 합니다. 여기에는 자산 속성과 데이터 스트림의 별칭이 모두 포함됩니다. 다른 속성 또는 데이터 스트림이 해당 별칭을 사용하는 경우 자산 속성에 별칭을 할당하지 마십시오.

Console

AWS IoT SiteWise 콘솔을 사용하여 자산 속성 별칭을 업데이트합니다.

자산 속성 별칭을 업데이트하려면(콘솔)

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 테이블에서 자산을 선택합니다.
4. Edit 버튼을 클릭합니다.
5. 속성 테이블에서 속성 유형을 선택합니다.
6. 속성을 찾고 속성 별칭 텍스트 필드에 새 별칭을 입력합니다.
7. 저장 버튼을 클릭하여 변경 사항을 저장합니다.

AWS CLI

자산 속성의 별칭을 업데이트하려면 다음 명령을 실행합니다.

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

Note

속성 알림이 현재 활성화된 경우 계속 활성화되도록 다시 제공해야 합니다.

일반적인 시나리오

데이터 스트림 이동

데이터 스트림의 연결을 다른 자산 속성으로 변경하려면 먼저 데이터 스트림을 현재 자산 속성에서 연결 해제합니다. 자산 속성에서 데이터 스트림의 연결을 해제할 때는 해당 자산 속성에 할당된 별칭이 있어야 합니다.

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

이제 데이터 스트림을 새 자산 속성에 다시 할당합니다.

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

자산 속성에 별칭을 할당할 때 오류 발생

UpdateAssetProperty API를 사용하여 속성에 별칭을 할당할 때 다음 오류 메시지가 표시될 수 있습니다.

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

이 오류 메시지는 별칭이 현재 다른 속성 또는 데이터 스트림에서 사용되므로 속성에 할당되지 않았음을 나타냅니다.

이는 데이터가 별칭 AWS IoT SiteWise 을 사용하여 수집되는 경우에 발생합니다. 다른 데이터 스트림 또는 자산 속성에서 사용하지 않는 별칭으로 데이터를 전송하면 해당 별칭으로 새 데이터 스트림이 생성됩니다. 아래 두 옵션은 문제를 해결합니다.

- AssociateTimeSeriesToAssetProperty API를 사용하여 데이터 스트림을 별칭과 자산 속성에 연결합니다.
- 데이터 수집을 일시적으로 중지하고 데이터 스트림을 삭제합니다. UpdateAssetProperty API를 사용하여 자산 속성에 별칭을 할당한 다음 데이터 수집을 다시 켭니다.

데이터 스트림을 자산 속성에 연결할 때 오류 발생

데이터 스트림을 자산 속성에 연결할 때 다음 오류 메시지가 표시됩니다.

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

이 오류 메시지는 자산 속성이 이미 데이터가 포함된 데이터 스트림과 연결되어 있음을 나타냅니다. 다른 데이터 스트림을 해당 자산 속성에 연결하기 전에 해당 데이터 스트림을 연결 해제하거나 삭제해야 합니다.

Note

자산 속성에서 데이터 스트림의 연결을 해제하면 속성에 할당된 별칭이 데이터 스트림에 제공 됩니다. 해당 별칭이 속성에 할당된 상태로 유지되도록 하려면 데이터 스트림의 연결을 해제하기 전에 해당 속성에 새 별칭을 할당합니다.

자산 속성에 저장된 데이터를 보존하려면 다음을 수행합니다.

- 새 데이터 스트림이 생성되지 않도록 자산 속성에 데이터가 수집되지 않도록 합니다.
- UpdateAssetProperty API를 사용하여 현재 할당된 데이터 스트림에 제공되는 새 별칭을 설정합니다.
- DisassociateTimeSeriesFromAssetProperty API를 사용하여 자산 속성에서 현재 데이터 스트림의 연결을 해제합니다.
- AssociateTimeSeriesToAssetProperty API를 사용하여 원하는 데이터 스트림을 자산 속성에 연결합니다.

자산 속성에 저장된 데이터를 삭제해야 하는 경우 다음을 수행합니다.

- 새 데이터 스트림이 생성되지 않도록 자산 속성에 데이터가 수집되지 않도록 합니다.
- DeleteTimeSeries API를 사용하여 현재 할당된 데이터 스트림을 삭제합니다.
- AssociateTimeSeriesToAssetProperty API를 사용하여 원하는 데이터 스트림을 자산 속성에 연결합니다.

AWS IoT SiteWise APIs 사용하여 데이터 수집

AWS IoT SiteWise APIs 사용하여 타임스탬프가 지정된 산업 데이터를 자산의 속성 및 측정 속성으로 전송합니다. APIs는 timestamp-quality-value(TQV) 구조를 포함하는 페이로드를 허용합니다.

BatchPutAssetPropertyValue API

[BatchPutAssetPropertyValue](#) 작업을 사용하여 데이터를 업로드할 수 있습니다. 이 작업을 사용하면 한 번에 여러 데이터 항목을 업로드할 수 있으므로, 여러 디바이스에서 데이터를 수집하여 단일 요청으로 모두 전송할 수 있습니다.

Important

[BatchPutAssetPropertyValue](#) 작업에는 다음 할당량이 적용됩니다.

- 요청당 최대 10개 [항목](#).
- 항목당 최대 10개의 [속성 값](#)(TQV 데이터 포인트).
- AWS IoT SiteWise 는 타임스탬프가 과거 7일 이상 또는 향후 10분 이상으로 기록된 데이터를 거부합니다.

이러한 할당량에 대한 자세한 내용은 AWS IoT SiteWise API 참조의 [BatchPutAssetPropertyValue](#)를 참조하세요.

자산 속성을 식별하려면 다음 중 하나를 지정합니다.

- 데이터를 보낼 자산 속성의 assetId 및 propertyId
- 데이터 스트림 별칭(예: /company/windfarm/3/turbine/7/temperature)인 propertyAlias. 이 옵션을 사용하려면 먼저 자산 속성의 별칭을 설정해야 합니다. 속성 별칭을 설정하려면 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 섹션을 참조하세요.

다음 예제는 JSON 파일에 저장된 페이로드로부터 풍력 터빈의 온도와 분당 회전 수(RPM) 판독값을 전송하는 방법을 보여줍니다.

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

의 예제 페이로드에는 다음과 같은 콘텐츠batch-put-payload.json가 있습니다.

```
{
  "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"
      }
    ]
  }
}

```

`enablePartialEntryProcessing` 로 지정 `true` 하면 실패를 초래하지 않는 모든 값을 수집할 수 있습니다. 기본값은 `false`입니다. 값이 유효하지 않으면 전체 항목이 수집되지 않습니다.

페이로드의 각 항목은 고유한 문자열로 정의할 수 있는 `entryId`를 포함합니다. 요청 항목이 실패하면 각 오류에 해당 요청의 `entryId`가 포함되므로 다시 시도할 요청을 알 수 있습니다.

`propertyValues` 목록에서 각 구조는 `value`, `timestamp` 및(`quality`선택 사항)를 포함하는 TQV(타임스탬프-품질-값) 구조입니다.

- `value` - 설정되는 속성 유형에 따라 다음 필드 중 하나를 포함하는 구조입니다.
 - `booleanValue`
 - `doubleValue`
 - `integerValue`
 - `stringValue`
 - `nullValue`
- `nullValue` - 값이 `Null`이고 품질이 `BAD` 또는 인 속성 값의 유형을 나타내는 다음 필드가 있는 구조입니다 `UNCERTAIN`.
 - `valueType` - {"B", "D", "S", "I"}의 열거형
- `timestamp` - 현재 Unix Epoch 시간(초) `timeInSeconds`를 포함하는 구조입니다. 시간적으로 정확한 데이터가 있는 경우 `timestamp` 구조에서 `offsetInNanos` 키를 설정할 수도 있습니다.는 타임스탬프가 과거 7일 이상 지난 데이터 포인트 또는 향후 10분 이상 지난 데이터 포인트를 AWS IoT SiteWise 거부합니다.
- `quality` - (선택 사항) 다음 품질 문자열 중 하나입니다.

- GOOD – (기본값) 데이터가 어떤 문제의 영향도 받지 않습니다.
- BAD – 데이터가 센서 고장과 같은 문제의 영향을 받습니다.
- UNCERTAIN – 데이터가 센서 부정확과 같은 문제의 영향을 받습니다.

가 계산에서 데이터 품질을 AWS IoT SiteWise 처리하는 방법에 대한 자세한 내용은 [수식 표현식의 데이터 품질을 참조하세요](#).

CreateBulkImportJob API

CreateBulkImportJob API를 사용하여 Amazon S3에서 대량의 데이터를 가져옵니다. Amazon S3에 데이터를 CSV 형식으로 저장해야 합니다. 데이터 파일에는 다음과 같은 열이 있을 수 있습니다.

Note

1970년 1월 1일 00:00:00 UTC보다 오래된 데이터는 지원되지 않습니다.
자산 속성을 식별하려면 다음 중 하나를 지정합니다.

- 데이터를 보낼 자산 속성의 ASSET_ID 및 PROPERTY_ID.
- 데이터 스트림 별칭(예: /company/windfarm/3/turbine/7/temperature)인 ALIAS. 이 옵션을 사용하려면 먼저 자산 속성의 별칭을 설정해야 합니다. 속성 별칭을 설정하는 방법에 대한 자세한 내용은 [the section called “데이터 스트림 관리”](#) 단원을 참조하세요.

- ALIAS – OPC UA 서버 데이터 스트림 경로(예: /company/windfarm/3/turbine/7/temperature)와 같이 속성을 식별하는 속성 별칭입니다. 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 단원을 참조하십시오.
- ASSET_ID – 자산의 ID입니다.
- PROPERTY_ID – 자산 속성의 ID입니다.
- DATA_TYPE – 속성의 데이터 형식입니다. 다음 중 하나일 수 있습니다.
 - STRING – 최대 1,024바이트의 문자열입니다.
 - INTEGER – 범위가 [-2,147,483,648, 2,147,483,647]인 부호 있는 32비트 정수입니다.
 - DOUBLE – 범위가 [-10^100, 10^100]이고 IEEE 754 배정밀도를 갖는 부동 소수점 숫자입니다.
 - BOOLEAN - true 또는 false입니다.
- TIMESTAMP_SECONDS - Unix Epoch 시간으로 표시된 데이터 포인트의 타임스탬프입니다.
- TIMESTAMP_NANO_OFFSET - TIMESTAMP_SECONDS에서 변환된 나노초의 오프셋입니다.

- QUALITY - (선택 사항) 자산 속성 값의 품질입니다. 값은 다음 중 하나일 수 있습니다.
 - GOOD - (기본값) 데이터가 어떤 문제의 영향도 받지 않습니다.
 - BAD - 데이터가 센서 고장과 같은 문제의 영향을 받습니다.
 - UNCERTAIN - 데이터가 센서 부정확과 같은 문제의 영향을 받습니다.

가 계산에서 데이터 품질을 AWS IoT SiteWise 처리하는 방법에 대한 자세한 내용은 [수식 표현식의 데이터 품질을 참조하세요](#).

- VALUE - 자산 속성의 값입니다.

Example .csv 형식의 데이터 파일

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

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

AWS IoT SiteWise 는 대량 가져오기 작업을 생성하고 기존 작업에 대한 정보를 가져오기 위해 다음과 같은 API 작업을 제공합니다.

- [CreateBulkImportJob](#) - 새 대량 가져오기 작업을 생성합니다.
- [DescribeBulkImportJob](#) - 대량 가져오기 작업에 대한 정보를 검색합니다.
- [ListBulkImportJob](#) - 모든 대량 가져오기 작업의 요약 목록을 페이지별로 나누어 검색합니다.

AWS IoT SiteWise 대량 가져오기 작업 생성(AWS CLI)

[CreateBulkImportJob](#) API 작업을 사용하여 Amazon S3에서 AWS IoT SiteWise에 데이터를 전송할 수 있습니다. [CreateBulkImportJob](#) API를 사용하면 대량의 기록 데이터를 수집하고 분석 데이터 스트림

을 소규모 배치로 버퍼링된 수집할 수 있습니다. 데이터 수집을 위한 비용 효과적인 기본 원칙을 제공합니다. 다음 예제에서는 AWS CLI를 사용합니다.

⚠ Important

대량 가져오기 작업을 생성하기 전에 AWS IoT SiteWise 웹 티어 또는 AWS IoT SiteWise 콜드 티어를 활성화해야 합니다. 자세한 내용은 [에서 스토리지 설정 구성 AWS IoT SiteWise](#) 단원을 참조하십시오.

[CreateBulkImportJob](#) API는 adaptive-ingestion-flag 파라미터를 설정하는 옵션을 AWS IoT SiteWise 사용하여 기록 데이터를 수집할 수 있도록 지원합니다.

- false로 설정하면 API는 계산 또는 알림을 트리거하지 않고 기록 데이터를 수집합니다.
- true로 설정하면 API는 새 데이터를 수집하여 지표를 계산하고 데이터를 변환하여 7일 이내에 지속적인 분석 및 알림을 최적화합니다.

다음 명령을 실행합니다. *file-name*을 대량 가져오기 작업 구성이 포함된 파일의 이름으로 바꿉니다.

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

Example 일괄 가져오기 작업 구성

다음은 구성 설정의 예입니다.

- *adaptive-ingestion-flag*를 true 또는 false로 바꿉니다.
 - false로 설정하면 일괄 가져오기 작업이 기록 데이터를 AWS IoT SiteWise로 수집합니다.
 - true로 설정하면 일괄 가져오기 작업은 다음을 수행합니다.
 - 새 데이터에 수집합니다 AWS IoT SiteWise.
 - 지표를 계산하고 변환하며 7일 이내에 타임스탬프가 지정된 데이터에 대한 알림을 지원합니다.
- AWS IoT SiteWise 웹 티어 스토리지에 수집한 후 Amazon S3 데이터 버킷에서 데이터를 삭제하려면 *delete-files-after-import-flag*를 로 바꿉니다.
- *amzn-s3-demo-bucket-for-errors*를 이 일괄 가져오기 작업과 관련된 오류가 전송되는 Amazon S3 버킷의 이름으로 바꿉니다.
- *amzn-s3-demo-bucket-for-errors-prefix*를 이 일괄 가져오기 작업과 관련된 오류가 전송되는 Amazon S3 버킷의 접두어로 바꿉니다.

Amazon S3에서는 접두사를 버킷의 폴더 이름으로 사용합니다. 각 Amazon S3 객체에는 버킷의 고유 식별자인 키가 있습니다. 버킷 내 각 객체는 정확히 하나의 키를 갖습니다. 접두사는 슬래시(/)로 끝나야 합니다. 자세한 내용을 알아보려면 Amazon Simple Storage Service 사용 설명서의 [접두어를 사용한 객체 구성](#)을 참조하세요.

- `amzn-s3-demo-bucket-data`를 데이터를 가져오는 Amazon S3 버킷의 이름으로 바꿉니다.
- `data-bucket-key`를 데이터가 들어 있는 Amazon S3 객체의 키로 대체합니다. 각 객체에는 고유 식별자인 키가 있습니다. 각 객체에는 정확히 하나의 키가 있습니다.
- `data-bucket-veron-id`를 버전 ID로 바꾸면 데이터가 들어 있는 Amazon S3 객체의 특정 버전을 식별할 수 있습니다. 이 파라미터는 선택 사항입니다.
- `column-name`을 .csv 파일에 지정된 열 이름으로 대체하세요.
- `job-name`을 대량 가져오기 작업을 식별하는 고유한 이름으로 대체하세요.
- `job-role-arn`을 AWS IoT SiteWise 에서 Amazon S3 데이터를 읽을 수 있는 IAM 역할로 대체하세요.

Note

역할에 다음 예에 표시된 것과 같은 권한이 있는지 확인합니다. `amzn-s3-demo-bucket-data`를 데이터가 포함된 Amazon S3 버킷의 이름으로 바꿉니다. 또한 `amzn-s3-demo-bucket-for-errors`를 이 일괄 가져오기 작업과 관련된 오류가 전송되는 Amazon S3 버킷의 이름으로 바꿉니다.

```
{
  "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 응답

```

{

```

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

AWS IoT SiteWise 대량 가져오기 작업 설명(AWS CLI)

[DescribeBulkImportJob](#) API 작업을 사용하여 AWS IoT SiteWise의 특정 일괄 가져오기 작업에 대한 정보를 검색할 수 있습니다. 이 작업은 작업의 상태, 만들어진 시간, 작업이 실패한 경우 오류 정보와 같은 세부 정보를 반환합니다. 이 작업을 사용하여 작업 진행 상황을 모니터링하고 문제를 해결할 수 있습니다. DescribeBulkImportJob을 사용하려면 CreateBulkImportJob 작업의 작업 ID가 필요합니다. API는 다음 정보를 반환합니다.

- Amazon S3 버킷 위치 및 키를 포함하여 가져오는 파일 목록
- 오류 보고서 위치(해당하는 경우)
- 파일 형식 및 CSV 열 이름과 같은 작업 구성 세부 정보
- 작업 생성 및 마지막 업데이트 타임스탬프
- 현재 작업 상태(예: 작업이 진행 중인지, 완료되었는지 또는 실패했는지 여부)
- 가져오기 작업에 사용되는 IAM 역할 ARN

완료된 작업의 경우 결과를 검토하여 성공적인 데이터 통합을 확인합니다. 작업이 실패하면 오류 세부 정보를 검사하여 문제를 진단하고 해결합니다.

*job-ID*를 검색하려는 대량 가져오기 작업의 ID로 대체하세요.

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

Example 응답

```
{
  "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"
}

```

AWS IoT SiteWise 대량 가져오기 작업 나열(AWS CLI)

[ListBulkImportJobs](#) API 작업을 사용하여 AWS IoT SiteWise의 일괄 가져오기 작업에 대한 요약 목록을 가져옵니다. 이 작업은 데이터 가져오기 프로세스를 모니터링하고 관리하는 효율적인 방법을 제공합니다. 각 작업에 대해 다음 키 정보를 반환합니다.

- 작업 ID. 각 일괄 가져오기 작업에 대한 고유 식별자
- 작업 이름. 생성 시 작업에 할당된 이름
- 현재 상태. 작업의 현재 상태(예: 완료됨, 실행 중, 실패)

ListBulkImportJobs는 모든 일괄 가져오기 작업에 대한 포괄적인 개요를 가져오는 데 특히 유용합니다. 이를 통해 여러 데이터 가져오기를 추적하고, 주의가 필요한 작업을 식별하고, 구성된 워크플로를 유지할 수 있습니다. 이 작업은 페이지 매김을 지원하므로 많은 수의 작업 요약에 효율적으로 검색할 수 있습니다. [DescribeBulkImportJob](#) 작업과 함께 이 작업에서 반환된 작업 ID를 사용하여 특정 작업에 대한 자세한 정보를 검색할 수 있습니다. 이 2단계 프로세스를 통해 먼저 모든 작업을 개괄적으로 확인한 다음 관심 작업의 세부 정보를 심층 분석할 수 있습니다. ListBulkImportJobs를 사용할 때 필터를 적용하여 결과를 좁힐 수 있습니다. 예를 들어 상태를 기준으로 작업을 필터링하여 완료된 작업만 검색하거나 실행 중인 작업만 검색할 수 있습니다. 이 기능을 사용하면 현재 작업과 가장 관련성이 높은 정보에 집중할 수 있습니다. 사용 가능한 결과가 더 있는 경우에도 작업은 nextToken을 반환합니다. 후속 직접 호출에서 이 토큰을 사용하여 다음 작업 요약 세트를 검색할 수 있으므로 일괄 가져오기 작업이 많더라도 모든 작업을 반복할 수 있습니다. 다음 예제에서는와 ListBulkImportJobs 함께를 사용하여 완료된 작업 목록을 검색 AWS CLI 하는 방법을 보여줍니다.

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

Example 완료된 작업 필터에 대한 응답

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

이 명령은 ListBulkImportJobs를 사용하여 실패로 완료된 작업 목록을 검색하는 방법을 보여 줍니다. 최댓값은 결과 50개로 설정되며 페이지 매김된 결과에 다음 토큰을 사용합니다.

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


AWS IoT Core 규칙을 AWS IoT SiteWise 사용하여 데이터 수집

의 규칙을 사용하여 AWS IoT 사물 및 기타 AWS 서비스 AWS IoT SiteWise 에서 로 데이터를 전송합니다 AWS IoT Core. 규칙은 MQTT 메시지를 변환하고 AWS 서비스와 상호 작용하는 작업을 수행합니다. AWS IoT SiteWise 규칙 작업은 AWS IoT SiteWise API에서 [BatchPutAssetPropertyValue](#) 작업으로 메시지 데이터를 전달합니다. 자세한 내용은 AWS IoT 개발자 가이드의 [규칙](#) 및 [AWS IoT SiteWise 작업을 참조하세요](#).

디바이스 새도우를 통해 데이터를 수집하는 규칙을 설정하는 데 필요한 단계를 안내하는 자습서를 따르면 [AWS IoT 사물 AWS IoT SiteWise 에서 로 데이터 수집](#) 섹션을 참조하세요.

에서 다른 AWS 서비스로 데이터를 전송할 수도 AWS IoT SiteWise 있습니다. 자세한 내용은 [다른 AWS 서비스와 상호 작용](#) 단원을 참조하십시오.

주제

- [AWS IoT 필요한 액세스 권한 부여](#)
- [AWS IoT SiteWise 규칙 작업 구성](#)
- [에서 Basic Ingest를 사용하여 비용 절감 AWS IoT SiteWise](#)

AWS IoT 필요한 액세스 권한 부여

IAM 역할을 사용하여 각 규칙이 액세스할 수 있는 AWS 리소스를 제어합니다. 규칙을 생성하기 전에 규칙이 필요한 AWS 리소스에 대한 작업을 수행하도록 허용하는 정책이 있는 IAM 역할을 생성해야 합니다. 규칙을 실행할 때이 역할을 수 AWS IoT 임합니다.

AWS IoT 콘솔에서 규칙 작업을 생성하는 경우 루트 자산을 선택하여 선택한 자산 계층 구조에 액세스할 수 있는 역할을 생성할 수 있습니다. 규칙에 대한 역할을 수동으로 정의하는 방법에 대한 자세한 내용은 AWS IoT 개발자 안내서 [AWS IoT 의 필수 액세스 권한 부여](#) 및 [역할 전달](#)을 참조하세요.

AWS IoT SiteWise 규칙 작업의 경우 규칙이 데이터를 전송하는 자산 속성에 대한 `iotsitewise:BatchPutAssetPropertyValue` 액세스를 허용하는 역할을 정의해야 합니다. 보안을 강화하기 위해 Condition 속성에서 AWS IoT SiteWise 자산 계층 구조 경로를 지정할 수 있습니다.

다음 예제 신뢰 정책은 특정 자산 및 해당 자산의 하위 자산에 대한 액세스를 허용합니다.

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

```
{
  "Effect": "Allow",
  "Action": "iotsitewise:BatchPutAssetPropertyValue",
  "Resource": "*",
  "Condition": {
    "StringLike": {
      "iotsitewise:assetHierarchyPath": [
        "/root node asset ID",
        "/root node asset ID/*"
      ]
    }
  }
}
```

정책에서 Condition을 제거하여 모든 자산에 대한 액세스를 허용합니다. 다음 예제 신뢰 정책은 현재 리전의 모든 자산에 대한 액세스를 허용합니다.

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

AWS IoT SiteWise 규칙 작업 구성

AWS IoT SiteWise 규칙 작업은 규칙을 시작한 MQTT 메시지의 데이터들의 자산 속성으로 전송합니다 AWS IoT SiteWise. 여러 데이터 항목을 동시에 서로 다른 자산 속성에 업로드하여 디바이스의 모든 센서에 대한 업데이트를 하나의 메시지로 보낼 수 있습니다. 각 데이터 입력에 대해 한 번에 여러 데이터 포인트를 업로드할 수도 있습니다.

Note

규칙 작업을 AWS IoT SiteWise 사용하여 데이터 전송할 때 데이터는 BatchPutAssetPropertyValue 작업의 모든 요구 사항을 충족해야 합니다. 예를 들어, 데

이터에 현재 Unix Epoch 시간에서 7일 이전인 타임스탬프가 있을 수 없습니다. 자세한 내용은 [AWS IoT SiteWise API를 사용하여 데이터 수집](#)을 참조하세요.

규칙 작업의 각 데이터 항목에 대해 자산 속성을 식별하고 해당 자산 속성에 대한 각 데이터 포인트의 타임스탬프, 품질 및 값을 지정합니다. 규칙 작업에는 모든 파라미터에 대한 문자열이 필요합니다.

항목에서 자산 속성을 식별하려면 다음 중 하나를 지정합니다.

- 데이터를 보낼 자산 속성의 자산 ID(assetId) 및 속성 ID(propertyId)입니다. AWS IoT SiteWise Console을 사용하여 자산 ID 및 속성 ID를 찾을 수 있습니다. 자산 ID를 알고 있는 경우를 사용하여 [DescribeAsset](#)을 호출 AWS CLI 하여 속성 ID를 찾을 수 있습니다.
- 데이터 스트림 별칭(예: /company/windfarm/3/turbine/7/temperature)인 속성 별칭(propertyAlias). 이 옵션을 사용하려면 먼저 자산 속성의 별칭을 설정해야 합니다. 속성 별칭을 설정하는 방법에 대한 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 단원을 참조하세요.

각 항목의 타임스탬프에 대해 장비에서 보고된 타임스탬프 또는 AWS IoT Core에서 제공한 타임스탬프를 사용할 수 있습니다. 타임스탬프에는 두 개의 매개변수가 있습니다.

- 초 단위 시간(timeInSeconds) – 센서 또는 장비가 데이터를 보고한 Unix Epoch 시간(초)입니다.
- 나노초 단위 오프셋(offsetInNanos) – (선택 사항) 초 단위 시간의 나노초 오프셋입니다.

Important

타임스탬프가 문자열이거나, 소수 부분이 있거나, AWS IoT SiteWise 에서 초 단위가 아닌 경우 요청을 거부합니다. 타임스탬프를 초 및 나노초 오프셋으로 변환해야 합니다. AWS IoT 규칙 엔진의 기능을 사용하여 타임스탬프를 변환합니다. 자세한 내용은 다음 자료를 참조하세요.

- [정확한 시간을 보고하지 않는 디바이스의 타임스탬프 가져오기](#)
- [문자열 형식의 타임스탬프 변환](#)

작업의 여러 파라미터에 대한 대체 템플릿을 사용하여 계산하고 함수를 호출하며 메시지 페이로드에서 값을 가져올 수 있습니다. 자세한 내용은 AWS IoT 개발자 가이드의 [대체 템플릿](#)을 참조하세요.

Note

대체 템플릿의 표현식은 SELECT 문과 별도로 평가되므로 대체 템플릿을 사용하면 AS 절을 사용하여 생성한 별칭은 참조할 수 없습니다. 지원되는 함수와 연산자 외에 원래 페이로드에 있는 정보만 참조할 수 있습니다.

주제

- [정확한 시간을 보고하지 않는 디바이스의 타임스탬프 가져오기](#)
- [문자열 형식의 타임스탬프 변환](#)
- [나노초 정밀도의 타임스탬프 문자열 변환](#)
- [규칙 구성의 예](#)
- [규칙 작업 문제 해결](#)

정확한 시간을 보고하지 않는 디바이스의 타임스탬프 가져오기

센서 또는 장비가 정확한 시간 데이터를 보고하지 않는 경우 [타임스탬프\(\)](#)를 사용하여 AWS IoT 규칙 엔진에서 현재 Unix epoch 시간을 가져옵니다. 이 함수는 시간을 밀리초 단위로 출력하므로 값을 초 단위 시간 및 나노초 단위 오프셋으로 변환해야 합니다. 이렇게 하려면 다음 변환을 사용합니다.

- 초 단위 시간(timeInSeconds)의 경우 $\text{\$}\{\text{floor}(\text{timestamp}() / 1\text{E}3)\}$ 를 사용하여 시간을 밀리초에서 초로 변환합니다.
- 나노초 단위 오프셋(offsetInNanos)의 경우 $\text{\$}\{(\text{timestamp}() \% 1\text{E}3) * 1\text{E}6\}$ 을 사용하여 타임스탬프의 나노초 오프셋을 계산합니다.

문자열 형식의 타임스탬프 변환

센서 또는 장비가 시간 데이터를 문자열 형식(예: 2020-03-03T14:57:14.699Z)으로 보고하는 경우 [time_to_epoch\(문자열, 문자열\)](#)를 사용합니다. 이 함수는 타임스탬프와 형식 패턴을 매개변수로 입력하고 시간을 밀리초 단위로 출력합니다. 그런 다음 시간을 초 단위의 시간으로, 나노초 단위의 오프셋으로 변환해야 합니다. 이렇게 하려면 다음 변환을 사용합니다.

- 초 단위 시간(timeInSeconds)의 경우 $\text{\$}\{\text{floor}(\text{time\_to\_epoch}(\text{"2020-03-03T14:57:14.699Z"}, \text{"yyyy-MM-dd'T'HH:mm:ss'Z'"})) / 1\text{E}3)\}$ 타임스탬프 문자열을 밀리초로 변환한 다음 초로 변환하는 데 사용합니다.

- 나노초 단위 오프셋(offsetInNanos)의 경우

`${(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z')) % 1E3) * 1E6}`을 사용하여 타임스탬프 문자열의 나노초 오프셋을 계산합니다.

Note

이 `time_to_epoch` 함수는 최대 밀리초 정밀도의 타임스탬프 문자열을 지원합니다. 마이크로초 또는 나노초 정밀도로 문자열을 변환하려면 규칙이 타임스탬프를 숫자 값으로 변환하기 위해 호출하는 AWS Lambda 함수를 구성합니다. 자세한 내용은 [나노초 정밀도의 타임스탬프 문자열 변환](#) 단원을 참조하십시오.

나노초 정밀도의 타임스탬프 문자열 변환

디바이스가 나노초 정밀도의 문자열 형식(예: 2020-03-03T14:57:14.699728491Z)으로 타임스탬프 정보를 전송하는 경우 다음 절차에 따라 규칙 작업을 구성합니다. 문자열의 타임스탬프를 초 단위의 시간(`timeInSeconds`) 및 나노 단위의 오프셋(`offsetInNanos`)으로 변환하는 AWS Lambda 함수를 생성할 수 있습니다. 규칙 작업 파라미터에 [aws_lambda\(functionArn, inputJson\)](#)를 사용하여 Lambda 함수를 간접적으로 호출하고 출력을 규칙에 사용합니다.

Note

이 단원에서는 다음 리소스를 만드는 방법을 잘 안다고 가정하고 고급 지침을 제공합니다.

- Lambda 함수. 자세한 내용은 AWS Lambda 개발자 안내서의 [첫 Lambda 함수 생성](#)을 참조하세요.
- AWS IoT 규칙 작업이 포함된 AWS IoT SiteWise 규칙. 자세한 내용은 [AWS IoT Core 규칙을 AWS IoT SiteWise 사용하여 데이터 수집](#) 단원을 참조하십시오.

타임스탬프 문자열을 구문 분석하는 AWS IoT SiteWise 규칙 작업을 생성하려면

1. 다음 속성을 사용하여 Lambda 함수를 만듭니다.

- 함수 이름 - 설명이 포함된 함수 이름(예: **ConvertNanosecondTimestampFromString**)을 사용합니다.
- 런타임 - Python 3 런타임(예: Python 3.11(`python3.11`))을 사용합니다.

- 권한 - 기본 Lambda 권한(AWS LambdaBasicExecutionRole)을 사용하여 역할을 생성합니다.
- 계층 - numpy를 사용하기 위해 Lambda 함수에 대한 AWS SDKPandas-Python311 계층을 추가합니다.
- 함수 코드 - timestamp라는 문자열 인수를 사용하고 해당 타임스탬프에 대해 timeInSeconds 및 offsetInNanos 값을 출력하는 다음 함수 코드를 사용합니다.

```
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
    }
```

이 Lambda 함수는 NumPy의 [datetime64](#)를 사용하여 [ISO 8601](#) 형식의 타임스탬프 문자열을 입력합니다.

Note

타임스탬프 문자열이 ISO 8601 형식이 아닌 경우 타임스탬프 형식을 정의하는 pandas로 솔루션을 구현할 수 있습니다. 자세한 내용은 [pandas.to_datetime](#)을 참조하세요.

2. 규칙에 대한 AWS IoT SiteWise 작업을 구성할 때 초 단위 시간(timeInSeconds) 및 나노 단위 오프셋()에 대해 다음 대체 템플릿을 사용합니다. 이러한 대체 템플릿은 메시지 페이로드에 타임스탬프 문자열(timestamp)이 포함되어 있다고 가정합니다. aws_lambda 함수는 두 번째 파라미터에 JSON 구조를 사용하므로 필요한 경우 아래 대체 템플릿을 수정할 수 있습니다.

- 초 단위 시간(timeInSeconds)에 다음 대체 템플릿을 사용합니다.

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

- 나노초 단위 오프셋(offsetInNanos)에 다음 대체 템플릿을 사용합니다.

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

각 파라미터에 대해 **region** 및 **account-id**를 리전 및 AWS 계정 ID로 바꿉니다. Lambda 함수에 다른 이름을 사용한 경우 해당 이름도 변경합니다.

- AWS IoT 권한을 사용하여 함수를 호출할 수 있는 `lambda:InvokeFunction` 권한을 부여합니다. 자세한 내용은 [aws_lambda\(functionArn, inputJson\)](#)를 참조하세요.
- 규칙을 테스트하고(예: AWS IoT MQTT 테스트 클라이언트 사용)가 전송하는 데이터를 AWS IoT SiteWise 수신하는지 확인합니다.

규칙이 예상대로 작동하지 않을 경우 [AWS IoT SiteWise 규칙 작업 문제 해결](#) 단원을 참조하세요.

Note

이 솔루션은 각 타임스탬프 문자열에 대해 Lambda 함수를 두 번 호출합니다. 현재 규칙이 각 페이로드에서 동일한 타임스탬프를 갖는 여러 데이터 포인트를 처리하는 경우 Lambda 함수 호출 수를 줄이기 위해 다른 규칙을 만들 수 있습니다.

이렇게 하려면 재게시 작업을 사용하여 Lambda를 호출하고 timeInSeconds 및 offsetInNanos로 변환된 타임스탬프 문자열을 갖는 원래 페이로드를 게시하는 규칙을 만듭니다. 그런 다음 변환된 페이로드를 사용하는 규칙 작업을 사용하여 AWS IoT SiteWise 규칙을 생성합니다. 이 접근 방식을 사용하면 규칙이 Lambda를 간접적으로 호출하는 횟수를 줄이지만 AWS IoT 규칙 작업 실행 횟수는 증가합니다. 이 솔루션을 사용 사례에 적용하는 경우 각 서비스의 요금을 고려하세요.

규칙 구성의 예

이 섹션에는 AWS IoT SiteWise 작업으로 규칙을 생성하는 예제 규칙 구성이 포함되어 있습니다.

Example 속성 별칭을 메시지 주제로 사용하는 규칙 작업의 예제

다음 예제에서는 ([topic\(\)](#))를 통해) 주제를 속성 별칭으로 사용하여 자산 속성을 식별하는 AWS IoT SiteWise 작업을 사용하여 규칙을 생성합니다. 이 예시를 사용하여 모든 풍력 발전소의 모든 풍력 터빈에 배정밀도 부동산소수점 유형 데이터를 수집하는 규칙 하나를 정의할 수 있습니다. 이 예제에서는 모든 터빈 자산의 속성에 속성 별칭을 정의해야 합니다. 정수 유형 데이터를 수집하기 위한 유사한 두 번째 규칙을 정의해야 합니다.

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

sitewise-rule-payload.json의 예제 페이로드에는 다음 내용이 포함되어 있습니다.

```
{
  "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"
  }
}
]
}

```

이 규칙 동작을 사용하여 데이터를 수집하기 위해 다음과 같은 메시지를 풍력 터빈 속성 별칭(예: /company/windfarm/3/turbine/7/temperature)에 보냅니다.

```

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

```

Example timestamp()를 사용하여 시간을 결정하는 규칙 동작 예제

다음 예제에서는 IDs로 자산 속성을 식별하고 [timestamp\(\)](#)를 사용하여 현재 시간을 결정하는 AWS IoT SiteWise 작업이 포함된 규칙을 생성합니다.

```

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

```

sitewise-rule-payload.json의 예제 페이로드에는 다음 내용이 포함되어 있습니다.

```

{
  "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"
}
}
]
}

```

이 규칙 작업을 사용하여 데이터를 수집하기 위해 my/asset/property/topic에 다음 메시지를 보냅니다.

```

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

```

규칙 작업 문제 해결

에서 AWS IoT SiteWise 규칙 작업 문제를 해결하려면 CloudWatch Logs를 AWS IoT Core구성하거나 규칙에 대한 재게시 오류 작업을 구성합니다. 자세한 내용은 [AWS IoT SiteWise 규칙 작업 문제 해결](#) 단원을 참조하십시오.

에서 Basic Ingest를 사용하여 비용 절감 AWS IoT SiteWise

AWS IoT Core 는 [AWS IoT 메시징 비용](#)을 발생시키지 AWS IoT Core 애플리케이션을 통해 데이터를 전송하는 데 사용할 수 있는 Basic Ingest라는 기능을 제공합니다. Basic Ingest는 수집 경로에서 게시/구독 메시지 브로커를 제거하여 대용량 데이터 수집 워크로드에 대한 데이터 흐름을 최적화합니다. 메시지를 라우팅해야 하는 규칙을 알고 있는 경우 Basic Ingest를 사용할 수 있습니다.

Basic Ingest를 사용하려면 특수 주제 \$aws/rules/*rule-name*을 사용하여 특정 규칙으로 직접 메시지를 보냅니다. 예를 들어 SiteWiseWindFarmRule이라는 규칙에 메시지를 보내려면 주제 \$aws/rules/SiteWiseWindFarmRule에 메시지를 보냅니다.

규칙 작업에서 [주제\(소수\)](#)가 포함된 대체 템플릿을 사용하는 경우 Basic Ingest 특수 주제(예: \$aws/rules/*rule-name/original-topic*)의 끝에서 원래 주제를 전달할 수 있습니다. 예를 들어 이전 섹션의 풍력 발전소 속성 별칭 예제와 함께 Basic Ingest를 사용하려면 다음 주제로 메시지를 보낼 수 있습니다.

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

Note

위의 예제에서는가 규칙 작업에 표시되는 주제에서 기본 수집 접두사(//)를 AWS IoT 제거하기 때문에 두 번째 슬래시(\$aws/rules/*rule-name/*)를 포함합니다. 이 예제에서 규칙은 주제 /company/windfarm/3/turbine/7/temperature를 받습니다.

자세한 내용은 AWS IoT 개발자 가이드의 [Basic Ingest를 사용하여 메시징 비용 절감](#)을 참조하세요.

AWS IoT SiteWise 에서 로 데이터 수집 AWS IoT Events

를 사용하면 AWS 클라우드에서 IoT 플릿에 대한 복잡한 이벤트 모니터링 애플리케이션을 구축할 AWS IoT Events수 있습니다. 이벤트가 발생할 AWS IoT SiteWise 때의 IoT SiteWise 작업을 사용하여의 자산 속성으로 데이터를 AWS IoT Events 전송합니다.

Note

지원 종료 알림: 2026 AWS 년 5월 20일에에 대한 지원이 종료됩니다 AWS IoT Events. 2026 년 5월 20일 이후에는 AWS IoT Events 콘솔 또는 AWS IoT Events 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 [AWS IoT Events 지원 종료를 참조하세요](#).

AWS IoT Events 는 AWS 클라우드 내 IoT 디바이스 및 시스템에 대한 이벤트 모니터링 애플리케이션의 개발을 간소화하도록 설계되었습니다. 를 사용하여 다음을 AWS IoT Events수행할 수 있습니다.

- IoT 플릿 전체에서 변경 사항, 이상 또는 특정 조건을 감지하고 이에 대응합니다.
- 운영 효율성을 높이고 IoT 에코시스템을 사전 예방적으로 관리합니다.

AWS IoT SiteWise 작업을 AWS IoT SiteWise 통해와를 통합하면가 기능을 AWS IoT Events 확장하므로 특정 이벤트에 대한 응답 AWS IoT SiteWise 으로에서 자산 속성을 자동으로 업데이트할 수 있습니다

다. 이러한 상호 작용은 데이터 수집 및 관리를 단순화할 수 있습니다. 또한 실행 가능한 인사이트를 제공할 수 있습니다.

자세한 내용은 AWS IoT Events 개발자 설명서에서 다음 주제를 참조하세요.

- [란 무엇입니까 AWS IoT Events?](#)
- [AWS IoT Events 작업](#)
- [IoT SiteWise 작업](#)

에서 AWS IoT Greengrass 스트림 관리자 사용 AWS IoT SiteWise

AWS IoT Greengrass 스트림 관리자는 로컬 소스에서 AWS 클라우드로 데이터 스트림을 쉽게 전송할 수 있는 통합 기능입니다. 데이터 흐름을 관리하는 중간 계층 역할을 하여 엣지에서 작동하는 디바이스가 데이터를 전송하기 전에 데이터를 수집 및 저장하여 AWS IoT SiteWise 추가 분석 및 처리할 수 있습니다.

AWS IoT SiteWise 콘솔에서 로컬 소스를 구성하여 데이터 대상을 추가합니다. 사용자 지정 AWS IoT Greengrass 솔루션에서 스트림 관리자를 사용하여 데이터에 수집할 수도 있습니다 AWS IoT SiteWise.

Note

OPC UA 소스에서 데이터를 수집하려면에서 실행되는 AWS IoT SiteWise Edge 게이트웨이를 구성합니다 AWS IoT Greengrass. 자세한 내용은 [AWS IoT SiteWise Edge 게이트웨이 사용](#) 단원을 참조하십시오.

로컬 소스 데이터의 대상을 구성하는 방법에 대한 자세한 내용은 [AWS IoT SiteWise 엣지 대상 이해](#) 섹션을 참조하세요.

사용자 지정 AWS IoT Greengrass 솔루션에서 스트림 관리자를 사용하여 데이터를 수집하는 방법에 대한 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 다음 주제를 참조하세요.

- [란 무엇입니까 AWS IoT Greengrass?](#)
- [AWS IoT Greengrass Core에서 데이터 스트림 관리](#)
- [AWS IoT SiteWise 자산 속성으로 데이터 내보내기](#)

AWS IoT SiteWise Edge 게이트웨이 사용

AWS IoT SiteWise Edge는 클라우드 기능을 산업 엣지 환경으로 확장하여 로컬 데이터 처리, 분석 및 의사 결정을 지원합니다. SiteWise Edge는 AWS IoT SiteWise 및 기타 AWS 서비스와 통합되어 포괄적인 산업 IoT 솔루션을 제공합니다. 게이트웨이는 산업 장비와 간의 중개자 역할을 합니다 AWS IoT SiteWise.

SiteWise Edge 게이트웨이는 두 가지 배포 대상에서 실행됩니다.

- AWS IoT Greengrass V2
- Siemens Industrial Edge

SiteWise Edge 게이트웨이를 사용하여 엣지에서 데이터를 수집하여 클라우드에 게시할 수 있습니다. 에서 실행되는 게이트웨이의 경우 자산 모델 및 자산을 사용하여 엣지에서 데이터를 처리할 수도 AWS IoT Greengrass 있습니다.

의 AWS IoT SiteWise Edge 애플리케이션은 산업 장비와 간의 통합을 Siemens Industrial Edge 지원 AWS IoT SiteWise 하므로 AWS 클라우드로 정교한 데이터를 보내기 전에 원시 기계 데이터를 집계 및 처리하고 로컬에서 분석을 실행할 수 있습니다.

SiteWise Edge 게이트웨이의 주요 개념

SiteWise Edge에는 산업 환경의 엣지 컴퓨팅에 유용한 몇 가지 기능이 있습니다.

로컬 데이터 수집 및 처리

OPC-UA 및 MQTT와 같은 프로토콜을 사용하여 산업 자산에서 데이터 수집을 지원합니다. 게이트웨이는 AWS IoT Greengrass 코어 디바이스 또는에서 실행됩니다 Siemens Industrial Edge.

오프라인 작업

인터넷 중단 시 데이터를 계속 수집하고 처리하여 연결이 복원될 때 클라우드와 동기화합니다.

AWS IoT Greengrass 구성 요소를 사용한 엣지 컴퓨팅

IoT SiteWise 게시자를 사용하여 로컬 변환 및 계산을 위해 클라우드 및 AWS IoT SiteWise 프로세서에 데이터를 전달합니다. 게시자와 프로세서는 모두 AWS IoT Greengrass V2 구성 요소입니다. AWS IoT Greengrass 구성 요소에 대한 자세한 내용은 제공 구성 [AWS요소를 참조하세요](#).

와 통합하여 클라우드 기능 AWS IoT SiteWise 확장

AWS IoT SiteWise 클라우드 기능과 함께 작동하여 자산 모델, 분석 및 대시보드를 엣지로 확장합니다.

데이터 처리 팩이 활성화된 게이트웨이의 경우 AWS OpsHub 용 AWS IoT SiteWise 를 사용하여 SiteWise Edge 게이트웨이를 중앙에서 관리할 수 있습니다.는 원격 관리 및 모니터링 기능을 AWS OpsHub 제공합니다. 자세한 내용은 [용을 사용하여 SiteWise Edge 게이트웨이 관리 AWS OpsHubAWS IoT SiteWise](#) 단원을 참조하십시오.

파트너 데이터 소스 통합

파트너 데이터 소스를 게이트웨이에 연결하고 SiteWise Edge 게이트웨이 및 AWS 클라우드의 파트너로부터 데이터를 수신합니다. 자세한 내용은 [SiteWise Edge 게이트웨이의 파트너 데이터 소스](#) 단원을 참조하십시오.

엣지의 로컬 시각화

엣지에서 실시간 인사이트를 얻을 수 있는 사용자 지정 대시보드를 제공합니다.

로컬 디바이스의 SiteWise Monitor 포털을 사용하여 시설에서 로컬로 데이터를 모니터링합니다. 자세한 내용은 [엣지에서 AWS IoT SiteWise 포털 활성화](#) 단원을 참조하십시오.

SiteWise Edge 구현의 이점

SiteWise Edge는 산업 운영 및 의사 결정 프로세스를 크게 개선할 수 있는 다양한 이점을 제공합니다.

- 클라우드 처리 지연이 없는 실시간 운영 인사이트
- 연결이 끊긴 환경의 운영 연속성
- 엣지 사전 처리를 통해 대역폭 및 스토리지 비용 절감
- 로컬 데이터 기반 의사 결정 기능으로 안정성 향상

를 사용하여 AWS IoT SiteWise Edge 게이트웨이 자체 호스팅 AWS IoT Greengrass V2

클라우드로 전송하기 전에 산업 장비에서 데이터를 로컬로 수집, 처리 및 시각화하도록 AWS IoT SiteWise Edge를 설정합니다. 를 사용하여 자체 호스팅합니다 AWS IoT Greengrass Version 2.

AWS IoT SiteWise Edge 게이트웨이는 산업 장비와 간의 중개자 역할을 합니다 AWS IoT SiteWise. 에서 실행되는 SiteWise Edge 게이트웨이 AWS IoT Greengrass Version 2는 온프레미스에서 데이터 수집 및 처리를 지원합니다. 데이터 처리 팩을 활성화하고 AWS OpsHub 설치한 상태에서 로컬 디바이스의 SiteWise Monitor 포털을 통해 시설 내에서 로컬로 데이터를 모니터링합니다.

자체 호스팅 게이트웨이에는 두 가지 유형이 있습니다.

MQTT 지원, V3 게이트웨이

MQTT 지원 V3 게이트웨이 아키텍처는 향상된 데이터 수집 기능을 제공합니다. 효율적인 데이터 통신을 위해 MQTT 프로토콜을 활용하고 구성 가능한 데이터 대상을 제공합니다. 여기에는 Amazon S3를 사용하여 버퍼링된 데이터를 수집하는 옵션과 실시간 데이터 수집이 포함됩니다. 경로 필터를 구현하여 특정 MQTT 주제를 구독하여 대상 데이터 수집을 활성화할 수 있습니다. MQTT 지원 V3 게이트웨이는 데이터 처리 팩 기능을 지원하지 않습니다. 자세한 내용은 [MQTT 지원, AWS IoT SiteWise Edge용 V3 게이트웨이](#) 단원을 참조하십시오.

클래식 스트림, V2 게이트웨이

클래식 스트림인 V2 게이트웨이는 기존 AWS IoT SiteWise 엣지 게이트웨이 아키텍처를 나타냅니다. 기존 SiteWise Edge 배포 및 설정된 워크플로에 익숙한 사용자에게 적합합니다. 클래식 스트림에서 V2 게이트웨이는 데이터 처리 팩을 지원하지만 데이터 처리 팩에서 생성된 데이터는 Amazon S3를 통해 수집할 수 없습니다. 기존 배포와의 호환성을 유지해야 하거나 데이터 처리 팩 기능이 필요한 경우 클래식 스트림, V2 게이트웨이를 사용합니다. 자세한 내용은 [클래식 스트림, AWS IoT SiteWise Edge용 V2 게이트웨이](#) 단원을 참조하십시오.

주제

- [AWS IoT SiteWise 엣지 자체 호스팅 게이트웨이 요구 사항](#)
- [자체 호스팅 SiteWise Edge 게이트웨이 생성](#)
- [로컬 디바이스에 AWS IoT SiteWise Edge 게이트웨이 소프트웨어 설치](#)
- [MQTT 지원, AWS IoT SiteWise Edge용 V3 게이트웨이](#)
- [클래식 스트림, AWS IoT SiteWise Edge용 V2 게이트웨이](#)
- [AWS IoT SiteWise Edge 게이트웨이에 데이터 소스 추가](#)
- [AWS IoT GreengrassAWS IoT SiteWise Edge용 구성 요소](#)
- [SiteWise Edge 게이트웨이에서 자산 필터링](#)
- [프록시 지원 구성 및 AWS IoT SiteWise Edge용 트러스트 스토어 관리](#)
- [엣지에서 AWS IoT SiteWise APIs 사용](#)

AWS IoT SiteWise 엣지 자체 호스팅 게이트웨이 요구 사항

AWS IoT SiteWise 엣지 게이트웨이는 온프레미스에서 데이터 수집, 처리 및 게시를 지원하는 구성 AWS IoT Greengrass 요소 세트 AWS IoT Greengrass V2 로에서 실행됩니다. 에서 실행되는 SiteWise Edge 게이트웨이를 구성하려면에서 게이트웨이를 AWS IoT Greengrass V2 생성하고 SiteWise Edge 게이트웨이 소프트웨어를 AWS 클라우드 실행하여 로컬 디바이스를 설정합니다. AWS Management Console 를 사용하여 SiteWise Edge 게이트웨이를 생성하면 설치 스크립트가 제공됩니다. 대상 게이트웨이 디바이스에서이 스크립트를 실행하여 필요한 소프트웨어 및 종속성을 설정합니다.

로컬 디바이스 요구 사항

SiteWise Edge 게이트웨이 소프트웨어를 설치하고 실행하려면 로컬 디바이스가 다음 요구 사항을 충족해야 합니다.

- AWS IoT Greengrass V2 코어 소프트웨어 버전 [v2.3.0](#) 이상을 지원합니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [권한](#)을 참조하세요.
- 다음 지원 플랫폼 중 하나입니다.
 - 운영 체제: Ubuntu 20.04 이상
아키텍처: x86_64 (AMD64) 또는 ARMv8 (Aarch64)
 - 운영체제: Red Hat Enterprise Linux (RHEL) 8
아키텍처: x86_64 (AMD64) 또는 ARMv8 (Aarch64)
 - 운영체제: Amazon Linux 2
아키텍처: x86_64 (AMD64) 또는 ARMv8 (Aarch64)
 - 운영체제: Debian 11
아키텍처: x86_64 (AMD64) 또는 ARMv8 (Aarch64)
 - 운영체제: Windows Server 2019 이상
아키텍처: x86_64 (AMD64)

Note

ARM 플랫폼은 데이터 수집 팩이 있는 SiteWise Edge 게이트웨이만 지원합니다. 데이터 처리 팩은 지원되지 않습니다.

- 최소 4GB RAM

- SiteWise Edge 게이트웨이 소프트웨어에 최소 10GB의 디스크 공간을 사용할 수 있습니다.
- 적절한 포트에 액세스할 수 있도록 로컬 디바이스를 구성합니다. 필요한 아웃바운드 서비스 엔드포인트의 전체 목록은 [AWS IoT SiteWise Edge 게이트웨이에 필요한 서비스 엔드포인트를 참조하세요](#).
- Java 런타임 환경(JRE) 버전 11 이상. 디바이스의 PATH 환경 변수에서 Java를 사용할 수 있어야 합니다. Java를 사용하여 사용자 정의 구성 요소를 개발하려면 JDP(Java Development Kit)를 설치해야 합니다. [Amazon Corretto](#) 또는 [OpenJDK](#)를 사용하는 것이 좋습니다.

로컬 디바이스에 대한 허용 목록에 대한 Amazon S3 버킷

다음 Amazon S3 버킷에 대한 방화벽 액세스를 제공하도록 로컬 디바이스를 구성합니다. 디바이스의 각 리전을 기반으로 액세스를 구성합니다.

지역	엔드포인트
아시아 태평양(도쿄)	<code>https://iot-sitewise-gateway-ap-northeast-1-785558802005.s3.ap-northeast-1.amazonaws.com</code>
아시아 태평양(서울)	<code>https://iot-sitewise-gateway-ap-northeast-2-310055672453.s3.ap-northeast-2.amazonaws.com</code>
아시아 태평양(뭄바이)	<code>https://iot-sitewise-gateway-ap-south-1-677656657204.s3.ap-south-1.amazonaws.com</code>
아시아 태평양(싱가포르)	<code>https://iot-sitewise-gateway-ap-southeast-1-475191558554.s3.ap-southeast-1.amazonaws.com</code>
아시아 태평양(시드니)	<code>https://iot-sitewise-gateway-ap-southeast-2-396319432685.s3.ap-southeast-2.amazonaws.com</code>
캐나다(중부)	<code>https://iot-sitewise-gateway-ca-central-1-842060018567.s3.ca-central-1.amazonaws.com</code>
중국(베이징)	<code>https://iot-sitewise-gateway-cn-north-1-237124890262.s3.cn-north-1.amazonaws.com.cn</code>
유럽(프랑크푸르트)	<code>https://iot-sitewise-gateway-eu-central-1-748875242063.s3.eu-central-1.amazonaws.com</code>

지역	엔드포인트
유럽(아일랜드)	https://iot-sitewise-gateway-eu-west-1-383414315062.s3.eu-west-1.amazonaws.com
미국 동부(버지니아 북부)	https://iot-sitewise-gateway-us-east-1-223558168232.s3.us-east-1.amazonaws.com 및 https://iot-sitewise-gateway-us-east-1-223558168232.s3.amazonaws.com/
미국 동부(오하이오)	https://iot-sitewise-gateway-us-east-2-005072661813.s3.us-east-2.amazonaws.com
AWS GovCloud(미국 서부)	https://iot-sitewise-gateway-us-gov-west-1-599984565679.s3.us-gov-west-1.amazonaws.com/
미국 서부(오리건)	https://iot-sitewise-gateway-us-west-2-502577205460.s3.us-west-2.amazonaws.com

데이터 처리 팩 요구 사항

- 엣지에서 데이터 처리 팩을 사용하려는 경우 AWS IoT SiteWise로컬 디바이스도 다음 요구 사항을 충족해야 합니다.
 - x86 64비트 쿼드코어 프로세서가 장착되어 있습니다.
 - 최소 16GB의 RAM이 있습니다.
 - 를 사용하는 경우 RAM의 경우 최소 32GB입니다Microsoft Windows.
 - 최소 256GB의 디스크 여유 공간이 있습니다.
 - 로컬 디바이스가 포트 443에서 인바운드 트래픽을 허용해야 합니다.
 - 다음 포트는 AWS IoT SiteWise 80, 443, 3001, 4569, 4572, 8000, 8081, 8082, 8084, 8085, 8445, 8086, 9000, 9500, 11080 및 50010에서 사용하도록 예약되어 있습니다. 예약된 포트를 트래픽용으로 사용하면 연결이 종료될 수 있습니다.

Note

AWS IoT Greengrass V2 스트림 관리자 구성 요소에는 자체 요구 사항이 있습니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [구성](#)을 참조하세요.

- 최소 디스크 공간 및 컴퓨팅 용량 요구 사항은 구현 및 사용 사례에 고유한 다양한 요인에 따라 달라집니다.
- 간헐적인 인터넷 연결을 위해 데이터 캐싱에 필요한 디스크 공간은 다음 요인에 따라 다릅니다.
 - 업로드되는 데이터 스트림의 개수
 - 초당 데이터 스트림당 데이터 포인트
 - 각 데이터 포인트의 크기
 - 통신 속도
 - 예기치 않은 네트워크 다운타임
- 데이터를 폴링하고 업로드하는 데 필요한 컴퓨팅 파워는 다음 요인에 따라 달라집니다.
 - 업로드되는 데이터 스트림의 개수
 - 초당 데이터 스트림당 데이터 포인트

SiteWise Edge 게이트웨이를 사용하도록 권한 구성

SiteWise Edge 게이트웨이를 사용하려면 다음 권한이 있어야 합니다.

Note

AWS IoT SiteWise 콘솔을 사용하여 SiteWise Edge 게이트웨이를 생성하는 경우 이러한 권한이 추가됩니다.

- SiteWise Edge 게이트웨이의 IAM 역할은 AWS IoT Greengrass V2 디바이스에서 SiteWise Edge 게이트웨이를 사용하여 자산 모델 데이터와 자산 데이터를 처리할 수 있도록 허용해야 합니다.

이 역할은 역할을 수입하기 위해 `credentials.iot.amazonaws.com` 서비스를 신뢰합니다.

권한 세부 정보

또한 이 역할에는 다음과 같은 권한도 있어야 합니다.

- `iotsitewise` - 보안 주체가 엣지에서 자산 모델 데이터 및 자산 데이터를 검색할 수 있도록 합니다.
- `iot` - AWS IoT Greengrass V2 디바이스가 상호 작용할 수 있도록 허용합니다 AWS IoT.
- `logs` - AWS IoT Greengrass V2 디바이스가 Amazon CloudWatch Logs로 로그를 전송할 수 있도록 허용합니다.

- s3 - AWS IoT Greengrass V2 디바이스가 Amazon S3에서 사용자 지정 구성 요소 아티팩트를 다운로드할 수 있도록 허용합니다.

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

자체 호스팅 SiteWise Edge 게이트웨이 생성

AWS IoT SiteWise 콘솔 또는를 사용하여 자체 호스팅 SiteWise Edge 게이트웨이를 AWS CLI 생성합니다. 이 절차에서는 자체 하드웨어에 설치할 자체 호스팅 SiteWise Edge 게이트웨이를 생성하는 방법

을 자세히 설명합니다. Siemens Industrial Edge에서 실행되는 SiteWise Edge 게이트웨이 생성에 대한 자세한 내용은 [Siemens Industrial Edge에서 SiteWise Edge 게이트웨이 호스팅](#) 섹션을 참조하세요.

SiteWise Edge 게이트웨이 생성

Console

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 게이트웨이 생성을 선택합니다.
4. 배포 대상 선택에서 자체 호스팅 게이트웨이를 선택합니다.
5. MQTT 지원 V3 게이트웨이 또는 클래식 스트림 V2 게이트웨이를 선택합니다. 각 옵션에 대한 자세한 내용은 섹션을 참조하세요. [를 사용하여 AWS IoT SiteWise Edge 게이트웨이 자체 호스팅 AWS IoT Greengrass V2](#). MQTT 지원 V3 게이트웨이는 미래 대비 기능에 권장됩니다.
6. 게이트웨이 구성 섹션에서 SiteWise Edge 게이트웨이의 이름을 입력하거나에서 생성된 이름을 사용합니다 AWS IoT SiteWise.
7. Greengrass 디바이스 OS에서 이 SiteWise Edge 게이트웨이를 설치할 디바이스의 운영 체제를 선택합니다.

Note

데이터 처리 팩은 x86 플랫폼에서만 사용할 수 있습니다. Classic 스트림, V2 게이트웨이에서만 사용할 수 있습니다.

8. (선택 사항) 엣지에서 데이터를 처리하고 구성하려면 엣지 기능에서 데이터 처리 팩을 선택합니다.

Note

회사 디렉터리의 사용자 그룹에 이 SiteWise Edge 게이트웨이에 대한 액세스 권한을 부여하려면 [SiteWise Edge에서 엣지 기능 설정](#) 섹션을 참조하세요.

9. (선택 사항) 고급 구성에서 다음을 수행합니다.
 - Greengrass 코어 디바이스의 경우 다음 옵션 중 하나를 선택합니다.
 - 기본 설정 -는 AWS 자동으로 기본 설정을 사용하여 Greengrass 코어 디바이스를 생성합니다 AWS IoT Greengrass V2.

1. Greengrass 코어 디바이스의 이름을 입력하거나에서 생성된 이름을 사용합니다
AWS IoT SiteWise.
- 고급 설정 – 기존 Greengrass Core 디바이스를 사용하거나 수동으로 생성하려는 경우
이 옵션을 선택합니다.
1. Greengrass 코어 디바이스를 선택하거나 Greengrass 코어 디바이스 생성을 선택하여 AWS IoT Greengrass V2 콘솔에서 생성합니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [AWS IoT Greengrass V2 코어 디바이스 설정](#) 섹션을 참조하세요.
10. 게이트웨이 생성을 선택합니다.
11. SiteWise Edge 게이트웨이 설치 관리자 생성 대화 상자에서 생성 및 다운로드를 선택합니다. AWS IoT SiteWise 는 로컬 디바이스를 구성하는 데 사용할 수 있는 설치 관리자를 자동으로 생성합니다.

Important

이 파일은 재생성할 수 없습니다. 설치 관리자 파일은 나중에 다시 사용하므로 안전한 위치에 저장해야 합니다.

AWS CLI

를 사용하여 자체 호스팅 게이트웨이를 생성하려면 게이트웨이의 이름을 AWS CLI 제공하고 플랫폼과 게이트웨이 버전을 지정합니다. 게이트웨이를 생성할 때 지정할 수 있는 다른 여러 옵션이 있습니다. 자세한 내용은에 대한 명령 참조의 [create-gateway](#)를 참조하세요. AWS CLI AWS IoT SiteWise

이 예시를 사용하려면 사용자 입력 자리 표시자를 실제 정보로 대체하세요.

```
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 - 게이트웨이의 고유한 이름입니다.

- `gateway-platform` - 게이트웨이 플랫폼 구성을 지정합니다. 자체 호스팅 게이트웨이에를 입력합니다 `greengrassv2`. 자세한 내용은 명령 참조의 `create-gateway` 섹션에서 [옵션을](#) 참조하세요 AWS IoT SiteWise. AWS CLI
- `coreDeviceThingName` - AWS IoT Greengrass V2 코어 디바이스에 대한 AWS IoT 사물의 이름입니다.
- `coreDeviceOperatingSystem` - 코어 디바이스의 운영 체제입니다 AWS IoT Greengrass V2. 운영 체제 지정은에 필수 `gateway-version 3`이며 에는 적용되지 않습니다 `gateway-version 2`. 옵션에는 `LINUX_AARCH64`, 및 `LINUX_AMD64`가 포함됩니다. `WINDOWS_AMD64`
- `gateway-version` - 게이트웨이의 버전입니다.
 - MQTT 지원 V3 게이트웨이를 생성하려면 게이트웨이 버전3에를 사용합니다.
 - Classic 스트림인 V2 게이트웨이를 생성하려면 게이트웨이 버전2에를 사용합니다.
- `cli-input-json` - 요청 파라미터가 포함된 JSON 파일입니다.

이제 SiteWise Edge 게이트웨이를 생성했습니다 [로컬 디바이스에 AWS IoT SiteWise Edge 게이트웨이 소프트웨어 설치](#).

로컬 디바이스에 AWS IoT SiteWise Edge 게이트웨이 소프트웨어 설치

AWS IoT SiteWise Edge 게이트웨이를 생성한 후 로컬 디바이스에 SiteWise Edge 게이트웨이 소프트웨어를 설치합니다. SiteWise Edge 게이트웨이 소프트웨어는 Linux 또는 Microsoft Windows 서버 운영 체제가 설치된 로컬 디바이스에 설치할 수 있습니다.

Important

로컬 디바이스가 인터넷에 연결되어 있는지 확인합니다.

Linux

다음 절차에서는 SSH를 사용하여 로컬 디바이스에 연결합니다. 또는 USB 플래시 드라이브 또는 기타 도구를 사용하여 설치 프로그램 파일을 로컬 디바이스로 전송할 수 있습니다. SSH를 사용하지 않으려면 아래의 2단계: SiteWise Edge 게이트웨이 소프트웨어 설치로 건너뛵니다.

SSH 사전 조건

SSH를 사용하여 디바이스에 연결하려면 먼저 다음 사전 조건을 완료하세요.

- Linux 및 macOS - OpenSSH를 다운로드하여 설치합니다. 자세한 내용은 <https://www.openssh.com>을 참조하세요

1단계: 설치 프로그램을 SiteWise Edge 게이트웨이 디바이스에 복사

다음 지침에서는 SSH 클라이언트를 사용하여 로컬 디바이스에 연결하는 방법을 설명합니다.

1. 디바이스에 연결하려면 컴퓨터의 터미널 창에서 다음 명령을 실행하여 **### ### IP**를 승격된 권한과 IP 주소가 있는 사용자 이름으로 바꿉니다.

```
ssh username@IP
```

2. 에서 AWS IoT SiteWise 생성한 설치 관리자 파일을 SiteWise Edge 게이트웨이 디바이스로 전송하려면 다음 명령을 실행합니다.

 Note

- *path-to-saved-installer*를 설치 프로그램 파일과 설치 프로그램 파일 이름을 저장하는 데 사용한 컴퓨터의 경로로 바꾸세요.
- *IP-address*를 로컬 디바이스의 IP 주소로 바꿉니다.
- *directory-to-receive-installer*를 설치 프로그램 파일을 받는 데 사용하는 로컬 장치의 경로로 바꿉니다.

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

2단계: SiteWise Edge 게이트웨이 소프트웨어 설치

다음 절차에서는 SiteWise Edge 게이트웨이 디바이스의 터미널 창에서 명령을 실행합니다.

1. 설치 파일에 실행 권한을 부여합니다.

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

2. 설치 관리자를 실행합니다.


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

Windows Server

사전 조건

SiteWise Edge 게이트웨이 소프트웨어를 설치하려면 다음과 같은 사전 조건이 필요합니다.

- Microsoft Windows Server 2019 이상 설치됨
- 관리자 권한
- PowerShell 버전 5.1 이상 설치
- 프로비저닝될 Windows Server에 SiteWise Edge 게이트웨이 설치 프로그램 다운로드

1단계: 관리자 권한으로 PowerShell 실행

1. SiteWise Edge 게이트웨이를 설치하려는 Windows 서버에 관리자로 로그인합니다.
2. Windows 검색창에 PowerShell을 입력합니다.
3. 검색 결과에서 Windows PowerShell 앱의 컨텍스트 메뉴를 엽니다(오른쪽 클릭). [Run as Administrator]를 선택합니다.

2단계: SiteWise Edge 게이트웨이 소프트웨어 설치

SiteWise Edge 게이트웨이 디바이스의 터미널 창에서 다음 명령을 실행합니다.

1. SiteWise Edge 게이트웨이 설치 프로그램의 차단을 해제합니다.

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

2. 설치 관리자를 실행합니다.

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

Note

시스템에서 스크립트 실행이 비활성화된 경우 스크립트 실행 정책을 RemoteSigned으로 변경합니다.

Set-ExecutionPolicy RemoteSigned

다음 단계는 필요한 자체 호스팅 게이트웨이 유형에 따라 다릅니다. [MQTT 지원, AWS IoT SiteWise Edge용 V3 게이트웨이](#) 또는 [로 계속 진행합니다클래식 스트림, AWS IoT SiteWise Edge용 V2 게이트웨이](#).

MQTT 지원, AWS IoT SiteWise Edge용 V3 게이트웨이

AWS IoT SiteWise 는 SiteWise Edge 게이트웨이 아키텍처의 상당한 발전을 나타내는 MQTT 지원 V3 게이트웨이를 사용할 수 있습니다. 이 게이트웨이 유형은 데이터 통신을 위해 MQTT(Message Queuing Telemetry Transport) 프로토콜을 활용하여 산업용 IoT 배포에서 향상된 유연성과 효율성을 제공합니다.

MQTT 지원 V3 게이트웨이는 데이터 전송에 MQTT를 사용하여 디바이스와 클라우드 간에 메시지를 효율적으로 전송하는 경량의 게시-구독 네트워크 프로토콜을 지원합니다. Amazon S3를 사용하여 직접 실시간 데이터 수집 AWS IoT SiteWise 및 버퍼링된 데이터 수집을 비롯한 다양한 데이터 대상을 설정할 수 있습니다. 정확한 데이터 수집을 활성화하기 위해 경로 필터를 구현하여 특정 MQTT 주제를 구독할 수 있습니다.

MQTT를 사용하는 V3 게이트웨이에는 필터가 "#" (모든 주제)로 설정된 사전 구성된 실시간 대상이 제공되며, 필요에 따라 사용자 지정하거나 제거할 수 있습니다. 데이터 관리를 간소화하기 위해 각 게이트웨이에는 하나의 실시간 대상만 존재할 수 있습니다.

MQTT 지원 아키텍처는 Classic 스트림인 V2 게이트웨이와 크게 다릅니다. V2는 스트림 기반 접근 방식을 사용하지만 V3는 MQTT를 사용하여 더 구성 가능한 데이터 대상과 필터링 옵션을 제공합니다. 그러나 V3는 V2에서 사용할 수 있는 데이터 처리 팩을 지원하지 않습니다.

MQTT 지원 V3 게이트웨이는 다음과 같은 몇 가지 이점을 제공합니다.

- MQTT의 경량 특성으로 인해 확장성이 개선되어 수많은 디바이스를 더 잘 처리하고 데이터를 자주 전송할 수 있습니다.
- 경로 필터를 통해 데이터 제어를 개선하여 데이터 수집을 세밀하게 관리하고 불필요한 데이터 전송 및 처리를 줄일 수 있습니다.
- 유연한 데이터 처리를 통해 특정 요구 사항에 따라 실시간 처리와 버퍼링된 스토리지 간에 구성할 수 있습니다.
- 최신 IoT 통신 표준에 맞게 조정하여 향후 개선 및 통합을 위한 단계를 설정합니다.

특히 유연한 데이터 수집 옵션과 데이터 수집에 대한 정확한 제어가 필요한 경우 새 배포를 위해 MQTT 지원 V3 게이트웨이를 채택하는 것이 좋습니다.

Note

데이터 처리 팩이 필요한 기존 배포 또는 시나리오의 경우 Classic 스트림, V2 게이트웨이는 여전히 실행 가능한 옵션입니다.

두 게이트웨이 유형을 모두 제공하여 고급 MQTT 기능을 우선시하든 기존 시스템과의 호환성을 우선시하든 관계없이 특정 산업 IoT 요구 사항에 가장 적합한 솔루션을 선택할 수 있는지 AWS IoT SiteWise 확인합니다.

대상 및 경로 필터

MQTT 지원 게이트웨이의 대상 및 경로 필터에 대해 자세히 알아보려면 다음 주제를 참조하세요.

- [AWS IoT SiteWise 엣지 대상 이해](#)
- [AWS IoT SiteWise Edge 실시간 대상 추가](#)
- [Amazon S3를 사용하여 AWS IoT SiteWise 버퍼링된 대상 추가](#)
- [AWS IoT SiteWise Edge 대상의 경로 필터 이해](#)
- [AWS IoT SiteWise Edge 대상에 경로 필터 추가](#)
- [AWS IoT SiteWise 엣지 대상 관리](#)

외부 애플리케이션을 EMQX 브로커에 연결

이 가이드에서는 배포된 MQTT 지원 V3 게이트웨이의 EMQX 브로커를 통해 외부 애플리케이션을 AWS IoT SiteWise Edge 게이트웨이에 연결하는 방법을 설명합니다. 외부 애플리케이션에는 사용자 지정 모니터링 도구, 타사 시각화 소프트웨어 또는 엣지에서 산업 데이터와 상호 작용해야 하는 레거시 시스템이 포함될 수 있습니다.

EMQX 배포 구성, 보안 연결을 위한 TLS 설정, 특정 주제에 대한 액세스를 제어하는 권한 부여 규칙을 포함하여 Linux 및 Microsoft Windows 환경 모두에 대한 구성 단계를 다룹니다.

Note

EMQX는 AWS IoT SiteWise Edge의 공급업체 또는 공급업체가 아닙니다.

⚠ Important

게이트웨이에 대한 연결을 보호하려면 AWS IoT Greengrass 클라이언트 디바이스 인증 기능을 통해 인증서 기반 인증을 사용하는 것이 좋습니다. 이 방법은 상호 TLS(mTLS) 인증을 통해 강력한 보안을 제공합니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [코어 디바이스에 클라이언트 디바이스 연결](#)을 참조하세요.

인증서 기반 인증을 사용할 수 없는 경우 이 가이드에 따라 사용자 이름과 암호를 사용하여 인증을 설정합니다.

사전 조건

- 배포되어 온라인 상태인 SiteWise Edge MQTT 지원 V3 게이트웨이
- 게이트웨이 호스트에 대한 액세스
- AWS IoT SiteWise 및 AWS IoT Greengrass 콘솔에 대한 액세스

주제

- [Edge의 EMQX 브로커에 대한 메시지 AWS IoT SiteWise 페이로드 형식](#)
- [EMQX 브로커 구성](#)
- [애플리케이션을 Edge의 EMQX AWS IoT SiteWise 브로커에 연결](#)
- [EMQX에서 AWS IoT SiteWise Edge에 대한 권한 부여 규칙 설정](#)

Edge의 EMQX 브로커에 대한 메시지 AWS IoT SiteWise 페이로드 형식

IoT SiteWise 게시자 구성 요소가 외부 애플리케이션의 데이터를 사용하고 AWS IoT SiteWise 클라우드에 게시하려면 브로커로 전송된 페이로드가 특정 요구 사항을 충족해야 합니다.

페이로드 형식을 이해하는 것이 AWS IoT SiteWise Edge와의 성공적인 MQTT 통신의 핵심입니다. 연결 설정 프로세스는 이후 단원에서 다루지만 구현을 계획하는 데 도움이 되도록 먼저 페이로드 요구 사항을 제공합니다.

MQTT 주제 요구 사항

사용된 수준 또는 문자 수를 포함하여 MQTT 주제 구조에는 제한이 없습니다. 그러나 주제가 페이로드의 `propertyAlias` 필드와 일치하는 것이 좋습니다.

Example 속성 별칭 예제

MQTT 주제가 인 경우 propertyAlias 일치하는지 site1/line1/compressor1/temperature 확인합니다.

```
{
  "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 페이로드 구조

MQTT 메시지 페이로드는 JSON으로 작성되며 [AWS IoT SiteWise API 참조](#)에 정의된 PutAssetPropertyValueEntry 메시지 형식을 따릅니다.

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

```

```

        "stringValue": "string"
      }
    }
  ]
}

```

Note

메시지가 유효한 것으로 간주되려면 다음 조건 중 하나만 true일 수 있습니다.

- propertyAlias가 설정되어 있거나
- assetId 및 모두 설정propertyId됨

PutAssetPropertyValueEntry 에는이 컨텍스트에서 필요하지 않은 entryId 필드가 있습니다.

EMQX 브로커 구성

이 섹션에서는 사용자 이름과 암호를 추가하는 방법을 다룹니다. 또한 추가된 사용자 이름과 암호를 사용하여 외부 소스에서 TLS 연결을 설정하는 방법도 다룹니다. Linux 또는를 사용하여 EMQX 브로커를 구성할 수 있습니다Microsoft Windows.

Note

브로커를 구성하려면 MQTT 지원 V3 게이트웨이에서 기본 EMQX 구성으로 설정된 코어 디바이스가 필요합니다.

Important

이 절차를 완료한 후에는 권한 부여 규칙을 구성하는 것이 좋습니다. 자세한 내용은 [EMQX에서 AWS IoT SiteWise Edge에 대한 권한 부여 규칙 설정](#) 단원을 참조하십시오. 추가된 사용자를 위한 권한 부여 규칙은 보안을 강화합니다.

인증을 위한 EMQX 배포 구성 업데이트

인증을 위해 EMQX 배포 구성을 업데이트하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 구성할 게이트웨이를 선택합니다.
4. 엣지 게이트웨이 구성 섹션에서 Greengrass 코어 디바이스 값을 복사합니다. 나중에 사용할 수 있도록 저장합니다.
5. [AWS IoT 콘솔](#)을 엽니다.
6. 왼쪽 탐색 창의 관리 섹션에서 Greengrass 디바이스를 선택한 다음 배포를 선택합니다.
7. 이전에 저장한 코어 디바이스 값을 찾아 해당 링크를 선택하여 배포를 엽니다.
8. 작업 드롭다운 버튼을 선택한 다음 수정을 선택합니다.
9. 표시되는 메시지를 읽은 다음 배포 수정을 선택합니다. 대상 지정 페이지가 나타납니다.
10. 구성 요소 구성 단계에 도달할 때까지 다음을 선택합니다.
11. `aws.greengrass.clientdevices.mqtt.EMQX` 라디오 버튼을 선택합니다.
12. 구성 요소 구성 버튼을 선택합니다. 구성 요소에 대한 구성 페이지가 나타납니다.
13. 구성 업데이트에서 구성 요소 버전 2.*.*의 기본 구성으로 재설정을 선택합니다.
14. OS를 기반으로 병합할 구성 섹션에 다음 구성을 입력합니다.

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"
}

```

`dockerOptions` 필드는 Linux 게이트웨이에만 해당됩니다.

15. 확인을 선택합니다.
16. 검토 단계에 도달할 때까지 다음을 선택합니다.
17. 배포(Deploy)를 선택합니다.

18. 배포가 성공하면 다음 단계로 진행합니다.

사용자 이름 및 암호 인증 활성화

이 섹션에서는 EMQX 대시보드 GUI를 통해 사용자 이름과 암호를 추가하는 방법을 보여줍니다.

Note

제공된 EMQX 관련 지침은 참조용일 뿐입니다. EMQX 설명서 및 기능은 시간이 지남에 따라 변경될 수 있으며 설명서를 유지 관리하지 않으므로 최신 정보는 [EMQX의 공식 설명서를](#) 참조하는 것이 좋습니다.

EMQX Dashboard

EMQX 대시보드를 통해 사용자 이름 및 암호 인증을 활성화하려면

1. 게이트웨이 호스트 내에 있는지 확인합니다.
2. 브라우저 창을 열고를 방문합니다<http://localhost:18083/>.
3. 의 기본 사용자 이름과의 **admin** 기본 암호를 입력합니다**public**. 자세한 내용은 [EMQX 문서의 EMQX 대시보드](#)를 참조하세요.
4. 로그인한 후 암호를 변경하라는 메시지가 표시됩니다. 암호를 업데이트하여 EMQX 대시보드로 계속 진행합니다.
5. 왼쪽 탐색 창에서 방패 아이콘을 선택한 다음 인증을 선택합니다.
6. 기본 제공 데이터베이스 행에서 사용자 버튼을 선택합니다.
7. 더하기 기호 아이콘 버튼을 선택하여 사용자를 추가합니다. 추가 화면이 나타납니다.
8. 외부 애플리케이션 사용자의 사용자 이름과 암호를 입력합니다.
9. 저장을 선택합니다. 선택한 사용자 이름이 인증 페이지의 테이블에 나타납니다.

Note

기존 또는 기본 권한 부여 규칙이 새 사용자에게 적용됩니다. 이를 검토하고 외부 애플리케이션 요구 사항에 맞게 조정하는 것이 좋습니다.

EMQX Management with Linux

에서 AWS IoT SiteWise EMQX CLI 도구를 사용합니다/greengrass/v2/bin/swe-emqx-cli.

Linux를 사용하여 EMQX Management를 통해 사용자 이름 및 암호 인증을 활성화하려면

1. 다음 명령을 실행하여 관리자 암호를 변경합니다.

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

2. 메시지가 표시되면 다음을 수행합니다.

1. 현재 관리자 사용자(기본값은 admin)와 암호(기본값은)를 입력합니다public.
2. 새 암호를 입력하고 확인합니다.

성공하면 다음 메시지가 표시됩니다.

```
admin password changed successfully
```

3. 다음 명령을 실행하여 외부 애플리케이션의 사용자를 추가합니다.

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

4. 메시지가 표시되면 다음을 수행합니다.

1. 새 사용자의 사용자 이름을 입력합니다.
2. 새 사용자의 암호를 입력하고 확인합니다.

성공하면 다음 메시지가 표시됩니다.

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

5. 다음 명령을 실행하여 사용자 구성을 확인합니다.

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

출력에는 구성된 모든 사용자가 표시됩니다.

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

```
Total users: 1
```

EMQX Management with Windows

다음 위치 중 하나에서 AWS IoT SiteWise EMQX CLI 도구를 사용합니다.

- PowerShell: C:\greengrass\v2\bin\swe-emqx-cli.ps1
- 명령 프롬프트: C:\greengrass\v2\bin\swe-emqx-cli.bat

Windows를 사용하여 EMQX Management를 통해 사용자 이름 및 암호 인증을 활성화하려면

1. 다음 명령을 실행하여 관리자 암호를 변경합니다.

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

2. 메시지가 표시되면 다음을 수행합니다.

1. 현재 관리자 사용자(기본값은 admin)와 암호(기본값은)를 입력합니다public.
2. 새 암호를 입력하고 확인합니다.

성공하면 다음 메시지가 표시됩니다.

```
admin password changed successfully
```

3. 다음 명령을 실행하여 외부 애플리케이션의 사용자를 추가합니다.

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

4. 메시지가 표시되면 다음을 수행합니다.

1. 새 사용자의 사용자 이름을 입력합니다.
2. 새 사용자의 암호를 입력하고 확인합니다.

성공하면 다음 메시지가 표시됩니다.

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

5. 다음 명령을 실행하여 사용자 구성을 확인합니다.

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

출력에는 구성된 모든 사용자가 표시됩니다.

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

Total users: 1
```

애플리케이션을 Edge의 EMQX AWS IoT SiteWise 브로커에 연결

EMQX 브로커는 포트 8883에서 전송 계층 보안(TLS)을 사용하여 모든 통신을 암호화하므로 전송 중에 데이터를 보호할 수 있습니다. 이 섹션에서는 애플리케이션과 EMQX 브로커 간의 연결을 설정하는 단계를 안내합니다. 이러한 단계를 따르면 산업 데이터의 무결성과 기밀성을 유지하는 데 도움이 됩니다. 연결 프로세스에는 구성 요소를 통한 자동 IP 검색을 사용하거나 TLS 인증서에서 DNS 이름과 IP 주소를 주체 대체 이름(SANs)으로 수동으로 구성하는 두 가지 주요 접근 방식이 포함됩니다. 각 메서드에는 네트워크 설정 및 보안 요구 사항에 따라 고유한 이점이 있습니다. 이 설명서는 두 옵션을 모두 안내합니다.

주제

- [AWS IoT SiteWise Edge에서 EMQX 브로커에 대한 보안 연결을 위해 TLS 구성](#)
- [AWS IoT SiteWise Edge에서 EMQX 브로커 연결 테스트](#)
- [자체 CA 사용](#)
- [외부 방화벽 연결을 위해 포트 8883 열기](#)

AWS IoT SiteWise Edge에서 EMQX 브로커에 대한 보안 연결을 위해 TLS 구성

기본적으로는 코어 디바이스 인증 기관(CA)에서 서명한 EMQX 브로커에 대한 TLS 서버 인증서를 AWS IoT Greengrass 생성합니다. 자세한 내용은 [MQTT 브로커를 사용하여 AWS IoT Greengrass 코어 디바이스에 클라이언트 디바이스 연결을 참조하세요](#).

TLS 인증서 검색

CA 인증서를 가져오려면 게이트웨이 호스트에서 다음 명령을 실행합니다.

Linux

게이트웨이 호스트의 셸 세션에서 다음 명령을 실행합니다.

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

이 명령은 인증서 위치를 표시하고 인증서의 콘텐츠를 인쇄합니다.

또는 다음 명령을 사용하여 인증서를 파일에 저장할 수 있습니다.

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

Windows

게이트웨이 호스트의 PowerShell 세션에서 다음 명령을 실행합니다.

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

이 명령은 인증서 위치를 표시하고 인증서의 콘텐츠를 인쇄합니다.

또는 다음 명령을 사용하여 인증서를 파일에 저장할 수 있습니다.

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

CLI는 시스템의 정확한 경로에 관계없이 인증서를 자동으로 찾습니다.

ca.pem 파일의 내용을 브로커에 연결하는 외부 애플리케이션에 복사합니다.

BrokerCoreDeviceCA.pem로 파일을 저장합니다.

TLS 서버 인증서에 사용자 지정 DNS 이름/IP 주소 추가

에서 생성된 인증서의 주체 대체 이름(SAN)은 AWS IoT Greengrass 입니다localhost. 게이트웨이 호스트 외부에서 TLS 연결을 설정하는 경우 브로커의 호스트 이름이 서버 인증서의 호스트 이름과 일치하지 않기 때문에 TLS 확인 단계가 실패localhost합니다.

일치하지 않는 호스트 이름 문제를 해결하기 위해는 코어 디바이스 엔드포인트를 관리하는 두 가지 방법을 AWS IoT Greengrass 제공합니다. 이 섹션에서는 두 옵션을 모두 다룹니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [코어 디바이스 엔드포인트 관리](#)를 참조하세요.

- 코어 디바이스의 IP 주소를 사용하여 EMQX 브로커에 연결하려면 자동 IP 검색 섹션을 사용합니다.
- IP 주소 대신 DNS 이름을 사용하여 EMQX 브로커에 연결하려면 수동 관리 섹션을 사용합니다.

Automated IP discovery

이 옵션을 사용하면 코어 디바이스가 IP 주소를 자동으로 검색하고 이를 브로커 인증서에 주체 대체 이름(SAN)으로 추가할 수 있습니다.

1. 코어 디바이스의 배포에 `aws.greengrass.clientdevices.IPDetector` 구성 요소를 추가합니다.
2. 디바이스에 변경 사항 배포
3. 배포가 완료될 때까지 기다립니다.

배포가 완료되면 브로커의 IP 주소를 사용하여 보안 TLS 연결을 설정할 수 있습니다.

IP 주소는 브로커 인증서에 자동으로 SAN으로 추가됩니다.

Manual DNS and IP Configuration

DNS 이름과 IP 주소를 TLS 인증서에 주체 대체 이름(SANs)으로 수동으로 추가할 수 있습니다. 이 방법은 게이트웨이 호스트의 DNS 이름을 구성한 경우에 유용합니다.

Important

IPDetector 구성 요소를 사용하는 경우 계속하기 전에 배포에서 제거합니다. IPDetector 구성 요소는 수동 엔드포인트 구성을 재정의합니다.

엔드포인트를 수동으로 구성하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 구성할 게이트웨이를 선택합니다.
4. 엣지 게이트웨이 구성 섹션에서 Greengrass 코어 디바이스 URL을 선택합니다. 코어 디바이스의 페이지가 나타납니다.
5. 클라이언트 디바이스 탭을 선택합니다.
6. 엔드포인트 관리를 선택합니다.
7. 엔드포인트 관리 대화 상자에서 DNS 이름(들)과 SANs으로 추가할 IP 주소를 입력합니다. 포트 8883을 사용합니다.

8. 업데이트를 선택합니다.

브로커의 TLS 서버 인증서는 새 엔드포인트를 포함하도록 자동으로 업데이트됩니다.

Linux를 사용하여 TLS 서버 인증서 업데이트를 확인하려면

1. 게이트웨이 호스트에서 셸 세션을 시작합니다.

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

2. 명령은 다음과 유사한 출력을 반환합니다.

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

3. 엔드포인트가 SANs 확인합니다.

Windows를 사용하여 TLS 서버 인증서 업데이트를 확인하려면

1. 게이트웨이 호스트에서 셸 세션을 시작합니다.

```
(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. 명령은 다음과 유사한 출력을 반환합니다.

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

3. 추가한 엔드포인트가 SANs 목록에 있는지 확인합니다.

AWS IoT SiteWise Edge에서 EMQX 브로커 연결 테스트

TLS 인증서 및 인증 자격 증명으로 EMQX 브로커를 구성한 후에는 설정이 올바르게 작동하는지 확인하는 것이 중요합니다. 연결을 테스트하면 보안 구성이 제대로 구현되고 클라이언트가 브로커에 암호화된 연결을 성공적으로 설정할 수 있습니다. 이 섹션에서는 TLS 암호화 및 인증을 지원하는 널리 사용

되는 MQTT 클라이언트 도구인 Mosquitto 명령줄 인터페이스(CLI) 클라이언트를 사용하여 브로커 연결을 테스트하는 방법을 보여줍니다.

Mosquitto CLI 클라이언트를 사용하여 EMQX 브로커 연결 테스트

이 단계에서는 mosquitto CLI 클라이언트를 사용하여 설정을 테스트하고 앞서 생성한 사용자 이름과 암호를 사용하여 브로커에 성공적으로 연결할 수 있는지 확인합니다. 3단계: TLS 설정에서 BrokerCoreDeviceCA.pem 다음 단계를 가져옵니다.

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

Note

연결하려는 호스트 이름/IP 주소가 클라이언트에 전달하는 CA 인증서에 있는 주체 대체 이름(SAN)과 일치하지 않는 경우 SSL:verify 오류가 발생할 수 있습니다. 올바른 SAN으로 인증서를 가져오는 방법은 3단계: TLS 설정에서 "TLS 서버 인증서에 사용자 지정 DNS 이름/IP 주소 추가"를 참조하세요.

이때 모든 사용자는 브로커의 모든 주제를 게시하고 구독할 수 있습니다. [EMQX에서 AWS IoT SiteWise Edge에 대한 권한 부여 규칙 설정](#)로 이동합니다.

자체 CA 사용

AWS IoT Greengrass에서는 자체 인증 기관(CA)을 사용하도록 자체 클라이언트 디바이스 인증 구성 요소를 구성하는 방법을 간략하게 설명합니다. 클라이언트 디바이스 인증 구성 요소(`aws.greengrass.clientdevices.Auth`)에서는 클라이언트 디바이스가 인증되고 클라이언트 디바이스 작업이 승인됩니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [자체 인증 기관 사용을 참조하세요](#).

자체 CA를 사용하려면 사용자 지정 구성을 지정할 수 있도록 배포에 `aws.greengrass.clientdevices.Auth` 구성 요소를 추가합니다.

외부 방화벽 연결을 위해 포트 8883 열기

Linux

Linux 호스트 방화벽 규칙에서 포트 8883에 대한 인바운드 규칙을 추가하여 게이트웨이 호스트 외부에서 들어오는 연결을 허용합니다. 방화벽이 있는 경우 포트 8883에서 들어오는 TLS 연결이 허용되는지 확인합니다.

Windows

Microsoft Windows 호스트 방화벽 규칙에서 포트 8883에 대한 인바운드 규칙을 추가하여 게이트웨이 호스트 외부에서 들어오는 연결을 허용합니다. 규칙이 포트 8883을 지정하는 포트 유형의 허용 규칙인지 확인합니다. 외부 애플리케이션에서 브로커로의 연결을 허용하도록 네트워크 구성에 따라 이를 구성할 수 있습니다.

EMQX에서 AWS IoT SiteWise Edge에 대한 권한 부여 규칙 설정

EMQX는 사용자 이름, IP 주소 또는 클라이언트 ID와 같은 식별자를 기반으로 권한 부여 규칙 추가를 지원합니다. 이는 다양한 작업 또는 주제에 연결하는 외부 애플리케이션의 수를 제한하려는 경우에 유용합니다.

주제

- [Linux에서 기본 제공 데이터베이스를 사용하여 권한 부여 구성](#)
- [Windows에서 기본 제공 데이터베이스를 사용하여 권한 부여 구성](#)
- [권한 부여를 위한 EMQX 배포 구성 업데이트](#)
- [사용자를 위한 EMQX 대시보드를 통해 권한 부여 규칙 추가](#)

Linux에서 기본 제공 데이터베이스를 사용하여 권한 부여 구성

권한 부여 규칙을 구성할 때 배포 설정에 따라 두 가지 구성 옵션이 있습니다.

- Docker - 없이 표준 Docker 설치를 실행하는 경우 Docker 브리지 게이트웨이 구성을 Litmus Edge 사용합니다. 이는 일반적으로 AWS IoT SiteWise 구성 요소만 배포한 경우입니다.
- Litmus Edge - 게이트웨이를 Litmus Edge 설치한 경우 Litmus Edge 네트워크 서브넷 구성을 사용합니다.

Note

Docker 브리지 게이트웨이를 처음 구성한 후 나중을 설치하는 경우 Litmus Edge 네트워크 서브넷 옵션을 사용하여 권한 부여 규칙을 Litmus Edge 재구성하여 모든 구성 요소 간에 적절한 통신을 보장합니다.

기본 권한 부여 규칙을 추가하려면

1. EMQX 브로커가 배포되고 실행 중인지 확인합니다.
2. 게이트웨이 호스트에서 셸 세션을 시작합니다.

Docker without Litmus Edge

없는 표준 Docker 설치의 경우 다음을 Litmus Edge 실행합니다.

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

Litmus Edge network subnet

를 사용하는 경우 Litmus Edge 네트워크 서브넷 IP를 Litmus Edge 확인합니다.

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

출력의 서브넷 값을 기록하고 다음 명령을 실행합니다. `litmus_subnet_ip`를 이전 단계의 서브넷 값으로 바꿉니다.

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

도구는 제공된 IP 주소에서 브로커로의 연결을 허용하는 권한 부여 규칙을 자동으로 생성하고 적용합니다. 이를 통해 모든 주제에 액세스할 수 있습니다. 여기에는 IoT SiteWise OPC UA 수집기 및 IoT SiteWise 게시자가 포함됩니다.

3. [권한 부여를 위한 EMQX 배포 구성 업데이트](#)로 이동합니다.

Windows에서 기본 제공 데이터베이스를 사용하여 권한 부여 구성

이 섹션에서는 Windows 배포용 기본 제공 데이터베이스를 사용하여 권한 부여 규칙을 구성하는 방법을 다룹니다.

기본 권한 부여 규칙을 추가하려면

1. EMQX 브로커가 배포되고 실행 중인지 확인합니다.
2. AWS IoT SiteWise EMQX CLI 도구를 실행합니다.

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

도구는 localhost(127.0.0.1)에서 브로커로의 연결을 허용하는 ACL 규칙을 자동으로 생성하고 적용합니다. 이를 통해 모든 주제에 액세스할 수 있습니다. 여기에는 IoT SiteWise OPC UA 수집기 및 IoT SiteWise 게시자가 포함됩니다.

3. [권한 부여를 위한 EMQX 배포 구성 업데이트](#)로 이동합니다.

권한 부여를 위한 EMQX 배포 구성 업데이트

권한 부여를 위해 EMQX 배포 구성을 업데이트하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 구성할 게이트웨이를 선택합니다.
4. 엣지 게이트웨이 구성 섹션에서 Greengrass 코어 디바이스 값을 복사합니다. 나중에 사용할 수 있도록 저장합니다.
5. [AWS IoT 콘솔](#)을 엽니다.
6. 왼쪽 탐색 창의 관리 섹션에서 Greengrass 디바이스를 선택한 다음 배포를 선택합니다.
7. 이전에 저장한 코어 디바이스 값을 찾아 해당 링크를 선택하여 배포를 엽니다.
8. 작업 드롭다운 버튼을 선택한 다음 수정을 선택합니다.
9. 표시되는 메시지를 읽은 다음 배포 수정을 선택합니다. 대상 지정 페이지가 나타납니다.
10. 구성 요소 구성 단계에 도달할 때까지 다음을 선택합니다.
11. `aws.greengrass.clientdevices.mqtt.EMQX` 라디오 버튼을 선택합니다.
12. 구성 요소 구성 버튼을 선택합니다. 구성 요소에 대한 구성 페이지가 나타납니다.
13. 구성 업데이트에서 구성 요소 버전의 기본 구성으로 재설정: 2.*.*를 선택합니다.

14. OS를 기반으로 병합할 구성 섹션에 다음 콘텐츠를 붙여 넣습니다.

Linux

```
{
  "emqxConfig": {
    "authorization": {
      "no_match": "deny",
      "sources": [
        {
          "type": "built_in_database"
        },
        {
          "type": "file",
          "path": "data/authz/acl.conf"
        }
      ]
    },
    "listeners": {
      "tcp": {
        "default": {
          "enabled": true,
          "enable_authn": false
        }
      },
      "ssl": {
        "default": {
          "enabled": true,
          "enable_authn": true,
          "ssl_options": {
            "verify": "verify_none",
            "fail_if_no_peer_cert": false
          }
        }
      }
    },
    "authentication": {
      "enable": true,
      "backend": "built_in_database",
      "mechanism": "password_based",
      "password_hash_algorithm": {
        "iterations": 210000,
        "mac_fun": "sha512",
```

```

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

```

Windows

```

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

```

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

15. 확인을 선택합니다.
16. 검토 단계에 도달할 때까지 다음을 선택합니다.
17. 배포(Deploy)를 선택합니다.

Note

이 시점부터는 ACL 파일을 편집하여 권한 부여 규칙을 업데이트할 수 없습니다. 또는 배포에 성공한 [사용자를 위한 EMQX 대시보드를 통해 권한 부여 규칙 추가](#) 후 로 진행할 수 있습니다.

사용자를 위한 EMQX 대시보드를 통해 권한 부여 규칙 추가

EMQX 대시보드 또는 AWS IoT SiteWise EMQX CLI 도구를 사용하여 권한 부여 규칙을 추가하거나 업데이트할 수 있습니다. EMQX CLI 도구는 AWS IoT SiteWise EMQX의 내장 데이터베이스를 사용하여 권한 부여를 관리합니다.

Note

권한 부여 규칙 추가는 MQTT 주제 패턴 및 액세스 제어를 이해해야 하는 고급 구성 단계입니다. EMQX의 기본 제공 데이터베이스를 사용하여 권한 부여 규칙을 생성하는 방법에 대한 자세한 내용은 EMQX 문서의 [기본 제공 데이터베이스 사용](#)을 참조하세요.

Note

제공된 EMQX 관련 지침은 참조용일 뿐입니다. EMQX 설명서 및 기능은 시간이 지남에 따라 변경될 수 있으며 설명서를 유지 관리하지 않으므로 최신 정보는 [EMQX의 공식 설명서](#)를 참조하는 것이 좋습니다.

EMQX dashboard

이 절차에서는 EMQX 대시보드에서 권한 부여 규칙을 추가하는 방법을 보여줍니다.

EMQX 대시보드는 게이트웨이 호스트 내에서만 액세스할 수 있습니다. 게이트웨이 호스트 외부에서 연결하려고 하면 대시보드에 액세스할 수 없습니다.

EMQX 대시보드를 사용하여 권한 부여 규칙을 추가하려면

1. 게이트웨이 호스트 내에 있는지 확인합니다.
2. 브라우저 창을 열고를 방문합니다 <http://localhost:18083/>.

3. EMQX 대시보드에 로그인합니다. 이 절차에서는 기본 로그인 자격 증명을 원하는 것으로 변경했다고 가정합니다. 초기 설정에 대한 자세한 내용은 섹션을 참조하세요 [사용자 이름 및 암호 인증 활성화](#).
4. 방패 아이콘을 선택한 다음 드롭다운 메뉴에서 권한 부여를 선택합니다.
5. 기본 제공 데이터베이스 행에서 권한 버튼을 선택합니다.
6. 기본 제공 데이터베이스 권한 부여 섹션에서 비즈니스 요구 사항에 맞는 사용자 권한 부여 규칙을 추가하거나 업데이트합니다. 규칙 생성에 대한 자세한 지침은 EMQX 문서의 [기본 제공 데이터베이스 사용](#) 섹션을 참조하세요.

AWS IoT SiteWise CLI tool using Linux

Linux에서 AWS IoT SiteWise EMQX CLI 도구를 사용하여 권한 부여 규칙을 관리하려면:

- 다음 형식을 사용하여 사용자에게 대한 권한 부여 규칙을 추가합니다.

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

Example 사용자에게 대한 권한 부여 규칙 추가

이 예제에서는 라는 사용자에게 대한 규칙을 추가하는 방법을 보여줍니다system1.

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

Example : 사용자에게 대한 권한 부여 규칙 보기

system1 사용자의 권한 부여 규칙을 보려면 다음 명령을 실행합니다.

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

Example : 모든 기존 권한 부여 규칙 보기

현재 보유한 모든 권한 부여 규칙을 보려면 다음 명령을 실행합니다.

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

Example : 사용자에게 대한 모든 권한 부여 규칙 삭제

특정 사용자에게 적용되는 모든 권한 부여 규칙을 삭제하려면 다음 명령을 실행합니다.

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

삭제를 확인하는 메시지가 표시됩니다.

AWS IoT SiteWise CLI tool using Windows

의 AWS IoT SiteWise EMQX CLI 도구를 사용하여 권한 부여 규칙을 관리하려면 Windows PowerShell:

- 다음 형식을 사용하여 사용자에게 대한 권한 부여 규칙을 추가합니다.

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

Example : 사용자에게 대한 권한 부여 규칙 추가

이 예제에서는 라는 사용자에게 대한 규칙을 추가하는 방법을 보여줍니다 system1.

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

Example : 사용자에게 대한 권한 부여 규칙 보기

system1 사용자의 권한 부여 규칙을 보려면 다음 명령을 실행합니다.

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

Example : 모든 기존 권한 부여 규칙 보기

현재 보유한 모든 권한 부여 규칙을 보려면 다음 명령을 실행합니다.

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

Example : 사용자에게 대한 모든 권한 부여 규칙 삭제

특정 사용자에게 적용되는 모든 권한 부여 규칙을 삭제하려면 다음 명령을 실행합니다.

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

삭제를 확인하는 메시지가 표시됩니다.

SiteWise Edge 및 오픈 소스 도구를 사용하여 데이터 처리 및 시각화

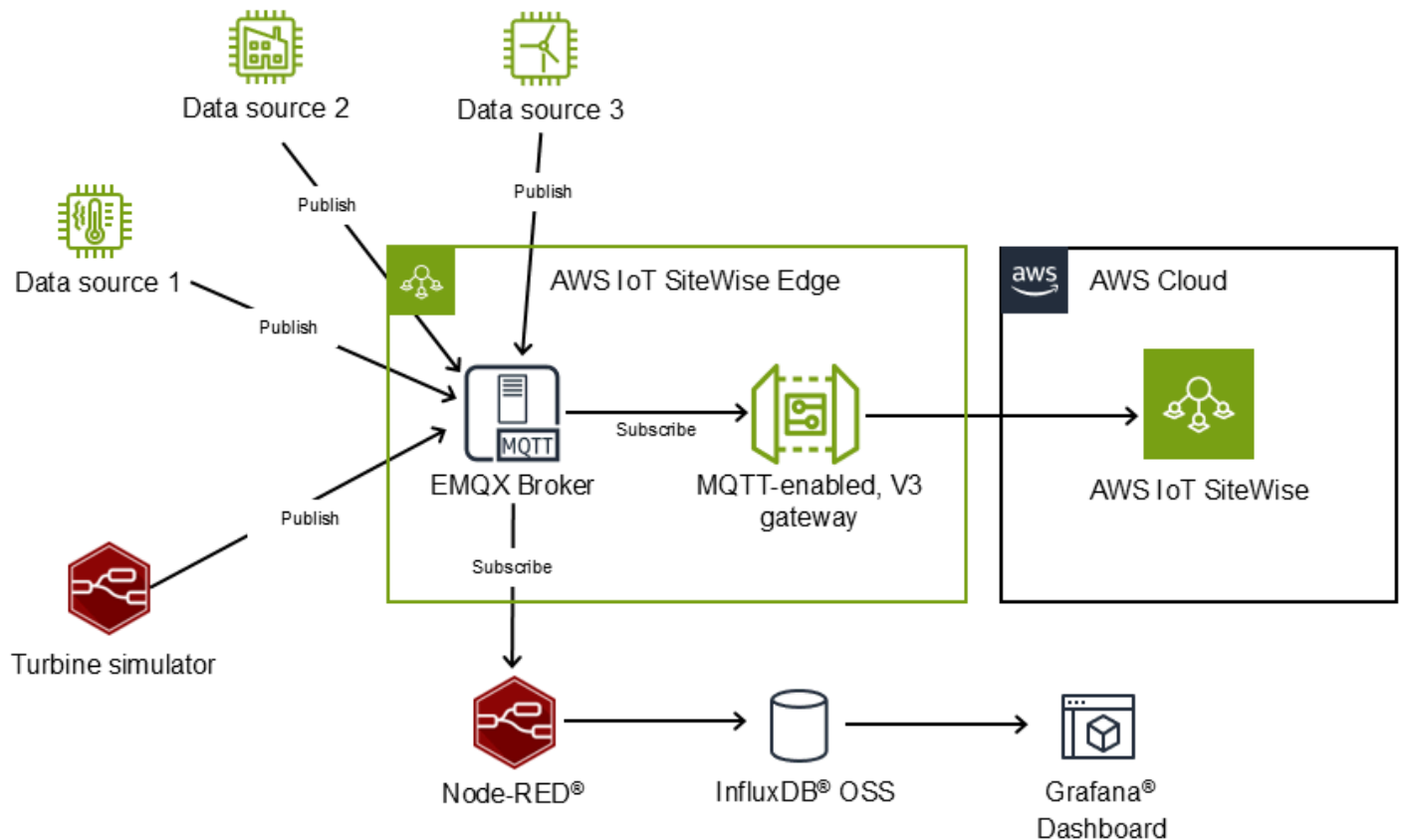
로컬 처리 및 시각화를 위한 오픈 소스 도구를 사용하여 AWS IoT SiteWise Edge MQTT 지원 게이트웨이를 구성하여 산업 데이터 관리 기능을 개선합니다.

SiteWise Edge를 사용하면 외부 오픈 소스 도구를 사용하여 로컬 데이터 처리 파이프라인을 생성할 수 있습니다. [Node-RED®](#)를 사용하여 [InfluxDB®](#)로 시계열 데이터를 저장하고 [Grafana®](#) 대시보드를 통해 작업을 모니터링할 수 있습니다.

Node-RED는 데이터 흐름을 처리하고 변환하는 반면, InfluxDB는 시계열 데이터 스토리지를 제공합니다. Grafana는 실시간 운영 데이터를 표시합니다. SiteWise Edge와 함께 이러한 도구를 사용하여 로컬 환경과 간에 데이터를 동기화하여 즉각적인 로컬 인사이트와 장기 클라우드 기반 분석 기능을 AWS 클라우드에 제공합니다.

Note

Node-RED®, InfluxDB® 및 Grafana®는 SiteWise Edge의 공급업체 또는 공급업체가 아닙니다.



Note

이 가이드에서는 Amazon Managed [Grafana](https://docs.aws.amazon.com/grafana/latest/userguide/what-is-Aman-Managed-Service-Grafana.html) 서비스와 달리 SiteWise Edge용 Grafana의 오픈 소스 버전을 사용합니다. <https://docs.aws.amazon.com/grafana/latest/userguide/what-is-Aman-Managed-Service-Grafana.html>

배포 옵션

두 가지 방법 중 하나를 사용하여 이 솔루션을 배포할 수 있습니다. Microsoft Windows 수동 설정을 사용하면 구성 요소 구성 및 인프라와의 통합을 제어할 수 있습니다. 를 사용하면 Docker를 사용하여 컨테이너에 사전 구성된 구성 요소를 배포할 Linux 수 있습니다.

운영 요구 사항을 충족하는 방법을 선택합니다.

- [수동으로 오픈 소스 통합 설정\(Windows\)](#) - 사용자 지정 구성 또는 기존 인프라의 경우
- [Docker\(Linux\)와 오픈 소스 통합 설정](#) - 사전 구성된 구성 요소를 사용하여 빠르게 배포

풍력 발전소 예제 개요

이 가이드에서는 풍력 발전소 예제를 사용하여 풍력 발전소의 터빈에 대한 풍력을 모니터링하는 방법을 보여줍니다. 이 실제 시나리오는 로컬 및 클라우드 기반 가시성이 운영 효율성에 중요한 일반적인 산업 모니터링 요구 사항을 보여줍니다.

이 통합을 통해 다음을 수행할 수 있습니다.

- AWS IoT SiteWise Edge 게이트웨이를 사용하여 산업 장비에서 데이터 수집
- Node-RED, InfluxDB 및 Grafana를 사용하여 로컬에서 데이터 처리
- InfluxDB를 사용하여 로컬에 데이터 저장
- Grafana 대시보드를 사용하여 실시간으로 데이터 모니터링

이 가이드에서는 풍력 발전소의 예를 사용합니다. Node-RED를 사용하여 풍속 데이터를 생성하는 터빈을 시뮬레이션합니다. Node-RED는 데이터 페이로드를 번역하고, SiteWise Edge MQTT 브로커에 데이터를 게시하고, 브로커로부터 데이터를 수신하도록 구독하고, InfluxDB에 로컬로 데이터를 저장합니다. 이 접근 방식을 사용하면 모든 운영 데이터를 로컬에서 즉시 액세스할 수 있고 클라우드에서 추가 분석을 수행할 수 있습니다. 이 패턴을 구현하면에서 고급 분석을 수행하는 기능을 유지하면서 네트워크 중단에 대한 복원력을 얻을 수 있습니다 AWS 클라우드. Grafana는 로컬 모니터링을 위해 InfluxDB에 연결하여 운영자에게 클라우드 종속성 없이 지표에 대한 실시간 가시성을 제공합니다. SiteWise Edge MQTT 지원 게이트웨이는 동일한 MQTT 브로커에 연결하여 로 데이터를 전송하여 엣지 작업과 클라우드 기반 서비스 간에 브리지를 AWS IoT SiteWise 생성합니다.

자체 데이터 및 구성을 사용하여 제조 장비, 유틸리티 인프라 또는 기타 산업 자산을 모니터링하는지 여부에 관계없이 특정 산업 요구 사항에 맞는 유사한 워크플로를 생성할 수 있습니다.

오픈 소스 통합 요구 사항

SiteWise Edge와 오픈 소스 통합을 구현하기 전에 환경이 필요한 요구 사항을 충족하는지 확인합니다.

- 하드웨어 요구 사항 - 게이트웨이 하드웨어는 SiteWise Edge 게이트웨이에 대한 요구 사항을 충족해야 합니다. 자세한 내용은 [AWS IoT SiteWise 엣지 자체 호스팅 게이트웨이 요구 사항](#) MQTT 지원, V3 게이트웨이 및 섹션을 참조하세요 [AWS IoT SiteWise Edge 애플리케이션에 대한 요구 사항](#).

Important

추가 오픈 소스 구성 요소를 배포할 때 하드웨어가 [InfluxDB](#), [Node-RED](#) 및 [Grafana](#)에 대한 요구 사항을 충족하는지 확인합니다.

- 네트워크 구성은 구성 요소 간의 로컬 통신과 SiteWise Edge의 클라우드 연결을 모두 지원해야 합니다.
- 모든 서비스는 동일한 호스트에서 실행되어야 합니다.

보안 고려 사항

특히 로컬이 아닌 네트워크에서 인터페이스에 액세스할 때 구성 요소 간의 모든 통신을 암호화하는 것이 좋습니다. 각 구성 요소에 대해 적절한 액세스 제어를 구현하고 AWS IoT SiteWise Edge 게이트웨이 구성 및 AWS 계정 보안 모범 AWS 사례를 따릅니다.

개발 환경

이 가이드는 게이트웨이 호스트에서 로컬로 실행되고 액세스되는 Node-RED, InfluxDB 및 Grafana를 보여줍니다. 외부 액세스가 필요한 프로덕션 배포의 경우 TLS 암호화, 인증 및 권한 부여를 포함한 보안 조치를 구현합니다. 각 애플리케이션의 보안 모범 사례를 따릅니다.

타사 소프트웨어

이 솔루션은 InfluxDB AWS, Node-RED, Grafana 및 `node-red-contrib-influxdb` 플러그인을 포함하여에서 유지 관리하지 않는 타사 소프트웨어를 사용합니다. 배포하기 전에 이러한 구성 요소가 조직의 보안 요구 사항, 규정 준수 표준 및 거버넌스 정책을 준수하는지 확인합니다.

Important

이 가이드는에서 소유하거나 유지 관리하지 않는 타사 소프트웨어를 참조하고 사용합니다. AWS. 구현하기 전에 모든 구성 요소가 보안, 규정 준수 및 거버넌스 요구 사항을 충족하는지 확인합니다. 모든 소프트웨어를 최신 보안 패치로 업데이트하고 엣지 배포 보안을 위한 모범 사례를 따르세요.

InfluxDB, Node-RED, Grafana는 SiteWise Edge의 공급업체 또는 공급업체가 아닙니다.

기타 고려 사항

SiteWise Edge와의 오픈 소스 통합을 구현할 때 이러한 추가 요소를 고려하세요.

- 모든 서비스, 도구 및 구성 요소의 최신 버전을 사용합니다.
- 클라우드 전송 전에 로컬에서 데이터를 필터링하고 집계하여 AWS IoT SiteWise 데이터 수집 비용을 줄입니다. InfluxDB에서 적절한 데이터 보존 기간을 구성하고 게이트웨이 하드웨어의 크기를 적절하게 조정합니다. 자세한 내용은 [AWS IoT SiteWise 요금](#)을 참조하십시오.

- 모든 데이터에 대해 정기적인 백업 절차를 구현합니다.
- 게이트웨이의 리소스 사용량을 모니터링하고 각 구성 요소에 대해 적절한 리소스 제한을 구성합니다. InfluxDB에서 데이터 보존 정책을 구현하여 디스크 사용량을 관리합니다.

수동으로 오픈 소스 통합 설정(Windows)

이 가이드를 사용하여 Grafana® 및 Node-RED®와 연결되는 풍속 데이터에 대한 시계열 버킷을 수동으로 생성합니다.

에서 Node-RED, InfluxDB® 및 Grafana를 수동으로 설치하고 구성Microsoft Windows하여 배포 구성을 제어합니다. InfluxDB를 사용하여 디바이스의 시계열 데이터를 저장하고 관리할 수 있습니다.

수동 설정 사전 조건

시작하기 전에 다음 요구 사항을 완료합니다.

Note

동일한 호스트에서 모든 서비스(SiteWise Edge, InfluxDB, Node-RED 및 Grafana)를 실행합니다.

- MQTT 지원 V3 게이트웨이를 설치합니다. 자세한 내용은 [MQTT 지원, AWS IoT SiteWise Edge용 V3 게이트웨이](#) 단원을 참조하십시오.
- 로컬에서 이러한 서비스를 설치하고 실행합니다.
 - InfluxDB OSS v2. 설치 단계는 [InfluxDB 설치](#)를 참조하세요.
 - 노드-빨간색. 설치 단계는 [로컬에 노드-RED 설치](#)를 참조하세요.
 - Grafana. 설치 단계는 [Grafana 설치](#)를 참조하세요.

InfluxDB를 사용하여 로컬 스토리지 설정

InfluxDB®를 사용하면 디바이스의 시계열 데이터를 로컬에 저장할 수 있습니다. 로컬 스토리지 기능의 목적은 네트워크 중단 시 운영 가시성을 유지하고 시간이 중요한 애플리케이션의 지연 시간을 줄이는 것입니다. 데이터를 클라우드로 선택적으로 전달할 수 있는 옵션을 유지하면서 엣지에서 분석 및 시각화를 수행할 수 있습니다.

이 단원에서는 터빈 바람 속도 데이터를 위한 시계열 버킷을 생성하고 Grafana® 및 Node-RED® 연결을 위한 API 토큰을 생성합니다. InfluxDB 버킷은 기존 시스템의 데이터베이스와 유사한 시계열 데이터

의 전용 스토리지 컨테이너 역할을 합니다. API 토큰을 사용하면 데이터에 프로그래밍 방식으로 안전하게 액세스할 수 있습니다.

InfluxDB를 설정하려면

1. 사전 조건 단계를 완료하고 모든 도구가 동일한 호스트에서 실행되고 있는지 확인한 후 웹 브라우저를 열고 <http://127.0.0.1:8086> 이동합니다.
2. (선택 사항) 보안 강화를 위해 TLS 암호화를 활성화합니다. 자세한 내용은의 [TLS 암호화 활성화](#)를 참조하세요InfluxData Documentation.
3. Node-RED의 데이터를 저장할 시계열 InfluxDB 버킷을 생성합니다. 버킷은 풍력 발전소 데이터의 전용 컨테이너 역할을 하므로이 데이터 세트와 관련된 보존 정책을 구성하고 관리할 수 있습니다. 자세한 내용은의 [버킷 관리를 참조하세요](#)InfluxData Documentation.
4. (선택 사항) 엣지 로케이션의 데이터 보존 기간을 구성합니다. 적절한 보존 기간을 설정하면 로컬 작업에 더 이상 필요하지 않은 이전 데이터를 자동으로 제거하여 스토리지 리소스를 효율적으로 관리하는 데 도움이 됩니다.

데이터 보존에 대한 자세한 내용은 [의 InfluxDB의 데이터 보존](#)을 참조하세요InfluxData Documentation.

5. 버킷에 대한 API 토큰을 생성합니다. 이 토큰을 사용하면 InfluxDB와 Node-RED 및 Grafana와 같은 다른 구성 요소 간의 보안 통신이 가능합니다. 이렇게 하면 승인된 서비스만 데이터 스토어에서 읽거나 쓸 수 있습니다. 자세한 내용은의 [토큰 생성](#)을 참조하세요InfluxData Documentation.

이 단계를 완료한 후 InfluxDB 인스턴스에 시계열 데이터를 저장하여 엣지 환경에서 로컬 데이터 지속성 및 분석을 위한 기반을 제공할 수 있습니다.

AWS IoT SiteWise 데이터 통합을 위한 노드-RED 흐름 구성

Node-RED®를 사용하면 두 흐름을 구현하여 디바이스와 간에 데이터를 관리할 수 있습니다 AWS IoT SiteWise. 이러한 흐름은 함께 작동하여 로컬 및 클라우드 데이터 흐름을 모두 처리하는 포괄적인 데이터 관리 솔루션을 생성합니다.

- 데이터 게시 흐름 - 클라우드에 게시합니다. 데이터 게시 흐름은 데이터를 로 전송합니다 AWS IoT SiteWise. 이 흐름은 센서 데이터를 생성하고, AWS IoT SiteWise 형식으로 변환하고, SiteWise Edge MQTT 브로커에 게시하여 터빈 디바이스를 시뮬레이션합니다. 이를 통해 AWS IoT SiteWise의 클라우드 기능을 활용하여 스토리지, 분석 및 다른 AWS 서비스와 통합할 수 있습니다.

자세한 내용은 [데이터 게시 흐름 구성](#) 단원을 참조하십시오.

- 데이터 보존 흐름 - 엣지에 데이터를 저장합니다. 데이터 보존 흐름은 SiteWise Edge MQTT 브로커를 구독하여 데이터를 수신하고 InfluxDB® 형식으로 변환한 다음 모니터링을 위해 로컬에 저장합니다. 이 로컬 스토리지는 운영 데이터에 대한 즉각적인 액세스를 제공하고, 시간이 중요한 애플리케이션의 지연 시간을 줄이고, 네트워크 중단 시 연속성을 보장합니다.

자세한 내용은 [데이터 보존 흐름 구성](#) 단원을 참조하십시오.

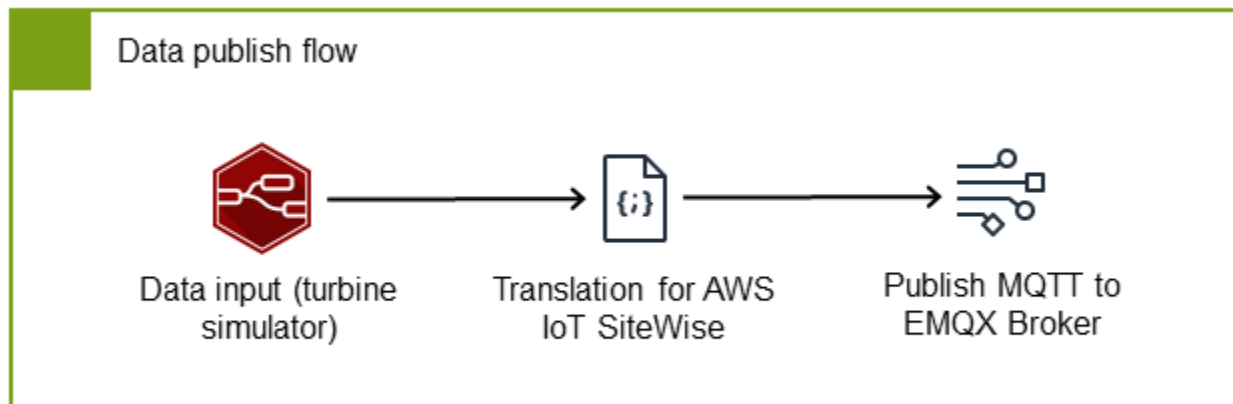
이 두 흐름은 함께 작동하여 즉각적인 액세스를 위해 데이터가 로컬로 전송 AWS IoT SiteWise 되고 저장되도록 합니다.

Node-RED 콘솔에 액세스하려면 <http://127.0.0.1:1880> 이동합니다. TLS 활성화에 대한 자세한 내용은 [TLS 암호화 활성화](#)를 참조하세요.

데이터 게시 흐름 구성

데이터 게시 흐름은 세 개의 노드를 사용하여 산업 데이터를 클라우드로 전송하는 파이프라인을 생성합니다. 이 흐름은 클라우드 기반 분석, 장기 스토리지 및 다른 AWS 서비스와의 통합을 활성화하는 데 필수적입니다. 먼저 시뮬레이션된 디바이스 데이터가 SiteWise Edge MQTT 브로커로 전송됩니다. 게이트웨이는 강력한 분석 및 시각화 기능을 활용할 수 있는 AWS IoT SiteWise 클라우드로의 전송을 허용하는 브로커에서 데이터를 선택합니다.

- 데이터 입력 - 산업 장비 또는 시뮬레이터에서 디바이스 데이터를 수신합니다.
- 용 데이터 변환기 AWS IoT SiteWise - SiteWise Edge 게이트웨이와의 호환성을 보장하기 위해 데이터를 AWS IoT SiteWise 형식으로 변환합니다.
- MQTT 게시자 - SiteWise Edge MQTT 브로커에 데이터를 게시하여 로컬 소비자와 클라우드 소비자가 모두 사용할 수 있도록 합니다.



데이터 입력 노드 구성

이 예제에서 데이터 입력 노드는 풍속 데이터를 생성하는 시뮬레이션된 풍력 터빈 디바이스를 사용합니다. 이 노드는 시뮬레이션된 소스(예제 참조)에서 왔든 프로덕션 환경의 실제 산업 장비에서 왔든 관계없이 산업 데이터의 진입점 역할을 합니다.

데이터 페이로드에 사용자 지정 JSON 형식을 사용하여 로컬 처리 도구와 AWS IoT SiteWise 클라우드 서비스 모두에서 효율적으로 작동하는 표준화된 구조를 제공합니다. 이 형식에는 실제 측정 값과 함께 타임스탬프 및 품질 지표와 같은 필수 메타데이터가 포함되어 있으므로 파이프라인 전체에서 포괄적인 데이터 관리 및 품질 추적이 가능합니다. 주입 노드를 가져와 타임스탬프, 품질 지표 및 값이 포함된 이 표준화된 JSON 형식으로 시뮬레이션된 데이터를 수신합니다.

Node-RED 주입 노드에 대한 자세한 내용은 Node-RED 설명서의 [주입](#) 섹션을 참조하세요.

터빈 시뮬레이터는 이 표준화된 JSON 형식으로 초당 풍속 데이터를 생성합니다.

Example : 터빈 데이터 페이로드

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

이 형식은 다음과 같은 몇 가지 이점을 제공합니다.

- name 필드는 특정 속성 또는 측정값을 식별하므로 동일한 디바이스에서 여러 데이터 포인트를 추적할 수 있습니다.
- 나노초 timestamp 단위로는 정확한 기록 분석을 위한 정확한 시간 추적을 보장합니다.
- quality 표시기는 신뢰성을 기반으로 데이터를 필터링하고 관리하는 데 도움이 됩니다.
- 유연한 value 필드는 다양한 센서 출력을 수용할 수 있도록 다양한 데이터 유형을 지원합니다.

Example : 터빈 시뮬레이터의 주입 노드

```
[
  {
    "id": "string",
    "type": "inject",
```

```

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

```

데이터 번역을 위한 노드 구성

SiteWise Edge 게이트웨이는 AWS IoT SiteWise 클라우드와의 호환성을 보장하기 위해 특정 형식의 데이터가 필요합니다. 변환기 노드는 입력 데이터를 필요한 AWS IoT SiteWise 페이로드 형식으로 변

환하는 중요한 구성 요소입니다. 이 번역 단계를 통해 클라우드 환경에서 산업 데이터를 올바르게 처리, 저장 및 나중에 분석할 수 있습니다 AWS IoT SiteWise .

이 단계에서 데이터 형식을 표준화하면 엣지 디바이스와 자산 모델링, 분석 및 시각화 기능을 사용할 수 있는 클라우드 서비스 간의 통합이 가능합니다. 다음 구조를 사용합니다.

Example : SiteWise Edge 데이터 구문 분석을 위한 페이로드 구조

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

Note

를 MQTT 주제 계층 구조(예: /Renton/WindFarm/Turbine/WindSpeed)propertyAlias와 일치시킵니다. 이렇게 하면 데이터가의 올바른 자산 속성과 올바르게 연결됩니다 AWS IoT SiteWise. 자세한 내용은의 "데이터 스트림 별칭" 개념을 참조하십시오 [AWS IoT SiteWise 개념](#).

1. AWS IoT SiteWise 페이로드 변환을 위해 예제 함수 노드를 가져옵니다. 이 함수는 표준화된 입력 형식에서 AWS IoT SiteWise 호환 형식으로의 변환을 처리하여 적절한 타임스탬프 형식 지정, 품질 지표 및 값 입력을 보장합니다.

```
[
  {
    "id": "string",
```

```

    "type": "function",
    "z": "string",
    "name": "Translate to SiteWise payload",
    "func": "let input = msg.payload;\nlet output = {};\n\noutput[\"propertyAlias\"] = input.name;\n\nlet propertyVal = {}\n\nlet timeInSeconds = Math.floor(input.timestamp / 1000);\nlet offsetInNanos = (input.timestamp % 1000) * 1000000;\n\npropertyVal[\"timestamp\"] = {\n  \"timeInSeconds\": timeInSeconds,\n  \"offsetInNanos\": offsetInNanos,\n};\n\npropertyVal[\"quality\"] = input.quality\n\nlet typeNameConverter = {\n  \"number\": (x) => Number.isInteger(x) ? \"integerValue\" : \"doubleValue\",\n  \"boolean\": (x) => \"booleanValue\",\n  \"string\": (x) => \"stringValue\",\n}\n\nlet typeName = typeNameConverter[typeof input.value]\n\npropertyVal[\"value\"] = {}\npropertyVal[\"value\"][typeName] = input.value;\n\noutput[\"propertyValues\"] = [propertyVal]\n\nreturn {\n  payload: JSON.stringify(output)\n};",
    "outputs": 1,
    "timeout": "",
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 530,
    "y": 200,
    "wires": [
      [
        "string"
      ]
    ]
  }
}
]

```

2. JavaScript 코드가 바람 속도 데이터를 올바르게 변환하는지 확인합니다. 함수는 몇 가지 중요한 작업을 수행합니다.
 - 입력에서 속성 이름을 추출하여 propertyAlias로 설정합니다.
 - 타임스탬프를 밀리초에서 필요한 초 및 나노초 형식으로 변환합니다.
 - 데이터 신뢰성 추적을 위한 품질 지표 보존
 - 값 유형을 자동으로 감지하고 요구 사항에 따라 형식을 지정합니다. AWS IoT SiteWise
3. 노드를 흐름에 연결하여 데이터 입력 노드와 MQTT 게시자 간에 연결합니다.

비즈니스 요구 사항에 맞는 함수 작성에 대한 지침은 Node-RED 설명서의 [함수 작성](#)을 참조하세요.

MQTT 게시자 구성

번역 후 SiteWise Edge MQTT 브로커에 데이터를 게시할 준비가 되었습니다.

SiteWise Edge MQTT 브로커로 데이터를 전송하도록 다음 설정으로 MQTT 게시자를 구성합니다.

MQTT 출력 노드를 가져오려면

1. 를 사용하여 MQTT Out 구성 노드를 가져옵니다 "type": "mqtt out". MQTT 출력 노드를 사용하면 브로커의 구성을 공유할 수 있습니다.
2. MQTT 브로커 연결 및 메시지 라우팅과 관련된 정보를 보려면 키-값 페어를 입력합니다.

예제 mqtt out 노드를 가져옵니다.

Example

```
[
  {
    "id": "string",
    "type": "mqtt out",
    "z": "string",
    "name": "Publish to MQTT broker",
    "topic": "/Renton/WindFarm/Turbine/WindSpeed",
    "qos": "1",
    "retain": "",
    "respTopic": "",
    "contentType": "",
    "userProps": "",
    "correl": "",
    "expiry": "",
    "broker": "string",
    "x": 830,
    "y": 200,
    "wires": []
  },
  {
    "id": "string",
    "type": "mqtt-broker",
    "name": "emqx",
    "broker": "127.0.0.1",
    "port": "1883",
    "clientid": "",
    "autoConnect": true,
```

```

    "usetls": false,
    "protocolVersion": "5",
    "keepalive": 15,
    "cleansession": true,
    "autoUnsubscribe": true,
    "birthTopic": "",
    "birthQos": "0",
    "birthPayload": "",
    "birthMsg": {},
    "closeTopic": "",
    "closePayload": "",
    "closeMsg": {},
    "willTopic": "",
    "willQos": "0",
    "willPayload": "",
    "willMsg": {},
    "userProps": "",
    "sessionExpiry": ""
  }
]

```

예제 MQTT 출력 노드는 다음 정보를 사용하여 MQTT 연결을 생성합니다.

- 서버: 127.0.0.1
- 포트: 1883
- 프로토콜: MQTT V5

그런 다음 MQTT 출력 노드는 다음 정보로 메시지 라우팅을 구성합니다.

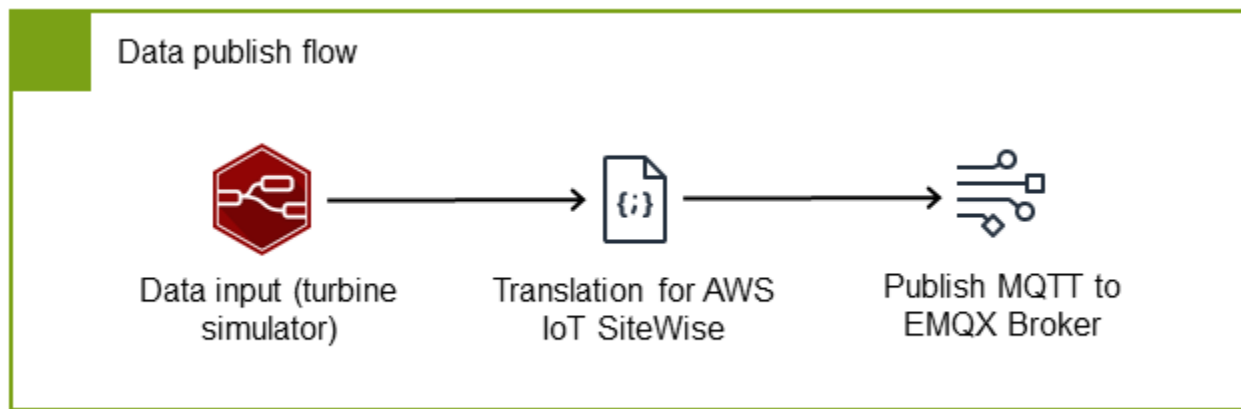
- 주제: /Renton/WindFarm/Turbine/WindSpeed
- QoS: 1

노드 배포 및 확인

세 개의 데이터 게시 흐름 노드를 구성한 후 다음 단계에 따라 흐름을 배포하고 데이터가 로 올바르게 전송되고 있는지 확인합니다. AWS IoT SiteWise

연결을 배포하고 확인하려면

1. 데이터 게시 흐름에 표시된 대로 노드 3개를 연결합니다.



2. 배포를 선택하여 모든 노드 연결 변경 사항을 적용합니다.
3. [AWS IoT SiteWise 콘솔](#)로 이동하여 데이터 스트림을 선택합니다.
4. 드롭다운 메뉴에서 별칭 접두사가 선택되어 있는지 확인합니다. 그런 다음 /Renton/WindFarm/Turbine/WindSpeed별칭을 검색합니다.

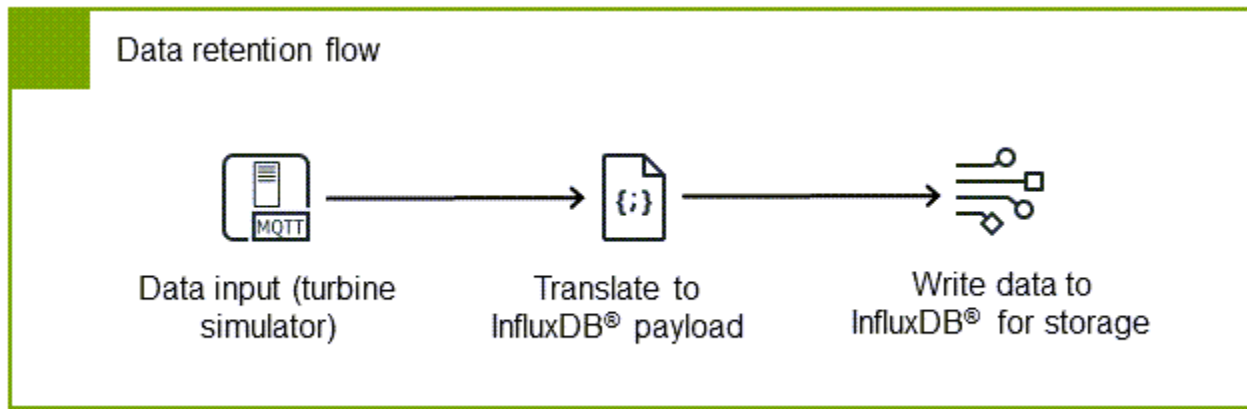
검색에 올바른 별칭이 표시되면 흐름을 배포하고 데이터 전송을 확인한 것입니다.

데이터 보존 흐름 구성

데이터 보존 흐름은 엣지에서 운영 가시성을 유지하는 데 사용할 수 있습니다. 이는 네트워크 중단 시 또는 데이터에 즉시 액세스해야 하는 경우에 유용합니다. 이 흐름은 MQTT 브로커를 구독하여 디바이스 데이터를 수신하고 InfluxDB® 형식으로 변환한 다음 로컬에 저장합니다. 이 흐름을 구현하면 운영자가 클라우드 종속성 없이 액세스할 수 있는 복원력이 뛰어난 로컬 데이터 스토어를 생성하여 엣지에서 실시간 모니터링 및 의사 결정을 수행할 수 있습니다.

흐름은 데이터를 올바르게 캡처하고 저장하기 위해 함께 작동하는 세 가지 주요 구성 요소로 구성됩니다.

- MQTT 구독 클라이언트 - 브로커로부터 데이터를 수신하여 모든 관련 산업 데이터를 캡처합니다.
- InfluxDB 변환기 - AWS IoT SiteWise 페이로드를 InfluxDB 형식으로 변환하여 효율적인 시계열 스토리지를 위한 데이터 준비
- InfluxDB 라이터 - 로컬 스토리지를 처리하여 로컬 애플리케이션의 데이터 지속성 및 가용성 보장



MQTT 구독 클라이언트 설정

- 아래 예제를 가져 AWS IoT SiteWise 와서에서 MQTT EMQX 브로커로부터 데이터를 수신하도록 Node-RED로 MQTT 구독 클라이언트를 구성합니다.

Example : 노드의 MQTT

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

```

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

```

이 구독을 통해 브로커에 게시된 모든 관련 데이터를 로컬 스토리지에 캡처하여 산업 운영에 대한 전체 레코드를 제공할 수 있습니다. 노드는 다음 구독 설정과 함께 [MQTT 게시자 구성](#) 섹션과 동일한 MQTT 연결 파라미터를 사용합니다.

- 주제 - /Renton/WindFarm/Turbine/WindSpeed
- QoS - 1

자세한 내용은 Node-RED 설명서의 [MQTT 브로커에 연결을](#) 참조하세요.

InfluxDB 변환기 구성

InfluxDB는 인덱싱에 [태그](#)를 사용하고 값에 [필드](#)를 사용하여 데이터를 구성합니다. 이 조직은 시계열 데이터의 쿼리 성능과 스토리지 효율성을 최적화합니다. JavaScript 코드가 포함된 예제 함수 노드를

가져와 AWS IoT SiteWise 페이로드를 InfluxDB 형식으로 변환합니다. 변환기는 속성을 두 그룹으로 분할합니다.

- 태그 - 효율적인 인덱싱을 위한 품질 및 이름 속성
- 필드 - 타임스탬프(에포크 이후 밀리초) 및 값

Example : InfluxDB 페이로드로 변환하는 함수 노드

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

추가 구성 옵션은 Node-RED GitHub 리포지토리의 [node-red-contrib-influxdb](#)를 참조하세요. GitHub

InfluxDB 라이터 설정

InfluxDB 라이터 노드는 데이터 보존 흐름의 마지막 구성 요소로, 산업 데이터를 로컬 InfluxDB 데이터베이스에 저장하는 역할을 합니다. 이 로컬 스토리지는 네트워크 중단 시 운영 가시성을 유지하고 시간이 중요한 애플리케이션의 데이터에 즉시 액세스할 수 있도록 하는 데 중요합니다.

1. 팔레트 관리 옵션을 통해 node-red-contrib-influxdb 패키지를 설치합니다. 이 패키지는 Node-RED를 InfluxDB와 연결하는 데 필요한 노드를 제공합니다.
2. 흐름에 InfluxDB 출력 노드를 추가합니다. 이 노드는 InfluxDB 데이터베이스에 대한 실제 데이터 쓰기를 처리합니다.
3. InfluxDB 인스턴스에 대한 보안 연결을 설정하도록 서버 속성을 구성합니다.
 - a. 버전을 2.0으로 설정 - 이전 버전과 다른 API를 사용하는 InfluxDB v2.x에 연결하도록 지정합니다.
 - b. URL을 로 설정 http://127.0.0.1:8086 - 로컬 InfluxDB 인스턴스를 가리킵니다.
 - c. InfluxDB 인증 토큰을 입력합니다. 이 보안 토큰은 데이터베이스에 대한 연결을 승인합니다. [InfluxDB를 사용하여 로컬 스토리지 설정](#) 절차 중에 토큰을 생성했습니다.
4. 스토리지 위치 파라미터를 지정하여 데이터가 저장되는 위치와 방법을 정의합니다.
 - a. InfluxDB 조직 이름 입력 - 조직은 버킷과 대시보드가 속한 사용자 그룹을 위한 워크스페이스입니다. 자세한 내용은 [조직 관리](#)를 참조하세요 InfluxData Documentation.
 - b. InfluxDB 버킷 지정(예: WindFarmData) - 버킷은 시계열 데이터의 컨테이너 역할을 하는 기존 시스템의 데이터베이스와 동일합니다.
 - c. InfluxDB 측정 설정(예: TurbineData) - 측정은 관계형 데이터베이스의 테이블과 유사하여 관련 데이터 포인트를 구성합니다.

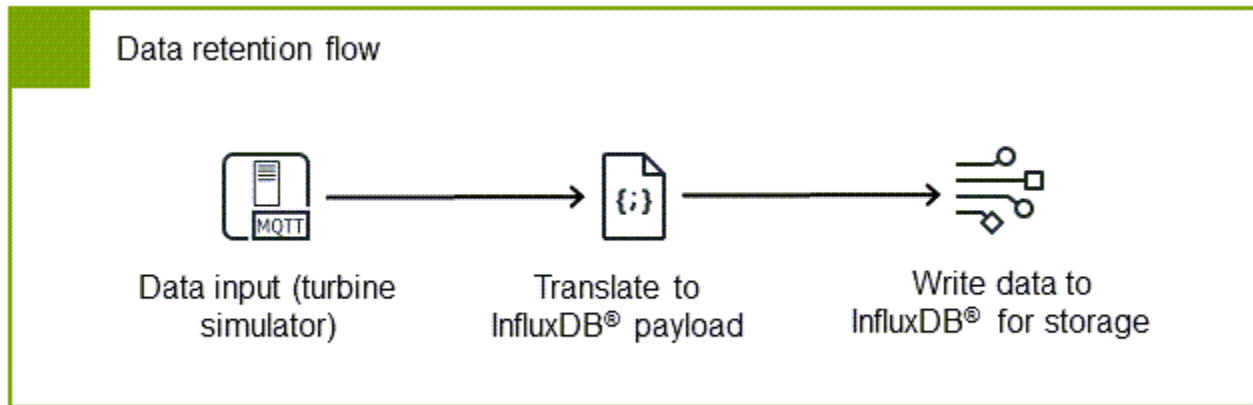
Note

InfluxDB 인스턴스의 왼쪽 사이드바에서 조직 이름을 찾습니다. 조직, 버킷 및 측정 개념은 InfluxDB의 데이터 조직 모델의 기본입니다. 자세한 내용은 [InfluxDB 설명서](#)를 참조하세요.

보존 흐름 배포 및 확인

데이터 보존 흐름의 모든 구성 요소를 구성한 후에는 시스템이 올바르게 작동하는지 배포하고 확인해야 합니다. 이 확인을 통해 산업 데이터가 즉각적인 액세스 및 분석을 위해 로컬에 올바르게 저장되고 있는지 확인할 수 있습니다.

1. 데이터 보존 흐름도에 표시된 대로 세 개의 노드를 연결합니다. 이렇게 하면 로컬 스토리지에 대한 데이터 구독에서 전체 파이프라인이 생성됩니다.



2. 배포를 선택하여 변경 사항을 적용하고 흐름을 활성화합니다. 그러면 데이터 수집 및 스토리지 프로세스가 시작됩니다.
3. InfluxDB Data Explorer를 사용하여 데이터를 쿼리하고 시각화합니다. 이 도구를 사용하면 데이터가 올바르게 저장되고 있는지 확인하고 시계열 데이터의 초기 시각화를 생성할 수 있습니다.

Data Explorer에서 시간 경과에 따라 풍속 측정값이 기록되는 것을 확인하여 데이터 생성부터 로컬 스토리지까지 전체 파이프라인이 올바르게 작동하는지 확인할 수 있습니다.

자세한 내용은 [의 Data Explorer에서 쿼리](#)를 참조하세요 InfluxData Documentation.

데이터 게시 흐름과 데이터 보존 흐름이 모두 배포되면 이제 즉각적인 액세스와 복원력을 위해 로컬 복사본을 유지하면서 AWS IoT SiteWise 클라우드로 데이터를 전송하는 완전한 시스템을 갖게 됩니다. 이 이중 경로 접근 방식을 사용하면 클라우드 기반 분석 및 스토리지의 이점을 얻는 동시에 엣지에서 운영 가시성을 유지할 수 있습니다.

SiteWise Edge용 Grafana 설정

Grafana®를 사용하면 산업 데이터에 대한 로컬 실시간 모니터링 대시보드를 생성할 수 있습니다. InfluxDB®에 저장된 데이터를 시각화하면 운영자에게 장비 성능, 프로세스 효율성 및 잠재적 문제에 대한 즉각적인 인사이트를 제공할 수 있습니다. 엣지에서의 이러한 가시성은 시간에 민감한 운영과 네트워크 중단 시 연속성 유지에 중요합니다.

데이터 소스 구성

Grafana를 InfluxDB 데이터베이스에 연결하면 산업 데이터를 위한 강력한 시각화 계층이 생성됩니다. 이 연결을 통해 운영자가 클라우드 종속성 없이 정보에 입각한 결정을 내리는 데 사용할 수 있는 실시간 모니터링 대시보드를 사용할 수 있습니다.

1. 브라우저에서 <http://127.0.0.1:3000> 이동하여 Grafana 인스턴스에 로컬로 액세스합니다. TLS를 활성화해야 하는 경우 Grafana Labs 설명서의 [안전한 웹 트래픽을 위한 Grafana HTTPS 설정을](#) 참조할 수 있습니다.
2. Node-RED가 데이터를 쓰는 InfluxDB 시계열 버킷을 가리키는 InfluxDB 데이터 소스를 추가합니다. 예를 들어 WindFarmData입니다. 이 연결은 저장된 데이터와 시각화 플랫폼 간의 링크를 설정합니다.
3. 자세한 지침은 Grafana Labs 설명서의 [InfluxDB 데이터 소스 구성을](#) 참조하세요.

SiteWise Edge 데이터에 대한 Grafana 대시보드 생성

대시보드 생성은 로컬 모니터링 솔루션을 구축하는 마지막 단계입니다. 대시보드는 산업 데이터를 시각적으로 표시하므로 추세, 이상 및 잠재적 문제를 한눈에 쉽게 식별할 수 있습니다.

- 가이드에 따라 대시보드를 생성합니다. 자세한 내용은 Grafana Labs 설명서의 [첫 번째 대시보드 빌드](#)를 참조하세요. 이 템플릿은 버킷의 이름이 WindFarmData 이고 측정값이 라고 가정합니다 TurbineData.

제공된 예제 대시보드 템플릿을 가져와서 빠른 시작 안내서를 사용하여 이전 섹션에서 Node-RED가 생성하는 데이터에 대한 시계열 플롯이 있는 대시보드를 빠르게 생성할 수도 있습니다. 이 템플릿은 특정 모니터링 요구 사항에 맞게 사용자 지정할 수 있는 시작점을 제공합니다.

```
{
  "__inputs": [
    {
      "name": "DS_WINDFARM-DEMO",
      "label": "windfarm-demo",
      "description": "",
      "type": "datasource",
      "pluginId": "influxdb",
      "pluginName": "InfluxDB"
    }
  ],
  "__elements": {},
  "__requires": [
```

```
{
  "type": "grafana",
  "id": "grafana",
  "name": "Grafana",
  "version": "11.6.0-pre"
},
{
  "type": "datasource",
  "id": "influxdb",
  "name": "InfluxDB",
  "version": "1.0.0"
},
{
  "type": "panel",
  "id": "timeseries",
  "name": "Time series",
  "version": ""
}
],
"annotations": {
  "list": [
    {
      "builtIn": 1,
      "datasource": {
        "type": "grafana",
        "uid": "-- Grafana --"
      },
      "enable": true,
      "hide": true,
      "iconColor": "rgba(0, 211, 255, 1)",
      "name": "Annotations & Alerts",
      "type": "dashboard"
    }
  ]
},
"editable": true,
"fiscalYearStartMonth": 0,
"graphTooltip": 0,
"id": null,
"links": [],
"panels": [
  {
    "datasource": {
      "type": "influxdb",
```

```
    "uid": "${DS_WINDFARM-DEMO}"
  },
  "fieldConfig": {
    "defaults": {
      "color": {
        "mode": "palette-classic"
      },
      "custom": {
        "axisBorderShow": false,
        "axisCenteredZero": false,
        "axisColorMode": "text",
        "axisLabel": "",
        "axisPlacement": "auto",
        "barAlignment": 0,
        "barWidthFactor": 0.6,
        "drawStyle": "line",
        "fillOpacity": 0,
        "gradientMode": "none",
        "hideFrom": {
          "legend": false,
          "tooltip": false,
          "viz": false
        },
        "insertNulls": false,
        "lineInterpolation": "linear",
        "lineWidth": 1,
        "pointSize": 5,
        "scaleDistribution": {
          "type": "linear"
        },
        "showPoints": "auto",
        "spanNulls": false,
        "stacking": {
          "group": "A",
          "mode": "none"
        },
        "thresholdsStyle": {
          "mode": "off"
        }
      },
      "mappings": [],
      "thresholds": {
        "mode": "absolute",
        "steps": [
```



```

        {
            "color": "green"
        },
        {
            "color": "red",
            "value": 80
        }
    ]
}
},
"overrides": []
},
"gridPos": {
    "h": 8,
    "w": 12,
    "x": 0,
    "y": 0
},
"id": 1,
"options": {
    "legend": {
        "calcs": [],
        "displayMode": "list",
        "placement": "bottom",
        "showLegend": true
    },
    "tooltip": {
        "hideZeros": false,
        "mode": "single",
        "sort": "none"
    }
},
"pluginVersion": "11.6.0-pre",
"targets": [
    {
        "datasource": {
            "type": "influxdb",
            "uid": "${DS_WINDFARM-DEMO}"
        },
        "query": "from(bucket: \"WindFarmData\")\n  |> range(start:
v.timeRangeStart, stop: v.timeRangeStop)\n  |> filter(fn: (r) => r[\"_measurement\"] == \"TurbineData\")\n  |> filter(fn: (r) => r[\"_field\"] == \"value\")\n  |> filter(fn: (r) => r[\"name\"] == \"/Renton/WindFarm/Turbine/WindSpeed\")\n"
    }
]

```

```

|> filter(fn: (r) => r["quality"] == "GOOD")\n  |> aggregateWindow(every:
v.windowPeriod, fn: mean, createEmpty: false)\n  |> yield(name: \"mean\"),
    "refId": "A"
  }
],
"title": "Panel Title",
"type": "timeseries"
}
],
"schemaVersion": 41,
"tags": [],
"templating": {
  "list": []
},
"time": {
  "from": "now-6h",
  "to": "now"
},
"timepicker": {},
"timezone": "browser",
"title": "demo dashboard",
"uid": "fejc0t08o6d4wb",
"version": 1,
"weekStart": ""
}

```

Docker(Linux)와 오픈 소스 통합 설정

간소화된 배포 프로세스를 위해 Docker를 사용하여 Linux 환경에서 Node-RED®, InfluxDB® 및 Grafana®를 설정할 수 있습니다. 이 방법은 사전 구성된 컨테이너를 사용하므로 구성 요소를 신속하게 배포하고 더 쉽게 관리할 수 있습니다.

Docker 설정 사전 조건

시작하기 전에 다음이 있는지 확인합니다.

- MQTT 지원 V3 게이트웨이입니다. 자세한 내용은 [MQTT 지원, AWS IoT SiteWise Edge용 V3 게이트웨이](#) 단원을 참조하십시오.
- Docker Compose 플러그인입니다. 설치 단계는 Docker 설명서 [의 Docker Compose 플러그인 설치를](#) 참조하세요.

서비스 배포

이 배포는 동일한 호스트에서 SiteWise Edge, InfluxDB, Node-RED 및 Grafana를 실행합니다.

환경 설정

1. 루트 액세스 권한 획득:

```
sudo -i
```

2. .env 파일을 생성하거나 다음 환경 변수를 내보냅니다.

```
export INFLUXDB_PASSWORD=your-secure-influxdb-password
export INFLUXDB_TOKEN=your-secure-influxdb-token
export GRAFANA_PASSWORD=your-secure-grafana-password
```

Docker 네트워크 구성

- 이름을 사용하여 브리지 네트워크를 생성합니다 SiteWiseEdgeNodeRedDemoNetwork.

```
docker network create --driver=bridge SiteWiseEdgeNodeRedDemoNetwork
```

SiteWise Edge 배포 업데이트

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 메뉴의 관리 섹션에서 Greengrass 디바이스를 선택한 다음 코어 디바이스를 선택합니다.
3. SiteWise Edge Gateway에 연결된 코어 디바이스를 선택합니다.
4. 배포 탭을 선택한 다음 배포 ID 값을 선택합니다.
5. 작업을 선택한 다음 수정을 선택합니다.
6. 팝업 메시지를 읽은 다음 배포 수정을 선택합니다.
7. 2단계 - 구성 요소 선택에서 다음 구성 요소를 선택한 후 다음을 선택합니다.
 - `aws.greengrass.clientdevices.mqtt.EMQX`
 - `aws.iot.SiteWiseEdgePublisher`
8. 3단계 - 구성 요소 구성에서 `aws.greengrass.clientdevices.mqtt.EMQX` 구성 요소 값을 선택하고 다음 네트워크 구성을 추가합니다.

```
{
  "emqxConfig": {
    "authorization": {
      "no_match": "allow"
    },
    "listeners": {
      "tcp": {
        "default": {
          "enabled": true,
          "enable_authn": false
        }
      }
    },
    "authMode": "bypass",
    "dockerOptions": "-p 127.0.0.1:1883:1883 --network=SiteWiseEdgeNodeRedDemoNetwork",
    "requiresPrivilege": "true"
  }
}
```

9. 다음을 선택합니다.
10. 4단계 - 고급 설정 구성에서 다음을 선택합니다.
11. 배포 선택

서비스 시작

1. Docker Compose 파일을 사용하여 서비스를 시작합니다. `compose.yaml` 파일이 포함된 디렉터리에서 다음 명령을 실행합니다.

```
docker compose up -d
```

2. SSH 터널을 생성하여 서비스에 액세스합니다.

```
ssh -i path_to_your_ssh_key -L 1880:127.0.0.1:1880 -L 3000:127.0.0.1:3000 -L 8086:127.0.0.1:8086 username@gateway_ip_address
```

이 배포는에서 SiteWiseEdgeNodeRedDemoNetwork network다음 서비스를 생성합니다.

InfluxDB v2(포트 8086)

사전 구성된 조직(iot-sitewise-edge), WindFarmData InfluxDB 버킷 및 관리자 자격 증명 포함
Node-RED(포트 1880)

AWS IoT SiteWise 통합을 위해 InfluxDB 노드 및 사전 구성된 흐름 포함
Grafana(포트 3000)

관리자 사용자, InfluxDB 데이터 소스 및 모니터링 대시보드 포함

서비스에 액세스

배포 후 다음 URLs.

 Note

호스트 또는 게이트웨이 시스템에서 각 서비스에 액세스할 수 있습니다.

서비스 액세스 세부 정보

Service	URL	보안 인증 정보
Node-RED	http://127.0.0.1:1880	자격 증명 필요 없음
InfluxDB	http://127.0.0.1:8086	사용자 이름: admin 암호: \$INFLUXDB_PASSWORD
Grafana	http://127.0.0.1:3000	사용자 이름: admin 암호: \$GRAFANA_PASSWORD

배포 확인

배포에 성공하려면 다음 검사를 수행합니다.

1. Node-RED의 경우 미리 로드된 흐름이 두 개 있는지 확인합니다.

- 데이터 게시 흐름
 - 데이터 보존 흐름
2. 의 경우 AWS IoT SiteWise 콘솔 AWS IoT SiteWise에서 별칭이 인 데이터 스트림이 있는지 확인합니다/Renton/WindFarm/Turbine/WindSpeed.
 3. InfluxDB의 경우 Data Explorer를 사용하여 WindFarmData 버킷 내 TurbineData 측정에서 데이터 스토리지를 확인합니다.
 4. Grafana의 경우 대시보드를 보고 Node-RED에서 생성된 시계열 데이터의 표시를 확인합니다.

오픈 소스 통합을 위한 데이터 처리

각기 다른 모니터링 요구 사항을 충족하는 다양한 도구를 사용하여 다양한 단계에서 데이터를 처리할 수 있습니다(예: 변환 또는 집계).

Node-RED 노드로 데이터 처리

Node-RED® 내장 처리 노드를 사용하여 데이터를 실시간으로 변환합니다. Node-RED 콘솔을 통해 이러한 노드를 구성하여 데이터 파이프라인을 생성합니다.

데이터 변환 노드

다음 노드를 AWS IoT SiteWise사용하여의 변환과 유사한 개별 데이터 포인트를 변환합니다.

- 변경 노드 - 데이터에 대한 간단한 값 수정을 수행합니다.
- 함수 노드 - 복잡한 데이터 처리를 위한 사용자 지정 JavaScript 변환을 활성화합니다.

지표 계산 노드

다음 노드를 AWS IoT SiteWise사용하여의 지표와 유사한 단일 출력으로 여러 데이터 포인트를 결합합니다.

- 배치 노드 - 배치 처리를 위해 여러 메시지를 그룹화합니다.
- 조인 노드 - 여러 데이터 스트림을 단일 출력으로 결합합니다.
- 애그리게이터 노드 - 여러 데이터 포인트의 집계 지표를 계산합니다.

추가 노드 옵션은 [Node-RED 라이브러리](#)를 참조하세요.

InfluxDB 작업 생성

Node-RED는 빠른 설정으로 기본 데이터 처리에 뛰어나지만 흐름 기반 프로그래밍에서는 복잡한 지표 계산이 어려울 수 있습니다. InfluxDB® 작업은 고급 처리 요구 사항을 위해 예약된 Flux 스크립트를 통해 대안을 제공합니다.

InfluxDB 작업 사용 대상:

- 대규모 데이터 세트 전반의 통계 집계
- 여러 속성에 대한 수학 작업
- 여러 소스에서 파생된 측정

작업 기능

- 예약된 실행 - cron 표현식을 기반으로 작업 실행
- 배치 처리 - 시계열 데이터에 대한 작업 최적화
- 오류 복구 - 실패한 작업을 자동으로 재시도합니다.
- 모니터링 - 세부 로그를 통해 실행 추적

InfluxDB UI, API 또는 CLI를 통해 작업을 관리합니다. 자세한 내용은 [InfluxDB 태스크로 데이터 처리](#)를 참조하세요.

Grafana 변환 사용

InfluxDB에서 소스 데이터를 수정하지 않고 Grafana®에서 데이터 시각화를 변환합니다. Grafana 변환은 시각화 계층에만 적용됩니다.

- Visual Builder - 코드를 작성하지 않고 변환 생성
- 라이브 미리 보기 - 변환 결과를 실시간으로 보기
- 다중 소스 - 여러 데이터베이스 소스의 데이터 처리
- 스토리지 효율성 - 중간 결과를 저장하지 않고 시각화 시 데이터 변환

자세한 내용은 [데이터 변환](#)을 참조하세요.

오픈 소스 통합 문제 해결

SiteWise Edge 게이트웨이의 오픈 소스 통합과 관련된 문제 해결 주제에 대한 자세한 내용은 섹션을 참조하세요 [Edge에서 오픈 소스 통합 문제 해결](#).

클래식 스트림, AWS IoT SiteWise Edge용 V2 게이트웨이

Classic 스트림, AWS IoT SiteWise Edge용 V2 게이트웨이의 기능과 제한 사항을 이해합니다.

클래식 스트림인 V2 게이트웨이는 MQTT 지원 V3 게이트웨이를 도입하기 전에 이전 AWS IoT SiteWise 배포에서 익숙한 기존 기능을 유지합니다. 이러한 SiteWise Edge 게이트웨이는 클래식 스트림, V2 게이트웨이로 간주됩니다. 이전 버전과의 호환성을 유지하고 데이터 처리 팩과 함께 작동합니다. 클래식 스트림인 V2 게이트웨이는 기존 설정에 안정적인 성능을 제공하지만 최신 게이트웨이 옵션에 비해 제한이 있습니다. 특히 이 게이트웨이 유형은 MQTT 지원 V3 게이트웨이 대상에서 사용할 수 있는 고급 기능과 완전히 호환되지 않습니다. MQTT 메시징 프로토콜을 사용하려면 새 MQTT 지원 V3 게이트웨이를 생성할 수 있습니다. 자세한 내용은 [MQTT 지원, AWS IoT SiteWise Edge용 V3 게이트웨이](#) 단원을 참조하십시오.

주제

- [팩을 사용하여 SiteWise Edge에서 데이터 수집 및 처리](#)
- [AWS IoT SiteWise 게시자 구성 요소 구성](#)
- [대상 및 AWS IoT Greengrass 스트림 관리자](#)
- [Edge에서 AWS IoT SiteWise 엣지 기능 구성](#)
- [AWS IoT SiteWise 모델 및 자산에 대한 엣지 데이터 처리 구성](#)

팩을 사용하여 SiteWise Edge에서 데이터 수집 및 처리

AWS IoT SiteWise 엣지 게이트웨이는 다양한 팩을 사용하여 데이터를 수집하고 처리하는 방법을 결정합니다.

현재 사용 가능한 팩은 다음과 같습니다.

- 데이터 수집 팩 - 이 팩을 사용하여 산업 데이터를 수집하고 AWS 클라우드 대상으로 라우팅합니다. 기본적으로 이 팩은 SiteWise Edge 게이트웨이에서 자동으로 활성화됩니다.
- 데이터 처리 팩 - 이 팩을 사용하면 엣지로 구성된 자산 모델 및 자산과 SiteWise Edge 게이트웨이 통신을 수행할 수 있습니다. 엣지 구성을 사용하여 현장에서 계산하고 처리할 자산 데이터를 제어할 수 있습니다. 그런 다음 AWS IoT SiteWise 또는 다른 AWS 서비스로 데이터를 전송할 수 있습니다. 데이터 처리 팩에 대한 자세한 내용은 [the section called “엣지 데이터 처리 구성”](#)을 참조하세요.

팩 업그레이드하기

 Important

데이터 처리 팩 버전을 2.0.x 이전(및 포함)에서 버전 2.1.x로 업그레이드하면 로컬에 저장된 측정값의 데이터가 손실됩니다.

SiteWise Edge 게이트웨이는 다양한 팩을 사용하여 데이터 수집 및 처리 방법을 결정합니다. AWS IoT SiteWise 콘솔을 사용하여 팩을 업그레이드할 수 있습니다.

팩을 업그레이드하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 게이트웨이 목록에서 업그레이드하려는 팩이 있는 SiteWise Edge 게이트웨이를 선택합니다.
4. 게이트웨이 구성 섹션에서 사용 가능한 소프트웨어 업데이트를 선택합니다.
5. 소프트웨어 버전 편집 페이지에서 업데이트를 선택합니다.

 Note

활성화된 팩만 업그레이드할 수 있습니다. 이 SiteWise Edge 게이트웨이에 대해 활성화된 팩 목록을 찾으려면 개요를 선택한 다음 엣지 기능 섹션을 참조하세요.

6. 소프트웨어 버전 편집 페이지의 게이트웨이 구성 요소 업데이트 섹션에서 다음을 수행합니다.
 - OPC UA 수집기를 업데이트하려면 버전을 선택한 다음 배포를 선택합니다.
 - 게시자를 업데이트하려면 버전을 선택한 다음 배포를 선택합니다.
 - 데이터 처리 팩을 업데이트하려면 버전을 선택한 다음 배포를 선택합니다.
7. 새 버전 배포가 완료되면 완료를 선택합니다.

팩을 업그레이드하는 데 문제가 있는 경우 [SiteWise Edge 게이트웨이에 팩을 배포할 수 없음](#)을 참조하세요.

AWS IoT SiteWise 게시자 구성 요소 구성

AWS IoT SiteWise Edge 게이트웨이를 생성하고 소프트웨어를 설치한 후 SiteWise Edge 게이트웨이가 AWS 클라우드로 데이터를 내보낼 수 있도록 게시자 구성 요소를 설정할 수 있습니다. 게시자

구성 요소를 사용하여 추가 기능을 활성화하거나 기본 설정을 구성합니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [AWS IoT SiteWise 게시자](#)를 참조하세요.

Note

게시자 구성은 사용 중인 게이트웨이 유형에 따라 다릅니다. 클래식 스트림, V2 게이트웨이의 경우 `iotsitewise:publisher:2` 네임스페이스를 사용합니다. MQTT 지원 V3 게이트웨이의 경우 `iotsitewise:publisher:3` 네임스페이스를 사용합니다.

Console

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 게시자를 구성할 SiteWise Edge 게이트웨이를 선택합니다.
4. 게시자 구성 섹션에서 편집을 선택합니다.
5. 게시 순서에서 다음 중 하나를 선택합니다.
 - 가장 오래된 데이터를 먼저 게시 - SiteWise Edge 게이트웨이가 기본적으로 가장 오래된 데이터를 클라우드에 먼저 게시합니다.
 - 최신 데이터를 먼저 게시 - SiteWise Edge 게이트웨이가 최신 데이터를 클라우드에 먼저 게시합니다.
6. (선택 사항) SiteWise Edge 게이트웨이가 데이터를 압축하지 않도록 하려면 데이터를 업로드할 때 압축 활성화를 선택 취소합니다.
7. (선택 사항) 오래된 데이터를 게시하지 않으려면 만료된 데이터 제외를 선택하고 다음을 수행합니다.
 - 마감 기간의 경우 값을 입력하고 단위를 선택합니다. 마감 기간은 5분에서 7일 사이여야 합니다. 예를 들어 마감 기간이 3일인 경우 3일 이전의 데이터는 클라우드에 게시되지 않습니다.
8. (선택 사항) 로컬 디바이스에서 데이터가 처리되는 방식에 대한 사용자 지정 설정을 지정하려면 로컬 스토리지 설정을 선택하고 다음을 수행합니다.
 - a. 보존 기간에 숫자를 입력하고 단위를 선택합니다. 보존 기간은 1분에서 30일 사이여야 하며, 교체 기간보다 길거나 같아야 합니다. 예를 들어 보존 기간이 14일인 경우 SiteWise Edge 게이트웨이는 14일 동안 저장된 후 지정된 마감 기간보다 오래된 모든 엣지 데이터를 삭제합니다.

- b. 교체 기간의 경우 숫자를 입력하고 단위를 선택합니다. 교체 기간은 1분보다 크며 보존 기간과 같거나 작아야 합니다. 예를 들어 교체 기간이 2일인 경우 SiteWise Edge 게이트웨이는 마감 기간보다 오래된 데이터를 일괄 처리하여 단일 파일에 저장합니다. 를 통한 자체 호스팅 게이트웨이의 경우 AWS IoT Greengrass V2 SiteWise Edge 게이트웨이는 2일에 한 번씩 다음 로컬 디렉터리로 데이터 배치를 전송합니다/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports.
- c. 스토리지 용량에 대해 1 이상의 숫자를 입력합니다. 스토리지 용량이 2GB인 경우 로컬에 2GB를 초과하는 데이터가 저장되면 SiteWise Edge 게이트웨이에서 데이터 삭제를 시작합니다.

9. 저장을 선택합니다.

AWS CLI

[UpdateGatewayCapabilityConfiguration](#) API를 사용하여 게시자를 구성합니다.

capabilityNamespace 파라미터를 iotsitewise:publisher:2로 설정합니다.

Example : Classic Stream, V2 게이트웨이에 대한 게시자 구성

게시자 네임스페이스: 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
        }
    }
}
]
}
}

```

게시자는 사용자 지정할 수 있는 다음과 같은 구성 파라미터를 제공합니다.

SiteWisePublisherConfiguration

publishingOrder

데이터가 클라우드에 게시되는 순서. 이 파라미터는 다음 값 중 하나일 수 있습니다.

- **TIME_ORDER**(가장 오래된 데이터부터 게시) - 기본적으로 가장 오래된 데이터가 클라우드에 먼저 게시됩니다.
- **RECENT_DATA**(최신 데이터를 먼저 게시) — 최신 데이터가 클라우드에 먼저 게시됩니다.

enableCompression

게시하기 전에 데이터를 압축 `true` 하려면 이 값을 `로` 설정합니다. 데이터 압축은 대역폭 사용량을 줄일 수 있습니다.

dropPolicy

(선택 사항) 클라우드에 게시되는 데이터를 제어하는 정책입니다.

cutoffAge

일, 시간 및 분으로 지정된 게시할 데이터의 최대 수명입니다. 예: 7d 또는 1d7h16m. 지정된 것보다 오래된 데이터는 로 전송되지 않습니다 AWS IoT SiteWise.

마감 기간 이전의 데이터는 클라우드에 게시되지 않습니다. 마감 기간은 5분에서 7일 사이여야 합니다.

마감 연령을 지정할 때 m, h, d를 사용할 수 있습니다. m은 분, h는 시간, d는 일을 나타냅니다.

exportPolicy

(선택 사항) 엣지의 데이터 스토리지를 관리하는 정책입니다. 이 정책은 마감 기간 이전의 데이터에 적용됩니다.

retentionPeriod

SiteWise Edge 게이트웨이는 지정된 보존 기간 동안 저장된 후 로컬 스토리지에서 마감 기간보다 이른 엣지의 모든 데이터를 삭제합니다. 보존 기간은 1분에서 30일 사이여야 하며, 교체 기간보다 길거나 같아야 합니다.

보존 기간을 지정할 때 m, h, d를 사용할 수 있습니다. m은 분, h는 시간, d는 일을 나타냅니다.

rotationPeriod

마감 기간보다 이른 데이터를 일괄 처리하여 단일 파일에 저장하는 시간 간격입니다. SiteWise Edge 게이트웨이는 각 교체 기간이 끝날 때 데이터 배치 하나를 로컬 디렉터리 /greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports로 전송합니다. 교체 기간은 1분보다 크며 보존 기간과 같거나 작아야 합니다.

교체 기간을 지정할 때 m, h, d를 사용할 수 있습니다. m은 분, h는 시간, d는 일을 나타냅니다.

exportSizeLimitGB

로컬에 저장된 최대 허용 데이터 크기(GB)입니다. 이 할당량이 위반되면 SiteWise Edge 게이트웨이는 로컬에 저장된 데이터 크기가 할당량 이하가 될 때까지 가장 이른 데이터를 삭제하기 시작합니다. 이 파라미터의 값은 1보다 크거나 같아야 합니다.

SiteWiseS3PublisherConfiguration

accessRoleArn

게시하려는 Amazon S3 버킷을 관리할 수 있는 AWS IoT SiteWise 권한을 부여하는 액세스 역할입니다.

streamToS3ConfigMapping

스트림을 Amazon S3 구성에 매핑하는 구성 배열입니다.

streamName

Amazon S3 구성에서 읽고 게시할 스트림입니다.

`targetBucketArn`

게시할 버킷 ARN입니다.

`publishPolicy`

`publishFrequency`

SiteWise Edge 게이트웨이가 Amazon S3 버킷에 게시하는 빈도입니다.

`localSizeLimitGB`

로컬 디스크에 기록된 파일의 최대 크기입니다. 이 임계값을 위반하면 게시자는 버퍼링된 모든 데이터를 대상에 게시합니다.

`siteWiseImportPolicy`

`enableSiteWiseStorageImport`

Amazon S3 버킷에서 AWS IoT SiteWise 스토리지로 데이터를 가져오려면 `true`로 설정합니다.

`enableDeleteAfterImport`

AWS IoT SiteWise 스토리지에 수집한 후 Amazon S3 버킷의 파일을 삭제하려면 `true`로 설정합니다.

대상 및 AWS IoT Greengrass 스트림 관리자

AWS IoT Greengrass 스트림 관리자를 사용하면의 채널 AWS IoT Analytics, Amazon Kinesis Data Streams의 스트림,의 자산 속성 AWS IoT SiteWise또는 Amazon Simple Storage Service(Amazon S3)의 객체와 같은 AWS 클라우드 대상으로 데이터를 전송할 수 있습니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [AWS IoT Greengrass Core에서의 데이터 스트림 관리](#)를 참조하세요.

Example : 데이터 스트림 메시지 구조

다음 예제에서는 스트림 관리자가 전송하는 필수 데이터 AWS IoT Greengrass 스트림 메시지 구조를 보여줍니다.

```
{
  "assetId": "string",
  "propertyAlias": "string",
  "propertyId": "string",
```

```

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

```

Note

데이터 스트림 메시지의 구조에는 (assetId 및 propertyId) 또는 propertyAlias가 포함되어야 합니다.

assetId

(선택 사항) 업데이트할 자산의 ID입니다.

propertyAlias

(선택 사항) OPC UA 서버 데이터 스트림 경로와 같이 속성을 식별하는 별칭입니다. 예시:

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

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [데이터 스트림 관리를 참조하세요](#).

propertyId

(선택 사항) 이 항목의 자산 속성 ID입니다.

propertyValues

(필수) 업로드할 속성 값 목록입니다. 최대 10개의 propertyValues 배열 요소를 지정할 수 있습니다.

quality

(선택 사항) 자산 속성 값의 품질입니다.

timestamp

(필수) 자산 속성 값의 타임스탬프입니다.

offsetInNanos

(선택 사항) timeInSeconds의 나노초 오프셋입니다.

timeInSeconds

Unix Epoch 형식의 타임스탬프 날짜(초)입니다. offsetInNanos에서 나노초 단위의 데이터를 제공합니다.

value

(필수) 자산 속성의 값입니다.

Note

value 필드에는 다음 값 중 하나만 포함될 수 있습니다.

booleanValue

(선택 사항) 부울(true 또는 false) 유형의 자산 속성 데이터.

doubleValue

(선택 사항) 배정밀도(부동 소수점 숫자) 유형의 자산 속성 데이터입니다.

integerValue

(선택 사항) 정수(범자연수) 유형의 자산 속성 데이터입니다.

stringValue

(선택 사항) 문자열(문자 시퀀스) 유형의 자산 속성 데이터입니다.

Edge에서 AWS IoT SiteWise 엣지 기능 구성

AWS IoT SiteWise Edge를 사용하여 데이터를 수집하고 임시로 저장하여 디바이스 데이터를 로컬에서 구성하고 처리할 수 있습니다. 엣지 처리를 활성화하면 집계된 데이터만 AWS 클라우드 로 전송하여 대역폭 사용량과 클라우드 스토리지 비용을 최적화할 수 있습니다. 와 함께 AWS IoT SiteWise 구

성 요소를 사용하면 로 전송하기 전에 엣지에서 데이터를 수집 및 처리 AWS 클라우드하거나 SiteWise Edge API를 사용하여 온프레미스에서 관리할 AWS IoT Greengrass 수 있습니다. APIs

데이터 수집은에서 실행되는 데이터 팩 및 AWS IoT SiteWise 구성 요소를 통해 이루어집니다 AWS IoT Greengrass.

Note

- AWS IoT SiteWise 는 SiteWise Edge 게이트웨이에 최대 30일까지 엣지 데이터를 유지합니다. 데이터 보존 기간은 디바이스의 사용 가능한 디스크 공간에 따라 다릅니다.
- SiteWise Edge 게이트웨이가에서 30일 AWS 클라우드 동안 연결 해제된 경우 [데이터 처리 팩](#)이 자동으로 비활성화됩니다.

주제

- [SiteWise Edge에서 엣지 기능 설정](#)

SiteWise Edge에서 엣지 기능 설정

AWS IoT SiteWise 는 SiteWise Edge 게이트웨이가 데이터를 수집하고 처리하는 방법을 결정하는 데 사용할 수 있는 다음 팩을 제공합니다. 팩을 선택하여 SiteWise Edge 게이트웨이의 엣지 기능을 활성화하세요.

- 데이터 수집 팩을 사용하면 SiteWise Edge 게이트웨이가 여러 OPC UA 서버에서 데이터를 수집한 다음 엣지에서 AWS 클라우드로 데이터를 내보낼 수 있습니다. SiteWise Edge 게이트웨이에 데이터 소스를 추가하면 활성화됩니다.
- 데이터 처리 팩을 사용하면 SiteWise Edge 게이트웨이가 엣지에서 장비 데이터를 처리할 수 있습니다. 예를 들어 자산 모델을 사용하여 지표와 변환을 계산할 수 있습니다. 자산 모델 및 자산에 대한 자세한 내용은 [산업 자산 모델링](#)을 참조하세요.

Note

- 데이터 처리 팩은 x86 플랫폼에서만 사용할 수 있습니다.

엣지 기능 구성

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.

2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 엣지 기능을 활성화하려는 SiteWise Edge 게이트웨이를 선택합니다.
4. 엣지 기능 섹션에서 편집을 선택합니다.
5. 엣지 기능 섹션에서 데이터 처리 팩 활성화(추가 요금 발생)를 선택합니다.
6. (선택 사항) 엣지 LDAP 연결 섹션에서 회사 디렉터리의 사용자 그룹에 이 SiteWise Edge 게이트웨이에 대한 액세스 권한을 부여할 수 있습니다. 사용자 그룹은 Lightweight Directory Access Protocol(LDAP) 자격 증명을 사용하여 SiteWise Edge 게이트웨이에 액세스할 수 있습니다. 그런 다음 AWS IoT SiteWise 애플리케이션, AWS IoT SiteWise API 작업 또는 기타 도구 AWS OpsHub 예를 사용하여 SiteWise Edge 게이트웨이를 관리할 수 있습니다. 자세한 내용은 [SiteWise Edge 게이트웨이 관리](#) 단원을 참조하십시오.

Note

Linux 또는 Microsoft Windows 자격 증명을 사용하여 SiteWise Edge 게이트웨이에 액세스할 수도 있습니다. 자세한 내용은 [Linux 운영 체제 자격 증명을 사용하여 SiteWise Edge 게이트웨이에 액세스](#) 단원을 참조하십시오.

- a. 활성화됨을 선택합니다.
 - b. 공급자 이름에 LDAP 공급자의 이름을 입력합니다.
 - c. 호스트 이름 또는 IP 주소에 LDAP 서버의 호스트 이름 또는 IP 주소를 입력합니다.
 - d. 포트에 포트 번호를 입력합니다.
 - e. 기본 고유 이름(DN)에 기본의 고유 이름(DN)을 입력합니다.
- 지원되는 속성 유형은 commonName(CN), localityName(L), stateOrProvinceName(ST), organizationName(O), organizationalUnitName(OU), countryName(C), streetAddress(STREET), domainComponent(DC), userid(UID)입니다.
- f. 관리 그룹 DN의 경우 DN을 입력합니다.
 - g. 사용자 그룹 DN의 경우 DN을 입력합니다.
7. 저장을 선택합니다.

이제 SiteWise Edge 게이트웨이에서 엣지 기능을 활성화했으므로 엣지에 대한 자산 모델을 구성해야 합니다. 자산 모델 엣지 구성은 자산 속성이 계산되는 위치를 지정합니다. 엣지에서 모든 속성을 계산하거나 자산 모델 속성을 개별적으로 구성할 수 있습니다. 자산 모델 속성에는 [지표](#), [변환](#) 및 [측정](#)이 포함됩니다.

자산 속성에 대한 자세한 내용은 [the section called “데이터 속성 정의”](#) 단원을 참조하세요.

자산 모델을 만든 후 엣지에 맞게 구성할 수 있습니다. 엣지용 자산 모델 구성에 대한 자세한 내용은 [the section called “자산 모델 생성\(콘솔\)”](#) 단원을 참조하세요.

Note

자산 모델 및 대시보드는 10분마다 AWS 클라우드와 SiteWise Edge 게이트웨이 간에 자동으로 동기화됩니다. 로컬 SiteWise Edge 게이트웨이 애플리케이션에서 수동으로 동기화할 수도 있습니다.

AWS IoT SiteWise 모델 및 자산에 대한 엣지 데이터 처리 구성

AWS IoT SiteWise Edge를 사용하여 장비 데이터를 로컬에서 수집, 저장, 구성 및 모니터링할 수 있습니다. SiteWise Edge를 사용하여 산업 데이터를 모델링하고 SiteWise Monitor를 사용하여 운영 직원이 로컬에서 데이터를 시각화할 수 있는 대시보드를 생성할 수 있습니다. 데이터를 로컬에서 처리하여 AWS 클라우드로 보내거나 AWS IoT SiteWise API를 사용하여 온프레미스에서 처리할 수 있습니다.

AWS IoT SiteWise Edge를 사용하면 원시 데이터를 로컬에서 처리하고 집계된 데이터만 AWS 클라우드로 전송하여 대역폭 사용량과 클라우드 스토리지 비용을 최적화할 수 있습니다.

Note

- AWS IoT SiteWise 는 SiteWise Edge 게이트웨이에 최대 30일까지 엣지 데이터를 유지합니다. 데이터 보존 기간은 디바이스의 사용 가능한 디스크 공간에 따라 다릅니다.
- SiteWise Edge 게이트웨이가 30일 동안 AWS 클라우드에서 연결 해제된 경우 [SiteWise Edge에서 OPC UA 소스 설정](#)가 자동으로 비활성화됩니다.

SiteWise Edge에서 데이터 처리를 위한 자산 모델 구성

엣지에서 SiteWise Edge 게이트웨이 데이터를 처리하려면 먼저 엣지에 대한 자산 모델을 구성해야 합니다. 자산 모델 엣지 구성은 자산 속성이 계산되는 위치를 지정합니다. 엣지의 모든 속성을 계산하여 결과를 AWS 클라우드로 보내거나 각 자산 속성을 별도로 계산할 위치를 사용자 지정할 수 있습니다. 자세한 내용은 [AWS IoT SiteWise 모델 및 자산에 대한 엣지 데이터 처리 구성](#) 단원을 참조하십시오.

자산 속성에는 지표, 변환 및 측정이 포함됩니다.

- 지표는 지정한 기간에 걸친 자산의 데이터 집계입니다. 기존 지표 데이터를 사용하여 새 지표를 계산할 수 있습니다. AWS IoT SiteWise 는 장기 스토리지를 AWS 위해 항상 지표를 클라우드로 전송합니다. 는 기본적으로 AWS 클라우드에서 지표를 AWS IoT SiteWise 계산합니다. 엣지에서 지표를 계산하도록 자산 모델을 구성할 수 있습니다. 는 처리된 결과를 AWS 클라우드로 AWS IoT SiteWise 보냅니다.
- 변형은 자산 속성의 데이터 포인트를 한 양식에서 다른 양식으로 매핑하는 수학적 표현식입니다. 변환은 지표를 입력 데이터로 사용할 수 있으며 입력과 동일한 위치에서 계산 및 저장되어야 합니다. 엣지에서 계산하도록 지표 입력을 구성한 경우 AWS IoT SiteWise 는 엣지에서 관련 변환도 계산합니다.
- 측정치는 기본적으로 디바이스가 수집하여 AWS 클라우드로 전송하는 원시 데이터로 형식이 지정됩니다. 이 데이터를 로컬 디바이스에 저장하도록 자산 모델을 구성할 수 있습니다.

자산 속성에 대한 자세한 내용은 [the section called “데이터 속성 정의”](#) 단원을 참조하세요.

자산 모델을 만든 후 엣지에 맞게 구성할 수 있습니다. 엣지용 자산 모델 구성에 대한 자세한 내용은 [the section called “자산 모델 생성\(콘솔\)”](#) 단원을 참조하세요.

Note

자산 모델 및 대시보드는 10분마다 AWS 클라우드와 SiteWise Edge 게이트웨이 간에 자동으로 동기화됩니다. [SiteWise Edge 게이트웨이 관리](#)에서 수동으로 동기화할 수도 있습니다.

AWS IoT SiteWise REST APIs 및 AWS Command Line Interface (AWS CLI)를 사용하여 SiteWise Edge 게이트웨이에서 엣지의 데이터를 쿼리할 수 있습니다. SiteWise Edge 게이트웨이에서 엣지에 있는 데이터를 쿼리하려면 먼저 다음 사전 조건을 충족해야 합니다.

- REST API에 대한 자격 증명을 설정해야 합니다. 자격 증명 설정에 대한 자세한 내용은 [the section called “게이트웨이 관리”](#) 단원을 참조하세요.
- SDK 엔드포인트가 SiteWise Edge 게이트웨이의 IP 주소를 가리켜야 합니다. 자세한 내용은 SDK 문서에서 확인할 수 있습니다. 예를 들어 AWS SDK for Java 2.x 개발자 가이드의 [사용자 지정 엔드포인트 지정](#)을 참조하세요.
- SiteWise Edge 게이트웨이 인증서를 등록해야 합니다. SiteWise Edge 게이트웨이 인증서 등록에 대한 자세한 내용은 SDK 설명서에서 확인할 수 있습니다. 예를 들어 AWS SDK for Java 2.x 개발자 가이드의 [Node.js에 인증서 번들 등록](#)을 참조하세요.

를 사용하여 데이터를 쿼리하는 방법에 대한 자세한 내용은 섹션을 [AWS IoT SiteWise참조하세요](#) [에서 데이터 쿼리 AWS IoT SiteWise](#).

AWS IoT SiteWise Edge 게이트웨이에 데이터 소스 추가

AWS IoT SiteWise Edge 게이트웨이를 설정한 후 데이터 소스를 추가하고 구성하여 로컬 산업 장비에서 데이터를 수집할 수 있습니다 AWS IoT SiteWise. SiteWise Edge는 OPC UA를 비롯한 다양한 프로토콜과 파트너 데이터 소스를 통해 사용할 수 있는 기타 여러 프로토콜을 지원합니다. 이러한 소스를 사용하면 게이트웨이가 로컬 서버에 연결하고 산업 데이터를 검색할 수 있습니다. 데이터 소스를 구성하면 다양한 데이터 소스에서 데이터를 수집한 다음 데이터 스트림을 자산 속성과 연결하여 포괄적인 산업 자산 모델링 및 데이터 매핑을 수행할 수 있습니다 AWS IoT SiteWise.

주제

- [AWS IoT SiteWise Edge 게이트웨이용 OPC UA 데이터 소스](#)
- [SiteWise Edge 게이트웨이의 파트너 데이터 소스](#)

AWS IoT SiteWise Edge 게이트웨이용 OPC UA 데이터 소스

AWS IoT SiteWise Edge 게이트웨이를 설정한 후 SiteWise Edge 게이트웨이가 로컬 산업 장비에서 데이터를 수집할 수 있도록 데이터 소스를 구성할 수 있습니다 AWS IoT SiteWise. 각 소스는 SiteWise Edge 게이트웨이가 산업 데이터 스트림을 연결하고 검색하는 OPC UA 서버와 같은 로컬 서버를 나타냅니다. SiteWise Edge 게이트웨이 설정에 대한 자세한 내용은 [자체 호스팅 SiteWise Edge 게이트웨이 생성](#)을 참조하세요.

게이트웨이 유형인 MQTT 지원 V3 게이트웨이와 클래식 스트림인 V2 게이트웨이는 OPC UA 데이터 처리 방식에 영향을 미칩니다. 클래식 스트림, V2 게이트웨이에서 OPC UA 데이터 소스는 게이트웨이 IoT SiteWise 게시자 구성에 직접 추가됩니다. 각 데이터 소스는 게이트웨이와 결합되고 데이터 라우팅은 각 소스에 대해 개별적으로 구성됩니다. 반대로 MQTT 지원 V3 게이트웨이를 사용하면 OPC UA 데이터 소스가 MQTT 주제로 변환되고 중앙 집중식 대상을 통해 관리됩니다. 각 유형에 대한 자세한 내용은 [MQTT 지원](#), [AWS IoT SiteWise Edge용 V3 게이트웨이](#) 및 단원을 참조하십시오 [클래식 스트림](#), [AWS IoT SiteWise Edge용 V2 게이트웨이](#).

Note

AWS IoT SiteWise 는 소스를 추가하거나 편집할 때마다 SiteWise Edge 게이트웨이를 다시 시작합니다. SiteWise Edge 게이트웨이는 소스 구성을 업데이트하는 동안 데이터를 수집하지 않습니다. SiteWise Edge 게이트웨이를 다시 시작하는 시간은 SiteWise Edge 게이트웨이 소스

의 태그 수에 따라 달라집니다. 다시 시작 시간은 몇 초(태그가 거의 없는 SiteWise Edge 게이트웨이)에서 몇 분(태그가 많은 SiteWise Edge 게이트웨이)에 이르기까지 다양할 수 있습니다.

소스를 생성한 후에는 데이터 스트림을 자산 속성과 연결할 수 있습니다. 자산을 만들고 사용하는 방법에 대한 자세한 내용은 [산업 자산 모델링](#) 단원을 참조하십시오.

CloudWatch 지표를 확인하여 데이터 소스가 AWS IoT SiteWise에 연결되어 있는지 확인할 수 있습니다. 자세한 내용은 [AWS IoT Greengrass Version 2 게이트웨이 지표](#) 단원을 참조하십시오.

현재는 다음 데이터 소스 프로토콜을 AWS IoT SiteWise 지원합니다.

- [OPC UA](#) – 산업 자동화를 위한 Machine-to-Machine(M2M) 통신 프로토콜입니다.

추가 산업 프로토콜 지원

SiteWise Edge는 데이터 소스 파트너와의 통합을 통해 광범위한 산업 프로토콜을 지원합니다. 이러한 파트너십을 통해 200개 이상의 서로 다른 프로토콜과 연결할 수 있으며 다양한 산업 시스템과 디바이스를 수용할 수 있습니다.

사용 가능한 데이터 소스 파트너 목록은 섹션을 참조하세요 [SiteWise Edge 게이트웨이 파트너 데이터 소스 옵션](#).

SiteWise Edge에서 OPC UA 소스 설정

AWS IoT SiteWise 콘솔 또는 SiteWise Edge 게이트웨이 기능을 사용하여 로컬 OPC UA 서버를 나타내는 OPC UA 소스를 정의하고 SiteWise Edge 게이트웨이에 추가할 수 있습니다.

주제

- [OPC UA 소스 구성\(콘솔\)](#)
- [OPC UA 소스 구성\(AWS CLI\)](#)

OPC UA 소스 구성(콘솔)

콘솔을 사용하여 다음 절차에 따라 OPC UA 소스를 구성할 수 있습니다.

Note

경고: TQVs 중복되면 이중 충전이 발생할 수 있습니다.

AWS IoT SiteWise 콘솔을 사용하여 OPC UA 소스를 구성하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. OPC UA 소스를 추가할 SiteWise Edge 게이트웨이를 선택합니다.
4. 데이터 소스 추가를 선택합니다.
5. 소스 이름을 입력합니다.
6. 데이터 소스 서버의 로컬 엔드포인트를 입력합니다. 엔드포인트는 IP 주소 또는 호스트 이름일 수 있습니다. 로컬 엔드포인트에 포트 번호를 추가할 수도 있습니다. 로컬 엔드포인트를 예를 들면 **opc.tcp://203.0.113.0:49320**과 같습니다.
7. (선택 사항) 선택할 노드 ID의 경우 노드 필터를 추가하여 AWS 클라우드에 수집되는 데이터 스트림을 제한합니다. 기본적으로 SiteWise Edge 게이트웨이는 서버의 루트 노드를 사용하여 모든 데이터 스트림을 수집합니다. 노드 필터를 사용하면 AWS IoT SiteWise에서 모델링하는 데이터에 대한 경로만 포함하여 SiteWise Edge 게이트웨이의 시작 시간과 CPU 사용량을 줄일 수 있습니다. 기본적으로 SiteWise Edge 게이트웨이는 /Server/로 시작하는 경로를 제외한 모든 OPC UA 경로를 업로드합니다. OPC UA 노드 필터를 정의하려면 노드 경로와 * 및 ** 와일드카드 문자를 사용하면 됩니다. 자세한 내용은 [SiteWise Edge에서 OPC UA 노드 필터 사용](#) 단원을 참조하십시오.
8. 대상은 MQTT 지원 V3 게이트웨이와 Classic 스트림, V2 게이트웨이 간에 다릅니다.
 - 클래식 스트림인 V2 게이트웨이 대상은 소스와 1:1 관계를 갖습니다. 각 소스는 특정 대상으로 데이터를 전송합니다.
 - 허브 및 스포크 모델을 사용하면 여러 게이트웨이에서 여러 데이터 소스의 구성 및 관리를 중앙 집중화할 수 있으므로 MQTT를 사용하는 V3 게이트웨이 대상은 별도로 설정됩니다. V3 게이트웨이에서 대상을 설정하려면 섹션을 참조하세요 [AWS IoT SiteWise 엣지 대상 이해](#).

Classic streams, V2 gateway destinations

- AWS IoT SiteWise 실시간 - 스토리지로 직접 데이터를 보내려면 AWS IoT SiteWise 선택합니다. 엣지에서 실시간으로 데이터를 수집하고 모니터링합니다.
- AWS IoT SiteWise Amazon S3를 사용하여 버퍼링 - Parquet 형식의 데이터를 Amazon S3로 전송한 다음 스토리지로 AWS IoT SiteWise 가져옵니다. 데이터를 배치로 수집하고 기록 데이터를 비용 효과적인 방식으로 저장하려면 이 옵션을 선택합니다. 원하는 Amazon S3 버킷 위치와 데이터를 Amazon S3에 업로드할 빈도를 구성할 수 있습니다. AWS IoT SiteWise에 수집한 후 데이터로 수행할 작업을 선택할 수도 있습니다. AWS IoT SiteWise 및 Amazon

S3 모두에서 데이터를 사용할 수 있도록 선택하거나 로 가져온 후 Amazon S3에서 자동으로 삭제하도록 선택할 수 있습니다 AWS IoT SiteWise.

- Amazon S3 버킷은 스테이징 및 버퍼링 메커니즘이며 Parquet 형식의 파일을 지원합니다.
- 스토리지로 AWS IoT SiteWise 데이터 가져오기 확인란을 선택하면 데이터가 먼저 Amazon S3에 업로드된 다음 AWS IoT SiteWise 스토리지에 업로드됩니다.
- Amazon S3에서 데이터 삭제 확인란을 선택하면 SiteWise 스토리지로 가져온 후 Amazon S3에서 데이터가 삭제됩니다.
- Amazon S3에서 데이터 삭제 확인란을 선택 취소하면 데이터가 Amazon S3와 SiteWise 스토리지 모두에 저장됩니다.
- AWS IoT SiteWise 스토리지로 데이터 가져오기 확인란을 선택 취소하면 데이터가 Amazon S3에만 저장됩니다. 데이터를 SiteWise 스토리지로 가져오지 않습니다.

AWS IoT SiteWise 에서 제공하는 다양한 스토리지 옵션에 대한 자세한 내용을 [데이터 스토리지 관리](#) 섹션을 참조하세요. 요금 옵션에 대한 자세한 내용은 [AWS IoT SiteWise 요금](#)을 참조하세요.

- AWS IoT Greengrass 스트림 관리자 - AWS IoT Greengrass 스트림 관리자를 사용하여의 채널 AWS IoT Analytics, Amazon Kinesis Data Streams의 스트림,의 자산 속성 AWS IoT SiteWise또는 Amazon Simple Storage Service(Amazon S3)의 객체와 같은 AWS 클라우드 대상으로 데이터를 전송합니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [AWS IoT Greengrass 코어에서 데이터 스트림 관리를 참조하세요](#).

AWS IoT Greengrass 스트림의 이름을 입력합니다.

MQTT-enabled, V3 gateway destinations

1. 관련 대상 추가에 [MQTT 지원, AWS IoT SiteWise Edge용 V3 게이트웨이](#) 대한 자세한 내용은 섹션을 참조하세요.
 2. 소스 대상을 추가한 후이 절차로 돌아갑니다.
9. 고급 구성 창에서 다음을 수행할 수 있습니다.
- a. 소스 서버와 SiteWise Edge 게이트웨이 간의 연결 및 전송 중 데이터에 대해 메시지 보안 모드를 선택합니다. 이 필드는 OPC UA 보안 정책과 메시지 보안 모드의 조합입니다. OPC UA 서버에 지정한 것과 동일한 보안 정책 및 메시지 보안 모드를 선택합니다.
 - b. 소스에 인증이 필요한 경우 인증 구성 목록에서 AWS Secrets Manager 암호를 선택합니다. SiteWise Edge 게이트웨이는 이 데이터 소스에 연결할 때 이 암호에 있는 인증 자격 증

명을 사용합니다. 데이터 소스 인증에 사용하려면 SiteWise Edge 게이트웨이의 AWS IoT Greengrass 구성 요소에 보안 암호를 연결해야 합니다. 자세한 내용은 [the section called “데이터 소스 인증 구성”](#) 단원을 참조하십시오.

 Tip

데이터 서버에는 [Allow anonymous login(익명 로그인 허용)]이라는 옵션이 있을 수 있습니다. 이 옵션이 [Yes(예)]인 경우 원본에 인증이 필요하지 않습니다.

- c. (선택 사항) 데이터 스트림 접두사 활성화 - 선택 사항을 선택하여 데이터 스트림 접두사를 활성화할 수 있습니다.
 - 데이터 스트림 접두사를 입력합니다. SiteWise Edge 게이트웨이는 이 접두사를 이 소스의 모든 데이터 스트림에 추가합니다. 데이터 스트림 접두사를 사용하여 서로 다른 소스에서 이름이 같은 데이터 스트림을 구분합니다. 각 데이터 스트림은 계정 내에서 고유한 이름을 가져야 합니다.
- d. (선택 사항) 지원되지 않는 OPC UA 데이터 유형을 AWS IoT SiteWise로 수집하기 전에 문자열로 변환하려면 데이터 유형 변환 옵션을 선택합니다. 간단한 데이터 형식의 배열 값을 JSON 문자열로 변환하고 DateTime 데이터 형식을 ISO 8601 문자열로 변환합니다. 자세한 내용은 [지원되지 않는 데이터 유형 변환](#) 단원을 참조하십시오.
- e. (선택 사항) 속성 그룹의 경우 새 그룹 추가를 선택합니다.
 - i. 속성 그룹의 이름을 입력합니다.
 - ii. 속성의 경우:
 - 1. 노드 경로의 경우 OPC UA 노드 필터를 추가하여 AWS IoT SiteWise에 업로드되는 OPC UA 경로를 제한합니다. 형식은 선택할 노드 ID와 유사합니다.
 - iii. 그룹 설정에서 다음을 수행합니다.
 - 1. 데이터 품질 설정에서 AWS IoT SiteWise 수집기가 수집할 데이터 품질 유형을 선택합니다.
 - 2. 스캔 모드 설정의 경우 스캔 모드를 사용하여 표준 구독 속성을 구성합니다. 구독 또는 폴링을 선택할 수 있습니다. 스캔 모드에 대한 자세한 내용은 [the section called “데이터 수집 범위 필터링”](#)을 참조하세요.

Subscribe

모든 데이터 포인트를 보내려면

- i. 구독을 선택하고 다음을 설정합니다.
 - A. [데이터 변경 트리거](#) - 데이터 변경 알림을 시작하는 조건입니다.
 - B. [구독 대기열 크기](#) - 모니터링되는 항목에 대한 알림이 대기열에 있는 특정 지표에 대한 OPC-UA 서버의 대기열 깊이입니다.
 - C. [구독 게시 간격](#) - 구독이 생성될 때 지정된 게시 주기의 간격(밀리초)입니다.
 - D. 스냅샷 간격 - 선택 사항 - AWS IoT SiteWise Edge가 안정적인 데이터 스트림을 수집하도록 하는 스냅샷 빈도 제한 시간 설정입니다.
 - E. 스캔 속도 - SiteWise Edge 게이트웨이가 registers를 읽도록 하려는 속도입니다.는 SiteWise Edge 게이트웨이에 허용되는 최소 스캔 속도를 AWS IoT SiteWise 자동으로 계산합니다.
 - F. 타임스탬프 - OPC UA 데이터 포인트에 포함할 타임스탬프입니다. 서버 타임스탬프 또는 디바이스의 타임스탬프를 사용할 수 있습니다.

Note

IoT SiteWise OPC UA 수집기 구성 요소의 버전 2.5.0 이상을 사용합니다. 이전 버전에서 타임스탬프 기능을 사용하는 경우 구성 업데이트가 실패합니다. 자세한 내용은 [구성 요소의 AWS IoT SiteWise 버전 업데이트](#) 단원을 참조하십시오.

- ii. 데드밴드 설정에서 데드밴드 유형을 구성합니다. 데드밴드 유형은 소스가에 전송하는 데이터와 폐기하는 AWS IoT SiteWise 데이터를 제어합니다. 데드밴드 설정에 대한 자세한 내용은 [the section called “데이터 수집 범위 필터링”](#)을 참조하세요.
 - 없음 - 연결된 서버가 이 속성 그룹에 대한 모든 데이터 포인트를 전송합니다.
 - 백분율 - 연결된 서버는 데이터 범위의 지정된 백분율을 벗어나는 데이터만 전송합니다. 이 범위는 각 노드에 대해 정의된 엔지니어링 단위 최소값 및 최대값을 기준으로 서버에서 계산합니다. 서버가 백분율 데드밴드를 지원하지

지 않거나 정의된 엔지니어링 단위가 없는 경우 게이트웨이는 아래에 제공된 최소값과 최대값을 사용하여 범위를 계산합니다.

- 절대 - 연결된 서버는 특정 범위를 벗어나는 데이터만 전송합니다.
- A. 데드밴드 값을 데드밴드에 대한 데이터 범위의 백분율로 설정합니다.
- B. (선택 사항) 최소 범위 - 선택 사항 및 최대 범위 - 선택 사항을 사용하여 데드밴드 범위의 최소값 및 최대값을 지정합니다.

Poll

특정 간격으로 데이터 포인트를 보내려면

- 폴링을 선택하고 다음을 설정합니다.
- A. 스캔 속도 - SiteWise Edge 게이트웨이가 registers를 읽도록 하려는 속도입니다.는 SiteWise Edge 게이트웨이에 허용되는 최소 스캔 속도를 AWS IoT SiteWise 자동으로 계산합니다.
- B. 타임스탬프 - OPC UA 데이터 포인트에 포함할 타임스탬프입니다. 서버 타임스탬프 또는 디바이스의 타임스탬프를 사용할 수 있습니다.



Note

IoT SiteWise OPC UA 수집기 구성 요소의 버전 2.5.0 이상을 사용합니다. 이전 버전에서 타임스탬프 기능을 사용하는 경우 구성 업데이트가 실패합니다. 자세한 내용은 [구성 요소의 AWS IoT SiteWise 버전 업데이트](#) 단원을 참조하십시오.



Note

데드밴드 설정은 스캔 모드 설정에서 구독을 선택한 경우에 적용됩니다.

10. 저장을 선택합니다.

OPC UA 소스 구성(AWS CLI)

AWS CLI를 사용하여 SiteWise Edge 게이트웨이에 대한 OPC UA 데이터 소스를 정의할 수 있습니다. 이렇게 하려면 OPC UA 기능 구성 JSON 파일을 생성하고 [update-gateway-capability-configuration](#) 명령을 사용하여 SiteWise Edge 게이트웨이 구성을 업데이트합니다. 단일 기능 구성에서 모든 OPC UA 소스를 정의해야 합니다.

MQTT-enabled, V3 gateway

이 기능에는 다음과 같은 네임스페이스가 있습니다.

- `iotsitewise:opcuacollector:3`

```
{
  "sources": [
    {
      "name": "string",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny" | "X509",
          "certificateBody": "string",
          "certificateChain": "string",
        },
        "endpointUri": "string",
        "securityPolicy": "NONE" | "BASIC128_RSA15" | "BASIC256" | "BASIC256_SHA256"
        | "AES128_SHA256_RSA0AEP" | "AES256_SHA256_RSAPSS",
        "messageSecurityMode": "NONE" | "SIGN" | "SIGN_AND_ENCRYPT",
        "identityProvider": {
          "type": "Anonymous" | "Username",
          "usernameSecretArn": "string"
        },
        "nodeFilterRules": [
          {
            "action": "INCLUDE",
            "definition": {
              "type": "OpcUaRootPath",
              "rootPath": "string"
            }
          }
        ]
      },
      "measurementDataStreamPrefix": "string",
    }
  ]
}
```

```

"typeConversions": {
  "array": "JsonArray",
  "datetime": "ISO8601String"
},
"destination": {
  {
    "type": "MQTT"
  }
},
"propertyGroups": [
  {
    "name": "string",
    "nodeFilterRuleDefinitions": [
      {
        "type": "OpcUaRootPath",
        "rootPath": "string"
      }
    ],
    "deadband": {
      "type": "PERCENT" | "ABSOLUTE",
      "value": double,
      "eguMin": double,
      "eguMax": double,
      "timeoutMilliseconds": integer
    },
    "scanMode": {
      "type": "EXCEPTION" | "POLL",
      "rate": integer,
      "timestampToReturn": "SOURCE_TIME" | "SERVER_TIME"
    },
    "dataQuality": {
      "allowGoodQuality": true | false,
      "allowBadQuality": true | false,
      "allowUncertainQuality": true | false
    },
    "subscription": {
      "dataChangeTrigger": "STATUS" | "STATUS_VALUE" |
"STATUS_VALUE_TIMESTAMP",
      "queueSize": integer,
      "publishingIntervalMilliseconds": integer,
      "snapshotFrequencyMilliseconds": integer
    }
  }
]

```

```

    }
  ]
}
```

Classic streams, V2 gateway

이 기능에는 다음과 같은 네임스페이스가 있습니다.

- `iotsitewise:opcuacollector:2`

요청 구문

```

{
  "sources": [
    {
      "name": "string",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny" | "X509",
          "certificateBody": "string",
          "certificateChain": "string",
        },
        "endpointUri": "string",
        "securityPolicy": "NONE" | "BASIC128_RSA15" | "BASIC256" | "BASIC256_SHA256"
        | "AES128_SHA256_RSA0AEP" | "AES256_SHA256_RSAPSS",
        "messageSecurityMode": "NONE" | "SIGN" | "SIGN_AND_ENCRYPT",
        "identityProvider": {
          "type": "Anonymous" | "Username",
          "usernameSecretArn": "string"
        },
        "nodeFilterRules": [
          {
            "action": "INCLUDE",
            "definition": {
              "type": "OpcUaRootPath",
              "rootPath": "string"
            }
          }
        ]
      },
      "measurementDataStreamPrefix": "string",
      "typeConversions": {
        "array": "JsonArray",
```

```

    "datetime": "ISO8601String"
  },
  "destination": {
    "type": "StreamManager",
    "streamName": "string",
    "streamBufferSize": integer,
  },
  "propertyGroups": [
    {
      "name": "string",
      "nodeFilterRuleDefinitions": [
        {
          "type": "OpcUaRootPath",
          "rootPath": "string"
        }
      ],
      "deadband": {
        "type": "PERCENT" | "ABSOLUTE",
        "value": double,
        "eguMin": double,
        "eguMax": double,
        "timeoutMilliseconds": integer
      },
      "scanMode": {
        "type": "EXCEPTION" | "POLL",
        "rate": integer,
        "timestampToReturn": "SOURCE_TIME" | "SERVER_TIME"
      },
      "dataQuality": {
        "allowGoodQuality": true | false,
        "allowBadQuality": true | false,
        "allowUncertainQuality": true | false
      },
      "subscription": {
        "dataChangeTrigger": "STATUS" | "STATUS_VALUE" |
"STATUS_VALUE_TIMESTAMP",
        "queueSize": integer,
        "publishingIntervalMilliseconds": integer,
        "snapshotFrequencyMilliseconds": integer
      }
    }
  ]
}
]

```

```
}
```

요청 본문

sources

다음 정보를 포함하는 각 OPC UA 소스 정의 구조의 목록입니다.

name

고유하고 친숙한 원본 이름입니다.

endpoint

다음 정보가 포함된 엔드포인트 구조입니다.

certificateTrust

다음 정보가 포함된 인증서 신뢰 정책 구조:

type

원본에 대한 인증서 신뢰 모드입니다. 다음 중 하나를 선택합니다.

- TrustAny – SiteWise Edge 게이트웨이는 OPC UA 소스에 연결될 때 모든 인증서를 신뢰합니다.
- X509 – SiteWise Edge 게이트웨이는 OPC UA 소스에 연결될 때 X.509 인증서를 신뢰합니다. 이 옵션을 선택하는 경우, certificateTrust에 certificateBody를 정의해야 합니다. certificateTrust에 certificateChain을 정의할 수도 있습니다.

certificateBody

(선택 사항) X.509 인증서의 본문입니다.

certificateTrust에서 type에 대해 X509을 선택하는 경우, 이 필드는 필수입니다.

certificateChain

(선택 사항) X.509 인증서에 대한 신뢰 체인입니다.

이 필드는 certificateTrust에서 type에 대해 X509를 선택할 때만 사용합니다.

endpointUri

OPC UA 소스의 로컬 엔드포인트입니다. 로컬 엔드포인트를 예를 들면 `opc.tcp://203.0.113.0:49320`과 같습니다.

securityPolicy

OPC UA 소스에서 읽은 메시지를 보호할 수 있도록 사용할 보안 정책입니다. 다음 중 하나를 선택합니다.

- NONE – SiteWise Edge 게이트웨이는 OPC UA 소스의 메시지를 보호하지 않습니다. 다른 보안 정책을 선택하는 것이 좋습니다. 이 옵션을 선택하는 경우, messageSecurityMode에 대해 NONE을 선택해야 합니다
- BASIC256_SHA256— Basic256Sha256 보안 정책.
- AES128_SHA256_RSA0AEP— Aes128_Sha256_Rsa0aep 보안 정책.
- AES256_SHA256_RSAPSS— Aes256_Sha256_RsaPss 보안 정책.
- BASIC128_RSA15 – (더 이상 사용되지 않음) Basic128Rsa15 보안 정책은 더 이상 안전한 것으로 간주되지 않으므로 OPC UA 사양에서 더 이상 사용되지 않습니다. 다른 보안 정책을 선택하는 것이 좋습니다. 자세한 내용은 [Profile SecurityPolicy – Basic128Rsa15](#)를 참조하세요.
- BASIC256 – (더 이상 사용되지 않음) Basic256 보안 정책은 더 이상 안전한 것으로 간주되지 않으므로 OPC UA 사양에서 더 이상 사용되지 않습니다. 다른 보안 정책을 선택하는 것이 좋습니다. 자세한 내용은 [SecurityPolicy – Basic256](#)을 참조하세요.

Important

이외의 NONE 보안 정책을 선택하는 경우 SIGN 또는 SIGN_AND_ENCRYPT 정책을 선택해야 합니다messageSecurityMode. 또한 SiteWise Edge 게이트웨이를 신뢰하도록 소스 서버를 구성해야 합니다. 자세한 내용은 [AWS IoT SiteWise Edge 게이트웨이를 신뢰하도록 OPC UA 서버 설정](#) 단원을 참조하십시오.

messageSecurityMode

OPC UA 소스에 대한 연결을 보호하는 데 사용할 메시지 보안 모드입니다. 다음 중 하나를 선택합니다.

- NONE – SiteWise Edge 게이트웨이는 OPC UA 소스에 대한 연결을 보호하지 않습니다. 다른 메시지 보안 모드를 선택하는 것이 좋습니다. 이 옵션을 선택하는 경우, securityPolicy에 대해 NONE을 선택해야 합니다
- SIGN – SiteWise Edge 게이트웨이와 OPC UA 소스 간 전송 중 데이터는 서명되지만 암호화되지는 않습니다.

- SIGN_AND_ENCRYPT – 게이트웨이와 OPC UA 소스 간 전송 중 데이터는 서명되고 암호화됩니다.

Important

NONE 이외의 메시지 보안 모드를 선택하는 경우 NONE 이외의 securityPolicy를 선택해야 합니다. 또한 SiteWise Edge 게이트웨이를 신뢰하도록 소스 서버를 구성해야 합니다. 자세한 내용은 [AWS IoT SiteWise Edge 게이트웨이를 신뢰하도록 OPC UA 서버 설정](#) 단원을 참조하십시오.

identityProvider

다음 정보가 포함된 자격 증명 공급자 구조입니다.

type

원본에 필요한 인증 보안 인증의 유형입니다. 다음 중 하나를 선택합니다.

- Anonymous – 원본에 연결하려면 인증이 필요하지 않습니다.
- Username – 원본에 연결하려면 사용자 이름과 암호가 필요합니다. 이 옵션을 선택하는 경우, identityProvider에 usernameSecretArn을 정의해야 합니다.

usernameSecretArn

(선택 사항) AWS Secrets Manager 보안 암호의 ARN입니다. SiteWise Edge 게이트웨이는 이 소스에 연결할 때 이 암호에 있는 인증 자격 증명을 사용합니다. 소스 인증에 사용하려면 SiteWise Edge 게이트웨이의 IoT SiteWise 커넥터에 암호를 첨부해야 합니다. 자세한 내용은 [SiteWise Edge에 대한 데이터 소스 인증 구성](#) 단원을 참조하십시오.

identityProvider에서 type에 대해 Username을 선택하는 경우, 이 필드는 필수입니다.

nodeFilterRules

AWS 클라우드로 전송할 OPC UA 데이터 스트림 경로를 정의하는 노드 필터 규칙 구조의 목록입니다. 노드 필터를 사용하면 AWS IoT SiteWise에서 모델링하는 데이터에 대한 경로만 포함하여 SiteWise Edge 게이트웨이의 시작 시간과 CPU 사용량을 줄일 수 있습니다. 기본적으로 SiteWise Edge 게이트웨이는 /Server/로 시작하는 경로를 제외한 모든 OPC UA 경로를 업로드합니다. OPC UA 노드 필터를 정의하려면 노드 경로와 * 및 ** 와일드카드 문자를 사용하면 됩니다. 자세한 내용은 [SiteWise Edge에서 OPC UA 노드 필터 사용](#) 단원을 참조하십시오.

각 목록 구조에는 다음 정보가 포함되어야 합니다.

action

이 노드 필터 규칙에 대한 작업입니다. 다음과 같은 옵션을 선택할 수 있습니다.

- INCLUDE – SiteWise Edge 게이트웨이에는 이 규칙과 일치하는 데이터 스트림만 포함됩니다.

definition

다음 정보가 포함된 노드 필터 규칙 구조:

type

이 규칙의 노드 필터 경로 유형입니다. 다음과 같은 옵션을 선택할 수 있습니다.

- OpcUaRootPath – SiteWise Edge 게이트웨이는 OPC UA 경로 계층의 루트에 대해 이 노드 필터 경로를 평가합니다.

rootPath

OPC UA 경로 계층의 루트에 대해 평가할 노드 필터 경로입니다. 이 경로는 /로 시작되어야 합니다.

measurementDataStreamPrefix

원본의 모든 데이터 스트림 앞에 추가할 문자열입니다. SiteWise Edge 게이트웨이는 이 접두사를 이 소스의 모든 데이터 스트림에 추가합니다. 데이터 스트림 접두사를 사용하여 서로 다른 소스에서 이름이 같은 데이터 스트림을 구분합니다. 각 데이터 스트림은 계정 내에서 고유한 이름을 가져야 합니다.

typeConversions

지원되지 않는 OPC UA 데이터 유형에 사용할 수 있는 변환 유형입니다. 각 데이터 유형은 문자열로 변환됩니다. 자세한 내용은 [지원되지 않는 데이터 유형 변환](#) 단원을 참조하십시오.

array

문자열로 변환되는 단순 배열 데이터 유형입니다. 다음과 같은 옵션을 선택할 수 있습니다.

- JSONArray - 단순 배열 데이터 유형을 문자열로 변환하도록 선택했음을 나타냅니다.

datetime

문자열로 변환되는 DateTime 데이터 유형입니다. 다음과 같은 옵션을 선택할 수 있습니다.

- ISO8601String – ISO 8601 데이터 유형을 문자열로 변환하도록 선택했음을 나타냅니다.

destination

OPC UA 태그의 대상에 대한 구성입니다. 클래식 스트림, v2 및 MQTT 지원 V3 게이트웨이는 대상에 대해 구성이 다릅니다.

type

목적지의 유형입니다.

streamName - Classic 스트림, V2 게이트웨이에만 해당

스트림의 이름입니다. 스트림 이름은 고유해야 합니다.

streamBufferSize - Classic 스트림, V2 게이트웨이에만 해당

스트림 버퍼의 크기입니다. 이는 OPC UA 소스의 데이터 흐름을 관리하는 데 중요합니다.

propertyGroups

(선택 사항) 프로토콜에서 요청한 deadband 및 scanMode을 정의하는 속성 그룹 목록입니다.

name

속성 그룹의 이름입니다. 고유 식별자여야 합니다.

deadband

deadband 값은 데이터가 클라우드로 전송되기 전에 발생해야 하는 데이터 포인트 값의 최소 변화를 정의합니다. 여기에는 다음 정보가 포함됩니다.

type

지원되는 데드밴드 유형입니다. 다음과 같은 옵션을 선택할 수 있습니다.

- ABSOLUTE - 클라우드로 전송하기에 충분히 중요한 데이터 포인트를 고려하는 데 필요한 최소 절대 변화를 지정하는 고정 값입니다.
- PERCENT - 필요한 최소 변화를 마지막으로 전송된 데이터 포인트 값의 백분율로 지정하는 동적 값입니다. 이러한 유형의 데드밴드는 데이터 값이 시간이 지남에 따라 크게 달라질 때 유용합니다.

value

데드밴드의 값입니다. type가 ABSOLUTE인 경우 이 값은 단위가 없는 이중 값입니다. type가 PERCENT인 경우 이 값은 1과 100 사이의 두 배입니다.

eguMin

(선택 사항) PERCENT 데드밴드를 사용할 때의 엔지니어링 단위 최소값입니다. OPC UA 서버에 엔지니어링 단위가 구성되어 있지 않은 경우 이를 설정합니다.

eguMax

(선택 사항) PERCENT 데드밴드를 사용할 때의 엔지니어링 단위 최대값입니다. OPC UA 서버에 엔지니어링 단위가 구성되어 있지 않은 경우 이를 설정합니다.

timeoutMilliseconds

타임아웃까지 남은 시간 (밀리초). 최소값은 100입니다.

scanMode

scanMode 구조에는 다음 정보가 포함되어 있습니다.

type

scanMode의 지원되는 유형입니다. 허용되는 값은 POLL 및 EXCEPTION입니다.

rate

스캔 모드의 샘플링 간격입니다.

timestampToReturn

타임스탬프의 소스입니다. 다음과 같은 옵션을 선택할 수 있습니다.

- SOURCE_TIME - 디바이스의 타임스탬프를 사용합니다.
- SERVER_TIME - 서버의 타임스탬프를 사용합니다.

Note

IoT SiteWise OPC UA 수집기 구성 요소의 버전 2.5.0 이상 TimestampToReturn에서 사용합니다. 이전 버전에서 이 기능을 사용하는 경우 구성 업데이트가 실패합니다. 자세한 내용은 [구성 요소의 AWS IoT SiteWise 버전 업데이트](#) 단원을 참조하십시오.

nodeFilterRuleDefinitions

(선택 사항) 속성 그룹에 포함할 노드 경로 목록입니다. 속성 그룹은 겹칠 수 없습니다. 이 필드에 값을 지정하지 않으면 그룹에 루트 아래의 모든 경로가 포함되므로 추가 속성 그룹을 생성할 수 없습니다. nodeFilterRuleDefinitions 구조에는 다음 정보가 포함되어 있습니다.

type

지원되는 유형은 OpcUaRootPath뿐입니다. rootPath 값이 OPC UA 브라우징 스페이스의 루트에 상대적인 경로임을 지정합니다.

rootPath

속성 그룹에 포함할 경로(루트에 대한 경로)를 지정하는 쉼표로 구분된 목록입니다.

Classic 스트림, V2 게이트웨이(AWS CLI)에 대한 추가 기능 구성 예제

다음 예시에서는 JSON 파일에 저장된 페이로드에서 OPC UA SiteWise Edge 게이트웨이 기능 구성을 정의합니다.

```
aws iotsitewise update-gateway-capability-configuration \
--capability-namespace "iotsitewise:opcuacollector:2" \
--capability-configuration file://opc-ua-configuration.json
```

Example : OPC UA 소스 구성

다음 opc-ua-configuration.json 파일은 안전하지 않은 기본 OPC UA 소스 구성을 정의합니다.

```
{
  "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 : 정의된 속성 그룹이 있는 OPC UA 소스 구성

다음 `opc-ua-configuration.json` 파일은 정의된 속성 그룹이 있는 안전하지 않은 기본 OPC UA 소스 구성을 정의합니다.

```
{
  "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 : 속성을 포함한 OPC UA 소스 구성

opc-ua-configuration.json에 대한 다음 JSON 예시는 다음 속성을 가진 OPC UA 소스 구성을 정의합니다.

- 모든 인증서를 신뢰합니다.
- BASIC256 보안 정책을 사용하여 메시지를 보호합니다.
- SIGN_AND_ENCRYPT 모드를 사용하여 연결을 보호합니다.
- Secrets Manager 암호에 저장된 인증 보안 인증을 사용합니다.
- 경로가 /WindFarm/2/WindTurbine/로 시작하는 데이터 스트림을 제외한 데이터 스트림을 필터링합니다.
- 모든 데이터 스트림 경로의 첫 부분에 /Washington을 추가하여 다른 영역에서 이 “Wind Farm #2”와 “Wind Farm #2”를 구분합니다.

```
{
  "sources": [
    {
      "name": "Wind Farm #2",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny"
        },
        "endpointUri": "opc.tcp://203.0.113.1:49320",
        "securityPolicy": "BASIC256",
        "messageSecurityMode": "SIGN_AND_ENCRYPT",
        "identityProvider": {
          "type": "Username",
          "usernameSecretArn":
            "arn:aws:secretsmanager:region:123456789012:secret:greenrass-windfarm2-auth-1ABCDE"
        },
        "nodeFilterRules": [
          {
            "action": "INCLUDE",
            "definition": {
              "type": "OpcUaRootPath",
              "rootPath": "/WindFarm/2/WindTurbine/"
            }
          }
        ]
      },
      "measurementDataStreamPrefix": "/Washington"
    }
  ]
}
```

Example : 인증서 신뢰가 있는 OPC UA 소스 구성

opc-ua-configuration.json에 대한 다음 JSON 예시는 다음 속성을 가진 OPC UA 소스 구성을 정의합니다.

- 지정된 X.509 인증서를 신뢰합니다.
- BASIC256 보안 정책을 사용하여 메시지를 보호합니다.
- SIGN_AND_ENCRYPT 모드를 사용하여 연결을 보호합니다.

```
{
  "sources": [
    {
      "name": "Wind Farm #3",
      "endpoint": {
        "certificateTrust": {
          "type": "X509",
          "certificateBody": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w
0BAQUFADCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZ
WF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAwTC0lBTsBDb25zb2x1MRIw
EAYDVQQDEw1UZXRhbnQ2IjYwMxHZAAdBgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5
jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTEwNDI1MjA0NTIxWjCBiDELMAKGA1UEBh
MCVVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBb
WF6b24xFDASBgNVBAwTC0lBTsBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnQ2IjYwMx
HZAAdBgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5jb20wgZ8wDQYJKoZIhvcNAQE
BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLyGVI
k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
ITx0USQv7c7ugFFDzQGBZsWY6786m86gpEibb30hjZnzcVQAaRHhd1QWIMm2nr
AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJI1J00zbhNYS5f6Guo
EDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjStbNYiytVbZPQUQ5Yaxu2jXnimvw
3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
          "certificateChain": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uX0jANBgkqhkiG9w
0BAQUFADCBiDELMAKGA1UEBhMCVVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZ
WF0dGx1MQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAwTC0lBTsBDb25zb2x1MRIw
EAYDVQQDEw1UZXRhbnQ2IjYwMxHZAAdBgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5
jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTEwNDI1MjA0NTIxWjCBiDELMAKGA1UEBh
MCVVMxCzAJBgNVBAGTAldBMRAwDgYDVQQHEwdTZWF0dGx1MQ8wDQYDVQQKEwZBb
WF6b24xFDASBgNVBAwTC0lBTsBDb25zb2x1MRIwEAYDVQQDEw1UZXRhbnQ2IjYwMx
HZAAdBgkqhkiG9w0BCQEWEG5vb25lQGFTYXpvi5jb20wgZ8wDQYJKoZIhvcNAQE
```

```

BBQADgY0AMIGJAOGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
ITx0USQv7c7ugFFDzQGBzZswY6786m86gpEIbb30hjZnzcvcQAaRHhd1QWIMm2nr
AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJI1J00zbhNYS5f6Guo
EDmFJl0ZxBHjJnyp3780D8uTs7fLvJx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
3rrszlaEXAMPLE=
-----END CERTIFICATE-----"
    },
    "endpointUri": "opc.tcp://203.0.113.2:49320",
    "securityPolicy": "BASIC256",
    "messageSecurityMode": "SIGN_AND_ENCRYPT",
    "identityProvider": {
        "type": "Anonymous"
    },
    "nodeFilterRules": []
  },
  "measurementDataStreamPrefix": ""
}
]
}

```

AWS IoT SiteWise Edge 게이트웨이를 신뢰하도록 OPC UA 서버 설정

OPC UA 소스를 구성할 때 없음 messageSecurityMode 이외의 것을 선택하는 경우 소스 서버가 AWS IoT SiteWise Edge 게이트웨이를 신뢰하도록 활성화해야 합니다. SiteWise Edge 게이트웨이는 소스 서버에 필요할 수 있는 인증서를 생성합니다. 프로세스는 소스 서버에 따라 다릅니다. 자세한 내용은 서버에 대한 설명서를 참조하세요.

다음 절차에서는 기본 단계를 간략하게 설명합니다.

OPC UA 서버에서 SiteWise Edge 게이트웨이를 신뢰하도록 설정하려면 다음을 수행하세요.

1. OPC UA 서버를 구성하기 위한 인터페이스를 엽니다.
2. OPC UA 서버 관리자의 사용자 이름과 암호를 입력합니다.
3. 인터페이스에서 신뢰할 수 있는 클라이언트를 찾은 다음, AWS IoT SiteWise 게이트웨이 클라이언트를 선택합니다.
4. 신뢰를 선택하세요.

OPC UA 클라이언트 인증서 내보내기

일부 OPC UA 서버는 SiteWise Edge 게이트웨이를 신뢰하기 위해 OPC UA 클라이언트 인증서 파일에 액세스해야 합니다. OPC UA 서버에 적용되는 경우, 다음 절차를 사용하여 SiteWise Edge 게이트웨이에서 OPC UA 클라이언트 인증서를 내보낼 수 있습니다. 그런 다음 OPC UA 서버에서 인증서를 가져올 수 있습니다.

소스에 대한 OPC UA 클라이언트 인증서 파일을 내보내려면

1. 다음 명령을 실행하여 인증서 파일이 있는 디렉터리로 변경합니다. *sitewise-work*를 *aws.iot.SiteWiseEdgeCollectorOpcua* Greengrass 작업 폴더의 로컬 스토리지 경로로 바꾸고 *source-name*을 데이터 소스의 이름으로 바꿉니다.

기본적으로 Greengrass 작업 폴더는 Linux의 경우 */greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua*이고의 경우 *C:/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua*입니다Microsoft Windows.

```
cd /sitewise-work/source-name/opcua-certificate-store
```

2. 이 소스에 대한 SiteWise Edge 게이트웨이의 OPC UA 클라이언트 인증서가 *aws-iot-opcua-client.pfx* 파일에 있습니다.

다음 명령을 실행하여 인증서를 *aws-iot-opcua-client-certificate.pem*라는 .pem 파일로 내보냅니다

```
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. SiteWise Edge 게이트웨이에서 OPC UA 서버로 인증서 파일(*aws-iot-opcua-client-certificate.pem*)을 전송합니다.

그러려면 SSH 프로토콜을 사용하여 파일을 전송하는 scp 프로그램과 같은 일반적인 소프트웨어를 사용하면 됩니다. 자세한 내용은 Wikipedia에서 [비밀번호 복사](#)를 참조하세요.

Note

SiteWise Edge 게이트웨이가 Amazon Elastic Compute Cloud(Amazon EC2)에서 실행 중이며 SiteWise Edge 게이트웨이에 처음 연결하는 경우, 연결하기 위한 필수 조건을 구성

해야 합니다. 자세한 내용은 Amazon EC2 사용 설명서의 [SSH를 사용하여 Linux 인스턴스에 연결](#)을 참조하세요.

4. SiteWise Edge 게이트웨이를 신뢰하기 위해 OPC UA 서버에서 인증서 파일(aws-iot-opcua-client-certificate.pem)을 가져옵니다. 사용하는 원본 서버에 따라 단계가 다를 수 있습니다. 서버 설명서를 참조하세요.

OPC UA를 사용하여 데이터 수집 범위 필터링

스캔 모드 및 데드밴드 범위를 사용하여 OPC UA 소스로 데이터를 수집하는 방식을 제어할 수 있습니다. 이러한 기능을 통해 수집할 데이터의 종류와 서버 및 SiteWise Edge 게이트웨이가 이 정보를 교환하는 방법과 시기를 제어할 수 있습니다.

품질을 기반으로 데이터 수집 또는 필터링

OPC UA 소스에서 수집되는 데이터를 제어하도록 데이터 품질 설정을 구성할 수 있습니다. 데이터 소스에는 전송 시 품질 등급이 메타데이터로 포함됩니다. 다음 옵션 중 하나 또는 전부를 선택할 수 있습니다.

- Good
- Bad
- Uncertain

NaN 또는 null 값 처리

SiteWise Edge는 NaN 및 null 값의 수집 및 처리를 지원합니다.

- NaN(숫자가 아님): 정의되지 않았거나 대표할 수 없는 숫자 결과를 나타냅니다.
- Null: 누락된 데이터를 나타냅니다.

IoT SiteWise OPC UA 수집기는 BAD 또는 UNCERTAIN 품질로 NaN 및 Null 값을 캡처합니다. 이러한 특수 값은 로컬 스트림에 기록되므로 보다 포괄적인 데이터 수집이 가능합니다.

스캔 모드로 데이터 수집 빈도 제어

OPC UA 스캔 모드를 구성하여 OPC UA 소스에서 데이터를 수집하는 방식을 제어할 수 있습니다. 구독 또는 폴링 모드를 선택할 수 있습니다.

- 구독 모드 – OPC UA 소스는 스캔 속도에 정의된 빈도로 SiteWise Edge 게이트웨이로 보낼 데이터를 수집합니다. 서버는 값이 변경된 경우에만 데이터를 전송하므로 이 빈도는 SiteWise Edge 게이트웨이가 데이터를 수신하는 최대 빈도입니다.
- 폴링 모드 – SiteWise Edge 게이트웨이는 스캔 속도에 정의된 설정된 빈도로 OPC UA 소스를 폴링합니다. 서버는 값 변경 여부에 관계없이 데이터를 전송하므로 SiteWise Edge 게이트웨이는 항상 이 간격으로 데이터를 수신합니다.

Note

폴링 모드 옵션은 이 소스의 데드밴드 설정보다 우선 적용됩니다.

데드밴드 범위를 사용하여 OPC UA 데이터 수집 필터링

OPC UA 소스 속성 그룹에 데드밴드를 적용하여 특정 데이터를 AWS 클라우드로 보내는 대신 필터링하여 폐기할 수 있습니다. 데드밴드는 OPC UA 소스에서 들어오는 데이터 값의 예상 변동 기간을 지정합니다. 값이 이 기간 내에 있으면 OPC UA 서버가 해당 값을 AWS 클라우드로 전송하지 않습니다. 데드밴드 필터링을 사용하여 처리하여 AWS 클라우드로 전송하는 데이터의 양을 줄일 수 있습니다. SiteWise Edge 게이트웨이에 대한 OPC UA 소스를 설정하는 방법은 [AWS IoT SiteWise Edge 게이트웨이용 OPC UA 데이터 소스](#) 섹션을 참조하세요.

Note

서버는 데드밴드에서 지정한 기간 내에 있는 모든 데이터를 삭제합니다. 이러한 폐기된 데이터는 복구할 수 없습니다.

데드밴드 유형

OPC UA 서버 속성 그룹에 두 가지 유형의 데드밴드를 지정할 수 있습니다. 이를 통해 AWS 클라우드로 전송할 데이터의 양과 폐기할 데이터의 양을 선택할 수 있습니다.

- 백분율 — 측정값의 예상 변동 비율을 사용하여 기간을 지정합니다. 서버는 이 백분율에서 정확한 기간을 계산하고 이를 초과하는 데이터를 AWS 클라우드로 전송하여 기간을 벗어납니다. 예를 들어 화씨 -100도에서 화씨 +100도 범위의 센서에 2% 데드밴드 값을 지정하면 값이 화씨 4도 이상 변경될 때 AWS 클라우드로 데이터를 보내도록 서버에 지시합니다.

Note

소스 서버에서 엔지니어링 단위를 정의하지 않는 경우 이 기간의 최소 및 최대 데드밴드 값을 선택적으로 지정할 수 있습니다. 엔지니어링 단위 범위가 제공되지 않은 경우 OPC UA 서버는 측정 데이터 유형의 전체 범위를 기본적으로 사용합니다.

- 절대 — 정확한 단위를 사용하여 기간을 지정합니다. 예를 들어 센서에서 데드밴드 값을 2로 지정하면 값이 2단위 이상 변경될 때 AWS 클라우드에 데이터를 보내도록 서버에 지시합니다. 정상 작동 중에 변동이 정기적으로 예상되는 동적 환경에는 절대 데드밴딩을 사용할 수 있습니다.

데드밴드 타임아웃

데드밴드 타임아웃 설정을 선택적으로 구성할 수 있습니다. 이 타임아웃이 지나면 OPC UA 서버는 예상 데드밴드 변동 범위 내에 있더라도 현재 측정값을 전송합니다. 제한 시간 설정을 사용하면 값이 정의된 데드밴드 기간을 초과하지 않더라도 AWS IoT SiteWise가 항상 안정적인 데이터 스트림을 수집하도록 할 수 있습니다.

SiteWise Edge에서 OPC UA 노드 필터 사용

SiteWise Edge 게이트웨이에 대한 OPC UA 데이터 소스를 정의할 때 노드 필터를 정의할 수 있습니다. 노드 필터를 사용하면 SiteWise Edge 게이트웨이가 클라우드로 보내는 데이터 스트림 경로를 제한할 수 있습니다. 노드 필터를 사용하면 AWS IoT SiteWise에서 모델링하는 데이터에 대한 경로만 포함하여 SiteWise Edge 게이트웨이의 시작 시간과 CPU 사용량을 줄일 수 있습니다. 기본적으로 SiteWise Edge 게이트웨이는 /Server/로 시작하는 경로를 제외한 모든 OPC UA 경로를 업로드합니다. 노드 필터에 * 및 ** 와일드카드 문자를 사용하여 하나의 필터에 여러 데이터 스트림 경로를 포함할 수 있습니다. SiteWise Edge 게이트웨이에 대한 OPC UA 소스를 설정하는 방법은 [AWS IoT SiteWise Edge 게이트웨이용 OPC UA 데이터 소스](#) 섹션을 참조하세요.

Note

AWS IoT SiteWise는 소스를 추가하거나 편집할 때마다 SiteWise Edge 게이트웨이를 다시 시작합니다. SiteWise Edge 게이트웨이는 소스 구성을 업데이트하는 동안 데이터를 수집하지 않습니다. SiteWise Edge 게이트웨이를 다시 시작하는 시간은 SiteWise Edge 게이트웨이 소스의 태그 수에 따라 달라집니다. 다시 시작 시간은 몇 초(태그가 거의 없는 SiteWise Edge 게이트웨이)에서 몇 분(태그가 많은 SiteWise Edge 게이트웨이)에 이르기까지 다양할 수 있습니다.

다음 표에는 OPC UA 데이터 소스를 필터링하는 데 사용할 수 있는 와일드카드가 나열되어 있습니다.

OPC UA 노드 필터 와일드카드

와일드카드	설명
*	데이터 스트림 경로의 단일 레벨과 일치합니다.
**	데이터 스트림 경로의 여러 레벨을 일치합니다.

Note

광범위한 필터로 소스를 구성한 다음 나중에 더 제한적인 필터를 사용하도록 소스를 변경하면 새 필터와 일치하지 않는 데이터 저장을 AWS IoT SiteWise 중지합니다.

Example : 노드 필터를 사용한 시나리오

다음의 가상 데이터 스트림을 고려하세요.

- /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

이전 데이터 스트림을 사용해 노드 필터를 정의하여 OPC UA 소스에서 포함할 데이터를 제한할 수 있습니다.

- 이 예제에서 모든 노드를 선택하려면 / 또는 /**/를 사용합니다. ** 와일드카드 문자가 있는 여러 디렉터리 또는 폴더를 포함할 수 있습니다.
- 모든 PLC 데이터 스트림을 선택하려면 /*/*/PLC* 또는 /**/PLC*을 사용합니다.
- 이 예제에서 모든 카운터를 선택하려면 /**/Counter* 또는 /*/*/Counter*를 사용합니다.
- Line 2의 모든 카운터를 선택하려면 /**/Line 2/Counter*를 사용합니다.

지원되지 않는 데이터 유형 변환

선택적으로 단순 배열 및 DateTime 데이터 형식에 AWS IoT SiteWise 대해에서 데이터 형식 변환을 활성화합니다. AWS IoT SiteWise 일부 OPC UA 데이터 형식은 지원하지 않습니다. 지원되지 않는 데이터를 AWS IoT Greengrass 데이터 스트림으로 보내면 해당 데이터가 손실됩니다. 그러나 지원되지 않는 기본 데이터 형식을 문자열로 변환하면 데이터를 삭제 AWS IoT SiteWise 하지 않고 로 수집할 수 있습니다.는 나중에 필요한 경우 자체 함수를 사용하여 문자열을 다시 원래 데이터 유형 다운스트림으로 변환할 수 있도록 변환된 데이터를 AWS IoT SiteWise 직렬화합니다.

언제든지 데이터 소스에 대한 데이터 유형 변환 설정을 업데이트할 수 있으며 각 데이터 소스에는 고유한 설정이 있을 수 있습니다.

에 데이터 소스를 추가할 때 고급 구성의 데이터 유형 변환 아래에 두 개의 확인란 AWS IoT SiteWise Console이 있습니다. 문자열로 변환할 데이터 유형을 지정할 수 있습니다.

또한 IoT SiteWise OPC UA 수집기는 옛지에서 NaN 또는 null 값을 수락할 수 있습니다.

- 간단한 데이터 유형의 배열 값을 JSON 문자열로 변환
- DateTime 값을 ISO 8601 문자열로 변환

전제 조건

- [IoT SiteWise OPC UA 수집기](#)의 버전 2.5.0 이상을 사용합니다.

제한 사항

다음은 AWS IoT SiteWise에서 OPC UA 데이터 유형을 문자열로 변환하는 데 따르는 제한 사항입니다.

- 복잡한 데이터 유형 변환은 지원되지 않습니다.
- 변환 후 문자열 제한은 1,024바이트입니다. 문자열이 1024바이트보다 길면가 문자열을 거부합니다 AWS IoT SiteWise.

SiteWise Edge에 대한 데이터 소스 인증 구성

OPC UA 서버에 연결하기 위해 인증 자격 증명이 필요한 경우 AWS Secrets Manager 를 사용하여 SiteWise Edge 게이트웨이에 보안 암호를 생성하고 배포할 수 있습니다.는 디바이스의 보안 암호를 AWS Secrets Manager 암호화하여 사용자 이름과 암호를 사용해야 할 때까지 안전하게 유지합니

다. AWS IoT Greengrass 보안 암호 관리자 구성 요소에 대한 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [보안 암호 관리자](#)를 참조하세요.

Secrets Manager 보안 암호에 대한 액세스 관리에 대한 자세한 내용은 다음을 참조하세요.

- [AWS Secrets Manager 보안 암호에 대한 권한이 있는 사용자.](#)
- [계정 내에서 요청이 허용 또는 거부되는지 여부를 결정합니다.](#)

1단계: 소스 인증 암호 만들기

AWS Secrets Manager 를 사용하여 데이터 소스에 대한 인증 암호를 생성할 수 있습니다. 암호에서 데이터 소스에 대한 인증 세부 정보를 포함하는 **username** 및 **password** 키-값 쌍을 정의합니다.

보안 암호(콘솔) 생성

1. [AWS Secrets Manager 콘솔](#)로 이동합니다.
2. 새 암호 저장을 선택합니다.
3. 암호 유형에서 다른 암호 유형을 선택합니다.
4. 키/값 페어에서 다음을 수행합니다.
 1. 첫 번째 입력 상자에 **username**을 입력하고 두 번째 입력 상자에 사용자 이름을 입력합니다.
 2. Add row(행 추가)를 선택합니다.
 3. 첫 번째 입력 상자에 **password**를 입력하고 두 번째 입력 상자에 암호를 입력합니다.
5. 암호화 키로 aws/secretsmanager를 선택하고 다음을 선택합니다.
6. 새 암호 저장 페이지에서 암호 이름을 입력합니다.
7. (선택 사항) 이 암호를 식별하는 데 도움이 되는 설명을 입력하고 다음을 선택합니다.
8. (선택 사항) 새 암호 저장 페이지에서 자동 교체를 켤 수 있습니다. 자세한 내용은 AWS Secrets Manager 사용 설명서의 [비밀번호 교체](#)를 참조하세요.
9. 교체 일정을 지정합니다.
10. 이 암호를 교체할 수 있는 Lambda 함수를 선택한 후 다음을 선택합니다.
11. 보안 구성을 검토한 다음 저장을 선택합니다.

SiteWise Edge 게이트웨이가 상호 작용할 수 있도록 권한을 부여하려면 SiteWise Edge 게이트웨이 AWS Secrets Manager의 IAM 역할이 secretsmanager:GetSecretValue 작업을 허용해야 합니다.

다. Greengrass 코어 디바이스를 사용하여 IAM 정책을 검색할 수 있습니다. IAM 정책 업데이트에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서의 [IAM 정책 편집](#)을 참조하세요.

Example 정책

*secret-arn*을 이전 단계에서 생성한 암호의 Amazon 리소스 이름(ARN)으로 변경합니다. 보안 암호의 ARN을 가져오는 방법에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 [에서 보안 암호 찾기 AWS Secrets Manager](#)를 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "secret-arn"
      ]
    }
  ]
}
```

2단계: SiteWise Edge 게이트웨이 디바이스에 암호 배포

AWS IoT SiteWise 콘솔을 사용하여 SiteWise Edge 게이트웨이에 보안 암호를 배포할 수 있습니다.

암호를 업데이트하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 게이트웨이를 선택합니다.
3. 게이트웨이 목록에서 대상 SiteWise Edge 게이트웨이를 선택합니다.
4. 게이트웨이 구성 섹션에서 Greengrass 코어 디바이스 링크를 선택하여 SiteWise Edge 게이트웨이와 연결된 AWS IoT Greengrass 코어를 엽니다.
5. 탐색 창에서 배포를 선택합니다.
6. 대상 배포를 선택한 다음 수정을 선택합니다.
7. 대상 지정 페이지에서 다음을 선택합니다.
8. 구성 요소 선택 페이지의 공용 구성 요소 섹션에서 선택한 구성 요소만 표시를 끕니다.

9. AWS.Greengrass.SecretManager 구성 요소를 검색하여 선택한 후 다음을 선택합니다.
10. 선택한 구성 요소 목록에서 aws.greengrass.SecretManager 구성 요소를 선택한 다음 구성 요소 구성을 선택합니다.
11. 병합할 구성 필드에 다음 JSON 객체를 추가합니다.

 Note

*secret-arn*을 이전 단계에서 생성한 암호의 ARN으로 변경합니다. 보안 암호의 ARN을 가져오는 방법에 대한 자세한 내용은 AWS Secrets Manager 사용 설명서의 [에서 보안 암호 찾기 AWS Secrets Manager](#)를 참조하세요.

```
{
  "cloudSecrets":[
    {
      "arn":"secret-arn"
    }
  ]
}
```

12. 확인을 선택합니다.
13. Next(다음)를 선택합니다.
14. 구성 고급 설정 페이지에서 다음을 선택합니다.
15. 배포 구성을 검토한 다음 배포를 선택합니다.

3단계: 인증 구성 추가

AWS IoT SiteWise 콘솔을 사용하여 SiteWise Edge 게이트웨이에 인증 구성을 추가할 수 있습니다.

인증 구성을 추가하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 게이트웨이 목록에서 대상 SiteWise Edge 게이트웨이를 선택합니다.
3. 데이터 소스 목록에서 대상 데이터 소스를 선택한 다음 편집을 선택합니다.
4. 데이터 소스 추가 페이지에서 고급 구성을 선택합니다.
5. 인증 구성의 경우 이전 단계에서 배포한 암호를 선택합니다.

6. 저장(Save)을 선택합니다.

SiteWise Edge 게이트웨이의 파트너 데이터 소스

Edge 게이트웨이를 사용하는 경우 파트너 데이터 소스를 SiteWise AWS IoT SiteWise Edge 게이트웨이에 연결하고 SiteWise Edge 게이트웨이 및 AWS 클라우드의 파트너로부터 데이터를 수신할 수 있습니다. 이러한 파트너 데이터 소스는 AWS와 파트너 간의 파트너십을 통해 개발된 AWS IoT Greengrass 구성 요소입니다. 파트너 데이터 소스를 추가하면 AWS IoT SiteWise가 이 구성 요소를 생성하여 SiteWise Edge 게이트웨이에 배포합니다.

Note

각 게이트웨이의 각 파트너에 대해 하나의 데이터 소스를 추가할 수 있습니다.

파트너 데이터 소스를 추가하려면 다음을 수행합니다.

1. [SiteWise Edge에 파트너 데이터 소스 추가](#)
2. 파트너의 웹 포털로 이동하여 SiteWise Edge 게이트웨이에 연결되도록 파트너 데이터 소스를 구성합니다.

주제

- [보안](#)
- [SiteWise Edge 게이트웨이Docker에 설정](#)
- [SiteWise Edge에 파트너 데이터 소스 추가](#)
- [SiteWise Edge 게이트웨이 파트너 데이터 소스 옵션](#)

보안

AWS, 고객 및 파트너 간의 [공동 책임 모델의](#) 일환으로 다음과 같이 보안의 다양한 측면을 담당하는 사람을 설명합니다.

고객의 책임

- 파트너를 심사합니다.
- 파트너에게 부여된 네트워크 액세스를 구성합니다.

- SiteWise Edge 게이트웨이 머신 리소스(CPU, 메모리 및 파일 시스템)의 합리적인 사용 모니터링.

AWS 책임

- 파트너가 필요로 하는 리소스를 제외한 고객 AWS 클라우드 리소스에서 파트너를 격리합니다. 이 경우에는 AWS IoT SiteWise 수집입니다.
- 파트너 솔루션을 SiteWise Edge 게이트웨이 컴퓨터 리소스(CPU 및 메모리)의 합리적인 사용으로 제한.

파트너의 책임

- 보안 기본값을 사용합니다.
- 패치 및 기타 적절한 업데이트를 통해 시간이 지나도 솔루션을 안전하게 유지합니다.
- 고객 데이터의 기밀을 유지합니다.

SiteWise Edge 게이트웨이 Docker에 설정

AWS IoT SiteWise 는 다양한 플랫폼 및 환경에서 SiteWise Edge 애플리케이션을 실행할 수 있는 Docker 이미지를 제공합니다. 이 Docker 이미지는 산업 장비에서 AWS 클라우드로 데이터를 수집, 처리 및 전송하는 데 필요한 모든 구성 요소와 종속성을 캡슐화합니다. Docker 이미지를 사용하면 서버, 엣지 디바이스 또는 클라우드 기반 컨테이너 서비스와 같은 Docker 호환 호스트에서 SiteWise Edge 애플리케이션을 배포하고 실행할 수 있습니다.

파트너 데이터 소스를 추가하려면 로컬 디바이스에 [Docker Engine](#) 1.9.1 이상을 설치해야 합니다.

Note

버전 20.10은 SiteWise Edge 게이트웨이 소프트웨어와 함께 작동하는 것으로 확인된 최신 버전입니다.

Docker가 설치되었는지 확인

Docker가 설치되었는지 확인하려면 SiteWise Edge 게이트웨이에 연결된 터미널에서 다음 명령을 실행합니다.

```
docker info
```

명령이 `docker is not recognized` 결과를 반환하거나 이전 버전의 Docker가 설치된 경우 계속하기 전에 [Docker Engine](#)을 설치합니다.

Docker 설정

Docker 컨테이너 구성 요소를 실행하는 시스템 사용자에게 루트 또는 관리자 권한이 있거나 루트가 아닌 사용자 또는 관리자가 아닌 사용자로 실행 Docker하도록 구성해야 합니다.

Linux 디바이스에서는 docker 그룹에 ggc_user 사용자를 추가하여 없이 Docker 명령을 호출해야 합니다 sudo.

또는 Docker 컨테이너 구성 요소를 실행하는 데 사용하는 루트가 ggc_user 아닌 사용자를 docker 그룹에 추가하려면 다음 명령을 실행합니다.

```
sudo usermod -aG docker ggc_user
```

자세한 내용은 [Docker 엔진의 Linux 설치 후 단계를 참조하세요](#).

SiteWise Edge에 파트너 데이터 소스 추가

파트너 데이터 소스를 SiteWise Edge 게이트웨이에 연결하려면 이를 데이터 소스로 추가합니다. 데이터 소스로 추가하면 AWS IoT SiteWise 가 SiteWise Edge 게이트웨이에 프라이빗 AWS IoT Greengrass 구성 요소를 배포합니다.

사전 조건

파트너 데이터 소스를 추가하려면 다음을 수행해야 합니다.

- EasyEdge 및 CloudRail의 경우 파트너와 계정을 생성한 다음 계정을 결합합니다.
- [SiteWise Edge 게이트웨이 Docker에 설정](#)

파트너 데이터 소스를 사용하여 SiteWise Edge 게이트웨이 생성

새 SiteWise Edge 게이트웨이를 만들려면 [자체 호스팅 SiteWise Edge 게이트웨이 생성](#)의 단계를 완료하세요. SiteWise Edge 게이트웨이를 만든 후에는 [기존 SiteWise Edge 게이트웨이에 파트너 데이터 소스 추가](#)의 단계에 따라 파트너 데이터 소스를 추가하세요.

기존 SiteWise Edge 게이트웨이에 파트너 데이터 소스 추가

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 파트너 데이터 소스를 연결하려는 SiteWise Edge 게이트웨이를 선택합니다.

4. 데이터 소스에서 데이터 소스 추가를 선택합니다.
5. 데이터 소스 추가 화면에서 소스 유형을 선택하여 SiteWise Edge 게이트웨이를 연결하는 파트너를 선택합니다. 각 데이터 소스에는 고유한 구성 옵션이 있습니다. 데이터 소스에는 AWS 소스와 파트너 소스의 두 가지 카테고리가 있습니다.

파트너 데이터 소스를 사용하여 게이트웨이당 하나의 소스를 선택할 수 있습니다. 데이터 소스 파트너 통합 옵션 목록은 [SiteWise Edge 게이트웨이 파트너 데이터 소스 옵션](#) 섹션을 참조하세요. 최대 100개의 OPC UA 데이터 소스(AWS 소스)를 추가할 수 있습니다. OPC UA 데이터 소스를 시작하려면 [AWS IoT SiteWise Edge 게이트웨이용 OPC UA 데이터 소스](#) 섹션을 참조하세요.

6. 소스 이름을 입력합니다.
7. 아래에서 데이터 소스의 탭을 선택하고 구성 절차를 따릅니다.

CloudRail

대부분의 CloudRail 구성은 SiteWise Edge 게이트웨이의 데이터 소스를 저장한 후 CloudRail 포털에서 수행됩니다. 그러나 연결에 권한을 부여해야 합니다.

Note

CloudRail 연결은 Linux에서만 사용할 수 있습니다.

1. [CloudRail 계정을 생성](#)하여 AWS IoT SiteWise에 대한 연결을 시작합니다.
2. Docker가 게이트웨이에 설치되어 있는지 확인합니다. 자세한 내용은 [SiteWise Edge 게이트웨이Docker에 설정](#) 단원을 참조하십시오.
3. 액세스 및 배포 권한 부여 계약을 읽은 다음 권한 부여를 선택합니다. 확인란을 선택하면 AWS 파트너에게 데이터 소스에 대한 액세스 권한이 부여되고 AWS가 파트너의 구성 요소에 배포할 수 있습니다.

Note

CloudRail 포털 내에 측정 접두사 – 선택 사항이 설정됩니다.

Note

파트너 소프트웨어는 AWS 파트너가 개발, 유지 관리 및 지원합니다. AWS 는 인터페이스, 구성 또는 소프트웨어에 대해 책임을 지지 않습니다.

자세한 내용은 [CloudRail](#) 단원을 참조하십시오.

EasyEdge

대부분의 EasyEdge 구성은 SiteWise Edge 게이트웨이의 데이터 소스를 저장한 후 EasyEdge 포털에서 수행됩니다. 그러나 연결에 권한을 부여해야 합니다.

Note

EasyEdge 연결은 Linux에서만 사용할 수 있습니다.

1. [EasyEdge 계정을 생성](#)하여 AWS IoT SiteWise에 대한 연결을 시작합니다.
2. Docker가 게이트웨이에 설치되어 있는지 확인합니다. 자세한 내용은 [SiteWise Edge 게이트웨이Docker에 설정](#) 단원을 참조하십시오.
3. 액세스 및 배포 권한 부여 계약을 읽은 다음 권한 부여를 선택합니다. 확인란을 선택하면 AWS 파트너에게 데이터 소스에 대한 액세스 권한이 부여되고 AWS 가 파트너의 구성 요소에 배포할 수 있습니다.

Note

EasyEdge 포털 내에 측정 접두사 – 선택 사항이 설정됩니다.

Note

파트너 소프트웨어는 AWS 파트너가 개발, 유지 관리 및 지원합니다. AWS 는 인터페이스, 구성 또는 소프트웨어에 대해 책임을 지지 않습니다.

자세한 내용은 [EasyEdge](#) 단원을 참조하십시오.

Litmus Edge

두 가지 방법으로 Litmus 구성을 활성화할 수 있습니다. Litmus Edge Manager 포털의 정보를 사용하여 AWS IoT SiteWise 를 통해 Litmus Edge를 직접 활성화합니다. 또는 Litmus Edge Manager를 통해 AWS IoT SiteWise 에 대해 Litmus Edge를 수동으로 활성화할 수 있습니다.

Note

Litmus Edge 연결은 Linux에서만 사용할 수 있습니다.

에서 활성화 코드를 사용하여 Litmus Edge를 활성화하려면 AWS IoT SiteWise

AWS IoT SiteWise Console에 Litmus Edge 활성화 코드가 있는 Litmus Edge 데이터 소스를 추가할 때 이 절차를 사용합니다.

1. 코드를 사용하여 지금 활성화를 선택합니다. 추가 구성 옵션이 나타납니다.
2. Litmus Edge를 SiteWise Edge 게이트웨이에 연결하려면 Litmus Edge Manager로 들어갑니다. 자세한 내용은 Litmus Edge Manager 설명서의 [3a단계: 데이터 및 디바이스 관리 엔드포인트 설정](#)을 참조하세요.
3. Litmus Edge Manager 활성화할 활성화 코드 제공 Litmus Edge AWS IoT SiteWise
4. 선택적으로 Litmus Edge Manager CA 인증서와 AWS IoT SiteWise 함께를 제공합니다. 인증서는 Litmus Edge가 권한 부여되지 않은 Litmus Edge Manager에서 활성화되지 않도록 합니다.
5. Docker가 게이트웨이에 설치되어 있는지 확인합니다. 자세한 내용은 [SiteWise Edge 게이트웨이Docker에 설정](#) 단원을 참조하십시오.

Note

AWS IoT SiteWise 는 파트너 애플리케이션을 Docker 컨테이너로 배포합니다. 애플리케이션은 Litmus Edge Manager를 통해 Litmus Edge Docker 컨테이너를 관리할 수 있도록 NET_ADMIN 기능과 함께 배포됩니다. Litmus Edge는 디바이스에서 실행하기 위해 이 권한 있는 액세스를 필요로 합니다. Litmus Edge Docker

요구 사항에 대한 자세한 내용은 Litmus Edge 설명서의 QuickStart 가이드에서 [Docker 설치](#)를 참조하세요.

6. 액세스 및 배포 권한 부여 계약을 읽은 다음 권한 부여를 선택합니다. 확인란을 선택하면 AWS 파트너에게 데이터 소스에 대한 액세스 권한이 부여되고 AWS가 파트너의 구성 요소에 배포할 수 있습니다.

Litmus Edge를 통해 수동으로 활성화하려면 다음을 수행하세요.

1. 나중에 Litmus Edge에서 활성화를 선택합니다.
2. Docker가 게이트웨이에 설치되어 있는지 확인합니다. 자세한 내용은 [SiteWise Edge 게이트웨이Docker에 설정](#) 단원을 참조하십시오.

Note

AWS IoT SiteWise는 파트너 애플리케이션을 Docker 컨테이너로 배포합니다. 애플리케이션은 Litmus Edge Manager를 통해 Litmus Edge Docker 컨테이너를 관리할 수 있도록 NET_ADMIN 기능과 함께 배포됩니다. Litmus Edge는 디바이스에서 실행하기 위해 이 권한 있는 액세스를 필요로 합니다. Litmus Edge Docker 요구 사항에 대한 자세한 내용은 Litmus Edge 설명서의 QuickStart 가이드에서 [Docker 설치](#)를 참조하세요.

3. 액세스 및 배포 권한 부여 계약을 읽은 다음 권한 부여를 선택합니다. 확인란을 선택하면 AWS 파트너에게 데이터 소스에 대한 액세스 권한이 부여되고 AWS가 파트너의 구성 요소에 배포할 수 있습니다.
4. 배포가 완료되면 Litmus Edge QuickStart Guide 설명서의 [Litmus Edge 웹 UI 액세스](#) 지침을 따릅니다.

Note

파트너 소프트웨어는 AWS 파트너가 개발, 유지 관리 및 지원합니다. AWS는 인터페이스, 구성 또는 소프트웨어에 대해 책임을 지지 않습니다.

자세한 내용은 [Litmus Edge](#) 단원을 참조하십시오.

8. 저장을 선택합니다.

SiteWise Edge 게이트웨이 파트너 데이터 소스 옵션

AWS IoT SiteWise 를 사용하면 산업 장비, 센서 및 기타 타사 시스템과 같은 다양한 파트너 데이터 소스에서 데이터를 연결하고 수집할 수 있습니다. 파트너 데이터 소스를 연결하려면 데이터를 전송할 데이터 소스 구성, 필요한 권한 및 인증 AWS IoT SiteWise 설정, 자산 모델에 데이터 매핑 등 몇 가지 단계를 따라야 합니다. 이 프로세스를 통해 파트너 데이터가 AWS IoT SiteWise 환경에 원활하게 통합되어 다른 데이터 소스와 함께 모니터링하고 분석할 수 있습니다.

이 섹션에서는 SiteWise Edge 게이트웨이에서 타사 데이터 소스 통합에 사용할 수 있는 파트너를 나열합니다. 아래 정보를 사용하여 파트너 데이터 소스를 구성하세요.

Note

각 게이트웨이의 각 파트너에 대해 하나의 데이터 소스를 추가할 수 있습니다.

CloudRail

포털:

<https://devices.cloudrail.com/>

요구 사항

CloudRail 요구 사항에 대한 자세한 내용은 CloudRail 웹 사이트의 [FAQ](#)를 참조하세요.

CloudRail 설명서:

[엣지 컴퓨팅: SiteWise Edge](#)

EasyEdge

포털:

<https://studio.easyedge.io/>

요구 사항

[EasyEdge 요구 사항](#) - 방화벽 구성에 필요한 엔드포인트 및 포트를 포함한 EasyEdge 요구 사항에 대한 정보입니다. 참고: 이 설명서에 액세스하려면 EasyEdge 계정이 필요합니다.

EasyEdge 설명서:

[EasyEdge\(AWS일 때\)](#)

Litmus Edge

Litmus Edge Manager에 액세스:

Litmus Edge에 액세스하려면 [Litmus Edge Manager 계정](#)을 설정합니다.

요구 사항

[Litmus Edge 요구 사항](#) - Litmus Edge 배포를 위한 권장 구성 및 시스템 요구 사항입니다.

Litmus 설명서:

- [에 통합 AWS IoT SiteWise](#)
- [Litmus Edge 설명서](#)

AWS IoT GreengrassAWS IoT SiteWise Edge용 구성 요소

SiteWise Edge는 AWS IoT Greengrass 구성 요소를 사용하여 엣지에서 산업 데이터를 수집, 처리 및 전송합니다. 이러한 구성 요소는 함께 작동하여 로컬 데이터 처리를 지원하고 AWS IoT SiteWise 클라우드 서비스와 원활하게 통합할 수 있습니다.

IoT SiteWise 게시자

IoT SiteWise 게시자 구성 요소(`aws.iot.SiteWiseEdgePublisher`)는 다음을 담당합니다.

- 수집된 데이터를 AWS IoT SiteWise 클라우드 서비스로 안전하게 전송
- 연결 문제 중 데이터 버퍼링 및 재시도 관리

SiteWise Edge용 게시자 구성에 대한 자세한 내용은 섹션을 참조하세요 [AWS IoT SiteWise 게시자 구성 요소 구성](#). 게시자 구성 요소에 대한 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [IoT SiteWise 게시자](#)를 참조하세요.

IoT SiteWise 프로세서

IoT SiteWise 프로세서 구성 요소(`aws.iot.SiteWiseEdgeProcessor`)는 다음 작업을 수행합니다.

- 엣지에서 데이터 변환 및 계산 실행
- 로컬에서 자산 속성 정의 및 계산 구현

- 전송 전에 데이터를 집계하거나 필터링하여 데이터 볼륨 줄이기

프로세서 구성 요소에 대한 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [IoT SiteWise 프로세서](#)를 참조하세요.

IoT SiteWise OPC UA 수집기

IoT SiteWise OPC UA 수집기(`aws.iot.SiteWiseEdgeCollectorOpcua`) 구성 요소는 다음과 같이 설계되었습니다.

- 산업 환경의 OPC UA 서버에 연결
- OPC UA 데이터 소스에서 효율적으로 데이터 수집
- OPC UA 데이터를와 호환되는 형식으로 변환 AWS IoT SiteWise

OPC UA 수집기 구성 요소에 대한 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [IoT SiteWise OPC UA 수집기](#)를 참조하세요.

IoT SiteWise OPC UA 데이터 소스 시뮬레이터

IoT SiteWise OPC UA 데이터 소스 시뮬레이터 구성 요소(`aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator`)는 다음 기능을 제공합니다.

- 샘플 데이터를 생성하는 로컬 OPC UA 서버를 시작합니다.
- AWS IoT SiteWise 게이트웨이의 AWS IoT SiteWise OPC UA 수집기 구성 요소에서 읽을 수 있는 데이터 소스를 시뮬레이션합니다.
- 생성된 샘플 데이터를 사용하여 AWS IoT SiteWise 기능을 탐색할 수 있습니다.

이 구성 요소는 테스트 및 개발 목적에 특히 유용하므로 물리적 장비 없이 산업 데이터 소스를 시뮬레이션할 수 있습니다.

데이터 소스 시뮬레이션 구성 요소에 대한 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [IoT SiteWise OPC UA 데이터 소스 시뮬레이터](#)를 참조하세요.

이러한 AWS IoT Greengrass 구성 요소는 SiteWise Edge 기능을 활성화하는 데 사용됩니다. IoT SiteWise 게시자는 데이터가 클라우드로 안정적으로 전송되도록 보장하고, IoT SiteWise 프로세서는 로컬 계산 및 데이터 최적화를 처리하며, IoT SiteWise OPC UA 수집기는 일반적인 산업 프로토콜과의 통합을 용이하게 합니다.

Note

이러한 구성 요소를 사용하려면 엣지 디바이스에 AWS IoT Greengrass V2 이상이 설치되어 있어야 합니다. SiteWise Edge의 성능을 최적화하려면 각 구성 요소의 적절한 구성이 중요합니다.

SiteWise Edge 게이트웨이에서 자산 필터링

엣지 필터링을 사용하면 데이터 처리에 사용하기 위한 자산의 일부만 특정 SiteWise Edge 게이트웨이로 전송하여 자산을 보다 효율적으로 관리할 수 있습니다. 자산이 트리 또는 상위-하위 구조로 정렬되어 있는 경우 SiteWise Edge 게이트웨이의 IAM 역할에 연결된 IAM 정책을 설정하여 트리의 루트, 즉 상위 항목 및 해당 하위 항목만 특정 SiteWise Edge 게이트웨이로 전송하도록 허용할 수 있습니다.

Note

기존 자산을 트리 구조로 정렬하는 경우 구조를 생성한 후 구조에 추가한 각 기존 자산으로 이동하여 편집을 선택한 다음 저장을 선택하여가 새 구조를 AWS IoT SiteWise 인식하는지 확인합니다.

엣지 필터링 설정

다음 IAM 정책을 SiteWise Edge 게이트웨이의 IAM 역할에 추가하고 *<root-asset-id>*를 SiteWise Edge 게이트웨이로 전송하려는 루트 자산의 ID로 대체하여 SiteWise Edge 게이트웨이에 엣지 필터링을 설정합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "iotsitewise:DescribeAsset",
        "iotsitewise:ListAssociatedAssets"
      ],
      "Resource": "arn:aws:iotsitewise:*:*:asset/*",
      "Condition": {
        "StringNotLike": {
```

```

        "iotsitewise:assetHierarchyPath": "/<root-asset-id>*"
      }
    }
  }
]
}

```

현재 SiteWise Edge 게이트웨이에 제거하려는 자산이 있는 경우 SiteWise Edge 게이트웨이에 로그인하고 다음 명령을 실행하여 캐시를 삭제 AWS IoT SiteWise 하여 SiteWise Edge 게이트웨이가와 동기화되도록 합니다.

```

sudo rm /greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/sync-app/
sync_resource_bundles/edge.json

```

프록시 지원 구성 및 AWS IoT SiteWise Edge용 트러스트 스토어 관리

AWS IoT SiteWise 엣지에서 트러스트 스토어를 구성하고 관리하여 엣지 디바이스에 대한 프록시 지원을 설정합니다. 먼저 프록시 구성을 설정한 다음 트러스트 스토어를 구성합니다. 게이트웨이 설치 중에 또는 게이트웨이가 설정된 후 수동으로 트러스트 스토어를 구성할 수 있습니다.

- 프록시 - 다양한 네트워크 환경에서 엣지 디바이스와 AWS 서비스 간의 연결을 촉진합니다.
- 트러스트 스토어 - 신뢰할 수 있는 인증서를 관리하여 보안 연결을 보장합니다. 적절한 구성을 사용하면 네트워크 보안 정책을 준수하고, 제한된 네트워크 환경에서 통신을 활성화하고, 엣지 디바이스와 클라우드 서비스 간의 데이터 전송을 최적화할 수 있습니다.

SiteWise Edge는 다양한 구성 요소 유형에 대해 여러 트러스트 스토어를 활용하여 엣지 디바이스에서 클라우드로 안전하고 효율적인 데이터 흐름을 보장합니다. 기존 게이트웨이에서 또는 새 게이트웨이를 생성할 때 설치 프로세스 중에 트러스트 스토어와 프록시를 구성할 수 있습니다.

트러스트 스토어 및 프록시 구성 요구 사항

트러스트 스토어를 구성하거나 프록시 설정으로 SiteWise Edge를 설치하기 전에 사전 조건을 충족하는지 확인합니다. 구성 요소 사용 및 기능 요구 사항에 따라 다양한 구현 요구 사항이 있습니다.

프록시 지원 요구 사항

- 프록시 서버의 URL입니다. URL에는 호스트의 포트 번호인 사용자 정보가 포함되어야 합니다. 예를 들어 `scheme://[userinfo@]host[:port]`입니다.
 - scheme - HTTP 또는 HTTPS여야 합니다.

- (선택 사항) userinfo - 사용자 이름 및 암호 정보
- host - 프록시 서버의 호스트 이름 또는 IP 주소
- port - 포트 번호
- 프록시를 우회할 주소 목록입니다.
- (선택 사항) 자체 서명된 인증서와 함께 HTTPS 프록시를 사용하는 경우 프록시 CA 인증서 파일입니다.

트러스트 스토어 요구 사항

- HTTPS 프록시를 사용하여 전체 데이터 처리 팩 기능을 사용하려면 세 개의 트러스트 스토어를 모두 업데이트해야 합니다.
- IoT SiteWise OPC UA 수집기 및 IoT SiteWise 게시자만 사용하는 경우 AWS IoT Greengrass 코어 및 Java 신뢰 스토어 인증서를 최신 버전으로 업데이트합니다.

트러스트 스토어 및 프록시 서버 엣지 구성 모범 사례

지속적인 유지 관리와 엣지 환경에서 최고 수준의 보안을 유지하려면:

- 네트워크 보안 요구 사항에 맞게 프록시 설정을 정기적으로 검토하고 업데이트합니다.
- 게이트웨이 연결 및 데이터 흐름을 모니터링하여 적절한 프록시 통신을 보장합니다.
- 조직의 인증서 관리 정책에 따라 트러스트 스토어 유지 관리 및 업데이트
- 다음과 같은 엣지 환경에서 보안 통신을 위한 권장 모범 사례를 구현하고 따를 수 있습니다.
- 운영 가시성을 위해 프록시 및 트러스트 스토어 구성 문서화
- 자격 증명 관리에 대한 조직의 보안 관행을 따릅니다.

이러한 관행은 광범위한 보안 정책에 따라 SiteWise Edge 게이트웨이에 대한 안전하고 신뢰할 수 있는 작업을 유지하는 데 도움이 됩니다.

AWS IoT SiteWise Edge 게이트웨이 설치 중 프록시 설정 구성

게이트웨이 설치 중에 프록시 서버와 함께 작동하도록 AWS IoT SiteWise Edge를 구성할 수 있습니다. 설치 스크립트는 HTTP 및 HTTPS 프록시를 모두 지원하며 보안 프록시 연결을 위해 트러스트 스토어를 자동으로 구성할 수 있습니다.

프록시 설정으로 설치 스크립트를 실행하면 다음과 같은 몇 가지 중요한 작업이 수행됩니다.

- 프록시 URL 형식 및 파라미터를 검증하여 올바르게 지정되었는지 확인합니다.
- 구성된 프록시를 통해 필요한 종속성을 다운로드하고 설치합니다.
- 프록시 CA 인증서가 제공된 경우 AWS IoT Greengrass 루트 CA 인증서에 추가되고 Java KeyStore 로 가져옵니다.
- 모든 아웃바운드 연결에 프록시를 사용하도록 AWS IoT Greengrass (SiteWise Edge가 사용하는)를 구성합니다.
- 적절한 프록시 및 트러스트 스토어 구성을 사용하여 SiteWise Edge 설치를 완료합니다.

게이트웨이 소프트웨어를 설치할 때 프록시 설정을 구성하려면

1. SiteWise Edge 게이트웨이를 생성합니다. 자세한 내용은 [자체 호스팅 SiteWise Edge 게이트웨이 생성](#) 및 [로컬 디바이스에 AWS IoT SiteWise Edge 게이트웨이 소프트웨어 설치](#) 섹션을 참조하세요.
2. 환경에 적합한 프록시 설정으로 설치 스크립트를 실행합니다. 자리 표시자를 특정 프록시 정보로 바꿉니다.

다음 각 항목을 바꿉니다.

- -p, --proxy-url- 프록시 서버의 URL입니다. URL은 http 또는 중 하나여야 합니다https.
- -n, --no-proxy - 프록시를 우회하기 위해 쉼표로 구분된 주소 목록입니다.
- (선택 사항) -c--proxy-ca-cert- 프록시 CA 인증서 파일의 경로입니다.
- (선택 사항) -j--javastorepass- Java KeyStore 암호입니다. 기본 암호는 changeit입니다.

Linux

Linux 시스템의 경우 다음 명령 구조를 사용합니다.

```
sudo ./install.sh -p proxy-url -n no-proxy-addresses [-c proxy-ca-cert-path] [-j javastorepass]
```

Windows

PowerShell을 사용하는 Microsoft Windows 시스템의 경우 다음 명령 구조를 사용합니다.

```
.\install.ps1 -ProxyUrl proxy-url -NoProxyAddresses no-proxy-addresses [-ProxyCaCertPath proxy-ca-cert-path] [-JavaStorePass javastorepass]
```

프록시 지원 설치 중 문제 해결

SiteWise Edge 게이트웨이와 관련된 트러스트 스토어 문제를 해결하는 방법에 대한 자세한 내용은 섹션을 참조하세요 [프록시 지원 설치 문제](#).

Edge에서 HTTPS 프록시 지원을 위한 AWS IoT SiteWise 트러스트 스토어 수동 구성

HTTPS 프록시를 통해 연결하도록 AWS IoT SiteWise Edge 구성 요소를 구성할 때 프록시 서버의 인증서를 적절한 트러스트 스토어에 추가합니다. SiteWise Edge는 여러 트러스트 스토어를 사용하여 통신을 보호합니다. 세 개의 트러스트 스토어가 있으며, 이를 사용하는 것은 게이트웨이 구현의 SiteWise Edge 구성 요소 유형에 따라 달라집니다.

트러스트 스토어는 프록시 설정이 제공되면 설치 프로세스 중에 자동으로 업데이트됩니다.

- [AWS IoT Greengrass 코어 구성 요소 트러스트 스토어 구성](#) - AWS IoT Greengrass 루트 CA 인증서는 AWS 서비스의 신뢰성을 확인하기 위해 트러스트 스토어에 포함됩니다.

이 트러스트 스토어는 AWS IoT Greengrass 구성 요소가 프록시를 통해 서비스와 안전하게 통신하는 AWS 동시에 해당 서비스의 신뢰성을 확인하는 데 도움이 됩니다.

- [Java 기반 구성 요소 트러스트 스토어 구성](#) - Java KeyStore(JKS)는 SSL/TLS 연결을 위해 Java 기반 구성 요소가 사용하는 기본 트러스트 스토어입니다.

Java 애플리케이션은 JKS를 사용하여 보안 연결을 설정합니다. 예를 들어 Java 기반 IoT SiteWise 게시자 또는 IoT SiteWise OPC UA 수집기를 사용하는 경우 이 트러스트 스토어를 구성해야 합니다. 이렇게 하면 클라우드로 데이터를 전송하거나 OPC UA 서버에서 데이터를 수집할 때 이러한 구성 요소가 HTTPS 프록시를 통해 안전하게 통신할 수 있습니다.

- [시스템 수준 구성 요소 트러스트 스토어 구성](#) - HTTPS 프록시를 사용하는 경우 보안 연결을 활성화하려면 인증서를 적절한 트러스트 스토어에 추가해야 합니다.

HTTPS 프록시를 사용하는 경우 보안 연결을 활성화하려면 인증서를 적절한 트러스트 스토어에 추가해야 합니다. 이는 Rust 또는 Go와 같은 언어로 작성되는 시스템 수준 구성 요소가 Java의 JKS가 아닌 시스템의 트러스트 스토어에 의존하기 때문에 필요합니다. 예를 들어 프록시를 통해 통신해야 하는 시스템 유틸리티(예: 소프트웨어 업데이트 또는 시간 동기화)를 사용하는 경우 시스템 수준 트러스트 스토어를 구성해야 합니다. 이렇게 하면 이러한 구성 요소와 유틸리티가 프록시를 통해 보안 연결을 설정할 수 있습니다.

AWS IoT Greengrass 코어 구성 요소 트러스트 스토어 구성

Amazon의 루트 CA를 사용하는 AWS IoT Greengrass 코어 함수의 경우:

1. 에서 인증서 파일 찾기 /greengrass/v2/AmazonRootCA1.pem
2. HTTPS 프록시 루트 인증서(자체 서명)를이 파일에 추가합니다.

```
-----BEGIN CERTIFICATE-----
MIIEFTCCA v2gAwIQWgIVAMHSAzWG/5YVRYtRQ0xXUTEpHuEmApzGCSqGSiB3DQEK
\nCwUAhuL9MQswCQwJVUzEPMAVUzEYMBYGA1UECgwP1hem9uLmNvbSBjbMUMRww
... content of proxy CA certificate ...
+vHIRlt0e5JAm5\noTIZGoFbK82A0/n07f/t5PSIDAim9V3Gc3pSXxCCAQoFYnui
GaPUlGk1gCE84a0X\n7Rp/1ND/PuMZ/s8Yj1kY2NmYmNjMCAXDTE5MTEyN2cM216
gJMIADggEPADf2/m45hzEXAMPLE=
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIDQTCCAimgF6AwIBAgITBmyfz/5mjAo54vB4ikPmljZKyjANJmApzyMZFo6qBg
ADA5MQswCQYDVQQGEwJVUzEPMA0tMVT8QtPHRh8jrdkGA1UEChMGDV3QQDExBBKW
... content of root CA certificate ...
o/ufQJQWUCyziar1hem9uMRkwFwYVPSHCb2XV4cdFyQzR1KldZwgJcIQ6XUDgHaa
5MsI+yMRQ+hDaXJioblXgjUka642M4UwtBV8oK2xJNDd2ZhwLnoQdeXeGADKkpy
rqXRfKoQnoZsG4q5WTP46EXAMPLE
-----END CERTIFICATE-----
```

설정된 게이트웨이에서 HTTPS 프록시 구성

포트 8883 대신 포트 443에 연결하여 설정된 게이트웨이에 프록시 지원을 추가할 수 있습니다. 프록시 서버 사용에 대한 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [포트 443에서 연결 또는 네트워크 프록시를 통해](#) 연결을 참조하세요. 새 게이트웨이를 생성하는 경우 게이트웨이 설치 중에 프록시 구성을 설정할 수 있습니다. 자세한 내용은 [AWS IoT SiteWise Edge 게이트웨이 설치 중 프록시 설정 구성](#) 단원을 참조하십시오.

SiteWise Edge AWS IoT Greengrass 에서와 함께 HTTPS 프록시를 사용하는 경우 소프트웨어는 제공된 URL을 기반으로 프록시 연결을 위해 HTTP와 HTTPS 중에서 자동으로 선택합니다.

Important

HTTPS 프록시를 통해 연결을 시도하기 전에 필요한 모든 트러스트 스토어를 업데이트합니다.

Java 기반 구성 요소 트러스트 스토어 구성

데이터 처리 팩의 IoT SiteWise 게시자, IoT SiteWise OPC UA 수집기 및 Java 서비스의 경우 기본 Java 트러스트 스토어 위치는 `$JAVA_HOME/jre/lib/security/cacerts`입니다.

인증서를 추가하려면

1. 와 같은 프록시 서버의 인증서를 저장할 파일을 생성합니다 `proxy.crt`.

Note

프록시 서버의 인증서를 사용하여 미리 파일을 생성합니다.

2. 다음 명령을 사용하여 Java의 트러스트 스토어에 파일을 추가합니다.

```
sudo keytool -import -alias proxyCert -keystore /usr/lib/jvm/java-11-openjdk-amd64/lib/security/cacerts -file proxy.crt
```

3. 메시지가 표시되면 기본 암호를 사용합니다. `changeit`

시스템 수준 구성 요소 트러스트 스토어 구성

Rust, Go 및 시스템 트러스트 스토어를 사용하는 기타 언어로 작성된 구성 요소의 경우:

Linux

Linux 시스템:에 인증서 추가 `/etc/ssl/certs/ca-certificates.crt`

Windows

Microsoft Windows 시스템: 트러스트 스토어를 구성하려면 Microsoft Ignite 설명서의 [인증서 스토어](#) 절차를 따릅니다.

Windows는 각각 여러 하위 저장소가 있는 사용자 및 컴퓨터 범위에 대한 별도의 저장소를 포함하여 여러 인증서 저장소를 제공합니다. 대부분의 SiteWise Edge 설정의 경우 COMPUTER | Trusted Root Certification Authorities 스토어에 인증서를 추가하는 것이 좋습니다. 그러나 특정 구성 및 보안 요구 사항에 따라 다른 스토어를 사용해야 할 수 있습니다.

트러스트 스토어 문제 해결

SiteWise Edge 게이트웨이와 관련된 트러스트 스토어 문제를 해결하는 방법에 대한 자세한 내용은 섹션을 참조하세요 [트러스트 스토어 문제](#).

엣지에서 AWS IoT SiteWise APIs 사용

AWS IoT SiteWise 는 엣지별 APIs와 함께 APIs의 하위 집합을 제공하므로 엣지에 배포된 자산 모델 및 관련 자산과 원활하게 상호 작용할 수 있습니다. 이러한 자산 모델은 엣지에서 실행되도록 구성해야 합니다. 자세한 내용은 [SiteWise Edge에서 데이터 처리를 위한 자산 모델 구성](#)에서 이 설정 프로세스에 대한 자세한 지침을 참조하세요.

이러한 API를 구성한 후 자산 모델 및 개별 자산에 대한 포괄적인 데이터를 검색할 수 있습니다. 자산 모델, 자산, 대시보드, 포털 및 프로젝트 정보를 검색하면 배포된 포털 및 대시보드를 모니터링하고 엣지 수준에서 수집된 자산 데이터에 액세스할 수 있습니다. 이렇게 하면 웹 API 호출 AWS IoT SiteWise 없이와 상호 작용할 수 있도록 네트워크의 중앙 호스트가 제공됩니다.

주제

- [사용 가능한 모든 AWS IoT SiteWise Edge 디바이스 APIs](#)
- [엣지 디바이스에 사용할 AWS IoT SiteWise 엣지 전용 APIs](#)
- [AWS IoT SiteWise Edge APIs에서 CORS 활성화](#)
- [AWS IoT SiteWise Edge에 대한 세션 제한 시간 구성](#)
- [자습서: AWS IoT SiteWise Edge 게이트웨이의 자산 모델 나열](#)

사용 가능한 모든 AWS IoT SiteWise Edge 디바이스 APIs

AWS IoT SiteWise 는 엣지 디바이스에서 사용할 수 있는 다양한 APIs를 제공하므로 디바이스에서 로컬로 작업을 완료할 수 있습니다. 사용 가능한 엣지 API 중 일부에는 자산 모델 검색, 자산 속성 생성 및 업데이트, 클라우드로 데이터 스트림 전송이 포함됩니다. 이러한 API를 활용하여 간헐적이거나 제한된 네트워크 연결이 있는 환경에서 작동할 수 있는 솔루션을 구축할 수 있습니다.

사용 가능한 AWS IoT SiteWise APIs

엣지 디바이스에서 사용할 수 AWS IoT SiteWise APIs는 다음과 같습니다.

- [ListAssetModels](#)
- [DescribeAssetModel](#)
- [ListAssets](#)

- [DescribeAsset](#)
- [DescribeAssetProperty](#)
- [ListAssociatedAssets](#)
- [GetAssetPropertyAggregates](#)
- [GetAssetPropertyValue](#)
- [GetAssetPropertyValueHistory](#)
- [ListDashboards](#)
- [ListPortals](#)
- [ListProjectAssets](#)
- [ListProjects](#)
- [DescribeDashboard](#)
- [DescribePortal](#)
- [DescribeProject](#)

사용 가능한 엣지 전용 API

다음 API는 엣지에 있는 디바이스에서 로컬로 사용됩니다.

- [인증](#) - 이 API를 사용하여 API 직접 호출에 사용할 SigV4 임시 보안 인증을 가져옵니다.

엣지 디바이스에 사용할 AWS IoT SiteWise 엣지 전용 APIs

엣지에서 사용할 수 있는 AWS IoT SiteWise APIs 외에도 엣지별 API가 있습니다. 이러한 엣지별 API가 아래에 설명되어 있습니다.

인증

SiteWise Edge 게이트웨이에서 자격 증명을 가져옵니다. 로컬 사용자를 추가하거나 LDAP 또는 Linux 사용자 풀을 사용하여 시스템에 연결해야 합니다. 사용자 추가에 대한 자세한 내용은 [LDAP](#) 또는 [Linux 사용자 풀](#)을 참조하세요.

요청 구문

```
POST /authenticate HTTP/1.1
Content-type: application/json
{
```

```
"username": "string",  
"password": "string",  
"authMechanism": "string"  
}
```

URI 요청 파라미터

요청은 URI 파라미터를 사용하지 않습니다.

요청 본문

요청은 JSON 형식으로 다음 데이터를 받습니다.

사용자 이름

요청 호출을 검증하는 데 사용되는 사용자 이름입니다.

유형: 문자열

필수 항목 여부: 예

비밀번호

보안 인증을 요청하는 사용자의 비밀번호입니다.

유형: 문자열

필수 항목 여부: 예

authMechanism

호스트에서 이 사용자의 유효성을 검사하기 위한 인증 방법입니다.

유형: 문자열

유효한 값: ldap, linux, winnt

필수 항목 여부: 예

응답 구문

```
HTTP/1.1 200  
Content-type: application/json  
{
```



```
"accessKeyId": "string",  
"secretAccessKey": "string",  
"sessionToken": "string",  
"region": "edge"  
}
```

응답 요소

작업이 성공하면 서비스가 HTTP 200 응답을 반환합니다.

다음 데이터는 JSON 형식으로 반환됩니다.

accessKeyId

임시 보안 인증 정보를 식별하는 액세스 키 ID입니다.

길이 제약: 최소 길이는 16입니다. 최대 길이 128.

패턴: `[\w]*`

secretAccessKey

요청에 서명하는 데 사용되는 비밀 액세스 키입니다.

유형: 문자열

sessionToken

사용자가 임시 보안 인증을 사용하기 위해 서비스 API에 전달해야 하는 토큰입니다.

유형: 문자열

리전

API 직접 호출 대상 지역입니다.

유형: CONSTANT - edge

오류

IllegalArgumentException

제공된 본문 문서의 형식이 잘못되어 요청이 거부되었습니다. 오류 메시지에 특정 오류가 설명되어 있습니다.

HTTP 상태 코드: 400

AccessDeniedException

사용자는 현재 ID 제공자에 기반한 유효한 보안 인증을 가지고 있지 않습니다. 오류 메시지에 인증 메커니즘이 설명되어 있습니다.

HTTP 상태 코드: 403

TooManyRequestsException

요청이 인증 시도 한도에 도달했습니다. 오류 메시지에는 새 인증 시도가 이루어질 때까지 기다리는 시간이 포함됩니다.

HTTP 상태 코드: 429

AWS IoT SiteWise Edge APIs에서 CORS 활성화

AWS IoT SiteWise Edge APIs에서 CORS(교차 오리진 리소스 공유)를 활성화하면 웹 애플리케이션이 여러 도메인에서 APIs를 사용할 수 있습니다. 이를 통해 중간 서버 또는 해결 방법 없이 원활한 통합, 실시간 데이터 교환 및 도메인 간 데이터 액세스를 수행할 수 있습니다. 허용 가능한 오리진을 지정하도록 CORS 설정을 구성하여 교차 오리진 액세스를 제어할 수 있습니다.

Note

CORS는 구성 `aws.iot.SiteWiseEdgeProcessor` 요소의 버전 3.3.1 이상에서 사용할 수 있습니다. 이 기능은 버전 3.3.1 이상에서 사용할 수 있습니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [AWS IoT SiteWise 프로세서](#)를 참조하세요.

SiteWise Edge APIs에서 CORS를 활성화하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. CORS를 활성화하려는 SiteWise Edge 게이트웨이를 선택합니다. AWS IoT Greengrass V2 배포 유형에서 CORS를 활성화할 수 있습니다.
4. 게이트웨이 구성 섹션에서 연결된 Greengrass 코어 디바이스를 선택합니다.
5. 배포 탭의 Greengrass 디바이스에서 적절한 배포 링크를 선택합니다.
6. 작업에서 수정을 선택한 다음 배포 수정을 선택합니다.

⚠ Important

수정된 CORS 지원 구성을 생성하면 디바이스의 현재 구성이 대체됩니다.

7. 1단계, 대상 지정에서 배포를 식별할 선택적 이름을 제공합니다.
8. 2단계, 구성 요소 선택 - 선택 사항에서 현재 선택한 모든 항목을 그대로 두고 다음을 선택할 수 있습니다.
9. 3단계, 구성 요소 구성 - 선택 사항에서 `aws.iot.SiteWiseEdgeProcessor`를 선택하고 구성 요소 구성을 선택합니다.
10. 구성 업데이트 섹션의 병합할 구성에 다음 JSON을 입력합니다.

```
{
  "AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN": "*"
}
```

i Note

를의 값으로 사용하면 모든 오*리진이 `AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN` 허용됩니다. 프로덕션 환경의 경우 보안을 강화하기 위해 정확한 오URLs을 지정하는 것이 좋습니다.

11. 확인을 선택합니다.
12. 다음을 선택하여 Step5, 검토에 도달할 때까지 나머지 단계를 진행합니다.
13. 구성 변경 사항을 검토한 다음 배포를 선택하여 SiteWise Edge 게이트웨이에 변경 사항을 적용합니다.

i Note

또는 전역 환경 변수를 AWS IoT SiteWise 게이트웨이에서 로 설정하여 CORS*를 활성화 `AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN`할 수 있습니다.

Note

인증된 프록시의 경우는 분리된 username 및 url 필드 대신 프록시 구성의 password 필드에 포함되어야 userinfo 합니다.

배포가 완료되면 SiteWise Edge API에서 CORS가 활성화되어 지정된 오리진이 API에 교차 오리진 요청을 할 수 있습니다.

AWS IoT SiteWise Edge에 대한 세션 제한 시간 구성

SiteWise Edge를 사용하면 SiteWise Edge API에 대한 세션 제한 시간을 구성할 수 있습니다. 이 기능은 지정된 기간이 지나면 비활성 세션을 자동으로 종료하여 보안을 강화합니다. 이 섹션에서는 이를 사용하여 세션 제한 시간을 구성하는 프로세스를 안내합니다 AWS IoT SiteWise Console.

Note

세션 제한 시간 구성은 `aws.iot.SiteWiseEdgeProcessor` 구성 요소의 버전 3.4.0 이상에서 사용할 수 있습니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [AWS IoT SiteWise 프로세서](#)를 참조하세요.

SiteWise Edge 게이트웨이에 대한 세션 제한 시간을 구성하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 세션 제한 시간을 구성할 SiteWise Edge 게이트웨이를 선택합니다.

Note

AWS IoT Greengrass V2 배포 유형에서 세션 제한 시간을 구성할 수 있습니다.

4. 게이트웨이 구성 섹션에서 연결된 Greengrass 코어 디바이스를 선택합니다.
5. 배포 탭의 Greengrass 디바이스에서 적절한 배포 링크를 선택합니다.
6. 작업에서 수정을 선택합니다. 경고를 읽은 다음 배포 수정을 선택합니다.

⚠ Important

수정된 세션 제한 시간 구성을 생성하면 디바이스의 현재 구성이 대체됩니다.

7. 1단계, 대상 지정에서 선택적 이름을 제공하여 수정된 배포를 식별한 후 다음을 선택합니다.
8. 2단계, 구성 요소 선택 - 선택 사항에서 모든 현재 선택을 그대로 두고 다음을 선택할 수 있습니다.
9. 3단계, 구성 요소 구성 - 선택 사항에서 `aws.iot.SiteWiseEdgeProcessor`를 선택하고 구성 요소 구성을 선택합니다.
10. 구성 업데이트 섹션의 병합할 구성에 다음 JSON을 입력합니다.

```
{
  "AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES": "240"
}
```

11. 의 값을 분 `AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES` 단위로 설정합니다. 세션 제한 시간 값은 1분에서 10080분(7일) 사이일 수 있습니다. 기본값은 240분(4시간)입니다.
12. 확인을 선택합니다.
13. 다음을 선택하여 5단계, 검토에 도달할 때까지 나머지 단계를 진행합니다.
14. 구성 변경 사항을 검토한 다음 배포를 선택하여 SiteWise Edge 게이트웨이에 변경 사항을 적용합니다.

i Note

또는 글로벌 환경 변수 `AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES`를 SiteWise Edge 게이트웨이에서 원하는 값(분)으로 설정하여 세션 제한 시간을 구성할 수 있습니다.

배포가 완료되면 새 세션 제한 시간 구성이 SiteWise Edge API에 적용됩니다.

자습서: AWS IoT SiteWise Edge 게이트웨이의 자산 모델 나열

엣지별 API와 함께 사용 가능한 AWS IoT SiteWise APIs의 하위 집합을 사용하여 엣지의 자산 모델 및 자산과 상호 작용할 수 있습니다. APIs 이 자습서에서는 Edge 게이트웨이에 대한 임시 자격 증명을 가

저오고 SiteWise AWS IoT SiteWise Edge 게이트웨이에서 자산 모델 목록을 가져오는 방법을 안내합니다.

사전 조건

이 자습서의 단계에서는 다양한 도구를 사용할 수 있습니다. 이러한 도구를 사용하려면 해당 필수 조건이 설치되어 있어야 합니다.

이 자습서를 완료하려면 다음이 필요합니다.

- 배포 및 실행 중인 [AWS IoT SiteWise 엣지 자체 호스팅 게이트웨이 요구 사항](#)
- 포트 443을 통해 동일한 네트워크의 SiteWise Edge 게이트웨이에 액세스합니다.
- [OpenSSL](#) 설치됨
- (AWS OpsHub for AWS IoT SiteWise) [AWS OpsHubAWS IoT SiteWise 애플리케이션용](#)
- (curl) [curl](#) 설치됨
- (Python) [urllib3](#) 설치됨
- (Python) [Python3](#) 설치됨
- (Python) [Boto3](#) 설치됨
- (Python) [BotoCore](#) 설치됨

1단계: SiteWise Edge 게이트웨이 서비스 서명 인증서 가져오기

SiteWise Edge 게이트웨이에서 사용할 수 있는 API에 TLS 연결을 설정하려면 신뢰할 수 있는 인증서가 필요합니다. OpenSSL 또는를 사용하여이 인증서를 생성할 수 AWS OpsHub 있습니다 AWS IoT SiteWise.

OpenSSL

Note

이 명령을 실행하려면 [OpenSSL](#)이 설치되어 있어야 합니다.

SiteWise Edge 게이트웨이에서 서명된 인증서를 가져오려면 터미널을 열고 다음 명령을 실행합니다. <sitewise_gateway_ip>를 SiteWise Edge 게이트웨이의 IP로 바꿉니다.

```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | openssl
x509 -outform PEM > GatewayCert.pem
```

AWS OpsHub for AWS IoT SiteWise

를 AWS OpsHub 에 사용할 수 있습니다 AWS IoT SiteWise. 자세한 내용은 [SiteWise Edge 게이트웨이 관리](#) 단원을 참조하십시오.

이 자습서에서는 다운로드한 SiteWise Edge 게이트웨이 인증서의 절대 경로를 사용합니다. 다음 명령을 실행하여 인증서의 전체 경로를 내보내고 <absolute_path_to_certificate>를 인증서 경로로 대체합니다.

```
export PATH_TO_CERTIFICATE='<absolute_path_to_certificate>'
```

2단계: SiteWise Edge 게이트웨이 호스트 이름 가져오기

Note

이 명령을 실행하려면 [OpenSSL](#)이 설치되어 있어야 합니다.

자습서를 완료하려면 SiteWise Edge 게이트웨이의 호스트 이름이 필요합니다. SiteWise Edge 게이트웨이의 호스트 이름을 가져오려면 <sitewise_gateway_ip>를 SiteWise Edge 게이트웨이의 IP로 대체하여 다음을 실행합니다.

```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | grep -Po 'CN = \K.*' | head -1
```

다음 명령을 실행하여 나중에 사용할 수 있도록 호스트 이름을 내보내고 <your_edge_gateway_hostname>을 SiteWise Edge 게이트웨이의 호스트 이름으로 대체합니다.

```
export GATEWAY_HOSTNAME='<your_edge_gateway_hostname>'
```

3단계: SiteWise Edge 게이트웨이의 임시 자격 증명 가져오기

이제 서명된 인증서와 SiteWise Edge 게이트웨이의 호스트 이름을 가지고 있으므로 SiteWise Edge 게이트웨이에서 API를 실행할 수 있도록 임시 자격 증명을 가져와야 합니다. AWS OpsHub 용을 통해 AWS IoT SiteWise 또는 API를 사용하여 SiteWise Edge 게이트웨이에서 직접 이러한 자격 증명을 가져올 수 있습니다. APIs

⚠ Important

자격 증명은 4시간마다 만료되므로 SiteWise Edge 게이트웨이에서 API를 사용하기 직전에 자격 증명을 가져와야 합니다. 보안 인증을 4시간 이상 캐시하지 마세요.

AWS OpsHub 용를 사용하여 임시 자격 증명 가져오기 AWS IoT SiteWise**ℹ Note**

[AWS OpsHub](#) [AWS IoT SiteWise 애플리케이션용](#)가 설치되어 있어야 합니다.

AWS OpsHub 를 AWS IoT SiteWise 애플리케이션에 사용하여 임시 자격 증명을 가져오려면 다음을 수행합니다.

1. 애플리케이션에 로그인합니다.
2. 설정을 선택합니다.
3. 인증에서 보안 인증 복사를 선택합니다.
4. 환경에 맞는 옵션을 확장하고 복사를 선택합니다.
5. 나중에 사용할 수 있도록 보안 인증을 저장합니다.

SiteWise Edge 게이트웨이 API를 사용하여 임시 자격 증명 가져오기

SiteWise Edge 게이트웨이 API를 사용하여 임시 자격 증명을 가져오려면 Python 스크립트 또는 curl 을 사용할 수 있습니다. 먼저 SiteWise Edge 게이트웨이의 사용자 이름과 암호가 있어야 합니다. SiteWise Edge 게이트웨이는 SigV4 인증 및 권한 부여를 사용합니다. 사용자 추가에 대한 자세한 내용은 [LDAP](#) 또는 [Linux 사용자 풀](#)을 참조하세요. 이러한 자격 증명은 다음 단계에서 API를 사용하는데 필요한 SiteWise Edge 게이트웨이의 로컬 자격 증명을 가져오는 데 사용됩니다 AWS IoT SiteWise APIs.

Python**ℹ Note**

[urllib3](#)와 [Python3](#)이 설치되어 있어야 합니다.

Python을 사용하여 보안 인증을 가져오려면

1. `get_credentials.py`라는 파일을 생성하고 다음 코드를 그 안에 복사합니다.

```
'''
The following demonstrates how to get the credentials from the SiteWise Edge
gateway. You will need to add local users or connect your system to LDAP/AD
https://docs.aws.amazon.com/iot-sitewise/latest/userguide/manage-gateways-ggv2.html#create-user-pool

Example usage:
    python3 get_credentials.py -e https://<gateway_hostname> -c
    <path_to_certificate> -u '<gateway_username>' -p '<gateway_password>' -m
    '<method>'
'''

import urllib3
import json
import urllib.parse
import sys
import os
import getopt

"""
This function retrieves the AWS IoT SiteWise Edge gateway credentials.
"""
def get_credentials(endpoint, certificatePath, user, password, method):
    http = urllib3.PoolManager(cert_reqs='CERT_REQUIRED', ca_certs=
    certificatePath)
    encoded_body = json.dumps({
        "username": user,
        "password": password,
        "authMechanism": method,
    })

    url = urllib.parse.urljoin(endpoint, "/authenticate")

    response = http.request('POST', url,
        headers={'Content-Type': 'application/json'},
        body=encoded_body)

    if response.status != 200:
        raise Exception(f'Failed to authenticate! Response status
        {response.status}')
```

```

auth_data = json.loads(response.data.decode('utf-8'))

accessKeyId = auth_data["accessKeyId"]
secretAccessKey = auth_data["secretAccessKey"]
sessionToken = auth_data["sessionToken"]
region = "edge"

return accessKeyId, secretAccessKey, sessionToken, region

def print_help():
    print('Usage:')
    print(f'{os.path.basename(__file__)} -e <endpoint> -c <path/to/certificate> -u <user> -p <password> -m <method> -a <alias>')
    print('')
    print('-e, --endpoint    edge gateway endpoint. Usually the Edge gateway hostname.')
    print('-c, --cert_path  path to downloaded gateway certificate')
    print('-u, --user        Edge user')
    print('-p, --password   Edge password')
    print('-m, --method     (Optional) Authentication method (linux, winnt, ldap), default is linux')
    sys.exit()

def parse_args(argv):
    endpoint = ""
    certificatePath = None
    user = None
    password = None
    method = "linux"

    try:
        opts, args = getopt.getopt(argv, "he:c:u:p:m:", ["endpoint=", "cert_path=", "user=", "password=", "method="])
    except getopt.GetoptError:
        print_help()

    for opt, arg in opts:
        if opt == '-h':
            print_help()
        elif opt in ("-e", "--endpoint"):
            endpoint = arg
        elif opt in ("-u", "--user"):

```

```
        user = arg
    elif opt in ("-p", "--password"):
        password = arg
    elif opt in ("-m", "--method"):
        method = arg.lower()
    elif opt in ("-c", "--cert_path"):
        certificatePath = arg

    if method not in ['ldap', 'linux', 'winnt']:
        print("not valid method parameter, required are ldap, linux, winnt")
        print_help()

    if (user == None or password == None):
        print("To authenticate against edge user, password have to be passed together, and the region has to be set to 'edge'")
        print_help()

    if(endpoint == ""):
        print("You must provide a valid and reachable gateway hostname")
        print_help()

    return endpoint,certificatePath, user, password, method

def main(argv):
    # get the command line args
    endpoint, certificatePath, user, password, method = parse_args(argv)

    accessKeyId, secretAccessKey, sessionToken, region=get_credentials(endpoint, certificatePath, user, password, method)

    print("Copy and paste the following credentials into the shell, they are valid for 4 hours:")
    print(f"export AWS_ACCESS_KEY_ID={accessKeyId}")
    print(f"export AWS_SECRET_ACCESS_KEY={secretAccessKey}")
    print(f"export AWS_SESSION_TOKEN={sessionToken}")
    print(f"export AWS_REGION={region}")
    print()

if __name__ == "__main__":
```

```
main(sys.argv[1:])
```

2. 터미널에서 `get_credentials.py`를 실행하여 `<gateway_username>` 및 `<gateway_password>`를 생성한 보안 인증으로 교체합니다.

```
python3 get_credentials.py -e https://$GATEWAY_HOSTNAME -c $PATH_TO_CERTIFICATE
-u '<gateway_username>' -p '<gateway_password>' -m 'linux'
```

curl

Note

[curl](#)이 설치되어 있어야 합니다.

curl을 사용하여 보안 인증을 가져오려면

1. `<gateway_username>` 및 `<gateway_password>`를 생성한 자격 증명으로 교체하여 터미널에서 다음 명령을 실행합니다.

```
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"
}'
```

응답은 다음과 같아야 합니다.

```
{
  "username": "sweuser",
  "accessKeyId": "<accessKeyId>",
  "secretAccessKey": "<secretAccessKey>",
  "sessionToken": "<sessionToken>",
  "sessionExpiryTime": "2022-11-17T04:51:40.927095Z",
  "authMechanism": "linux",
  "role": "edge-user"
}
```

2. 터미널에서 다음 명령을 실행합니다.

```
export AWS_ACCESS_KEY_ID=<accessKeyId>
export AWS_SECRET_ACCESS_KEY=<secretAccessKey>
export AWS_SESSION_TOKEN=<sessionToken>
export AWS_REGION=edge
```

4단계: SiteWise Edge 게이트웨이의 자산 모델 목록 가져오기

서명된 인증서, SiteWise Edge 게이트웨이 호스트 이름, SiteWise Edge 게이트웨이의 임시 자격 증명 이 준비되었으므로 ListAssetModels API를 사용하여 SiteWise Edge 게이트웨이의 자산 모델 목록 을 가져올 수 있습니다.

Python

Note

[Python3](#), [Boto3](#), [BotoCore](#)가 설치되어 있어야 합니다.

Python을 사용하여 자산 모델 목록을 가져오려면

1. list_asset_model.py라는 파일을 만들고 그 안에 다음 코드를 복사합니다.

```
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. 터미널에서 `list_asset_model.py`를 실행합니다.

```
python3 list_asset_model.py
```

curl

 Note

[curl](#)이 설치되어 있어야 합니다.

curl을 사용하여 자산 모델 목록을 가져오려면

터미널에서 다음 명령을 실행합니다.

```
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"
```

응답은 다음과 같아야 합니다.

```
{
  "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
}
```

Siemens Industrial Edge에서 SiteWise Edge 게이트웨이 호스팅

Edge 애플리케이션을 사용하여 Siemens Industrial AWS IoT SiteWise Edge에서 게이트웨이를 호스팅합니다. AWS IoT Greengrass V2와 마찬가지로 Siemens Industrial Edge에서 SiteWise Edge를 사용하여 제조 프로세스를 최적화하거나 운영 워크플로를 개선할 수 있습니다.

디바이스에서 SiteWise Edge 게이트웨이를 실행하여 Siemens Industrial Edge 디바이스에서 AWS 계정으로 데이터를 수집할 수 있습니다. 이렇게 하려면 SiteWise AWS IoT SiteWise Edge 지원 팀에 Edge 애플리케이션에 대한 액세스를 요청합니다. 그런 다음 배포 대상이 Siemens Industrial Edge 디바이스 - 신규인 SiteWise Edge 게이트웨이 리소스를 생성합니다. 그런 다음 구성 파일을 다운로드하고 Siemens Industrial Edge Management 포털을 통해 애플리케이션에 업로드합니다. 필요한 Siemens 리소스를 설정하는 방법을 포함하여 Siemens Industrial Edge에서 애플리케이션을 실행하는 방법에 대한 자세한 내용은 Siemens 설명서의 [Industrial Edge란 무엇인가요?](#)를 참조하세요.

Note

Siemens는 SiteWise Edge의 공급업체 또는 공급자가 아닙니다. Siemens Industrial Edge Marketplace는 독립적인 마켓플레이스입니다.

주제

- [보안](#)
- [Siemens Secure Storage 및 AWS IoT SiteWise Edge 애플리케이션](#)
- [Siemens Industrial Edge 디바이스의 대상](#)
- [미리 보기 애플리케이션에서 마이그레이션](#)
- [문제 해결](#)
- [AWS IoT SiteWise 엣지 애플리케이션 변경 로그](#)
- [AWS IoT SiteWise Edge 애플리케이션에 대한 요구 사항](#)
- [Siemens Industrial Edge의 게이트웨이 생성](#)

- [애플리케이션을 위한 Siemens Databus user 생성](#)
- [AWS IoT SiteWise Edge 애플리케이션에 액세스](#)
- [Siemens 디바이스에 애플리케이션 설치](#)
- [AWS IoT SiteWise Edge 애플리케이션 구성 업데이트](#)

보안

AWS, 고객 및 파트너 간의 [공동 책임 모델](#)의 일환으로 다음과 같이 보안의 다양한 측면을 담당하는 사람을 설명합니다.

고객의 책임

- 파트너를 심사합니다.
- 파트너에게 부여된 네트워크 액세스를 구성합니다.
- SiteWise Edge를 실행하는 디바이스를 물리적으로 보호합니다.

AWS 책임

- 고객 AWS 클라우드 리소스에서 파트너를 격리합니다.

파트너의 책임

- 보안 기본값을 사용합니다.
- 패치 및 기타 적절한 업데이트를 통해 시간이 지나도 솔루션을 안전하게 유지합니다.
- 고객 데이터의 기밀을 유지합니다.
- 파트너 마켓플레이스에서 사용할 수 있는 다른 애플리케이션 확인

Siemens Secure Storage 및 AWS IoT SiteWise Edge 애플리케이션

Edge 애플리케이션을 실행하는 데 필요한 자격 증명과 보안 암호를 보호하기 위해 AWS IoT SiteWise Siemens Industrial Edge는 자격 증명을 디바이스에 안전하게 저장하는 메커니즘을 제공합니다. AWS IoT SiteWise Edge 애플리케이션은 이러한 자격 증명을 안전하게 저장할 수 없는 경우 디바이스에서 실행되지 않습니다. 안전한 저장 지원이 누락되어 발생한 실행 실패는 로그 파일에 기록됩니다.

AWS IoT SiteWise Edge 애플리케이션을 설치하고 실행하려면 다음과 같은 최소 OS 버전이 필요합니다. 디바이스를 최신 버전으로 업그레이드하여 애플리케이션을 설치합니다.

- 가상 디바이스의 경우: IEVD 버전 1.19 이상
- 물리적 디바이스의 경우: IED-OS 버전 2.2 이상

의 AWS IoT SiteWise Edge 애플리케이션 Siemens Industrial Edge은 디바이스를 업그레이드할 때까지 실행되지 않습니다.

Siemens Industrial Edge 디바이스의 대상

에서 AWS IoT SiteWise Edge 애플리케이션을 사용하는 경우 Siemens Industrial Edge 대상은 추가 분석 및 배포 AWS IoT SiteWise 를 위해 로 전송하기 전에 데이터를 준비하는 데 도움이 됩니다. Amazon S3를 사용하여 버퍼링된 데이터 수집을 위한 데이터 대상 설정을 구성하거나 실시간 데이터 수집을 사용할 수 있습니다. 둘 다 디바이스 Siemens Industrial Edge 배포 대상의 경로 필터를 사용하여 MQTT 주제를 구독할 수 있습니다.

게이트웨이의 Siemens Industrial Edge 배포 대상은 두 가지 기본 데이터 처리 방법을 지원합니다.

- AWS IoT SiteWise 실시간 설정 - 데이터가 수집되면 AWS IoT SiteWise 로 직접 전송됩니다.
- AWS IoT SiteWise Amazon S3 설정을 사용하여 버퍼링됨 - Amazon S3로 전송되기 전에 데이터가 일시적으로 수집되고 배치로 저장됩니다.

이러한 옵션 구성에 대한 자세한 내용은 [Amazon S3를 사용하여 AWS IoT SiteWise 버퍼링된 대상 추가](#) 및 섹션을 참조하세요 [AWS IoT SiteWise Edge 실시간 대상 추가](#).

경로 필터의 접두사

Siemens Industrial Edge 배포 대상을 사용하는 게이트웨이의 경로 필터는 주제와 데이터 스트림 이름을 모두 결합하여 데이터에 대한 고유 식별자를 생성합니다. 데이터 스트림 이름과 결합된 주제를 Siemens Industrial Edge 게이트웨이의 접두사라고 합니다. 이는 경로 필터가 MQTT 주제만을 기반으로 하는 자체 호스팅 게이트웨이와 다릅니다.

Example Siemens 데이터 스트림의 경로 필터 구조

Siemens 데이터 스트림의 일반적인 경로 필터에는 주제 경로와 데이터 스트림 이름이 모두 포함됩니다.

```
ie/d/device1/application1/datastream1
```

위치:

- ie/d/는 Siemens 데이터 스트림에 필요한 접두사입니다.
- device1/application1는 계층적 경로를 나타냅니다.

- `datastream1`는 특정 데이터 스트림 이름입니다.

Note

Siemens Industrial Edge 데이터 스트림으로 작업할 때 데이터 스트림에 대한 전체 정보를 수신하려면 경로 필터에 메타데이터(`ie/m/`) 및 데이터(`ie/d/`) 주제를 모두 포함해야 합니다.

대상 및 경로 필터

MQTT 지원 게이트웨이의 대상 및 경로 필터에 대해 자세히 알아보려면 다음 주제를 참조하세요.

- [AWS IoT SiteWise 엣지 대상 이해](#)
- [AWS IoT SiteWise Edge 실시간 대상 추가](#)
- [Amazon S3를 사용하여 AWS IoT SiteWise 버퍼링된 대상 추가](#)
- [AWS IoT SiteWise Edge 대상의 경로 필터 이해](#)
- [AWS IoT SiteWise Edge 대상에 경로 필터 추가](#)
- [AWS IoT SiteWise 엣지 대상 관리](#)

미리 보기 애플리케이션에서 마이그레이션

미리 보기 단계 중에 Siemens Industrial Edge에서 SiteWise Edge를 실행한 경우 미리 보기 버전 1.0.1에서 최신 버전으로 업그레이드해야 합니다. 마이그레이션하려면 다음을 수행합니다.

1. 새 SiteWise Edge 게이트웨이를 생성합니다. 자세한 내용은 [Siemens Industrial Edge의 게이트웨이 생성](#) 단원을 참조하십시오.
2. 각 새 게이트웨이에 대해 새 Siemens Databus user를 생성합니다. 자세한 내용은 [애플리케이션을 위한 Siemens Databus user 생성](#) 단원을 참조하십시오.
3. IED에서 버전 1.0.1 AWS IoT SiteWise Edge 게이트웨이 애플리케이션을 제거합니다.

Note

AWS IoT SiteWise Edge 애플리케이션의 미리 보기 버전에서 이전에 사용한 AWS IoT SiteWise 자산을 재구성할 때 데이터 흐름 중단에 대비합니다. 데이터 기록은 보존되지만 새 게이트웨이를 다시 설치하는 동안 데이터가 손실될 수 있습니다.

4. [AWS IoT SiteWise Console](#)에서 미리 보기 중에 생성한 SiteWise Edge 게이트웨이를 삭제합니다.
5. 새 게이트웨이 구성 파일을 사용하여 IED에 AWS IoT SiteWise Edge 게이트웨이 애플리케이션을 설치합니다. 자세한 내용은 [Siemens 디바이스에 애플리케이션 설치](#) 단원을 참조하십시오.

Important

새 게이트웨이를 설치하면 SiteWise Edge 애플리케이션의 미리 보기 버전을 덮어쓰게 됩니다. 버전 2.0.0을 설치한 후에는 버전 1.0.1로 돌아갈 수 없습니다.

새 게이트웨이 및 Siemens Databus user를 구성한 후 데이터가 속성으로 전송됩니다.

SiteWise Edge 애플리케이션을 버전에서 직접 업그레이드할 수도 있습니다. 그러나 새 게이트웨이 구성은 여전히 필요합니다.

문제 해결

Siemens Industrial Edge 디바이스의 SiteWise Edge 게이트웨이 문제를 해결하려면 [여기서 AWS IoT SiteWise Edge 애플리케이션 문제 해결 Siemens Industrial Edge](#) 섹션을 참조하세요.

[AWS re:Post](#)에 액세스하여 질문에 대한 답변을 찾을 수도 있습니다.

AWS IoT SiteWise 엣지 애플리케이션 변경 로그

다음 표에서는 AWS IoT SiteWise Edge 애플리케이션의 각 버전에 대한 변경 사항을 설명합니다.

버전	변경 사항	
3.0.0	새로운 특성 • AWS IoT SiteWise 콘솔에서 설치된 게이트웨이에 대한 추가 구성 옵션을 추가합니다. Siemens Industrial Edge • Amazon S3 대상을 추가하고 구성합니다. • AWS IoT SiteWise 실시간 대상을 제거합니다. • 각 대상에 대한 필터를 추가하여 데이터 라우팅을 제어합니다.	

버전	변경 사항	
	- 글로벌 접두사 구성에 대한 옵션을 추가하여 Siemens IE 설치 관리자의 IoT SiteWise 게시자에서 모든 데이터를 전송합니다. 접두사는 필터링 후 적용됩니다. 버그 수정 및 개선 사항 - Edge 애플리케이션의 초기 시도가 실패할 경우 Siemens Industrial AWS IoT SiteWise Edge Databus에 대한 연결을 재 시도하는 기능을 추가합니다.	
2.0.1	버그 수정 및 개선 사항 시작 시 자격 AWS 증명을 검색할 수 없는 경우 앱이 실패 상태가 되고 종료되는 문제를 해결합니다. 자격 증명 검색이 성공할 때까지 재시도할 수 있는 지원을 추가합니다.	
2.0.0	- 이제 AWS IoT SiteWise Edge 애플리케이션을 정식 버전으로 사용할 수 있습니다. - 애플리케이션에는 Siemens IEVD 버전 1.19 또는 Siemens IED-OS 버전 2.2가 필요합니다. - 성능 개선: 메모리 및 CPU 사용량이 감소했습니다. - 디버깅 개선 사항: 이제 선택적 구성 파일을 업로드하여 디버그 로그를 활성화할 수 있습니다. - 보안 개선 사항: 애플리케이션은 SecureStorage API를 사용하여 자격 증명을 디바이스에 안전하게 저장합니다. - Docker 다이제스트 값: sha256:4a960f29234a190ebb5224c1fd0f3e99faafccc4cb3d93ca13fef247b6656d18	
1.0.1	초기 릴리스	

AWS IoT SiteWise Edge 애플리케이션에 대한 요구 사항

에서 AWS IoT SiteWise Edge를 실행하려면 다음이 Siemens Industrial Edge 필요합니다.

- [Siemens Digital Exchange Platform](#) 계정.
- Siemens Industrial Edge Hub(iehub) 계정.
- Siemens Industrial Edge Management 인스턴스.
 - IE 앱 구성 서비스. 자세한 내용은 Siemens Industrial Edge Management 설명서의 [Installing the IE App Configuration Service manually](#) 섹션을 참조하세요.
- AWS IoT SiteWise Edge 애플리케이션의 버전 2.0.1 이상에 대한 액세스. 자세한 내용은 [AWS IoT SiteWise Edge 애플리케이션에 액세스](#) 단원을 참조하십시오.
- Siemens Industrial Edge 디바이스(IED) 또는 Siemens Industrial Edge 가상 디바이스(IEVD).
 - 하드웨어 요구 사항을 위한 최소 15GB의 디스크 공간.
 - 1GB의 RAM과 추가 1GB 스왑 메모리.
 - 포트 443 및 8883에서 아웃바운드 트래픽을 허용하는 디바이스 구성.
 - x86-64비트 프로세서.
 - Siemens Industrial Edge Management 버전 1.13.10 이상.
 - Siemens Secure Storage 요구 사항에 대한 디바이스 적합성.
 - 가상 디바이스의 경우 IEVD 버전 1.19 이상.
 - 물리적 디바이스의 경우 IED-OS 버전 2.2 이상.
 - Docker Compose의 최신 버전.
 - Docker Engine 버전 18.091 이상.
- 필수 도메인 액세스. 자세한 내용은 [AWS IoT SiteWise 엔드포인트](#) 단원을 참조하십시오.

Siemens Industrial Edge의 게이트웨이 생성

적절한 Siemens 계정과 IEM 인스턴스가 있으면 배포 유형 Siemens Industrial Edge 디바이스의 SiteWise Edge 게이트웨이를 생성할 수 있습니다.

Note

Siemens Industrial Edge Management에서 디바이스를 실행하기 위한 모든 요구 사항을 충족하는지 확인합니다. 자세한 내용은 [AWS IoT SiteWise Edge 애플리케이션에 대한 요구 사항](#) 단원을 참조하십시오.

구성 파일을 생성하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 게이트웨이 생성을 선택합니다.
4. 배포 유형에서 Siemens Industrial Edge 디바이스를 선택합니다.
5. SiteWise Edge 게이트웨이의 이름을 입력하거나 AWS IoT SiteWise에서 생성한 이름을 사용합니다.
6. (선택 사항) 고급 구성에서 다음을 수행합니다.
 - AWS IoT Core 사물의 이름을 입력하거나에서 생성된 이름을 사용합니다 AWS IoT SiteWise.
7. 게이트웨이 생성을 선택합니다.
8. SiteWise Edge 게이트웨이 구성 파일 생성 대화 상자에서 생성 및 다운로드를 선택합니다.는 AWS IoT SiteWise Edge 애플리케이션을 구성하는 데 사용할 구성 파일을 AWS IoT SiteWise 자동으로 생성합니다.

Important

게이트웨이 구성 파일을 사용하여 AWS IoT SiteWise Edge 애플리케이션을 백업하고 복원합니다. SiteWise Edge 게이트웨이 구성 파일을 [AWS Secrets Manager](#)에 저장하여 파일을 안전하게 저장하고 관리합니다. Secrets Manager는 민감한 정보를 안전하게 저장, 관리 및 검색합니다.

애플리케이션을 위한 Siemens Databus user 생성

AWS IoT SiteWise Edge on Siemens Industrial Edge는 Siemens Databus 애플리케이션에서 데이터를 수집합니다. SiteWise Edge를 Siemens Databus에 연결하려면 AWS IoT SiteWise로 안전하게 전송하려는 데이터에 대한 액세스를 제공하는 Siemens Databus user가 필요합니다. 시작하려면 Siemens Databus user를 생성한 다음 SiteWise Edge 애플리케이션에 자격 증명을 제공합니다.

Siemens Databus user를 생성하려면 다음을 수행하세요.

1. Siemens Industrial Edge Management 인스턴스의 플랫폼 애플리케이션 섹션에서 엣지 관리를 선택합니다.
2. 데이터 연결 아이콘을 선택합니다.

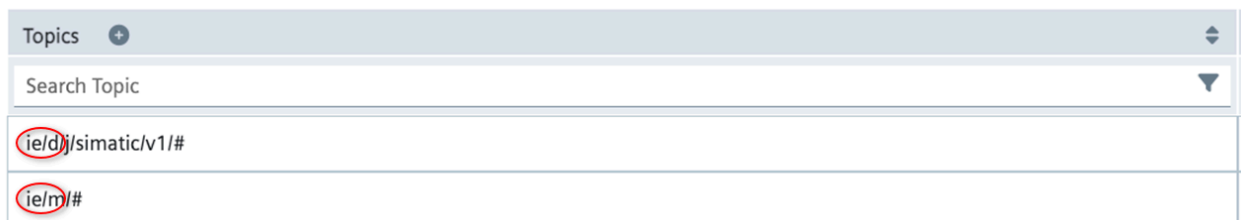
3. Databus를 선택합니다. 연결된 디바이스 목록이 나타납니다.
4. AWS IoT SiteWise Edge 애플리케이션에 연결할 디바이스를 선택합니다.
5. 시작을 선택합니다. 선택한 디바이스의 Databus Configurator가 나타납니다.
6. 사용자에서 Edge 디바이스에 대한 사용자를 생성합니다. 사용자 생성에 대한 자세한 내용은 Siemens Industrial Edge Management 설명서의 [사용자를](#) 참조하세요.
7. 이 Siemens Databus에 액세스 권한이 있는 주제를 선택합니다. 이 주제에서는 AWS IoT SiteWise Edge가 액세스할 수 있는 항목을 제한합니다.

Important

Siemens Databus user가 액세스할 수 있는 모든 주제는 AWS IoT SiteWise에 게시됩니다.

Note

Siemens Databus user는 데이터 및 메타데이터 주제에 모두 액세스해야 합니다. ie/d로 시작하는 주제는 데이터 주제입니다. ie/m으로 시작하는 주제는 메타데이터 주제입니다. SiteWise Edge가 각 주제의 데이터와 메타데이터 모두에 액세스할 수 있도록 주제를 쌍으로 공유합니다.



8. Siemens Databus 구성에 대한 적절한 권한을 설정합니다.

Siemens Databus 구성을 생성한 후에 AWS IoT SiteWise Edge 애플리케이션을 설치할 수 있습니다. 자세한 내용은 [Siemens 디바이스에 애플리케이션 설치](#) 단원을 참조하십시오.

Siemens Industrial Edge 게이트웨이의 대상 및 경로 필터를 선택적으로 구성할 수도 있습니다. 자세한 내용은 [대상 및 경로 필터](#) 단원을 참조하십시오.

AWS IoT SiteWise Edge 애플리케이션에 액세스

에서 AWS IoT SiteWise Edge 애플리케이션에 액세스하려면 SiteWise Edge 지원 팀에 액세스를 요청하는 이메일을 Siemens Industrial Edge보냅니다. <mailto:aws-iot-swe-siemens-app-support@amazon.com?subject=Access request for SiteWise Edge on Siemens Industrial Edge>

사용자의 이메일에 다음 정보를 포함합니다.

- 이름 및 연락처 정보
- 회사 이름
- Siemens Industrial Edge 테넌트 ID

Siemens 디바이스에 애플리케이션 설치

에 대한 SiteWise AWS IoT SiteWise Edge 지원 팀에 이메일을 보내 Edge 애플리케이션에 액세스할 수 있게 되면 애플리케이션을 인스턴스에 Siemens Industrial Edge할당합니다Siemens Industrial Edge Management. 그런 다음 디바이스에 AWS IoT SiteWise Edge 애플리케이션을 설치할 수 있습니다.

AWS IoT SiteWise Edge 애플리케이션을 설치하려면

1. 에 제공된 다이제스트가 Docker에 나열된 최신 버전Siemens Industrial Edge Management과 일치하는지 확인합니다[AWS IoT SiteWise 엣지 애플리케이션 변경 로그](#).

Siemens의 Docker 다이제스트 값을 찾는 방법에 대한 자세한 내용은 Siemens 설명서의 Siemens Industrial Edge 디바이스에서 [앱 관리](#)를 참조하세요.

Siemens Industrial Edge Management는 한 번에 하나의 AWS IoT SiteWise Edge 애플리케이션 버전을 지원합니다. Siemens Industrial Edge 디바이스에 AWS IoT SiteWise Edge 애플리케이션을 설치하기 전에이 단계를 수행하여 최신 버전의 애플리케이션을 사용하고 있는지 확인합니다.

2. Siemens Industrial Edge Management에 AWS IoT SiteWise Edge 애플리케이션을 할당합니다. 자세한 내용은 Siemens 설명서의 Industrial Edge Management 섹션에서 [앱 관리](#)를 참조하세요.
3. Edge Management에서 AWS IoT SiteWise Edge 카탈로그를 찾아 선택합니다.
4. 설치를 선택합니다.

Note

문의 버튼이 표시되면 해당 버튼을 선택하고 단계에 따라의 AWS IoT SiteWise Edge 애플리케이션에 대한 액세스를 요청합니다. 자세한 내용은 [AWS IoT SiteWise Edge 애플리케이션에 액세스](#) 단원을 참조하십시오.

- 스키마 구성 옵션에서 Databus_Configuration을 선택합니다.
- Databus 구성의 사용자 이름과 암호를 입력합니다. Siemens Databus user 생성 방법에 대한 자세한 내용은 [애플리케이션을 위한 Siemens Databus user 생성](#) 섹션을 참조하세요.
- Databus_Configuration 옆의 작고 둥근 회색 체크 표시 아이콘을 선택하여 아이콘 색상을 녹색으로 바꿉니다.

Note

입력 구성은 체크 표시 아이콘이 회색에서 녹색으로 변경된 경우에만 적용됩니다. 그렇지 않으면 입력 구성이 무시됩니다.

Databus_Configuration Invalid Configuration

Provide Databus user credentials for the AWS IoT SiteWise Edge application to collect data from the Databus application. SiteWise Edge will publish all topics to the cloud that the user has access to. You can manage access using the Databus app

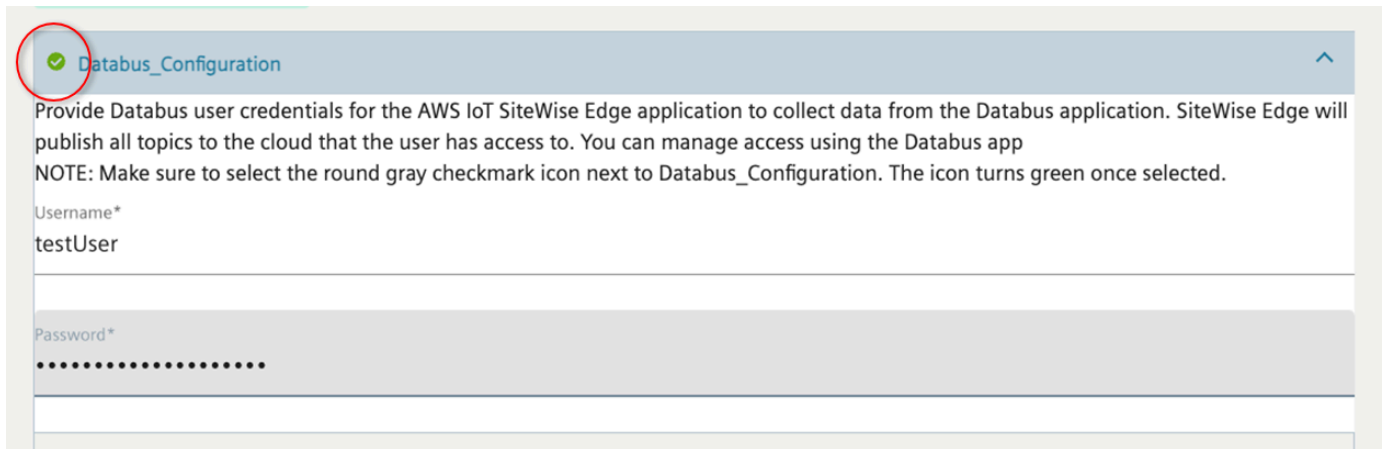
NOTE: Make sure to select the round gray checkmark icon next to Databus_Configuration. The icon turns green once selected.

Username*

is a required property

Password*

is a required property



Provide Databus user credentials for the AWS IoT SiteWise Edge application to collect data from the Databus application. SiteWise Edge will publish all topics to the cloud that the user has access to. You can manage access using the Databus app

NOTE: Make sure to select the round gray checkmark icon next to Databus_Configuration. The icon turns green once selected.

Username*

testUser

Password*

.....

8. 다음을 선택하여 게이트웨이 구성 파일을 업로드할 수 있는 다른 구성으로 이동합니다.
9. 게이트웨이 구성 파일을 업로드할 위치로 SiteWise_Edge_Gateway_Config를 선택합니다.

Note

SiteWise_Edge_Support_Config_Optional 대신 SiteWise_Edge_Gateway_Config를 선택해야 합니다.

10. 애플리케이션을 설치할 디바이스를 선택합니다.
11. 지금 설치를 선택합니다.

선택적으로 데이터를 AWS 클라우드로 내보내도록 게시자 구성 요소를 구성할 수 있습니다. 자세한 내용은 [AWS IoT SiteWise 게시자 구성 요소 구성](#)을 참조하세요.

Siemens Industrial Edge 게이트웨이의 대상을 구성하려면 섹션을 참조하세요 [대상 및 경로 필터](#).

AWS IoT SiteWise Edge 애플리케이션 구성 업데이트

에서 AWS IoT SiteWise Edge 애플리케이션 구성을 업데이트할 때 고려해야 할 몇 가지 사항이 있습니다Siemens Industrial Edge.

Note

AWS IoT SiteWise Edge 애플리케이션 구성을 변경하려면 애플리케이션을 다시 시작해야 합니다.

AWS IoT SiteWise Edge 애플리케이션을 다시 시작해야 하는 이유

- Edge 애플리케이션을 Siemens Databus user 위한 새로운 AWS IoT SiteWise 입니다.
- 게이트웨이 구성 파일(SiteWise_Edge_Gateway_Config 파일)에 대한 변경 사항.
- 프록시 구성 업데이트(전체 IEVD 재부팅 필요)
- 디버깅 문제에 대한 디버깅 로그를 활성화하려면 다음을 수행하세요.

애플리케이션 다시 시작

1. Siemens Industrial Edge Management 인스턴스의 플랫폼 애플리케이션 섹션에서 엣지 관리를 선택합니다.
2. 내 설치된 앱을 선택합니다.
3. AWS IoT SiteWise Edge 애플리케이션을 선택합니다.
4. 다시 시작을 선택합니다.

대상 및 경로 필터

AWS IoT SiteWise 엣지의 대상은 엣지 디바이스에서 클라우드로 산업 데이터가 흐르는 방식을 유연하고 효율적으로 관리할 수 있는 방법을 제공합니다. 이 섹션에서는 대상을 구성하고, 경로 필터를 사용하여 특정 데이터 스트림을 라우팅하고, 사용 사례에 적합한 대상 유형을 선택하는 방법을 설명합니다.

에서 호스팅되는 AWS IoT SiteWise Edge 애플리케이션과 함께 사용되는 자체 호스팅 MQTT 지원 V3 게이트웨이 및 게이트웨이에서 대상 및 경로 필터를 사용할 수 있습니다Siemens Industrial Edge. 대상 및 경로 필터는 Classic Streams, V2 게이트웨이에서 작동하지 않습니다.

주제

- [AWS IoT SiteWise 엣지 대상 이해](#)
- [AWS IoT SiteWise Edge 대상의 경로 필터 이해](#)
- [AWS IoT SiteWise Edge 실시간 대상 추가](#)
- [Amazon S3를 사용하여 AWS IoT SiteWise 버퍼링된 대상 추가](#)
- [AWS IoT SiteWise Edge 대상에 경로 필터 추가](#)
- [AWS IoT SiteWise 엣지 대상 관리](#)

AWS IoT SiteWise 엣지 대상 이해

AWS IoT SiteWise Edge 대상을 사용하여 소스 데이터를 전송할 위치를 결정합니다. 비용 효율성, 짧은 지연 시간 또는 스토리지 요구 사항과 같이 필요한 특정 특성에 따라 데이터 대상을 선택할 수 있습니다. AWS IoT SiteWise, 파트너 또는 사용자 지정 애플리케이션에서 캡처한 디바이스 데이터를 통합하여 엣지에서 경로 필터(주제)를 게시하고 구독합니다. 그런 다음 디바이스 데이터를 모델링, 전송 및 클라우드에 저장할 수 있습니다.

Note

자체 호스팅 게이트웨이에서 모든 대상 기능을 완전히 사용하려면 IoT SiteWise 게시자 및 IoT SiteWise OPC UA 수집기의 최신 버전으로 업그레이드하세요. 스트림 지원은 기존 설정과의 호환성을 유지하기 위해 V2 게이트웨이인 Classic 스트림에서 계속됩니다. 자세한 내용은 [클래식 스트림](#), [AWS IoT SiteWise Edge용 V2 게이트웨이](#) 섹션을 참조하세요.

주제

- [SiteWise Edge 대상이 데이터 관리를 개선하는 방법](#)
- [대상 유형](#)
- [게이트웨이 버전 간 대상 기능 비교](#)
- [대상 제한 사항](#)
- [SiteWise Edge 대상 사용 사례](#)

SiteWise Edge 대상이 데이터 관리를 개선하는 방법

Amazon S3를 사용하여 엣지에서 AWS IoT SiteWise 실시간으로 또는 배치로 데이터를 내보냅니다.

대상은 AWS IoT SiteWise 환경의 유연성과 확장성을 향상시킵니다. 대상은 소스가 중앙 시스템에 데이터를 게시하는 중앙 집중식 데이터 관리 모델을 구현합니다. 대상은 경로 필터를 사용하여 데이터가 전송되는 위치를 결정합니다. 대상은 여러 경로 필터를 구독할 수 있습니다.

자체 호스팅되거나에서 실행되는 MQTT 지원 게이트웨이는 로컬 통신에 MQTT를 Siemens Industrial Edge 사용하고 필터가 로 설정된 기본 실시간 대상과 함께 제공됩니다#. 즉, 기본적으로 모든 주제의 모든 메시지가 AWS IoT SiteWise 실시간 대상에 게시됩니다. 자세한 내용은 [AWS IoT SiteWise Edge 대상의 경로 필터 이해](#) 단원을 참조하십시오. 각 게이트웨이에 하나의 실시간 대상을 추가할 수 있습니다.

대상 유형

게이트웨이의 대상을 구성할 때를 사용한 실시간 구성 AWS IoT SiteWise와 Amazon S3를 사용한 버퍼링된 구성이라는 두 가지 기본 옵션이 있습니다. 각 대상 유형에는 고유한 설정 및 고려 사항이 있습니다.

AWS IoT SiteWise 실시간 설정

이 옵션을 선택하면 AWS IoT SiteWise 데이터를 핫 티어 스토리지로 직접 전송하여 데이터를 실시간으로 쉽게 수집하고 모니터링할 수 있습니다. 실시간 설정은 데이터 흐름을 관리합니다. 특히 게이트웨이에 클라우드 연결 문제가 있는 경우 더욱 그렇습니다. 연결이 끊어지는 동안 데이터는 게이트웨이에 로컬로 임시로 저장됩니다. 연결이 다시 설정되면 저장된 데이터가 자동으로 클라우드로 전송됩니다.

로컬에 저장할 최대 데이터 양, 재연결 시 클라우드로 데이터가 전송되는 속도, 스토리지가 용량에 도달한 후 데이터를 삭제하는 시기 등 데이터 게시 프로세스의 다양한 측면을 조정할 수 있습니다.

AWS IoT SiteWise 스토리지 계층에 대한 자세한 내용은 섹션을 참조하세요 [에서 데이터 스토리지 관리 AWS IoT SiteWise](#).

AWS IoT SiteWise Amazon S3 설정을 사용하여 버퍼링됨

이 대상 유형을 사용하면 게이트웨이에서 로컬로 데이터를 버퍼링하고 주기적으로 Amazon S3 버킷에 배치로 전송할 수 있습니다. 데이터는 분석 워크로드에 최적화된 효율적인 Parquet 형식으로 저장됩니다. Amazon S3에 데이터가 저장, 처리 및 분석을 AWS IoT SiteWise 위해 로 데이터를 가져올 수 있습니다.

데이터를 배치로 수집하고 기록 데이터를 비용 효과적인 방식으로 저장하려면 이 옵션을 선택합니다. 원하는 Amazon S3 버킷 위치와 데이터를 Amazon S3에 업로드할 빈도를 구성할 수 있습니다. AWS IoT SiteWise에 수집한 후 데이터로 수행할 작업을 선택할 수도 있습니다. SiteWise와 Amazon S3 모두에서 데이터를 사용할 수 있도록 선택하거나 Amazon S3에서 데이터를 자동으로 삭제하도록 선택할 수 있습니다.

게이트웨이 버전 간 대상 기능 비교

MQTT 지원 게이트웨이의 대상 기능은 데이터 흐름 관리를 간소화합니다. 대상은 다양한 엔드포인트로의 데이터 라우팅을 중앙 집중식으로 구성하여 데이터 관리를 간소화합니다. 이 접근 방식을 사용하면 복잡한 개별 스트림 설정이 필요하지 않으므로 전체 시스템을 보다 유연하고 쉽게 관리할 수 있습니다.

이에 비해 클래식 스트림, V2 게이트웨이, SiteWise Edge는 AWS IoT Greengrass 스트림을 통해 데이터 소스의 데이터를 게시자에게 전송하여 각 데이터 소스에 대해 데이터 대상을 개별적으로 구성합니다.

AWS IoT SiteWise 대상 기능을 사용하면 게시자 라우팅 구성이 통합됩니다. 대상 구성을 사용하면 중앙 집중식 방식으로 대상 및 경로 필터를 관리할 수 있습니다. 필요에 따라 대상을 쉽게 추가하고, 경로 필터를 관리하고, 불필요한 필터 또는 대상을 삭제할 수 있습니다.

또한 대상 기능은 산업용 IoT 애플리케이션에 널리 사용되는 업계 표준 프로토콜인 MQTT(Message Queuing Telemetry Transport)를 활용합니다. MQTT를 채택 AWS IoT SiteWise 하면 다양한 디바이스 및 시스템과 쉽게 통합할 수 있습니다.

대상 제한 사항

SiteWise Edge 게이트웨이의 대상에 대한 현재 제한 사항은 다음과 같습니다.

- 데이터 처리 팩은 MQTT 지원 게이트웨이에서 지원되지 않습니다.
- 데이터 형식 지원은 AWS IoT SiteWise 데이터 형식으로 제한됩니다. 데이터 유형 변환 활성화에 대한 자세한 내용은 섹션을 참조하세요 [지원되지 않는 데이터 유형 변환](#).

SiteWise Edge 대상 사용 사례

SiteWise Edge 대상은 다양한 애플리케이션에서 사용됩니다. 다음은 몇 가지 주요 예입니다.

산업 자동화, 실시간 모니터링 및 예측 유지 관리

산업 환경에서는 공장 현장의 센서와 디바이스가 SiteWise Edge에 데이터를 게시할 수 있습니다. 관련 데이터를 필터링하고 라우팅하도록 대상을 구성하여 기계 성능을 실시간으로 모니터링하고 분석할 수 있습니다. 경로 필터를 사용하여 관련 MQTT 주제를 구독하고 데이터를 처리한 다음 처리된 데이터를 게시할 수 있습니다. 이렇게 하면 처리된 데이터를 AWS 클라우드 분석 서비스 또는 온프레미스 시스템으로 선택적으로 라우팅할 수 있습니다. 그런 다음 제조업체는 예측 유지 관리 전략을 구현하고, 프로덕션 프로세스를 최적화하고, 가동 중지 시간을 줄일 수 있습니다.

스마트 빌딩, 에너지 효율성 및 점유율 최적화

빌딩 자동화 시스템은 데이터 스트림을 생성하여 HVAC 시스템, 조명 및 액세스 제어와 같은 빌딩의 다양한 측면을 모니터링하고 제어합니다. SiteWise Edge를 사용하면 이러한 데이터 스트림을 수집, 처리 및 다른 대상으로 라우팅할 수 있습니다. 시설 관리자는 관련 데이터를 필터링하고 전달하도록 대상을 구성하여 데이터 프라이버시 및 규정 준수를 보장하는 동시에 에너지 효율성 측정 및 점유율 최적화와 같은 고급 기능을 사용할 수 있습니다.

이러한 사용 사례는 SiteWise Edge의 대상 기능을 다양한 산업에서 활용하여 데이터를 효율적으로 수집, 처리 및 라우팅하는 방법을 보여줍니다. 이를 통해 실시간 모니터링, 예측 유지 관리, 에너지 효율성, 원격 진단과 같은 고급 기능을 제공하는 동시에 데이터 프라이버시 및 규정 준수를 보장할 수 있습니다.

AWS IoT SiteWise Edge 대상의 경로 필터 이해

주제

- [경로 필터 요구 사항](#)
- [경로 필터 모범 사례](#)
- [OPC UA 서버의 경로 필터](#)
- [경로 필터 이름의 특수 문자](#)

각 대상은 AWS IoT SiteWise 또는 Amazon S3로 데이터를 라우팅하도록 구성됩니다. 경로 필터를 사용하면 대상에 대한 MQTT 메시지를 수신할 때 필터링할 특정 데이터를 선택할 수 있습니다. 경로 필터는 원하는 MQTT 주제에 대한 구독 역할을 하는 데이터 스트림의 논리적 이름을 나타냅니다.

MQTT에서 데이터는 슬래시(/)로 구분된 계층적 문자열인 주제로 구성됩니다/. 예를 들어 디바이스는 주제에 온도 데이터를 게시할 수 있습니다 `home/livingroom/sensor1/temperature`. 여기서 `home/livingroom/sensor1`는 센서의 경로 또는 논리적 이름을 나타내며 `temperature`는 게시되는 데이터 유형입니다.

경로 필터를 사용하여 와일드카드(+ 및 #)를 사용하여 특정 주제 또는 다양한 주제를 구독할 수 있습니다#. + 와일드카드는 주제 계층 구조의 단일 수준과 일치합니다. 예를 들어 `home/+ /sensor1/temperature`는 `home/livingroom/sensor1/temperature` 및 `home/bedroom/sensor1/temperature`와 일치합니다. # 와일드카드는 필터 끝에 사용할 때 여러 수준과 일치합니다.

경로 필터 이름 내에서 MQTT 사양에 일반적으로 허용되지 않는 다양한 문자를 사용할 수도 있습니다. 이러한 문자는 이름 내에서 사용할 때 와일드카드로 작동하지 않습니다. 원래 이름 지정 구조를 유지하면서 MQTT 규정 준수를 보장하기 위해 인코딩을 사용하여 이러한 문자를 AWS IoT SiteWise 변환합니다. 이 기능은 다른 시스템의 기존 이름 지정 규칙을 수용하는 데 특히 유용합니다. 자세한 내용은 [경로 필터 이름의 특수 문자](#) 단원을 참조하십시오.

적절한 경로 필터를 신중하게 선택하면 특정 대상으로 전송되는 데이터를 제어할 수 있습니다. 경로 필터를 사용하여 IoT 시스템의 요구 사항에 맞게 데이터 흐름을 조정합니다.

경로 필터 요구 사항

를 사용하여 경로 필터를 입력할 때는 다음 사항에 유의 AWS IoT SiteWise Console하세요.

- 경로 필터는 새 줄로 구분되며 각 줄은 별도의 경로 필터를 나타냅니다.
- 개별 경로 필터는 1~65,535바이트일 수 있습니다.
- 경로 필터는 비워둘 수 없습니다.
- Null 값(U+0000)은 허용되지 않습니다.
- 한 번에 최대 100개의 경로 필터 또는 65,535자 중 먼저 도달하는 값을 입력할 수 있습니다.
- 전체 제한은 게이트웨이의 모든 대상에 대해 결합된 20,000개의 경로 필터입니다.
- 경로 필터 이름 내에서 %, +, 및 \$ 문자를 사용할 수 있지만 #는 이를 URI 인코딩으로 AWS IoT SiteWise 자동 변환합니다.

경로 필터 모범 사례

AWS IoT SiteWise 대상에 대한 경로 필터를 생성할 때 데이터를 효과적으로 관리하기 위해 다음 전략을 고려하세요.

- 디바이스 계층 구조를 미러링하도록 필터를 구성합니다. 예를 들어 제조 설정에서는 서로 다른 생산 라인의 모든 시스템에서 데이터를 `factory/+/machine/#` 캡처합니다.
- 디바이스 유형, 위치 또는 함수에 특정 수준을 사용합니다. 예를 들어 `factory/assembly-line/robot/temperature`입니다. 또는 스마트 농업에서 `farm/+/crop/+/moisture`를 사용하여 다양한 필드의 다양한 자르기에 대한 습도 수준을 모니터링합니다.
- 와일드카드를 전략적으로 활용: 단일 수준에서 변형+에를 사용하고 모든 후속 수준을 캡처#합니다. 예를 들어 `building/+/+/energy-consumption`는 건물의 여러 영역과 바닥에서 에너지 사용량을 추적합니다. 이는 첫 번째가 모든 층을 + 캡처하고 두 번째가 모든 영역을 + 캡처한다고 가정합니다.
- 관련 데이터를 캡처할 수 있을 만큼 구체적이지만 향후 변경 사항을 수용할 수 있을 만큼 유연한 필터를 생성하여 특이도와 유연성의 균형을 맞춥니다. 예를 들어 `site/+/equipment-type/+/measurement`에서는 필터 구조를 변경하지 않고 새 사이트 또는 장비 유형을 추가할 수 있습니다.

필터를 철저히 테스트하여 의도한 데이터를 캡처하고 IoT 시스템의 아키텍처 및 목표에 맞는지 확인합니다.

OPC UA 서버의 경로 필터

OPC UA 서버의 경우 경로 필터가 OPC UA 태그 이름과 일치해야 합니다. 경로 필터의 최종 수준은 OPC UA 태그 이름과 정확히 일치해야 합니다. 예를 들어 OPC UA 태그가 Device1.Temperature인 경우 경로 필터는 일 수 있습니다 factory/line1/Device1.Temperature. 이전 수준에서 와일드카드를 사용하여 여러 프로덕션 라인에서 태그를 캡처 factory/+ /Device1.Temperature 할 수 있습니다. 경로 필터 이름에 특수 문자가 있는 경우 자세한 내용은 [경로 필터 이름의 특수 문자](#) 섹션을 참조하세요.

경로 필터 이름의 특수 문자

AWS IoT SiteWise 는 OPC UA와 같은 산업 프로토콜에서 일반적으로 사용되는 문자를 수용하며, 표준 MQTT 주제 이름에서는 일반적으로 허용되지 않습니다. 이 기능을 사용하면 산업 시스템을 MQTT 기반 아키텍처와 보다 원활하게 통합할 수 있습니다.

Note

특수 문자 처리는 통합 및 마이그레이션에 유용하지만 더 광범위한 호환성을 보장하기 위해 가능한 경우 새 구현에 대한 표준 MQTT 이름 지정 규칙에 맞추는 것이 좋습니다.

산업 소스에서 데이터를 수신할 때는 특수 문자에 대한 URI 인코딩을 사용하여 주제 이름을 AWS IoT SiteWise 정규화합니다.

- %가 됨%25(먼저 이스케이프 문자로 인코딩됨)
- #는 %23가 됩니다.
- +는 %2B가 됩니다.
- \$가 됩니다%24(주제 시작 시에만).

이 인코딩을 사용하면 이러한 특수 MQTT 문자가 포함된 소스 데이터를 MQTT 주제 이름으로 안전하게 사용할 수 있는 동시에 원래 산업 명명 규칙을 유지할 수 있습니다.

Example : 경로 필터 이름의 특수 문자

다음은 AWS IoT SiteWise 경로 필터에 산업 주제 이름이 표시되는 방법의 예입니다.

- Factory1/Line#2/Sensor+3는 Factory1/Line%232/Sensor%2B3가 됩니다.
- Plant%A/Unit\$1/Temp는 Plant%25A/Unit%241/Temp가 됩니다.

- Site1/#Section/+Node는 Site1/%23Section/%2BNode가 됩니다.

에서 구독을 생성하거나 주제 이름을 볼 때 인코딩되지 않은 원본 버전이 AWS IoT SiteWise 표시됩니다. 인코딩은 MQTT 규정 준수를 보장하기 위해 자동으로 처리됩니다.

AWS IoT SiteWise Edge 실시간 대상 추가

실시간 대상 유형을 사용하면 디바이스 및 게이트웨이에서 AWS IoT SiteWise 스토리지로 IoT 데이터를 실시간으로 직접 스트리밍할 수 있습니다. 이 옵션은 일괄 처리 또는 버퍼링 없이 생성된 데이터를 즉시 수집하고 처리해야 하는 사용 사례에 적합합니다. 데이터를 지속적으로 스트리밍하므로 각 게이트웨이에 하나의 실시간 대상만 구성할 수 있습니다 AWS IoT SiteWise.

Note

TQVs 중복되면 이중 충전이 발생할 수 있습니다.

실시간 대상을 추가하려면

AWS IoT SiteWise 콘솔 또는를 사용하여 SiteWise Edge MQTT 지원 게이트웨이에 실시간 대상을 AWS CLI 추가합니다.

Console

1. [AWS IoT SiteWise 콘솔](#)을 엽니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 대상을 추가할 게이트웨이를 선택합니다.
4. 대상 섹션에서 대상 추가를 선택합니다.
5. 대상 추가 페이지에서 대상 세부 정보를 입력합니다.
 - a. 대상 이름 필드의 대상 이름입니다.
 - b. 대상 유형의 AWS IoT SiteWise 실시간을 선택합니다.
6. 게시 순서를 이전 데이터 먼저 게시 또는 최신 데이터 먼저 게시로 설정하여 게이트웨이 게시 순서를 구성합니다. 기본적으로 게이트웨이는 가장 오래된 데이터를 먼저 게시합니다.
7. 최대 배치 대기 시간을 사용하여 데이터 배치를 보내기 전에 게시자가 대기할 최대 시간을 설정합니다 AWS IoT SiteWise. 이 설정은 각 별칭에 적용됩니다. 데이터는 다음 중 하나가 발생할 때까지 로컬에 저장됩니다.

- 설정된 시간이 경과한 경우 또는
- 별칭에 대해 10개의 time-quality-value(TQV) 항목이 수신됩니다.

어떤 조건이 충족되든 먼저 배치가 클라우드로 전송되도록 트리거합니다.

- 업로드된 데이터를 압축하려면 데이터 업로드 시 압축 활성화 확인란을 선택합니다. 클라우드에 업로드하기 전에 게이트웨이가 데이터를 압축하도록 하면 대역폭 사용량이 줄어듭니다.
- 만료된 게시자 데이터를 필터링하려면 만료된 데이터 제외 확인란을 선택합니다. 이 선택은 활성 및 현재 데이터만 로 전송합니다 AWS IoT SiteWise.
- 마감 기간 필드에 데이터 세트 내에서 데이터가 만료된 것으로 간주되어야 하는 빈도를 입력합니다. 데이터가 분 또는 일 단위로 계산되는지 확인할 수 있습니다. 최소 마감 기간은 5분입니다. 최대 마감 기간은 7일입니다.
- 선택적으로 로컬 스토리지 설정을 구성합니다.
 - 보존 기간 빈도 설정 - 게이트웨이가 마감 기간보다 오래된 데이터를 로컬에 저장하는 시간입니다. 최소 보존 기간은 1분입니다.

최대 보존 기간은 30일이며 교체 기간 이상입니다.

- 교체 기간 설정 - 단일 파일의 마감 기간보다 오래된 데이터를 저장할 때 지정할 시간 간격입니다. 게이트웨이는 각 교체 기간이 끝날 때 데이터 배치 하나를 로컬 디렉터리 /greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports로 전송합니다.

보존 기간은 1분보다 크고 보존 기간과 같아야 합니다.

- 스토리지 용량(GB) 값을 제공하여 로컬에 GB로 저장된 데이터의 최대 크기를 설정합니다. 데이터가 결정된 최대 로컬 스토리지 크기를 초과하면 게이트웨이가 가장 오래된 데이터를 먼저 삭제하기 시작합니다. 게이트웨이는 로컬에 저장된 데이터 크기가 할당량과 같거나 작을 때까지 계속 삭제됩니다.

스토리지 용량은 1GB 이상이어야 합니다.

- 대상에 경로 필터를 추가합니다. 자세한 내용은 단원을 참조하십시오 [AWS IoT SiteWise Edge 대상에 경로 필터 추가](#).

자세한 내용은 [대상 유형](#) 단원을 참조하십시오.

AWS CLI

Example : 새 AWS IoT SiteWise 실시간 대상 생성

[UpdateGatewayCapabilityConfiguration](#) API를 사용하여 게시자를 구성합니다.

capabilityNamespace 파라미터를 iotsitewise:publisher:3로 설정합니다.

```
{
  "sources": [
    {
      "type": "MQTT"
    }
  ],
  "destinations": [
    {
      "type": "SITEWISE_REALTIME",
      "name": "your-destination-name",
      "config": {
        "publishingOrder": "TIME_ORDER",
        "enableCompression": true,
        "maxBatchWaitTime": "10s"
      },
      "filters": [
        {
          "type": "PATH",
          "config": {
            "paths": [
              "#"
            ]
          }
        }
      ]
    }
  ]
}
```

기존 AWS IoT SiteWise 실시간 대상을 업데이트하려면 먼저 DescribeGatewayCapabilityConfiguration API를 사용하여 `destinationId`를 찾습니다.

Example : AWS IoT SiteWise 실시간 대상 업데이트

[UpdateGatewayCapabilityConfiguration](#) API를 사용하여 게시자를 구성합니다.

capabilityNamespace 파라미터를 iotsitewise:publisher:3로 설정합니다.

```
{
  "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": [
              "#"
            ]
          }
        }
      ]
    }
  ]
}
```

다음 구성 옵션은 iotsitewise:publisher:3 네임스페이스를 사용하는 게이트웨이에만 해당됩니다.

sources

산업 장비에서 로 데이터를 전송할 데이터 소스를 정의합니다 AWS IoT SiteWise. MQTT 지원 게이트웨이의 경우를 사용합니다 MQTT.

타입: 객체 배열

필수 여부: 예

destinations

데이터를 전송할 위치를 정의합니다. 대상은 Amazon S3를 사용하여 실시간으로 또는 버퍼링됩니다. 하나 이상의 대상 객체가 필요하지만 빈 배열을 추가할 수 있습니다. 각 게이트웨이에 대해 하나의 실시간 대상을 가질 수 있습니다. 자세한 내용은 [AWS IoT SiteWise 엣지 대상 이해](#) 단원을 참조하십시오.

유형: 객체 배열

필수 여부: 예

id

대상의 고유 식별자입니다. 기존 대상 ID를 제공하거나 비워 둘 수 있습니다. ID를 지정하지 않으면 UUID가 기본적으로 생성됩니다.

유형: 문자열

필수 항목 여부: 아니요

type

대상의 유형입니다. 옵션에는 SITEWISE_REALTIME 및가 포함됩니다 SITEWISE_BUFFERED.

- SITEWISE_REALTIME - 실시간으로 스토리지로 AWS IoT SiteWise 직접 데이터를 전송합니다.
- SITEWISE_BUFFERED - 데이터를 Parquet 형식으로 Amazon S3에 배치로 전송한 다음 AWS IoT SiteWise 스토리지로 가져옵니다.

유형: 문자열

필수 항목 여부: 예

name

대상의 고유한 이름입니다.

유형: 문자열

필수 항목 여부: 예

config

JSON 형식의 대상 유형별 구성입니다. 구성은 실시간 대상과 버퍼링된 대상에 따라 다릅니다.

유형: 객체

필수 여부: 예

publishingOrder

데이터가 게시되는 순서를 결정합니다. 데이터는 타임스탬프를 기반으로 게시됩니다. 옵션에는 TIME_ORDER 및 RECENT_DATA가 포함됩니다.

- TIME_ORDER (기본값) - 오래된 데이터를 먼저 게시합니다.
- RECENT_DATA - 최신 데이터를 먼저 게시합니다.

유형: 문자열

필수 항목 여부: 아니요

enableCompression

로 설정하면 로 전송하기 전에 데이터 압축을 true활성화합니다 AWS IoT SiteWise. 클라우드에 업로드하기 전에 게이트웨이가 데이터를 압축하도록 하면 대역폭 사용량이 줄어듭니다. 기본값은 true입니다.

유형: 부울

필수 항목 여부: 아니요

dropPolicy

이전 데이터를 처리하는 방법을 정의합니다.

유형: 객체

필수 여부: 아니요

- `cutoffAge`

일, 시간 및 분으로 지정된 게시할 데이터의 최대 수명입니다. 예: 7d 또는 1d7h16m. 지정한 것보다 오래된 데이터는 로 전송되지 않습니다 AWS IoT SiteWise.

마감 기간 이전의 데이터는 클라우드에 게시되지 않습니다. 마감 기간은 5분에서 7일 사이여야 합니다.

마감 연령을 지정할 때 m, h, d를 사용할 수 있습니다. m은 분, h는 시간, d는 일을 나타냅니다.

유형: 문자열

필수 항목 여부: 예

- `exportPolicy`

컷오프 기간을 초과하는 데이터를 처리하는 방법을 정의합니다.

유형: 객체

필수 여부: 아니요

- `retentionPeriod`

SiteWise Edge 게이트웨이는 지정된 보존 기간 동안 저장된 후 로컬 스토리지에서 마감 기간보다 이른 엣지의 모든 데이터를 삭제합니다. 보존 기간은 1분에서 30일 사이여야 하며, 교체 기간보다 길거나 같아야 합니다.

보존 기간을 지정할 때 m, h, d를 사용할 수 있습니다. m은 분, h는 시간, d는 일을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

- `rotationPeriod`

마감 기간보다 이른 데이터를 일괄 처리하여 단일 파일에 저장하는 시간 간격입니다. SiteWise Edge 게이트웨이는 각 교체 기간이 끝날 때 데이터 배치 하나를 로컬 디렉터리 `/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`로 전송합니다. 교체 기간은 1분보다 크며 보존 기간과 같거나 작아야 합니다.

교체 기간을 지정할 때 m, h, d를 사용할 수 있습니다. m은 분, h는 시간, d는 일을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

- exportSizeLimitGB

로컬에 저장된 최대 허용 데이터 크기(GB)입니다. 이 할당량이 위반되면 SiteWise Edge 게이트웨이는 로컬에 저장된 데이터 크기가 할당량 이하가 될 때까지 가장 이른 데이터를 삭제하기 시작합니다. 이 파라미터의 값은 1보다 크거나 같아야 합니다.

유형: 정수

필수 항목 여부: 아니요

maxBatchWaitTime

게시자가 데이터 배치를 보내기 전에 대기할 최대 시간을 설정합니다 AWS IoT SiteWise. 이 설정은 각 별칭에 적용됩니다. 데이터는 다음 중 하나가 발생할 때까지 로컬에 저장됩니다.

- 설정된 시간이 경과한 경우 또는
- 별칭에 대해 10개의 time-quality-value(TQV) 항목이 수신됩니다.

m, h 및 d를 사용하여 마감 시간을 지정합니다. m은 분, h는 시간, d는 일을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

filters

데이터에 적용할 필터입니다. 하나 이상의 필터가 필요합니다.

유형: 문자열

필수 항목 여부: 예

type

필터 유형입니다. PATH를 사용합니다.

유형: 문자열

필수 항목 여부: 예

config

JSON 형식의 필터 유형별 구성입니다. 하나 이상의 객체가 필요하지만 배열이 비어 있을 수 있습니다.

유형: 객체

필수 여부: 예

- paths

경로 필터의 배열입니다. 자세한 내용은 [AWS IoT SiteWise Edge 대상의 경로 필터 이해](#) 단원을 참조하십시오. 기본 경로는 #입니다.

유형: 문자열 배열

필수 여부: 예

Amazon S3를 사용하여 AWS IoT SiteWise 버퍼링된 대상 추가

버퍼링된 대상 유형을 사용하면 데이터가 실시간으로 필요하지 않은 AWS IoT SiteWise 경우 수집 비용을 절감할 수 있습니다. 이를 통해 IoT 데이터를 가져오기 전에 Amazon S3 버킷에 임시로 저장할 수 있습니다 AWS IoT SiteWise. 또는 데이터를 가져올 계획인지 여부에 관계없이 스토리지를 위해 S3에 데이터를 업로드할 수 있습니다 AWS IoT SiteWise. 이는 데이터를 수집하기 전에 디바이스 및 게이트웨이에서 데이터를 일괄 처리하고 버퍼링하는 데 유용합니다 AWS IoT SiteWise. 이 옵션을 사용하면 데이터가 Parquet 형식으로 지정된 S3 버킷에 구성된 빈도로 업로드됩니다. 그런 다음 추가 분석 및 처리를 위해 데이터를 AWS IoT SiteWise 스토리지로 가져올 수 있습니다.

Amazon S3를 사용하여 버퍼링된 대상을 추가하려면

AWS IoT SiteWise 콘솔 또는를 사용하여 Amazon S3 AWS CLI 를 사용하여 데이터를 SiteWise Edge 게이트웨이에 버퍼링하는 대상을 추가합니다.

Console

AWS Management Console 를 사용하여 Amazon S3를 사용하여 버퍼링된 AWS IoT SiteWise 대상을 추가합니다.

1. [AWS IoT SiteWise 콘솔](#)을 엽니다.

2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 대상을 추가할 게이트웨이를 선택합니다.
4. 대상 섹션에서 대상 추가를 선택합니다.
5. 대상 추가 페이지에서 대상 세부 정보를 입력합니다.
 - a. 대상 이름 필드의 대상 이름입니다.
 - b. AWS IoT SiteWise Amazon S3 for Destination 유형을 사용하여 버퍼링된를 선택합니다. Amazon S3를 사용하여 AWS IoT SiteWise 버퍼링된 데이터를 Parquet 형식으로 Amazon Simple Storage Service에 배치로 전송한 다음 스토리지로 AWS IoT SiteWise 가져옵니다.
6. 게이트웨이 데이터를 저장할 위치의 Amazon S3 URL을 입력합니다. S3 찾아보기를 선택하여 경로를 찾을 수 있습니다. 버킷이 추가되면 보기를 선택하여 버킷을 볼 수도 있습니다.
7. 기간을 입력하고 데이터 업로드 빈도에 대한 시간 증분을 선택하여 게이트웨이가 Amazon S3에 데이터를 업로드하는 빈도를 지정합니다. 빈도 값은 0보다 크고 30일보다 작거나 같아야 합니다.
8. 데이터 스토리지 설정에서 게이트웨이 데이터를 가져온 후 게이트웨이 데이터로 수행할 작업을 결정합니다 AWS IoT SiteWise. 데이터 스토리지와 관련하여 두 가지 결정을 내려야 합니다.
 - 가져온 데이터를 AWS IoT SiteWise 스토리지에 복사하려면 스토리지에 데이터 복사 확인란을 선택합니다. 이 옵션은 구성된 Amazon S3 버킷에서 가져온 데이터를 AWS IoT SiteWise 스토리지로 복제합니다.
 - Amazon S3 버킷에서 AWS IoT SiteWise 스토리지로 데이터를 가져오도록 선택한 경우 가져오기가 완료된 후 가져온 데이터를 삭제할지 여부도 지정할 수 있습니다. Amazon S3에서 데이터 삭제 확인란을 선택하여 AWS IoT SiteWise 스토리지로 가져온 후 구성된 Amazon S3 버킷에서 가져온 날짜를 삭제합니다.
9. 대상에 경로 필터를 추가합니다. 자세한 내용은 단원을 참조하십시오 [AWS IoT SiteWise Edge 대상에 경로 필터 추가](#).

AWS CLI

Example : Amazon S3를 사용하여 버퍼링된 새 AWS IoT SiteWise 대상 생성

[UpdateGatewayCapabilityConfiguration](#) API를 사용하여 게시자를 구성합니다.

capabilityNamespace 파라미터를 `iotsitewise:publisher:3`로 설정합니다.

```
{
  "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 : Amazon S3를 사용하여 버퍼링된 AWS IoT SiteWise 대상 업데이트

기존 AWS IoT SiteWise 실시간 대상을 업데이트하려면 먼저 DescribeGatewayCapabilityConfiguration API를 사용하여 찾습니다destinationId.

게시자 네임스페이스: 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": [
              "#"
            ]
          }
        }
      ]
    }
  ]
}

```

다음 구성 옵션은 `iotsitewise:publisher:3` 네임스페이스를 사용하는 MQTT 지원 게이트웨이에만 적용됩니다.

sources

산업 장비에서 로 데이터를 전송할 데이터 소스를 정의합니다 AWS IoT SiteWise. MQTT 지원 게이트웨이의 경우를 사용합니다 MQTT.

타입: 객체 배열

필수 여부: 예

destinations

데이터를 전송할 위치를 정의합니다. 대상은 Amazon S3를 사용하여 실시간으로 또는 버퍼링됩니다. 하나 이상의 대상 객체가 필요하지만 빈 배열을 추가할 수 있습니다. 각 게이트웨이에 대해 하나의 실시간 대상을 가질 수 있습니다. 자세한 내용은 [AWS IoT SiteWise 엣지 대상 이해](#) 단원을 참조하십시오.

유형: 객체 배열

필수 여부: 예

id

대상의 고유 식별자입니다. 기존 대상 ID를 제공하거나 비워 두어 대상에 대해 새 ID가 자동으로 생성되도록 할 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

type

대상의 유형입니다. 옵션에는 SITEWISE_REALTIME 및가 포함됩니다 SITEWISE_BUFFERED. SITEWISE_BUFFERED를 선택합니다.

- SITEWISE_REALTIME (기본값) - 실시간으로 AWS IoT SiteWise 스토리지로 직접 데이터를 전송합니다. 자세한 내용은 [AWS IoT SiteWise Edge 실시간 대상 추가](#) 단원을 참조하십시오.
- SITEWISE_BUFFERED - 데이터를 Parquet 형식으로 Amazon S3에 배치로 전송한 다음 AWS IoT SiteWise 스토리지로 가져옵니다.

유형: 문자열

필수 항목 여부: 예

name

대상의 고유한 이름입니다.

유형: 문자열

필수 항목 여부: 예

config

JSON 형식의 대상 유형별 구성입니다. 구성은 실시간 대상과 버퍼링된 대상에 따라 다릅니다.

유형: 객체

필수 여부: 예

targetBucketArn

게시할 버킷 ARN입니다. AWS IoT SiteWise 및 Amazon S3 모두에 AWS 리전 대해 동일한를 선택합니다. 접두사를 선택한 경우 1~255자여야 합니다.

Note

AWS IoT SiteWise 게이트웨이를 포함하는 지정된 전체 S3 버킷에 액세스할 수 있습니다. 버퍼링된 데이터 수집에는 전용 버킷을 사용하는 것이 좋습니다.

유형: 문자열

필수 항목 여부: 예

publishPolicy

게시 정책의 세부 정보입니다.

유형: 객체

필수 여부: 예

publishFrequency

SiteWise Edge 게이트웨이가 Amazon S3 버킷에 게시하는 빈도입니다. Amazon S3에 대한 데이터 업로드 빈도는 0분보다 크고 30일 이하여야 합니다. 게시 빈도 기간을

지정할 d 때 mh, 및를 사용할 수 있습니다. m은 분, h는 시간, d는 일을 나타냅니다. 기본값은 15분입니다.

유형: 문자열

필수 항목 여부: 예

`localSizeLimitGB`

로컬 디스크에 기록된 파일의 최대 크기는 GB입니다. 이 임계값을 위반하면 게시자는 버퍼링된 모든 데이터를 대상에 게시합니다.

유형: 정수

필수 여부: 예

`siteWiseImportPolicy`

데이터를 가져오기 위한 가져오기 정책의 세부 정보입니다 AWS IoT SiteWise.

유형: 객체

필수 여부: 예

`enableSiteWiseStorageImport`

Amazon S3 버킷에서 AWS IoT SiteWise 스토리지로 데이터를 가져오려면이 값을 `true`로 설정합니다. 처음에는 데이터 사본을 만듭니다 AWS IoT SiteWise. 그런 다음을 `trueenableDeleteAfterImport`로 설정하면에 복사한 후 S3의 데이터가 삭제됩니다 AWS IoT SiteWise. 요금 영향이 적용됩니다. 기본값은 `true`입니다.

유형: 부울

필수 여부: 예

`enableDeleteAfterImport`

AWS IoT SiteWise 스토리지에 수집한 후 Amazon S3 버킷의 파일을 삭제하려면이 값을 `true`로 설정합니다. 기본값은 `true`입니다.

유형: 부울

필수 여부: 예

bulkImportJobRoleArn

데이터 수집 중에는 Amazon S3에서 버퍼링된 데이터를 읽도록 AWS IoT SiteWise 수입하는 IAM 역할의 ARN입니다. 이 역할은 엣지 디바이스가 AWS IoT SiteWise APIs 호출하여 대량 가져오기 프로세스를 시작할 때 사용됩니다.

Note

enableSiteWiseStorageImport가 로 설정된 경우 true이 파라미터가 필요합니다.

유형: 문자열

필수 항목 여부: 아니요

대상에 대한 경로 필터를 추가합니다. 자세한 내용은 [AWS IoT SiteWise Edge 대상에 경로 필터 추가 단원](#)을 참조하십시오.

AWS IoT SiteWise Edge 대상에 경로 필터 추가

대상에 경로 필터를 추가합니다. 경로 필터는 MQTT 주제 구문을 사용합니다. 여기서 #는 레벨 수와 일치하는 와일드카드 문자이고 +는 단일 레벨과 일치하는 와일드카드 문자입니다. 게이트웨이에 여러 대상을 추가할 수 있으며, 각 대상에는 장비 원격 측정을 구독하는 고유한 경로 필터 세트가 있습니다.

Siemens Industrial Edge 게이트웨이는 호환성을 위해 접두사를 사용합니다. 자세한 내용은 [경로 필터의 접두사](#) 단원을 참조하십시오.

Console

경로 필터를 추가하려면

1. [AWS IoT SiteWise 콘솔](#)을 엽니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 경로 필터를 추가할 게이트웨이를 선택합니다.
4. 경로 필터 섹션의 대상 추가에서 경로 필터 추가를 선택합니다.
5. 이 대상이 구독할 경로 필터를 입력합니다. 와일드카드 문자(# 및 +)를 사용하여 여러 경로를 구독할 수 있습니다.

6. 경로 필터 추가를 선택하여 경로 필터를 목록에 추가합니다.
7. 필요한 경우 단계를 반복하여 경로 필터를 추가합니다.
8. 필요한 경로 필터를 모두 추가했으면 생성을 선택합니다.

AWS CLI for self-hosted gateways

Example : 경로 필터 구성

```
{
  "destinations": [
    {
      ...
    }
  ],
  "filters": [
    {
      "type": "PATH",
      "config": {
        "paths": [
          "home/+/sensor1/temperature",
          "home/livingroom/sensor1/temperature",
          "home/livingroom/sensor1/temperature",
          "building/#"
        ]
      }
    }
  ]
}
```

AWS CLI for Siemens IEGateways

Example : 경로 필터의 접두사 구성

각 경로 필터에 대한 데이터(ie/d) 및 메타데이터(ie/m) 접두사를 모두 사용하여 모든 데이터를 캡처합니다.

```
{
  "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

경로 필터 목록을 다운로드하여 대상 간에 경로 필터를 복사합니다. 자세한 내용은 [대상의 모든 경로 필터 다운로드\(콘솔\)](#) 단원을 참조하십시오.

경로 필터를 대량으로 업로드

경로 필터를 대량으로 업로드하려면 CSV 또는 텍스트 파일을 사용합니다. 파일을 업로드할 때가 정확한 중복을 AWS IoT SiteWise 자동으로 제거합니다. 예를 들어, windfarm/site1/ 및 windfarm/site1/는 문자열이 정확히 동일하기 때문이 AWS IoT SiteWise 포착하는 정확한 중복입니다. 부분 중복은 제거되지 않으며 추가 요금이 발생합니다. 예를 들어, windfarm/# 및 windfarm/site1/는 이미 포함되어 있기 때문에 주제 windfarm/site1가 중복됩니다 windfarm/#.

Note

추가 요금을 방지하기 위해 중복을 방지합니다. 업로드된 파일은 .csv 또는 .txt 형식이어야 합니다. 헤더는 포함할 수 없으며 단일 열로 구성되어야 합니다. 열에서 경로 필터를 나열하고 각 필터를 별도의 줄에 표시합니다. 파일에 다른 정보를 포함해서는 안 됩니다.

파일 업로드 요구 사항

추가 경로 필터 요구 사항입니다.

- .csv 또는 .txt 파일 하나를 업로드할 수 있습니다. 다른 파일 형식은 지원되지 않습니다.
- CSV(.csv) 파일은 헤더를 가질 수 없으며 하나의 열만 포함해야 합니다.
- 각 줄에 하나의 경로 필터가 있을 수 있습니다.
- 업로드된 파일은 비워둘 수 없습니다.
- 를 와일드카드#로 사용하는 경우 주제 필터의 마지막 문자여야 합니다. 예를 들어, topic/# 또는 특정 주제 수준에서 독립 실행형 캐릭터로 사용할 수 있습니다. 그러나와 같은 주제 수준 이름 내에서 일반 문자로도 사용할 # 수 있습니다factory/machine#1/topic. 자세한 내용은 [경로 필터 이름의 특수 문자](#) 섹션을 참조하세요.
- + 캐릭터를 사용할 수도 있습니다. 예를 들어 factory/+temp를 사용하여 공장의 모든 온도를 factory/machine2/temp 및 factory/machine3/temp 개별적으로 가져옵니다.

AWS IoT SiteWise 엣지 대상 관리

대상을 추가한 후 대상 구성 편집, 대상 삭제, 경로 필터 관리 등 다양한 작업을 수행하여 대상을 관리할 수 있습니다.

대상 편집

테이블의 대상 옆에 있는 라디오 버튼을 선택하고 편집 버튼을 선택하여 대상을 편집합니다.

Console

AWS IoT SiteWise 콘솔을 사용하여 대상을 편집하려면

1. [AWS IoT SiteWise 콘솔](#)을 엽니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 적절한 게이트웨이를 선택합니다.
4. 대상 섹션에서 편집할 대상을 선택한 다음 편집을 선택합니다.
5. 대상을 수정한 다음 저장을 선택합니다.

AWS CLI

를 사용하여 대상을 편집하려면 AWS CLI

- JSON 기능 구성 정보를 수정하여 대상을 편집할 수 있습니다.

```
aws iotsitewise update-gateway-capability-configuration \
--gateway-id your-gateway-id \
--capability-namespace "iotsitewise:publisher:3" \
--capability-configuration '{
  "sources": [
    {
      "type": "MQTT"
    }
  ],
  "destinations": [
    {
      "id": "your-existing-destination-id",
      "type": "SITEWISE_REALTIME",
      "name": "your-updated-destination-name",
      "config": {
        "publishingOrder": "TIME_ORDER",
        "enableCompression": true,
        "dropPolicy": {
          "cutoffAge": "10d",
          "exportPolicy": {
            "retentionPeriod": "10d",
            "rotationPeriod": "6h",
            "exportSizeLimitGB": 10
          }
        }
      },
      "maxBatchWaitTime": "15s"
    }
  ],
  "filters": [
    {
      ...
    }
  ]
}
```

Note

대상 type 또는는 업데이트할 수 없습니다capability-namespace. 예를 들어의 유형에서 SITEWISE_REALTIME 로 전환할 수 없습니다SITEWISE_BUFFERED. 각 MQTT 지원 게이트웨이에 대해 하나의 실시간 대상을 가질 수 있습니다.

대상 삭제

대상이 더 이상 필요하지 않은 경우 SiteWise Edge 게이트웨이에서 삭제할 수 있습니다.

Console

AWS IoT SiteWise 콘솔을 사용하여 대상을 삭제하려면

1. [AWS IoT SiteWise 콘솔](#)을 엽니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 적절한 게이트웨이를 선택합니다.
4. 대상 섹션에서 삭제할 대상을 선택한 다음 삭제를 선택합니다. 확인 화면이 나타납니다.
5. 대상 삭제 선택을 확인하려면 확인 상자에 "delete"을 입력합니다.

AWS CLI

를 사용하여 대상을 삭제하려면 AWS CLI

- 게이트웨이 ID를 지정하고 삭제하려는 대상을 제거하도록 기능 구성을 수정하여 게이트웨이 기능 구성을 삭제합니다.

```
aws iotsitewise update-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
  --capability-configuration '{
    "sources": [
      {
        "type": "MQTT"
      }
    ],
    "destinations": []
  }'
```

Note

대상 배열은 비어 있을 수 있지만([]) 대상 객체 자체가 기능 구성에 포함되어야 합니다.

대상의 모든 경로 필터 다운로드(콘솔)

AWS IoT SiteWise 콘솔에서 모든 경로 필터가 포함된 CSV 파일을 다운로드합니다. 다운로드한 경로 필터 목록을 사용하여 게이트웨이 대상 간에 경로 필터 목록을 쉽게 공유할 수 있습니다.

AWS IoT SiteWise 콘솔을 사용하여 모든 경로 필터의 CSV 파일을 다운로드하려면

1. [AWS IoT SiteWise 콘솔](#)을 엽니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 경로 필터가 포함된 게이트웨이를 선택합니다.
4. 대상 추가 또는 대상 편집을 선택합니다.
5. 경로 필터 섹션으로 이동하여 CSV 다운로드를 선택합니다.

Note

CSV 파일에는 경로 필터 목록에서 선택한 경로 필터와 관계없이 특정 대상의 모든 경로 필터가 포함됩니다.

경로 필터 편집

개별 경로 필터를 편집하여 대상이 수신하는 데이터를 세분화할 수 있습니다.

Console

AWS IoT SiteWise 콘솔을 사용하여 각 텍스트 상자 내에서 각 개별 경로 필터를 편집할 수 있습니다.

AWS IoT SiteWise 콘솔을 사용하여 경로 필터를 편집하려면

1. [AWS IoT SiteWise 콘솔](#)을 엽니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.

3. 경로 필터가 포함된 게이트웨이를 선택합니다.
4. 적절한 대상을 선택합니다.
5. 편집을 선택합니다.
6. 편집하려는 경로 필터가 포함된 행의 텍스트 상자를 선택합니다.
7. 경로 필터의 텍스트를 업데이트하여 편집된 경로 필터의 확인란이 선택되어 있는지 확인합니다.
8. 저장을 선택합니다.

AWS CLI

를 사용하여 대상의 경로 필터를 편집하려면 AWS CLI 먼저 현재 구성을 검색하고 수정한 다음 `update-gateway-capability-configuration` 명령을 사용하여 업데이트합니다.

를 사용하여 경로 필터를 편집하려면 AWS CLI

1. 현재 기능 구성을 검색합니다.

```
aws iotsitewise describe-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
  --query "capabilityConfiguration"
```

2. JSON을 편집하여 필요에 따라 경로 필터를 수정합니다.
3. 수정된 경로 필터로 기능 구성을 업데이트합니다.

```
aws iotsitewise update-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
  --capability-configuration json-containing-your-updated-path-filters
```

경로 필터 삭제

대상의 경로 필터를 삭제하여 MQTT 소스 및 데이터 처리 파이프라인에서 수신하는 데이터를 제어할 수 있습니다.

Console

AWS IoT SiteWise 콘솔을 사용하여 경로 필터를 삭제하려면

1. [AWS IoT SiteWise 콘솔](#)을 엽니다.
2. 왼쪽 탐색의 엣지 섹션에서 엣지 게이트웨이를 선택합니다.
3. 경로 필터가 포함된 게이트웨이를 선택합니다.
4. 적절한 대상을 선택합니다.
5. 편집을 선택합니다.
6. 대상 편집 화면의 경로 필터 섹션에서 삭제할 경로 필터를 하나 이상 선택합니다.
7. 삭제를 선택합니다. 삭제 확인 메시지가 나타납니다. 경로 필터 삭제를 계속하려면 확인 화면에서 삭제를 선택합니다.

AWS CLI

를 사용하여 대상을 삭제하려면 AWS CLI

- 기능 구성에서 경로 필터를 제거하여 삭제합니다.

```
aws iotsitewise update-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
  --capability-configuration '{
    "sources": [
      {
        "type": "MQTT"
      }
    ],
    "destinations": [
      {
        "id": "your-destination-id",
        "type": "SITEWISE_REALTIME",
        "name": "your-destination-name",
        "config": {
          ...
        },
        "filters": [
          {
            "type": "PATH",
            "config": {
```

```

      "paths": [
        "/path1",
        "/path2",
        "/delete-a-path-to-remove-it"
      ]
    }
  ]
}

```

Note

필터 배열은 비어 있을 수 있지만([]) 필터 객체 자체가 기능 구성에 포함되어야 합니다.

SiteWise Edge 게이트웨이 관리

AWS IoT SiteWise 콘솔 및 API 작업을 사용하여 AWS IoT SiteWise Edge 게이트웨이를 관리할 수 있습니다. [AWS OpsHub Windows AWS IoT SiteWise 용](#) 애플리케이션을 사용하여 로컬 디바이스에서 SiteWise Edge 게이트웨이의 일부 측면을 관리할 수도 있습니다.

AWS IoT SiteWise 애플리케이션 AWS OpsHub 용을 사용하여 로컬 디바이스의 디스크 사용량을 모니터링하는 것이 좋습니다. 또한 Gateway.AvailableDiskSpace 및 Gateway.UsedPercentageDiskSpace Amazon CloudWatch 지표를 모니터링하고 디스크 공간이 부족해질 때 알림을 받기 위한 경보를 생성할 수 있습니다. Amazon CloudWatch 경보에 대한 자세한 내용은 [정적 임계값을 기반으로 CloudWatch 경보 생성](#)을 참조하세요.

디바이스에 향후 데이터를 저장할 공간이 충분한지 확인합니다. 로컬 디바이스의 공간이 거의 부족해지면 서비스는 가장 오래된 타임스탬프가 포함된 소량의 데이터를 자동으로 삭제하여 앞으로 데이터를 저장할 공간을 확보합니다.

서비스가 데이터를 삭제했는지 확인하려면 다음을 수행합니다.

1. AWS IoT SiteWise 애플리케이션을 AWS OpsHub 위해에 로그인합니다.
2. 설정을 선택합니다.
3. 로그의 경우 시간 범위를 지정한 다음 다운로드를 선택합니다.

4. 로그 파일 압축을 끕니다.
5. 로그 파일에 다음 메시지가 포함된 경우 서비스가 데이터를 삭제한 것입니다. SiteWise Edge 게이트웨이 스토리지의 공간이 부족해지는 것을 방지하기 위해 ##바이트의 데이터가 삭제되었습니다.

AWS IoT SiteWise 콘솔을 사용하여 SiteWise Edge 게이트웨이 관리

AWS IoT SiteWise 콘솔을 사용하여 AWS 계정의 모든 SiteWise Edge 게이트웨이를 구성, 업데이트 및 모니터링할 수 있습니다.

[AWS IoT SiteWise 콘솔](#)의 엣지 게이트웨이 페이지로 이동하여 SiteWise Edge 게이트웨이를 볼 수 있습니다. 특정 게이트웨이의 엣지 게이트웨이 세부 정보 페이지에 액세스하려면 엣지 게이트웨이의 이름을 선택합니다.

엣지 게이트웨이 세부 정보 페이지의 개요 탭에서 다음을 수행할 수 있습니다.

- 데이터 소스 섹션에서 데이터 소스 구성을 업데이트하고 추가 데이터 소스를 구성합니다.
- Open CloudWatch 지표를 선택하여 CloudWatch 지표 콘솔에서 데이터 소스당 수집된 데이터 포인트 수를 확인합니다.
- 엣지 기능 섹션에서 편집을 클릭하여 SiteWise Edge 게이트웨이에 데이터 팩을 추가합니다.
- 게이트웨이 구성 섹션에서 SiteWise Edge 게이트웨이의 연결 상태를 확인합니다.
- 게시자 구성 섹션에서 게시자 구성 요소의 SiteWise Edge 게이트웨이 동기화 상태 및 구성을 확인합니다 AWS IoT SiteWise .

엣지 게이트웨이 세부 정보 페이지의 업데이트 탭에서 엣지 게이트웨이에 배포된 현재 구성 요소 및 팩 버전을 볼 수 있습니다. 또한 새 버전을 사용할 수 있을 때 여기에서 배포할 수 있습니다.

용를 사용하여 SiteWise Edge 게이트웨이 관리 AWS OpsHubAWS IoT SiteWise

AWS IoT SiteWise 애플리케이션 AWS OpsHub 용를 사용하여 자체 호스팅 SiteWise Edge 게이트웨이를 관리하고 모니터링합니다. 이 애플리케이션은 다음과 같은 모니터링 및 관리 옵션을 제공합니다.

- 개요에서 다음을 수행할 수 있습니다.
 - SiteWise Edge 게이트웨이 디바이스 데이터에 대한 인사이트를 얻고, 문제를 식별하고, SiteWise Edge 게이트웨이 성능을 개선하는 데 도움이 되는 SiteWise Edge 게이트웨이 세부 정보를 확인합니다.

- 엣지에 있는 로컬 서버 및 장비의 데이터를 모니터링하는 SiteWise Monitor 포털을 확인하세요. 자세한 내용은 AWS IoT SiteWise Monitor 사용 설명서에서 [AWS IoT SiteWise Monitor란 무엇입니까?](#)를 참조하세요.
- 상태에는 SiteWise Edge 게이트웨이의 데이터를 표시하는 대시보드가 있습니다. 프로세스 엔지니어와 같은 도메인 전문가가 대시보드를 사용하여 SiteWise Edge 게이트웨이 동작의 개요를 볼 수 있습니다.
- 자산에서 로컬 디바이스에 배포된 자산과 자산 속성에 대해 수집 또는 계산된 마지막 값을 볼 수 있습니다.
- 설정에서 다음을 수행합니다.
 - 데이터 처리 팩이 설치된 경우 SiteWise Edge 게이트웨이 구성 정보를 보고 리소스를 AWS 클라우드와 동기화합니다.
 - 다른 도구를 사용하여 SiteWise Edge 게이트웨이에 액세스하는 데 사용할 수 있는 인증 파일을 다운로드합니다.
 - SiteWise Edge 게이트웨이 문제를 해결하는 데 사용할 수 있는 로그를 다운로드합니다.
 - SiteWise Edge 게이트웨이에 배포된 AWS IoT SiteWise 구성 요소를 확인합니다.

Important

를 사용하려면 다음이 필요합니다 AWS OpsHub AWS IoT SiteWise.

- 로컬 디바이스와 AWS IoT SiteWise 애플리케이션 AWS OpsHub 용가 동일한 네트워크에 연결되어 있어야 합니다.
- 데이터 처리 팩을 활성화해야 합니다.

를 사용하여 SiteWise Edge 게이트웨이를 관리하려면 AWS OpsHub

1. [AWS OpsHubAWS IoT SiteWise for Windows](#) 애플리케이션을 다운로드하여 설치합니다.
2. 애플리케이션을 엽니다.
3. 게이트웨이에 대한 로컬 자격 증명이 설정되어 있지 않은 경우 [로컬 운영 체제 자격 증명을 사용하여 SiteWise Edge 게이트웨이에 액세스](#)의 단계에 따라 설정합니다.
4. Linux 또는 Lightweight Directory Access Protocol(LDAP) 자격 증명으로 SiteWise Edge 게이트웨이에 로그인할 수 있습니다. SiteWise Edge 게이트웨이에 로그인하려면 다음 중 하나를 수행하세요.

Linux

1. 호스트 이름 또는 IP 주소에 로컬 디바이스의 호스트 이름 또는 IP 주소를 입력합니다.
2. 인증에서 Linux를 선택합니다.
3. 사용자 이름에는 Linux 운영 체제의 사용자 이름을 입력합니다.
4. 암호에는 Linux 운영 체제의 비밀번호를 입력합니다.
5. 로그인을 선택합니다.

LDAP

1. 호스트 이름 또는 IP 주소에 로컬 디바이스의 호스트 이름 또는 IP 주소를 입력합니다.
2. 인증에서 LDAP를 선택합니다.
3. 사용자 이름에 LDAP 사용자 이름을 입력합니다.
4. 비밀번호에 LDAP 비밀번호를 입력합니다.
5. 로그인을 선택합니다.

로컬 운영 체제 자격 증명을 사용하여 SiteWise Edge 게이트웨이에 액세스

Lightweight Directory Access Protocol(LDAP) 외에도 Linux 또는 Windows 자격 증명을 사용하여 자체 호스팅된 SiteWise Edge 게이트웨이에 액세스할 수 있습니다.

Important

Linux 자격 증명으로 SiteWise Edge 게이트웨이에 액세스하려면 SiteWise Edge 게이트웨이의 데이터 처리 팩을 활성화해야 합니다.

Linux 운영 체제 자격 증명을 사용하여 SiteWise Edge 게이트웨이에 액세스

다음 단계에서는 Ubuntu가 설치된 디바이스를 사용한다고 가정합니다. 다른 Linux 배포판을 사용하는 경우 해당 디바이스의 관련 설명서를 참조하세요.

Linux 사용자 풀을 생성하려면 다음을 수행하세요.

1. 다음 명령을 실행하여 관리자 그룹을 생성합니다.

```
sudo groupadd --system SWE_ADMIN_GROUP
```

SWE_ADMIN_GROUP 그룹의 사용자는 SiteWise Edge 게이트웨이에 대한 관리자 액세스를 허용할 수 있습니다.

2. 다음 명령을 실행하여 사용자 그룹을 생성합니다.

```
sudo groupadd --system SWE_USER_GROUP
```

SWE_USER_GROUP 그룹의 사용자는 SiteWise Edge 게이트웨이에 대한 읽기 전용 액세스를 허용할 수 있습니다.

3. 관리자 그룹에 사용자를 추가하려면 다음 명령을 실행합니다. **### ##**과 **####**를 추가하려는 사용자 이름 및 비밀번호로 변경합니다.

```
sudo useradd -p $(openssl passwd -1 password) user-name
```

4. SWE_ADMIN_GROUP 또는 SWE_USER_GROUP에 사용자를 추가하려면 **### ##**을 이전 단계에서 추가한 사용자 이름으로 바꾸세요.

```
sudo usermod -a -G SWE_ADMIN_GROUP user-name
```

이제 사용자 이름과 암호를 사용하여 AWS IoT SiteWise 애플리케이션 AWS OpsHub 용에서 SiteWise Edge 게이트웨이에 로그인할 수 있습니다.

Windows 자격 증명을 사용하여 SiteWise Edge 게이트웨이에 액세스

다음 단계에서는 Windows가 설치된 디바이스를 사용한다고 가정합니다.

Important

보안은 AWS 와 사용자 간의 공동 책임입니다. 대문자, 소문자, 숫자 및 기호 조합을 사용하여 12자 이상의 강력한 비밀번호 정책을 생성합니다. 또한 포트 443에서 들어오는 트래픽은 허용하고 다른 모든 포트에서 들어오는 트래픽은 차단하도록 Windows 방화벽 규칙을 설정합니다.

Windows Server 사용자 풀을 만들려면

1. PowerShell을 관리자로 실행합니다.

- a. SiteWise Edge 게이트웨이를 설치하려는 Windows 서버에 관리자로 로그인합니다.
 - b. Windows 검색창에 PowerShell을 입력합니다.
 - c. 검색 결과에서 Windows PowerShell 앱을 마우스 오른쪽 버튼으로 클릭합니다. [Run as Administrator]를 선택합니다.
2. 다음 명령을 실행하여 관리자 그룹을 생성합니다.

```
net localgroup SWE_ADMIN_GROUP /add
```

SWE_ADMIN_GROUP 그룹에 속한 사용자만 SiteWise Edge 게이트웨이에 대한 관리자 액세스를 허용할 수 있습니다.

3. 다음 명령을 실행하여 사용자 그룹을 생성합니다.

```
net localgroup SWE_USER_GROUP /add
```

SiteWise Edge 게이트웨이에 대한 읽기 전용 액세스를 허용하려면 SWE_USER_GROUP 그룹 내 사용자여야 합니다.

4. 사용자를 추가하려면 다음 명령을 실행합니다. **### ##**과 **####**를 생성하려는 사용자 이름 및 비밀번호로 바꾸세요.

```
net user user-name password /add
```

5. 관리자 그룹에 사용자를 추가하려면 다음 명령을 실행합니다. **### ##**을 추가하려는 사용자 이름으로 바꿉니다.

```
net localgroup SWE_ADMIN_GROUP user-name /add
```

이제 사용자 이름과 암호를 사용하여 AWS IoT SiteWise 애플리케이션 AWS OpsHub 용에서 SiteWise Edge 게이트웨이에 로그인할 수 있습니다.

SiteWise Edge 게이트웨이 인증서 관리

SiteWise Edge 게이트웨이 디바이스에서 SiteWise Monitor 및 Grafana와 같은 타사 애플리케이션을 사용할 수 있습니다. 이러한 애플리케이션을 사용하려면 서비스에 대한 TLS 연결이 필요합니다. SiteWise Edge 게이트웨이는 현재 서비스 서명 인증서를 사용합니다. 브라우저를 사용하여 SiteWise

Monitor 포털과 같은 애플리케이션을 여는 경우 신뢰할 수 없는 인증서에 대한 경고 메시지가 표시될 수 있습니다.

다음은 AWS IoT SiteWise 애플리케이션 AWS OpsHub 용에서 신뢰할 수 있는 인증서를 다운로드하는 방법을 보여줍니다.

1. 애플리케이션에 로그인합니다.
2. 설정을 선택합니다.
3. 인증에서 인증서 다운로드를 선택합니다.

다음은 Google 크롬이나 FireFox를 사용한다고 가정합니다. 다른 브라우저를 사용하는 경우 해당 브라우저의 관련 설명서를 참조하세요. 이전 단계에서 다운로드한 인증서를 브라우저에 추가하려면 다음 중 하나를 수행합니다.

- Google 크롬을 사용하는 경우 Google 크롬 엔터프라이즈 도움말 문서의 [인증서 설정](#)을 따르세요.
- Firefox를 사용하는 경우 Oracle 설명서의 [Mozilla 또는 Firefox 브라우저에 인증서를 로드하는 방법](#)을 따르세요.

SiteWise Edge 게이트웨이 구성 요소 팩 버전 변경

AWS IoT SiteWise 콘솔을 사용하여 SiteWise Edge 게이트웨이에서 구성 요소 팩의 버전을 변경할 수 있습니다.

SiteWise Edge 게이트웨이 구성 요소 팩의 버전을 변경하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 게이트웨이를 선택합니다.
3. 팩 버전을 변경하려는 SiteWise Edge 게이트웨이를 선택합니다.
4. 게이트웨이 구성에서 소프트웨어 버전 보기를 선택합니다.
5. 소프트웨어 버전 편집 페이지에서 버전을 업데이트하려는 팩에 대해 배포할 버전을 선택하고 배포를 선택합니다.
6. 완료를 선택합니다.

SiteWise Edge 게이트웨이 나열

Console

SiteWise Edge 게이트웨이를 나열하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 모든 SiteWise Edge 게이트웨이 목록을 봅니다.

AWS CLI

를 사용하여 게이트웨이를 나열하려면 다음 단계를 AWS CLI 따릅니다.

- list-gateways 명령을 사용하여 모든 게이트웨이를 봅니다.

```
aws iotsitewise list-gateways
```

이 명령은 게이트웨이 목록을 IDs, 이름 및 기타 정보와 함께 반환합니다.

페이지 매김 파라미터를 지정할 수도 있습니다.

```
aws iotsitewise list-gateways --max-results 10 --next-token your-token
```

자세한 내용은 AWS CLI 명령 참조의 [list-gateways](#)를 참조하세요.

SiteWise Edge 게이트웨이 설명

Console

게이트웨이 세부 정보를 보려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 세부 정보를 보려는 게이트웨이의 이름을 선택합니다.
4. 엣지 게이트웨이 세부 정보 페이지에서 게이트웨이 세부 정보를 봅니다.

AWS CLI

를 사용하여 특정 게이트웨이에 대한 자세한 정보를 가져오려면 다음 단계를 AWS CLI 따르세요.

- describe-gateway 명령을 게이트웨이 ID와 함께 사용합니다.

```
aws iotsitewise describe-gateway --gateway-id a1b2c3d4-5678-90ab-  
cdef-1a1a1EXAMPLE
```

이 명령은 게이트웨이에 대한 자세한 정보를 반환합니다.

자세한 내용은 AWS CLI 명령 참조의 [describe-gateway](#)를 참조하세요.

SiteWise Edge 게이트웨이 생성

Console

SiteWise Edge 게이트웨이를 생성하려면

- [AWS IoT SiteWise 콘솔](#)로 이동합니다.
- 탐색 창에서 엣지 게이트웨이를 선택합니다.
- 게이트웨이 생성을 선택합니다.
- 게이트웨이의 이름을 입력합니다.
- 게이트웨이의 Greengrass 그룹을 선택합니다.
- 선택적으로 게이트웨이에 태그를 추가합니다.
- 생성(Create)을 선택합니다.

AWS CLI

를 사용하여 새 IoT SiteWise 게이트웨이를 생성하려면 다음 단계를 AWS CLI 따릅니다.

- create-gateway 명령을 사용하여 새 게이트웨이를 생성합니다.

```
aws iotsitewise create-gateway \  
  --gateway-name "NewSiteWiseGateway" \  
  --gateway-platform '{  
    "greengrass": {
```

```

    "groupArn": "arn:aws:greengrass:us-east-1:123456789012:group/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
  }
}' \
--tags '{
  "Environment": "Production",
  "Location": "Factory1"
}'

```

이 명령은 새 게이트웨이의 ID와 ARN을 반환합니다.

```

{
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayArn": "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
}

```

자세한 내용은 AWS CLI 명령 참조의 [create-gateway](#)를 참조하세요.

SiteWise Edge 게이트웨이 업데이트

Console

SiteWise Edge 게이트웨이를 업데이트하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 업데이트할 게이트웨이를 선택합니다.
4. 편집을 선택합니다.
5. 필요에 따라 게이트웨이 이름 또는 기타 설정을 업데이트합니다.
6. 저장을 선택합니다.

AWS CLI

를 사용하여 기존 게이트웨이를 업데이트하려면 다음 단계를 AWS CLI따릅니다.

- `update-gateway` 명령을 사용하여 게이트웨이의 이름을 업데이트합니다.

```
aws iotsitewise update-gateway \
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE \
  --gateway-name "UpdatedGatewayName"
```

이 명령은 성공 시 출력을 생성하지 않습니다.

자세한 내용은 AWS CLI 명령 참조의 [update-gateway](#)를 참조하세요.

게이트웨이 기능 구성 업데이트

Console

게이트웨이 기능 구성을 업데이트하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 업데이트할 게이트웨이의 이름을 선택합니다.
4. 데이터 소스 섹션에서 편집을 선택합니다.
5. 필요에 따라 데이터 소스 구성을 업데이트합니다.
6. 저장을 선택합니다.

AWS CLI

를 사용하여 게이트웨이의 기능 구성을 업데이트하려면 다음 단계를 AWS CLI따릅니다.

- `update-gateway-capability-configuration` 명령을 사용하여 기능 구성을 업데이트합니다.

```
aws iotsitewise update-gateway-capability-configuration \
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE \
  --capability-namespace "iotsitewise:opcuacollector:1" \
  --capability-configuration '{
    "sources": [
      {
        "name": "OPC-UA Server",
        "endpoint": {
          "certificateTrust": {
            "type": "TrustAny"
          }
        }
      }
    ]
  }'
```

```

        },
        "endpointUri": "opc.tcp://10.0.0.1:4840",
        "securityPolicy": "NONE",
        "messageSecurityMode": "NONE",
        "identityProvider": {
            "type": "Anonymous"
        }
    },
    "measurementDataStreamPrefix": ""
}
]
}'

```

이 명령은 기능 네임스페이스와 동기화 상태를 반환합니다.

```

{
  "capabilityNamespace": "iotsitewise:opcuacollector:1",
  "capabilitySyncStatus": "CONFIGURING"
}

```

자세한 내용은 AWS CLI 명령 참조의 [update-gateway-capability-configuration](#)을 참조하세요.

게이트웨이 리소스에 태그 지정

Console

게이트웨이 리소스에 태그를 지정하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 태그를 지정할 게이트웨이의 이름을 선택합니다.
4. 태그 탭을 선택합니다.
5. 태그 관리를 선택합니다.
6. 새 태그 추가를 선택하고 각 태그의 키와 값을 입력합니다.
7. 저장을 선택합니다.

AWS CLI

를 사용하여 게이트웨이에 태그를 추가하려면 다음 단계를 AWS CLI따릅니다.

- `tag-resource` 명령을 사용하여 게이트웨이에 태그를 추가합니다.

```
aws iotsitewise tag-resource \  
  --resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/  
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE" \  
  --tags '{  
    "Department": "Operations",  
    "Project": "FactoryAutomation"  
  }'
```

이 명령은 성공 시 출력을 생성하지 않습니다.

자세한 내용은 AWS CLI 명령 참조의 [tag-resource](#)를 참조하세요.

게이트웨이에 대한 태그 나열

Console

게이트웨이의 태그를 나열하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 태그를 보려는 게이트웨이의 이름을 선택합니다.
4. 태그 탭을 선택합니다.
5. 게이트웨이와 연결된 태그 목록을 봅니다.

AWS CLI

를 사용하여 게이트웨이와 연결된 태그를 나열하려면 다음 단계를 AWS CLI따릅니다.

- `list-tags-for-resource` 명령을 사용하여 게이트웨이의 태그를 나열합니다.

```
aws iotsitewise list-tags-for-resource \  
  --resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/  
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE" \  
  --tags '{  
    "Department": "Operations",  
    "Project": "FactoryAutomation"  
  }'
```

```
--resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
```

이 명령은 게이트웨이와 연결된 태그를 반환합니다.

```
{
  "tags": {
    "Environment": "Production",
    "Location": "Factory1",
    "Department": "Operations",
    "Project": "FactoryAutomation"
  }
}
```

자세한 내용은 AWS CLI 명령 참조의 [list-tags-for-resource](#)를 참조하세요.

게이트웨이에서 태그 제거

Console

게이트웨이에서 태그를 제거하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 태그를 제거할 게이트웨이의 이름을 선택합니다.
4. 태그 탭을 선택합니다.
5. 태그 관리를 선택합니다.
6. 제거할 각 태그 옆에 있는 제거 아이콘(X)을 선택합니다.
7. 저장을 선택합니다.

AWS CLI

를 사용하여 게이트웨이에서 태그를 제거하려면 다음 단계를 AWS CLI 따릅니다.

- `untag-resource` 명령을 사용하여 게이트웨이에서 태그를 제거합니다.

```
aws iotsitewise untag-resource \
```

```
--resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE" \  
--tag-keys '["Project", "Department"]'
```

이 명령은 성공 시 출력을 생성하지 않습니다.

자세한 내용은 AWS CLI 명령 참조의 [untag-resource](#)를 참조하세요.

구성 요소의 AWS IoT SiteWise 버전 업데이트

최신 기능, 성능 개선 및 보안 패치에 액세스할 수 있도록 AWS IoT Greengrass 코어 디바이스의 AWS IoT SiteWise 게이트웨이 구성 요소를 업데이트합니다.

에서 AWS IoT SiteWise 구성 요소를 업데이트하려면 AWS IoT Greengrass

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 편집할 게이트웨이를 선택하고 편집을 선택합니다.
4. 엣지 기능의 소프트웨어 버전에서 사용 가능한 소프트웨어 업데이트를 선택합니다. 소프트웨어 버전 편집 페이지가 나타납니다.
5. 구성 요소 버전을 선택합니다.

Note

사용 가능한 최신 버전을 선택하는 것이 좋습니다. 게이트웨이 구성 요소를 up-to-date 상태로 유지하면 산업 데이터 수집 및 처리를 위한 최적의 기능을 유지하는 데 도움이 됩니다.

6. 배포(Deploy)를 선택합니다. 그러면 AWS IoT Greengrass V2 배포가 시작되어 게이트웨이의 AWS IoT SiteWise 구성 요소가 업데이트됩니다.

SiteWise Edge 게이트웨이 삭제

Console

SiteWise Edge 게이트웨이를 삭제하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 삭제하려는 게이트웨이를 선택합니다.
4. 삭제를 선택합니다.
5. 게이트웨이를 삭제할지 확인하려면 "delete"을 입력한 다음 나타나는 창에서 삭제를 선택합니다.

AWS CLI

를 사용하여 게이트웨이를 삭제하려면 다음 단계를 AWS CLI 따릅니다.

1. 게이트웨이를 나열하여 삭제하려는 게이트웨이의 게이트웨이 ID를 식별합니다.

```
aws iotsitewise list-gateways
```

이 명령은 ID, 이름 및 기타 정보가 포함된 게이트웨이 목록을 반환IDs

```
{
  "gatewaySummaries": [
    {
      "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
      "gatewayName": "ExampleCorpGateway",
      "gatewayCapabilitySummaries": [
        {
          "capabilityNamespace": "iotsitewise:opcuacollector:1",
          "capabilitySyncStatus": "IN_SYNC"
        }
      ],
      "creationDate": 1588369971.457,
      "lastUpdateDate": 1588369971.457
    }
  ]
}
```

2. ID를 지정하여 게이트웨이를 삭제합니다.

```
aws iotsitewise delete-gateway --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

이 명령은 성공 시 출력을 생성하지 않습니다.

Note

게이트웨이를 삭제하면 게이트웨이의 일부 파일이 게이트웨이의 파일 시스템에 남아 있습니다.

3. 게이트웨이가 삭제되었는지 확인하려면 게이트웨이를 다시 나열할 수 있습니다.

```
aws iotsitewise list-gateways
```

삭제된 게이트웨이가 목록에 더 이상 표시되지 않습니다.

자세한 내용은 AWS CLI 명령 참조의 [delete-gateway](#)를 참조하세요.

SiteWise Edge 게이트웨이 백업 및 복원

이 항목에서는 SiteWise Edge 게이트웨이를 복원하고 지표 데이터를 백업하는 방법을 다룹니다. 동일한 시스템에서 SiteWise Edge 게이트웨이가 손상된 데 문제가 발생하여 문제를 해결해야 하는 경우 [SiteWise Edge 게이트웨이 문제 해결](#) AWS IoT SiteWise 설명서를 참조하십시오.

Note

이 주제에서 다루는 지침은 AWS IoT Greengrass V2 버전 2.1.0 이상에 설치된 SiteWise Edge 게이트웨이에 대한 것입니다.

지표 데이터의 일일 백업

새 머신에서 데이터를 전송하거나 복원하려는 경우 백업을 만드는 것이 중요합니다. 데이터를 백업하면 전송 또는 복원 프로세스 중에 운영 데이터가 손실될 위험이 크게 줄어듭니다.

이 섹션은 데이터 처리 팩을 사용하는 게이트웨이에 적용됩니다. 데이터 처리 팩에 대한 자세한 내용은 [SiteWise Edge에서 데이터 처리를 위한 자산 모델 구성](#) 섹션을 참조하세요.

influxdb 폴더 경로는 다음과 같습니다.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/influxdb
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeProcessor\influxdb
```

폴더 전체와 그 아래의 모든 항목을 백업하는 것이 좋습니다.

1.0 SiteWise Edge에서 외부 하드 드라이브 또는 AWS 클라우드로 지표 데이터를 주기적으로 백업하는 것이 좋습니다.

SiteWise Edge 게이트웨이 복원

SiteWise Edge 게이트웨이를 복원하기 전에 게이트웨이에 연결된 모든 엣지 디바이스가 중지되거나 연결이 끊어졌는지 확인합니다.

SiteWise Edge 게이트웨이를 복원하려면 다음 절차를 수행합니다.

1. SiteWise Edge 게이트웨이를 생성할 때 다운로드한 설치 스크립트를 사용하여 새 시스템에서 SiteWise Edge 게이트웨이를 복원합니다. SiteWise Edge 게이트웨이를 설정하려면 [로컬 디바이스에 SiteWise Edge 게이트웨이 소프트웨어 설치](#) 절차를 확인하세요.

설치 스크립트를 분실했거나 찾을 수 없는 경우 [AWS 고객 지원](#)에 문의하세요.
2. SiteWise Edge 게이트웨이가 설치되면 [AWS IoT Greengrass 콘솔](#)에 로그인합니다.
3. 구성 요소를 재배포하려면 관리로 이동한 다음 AWS IoT Greengrass 디바이스에서 Core 디바이스를 선택합니다.
4. AWS IoT Greengrass Core 디바이스 표에서 SiteWise Edge 게이트웨이에 해당하는 Core 디바이스를 선택합니다.
5. 디바이스 페이지에서 배포 탭을 열고 배포 ID를 선택하면 선택한 ID로 배포 페이지가 열립니다.
6. 배포 페이지로 이동한 후 오른쪽 상단의 작업 버튼을 누르고 수정 옵션을 선택하여 새 배포를 시작합니다. 배포를 구성합니다. 배포를 그대로 유지하려면 검토 및 배포로 건너뛰세요.
7. 배포 상태가 Completed가 될 때까지 기다리세요.

 Note

또한 SiteWise Edge의 모든 구성 요소가 완전히 설정되고 실행되는 데 몇 분 정도 걸립니다.

AWS IoT SiteWise 데이터 복원

새 시스템에 데이터를 복원하려면 다음 절차를 사용하세요.

1. influxdb 폴더를 새 컴퓨터에 복사합니다.
2. 터미널에서 다음 명령을 실행하여 SiteWise EdgeProcessor 구성 요소를 중지합니다.

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. 데이터를 백업한 경로를 찾고 다음 명령을 실행합니다.

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. SiteWiseEdgeProcessor 구성 요소를 다시 시작합니다.

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
```

백업 및 복원의 성공 여부 확인

이 절차를 사용하여 백업된 데이터 및 SiteWise Edge 게이트웨이 복원을 검증하세요.

Note

이 절차에서는에 대한 AWS OpsHub를 설치해야 합니다 AWS IoT SiteWise. 자세한 내용은 [용 AWS OpsHub를 사용하여 SiteWise Edge 게이트웨이 관리를 참조하세요 AWS IoT SiteWise.](#)

1. 용 Open AWS OpsHub AWS IoT SiteWise.
2. SiteWise Edge 게이트웨이 설정 페이지에서 구성 요소 표에 나열된 각 구성 요소의 상태를 확인합니다. 상태 색상이 녹색이고 판독값에 RUNNING이 표시되는지 확인합니다.
3. 포털 대시보드에서 과거 데이터를 검증하여 이전 데이터와 새 데이터가 모두 제대로 설정되었는지 확인하세요. 과거 데이터와 새 데이터 사이에는 다운타임이 있을 수 있습니다. 단, 데이터 포인트가 수집되지 않는 기간은 예외입니다.

SiteWise Edge 게이트웨이 백업 또는 복원과 관련된 문제가 발생하는 경우 다음 문제 해결 항목 [AWS IoT SiteWise Edge 게이트웨이 문제 해결](#)을 참조하세요.

레거시 게이트웨이(AWS IoT Greengrass Version 1)

Note

에서 실행되는 SiteWise Edge 게이트웨이 AWS IoT Greengrass V1 는 2021년 7월 29일 이전에 이 기능을 사용하기 시작한 경우에만 사용할 수 있습니다. 를 사용하여 AWS IoT SiteWise

게이트웨이를 실행하는 방법에 대한 자세한 내용은 단원을 [AWS IoT Greengrass V2](#) 참조하십시오. [시오를 사용하여 AWS IoT SiteWise Edge 게이트웨이 자체 호스팅 AWS IoT Greengrass V2.](#)

SiteWise Edge 게이트웨이는 이제 독립적으로 실행되어 산업용 IoT 애플리케이션에 향상된 기능과 향상된 성능을 AWS IoT Greengrass V2 제공합니다. 이 최신 버전은 모듈식 소프트웨어 배포를 지원하는 최신 구성 요소 기반 프레임워크를 기반으로 구축된 아키텍처 진화를 AWS IoT Greengrass V2 나타냅니다. 통합 설치 관리자를 통해 설치를 간소화하는 동시에 개발자에게 사용자 지정 구성 요소를 배포하고 로컬 테스트를 수행할 수 있는 더 큰 유연성을 제공합니다. 구성 요소 기반 모델을 사용하면 리소스 관리를 더 효율적으로 수행할 수 있으며 구성 요소 레시피를 통해 간소화된 구성 접근 방식을 도입할 수 있습니다. 이 설계는 구성 요소 간의 종속성 처리를 용이하게 하고, 지속적인 배포 사례를 지원하며, 로컬 개발을 위한 향상된 CLI 기능을 제공합니다. 또한 이를 통해 구성 관리를 AWS IoT Greengrass V2 중앙 집중화 AWS IoT Core 하고 향상된 로깅 및 모니터링 기능을 제공하며, 모두 보다 세분화된 보안 권한 모델로 보호됩니다.

를 사용하여 SiteWise Edge 게이트웨이를 시작하는 방법에 대한 자세한 내용은 AWS IoT Greengrass V2 섹션을 참조하세요 [AWS IoT SiteWise 엣지 자체 호스팅 게이트웨이 요구 사항](#). 이러한 리소스는 게이트웨이 설정, 데이터 소스 구성 및 산업 IoT 인프라 관리에 대한 step-by-step 지침을 제공합니다.

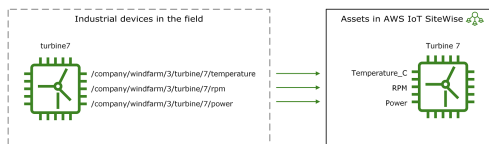
Note

IoT 서비스가 AWS 계속 혁신되고 개선됨에 따라 최신 기능 및 개선 사항을 최신 상태로 유지하는 것이 좋습니다. AWS IoT SiteWise 및 AWS IoT Greengrass 설명서에서 산업 IoT 솔루션을 추가로 최적화할 수 있는 새로운 기능을 정기적으로 확인하세요.

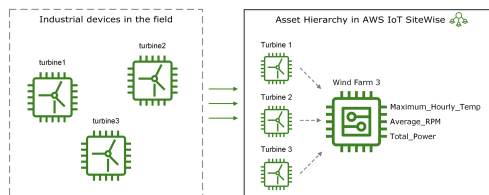
산업 자산 모델링

AWS IoT SiteWise 자산을 사용하여 산업 운영의 가상 표현을 생성할 수 있습니다. 자산은 디바이스, 장비 또는 하나 이상의 데이터 스트림을 AWS 클라우드에 업로드하는 프로세스를 나타냅니다. 예를 들어 자산 디바이스는 공기 온도, 프로펠러 회전 속도 및 전력 출력 시계열 측정을 AWS IoT SiteWise의 자산 속성으로 전송하는 풍력 터빈일 수 있습니다.

각 데이터 스트림은 고유한 속성 별칭에 해당합니다. 예를 들어 별칭 `/company/windfarm/3/turbine/7/temperature`는 풍력 발전소 #3의 터빈 #7에서 나오는 온도 데이터 스트림을 고유하게 식별합니다. 온도 데이터를 섭씨에서 화씨로 변환하는 등 수학 표현식을 사용하여 수신되는 측정 데이터를 변환하도록 AWS IoT SiteWise 자산을 구성할 수 있습니다.



자산은 전체 풍력 단지와 같은 디바이스의 논리적 그룹화를 나타낼 수도 있습니다. 자산을 다른 자산에 연결하여 복잡한 산업 조직을 나타내는 자산 계층 구조를 생성할 수 있습니다. 자산은 연결된 하위 자산 내의 데이터에 액세스할 수 있습니다. 이렇게 하면 AWS IoT SiteWise 표현식을 사용하여 풍력 발전소의 순 전력 출력과 같은 집계 지표를 계산할 수 있습니다.



자산 모델의 모든 자산을 생성해야 합니다. 자산 모델은 자산의 형식을 표준화하는 선언형 구조입니다. 자산 모델은 동일한 유형의 여러 자산에 걸쳐 일관된 정보를 적용하므로 디바이스 그룹을 나타내는 자산의 데이터를 처리할 수 있습니다. 앞의 다이어그램에서는 모든 터빈이 공통된 속성 집합을 공유하므로 세 터빈 모두에 동일한 자산 모델을 사용합니다.

구성 요소 모델을 생성할 수도 있습니다. 구성 요소 모델은 자산 모델 또는 기타 구성 요소 모델에 포함할 수 있는 특수 유형의 자산 모델입니다. 구성 요소 모델을 사용하여 여러 자산 모델에서 공유하는 센서, 모터 등의 일반적인 재사용 가능 하위 어셈블리를 정의할 수 있습니다.

자산 모델을 정의한 후 산업 자산을 생성할 수 있습니다. 자산을 생성하려면 ACTIVE 자산 모델을 선택하여 해당 모델의 자산을 생성합니다. 그런 다음, 데이터 스트림 별칭, 속성 같은 자산별 정보를 채울 수

있습니다. 앞의 다이어그램에서는 하나의 자산 모델에서 3개의 터빈 자산을 생성한 다음 각 터빈에 `/company/windfarm/3/turbine/7/temperature` 같은 데이터 스트림 별칭을 연결합니다.

기존 자산, 자산 모델 및 구성 요소 모델을 업데이트하고 삭제할 수도 있습니다. 자산 모델을 업데이트 하면 해당 자산 모델을 기반으로 하는 모든 자산에 기본 모델의 변경 사항이 반영됩니다. 구성 요소 모델을 업데이트하면 구성 요소 모델을 참조하는 모든 자산 모델을 기반으로 하는 모든 자산에 적용됩니다.

예를 들어 하위 구성 요소가 많은 복잡한 장비를 모델링할 때 자산 모델이 매우 복잡할 수 있습니다. 이러한 자산 모델을 체계적이고 유지 관리 가능하게 하기 위해 사용자 지정 복합 모델을 사용하여 관련 속성을 그룹화하거나 공유 구성 요소를 재사용할 수 있습니다. 자세한 내용은 [사용자 지정 복합 모델\(구성 요소\)](#) 단원을 참조하십시오.

주제

- [자산 및 모델 상태](#)
- [자산 모델 버전](#)
- [사용자 지정 복합 모델\(구성 요소\)](#)
- [AWS IoT SiteWise 객체 IDs 설정](#)
- [에 대한 자산 및 구성 요소 모델 생성 AWS IoT SiteWise](#)
- [에서 자산 모델에 대한 자산 생성 AWS IoT SiteWise](#)
- [에서 자산 검색 AWS IoT SiteWise Console](#)
- [속성 값 업데이트](#)
- [자산 연결 및 연결 해제](#)
- [자산 및 모델 업데이트](#)
- [에서 자산 및 모델 삭제 AWS IoT SiteWise](#)
- [자산 및 모델을 사용한 일괄 작업](#)

자산 및 모델 상태

자산, 자산 모델 또는 구성 요소 모델을 생성, 업데이트 또는 삭제하면 변경 사항이 전파되는 데 시간이 걸립니다. 이러한 작업을 비동기적으로 AWS IoT SiteWise 해결하고 각 리소스의 상태를 업데이트합니다. 각 자산, 자산 모델 및 구성 요소 모델에는 리소스 상태 및 오류 메시지(해당되는 경우)가 포함된 상태 필드가 있습니다. 상태는 다음 값 중 하나일 수 있습니다.

- ACTIVE – 리소스가 활성 상태입니다. 이 상태는 자산, 자산 모델 및 구성 요소 모델을 쿼리하고 상호 작용할 수 있는 유일한 상태입니다.
- CREATING - 리소스가 생성 중입니다.
- UPDATING - 리소스가 업데이트 중입니다.
- DELETING - 리소스가 삭제되는 중입니다.
- PROPAGATING – (자산 모델 및 구성 요소 모델만 해당) 변경 사항이 모든 종속 리소스로 전파됩니다 (자산 모델에서 자산으로, 구성 요소 모델에서 자산 모델로).
- FAILED – 표현식의 순환 참조로 인해 생성 또는 업데이트 작업 중에 리소스를 검증하지 못했습니다. FAILED 상태인 리소스를 삭제할 수 있습니다.

일부 생성, 업데이트 및 삭제 작업은 자산, 자산 모델 또는 구성 요소 모델을 작업이 해결되는 ACTIVE 동안 이외의 상태로 AWS IoT SiteWise 설정합니다. 이러한 작업 중 하나를 수행한 후 리소스를 쿼리하거나 상호 작용하려면 상태가 ACTIVE로 변경될 때까지 기다려야 합니다. 그렇지 않으면 요청이 실패합니다.

주제

- [자산의 상태 확인](#)
- [자산 또는 구성 요소 모델의 상태 확인](#)

자산의 상태 확인

AWS IoT SiteWise 콘솔 또는 API를 사용하여 자산의 상태를 확인할 수 있습니다.

주제

- [자산의 상태 확인\(콘솔\)](#)
- [자산의 상태 확인\(AWS CLI\)](#)

자산의 상태 확인(콘솔)

AWS IoT SiteWise 콘솔에서 자산의 상태를 확인하려면 다음 절차를 따르세요.

자산의 상태를 확인하려면(콘솔) 다음을 수행하세요.

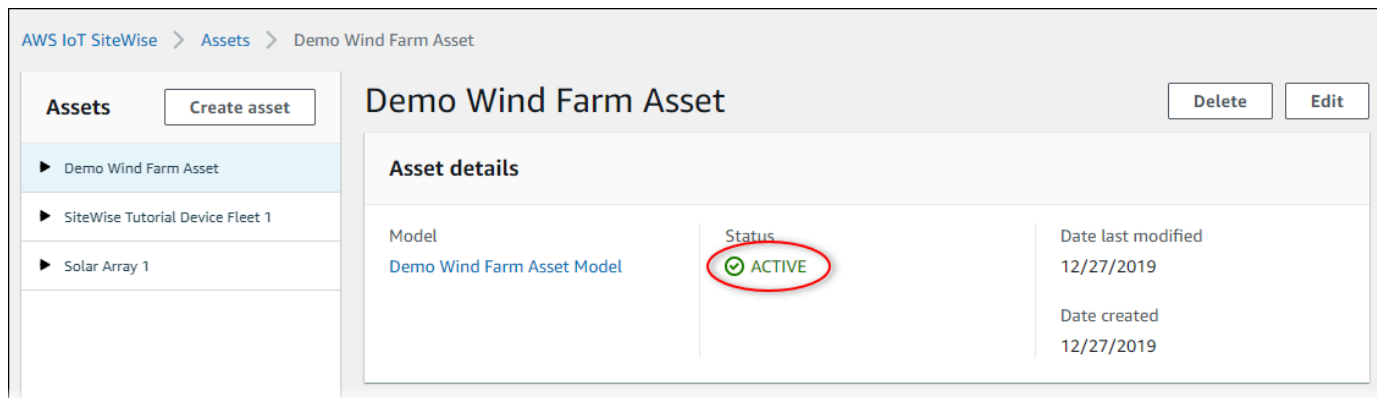
1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.

3. 확인할 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 자산 세부 정보 패널에서 상태를 찾습니다.



자산의 상태 확인(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산의 상태를 확인할 수 있습니다.

자산의 상태를 확인하려면 `assetId` 파라미터와 함께 [DescribeAsset](#) 작업을 사용하세요.

자산의 상태를 확인하려면(AWS CLI) 다음을 수행하세요.

- 다음 명령을 실행하여 자산을 설명합니다. **`asset-id`**를 자산의 ID 또는 외부 ID로 바꿉니다. 외부 ID는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

```
aws iotsitewise describe-asset --asset-id asset-id
```

이 작업은 자산의 세부 정보가 포함된 응답을 반환합니다. 응답에는 다음과 같은 구조의 `assetStatus` 객체가 포함됩니다.

```
{
  ...
  "assetStatus": {
    "state": "String",
    "error": {
```

```
    "code": "String",  
    "message": "String"  
  }  
}  
}
```

자산의 상태는 JSON 객체의 `assetStatus.state`에 있습니다.

자산 또는 구성 요소 모델의 상태 확인

AWS IoT SiteWise 콘솔 또는 API를 사용하여 자산 모델 또는 구성 요소 모델의 상태를 확인할 수 있습니다.

주제

- [자산 모델 또는 구성 요소 모델의 상태 확인\(콘솔\)](#)
- [자산 모델 또는 구성 요소 모델의 상태 확인\(AWS CLI\)](#)

자산 모델 또는 구성 요소 모델의 상태 확인(콘솔)

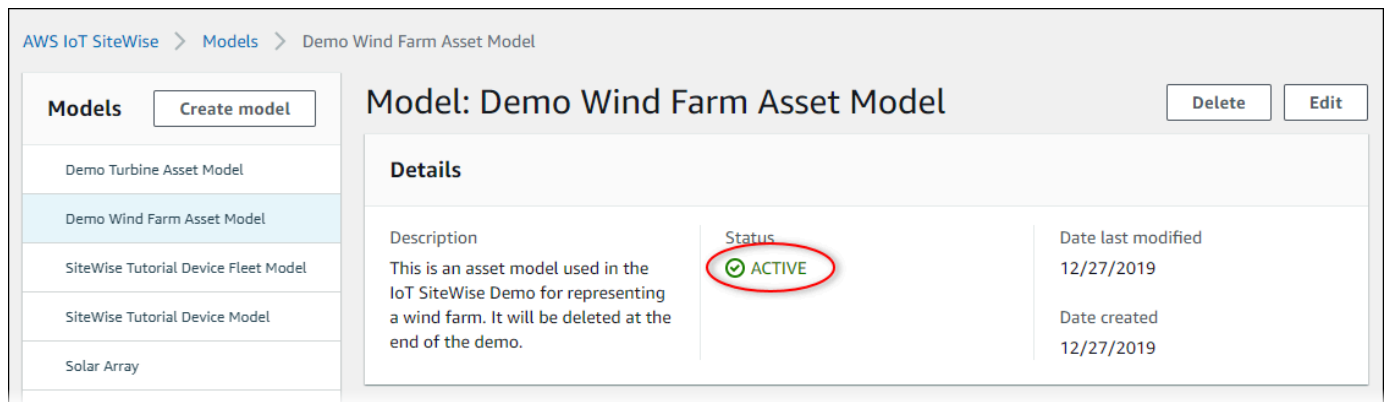
다음 절차에 따라 AWS IoT SiteWise 콘솔에서 자산 모델 또는 구성 요소 모델의 상태를 확인합니다.

Tip

자산 모델과 구성 요소 모델은 모두 탐색 창의 모델 아래에 나열됩니다. 선택한 자산 모델 또는 구성 요소 모델의 세부 정보 패널은 모델이 어떤 유형인지 나타냅니다.

자산 모델 또는 구성 요소 모델의 상태를 확인하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 확인할 모델을 선택합니다.
4. 세부 정보 패널에서 상태를 찾습니다.



자산 모델 또는 구성 요소 모델의 상태 확인(AWS CLI)

AWS CLI 를 사용하여 자산 모델 또는 구성 요소 모델의 상태를 확인할 수 있습니다.

자산 모델 또는 구성 요소 모델의 상태를 확인하려면 `assetModelId` 파라미터와 함께 [DescribeAssetModel](#) 작업을 사용하세요.

Tip

는 구성 요소 모델을 자산 모델의 유형으로 AWS CLI 정의합니다. 따라서 두 유형의 모델에 동일한 [DescribeAssetModel](#) 작업을 사용합니다. 응답의 `assetModelType` 필드는 모델이 `COMPONENT_MODEL`인지 아니면 `ASSET_MODEL`인지를 나타냅니다.

자산 모델 또는 구성 요소 모델의 상태를 확인하려면(AWS CLI) 다음을 수행하세요.

- 다음 명령을 실행하여 모델을 설명합니다. *asset-model-id*를 자산 모델 또는 구성 요소 모델의 ID 또는 외부 ID로 바꿉니다. 외부 ID는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

이 작업은 자산 모델의 세부 사항이 포함된 응답을 반환합니다. 응답에는 다음과 같은 구조의 `assetModelStatus` 객체가 포함됩니다.

```
{
  ...
  "assetModelStatus": {
    "state": "String",
```

```

    "error": {
      "code": "String",
      "message": "String"
    }
  }
}

```

모델의 상태는 JSON 객체의 `assetModelStatus.state`에 있습니다.

자산 모델 버전

AWS IoT SiteWise 는 자산 모델 및 구성 요소 모델에서 생성 및 업데이트 작업의 비동기 처리를 지원합니다. 또한 모델의 상태도 업데이트합니다.

AWS IoT SiteWise 는 생성 및 업데이트 요청에서 유효한 모델의 변경 사항을 종속 리소스(자산 모델에서 자산으로 또는 구성 요소 모델에서 자산 모델로)로 전파합니다. 그런 다음 모델을 ACTIVE 상태로 전환합니다.

제공된 모델 정의가 유효하지 않은 경우는 모델을 FAILED 상태로 AWS IoT SiteWise 둡니다. 변경 사항은 종속 리소스에 전파되지 않습니다. 종속 리소스는 모델이 ACTIVE 상태에 있을 때 전파된 마지막 모델 정의를 나타냅니다.

위의 정보를 기반으로 모델 정의에는 다음과 같은 두 가지 유형의 모델 버전이 있습니다.

1. 최신 버전 - 생성 또는 업데이트 요청의 일부로 허용되는 최신 정의입니다.
2. 활성 버전 - 최신 정의가 성공적으로 처리되었으며 모델 상태는 ACTIVE입니다.

기본적으로 자산 모델 또는 구성 요소 모델에서 설명 API를 직접적으로 호출하면 모델의 최신 버전에 대한 세부 정보가 반환됩니다. 자산 모델 또는 구성 요소 모델의 활성 버전이 필요한 시나리오가 있습니다. 아래 예시 시나리오를 참조하세요.

- 정의가 유효하지 않은 업데이트 작업은 자산 모델을 FAILED 상태로 전환합니다. 자산 모델의 활성 버전을 검색하고 이 유효한 정의를 참조하여 다른 업데이트 요청을 생성하는 방법으로 변경 사항을 되돌려야 합니다.
- 고객이 자산과 해당 자산 모델을 볼 수 있는 애플리케이션이 AWS IoT SiteWise 존재합니다. 사용자가 특정 자산에 해당하는 자산 모델 정의를 참조하고 자산 모델이 임시 UPDATING, PROPAGATING 또는 FAILED 상태에 있는 경우, 최신 버전은 아직 자산에 전파되지 않은 자산 모델 정의를 반환합니다. 이 경우 고객을 위해 자산 모델의 활성 버전을 검색해야 합니다.

주제

- [자산 모델 또는 구성 요소 모델의 활성 버전 검색\(콘솔\)](#)
- [자산 모델 또는 구성 요소 모델의 활성 버전 검색\(AWS CLI\)](#)

자산 모델 또는 구성 요소 모델의 활성 버전 검색(콘솔)

다음 절차에 따라 AWS IoT SiteWise 콘솔에서 자산 모델 또는 구성 요소 모델의 활성 버전을 검색합니다.

Tip

자산 모델과 구성 요소 모델은 모두 탐색 창의 모델 아래에 나열됩니다. 선택한 자산 모델 또는 구성 요소 모델의 세부 정보 패널은 모델이 어떤 유형인지 나타냅니다.

자산 모델 또는 구성 요소 모델의 활성 버전을 검색하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 활성 버전을 검색할 모델을 선택합니다.
 - a. 모델이 ACTIVE 상태인 경우 활성 버전을 보고 있는 것입니다.
 - b. 모델이 일시적 UPDATING, PROPAGATING 또는 FAILED 상태인 경우 세부 정보 패널의 상태 아래에서 활성 버전 보기를 찾습니다.

자산 모델 또는 구성 요소 모델의 활성 버전 검색(AWS CLI)

AWS CLI 를 사용하여 자산 모델 또는 구성 요소 모델의 활성 버전을 검색합니다.

자산 모델 또는 구성 요소 모델의 활성 버전을 검색하려면 `assetModelVersion` 파라미터와 함께 [DescribeAssetModel](#) 작업을 사용합니다.

Tip

는 구성 요소 모델을 자산 모델의 유형으로 AWS CLI 정의합니다. 따라서 두 유형의 모델에 동일한 [DescribeAssetModel](#) 작업을 사용합니다. 응답의 `assetModelType` 필드는 모델이 COMPONENT_MODEL인지 아니면 ASSET_MODEL인지를 나타냅니다.

자산 모델 또는 구성 요소 모델의 활성 버전을 검색하려면(AWS CLI) 다음을 수행하세요.

- 다음 명령을 실행하여 모델을 설명합니다. *asset-model-id*를 자산 모델 또는 구성 요소 모델의 ID 또는 외부 ID로 바꿉니다. 외부 ID는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id --asset-model-version ACTIVE
```

이 작업은 모델의 세부 사항이 포함된 응답을 반환합니다. 응답에는 다음과 같은 구조의 `assetModelStatus` 객체가 포함됩니다.

```
{
  ...
  "assetModelName": "string",
  "assetModelProperties": [ ... ],
  ...,
  "assetModelVersion": "string"
}
```

사용자 지정 복합 모델(구성 요소)

많은 부품이 있는 복잡한 기계와 같이 특히 복잡한 산업 자산을 모델링할 때는 자산 모델을 정리하고 유지 관리하는 것이 어려울 수 있습니다.

이러한 경우 기존 자산 모델 및 구성 요소 모델에 사용자 지정 복합 모델 또는 구성 요소(콘솔을 사용하는 경우)를 추가할 수 있습니다. 이를 통해 관련 속성을 그룹화하고 하위 구성 요소 정의를 재사용하여 정리 상태를 유지할 수 있습니다.

사용자 지정 복합 모델에는 다음과 같은 두 가지 유형이 있습니다.

- 인라인 사용자 지정 복합 모델은 사용자 지정 복합 모델이 속한 자산 모델 또는 구성 요소 모델에 적용되는 그룹화된 속성 집합을 정의합니다. 이를 사용하여 관련 속성을 그룹화합니다. 이 모델은 이름, 설명 및 자산 모델 속성 집합으로 구성됩니다. 또한 재사용이 불가능합니다.
- 구성 요소 모델 기반 사용자 지정 복합 모델은 자산 모델 또는 구성 요소 모델에 포함할 구성 요소 모델을 참조합니다. 이 모델을 사용하여 모델에 표준 하위 어셈블리를 포함합니다. 이 모델은 참조하는 구성 요소 모델의 이름, 설명 및 ID로 구성됩니다. 또한 고유한 속성이 없습니다. 참조된 구성 요소 모델은 생성된 자산에 연결된 속성을 제공합니다.

다음 섹션에서는 설계에서 사용자 지정 복합 모델을 사용하는 방법을 보여줍니다.

주제

- [인라인 사용자 지정 복합 모델](#)
- [구성 요소 모델 기반 사용자 지정 복합 모델](#)
- [경로를 사용하여 사용자 지정 복합 모델 속성 참조](#)

인라인 사용자 지정 복합 모델

인라인 사용자 지정 복합 모델은 관련 속성을 그룹화하여 자산 모델을 구성하는 방법을 제공합니다.

예를 들어 로봇 자산을 모델링한다고 가정해 보겠습니다. 로봇에는 서보 모터, 전원 공급 장치 및 배터리가 포함되어 있습니다. 이러한 각 구성 부품에는 모델에 포함하려는 고유한 속성이 있습니다. 다음과 같은 속성을 가진 `robot_model`이라는 자산 모델을 정의할 수 있습니다.

- `robot_model`
 - `servo_status` (정수)
 - `servo_position` (배정밀도 실수)
 - `powersupply_status` (정수)
 - `powersupply_temperature` (배정밀도 실수)
 - `battery_status` (정수)
 - `battery_charge` (배정밀도 실수)

그러나 경우에 따라 하위 어셈블리가 많거나 하위 어셈블리 자체에 속성이 많을 수 있습니다. 이러한 경우 속성이 너무 많아 이전 예시에서와 같이 모델 루트의 단일 플랫폼 목록에서 참조하고 유지하는 것이 번거로워질 수 있습니다.

이러한 상황을 처리하기 위해 인라인 사용자 지정 복합 모델을 사용하여 속성을 그룹화할 수 있습니다. 인라인 사용자 지정 복합 모델은 자체 속성을 정의하는 사용자 지정 복합 모델입니다. 예를 들어 다음과 같이 로봇을 모델링할 수 있습니다.

- `robot_model`
 - `servo`
 - `status`(정수)

- position(배정밀도 실수)
- powersupply
 - status(정수)
 - temperature (배정밀도 실수)
- battery
 - status(정수)
 - charge(배정밀도 실수)

앞의 예에서 servo, powersupply 및 battery는 robot_model 자산 모델 내에 정의된 인라인 사용자 지정 복합 모델의 이름입니다. 이러한 복합 모델 각각은 자체 속성을 정의합니다.

Note

이 경우 각 사용자 지정 복합 모델은 자체 속성을 정의하여 모든 속성이 자산 모델 자체의 일부가 되도록 합니다(이 경우 robot_model). 이러한 속성은 다른 자산 모델 또는 구성 요소 모델과 공유되지 않습니다. 예를 들어 servo라는 인라인 사용자 지정 복합 모델도 있는 다른 자산 모델을 생성한 경우 robot_model 내에서 servo를 변경해도 다른 자산 모델의 servo 정의에는 영향을 미치지 않습니다.

이러한 공유를 구현하려는 경우(예: servo에 대한 단 하나의 정의를 모든 자산 모델이 공유할 수 있도록 하려는 경우) 대신 해당 구성 요소 모델을 생성한 다음 이를 참조하는 구성 요소 모델 기반 복합 모델을 생성합니다. 자세한 내용은 다음 섹션을 참조하시기 바랍니다.

인라인 사용자 지정 복합 모델의 생성 방법에 대한 자세한 내용은 [사용자 지정 복합 모델\(구성 요소\) 생성](#) 섹션을 참조하세요.

구성 요소 모델 기반 사용자 지정 복합 모델

에서 구성 요소 모델을 생성 AWS IoT SiteWise 하여 재사용 가능한 표준 하위 어셈블리를 정의할 수 있습니다. 구성 요소 모델을 생성한 후에는 다른 자산 모델 및 구성 요소 모델에서 해당 모델에 참조를 추가할 수 있습니다. 구성 요소를 참조하려는 모든 모델에 구성 요소 모델 기반 사용자 지정 복합 모델을 추가하여 이 작업을 수행합니다. 여러 모델의 구성 요소에 참조를 추가하거나 동일한 모델 내에서 여러 번 참조를 추가할 수 있습니다.

이렇게 하면 모델 간에 동일한 정의가 복제되는 것을 방지할 수 있습니다. 또한 구성 요소 모델에 대한 모든 변경 사항은 해당 모델을 사용하는 모든 자산 모델에 반영되므로 모델 유지 관리가 간소화됩니다.

예를 들어 산업용 시설에 모두 동일한 종류의 서보 모터를 사용하는 다양한 유형의 장비가 있다고 가정해 보겠습니다. 그중 일부는 단일 장비에 많은 서보 모터가 있습니다. 각 장비 유형에 대한 자산 모델을 생성하지만 매번 servo의 정의를 복제하고 싶지는 않습니다. 한 번만 모델링하고 다양한 자산 모델에 사용하려고 합니다. 나중에 servo의 정의를 변경하면 모든 모델과 자산에서 업데이트됩니다.

이러한 방식으로 이전 예시의 로봇을 모델링하려면 다음과 같이 서보 모터, 전원 공급 장치 및 배터리를 구성 요소 모델로 정의할 수 있습니다.

- servo_component_model
 - status(정수)
 - position(배정밀도 실수)

- powersupply_component_model
 - status(정수)
 - temperature (배정밀도 실수)

- battery__component_model
 - status(정수)
 - charge(배정밀도 실수)

그런 다음 이러한 구성 요소를 참조하는 robot_model과 같은 자산 모델을 정의할 수 있습니다. 여러 자산 모델은 동일한 구성 요소 모델을 참조할 수 있습니다. 로봇에 여러 서보 모터가 있는 경우와 같이 하나의 자산 모델에서 동일한 구성 요소 모델을 여러 번 참조할 수도 있습니다.

- robot_model
 - servo1(참조: servo_component_model)
 - servo2(참조: servo_component_model)
 - servo3(참조: servo_component_model)
 - powersupply (참조: powersupply_component_model)
 - battery(참조: battery_component_model)

구성 요소 모델을 생성하는 방법에 대한 자세한 내용은 [구성 요소 모델 생성](#) 섹션을 참조합니다.

다른 모델의 구성 요소 모델을 참조하는 방법에 대한 자세한 내용은 [사용자 지정 복합 모델\(구성 요소\) 생성](#) 섹션을 참조하세요.

경로를 사용하여 사용자 지정 복합 모델 속성 참조

자산 모델, 구성 요소 모델 또는 사용자 지정 복합 모델에 속성을 생성할 때 [변환](#) 및 [지표](#)와 같이 해당 값을 사용하는 다른 속성에서 이를 참조할 수 있습니다.

AWS IoT SiteWise 는 속성을 참조할 수 있는 다양한 방법을 제공합니다. 가장 간단한 방법은 속성 ID 를 사용하는 것입니다. 그러나 참조하려는 속성이 사용자 지정 복합 모델에 있는 경우 경로별로 참조하는 것이 더 유용할 수 있습니다.

경로는 자산 모델 및 복합 모델 내 중첩된 복합 모델 간의 위치 측면에서 속성을 지정하는 경로 세그먼트의 순서가 지정된 시퀀스입니다.

속성 경로 가져오기

[AssetModelProperty](#)의 path 필드에서 속성의 경로를 가져올 수 있습니다.

예를 들어 속성이 position인 사용자 지정 복합 모델 servo가 포함된 자산 모델 robot_model이 있다고 가정해 보겠습니다. servo에서 [DescribeAssetModelCompositeModel](#)을 직접적으로 호출하면 position 속성에 다음과 같은 path 필드가 나열됩니다.

```
"path": [
  {
    "id": "asset model ID",
    "name": "robot_model"
  },
  {
    "id": "composite model ID",
    "name": "servo"
  },
  {
    "id": "property ID",
    "name": "position"
  }
]
```

속성 경로 사용

변환 또는 지표와 같은 다른 속성을 참조하는 속성을 정의할 때 속성 경로를 사용할 수 있습니다.

속성은 변수를 사용하여 다른 속성을 참조합니다. 변수 작업에 대한 자세한 내용은 [수식 표현식에서 변수 사용](#) 섹션을 참조하세요.

속성을 참조하도록 변수를 정의할 때 속성의 ID 또는 경로를 사용할 수 있습니다.

참조된 속성의 경로를 사용하는 변수를 정의하려면 해당 값의 `propertyPath` 필드를 지정합니다.

예를 들어 경로를 사용하여 속성을 참조하는 지표가 있는 자산 모델을 정의하려면 다음과 같은 페이로드를 [CreateAssetModel](#)에 전달할 수 있습니다.

```
{
  ...
  "assetModelProperties": [
    {
      ...
      "type": {
        "metric": {
          ...
          "variables": [
            {
              "name": "variable name",
              "value": {
                "propertyPath": [
                  path segments
                ]
              }
            }
          ]
        }
      },
      ...
    },
    ...
  ],
  ...
}
```

AWS IoT SiteWise 객체 IDs 설정

AWS IoT SiteWise 는 자산, 자산 모델, 속성 및 계층 구조와 같은 다양한 유형의 영구 객체를 정의합니다. 이러한 모든 객체에는 해당 객체를 검색, 업데이트 및 삭제하는 데 사용할 수 있는 고유 식별자가 있습니다.

AWS IoT SiteWise 에는 ID 생성을 위한 다양한 고객 옵션이 있습니다.는 기본적으로 객체 AWS IoT SiteWise 생성 시 ID 생성을 생성합니다. 사용자는 객체에 자체 ID를 제공할 수도 있습니다.

주제

- [객체 UUID 작업](#)
- [외부 ID 재사용](#)

객체 UUID 작업

의 모든 영구 객체 AWS IoT SiteWise 에는 이를 식별하는 [UUID](#)가 있습니다. 예를 들어 자산 모델에는 자산 모델 ID가 있고 자산에는 자산 ID가 있는 식입니다. 이 ID는 객체를 생성할 때 할당되며 객체의 수명 동안 변경되지 않습니다.

새 객체를 생성하면가 기본적으로 고유한 ID를 AWS IoT SiteWise 생성합니다. 생성 시 자체 ID를 UUID 형식으로 제공할 수도 있습니다.

Note

UUIDs는 생성된 AWS 리전 내에서 동일한 객체 유형에 대해 전역적으로 고유해야 합니다. 가 ID를 AWS IoT SiteWise 자동으로 생성하면 항상 고유합니다. 자체 ID를 선택하는 경우 고유한 ID를 선택해야 합니다.

예를 들어 [CreateAssetModel](#)을 직접적으로 호출하여 새 자산 모델을 생성하는 경우 요청의 선택적 `assetModelId` 필드에 자체 UUID를 제공할 수 있습니다.

반대로 요청에서 `assetModelId`를 생략하면 AWS IoT SiteWise 는 새 자산 모델에 대한 UUID를 생성합니다.

외부 ID 재사용

UUID 이외의 형식으로 자체 ID를 정의하려면 외부 ID를 할당할 수 있습니다. 예를 들어, 시스템에서 사용하지 않는 ID를 재사용 AWS하거나 사람이 더 읽을 수 있도록 재사용할 수 있습니다. 외부 ID는 더 유

연한 형식입니다. 다른 경우에는 UUID를 사용했을 AWS IoT SiteWise API 작업에서 객체를 참조하기 위해 이를 사용할 수 있습니다.

UUID와 마찬가지로 각 외부 ID는 컨텍스트 내에서 고유해야 합니다. 예를 들어, 외부 ID가 동일한 자산 모델은 두 개 있을 수 없습니다. 또한 UUID와 마찬가지로 객체는 수명 동안 외부 ID를 하나만 가질 수 있으며, 이 ID는 변경할 수 없습니다.

외부 ID와 UUID의 차이점

외부 ID는 다음과 같은 점에서 UUID와 다릅니다.

- 모든 객체에는 UUID가 있지만 외부 ID는 선택 사항입니다.
- AWS IoT SiteWise 는 외부 IDs 생성하지 않습니다. 외부 ID는 직접 제공할 수 있습니다.
- 객체에 외부 ID가 아직 없는 경우 언제든지 외부 ID를 할당할 수 있습니다.

외부 ID 형식

유효한 외부 ID에는 다음과 같은 속성이 있습니다.

- 2~128자 이내입니다.
- 첫 번째와 마지막 문자는 영숫자(A~Z, a~z, 0~9)여야 합니다.
- 첫 번째 및 마지막 문자 이외의 문자는 영숫자이거나 다음 중 하나여야 합니다. _ - . :

예를 들어 외부 ID는 다음 정규식을 준수해야 합니다.

```
[a-zA-Z0-9][a-zA-Z0-9_\-\.]*[a-zA-Z0-9]+
```

외부 ID가 있는 참조 객체

UUID를 사용하여 객체를 참조할 수 있는 많은 위치에서는 외부 ID가 있는 경우 이를 대신 사용할 수 있습니다. 이렇게 하려면 문자열 `externalId:`에 외부 ID를 추가합니다.

예를 들어 UUID(자산 모델 ID)가 `a1b2c3d4-5678-90ab-cdef-11111EXAMPLE`이고 `myExternalId`라는 외부 ID도 있는 자산 모델이 있다고 가정해 보겠습니다. [DescribeAssetModel](#)을 호출하여 자세한 내용을 확인합니다. 다음 중 하나를 `assetModelId`의 값으로 사용할 수 있습니다.

- 자산 모델 ID(UUID) 자체 사용: `a1b2c3d4-5678-90ab-cdef-11111EXAMPLE`
- 외부 ID 사용: `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`: 접두사는 그 자체로는 외부 ID의 일부가 아닙니다. UUID 또는 외부 ID를 허용하는 API 작업에 외부 ID를 제공할 때만 접두사를 제공하면 됩니다. 예를 들어 기존 객체를 쿼리하거나 업데이트할 때 접두사를 제공합니다.

자산 모델을 생성할 때와 같이 객체에 대한 외부 ID를 정의할 때는 접두사를 포함하지 않습니다.

이러한 방식으로 UUIDs IDs를 사용하여의 많은 API 작업을 수행할 수 AWS IoT SiteWise 있지만 전부는 사용할 수 없습니다. 예를 들어 [GetAssetPropertyValue](#)는 반드시 UUID를 사용해야 합니다. 외부 ID 사용은 지원하지 않습니다.

특정 API 작업이 이 사용을 지원하는지 확인하려면 [API 참조](#)를 참조하세요.

에 대한 자산 및 구성 요소 모델 생성 AWS IoT SiteWise

AWS IoT SiteWise 자산 모델 및 구성 요소 모델은 산업 데이터의 표준화를 촉진합니다. 자산 및 구성 요소 모델은 산업 자산 및 해당 구성 요소의 구조와 속성을 나타냅니다. 자산 모델은 풍력 터빈 또는 제조 라인과 같은 전체 자산을 정의합니다. 구성 요소 모델은 블레이드, 발전기 또는 센서와 같이 자산을 구성하는 개별 구성 요소를 나타냅니다. 이러한 모델을 생성하면 산업 장비의 실제 관계 및 계층 구조를 반영하는 방식으로 자산 데이터를 구성하고 구조화하여 더욱 쉽게 모니터링, 분석 및 유지 관리할 수 있습니다.

자산 모델 또는 구성 요소 모델에는 속성을 그룹화하거나 하위 어셈블리의 구성 요소 모델을 참조하는 이름, 설명, 자산 속성 및 (선택 사항) 사용자 지정 복합 모델이 포함되어 있습니다.

에서 자산 모델 및 구성 요소 모델을 생성하여 산업 자산 및 해당 구성 요소의 구조와 속성을 나타낼 AWS IoT SiteWise 수 있습니다.

- 자산 모델을 사용하여 자산을 생성할 수 있습니다. 위에 나열된 기능 외에도 자산 모델에는 자산 간의 관계를 정의하는 계층 구조 정의가 포함될 수 있습니다.

- 구성 요소 모델은 자산 모델 또는 다른 구성 요소 모델 내의 하위 어셈블리를 나타냅니다. 구성 요소 모델을 생성할 때 자산 모델 및 다른 구성 요소 모델에서 참조를 추가할 수 있습니다. 하지만 구성 요소 모델에서 직접 자산을 생성할 수는 없습니다.

자산 모델 또는 구성 요소 모델을 생성한 후 속성을 그룹화하거나 기존 구성 요소 모델을 참조하기 위해 사용자 지정 복합 모델을 생성할 수 있습니다.

자산 모델 및 구성 요소 모델을 생성하는 방법에 대한 자세한 내용은 다음 섹션을 참조하세요.

주제

- [에서 자산 모델 생성 AWS IoT SiteWise](#)
- [구성 요소 모델 생성](#)
- [데이터 속성 정의](#)
- [사용자 지정 복합 모델\(구성 요소\) 생성](#)

에서 자산 모델 생성 AWS IoT SiteWise

AWS IoT SiteWise 자산 모델은 산업 데이터의 표준화를 촉진합니다. 자산 모델에는 이름, 설명, 자산 속성 및 자산 계층 구조 정의가 포함되어 있습니다. 예를 들어 온도, 분당 회전 수(RPM) 및 전력 속성을 사용하여 풍력 터빈 모델을 정의할 수 있습니다. 그런 다음, 순 전력 출력 속성 및 풍력 터빈 계층 구조 정의를 사용하여 풍력 발전소 모델을 정의할 수 있습니다.

Note

- 가장 낮은 수준의 노드부터 시작해서 조직을 모델링하는 것이 좋습니다. 예를 들어 풍력 발전소 모델을 생성하기 전에 풍력 터빈 모델을 생성합니다. 자산 계층 정의에는 기존 자산 모델에 대한 참조가 포함되어 있습니다. 이 방법을 사용하면 모델을 생성할 때 자산 계층 구조를 정의할 수 있습니다.
- 자산 모델은 다른 자산 모델을 포함할 수 없습니다. 다른 모델 내에서 하위 어셈블리로 참조할 수 있는 모델을 정의해야 하는 경우 대신 구성 요소--> 모델을 생성해야 합니다. 자세한 내용은 [구성 요소 모델 생성](#) 단원을 참조하십시오.

다음 섹션에서는 AWS IoT SiteWise 콘솔 또는 API를 사용하여 자산 모델을 생성하는 방법을 설명합니다. 다음 단원에서는 모델을 생성하는 데 사용할 수 있는 다양한 유형의 자산 속성과 자산 계층 구조에 대해서도 설명합니다.

주제

- [자산 모델 생성\(콘솔\)](#)
- [자산 모델 생성\(AWS CLI\)](#)
- [자산 모델 예](#)
- [자산 모델 계층 정의](#)

자산 모델 생성(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자산 모델을 생성할 수 있습니다. AWS IoT SiteWise 콘솔은 유효한 자산 모델을 정의하는 데 도움이 되는 공식 자동 완성과 같은 다양한 기능을 제공합니다.

자산 모델을 생성하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. Create model(모델 생성)을 선택합니다.
4. 모델 생성 페이지에서 다음을 수행합니다.
 - a. 자산 모델의 이름을 입력합니다(예: **Wind Turbine** 또는 **Wind Turbine Model**). 이 이름은 이 리전의 계정에 있는 모든 모델에서 고유해야 합니다.
 - b. (선택 사항) 모델의 외부 ID를 추가합니다. 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.
 - c. (선택 사항) 모델의 측정 정의를 추가합니다. 측정값은 장비의 데이터 스트림을 나타냅니다. 자세한 내용은 [장비의 데이터 스트림 정의\(측정값\)](#) 단원을 참조하십시오.
 - d. (선택 사항) 모델의 변환 정의를 추가합니다. 변환은 데이터를 한 양식에서 다른 양식으로 매핑하는 공식입니다. 자세한 내용은 [데이터 변환\(변환\)](#) 단원을 참조하십시오.
 - e. (선택 사항) 모델의 지표 정의를 추가합니다. 지표는 일정 간격으로 데이터를 집계하는 공식입니다. 지표는 연결된 자산에서 데이터를 입력할 수 있으므로 작업 또는 작업의 하위 집합을 나타내는 값을 계산할 수 있습니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하십시오.
 - f. (선택 사항) 모델의 계층 구조 정의를 추가합니다. 계층은 자산 간의 관계입니다. 자세한 내용은 [자산 모델 계층 정의](#) 단원을 참조하십시오.
 - g. (선택 사항) 자산 모델에 대한 태그를 추가합니다. 자세한 내용은 [AWS IoT SiteWise 리소스에 태그 지정](#) 단원을 참조하십시오.
 - h. Create model(모델 생성)을 선택합니다.

자산 모델을 생성하면 AWS IoT SiteWise 콘솔이 새 모델의 페이지로 이동합니다. 이 페이지에서 모델의 상태를 볼 수 있습니다. 상태는 처음에 CREATING(생성 중)입니다. 이 페이지는 자동으로 업데이트되므로 모델 상태가 업데이트될 때까지 기다릴 수 있습니다.

Note

복잡한 모델의 경우, 자산 모델 생성 프로세스가 몇 분 정도 소요될 수 있습니다. 자산 모델 상태가 활성이면 자산 모델을 사용하여 자산을 생성할 수 있습니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

5. (선택 사항) 자산 모델을 만든 후 엣지에 맞게 자산 모델을 구성할 수 있습니다. SiteWise Edge에 대한 자세한 내용은 [Edge에서 AWS IoT SiteWise 엣지 기능 구성](#) 섹션을 참조하세요.
 - a. 모델 페이지에서 엣지용 구성을 선택합니다.
 - b. 모델 구성 페이지에서 모델의 엣지 구성을 선택합니다. 이 제어는가이 자산 모델과 연결된 속성을 계산하고 저장할 AWS IoT SiteWise 수 있는 위치를 제어합니다. 엣지용 모델 구성에 대한 자세한 내용은 [SiteWise Edge에서 OPC UA 소스 설정](#) 단원을 참조하세요.
 - c. 사용자 지정 엣지 구성에서 각 자산 모델 속성을 AWS IoT SiteWise 계산하고 저장할 위치를 선택합니다.

Note

연결된 변환 및 지표는 동일한 위치에 구성되어야 합니다. 엣지용 모델 구성에 대한 자세한 내용은 [SiteWise Edge에서 OPC UA 소스 설정](#) 단원을 참조하세요.

- d. 저장을 선택합니다. 이제 모델 페이지에서 엣지 구성이 구성됨이어야 합니다.

자산 모델 생성(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산 모델을 생성할 수 있습니다.

[CreateAssetModel](#) 작업을 사용하여 속성과 계층 구조가 있는 자산 모델을 생성합니다. 이 작업에는 다음과 같은 구조의 페이로드가 필요합니다.

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "String",
```

```

"assetModelDescription": "String",
"assetModelProperties": Array of AssetModelProperty,
"assetModelHierarchies": Array of AssetModelHierarchyDefinition
}

```

자산 모델을 생성하려면(AWS CLI) 다음을 수행하세요.

1. `asset-model-payload.json`이라는 파일을 만들고 다음 JSON 객체를 파일에 복사합니다.

```

{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ],
  "assetModelHierarchies": [

  ],
  "assetModelCompositeModels": [

  ]
}

```

2. 선호하는 JSON 텍스트 편집기를 사용하여 다음에 대한 `asset-model-payload.json` 파일을 편집합니다.
 - a. 자산 모델의 이름(`assetModelName`)을 입력합니다(예: **Wind Turbine** 또는 **Wind Turbine Model**). 이 이름은 이 AWS 리전의 계정에 있는 모든 자산 모델 및 구성 요소 모델 중에 고유해야 합니다.
 - b. (선택 사항) 자산 모델의 외부 ID(`assetModelExternalId`)를 입력합니다. 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.
 - c. (선택 사항) 자산 모델에 대한 설명(`assetModelDescription`)을 입력하거나 `assetModelDescription` 카값 페어를 제거합니다.
 - d. (선택 사항) 모델의 자산 속성(`assetModelProperties`)을 정의합니다. 자세한 내용은 [데이터 속성 정의](#) 단원을 참조하십시오.
 - e. (선택 사항) 모델의 자산 계층 구조(`assetModelHierarchies`)를 정의합니다. 자세한 내용은 [자산 모델 계층 정의](#) 단원을 참조하십시오.

- f. (선택 사항) 모델의 경보를 정의합니다. 경보는 다른 속성을 모니터링하므로 장비나 프로세스에 주의가 필요한 시점을 식별할 수 있습니다. 각 경고 정의는 경고가 사용하는 속성 집합을 표준화하는 복합 모델(assetModelCompositeModels)입니다. 자세한 내용은 [에서 경보를 사용하여 데이터 모니터링 AWS IoT SiteWise](#) 및 [에서 자산 모델에 대한 경고 정의 AWS IoT SiteWise](#) 섹션을 참조하세요.
 - g. (선택 사항) 자산 모델에 대한 태그(tags)를 추가합니다. 자세한 내용은 [AWS IoT SiteWise 리소스에 태그 지정](#) 단원을 참조하십시오.
3. 다음 명령을 실행하여 JSON 파일의 정의에서 자산 모델을 생성합니다.

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-payload.json
```

이 작업은 자산을 생성할 때 참조하는 assetModelId가 포함된 응답을 반환합니다. 응답에는 모델의 상태(assetModelStatus.state)도 포함됩니다. 상태는 처음에 CREATING입니다. 자산 모델의 상태는 변경 사항이 전파될 때까지 CREATING입니다.

Note

복잡한 모델의 경우, 자산 모델 생성 프로세스가 몇 분 정도 소요될 수 있습니다. 자산 모델의 현재 상태를 확인하려면 assetModelId를 지정하여 [DescribeAssetModel](#) 작업을 사용하세요. 자산 모델 상태가 ACTIVE이면 자산 모델을 사용하여 자산을 생성할 수 있습니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

4. (선택 사항) 자산 모델에 대한 사용자 지정 복합 모델을 생성합니다. 사용자 지정 복합 모델을 사용하면 모델 내에서 속성을 그룹화하거나 구성 요소 모델을 참조하여 하위 어셈블리를 포함할 수 있습니다. 자세한 내용은 [사용자 지정 복합 모델\(구성 요소\) 생성](#) 단원을 참조하십시오.

자산 모델 예

이 단원에는 AWS CLI 및 AWS IoT SiteWise SDKs. 이러한 자산 모델은 풍력 터빈과 풍력 발전 단지를 나타냅니다. 풍력 터빈 자산은 원시 센서 데이터를 수집하고 전력 및 평균 풍속과 같은 값을 계산합니다. 풍력 발전 단지 자산은 풍력 발전 단지 내 모든 풍력 터빈의 총 출력과 같은 값을 계산합니다.

주제

- [풍력 터빈 자산 모델](#)
- [풍력 발전소 자산 모델](#)

풍력 터빈 자산 모델

다음 자산 모델은 풍력 발전소의 터빈을 나타냅니다. 풍력 터빈은 센서 데이터를 수집하여 전력 및 평균 풍속과 같은 값을 계산합니다.

Note

이 예제 모델은 AWS IoT SiteWise 데모의 풍력 터빈 모델과 유사합니다. 자세한 내용은 [AWS IoT SiteWise 데모 사용](#) 단원을 참조하십시오.

```
{
  "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": []
}
}

```

풍력 발전소 자산 모델

다음 자산 모델은 여러 풍력 터빈으로 구성된 풍력 발전소를 나타냅니다. 이 자산 모델은 풍력 터빈 모델의 [계층 구조](#)를 정의합니다. 이를 통해 풍력 발전소는 풍력 발전소의 모든 풍력 터빈에 대한 데이터에서 값(예: 평균 출력)을 계산할 수 있습니다.

Note

이 예제 모델은 AWS IoT SiteWise 데모의 풍력 발전소 모델과 유사합니다. 자세한 내용은 [AWS IoT SiteWise 데모 사용](#) 단원을 참조하십시오.

이 자산 모델은 [풍력 터빈 자산 모델](#)에 따라 다릅니다. `propertyId` 및 `childAssetModelId` 값을 기존 풍력 터빈 자산 모델의 값으로 바꿉니다.

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

```

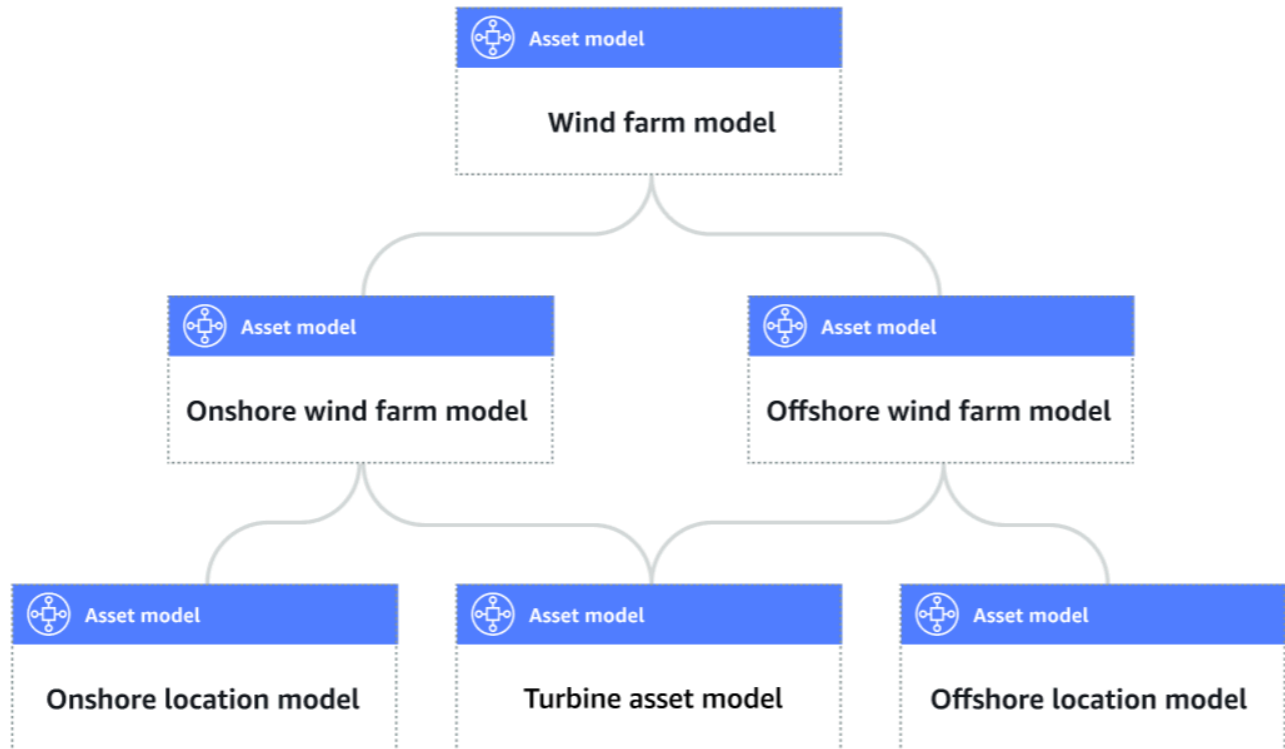
```
    }  
  }  
}  
},  
"assetModelHierarchies": [  
  {  
    "name": "Turbine Asset Model",  
    "childAssetModelId": "ID of Wind Turbine Asset Model"  
  }  
]  
}
```

자산 모델 계층 정의

자산 모델 계층 구조를 정의하여 산업 운영의 자산 모델 간에 논리적 연관성을 만들 수 있습니다. 예를 들어, 육상 및 해상 풍력 단지로 구성된 풍력 발전 단지를 정의할 수 있습니다. 육상 풍력 발전 단지에는 터빈과 육상 위치가 포함됩니다. 해상 풍력 발전 단지에는 터빈과 해상 위치가 있습니다.



Asset model hierarchy



계층 구조를 통해 하위 자산 모델을 상위 자산 모델에 연결하면 상위 자산 모델의 지표에서 하위 자산 모델 지표의 데이터를 입력할 수 있습니다. 자산 모델 계층 구조 및 지표를 사용하여 작업 또는 작업의 하위 집합에 대한 통찰을 제공하는 통계를 계산할 수 있습니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하십시오.

각 계층 구조는 상위 자산 모델과 하위 자산 모델 간의 관계를 정의합니다. 상위 자산 모델에서는 동일한 하위 자산 모델에 여러 계층 구조를 정의할 수 있습니다. 예를 들어 풍력 발전 단지에 두 가지 유형의 풍력 터빈이 있고 모든 풍력 터빈이 동일한 자산 모델로 표시되는 경우 각 유형에 대한 계층 구조를 정의할 수 있습니다. 그런 다음 풍력 발전 단지 모델에서 지표를 정의하여 각 풍력 터빈 유형에 대한 독립된 통계와 결합된 통계를 계산할 수 있습니다.

상위 자산 모델을 여러 하위 자산 모델과 연결할 수 있습니다. 예를 들어 두 개의 서로 다른 자산 모델로 표현되는 육상 풍력 발전 단지와 해상 풍력 발전 단지가 있는 경우 이러한 자산 모델을 동일한 상위 풍력 발전 단지 자산 모델과 연결할 수 있습니다.

하위 자산 모델을 여러 상위 자산 모델과 연결할 수도 있습니다. 예를 들어, 모든 풍력 터빈이 동일한 자산 모델로 표현되는 두 가지 유형의 풍력 발전 단지가 있는 경우 풍력 터빈 자산 모델을 서로 다른 풍력 발전 단지 자산 모델과 연결할 수 있습니다.

Note

자산 모델 계층 구조를 정의할 때 하위 자산 모델은 ACTIVE이거나 이전 ACTIVE 버전이어야 합니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

계층적 자산 모델을 정의하고 자산을 생성한 후에는 자산을 연결하여 상위-하위 관계를 작성할 수 있습니다. 자세한 내용은 [에서 자산 모델에 대한 자산 생성 AWS IoT SiteWise](#) 및 [자산 연결 및 연결 해제](#) 섹션을 참조하세요.

주제

- [자산 모델 계층 구조 정의\(콘솔\)](#)
- [자산 계층 정의\(AWS CLI\)](#)

자산 모델 계층 구조 정의(콘솔)

AWS IoT SiteWise 콘솔에서 자산 모델의 계층 구조를 정의할 때 다음 파라미터를 지정합니다.

- 계층 구조 이름 - 계층 구조의 이름입니다(예: **Wind Turbines**).
- 계층 구조 모델 - 하위 자산 모델입니다.
- 계층 구조 외부 ID(선택 사항) - 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

자세한 내용은 [자산 모델 생성\(콘솔\)](#) 단원을 참조하십시오.

자산 계층 정의(AWS CLI)

AWS IoT SiteWise API를 사용하여 자산 모델의 계층 구조를 정의할 때 다음 파라미터를 지정합니다.

- name - 계층 구조의 이름입니다(예: **Wind Turbines**).
- childAssetModelId - 계층 구조에 대한 하위 자산 모델의 ID 또는 외부 ID입니다. [ListAssetModels](#) 작업을 사용하여 기존 자산 모델의 ID를 찾을 수 있습니다.

Example 계층 구조 정의 예

다음 예제에서는 풍력 터빈에 대한 풍력 발전소의 관계를 나타내는 자산 모델 계층 구조를 보여 줍니다. 이 객체는 [AssetModelHierarchy](#)의 예입니다. 자세한 내용은 [자산 모델 생성\(AWS CLI\)](#) 단원을 참조하십시오.

```
{
  ...
  "assetModelHierarchies": [
    {
      "name": "Wind Turbines",
      "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
    },
  ]
}
```

구성 요소 모델 생성

AWS IoT SiteWise 구성 요소 모델을 사용하여 자산 모델 또는 기타 구성 요소 모델에서 참조할 수 있는 하위 어셈블리를 정의합니다. 이렇게 하면 여러 다른 모델에서 구성 요소의 정의를 재사용하거나 동일한 모델 내에서 여러 번 재사용할 수 있습니다.

구성 요소 모델을 정의하는 프로세스는 자산 모델을 정의하는 프로세스와 매우 유사합니다. 자산 모델과 마찬가지로 구성 요소 모델에는 이름, 설명 및 자산 속성이 있습니다. 그러나 구성 요소 모델 자체는 자산을 직접 생성하는 데 사용할 수 없으므로 구성 요소 모델에는 자산 계층 구조 정의가 포함될 수 없습니다. 구성 요소 모델도 경보를 정의할 수 없습니다.

예를 들어 모터 온도, 인코더 온도 및 절연 저항 속성이 있는 서보 모터의 구성 요소를 정의할 수 있습니다. 그런 다음 CNC 기계와 같은 서보 모터가 포함된 장비의 자산 모델을 정의할 수 있습니다.

Note

- 가장 낮은 수준의 노드부터 시작해서 조직을 모델링하는 것이 좋습니다. 예를 들어 CNC 기계의 자산 모델을 생성하기 전에 서보 모터 구성 요소를 생성합니다. 자산 모델에는 기존 구성 요소 모델에 대한 참조가 포함되어 있습니다.
- 구성 요소 모델에서 직접 자산을 생성할 수는 없습니다. 구성 요소를 사용하는 자산을 생성하려면 자산에 대한 자산 모델을 생성해야 합니다. 그런 다음 구성 요소를 참조하는 사용자 지정 복합 모델을 생성합니다. 자산 모델 생성에 대한 자세한 내용은 [에서 자산 모델 생성](#)

[AWS IoT SiteWise](#) 섹션을 참조하세요. 사용자 지정 복합 모델 생성에 대한 자세한 내용은 [사용자 지정 복합 모델\(구성 요소\) 생성](#) 섹션을 참조하세요.

다음 섹션에서는 AWS IoT SiteWise API를 사용하여 구성 요소 모델을 생성하는 방법을 설명합니다.

주제

- [구성 요소 모델 생성\(AWS CLI\)](#)
- [구성 요소 모델 예시](#)

구성 요소 모델 생성(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 구성 요소 모델을 생성할 수 있습니다.

[CreateAssetModel](#) 작업을 사용하여 속성이 포함된 구성 요소 모델을 생성합니다. 이 작업에는 다음과 같은 구조의 페이로드가 필요합니다.

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
}
```

구성 요소 모델을 생성하려면(AWS CLI) 다음을 수행하세요.

1. component-model-payload.json이라는 파일을 만들고 다음 JSON 객체를 파일에 복사합니다.

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ]
}
```

2. 선호하는 JSON 텍스트 편집기를 사용하여 다음에 대한 component-model-payload.json 파일을 편집합니다.

- a. 구성 요소 모델의 이름(assetModelName)을 입력합니다(예: **Servo Motor** 또는 **Servo Motor Model**). 이 이름은 이 AWS 리전의 계정에 있는 모든 자산 모델 및 구성 요소 모델 중에 고유해야 합니다.
 - b. (선택 사항) 구성 요소 모델의 외부 ID(assetModelExternalId)를 입력합니다. 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.
 - c. (선택 사항) 자산 모델에 대한 설명(assetModelDescription)을 입력하거나 assetModelDescription 카값 페어를 제거합니다.
 - d. (선택 사항) 구성 요소 모델의 자산 속성(assetModelProperties)을 정의합니다. 자세한 내용은 [데이터 속성 정의](#) 단원을 참조하십시오.
 - e. (선택 사항) 자산 모델에 대한 태그(tags)를 추가합니다. 자세한 내용은 [AWS IoT SiteWise 리소스에 태그 지정](#) 단원을 참조하십시오.
3. 다음 명령을 실행하여 JSON 파일의 정의에서 구성 요소 모델을 생성합니다.

```
aws iotsitewise create-asset-model --cli-input-json file:///component-model-payload.json
```

작업은 자산 모델 또는 다른 구성 요소 모델의 구성 요소 모델에 참조를 추가할 때 참조하는 assetModelId가 포함된 응답을 반환합니다. 응답에는 모델의 상태(assetModelStatus.state)도 포함됩니다. 상태는 처음에 CREATING입니다. 구성 요소 모델의 상태는 변경 사항이 전파될 때까지 CREATING입니다.

Note

복잡한 모델의 경우, 구성 요소 모델 생성 프로세스가 몇 분 정도 소요될 수 있습니다. 구성 요소 모델의 현재 상태를 확인하려면 assetModelId를 지정하여 [DescribeAssetModel](#) 작업을 사용하세요. 구성 요소 모델 상태가 ACTIVE이면 자산 모델 또는 기타 구성 요소 모델의 구성 요소 모델에 참조를 추가할 수 있습니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

4. (선택 사항) 구성 요소 모델에 대한 사용자 지정 복합 모델을 생성합니다. 사용자 지정 복합 모델을 사용하면 모델 내에서 속성을 그룹화하거나 다른 구성 요소 모델을 참조하여 하위 어셈블리를 포함할 수 있습니다. 자세한 내용은 [사용자 지정 복합 모델\(구성 요소\) 생성](#) 단원을 참조하십시오.

구성 요소 모델 예시

이 섹션에는 AWS CLI 및 AWS IoT SiteWise SDKs. 이 구성 요소 모델은 CNC 기계와 같은 다른 장비 내에서 사용할 수 있는 서보 모터를 나타냅니다.

주제

- [서보 모터 구성 요소 모델](#)

서보 모터 구성 요소 모델

다음 구성 요소 모델은 CNC 기계와 같은 장비 내에서 사용할 수 있는 서보 모터를 나타냅니다. 서보 모터는 온도 및 전기 저항과 같은 다양한 측정값을 제공합니다. 이러한 측정값은 서보 모터 구성 요소 모델을 참조하는 자산 모델에서 생성된 자산에서 속성으로 사용할 수 있습니다.

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

데이터 속성 정의

자산 속성은 자산 데이터를 포함하는 각 자산 내부의 구조입니다. 자산 속성은 다음 유형 중 하나일 수 있습니다.

- 속성 - 디바이스 제조업체 또는 지리적 위치 같은, 일반적으로 정적인 자산 속성입니다. 자세한 내용은 [정적 데이터\(속성\) 정의](#) 단원을 참조하십시오.
- 측정 - 타임스탬프가 찍힌 회전 속도 값 또는 타임스탬프가 찍힌 섭씨 온도 값과 같은 자산의 원시 기기 센서 데이터 스트림입니다. 측정은 데이터 스트림 별칭에 의해 정의됩니다. 자세한 내용은 [장비의 데이터 스트림 정의\(측정값\)](#) 단원을 참조하십시오.
- 변형 - 타임스탬프가 찍힌 화씨 온도 값 같은, 자산의 변형된 시계열 값입니다. 변형은 표현식 및 해당 표현식과 함께 사용할 변수로 정의됩니다. 자세한 내용은 [데이터 변환\(변환\)](#) 단원을 참조하십시오.
- 지표 - 시간 단위의 평균 온도 같은, 지정된 시간 간격 동안 집계된 자산의 데이터입니다. 지표는 시간 간격, 표현식 및 해당 표현식과 함께 사용할 변수로 정의됩니다. 지표 표현식은 관련 자산의 지표 속성을 입력할 수 있으므로 작업 또는 작업의 하위 집합을 나타내는 지표를 계산할 수 있습니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하십시오.

자세한 내용은 [에서 자산 모델 생성 AWS IoT SiteWise](#) 단원을 참조하십시오.

측정, 변형 및 지표를 사용하여 OEE(전반적인 장비 효율성)를 계산하는 방법의 예는 [에서 OEE 계산 AWS IoT SiteWise](#) 단원을 참고하세요.

주제

- [정적 데이터\(속성\) 정의](#)
- [장비의 데이터 스트림 정의\(측정값\)](#)
- [데이터 변환\(변환\)](#)
- [속성과 기타 자산에서 데이터 집계\(지표\)](#)
- [수식 표현식 사용](#)

정적 데이터(속성) 정의

자산 속성은 디바이스 제조업체 또는 지리적 위치 같은, 일반적으로 정적인 정보를 나타냅니다. 자산 템플릿에서 생성한 각 자산은 해당 모델의 속성을 포함합니다.

주제

- [속성 정의\(콘솔\)](#)
- [속성 정의\(AWS CLI\)](#)

속성 정의(콘솔)

AWS IoT SiteWise 콘솔에서 자산 모델의 속성을 정의할 때 다음 파라미터를 지정합니다.

- 이름 - 속성의 이름입니다.
- 기본값 - (선택 사항) 이 속성의 기본값입니다. 모델에서 생성된 자산은 속성에 대해 이 값을 가집니다. 모델에서 생성된 자산의 기본값을 재정의하는 방법에 대한 자세한 내용은 [속성 값 업데이트](#) 단원을 참조하세요.
- 데이터 형식 - 속성의 데이터 형식으로 다음 중 하나입니다.
 - 문자열 - 최대 1,024바이트의 문자열입니다.
 - 정수 - 범위가 [-2,147,483,648, 2,147,483,647]인 부호 있는 32비트 정수입니다.
 - 실수 - 범위가 [-10¹⁰⁰, 10¹⁰⁰]이고 IEEE 754 배정밀도를 갖는 부동 소수점 숫자입니다.
 - 부울 - true 또는 false입니다.
- 외부 ID - (선택 사항) 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

자세한 내용은 [자산 모델 생성\(콘솔\)](#) 단원을 참조하십시오.

속성 정의(AWS CLI)

AWS IoT SiteWise API를 사용하여 자산 모델의 속성을 정의할 때 다음 파라미터를 지정합니다.

- name - 속성의 이름입니다.
- defaultValue - (선택 사항) 이 속성의 기본값입니다. 모델에서 생성된 자산은 속성에 대해 이 값을 가집니다. 모델에서 생성된 자산의 기본값을 재정의하는 방법에 대한 자세한 내용은 [속성 값 업데이트](#) 단원을 참조하세요.
- dataType - 속성의 데이터 형식으로 다음 중 하나입니다.
 - STRING - 최대 1,024바이트의 문자열입니다.
 - INTEGER - 범위가 [-2,147,483,648, 2,147,483,647]인 부호 있는 32비트 정수입니다.
 - DOUBLE - 범위가 [-10¹⁰⁰, 10¹⁰⁰]이고 IEEE 754 배정밀도를 갖는 부동 소수점 숫자입니다.
 - BOOLEAN - true 또는 false입니다.
- externalId - (선택 사항) 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

Example 속성 정의 예

다음 예제에서는 자산의 모델 번호를 나타내는 속성을 기본값과 함께 보여 줍니다. 이 객체는 [속성](#)이 포함된 [AssetModelProperty](#)의 예입니다. 이 객체를 [CreateAssetModel](#) 요청 페이로드의 일부로 지정하여 속성 특성을 생성할 수 있습니다. 자세한 내용은 [자산 모델 생성\(AWS CLI\)](#) 단원을 참조하십시오.

```
{
  ...
  "assetModelProperties": [
    {
      "name": "Model number",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "BLT123"
        }
      }
    }
  ],
  ...
}
```

장비의 데이터 스트림 정의(측정값)

측정은 타임스탬프가 지정된 온도 값 또는 타임스탬프가 지정된 분당 회전 수(RPM) 값 같은, 디바이스의 원시 센서 데이터 스트림을 나타냅니다.

주제

- [측정값 정의\(콘솔\)](#)
- [측정값 정의\(AWS CLI\)](#)

측정값 정의(콘솔)

AWS IoT SiteWise 콘솔에서 자산 모델에 대한 측정을 정의할 때 다음 파라미터를 지정합니다.

- 이름 – 속성의 이름입니다.
- 단위 – (선택 사항) 속성에 대한 과학적 단위(예: mm 또는 섭씨)입니다.
- 데이터 형식 - 속성의 데이터 형식으로 하나입니다.
 - 문자열 - 최대 1,024바이트의 문자열입니다.

- 정수 - 범위가 [-2,147,483,648, 2,147,483,647]인 부호 있는 32비트 정수입니다.
- 실수 - 범위가 [-10¹⁰⁰, 10¹⁰⁰]이고 IEEE 754 배정밀도를 갖는 부동 소수점 숫자입니다.
- 부울 - true 또는 false입니다.
- 외부 ID - (선택 사항) 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

자세한 내용은 [자산 모델 생성\(콘솔\)](#) 단원을 참조하십시오.

측정값 정의(AWS CLI)

AWS IoT SiteWise API를 사용하여 자산 모델에 대한 측정을 정의할 때 다음 파라미터를 지정합니다.

- name - 속성의 이름입니다.
- dataType - 속성의 데이터 형식으로 하나입니다.
 - STRING - 최대 1,024바이트의 문자열입니다.
 - INTEGER - 범위가 [-2,147,483,648, 2,147,483,647]인 부호 있는 32비트 정수입니다.
 - DOUBLE - 범위가 [-10¹⁰⁰, 10¹⁰⁰]이고 IEEE 754 배정밀도를 갖는 부동 소수점 숫자입니다.
 - BOOLEAN - true 또는 false입니다.
- unit - (선택 사항) 속성에 대한 과학적 단위(예: mm 또는 섭씨)입니다.
- externalId - (선택 사항) 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

Example 측정 정의 예

다음 예제에서는 자산의 온도 센서 판독값을 나타내는 측정을 보여줍니다. 이 객체는 [측정](#)이 포함된 [AssetModelProperty](#)의 예입니다. 이 객체를 [CreateAssetModel](#) 요청 페이로드의 일부로 지정하여 측정 속성을 생성할 수 있습니다. 자세한 내용은 [자산 모델 생성\(AWS CLI\)](#) 단원을 참조하십시오.

나중에 고유한 디바이스 데이터 스트림을 사용하도록 각 자산을 구성해야 하므로 자산 모델을 정의할 때 [측정](#) 구조는 빈 구조입니다. 자산의 측정 속성을 디바이스의 센서 데이터 스트림에 연결하는 방법에 대한 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 단원을 참조하세요.

```
{
    ...
    "assetModelProperty": [
        {
```

```

        "name": "Temperature C",
        "dataType": "DOUBLE",
        "type": {
            "measurement": {}
        },
        "unit": "Celsius"
    },
    ...
}

```

데이터 변환(변환)

변환은 자산 속성의 데이터 포인트를 한 양식에서 다른 양식으로 매핑하는 수학적 표현식입니다. 변환 표현식은 자산 속성 변수, 리터럴, 연산자 및 함수로 구성됩니다. 변환된 데이터 포인트는 입력 데이터 포인트와 one-to-one 관계를 유지합니다.는 입력 속성이 새 데이터 포인트를 수신할 때마다 변환된 새 데이터 포인트를 AWS IoT SiteWise 계산합니다.

Note

타임스탬프가 동일한 속성 업데이트의 경우 다른 수신 속성의 업데이트로 출력 값을 덮어쓸 수 있습니다.

예를 들어, 자산에 섭씨 단위의 Temperature_C라는 온도 측정 스트림이 있는 경우 $Temperature\_F = 9/5 * Temperature\_C + 32$ 공식을 사용하여 각 데이터 포인트를 화씨로 변환할 수 있습니다. 가 Temperature_C 측정 스트림에서 데이터 포인트를 AWS IoT SiteWise 수신할 때마다 해당 Temperature_F 값은 몇 초 이내에 계산되고 Temperature_F 속성으로 사용할 수 있습니다.

변환에 둘 이상의 변수가 포함된 경우 이전에 도착한 데이터 포인트가 계산을 즉시 시작합니다. 부품 제조업체가 변환을 사용하여 제품 품질을 모니터링하는 예를 들어 보겠습니다. 제조업체는 부품 유형에 따라 다른 표준을 사용하여 다음 측정값을 사용하여 공정을 나타냅니다.

- Part_Number - 부품 유형을 식별하는 문자열입니다.
- Good_Count - 부품이 표준을 충족할 경우 1씩 증가하는 정수입니다.
- Bad_Count - 부품이 표준을 충족하지 않을 경우 1씩 증가하는 정수입니다.

제조업체에서는 `if(eq(Part_Number, "BLT123") and (Bad_Count / (Good_Count + Bad_Count) > 0.1), "Caution", "Normal")`와 같은 Quality_Monitor 변환도 생성합니다.

이 변환은 특정 부품 유형에서 생산된 불량 부품의 비율을 모니터링합니다. 부품 번호가 BLT123 이고 불량 부품의 비율이 10%(0.1)를 초과하면 변환이 "Caution"으로 반환됩니다. 그렇지 않으면 변환이 "Normal"로 반환됩니다.

Note

- Part_Number에서 다른 측정값보다 먼저 새 데이터 포인트를 받으면 Quality_Monitor 변환은 새 Part_Number 값과 최신 Good_Count 및 Bad_Count 값을 사용합니다. 오류를 방지하려면 다음 제조 실행 전에 Good_Count 및 Bad_Count를 다시 설정하세요.
- 모든 변수에 새 데이터 포인트가 입력된 후에만 표현식을 평가하려면 [지표](#)를 사용하세요.

주제

- [변환 정의\(콘솔\)](#)
- [변환 정의\(AWS CLI\)](#)

변환 정의(콘솔)

AWS IoT SiteWise 콘솔에서 자산 모델에 대한 변환을 정의할 때 다음 파라미터를 지정합니다.

- 이름 - 속성의 이름입니다.
- 단위 - (선택 사항) 속성에 대한 과학적 단위(예: mm 또는 섭씨)입니다.
- 데이터 유형 - 변환의 데이터 유형으로, 실수 또는 문자열일 수 있습니다.
- 외부 ID - (선택 사항) 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.
- 공식 - 변환 표현식입니다. 변환 표현식에서는 집계 함수나 시간 함수를 사용할 수 없습니다. 입력을 시작하거나 아래쪽 화살표 키를 눌러 자동 완성 기능을 엽니다. 자세한 내용은 [수식 표현식 사용](#) 단원을 참조하십시오.

Important

변환은 정수, 실수, 부울 또는 문자열 유형인 속성을 입력할 수 있습니다. 부울은 0(거짓)과 1(참)로 변환됩니다.

변환은 속성이 아닌 하나 이상의 특성과 원하는 수의 속성인 특성을 입력해야 합니다. AWS IoT SiteWise 는 속성이 아닌 입력 특성이 새 데이터 포인트를 수신할 때마다 새로 변환된 데

이터 포인트를 계산합니다. 새 속성 값은 변환 업데이트를 실행하지 않습니다. 자산 속성 데이터 API 작업에 대한 동일한 요청 비율이 변환 계산 결과에 적용됩니다.

공식 표현식은 실수 또는 문자열 값만 출력할 수 있습니다. 중첩 표현식은 문자열과 같은 다른 데이터 유형을 출력할 수 있지만 공식 전체는 숫자나 문자열로 평가되어야 합니다. [jp 함수](#)를 사용하여 문자열을 숫자로 변환할 수 있습니다. 부울 값은 1(참) 또는 0(거짓)이어야 합니다. 자세한 내용은 [정의되지 않은 값, 무한 값 및 오버플로우 값](#) 단원을 참조하십시오.

자세한 내용은 [자산 모델 생성\(콘솔\)](#) 단원을 참조하십시오.

변환 정의(AWS CLI)

AWS IoT SiteWise API를 사용하여 자산 모델에 대한 변환을 정의할 때 다음 파라미터를 지정합니다.

- name - 속성의 이름입니다.
- unit - (선택 사항) 속성에 대한 과학적 단위(예: mm 또는 섭씨)입니다.
- dataType - 변환의 데이터 유형으로 DOUBLE 또는 STRING이어야 합니다.
- externalId - (선택 사항) 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.
- expression - 변환 표현식입니다. 변환 표현식에서는 집계 함수나 시간 함수를 사용할 수 없습니다. 자세한 내용은 [수식 표현식 사용](#) 단원을 참조하십시오.
- variables - 표현식에서 사용할 자산의 기타 특성을 정의하는 변수 목록입니다. 각 변수 구조에는 표현식에 사용할 간단한 이름과 해당 변수에 연결할 속성을 식별하는 value 구조가 포함되어 있습니다. value 구조에는 다음 정보가 포함되어 있습니다.
- propertyId - 값을 입력할 속성의 ID입니다. ID 대신 속성 이름을 사용할 수 있습니다.

Important

변환은 정수, 실수, 부울 또는 문자열 유형의 속성을 입력할 수 있습니다. 부울은 0(거짓)과 1(참)로 변환됩니다.

변환은 속성이 아닌 하나 이상의 특성과 원하는 수의 속성인 특성을 입력해야 합니다. AWS IoT SiteWise 는 속성이 아닌 입력 특성이 새 데이터 포인트를 수신할 때마다 새로 변환된 데이터 포인트를 계산합니다. 새 속성 값은 변환 업데이트를 실행하지 않습니다. 자산 속성 데이터 API 작업에 대한 동일한 요청 비율이 변환 계산 결과에 적용됩니다.

공식 표현식은 실수 또는 문자열 값만 출력할 수 있습니다. 중첩 표현식은 문자열과 같은 다른 데이터 유형을 출력할 수 있지만 공식 전체는 숫자나 문자열로 평가되어야 합니다. [jp 함수](#)

[수](#)를 사용하여 문자열을 숫자로 변환할 수 있습니다. 부울 값은 1(참) 또는 0(거짓)이어야 합니다. 자세한 내용은 [정의되지 않은 값, 무한 값 및 오버플로우 값](#) 단원을 참조하십시오.

Example 변환 정의

다음 예는 자산의 온도 측정 데이터를 섭씨에서 화씨로 변환하는 변형 속성을 보여줍니다. 이 객체는 [변환](#)이 포함된 [AssetModelProperty](#)의 예입니다. 이 객체를 [CreateAssetModel](#) 요청 페이로드의 일부로 지정하여 변환 속성을 생성할 수 있습니다. 자세한 내용은 [자산 모델 생성\(AWS CLI\)](#) 단원을 참조하십시오.

```
{
  ...
  "assetModelProperty": [
    ...
    {
      "name": "Temperature F",
      "dataType": "DOUBLE",
      "type": {
        "transform": {
          "expression": "9/5 * temp_c + 32",
          "variables": [
            {
              "name": "temp_c",
              "value": {
                "propertyId": "Temperature C"
              }
            }
          ]
        }
      },
      "unit": "Fahrenheit"
    }
  ],
  ...
}
```

Example 세 개의 변수를 포함하는 변환 정의

다음 예제는 BLT123 부품 중 10% 이상이 표준을 충족하지 않는 경우 경고 메시지("Caution")를 반환하는 변환 속성을 보여줍니다. 그렇지 않으면 정보 메시지("Normal")를 반환합니다.

```

{
  ...
  "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"
              }
            }
          ]
        }
      }
    }
  ]
}
...
}

```

속성과 기타 자산에서 데이터 집계(지표)

지표는 집계 함수를 사용하여 모든 입력 데이터 포인트를 처리하고 지정된 시간 간격당 단일 데이터 포인트를 출력하는 수학적 표현식입니다. 예를 들어, 어떤 지표는 온도 데이터 스트림의 시간별 평균 온도를 계산할 수 있습니다.

지표는 연결된 자산의 지표에서 데이터를 입력할 수 있으므로 작업 또는 작업의 하위 집합에 대한 통찰을 제공하는 통계를 계산할 수 있습니다. 예를 들어, 지표는 풍력 발전소의 모든 풍력 터빈에 대한 시간당 평균 온도를 계산할 수 있습니다. 자산 간의 연결을 정의하는 방법에 대한 자세한 내용은 [자산 모델 계층 정의](#) 단원을 참조하세요.

또한 지표는 각 시간 간격의 데이터를 집계하지 않고도 다른 속성의 데이터를 입력할 수 있습니다. 공식에서 [속성](#)을 지정하는 경우 공식을 계산할 때 AWS IoT SiteWise 는 해당 속성의 [최신](#) 값을 사용합니다. 공식에 지표를 지정하는 경우는 공식을 계산하는 시간 간격의 [마지막](#) 값을 AWS IoT SiteWise 사용합니다. 즉, $OEE = Availability * Quality * Performance$ 와 같은 지표를 정의할 수 있습니다. 여기서 Availability, Quality, Performance는 모두 동일한 자산 모델의 다른 지표입니다.

AWS IoT SiteWise 또한 모든 자산 속성에 대한 기본 집계 지표 집합을 자동으로 계산합니다. 계산 비용을 줄이기 위해, 기본 계산에 대한 사용자 지정 지표를 정의하는 대신 이러한 집계를 사용할 수 있습니다. 자세한 내용은 [에서 자산 속성 집계 쿼리 AWS IoT SiteWise](#) 단원을 참조하십시오.

주제

- [지표 정의\(콘솔\)](#)
- [지표 정의\(AWS CLI\)](#)

지표 정의(콘솔)

AWS IoT SiteWise 콘솔에서 자산 모델의 지표를 정의할 때 다음 파라미터를 지정합니다.

- 이름 - 속성의 이름입니다.
- 데이터 유형 - 변환의 데이터 유형으로, 실수 또는 문자열일 수 있습니다.
- 외부 ID - (선택 사항) 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.
- 공식 - 지표 표현식입니다. 지표 표현식은 [집계 함수](#)를 사용하여 계층 구조의 모든 관련 자산에 대한 속성 데이터를 입력할 수 있습니다. 입력을 시작하거나 아래쪽 화살표 키를 눌러 자동 완성 기능을 엽니다. 자세한 내용은 [수식 표현식 사용](#) 단원을 참조하십시오.

Important

지표는 정수, 실수, 부울 또는 문자열 유형의 속성만 입력할 수 있습니다. 부울은 0(거짓)과 1(참)로 변환됩니다.
지표의 표현식에서 지표 입력 변수를 정의하는 경우, 이러한 입력의 시간 간격은 출력 지표와 동일해야 합니다.

공식 표현식은 실수 또는 문자열 값만 출력할 수 있습니다. 중첩 표현식은 문자열과 같은 다른 데이터 유형을 출력할 수 있지만 공식 전체는 숫자나 문자열로 평가되어야 합니다. [jp 함수](#)를 사용하여 문자열을 숫자로 변환할 수 있습니다. 부울 값은 1(참) 또는 0(거짓)이어야 합니다. 자세한 내용은 [정의되지 않은 값, 무한 값 및 오버플로우 값](#) 단원을 참조하십시오.

- 시간 간격 - 시간 간격 지표입니다. AWS IoT SiteWise 는 다음 연속 창 시간 간격을 지원하며, 여기서 각 간격은 이전 간격이 끝날 때 시작됩니다.
 - 1분 - 1분입니다. 1분이 끝날 때마다 계산됩니다(오전 12:00:00, 오전 12:01:00, 오전 12:02:00 등).
 - 5분 - 5분입니다. 정각에 시작하여 5분이 끝날 때마다 계산됩니다(오전 12:00:00, 오전 12:05:00, 오전 12:10:00 등).
 - 15분 - 15분입니다. 정각에 시작하여 15분이 끝날 때마다 계산됩니다(오전 12:00:00, 오전 12:15:00, 오전 12:30:00 등).
 - 1시간 - 1시간(60분)입니다. UTC를 기준으로 한 시간이 끝날 때마다 계산됩니다(오전 12:00:00, 오전 01:00:00, 오전 02:00:00 등).
 - 1일 - 1일(24시간)입니다. UTC를 기준으로 하루가 끝날 때마다 계산됩니다(월요일 오전 12:00:00, 화요일 오전 12:00:00 등).
 - 1주 - 1주(7일)입니다. UTC를 기준으로 일요일이 끝날 때마다 계산됩니다(매주 월요일 오전 12:00:00).
 - 사용자 지정 간격 - 1분에서 1주 사이의 원하는 시간 간격을 입력할 수 있습니다.
- 오프셋 날짜 - (선택 사항) 데이터를 집계할 기준 날짜입니다.
- 오프셋 시간 - (선택 사항) 데이터를 집계할 기준 시간입니다. 오프셋 시간은 00:00:00 에서 23:59:59 사이여야 합니다.
- 오프셋 시간대 - (선택 사항) 오프셋의 시간대입니다. 지정되지 않은 경우 기본 오프셋 시간대는 협정 세계시(UTC)입니다.

지원되는 시간대

- (UTC+00:00) 협정 세계시
- (UTC+01:00) 유럽 중부 표준시
- (UTC+02:00) 동유럽
- (UTC+03:00) 동부 아프리카 표준시
- (UTC+04:00) 근동 표준시
- (UTC+05:00) 파키스탄 라호르 표준시
- (UTC+05:30) 인도 표준시

- (UTC+06:00) 방글라데시 표준시
- (UTC+07:00) 베트남 표준시
- (UTC+08:00) 중국 대만 표준시
- (UTC+09:00) 일본 표준시
- (UTC+09:30) 오스트레일리아 중부 표준시
- (UTC+10:00) 오스트레일리아 동부 표준시
- (UTC+11:00) 솔로몬 표준시
- (UTC+12:00) 뉴질랜드 표준시
- (UTC-11:00) 미드웨이 제도 표준시
- (UTC-10:00) 하와이 표준시
- (UTC-09:00) 알래스카 표준시
- (UTC-08:00) 태평양 표준시
- (UTC-07:00) 피닉스 표준시
- (UTC-06:00) 미국 중부 표준시
- (UTC-05:00) 미국 동부 표준시
- (UTC-04:00) 푸에르토리코 및 미국령 버진아일랜드 표준시
- (UTC-03:00) 아르헨티나 표준시
- (UTC-02:00) 사우스조지아 표준시
- (UTC-01:00) 중앙아프리카 표준시

Example 오프셋 사용 사용자 지정 시간 간격(콘솔)

다음 예제는 2021년 2월 20일 오후 6시 30분(PST)에 오프셋을 사용하여 12시간 간격을 정의하는 방법을 보여줍니다.

오프셋을 사용하여 사용자 지정 구간을 정의하려면 다음을 수행하세요.

1. 시간 간격의 경우 사용자 지정 간격을 선택합니다.
2. 시간 간격의 경우 다음 중 하나를 수행하세요.
 - **12**를 입력한 다음 시간을 선택합니다.
 - **720**을 입력한 다음 분을 선택합니다.
 - **43200**을 입력한 다음 초를 선택합니다.

⚠ Important

시간 간격은 단위에 관계없이 정수여야 합니다.

3. 오프셋 날짜로 2021/02/20을 선택합니다.
4. 오프셋 시간에는 **18:30:30**을 입력합니다.
5. 오프셋 시간대의 경우 (UTC-08:00) 태평양 표준시를 선택합니다.

2021년 7월 1일 오후 6시 30분 30초(PST) 이전에 지표를 생성하면 2021년 7월 1일 오후 6시 30분 30초(PST)에 첫 번째 집계 결과를 얻을 수 있습니다. 두 번째 집계 결과는 2021년 7월 2일 오전 6시 30분 30초(PST)이고, 이러한 간격으로 계속 진행됩니다.

지표 정의(AWS CLI)

AWS IoT SiteWise API를 사용하여 자산 모델에 대한 지표를 정의할 때 다음 파라미터를 지정합니다.

- name - 속성의 이름입니다.
- dataType - 지표의 데이터 유형으로 DOUBLE 또는 STRING일 수 있습니다.
- externalId - (선택 사항) 이는 사용자 정의 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.
- expression - 지표 표현식입니다. 지표 표현식은 [집계 함수](#)를 사용하여 계층 구조의 모든 관련 자산에 대한 속성 데이터를 입력할 수 있습니다. 자세한 내용은 [수식 표현식 사용](#) 단원을 참조하십시오.
- window - 이전 간격이 끝날 때 각 간격이 시작되는 지표의 연속 창에 대한 시간 간격 및 오프셋:
 - interval - 연속 창의 시간 간격입니다. 시간 간격은 1분과 1주 사이여야 합니다.
 - offsets - 연속 창의 오프셋입니다.

자세한 내용은 AWS IoT SiteWise API 참조의 [TumblingWindow](#)를 참조하세요.

Example 오프셋이 있는 사용자 지정 시간 간격(AWS CLI)

다음 예제는 2021년 2월 20일 오후 6시 30분(PST)에 오프셋을 사용하여 12시간 간격을 정의하는 방법을 보여줍니다.

```
{
  "window": {
    "tumbling": {
      "interval": "12h",
      "offset": " 2021-07-23T18:30:30-08"
    }
  }
}
```

2021년 7월 1일 오후 6시 30분 30초(PST) 이전에 지표를 생성하면 2021년 7월 1일 오후 6시 30분 30초(PST)에 첫 번째 집계 결과를 얻을 수 있습니다. 두 번째 집계 결과는 2021년 7월 2일 오전 6시 30분 30초(PST)이고, 이러한 간격으로 계속 진행됩니다.

- **variables** – 표현식에 사용할 자산 또는 하위 자산의 기타 속성을 정의하는 변수 목록입니다. 각 변수 구조에는 표현식에 사용할 간단한 이름과 해당 변수에 연결할 속성을 지정하는 `value` 구조가 포함되어 있습니다. `value` 구조에는 다음 정보가 포함되어 있습니다.
- **propertyId** – 값을 가져올 속성의 ID입니다. 속성이 계층의 모델에 정의되어 있지 않고 현재 모델에 정의된 경우 ID 대신 속성 이름을 사용할 수 있습니다.
- **hierarchyId** – (선택 사항) 속성에 대한 하위 자산을 쿼리할 계층의 ID입니다. ID 대신 계층 정의의 이름을 사용할 수 있습니다. 이 값을 생략하면 현재 모델에서 속성을 AWS IoT SiteWise 찾습니다.

Important

지표는 정수, 실수, 부울 또는 문자열 유형의 속성만 입력할 수 있습니다. 부울은 0(거짓)과 1(참)로 변환됩니다.

지표의 표현식에서 지표 입력 변수를 정의하는 경우, 이러한 입력의 시간 간격은 출력 지표와 동일해야 합니다.

공식 표현식은 실수 또는 문자열 값만 출력할 수 있습니다. 중첩 표현식은 문자열과 같은 다른 데이터 유형을 출력할 수 있지만 공식 전체는 숫자나 문자열로 평가되어야 합니다. [jp 함수](#)를 사용하여 문자열을 숫자로 변환할 수 있습니다. 부울 값은 1(참) 또는 0(거짓)이어야 합니다. 자세한 내용은 [정의되지 않은 값, 무한 값 및 오버플로우 값](#) 단원을 참조하십시오.

- **unit** – (선택 사항) 속성에 대한 과학적 단위(예: mm 또는 섭씨)입니다.

Example 지표 정의 예

다음 예제에서는 자산의 온도 측정 데이터를 집계하여 시간당 최대 화씨 온도를 계산하는 지표 속성을 보여줍니다. 이 객체는 [지표](#)가 포함된 [AssetModelProperty](#)의 예입니다. 이 객체를 [CreateAssetModel](#)

요청 페이로드의 일부로 지정하여 지표 속성을 생성할 수 있습니다. 자세한 내용은 [자산 모델 생성 \(AWS CLI\)](#) 단원을 참조하십시오.

```
{
  ...
  "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 관련 자산에서 데이터를 입력하는 지표 정의 예시

다음 예는 여러 풍력 터빈의 평균 전력 데이터를 집계하여 풍력 발전 단지의 총 평균 전력을 계산하는 지표 속성을 보여줍니다. 이 객체는 [지표](#)가 포함된 [AssetModelProperty](#)의 예입니다. 이 객체를 [CreateAssetModel](#) 요청 페이로드의 일부로 지정하여 지표 속성을 생성할 수 있습니다.

```
{
  ...
}
```

```

    "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"
      }
    ],
    ...
  }
}

```

수식 표현식 사용

수식 표현식을 사용하면 수학 함수를 정의하여 원시 산업 데이터를 변형하고 집계함으로써 작업에 대한 통찰력을 얻을 수 있습니다. 수식 표현식은 리터럴, 연산자, 함수 및 변수를 결합하여 데이터를 처리합니다. 공식 표현식을 사용하는 자산 속성을 정의하는 방법에 대한 자세한 내용은 [데이터 변환\(변환\)](#) 및 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하세요. 변환과 지표는 공식 속성입니다.

주제

- [수식 표현식에서 변수 사용](#)
- [수식 표현식에 리터럴 사용](#)
- [수식 표현식에 연산자 사용](#)

- [수식 표현식에 상수 사용](#)
- [수식 표현식에서 함수 사용](#)
- [공식 표현식 튜토리얼](#)

수식 표현식에서 변수 사용

변수는 수식 표현식의 AWS IoT SiteWise 자산 속성을 나타냅니다. 변수를 사용하여 표현식에 다른 자산 속성의 값을 입력하면 상수 속성([속성](#)), 원시 데이터 스트림([측정](#)) 및 기타 공식 속성의 데이터를 처리할 수 있습니다.

변수는 동일한 자산 모델 또는 관련 하위 자산 모델의 자산 속성을 나타낼 수 있습니다. 지표 공식만 하위 자산 모델의 변수를 입력할 수 있습니다.

콘솔과 API에서 변수를 서로 다른 이름으로 식별할 수 있습니다.

- AWS IoT SiteWise 콘솔 - 표현식에서 자산 속성 이름을 변수로 사용합니다.
- AWS IoT SiteWise API(AWS CLI, AWS SDKs) - 변수 이름과 자산 속성에 대한 참조가 필요한 [ExpressionVariable](#) 구조로 변수를 정의합니다. 변수 이름에는 소문자, 숫자, 밑줄이 포함될 수 있습니다. 그런 다음 변수 이름을 사용하여 표현식의 자산 속성을 참조합니다.

변수 이름은 대소문자를 구분합니다.

자세한 내용은 [변환 정의](#) 및 [지표 정의](#)를 참조하세요.

변수를 사용하여 속성 참조

변수의 값은 변수가 참조하는 속성을 정의합니다. AWS IoT SiteWise 는 이를 수행하는 다양한 방법을 제공합니다.

- 속성 ID별: 속성의 고유 ID(UUID)를 지정하여 식별할 수 있습니다.
- 이름별: 속성이 동일한 자산 모델에 있는 경우 속성 ID 필드에 해당 이름을 지정할 수 있습니다.
- 경로별: 변수 값은 경로별 속성을 참조할 수 있습니다. 자세한 내용은 [경로를 사용하여 사용자 지정 복합 모델 속성 참조](#) 단원을 참조하십시오.

Note

변수는 AWS IoT SiteWise 콘솔에서 지원되지 않습니다. AWS Command Line Interface (AWS CLI) 및 AWS SDKs를 포함하여 AWS IoT SiteWise API에서 사용됩니다.

에서 응답으로 수신하는 변수에는 ID와 경로를 포함하여 값에 대한 전체 정보가 AWS IoT SiteWise 포함됩니다.

그러나 변수를 전달할 때 AWS IoT SiteWise (예: "생성" 또는 "업데이트" 호출에서)이 중 하나만 지정하면 됩니다. 예를 들어 경로를 지정하는 경우 ID를 제공할 필요가 없습니다.

수식 표현식에 리터럴 사용

AWS IoT SiteWise 는 표현식 및 수식에서 리터럴 사용을 지원합니다. 리터럴은 특정 데이터 유형을 나타내는 고정 값입니다. 예에서는 수식 표현식에서 숫자 및 문자열 리터럴을 정의할 AWS IoT SiteWise 수 있습니다. 리터럴은 데이터 변환, 경보 조건 및 시각화 계산을 비롯한 다양한 컨텍스트에서 사용할 수 있습니다.

- 숫자**

숫자와 과학 표기법을 사용하여 정수와 실수를 정의합니다. [E 표기법](#)을 사용하여 숫자를 과학적 표기법으로 표현할 수 있습니다.

예: 1, 2.0, .9, -23.1, 7.89e3, 3.4E-5

- 문자열**

'(따옴표) 및 "(큰따옴표) 문자를 사용하여 문자열을 정의합니다. 시작과 끝의 따옴표 유형은 일치해야 합니다. 문자열을 선언할 때 사용하는 것과 일치하는 따옴표를 이스케이프하려면 해당 따옴표를 두 번 포함시키세요. AWS IoT SiteWise 문자열에서 유일한 이스케이프 문자입니다.

예: 'active', "inactive", '{"temp": 52}', {"temp": "high"}

수식 표현식에 연산자 사용

공식 표현식에는 다음과 같은 일반적인 수학 연산자를 사용할 수 있습니다.

연산자	설명
+	두 피연산자가 모두 숫자인 경우 이 연산자는 왼쪽 및 오른쪽 피연산자를 더합니다. 피연산자 중 하나가 문자열인 경우 이 연산자는 왼쪽 및 오른쪽 피연산자를 문자열로 연결합니다. 예를 들어, 표현식 <code>1 + 2 + " is three"</code>은 <code>"3 is three"</code>로 평가됩니다. 문자열은 최대 1,024자를 가질 수 있습니다. 문자열이 1,024자를 초과하는 경우 AWS IoT SiteWise에서는 해당 계산에 대한 데이터 포인트를 출력하지 않습니다.
-	왼쪽 피연산자에서 오른쪽 피연산자를 뺍니다. 이 연산자는 숫자 피연산자에만 사용할 수 있습니다.
/	왼쪽 피연산자를 오른쪽 피연산자로 나눕니다. 이 연산자는 숫자 피연산자에만 사용할 수 있습니다.
*	왼쪽 피연산자와 오른쪽 피연산자를 곱합니다. 이 연산자는 숫자 피연산자에만 사용할 수 있습니다.
^	왼쪽 피연산자를 오른쪽 피연산자의 거듭제곱으로 올립니다(지수). 이 연산자는 숫자 피연산자에만 사용할 수 있습니다.
%	왼쪽 피연산자를 오른쪽 피연산자로 나눈 나머지를 반환합니다. 결과는 왼쪽 피연산자와 동일한 부호를 가집니다. 이 동작은 모듈로 작업과 다릅니다.

연산자	설명
	이 연산자는 숫자 피연산자에만 사용할 수 있습니다.
$x < y$	x가 y보다 작으면 1을 반환하고, 그렇지 않으면 0를 반환합니다.
$x > y$	x가 y보다 크면 1을 반환하고, 그렇지 않으면 0를 반환합니다.
$x \leq y$	x가 y보다 작거나 같으면 1을 반환하고, 그렇지 않으면 0를 반환합니다.
$x \geq y$	x가 y보다 크거나 같으면 1을 반환하고, 그렇지 않으면 0를 반환합니다.
$x == y$	x가 y와 같으면 1을 반환하고, 그렇지 않으면 0를 반환합니다.
$x != y$	x가 y와 같지 않으면 1을 반환하고, 그렇지 않으면 0를 반환합니다.
$!x$	x가 0(거짓)으로 평가되면 1을 반환하고, 그렇지 않으면 0를 반환합니다. 다음과 같은 경우 x는 거짓으로 평가됩니다. - x는 숫자형 피연산자이며, 0로 평가됩니다. - x는 빈 문자열로 평가됩니다. - x는 빈 배열로 평가됩니다. - x는 None로 평가됩니다.

연산자	설명
<code>x and y</code>	<code>x</code>가 0(거짓)로 평가되면 0을 반환합니다. 그렇지 않으면 <code>y</code>의 평가된 결과를 반환합니다. <code>x</code> 또는 <code>y</code>는 다음과 같은 경우 거짓으로 평가됩니다. <code>x</code> 또는 <code>y</code>는 숫자형 피연산자이며, 0로 평가됩니다. <code>x</code> 또는 <code>y</code>는 빈 문자열로 평가됩니다. <code>x</code> 또는 <code>y</code>는 빈 배열로 평가됩니다. <code>x</code> 또는 <code>y</code>는 None으로 평가됩니다.
<code>x or y</code>	<code>x</code>가 1(참)로 평가되면 1을 반환합니다. 그렇지 않으면 <code>y</code>의 평가된 결과를 반환합니다. <code>x</code> 또는 <code>y</code>는 다음과 같은 경우 거짓으로 평가됩니다. <code>x</code> 또는 <code>y</code>는 숫자형 피연산자이며, 0로 평가됩니다. <code>x</code> 또는 <code>y</code>는 빈 문자열로 평가됩니다. <code>x</code> 또는 <code>y</code>는 빈 배열로 평가됩니다. <code>x</code> 또는 <code>y</code>는 None으로 평가됩니다.
<code>not x</code>	<code>x</code>가 0(거짓)으로 평가되면 1을 반환하고, 그렇지 않으면 0을 반환합니다. 다음과 같은 경우 <code>x</code>는 거짓으로 평가됩니다. <code>x</code>는 숫자형 피연산자이며, 0로 평가됩니다. <code>x</code>는 빈 문자열로 평가됩니다. <code>x</code>는 빈 배열로 평가됩니다. <code>x</code>는 None로 평가됩니다.

연산자	설명
<code>[]</code> <code>s[index]</code>	문자열 <code>s</code>의 인덱스 <code>index</code>에 있는 문자를 반환합니다. 이 구문은 Python의 인덱스 구문과 동등합니다. Example 예시 • <code>"Hello!"[1]</code> 에서 <code>e</code>를 반환합니다. • <code>"Hello!"[-2]</code> 에서 <code>o</code>를 반환합니다.

연산자	설명
[] s[start:end:step]	문자열 s의 한 조각을 반환합니다. 이 구문은 Python의 조각 구문과 동등합니다. 이 연산자에는 다음과 같은 인수가 사용됩니다. • start - (선택 사항) 조각의 포함 시작 인덱스입니다. 기본값은 0입니다. • end - (선택 사항) 조각의 배타적 끝 인덱스입니다. 기본값은 문자열 길이입니다. • step - (선택 사항) 조각의 각 단계마다 증가할 수 있습니다. 예를 들어, 다른 모든 문자가 포함된 조각을 반환하도록 2를 지정하거나 조각을 반대로 -1을 반환하도록 지정할 수 있습니다. 기본값은 1입니다. step 인수를 생략하여 기본값을 사용할 수 있습니다. 예를 들어, s[1:4:1]은 s[1:4]과 같습니다. 인수는 정수이거나 없음 상수여야 합니다. 를 지정하면 해당 인수의 기본값을 none AWS IoT SiteWise 사용합니다. Example 예시 • "Hello!"[1:4] 에서 "ell"를 반환합니다. • "Hello!"[:2] 에서 "He"를 반환합니다. • "Hello!"[3:] 에서 "lo!"를 반환합니다. • "Hello!"[:-4] 에서 "He"를 반환합니다. • "Hello!"[:2] 에서 "Hl"를 반환합니다. • "Hello!"[::-1] 에서 "!olleH"를 반환합니다.

수식 표현식에 상수 사용

에서 표현식 및 공식에 상수를 사용하여 고정 값 또는 사전 정의된 파라미터를 나타낼 AWS IoT SiteWise 수 있습니다. 상수는 데이터 변환, 경보 조건 또는 시각화 계산과 같은 다양한 컨텍스트에서 사용할 수 있습니다. 상수를 사용하면 표현식을 단순화하고 더 쉽게 읽고 유지 관리할 수 있습니다.

표현식에는 다음과 같은 일반적인 수학 상수를 사용할 수 있습니다. 모든 상수는 대소문자를 구분하지 않습니다.

Note

상수와 이름이 같은 변수를 정의하면 변수가 상수보다 우선합니다.

상수	설명
pi	숫자 pi(π): 3.141592653589793
e	숫자 e: 2.718281828459045
true	숫자 1에 해당합니다. 에서 AWS IoT SiteWise 부울은 해당 숫자로 변환됩니다.
false	숫자 0에 해당합니다. 에서 AWS IoT SiteWise 부울은 해당 숫자로 변환됩니다.
none	값이 없는 것과 같습니다. 이 상수를 사용하여 조건부 표현식 의 결과로 아무것도 출력하지 않을 수 있습니다.

수식 표현식에서 함수 사용

다음 함수를 사용하여 공식 표현식의 데이터에 대해 작업할 수 있습니다.

변환과 지표는 서로 다른 함수를 지원합니다. 다음 표에는 각 유형의 공식 속성과 호환되는 함수 유형이 나와 있습니다.

Note

공식 표현식에는 최대 10개의 함수를 포함할 수 있습니다.

함수 유형	변형	Metrics
수식 표현식에서 공통 함수 사용	 예	 예
수식 표현식에서 비교 함수 사용	 예	 예
수식 표현식에서 조건부 함수 사용	 예	 예
수식 표현식에서 문자열 함수 사용	 예	 예
수식 표현식에서 집계 함수 사용	 아니요	 예
수식 표현식에서 시간 함수 사용	 예	 예

함수 유형	변형	Metrics
수식 표현식에서 날짜 및 시간 함수 사용	 예	 예

함수 구문

다음 구문을 사용하여 함수를 만들 수 있습니다.

일반 구문

일반 구문에서는 함수 이름 뒤에 0개 이상의 인수가 포함된 괄호가 옵니다.

function_name(argument1, argument2, argument3, ...). 예를 들어, 일반 구문을 사용하는 함수는 `log(x)` 및 `contains(s, substring)`와 비슷할 수 있습니다.

UFCS(균일한 함수 호출 구문)

UFCS를 사용하면 객체 지향 프로그래밍의 메서드 호출 구문을 사용하여 함수를 호출할 수 있습니다. UFCS에서는 첫 번째 인수 다음에 점(.), 함수 이름, 나머지 인수(있는 경우)가 괄호 안에 옵니다.

argument1.function_name(argument2, argument3, ...). 예를 들어, UFCS를 사용하는 함수는 `x.log()` 및 `s.contains(substring)`와 비슷할 수 있습니다.

UFCS를 사용하여 후속 함수를 연결할 수도 있습니다.는 현재 함수의 평가 결과를 다음 함수의 첫 번째 인수로 AWS IoT SiteWise 사용합니다.

예를 들어, `contains(lower(jp(message, '$.status')), 'fail')` 대신 `message.jp('$status').lower().contains('fail')`를 사용할 수 있습니다.

자세한 내용은 [D 프로그래밍 언어](#) 웹 사이트를 방문하세요.

Note

모든 AWS IoT SiteWise 함수에 UFCS를 사용할 수 있습니다.

AWS IoT SiteWise 함수는 대/소문자를 구분하지 않습니다. 예를 들어, `lower(s)` 및 `Lower(s)`를 서로 바꿔서 사용할 수 있습니다.

수식 표현식에서 공통 함수 사용

[변환](#)과 [지표](#)에서는 다음 함수를 사용하여 변환과 지표의 일반적인 수학 함수를 계산할 수 있습니다.

함수	설명
<code>abs(x)</code>	x의 절대값을 반환합니다.
<code>acos(x)</code>	x의 아크코사인을 반환합니다.
<code>asin(x)</code>	x의 아크사인을 반환합니다.
<code>atan(x)</code>	x의 아크탄젠트를 반환합니다.
<code>cbrt(x)</code>	x의 세제곱근을 반환합니다.
<code>ceil(x)</code>	x보다 큰 가장 가까운 정수를 반환합니다.
<code>cos(x)</code>	x의 코사인을 반환합니다.
<code>cosh(x)</code>	x의 쌍곡코사인을 반환합니다.
<code>cot(x)</code>	x의 코탄젠트를 반환합니다.
<code>exp(x)</code>	e를 x의 거듭제곱으로 반환합니다.
<code>expm1(x)</code>	$\exp(x) - 1$ 을 반환합니다. 이 함수를 사용하면 x의 작은 값에 대해 $\exp(x) - 1$ 를 더 정확하게 계산할 수 있습니다.
<code>floor(x)</code>	x보다 작은 가장 가까운 정수를 반환합니다.
<code>log(x)</code>	x의 $\log_e$ (밑 e) 반환합니다.
<code>log10(x)</code>	x의 $\log_{10}$ (밑 10) 반환합니다.

함수	설명
<code>log1p(x)</code>	$\log(1 + x)$ 을 반환합니다. 이 함수를 사용하면 x 의 작은 값에 대해 $\log(1 + x)$ 를 더 정확하게 계산할 수 있습니다.
<code>log2(x)</code>	x 의 $\log_2$ (밑 2)반환합니다.
<code>pow(x, y)</code>	x 를 y 의 거듭제곱으로 반환합니다. 이는 x^y 와 동일합니다.
<code>signum(x)</code>	x 의 부호를 반환합니다(음수를 입력하면 -1, 0을 입력하면 0, 양수를 입력하면 +1).
<code>sin(x)</code>	x 의 사인을 반환합니다.
<code>sinh(x)</code>	x 의 쌍곡사인을 반환합니다.
<code>sqrt(x)</code>	x 의 제곱근을 반환합니다.
<code>tan(x)</code>	x 의 탄젠트를 반환합니다.
<code>tanh(x)</code>	x 의 쌍곡탄젠트를 반환합니다.

수식 표현식에서 비교 함수 사용

[변환](#) 및 [지표](#)에서 다음 비교 함수를 사용하여 두 값과 출력1(true) 또는 0 (false)을 비교할 수 있습니다. 는 [어휘 순서](#)별로 문자열을 AWS IoT SiteWise 비교합니다.

함수	설명
<code>gt(x, y)</code>	x가 y보다 크면 1을 반환하고, 그렇지 않으면 0을 반환합니다($x > y$). 이 함수는 x 및 y가 숫자 및 문자열과 같이 호환되지 않는 유형인 경우 값을 반환하지 않습니다.
<code>gte(x, y)</code>	x 가 y 보다 크거나 같으면 1을 반환하고, 그렇지 않으면 0을 반환합니다($x \geq y$).

함수	설명
<code>eq(x, y)</code>	AWS IoT SiteWise 는 인수가의 상대 허용 오차 내에 있는 경우 인수를 동일하게 간주합니다 $1E-9$. 이것은 Python의 isclose 함수와 비슷하게 동작합니다. 이 함수는 x 및 y가 숫자 및 문자열과 같이 호환되지 않는 유형인 경우 값을 반환하지 않습니다. x가 y와 같으면 1을 반환하고, 그렇지 않으면 0을 반환합니다(<code>x == y</code>). AWS IoT SiteWise 는 인수가의 상대 허용 오차 내에 있는 경우 인수를 동일하게 간주합니다 $1E-9$. 이것은 Python의 isclose 함수와 비슷하게 동작합니다. 이 함수는 x 및 y가 숫자 및 문자열과 같이 호환되지 않는 유형인 경우 값을 반환하지 않습니다.
<code>lt(x, y)</code>	x가 y보다 작으면 1을 반환하고, 그렇지 않으면 0을 반환합니다(<code>x < y</code>). 이 함수는 x 및 y가 숫자 및 문자열과 같이 호환되지 않는 유형인 경우 값을 반환하지 않습니다.
<code>lte(x, y)</code>	x가 y보다 작거나 같으면 1을 반환하고, 그렇지 않으면 0을 반환합니다(<code>x ≤ y</code>). AWS IoT SiteWise 는 인수가의 상대 허용 오차 내에 있는 경우 인수를 동일하게 간주합니다 $1E-9$. 이것은 Python의 isclose 함수와 비슷하게 동작합니다. 이 함수는 x 및 y가 숫자 및 문자열과 같이 호환되지 않는 유형인 경우 값을 반환하지 않습니다.

함수	설명
<code>isnan(x)</code>	<code>x</code>가 NaN와 같으면 1을 반환하고, 그렇지 않으면 0을 반환합니다. 이 함수는 <code>x</code>가 문자열인 경우 값을 반환하지 않습니다.

수식 표현식에서 조건부 함수 사용

[변환](#) 및 [지표](#)에서 다음 함수를 사용하여 조건을 확인하고 조건이 참으로 평가되는지 거짓으로 평가되는지에 관계없이 다양한 결과를 반환할 수 있습니다.

함수	설명
<code>if(condition, result_if_true, result_if_false)</code>	<code>condition</code> 을 평가하고 조건이 참으로 평가 되면 <code>result_if_true</code> 를 반환하고 조건이 false로 평가되면 <code>result_if_false</code> 를 반환합니다. <code>condition</code> 은 숫자여야 합니다. 이 함수는 0 및 빈 문자열을 false로 간주하고 그 밖의 모든 것(NaN 포함)을 true로 간주합니다. 부울은 0(거짓)과 1(참)로 변환됩니다. 이 함수에서 none 상수를 반환하여 특정 조건에 대한 출력값을 무시할 수 있습니다. 즉, 조건을 충족하지 않는 데이터 포인트를 필터링할 수 있습니다. 자세한 내용은 데이터 포인트 필터링 단원을 참조하십시오. Example 예시 <code>if(0, x, y)</code>에서는 변수 <code>y</code>를 반환합니다. <code>if(5, x, y)</code>에서는 변수 <code>x</code>를 반환합니다. <code>if(gt(temp, 300), x, y)</code>에서는 변수 <code>temp</code>가 300보다 크면 변수 <code>x</code>를 반환합니다.

함수	설명
	• <code>if(gt(temp, 300), temp, none)</code> 에서 는 300보다 크거나 같으면 변수 <code>temp</code>를 반환 하고, <code>temp</code>가 300보다 작으면 <code>none</code>(값 없음) 을 반환합니다. 하나 이상의 인수가 조건 함수인 중첩 조건 함수에는 UFCS를 사용하는 것이 좋습니다. <code>if(condition, result_if_true)</code> 를 사용하여 조건을 평가하고 <code>elif(cond ition, result_if_true, result_if _false)</code> 를 사용하여 추가 조건을 평가할 수 있습니다. 예를 들어, <code>if(condition1, result1_i f_true, if(condition2, result2_i f_true, result2_if_false))</code> 대신 <code>if(condition1, result1_if_true).e lif(condition2, result2_if_true, result2_if_false)</code> 를 사용할 수 있습니다. 중간 조건 함수를 추가로 연결할 수도 있습니다. 예를 들어, <code>if(condition1, result1_i f_true).elif(condition2, result2_if_true).elif(condi tion3, result3_if_true, result3_i f_false)</code> 와 같이 여러 <code>if</code> 문을 중첩하는 대 신 <code>if(condition1, result1_if_true, if(condition2, result2_if_true, if(condition3, result3_if_true result3_if_false)))</code> 를 사용할 수 있습 니다.

함수	설명
	 Important UFCS와 함께 <code>elif(condition, result_if_true, result_if_false)</code> 를 사용해야 합니다.

수식 표현식에서 문자열 함수 사용

[변환](#) 및 [지표](#)에서는 다음 함수를 사용하여 문자열에 대한 작업을 할 수 있습니다. 자세한 내용은 [수식에서 문자열 사용](#) 단원을 참조하십시오.

Important

공식 표현식은 실수 또는 문자열 값만 출력할 수 있습니다. 중첩 표현식은 문자열과 같은 다른 데이터 유형을 출력할 수 있지만 공식 전체는 숫자나 문자열로 평가되어야 합니다. [jp 함수](#)를 사용하여 문자열을 숫자로 변환할 수 있습니다. 부울 값은 1(참) 또는 0(거짓)이어야 합니다. 자세한 내용은 [정의되지 않은 값, 무한 값 및 오버플로우 값](#) 단원을 참조하십시오.

함수	설명
<code>len(s)</code>	문자열 <code>s</code> 의 길이를 반환합니다.
<code>find(s, substring)</code>	문자열 <code>s</code> 에 있는 문자열 <code>substring</code> 의 인덱스를 반환합니다.
<code>contains(s, substring)</code>	문자열 <code>s</code> 에 문자열 <code>substring</code> 가 포함되어 있으면 1을 반환하고, 그렇지 않으면 0을 반환합니다.
<code>upper(s)</code>	문자열 <code>s</code> 를 대문자 형태로 반환합니다.
<code>lower(s)</code>	문자열 <code>s</code> 를 소문자 형식으로 반환합니다.
	JSONPath 표현식 <code>json_path</code> 로 문자열 <code>s</code> 를 평가하고 결과를 반환합니다.

함수	설명
<code>jp(s, json_path)</code>	이 함수를 사용하여 다음을 수행합니다. - 직렬화된 JSON 구조에서 값, 배열 또는 객체를 추출합니다. - 문자열을 숫자로 변환합니다. 예를 들어, 공식 <code>jp('111', '\$')</code>에서 숫자로 111을 반환합니다. JSON 구조에서 문자열 값을 추출하여 숫자로 반환하려면 여러 중첩 <code>jp</code> 함수를 사용해야 합니다. 외부 <code>jp</code> 함수는 JSON 구조에서 문자열을 추출하고, 내부 <code>jp</code> 함수는 문자열을 숫자로 변환합니다. 문자열 <code>json_path</code> 는 문자열 리터럴을 포함해야 합니다. 즉, <code>json_path</code> 는 문자열로 평가되는 표현식일 수 없습니다. Example 예시 <code>jp('{ "status": "active", "value": 15 }', '\$.value')</code> 에서 15를 반환합니다. <code>jp('{ "measurement": { "reading": 25, "confidence": 0.95 } }', '\$.measurement.reading')</code> 에서 25를 반환합니다. <code>jp('[2,8,23]', '\$[2]')</code> 에서 23를 반환합니다. <code>jp('{ "values": [3,6,7] }', '\$.values[1]')</code> 에서 6를 반환합니다. <code>jp('111', '\$')</code>에서 111를 반환합니다. <code>jp(jp('{ "measurement": { "reading": 25, "confidence": 0.95</code>

함수	설명
<code>join(s0, s1, s2, s3, ...)</code>	"}}', '\$.measurement.confidence'), '\$') 에서 0.95를 반환합니다. 구분 기호와 함께 연결된 문자열을 반환합니다. 이 함수는 첫 번째 입력 문자열을 구분 기호로 사용하고 나머지 입력 문자열을 결합합니다. 이는 Java의 조인(CharSequence 구분 기호, CharSequence... 요소) 함수와 비슷하게 동작합니다. Example 예시 <code>join("-", "aa", "bb", "cc")</code>에서 <code>aa-bb-cc</code> 반환
<code>format(expression: "format")</code> 또는 <code>format("format", expression)</code>	지정된 형식의 문자열을 반환합니다. 이 함수는 <code>expression</code> 값으로 평가한 다음 지정된 형식으로 값을 반환합니다. 이 함수는 Java의 형식(문자열 형식, Object... args) 함수와 비슷하게 동작합니다. 지원되는 형식에 대한 자세한 내용은 Java 플랫폼, Standard Edition 7 API 사양의 클래스 포맷터에서 변환을 참조하세요. Example 예시 <code>format(100+1: "d")</code> 에서 문자열 101을 반환합니다. <code>format("The result is %d", 100+1)</code>에서 문자열 The result is 101을 반환합니다.

함수	설명
<code>f'expression'</code>	연결된 문자열을 반환합니다. 이 형식이 지정된 함수를 사용하면 간단한 표현식을 사용하여 문자열을 연결하고 서식을 지정할 수 있습니다. 이러한 함수에는 중첩된 표현식이 포함될 수 있습니다. {}(중괄호)를 사용하여 표현식을 보간할 수 있습니다. 이것은 Python의 형식이 지정된 문자열 리터럴과 유사하게 동작합니다. Example 예시 <code>f'abc{1+2: "f"}d'</code> 에서 <code>abc3.000000d</code> 를 반환합니다. 이 예제 표현식을 평가하려면 다음을 수행하세요. <code>format(1+2: "f")</code> 에서 부동 소수점 숫자 <code>3.000000</code>를 반환합니다. <code>join('', "abc", 1+2, 'd')</code>에서 문자열 <code>abc3.000000d</code> 을 반환합니다. <code>join('', "abc", format(1+2: "f"), 'd')</code>와 같은 방법으로 표현식을 작성할 수도 있습니다.

수식 표현식에서 집계 함수 사용

[지표](#)에 한해, 다음 함수를 사용하여 각 시간 간격 동안 입력 값을 집계하고 단일 출력 값을 계산할 수 있습니다. 집계 함수는 연결된 자산의 데이터를 집계할 수 없습니다.

집계 함수 인수는 [변수](#), [숫자 리터럴](#), [시간 함수](#), 중첩 표현식 또는 집계 함수일 수 있습니다. 이 공식 `max(latest(x), latest(y), latest(z))`는 집계 함수를 인수로 사용하고 `x`, `y`, `z` 속성의 현재 최대 값을 반환합니다.

집계 함수에서 중첩 표현식을 사용할 수 있습니다. 중첩된 표현식을 사용할 때는 다음 규칙이 적용됩니다.

- 각 인수에는 변수가 하나만 있을 수 있습니다.

Example

예를 들어, $\text{avg}(x \cdot (x-1))$ 및 $\text{sum}(x/2) / \text{avg}(y^2)$ 가 지원됩니다.

예를 들어, $\min(x/y)$ 는 지원되지 않습니다.

- 각 인수에는 다단계 중첩 표현식이 있을 수 있습니다.

Example

예를 들어 $\text{sum}(\text{avg}(x^2)/2)$ 은 지원됩니다.

- 인수마다 변수가 다를 수 있습니다.

Example

예를 들어 $\text{sum}(x/2, y \cdot 2)$ 은 지원됩니다.

Note

- 표현식에 측정값이 포함된 경우는 측정값에 대해 현재 시간 간격의 마지막 값을 AWS IoT SiteWise 사용하여 집계를 계산합니다.
- 표현식에 속성이 포함된 경우는 속성의 최신 값을 AWS IoT SiteWise 사용하여 집계를 계산합니다.

함수	설명
$\text{avg}(x_0, \dots, x_n)$	현재 시간 간격 동안 해당 변수 값의 평균을 반환합니다. 이 함수는 지정된 변수에 현재 시간 간격 동안 하나 이상의 데이터 요소가 있는 경우에만 데이터 포인트를 출력합니다.
$\text{sum}(x_0, \dots, x_n)$	현재 시간 간격 동안 해당 변수 값의 합계를 반환합니다.

함수	설명
	이 함수는 지정된 변수에 현재 시간 간격 동안 하나 이상의 데이터 요소가 있는 경우에만 데이터 포인트를 출력합니다.
$\min(x_0, \dots, x_n)$	현재 시간 간격 동안 해당 변수 값의 최소값을 반환합니다. 이 함수는 지정된 변수에 현재 시간 간격 동안 하나 이상의 데이터 요소가 있는 경우에만 데이터 포인트를 출력합니다.
$\max(x_0, \dots, x_n)$	현재 시간 간격 동안 해당 변수 값의 최대값을 반환합니다. 이 함수는 지정된 변수에 현재 시간 간격 동안 하나 이상의 데이터 요소가 있는 경우에만 데이터 포인트를 출력합니다.
$\text{count}(x_0, \dots, x_n)$	현재 시간 간격 동안 해당 변수에 대한 총 데이터 포인트 수를 반환합니다. 조건을 충족하는 데이터 포인트 수를 계산하는 방법에 대한 자세한 내용은 조건과 일치하는 데이터 포인트를 계산하려면 다음을 수행하세요. 단원을 참조하세요. 이 함수는 모든 시간 간격에 대한 데이터 포인트를 계산합니다.
$\text{stdev}(x_0, \dots, x_n)$	현재 시간 간격 동안 지정된 변수 값의 표준 편차를 반환합니다. 이 함수는 지정된 변수에 현재 시간 간격 동안 하나 이상의 데이터 요소가 있는 경우에만 데이터 포인트를 출력합니다.

수식 표현식에서 시간 함수 사용

시간 함수를 사용하여 데이터 포인트의 타임스탬프를 기반으로 값을 반환합니다.

지표에서 시간 함수 사용

[지표](#)에서만 데이터 포인트의 타임스탬프를 기반으로 값을 반환하는 다음 함수를 사용할 수 있습니다.

시간 함수 인수는 로컬 자산 모델 또는 중첩 표현식의 속성이어야 합니다. 즉, 하위 자산 모델의 속성을 시간 함수에 사용할 수 없습니다.

시간 함수의 중첩 표현식을 사용할 수 있습니다. 중첩된 표현식을 사용할 때는 다음 규칙이 적용됩니다.

- 각 인수에는 변수가 하나만 있을 수 있습니다.

예를 들어 `latest( t*9/5 + 32 )`은 지원됩니다.

- 인수는 집계 함수일 수 없습니다.

예를 들어 `first( sum(x) )`은 지원되지 않습니다.

함수	설명
<code>first(x)</code>	현재 시간 간격 동안 해당 변수의 가장 오래된 타임스탬프가 있는 값을 반환합니다.
<code>last(x)</code>	현재 시간 간격 동안 해당 변수의 가장 최근 타임스탬프가 있는 값을 반환합니다.
<code>earliest(x)</code>	현재 시간 간격 시작 이전에 해당 변수의 가장 최근 값을 반환합니다. 입력 속성의 기록에 데이터 포인트가 하나 이상 있는 경우 이 함수는 1시간 간격으로 데이터 포인트를 계산합니다. 세부 정보는 time-range-definition 섹션을 참조하세요.
<code>latest(x)</code>	현재 시간 간격 종료 이전에 해당 변수의 가장 최근 타임스탬프가 있는 값을 반환합니다. 입력 속성의 기록에 데이터 포인트가 하나 이상 있는 경우 이 함수는 1시간 간격으로 데이터

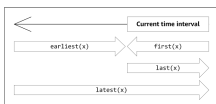
함수	설명
	포인트를 계산합니다. 세부 정보는 time-range-definition 섹션을 참조하세요.
statetime(x)	현재 시간 간격 동안 해당 변수가 양수인 시간(초)을 반환합니다. 비교 함수를 사용하면 statetime 함수가 사용할 변형 속성을 만들 수 있습니다. 예를 들어 Idle 속성(0 또는 1)이 있는 경우, IdleTime = statetime(Idle) 표현식을 사용하여 시간 간격당 유휴 시간을 계산할 수 있습니다. 자세한 내용은 상태 시간 시나리오 예를 참조하세요. 이 함수는 지표 속성을 입력 변수로 지원하지 않습니다. 입력 속성의 기록에 데이터 포인트가 하나 이상 있는 경우 이 함수는 1시간 간격으로 데이터 포인트를 계산합니다.

함수	설명
<code>TimeWeightedAvg(x, [interpolation])</code>	포인트 간 시간 간격을 두고 가중치를 적용한 입력 데이터의 평균을 반환합니다. 계산 및 간격에 대한 자세한 내용은 시간 가중 함수 파라미터를 참조하세요. 선택적 인수 <code>interpolation</code> 는 문자열 상수여야 합니다. • <code>locf</code> - 이 값이 기본값입니다. 계산에서는 데이터 포인트 간 간격에 대해 마지막으로 관찰된 캐리 포워드 계산 알고리즘을 사용합니다. 이 접근 방식에서는 데이터 포인트가 다음 입력 데이터 포인트 타임스탬프까지 마지막으로 관측된 값으로 계산됩니다. 양호한 데이터 지점 이후의 값은 다음 데이터 포인트 타임스탬프까지의 값으로 추정됩니다. • <code>linear</code> - 계산에서는 데이터 포인트 간 간격에 대한 선형 보간 계산 알고리즘을 사용합니다. 양호한 두 데이터 포인트 사이의 값은 해당 데이터 포인트 값 간의 선형 보간으로 추정됩니다. 양호한 데이터 포인트와 불량 데이터 포인트 사이의 값 또는 마지막 양호한 데이터 포인트 이후의 값은 양호한 데이터 포인트로 추정됩니다.

함수	설명
<code>TimeWeightedStDev(x, [algo])</code>	포인트 간 시간 간격으로 가중치를 적용한 입력 데이터의 표준 편차를 반환합니다. 계산 및 간격에 대한 자세한 내용은 시간 가중 함수 파라미터를 참조하세요. 계산에서는 데이터 포인트 간 간격에 대해 마지막으로 관찰된 캐리 포워드 계산 알고리즘을 사용합니다. 이 접근 방식에서는 데이터 포인트가 다음 입력 데이터 포인트 타임스탬프까지 마지막으로 관찰된 값으로 계산됩니다. 가중치는 데이터 포인트 또는 창 경계 사이의 시간 간격(초)으로 계산됩니다. 선택적 인수 <code>algo</code>는 문자열 상수여야 합니다. • <code>f</code> - 이 값이 기본값입니다. 빈도 가중치를 사용하여 편향 없는 가중치 표본 분산을 반환합니다. 여기서 <code>TimeWeight</code>는 초 단위로 계산됩니다. 이 알고리즘은 일반적으로 표준 편차 이하로 가정되며 가중치 적용 표본에 대한 Bessel의 표준 편차 보정이라고 합니다. • <code>p</code> - 모집단 분산이라고도 하는 편향 가중 표본 분산을 반환합니다. 다음과 같은 경우 계산에 다음 공식이 사용됩니다. • S_p = 모집단 표준 편차 • S_f = 빈도 표준 편차 • X_i = 수신 데이터 • ω_i = 시간 간격(초)과 동일한 가중치 • μ^* = 수신 데이터의 가중 평균

함수	설명
	모집단 표준 편차 방정식: $S_p^2 = \frac{\sum_{i=1}^N \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^N \omega_i}$ 빈도 표준 편차 방정식: $S_f^2 = \frac{\sum_{i=1}^N \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^N \omega_i - 1}$

다음 다이어그램은가 현재 시간 간격을 기준으로 시간 함수 first, latest, last, earliest를 AWS IoT SiteWise 계산하는 방법을 보여줍니다.



Note

- first(x), last(x)의 시간 범위는 (현재 기간 시작, 현재 기간 종료]입니다.
- latest(x)의 시간 범위는 (시간 시작, 현재 기간 종료]입니다.
- earliest(x)의 시간 범위는 (시간 시작, 이전 기간 종료]입니다.

시간 가중 함수 파라미터

집계 기간에 대해 계산된 시간 가중 함수는 다음 사항을 고려합니다.

- 기간 내부의 데이터 포인트
- 데이터 포인트 간 시간 간격
- 기간 전의 마지막 데이터 포인트
- 기간 뒤의 첫 번째 데이터 포인트(일부 알고리즘의 경우)

용어:

- 잘못된 데이터 포인트 - 품질이 좋지 않거나 숫자 값이 아닌 모든 데이터 포인트. 이는 기간 결과 계산에서 고려되지 않습니다.
- 잘못된 간격 - 잘못된 데이터 지점 이후의 간격입니다. 알려진 첫 번째 데이터 지점 이전의 간격도 잘못된 간격으로 간주됩니다.
- 양호한 데이터 포인트 - 품질이 우수하고 수치 값이 있는 모든 데이터 포인트.

 Note

- AWS IoT SiteWise 는 변환 및 지표를 계산할 때만 GOOD 품질 데이터를 사용합니다. UNCERTAIN 및 BAD 데이터 포인트를 무시합니다.
- 알려진 첫 번째 데이터 포인트 이전의 간격은 잘못된 간격으로 간주됩니다. 자세한 정보는 [the section called “공식 표현식 튜토리얼”](#)을 참조하세요.

마지막으로 알려진 데이터 포인트 이후의 간격은 무기한 계속되므로 이후의 모든 기간에 영향을 미칩니다. 새 데이터 포인트가 도착하면 함수는 간격을 다시 계산합니다.

위 규칙에 따라 기간 집계 결과가 계산되고 기간 경계로 제한됩니다. 기본적으로 함수는 전체 기간의 간격이 적절한 경우에만 기간 결과를 전송합니다.

기간의 적절한 간격이 기간 길이보다 작은 경우 함수는 기간을 전송하지 않습니다.

기간 결과에 영향을 미치는 데이터 포인트가 변경되면 함수는 데이터 포인트가 기간 외부에 있더라도 기간을 다시 계산합니다.

입력 속성의 기록에 데이터 요소가 하나 이상 있고 계산이 시작된 경우 함수는 시간 간격마다 시간 가중 집계 함수를 계산합니다.

Example 상태 시간 시나리오 예

다음 속성을 가진 자산이 있는 예를 고려하세요.

- Idle - 0 또는 1인 측정입니다. 값이 1이면 머신이 유휴 상태입니다.
- Idle Time - 공식 `statetime(Idle)`를 사용하여 1분 간격으로 머신이 유휴 상태인 시간(초)을 계산하는 지표입니다.

Idle 속성은 다음과 같은 데이터 포인트를 갖습니다.

타임스탬프	2:00:00 PM	2:00:30 PM	2:01:15 PM	2:02:45 PM	2:04:00 PM
Idle	0	1	1	0	0

AWS IoT SiteWise 는 값에서 1분마다 Idle Time 속성을 계산합니다Idle. 이 계산이 완료되면 Idle Time 속성은 다음과 같은 데이터 포인트를 갖습니다.

타임스탬프	2:00:00 PM	2:01:00 PM	2:02:00 PM	2:03:00 PM	2:04:00 PM
Idle Time	N/A	30	60	45	0

AWS IoT SiteWise 는 1분마다 다음과 같은 계산Idle Time을 수행합니다.

- 2:00 PM(1:59 PM ~ 2:00 PM)
 - 2:00 PM 이전에는 Idle에 대한 데이터가 없으므로 데이터 포인트가 계산되지 않습니다.
- 2:01 PM(2:00 PM ~ 2:01 PM)
 - 2:00:00 PM에 머신이 활성 상태입니다(Idle이 0임).
 - 2:00:30 PM에 머신이 유휴 상태입니다(Idle가 1임).
 - Idle은 2:01:00 PM에 간격이 끝나기 전에 다시 변경되지 않으므로 Idle Time은 30초입니다.
- 2:02 PM(2:01 PM ~ 2:02 PM)
 - 2:01:00 PM에 머신이 유휴 상태입니다(2:00:30 PM의 마지막 데이터 포인트에 따라).
 - 2:01:15 PM에 머신이 유휴 상태입니다.
 - Idle은 2:02:00 PM에 간격이 끝나기 전에 다시 변경되지 않으므로 Idle Time은 60초입니다.
- 2:03 PM(2:02 PM ~ 2:03 PM)
 - 2:02:00 PM에 머신이 유휴 상태입니다(2:01:15 PM의 마지막 데이터 포인트에 따라).
 - 2:02:45 PM에 머신이 활성 상태입니다.
 - Idle은 2:03:00 PM에 간격이 끝나기 전에 다시 변경되지 않으므로 Idle Time은 45초입니다.
- 2:04 PM(2:03 PM ~ 2:04 PM)
 - 2:03:00 PM에 머신이 유휴 상태입니다(2:02:45 PM의 마지막 데이터 포인트에 따라).
 - Idle은 2:04:00 PM에 간격이 끝나기 전에 다시 변경되지 않으므로 Idle Time은 0초입니다.

Example TimeWeightedAvg 및 TimeWeightedStDev 시나리오의 예

다음 표에는 이러한 1분 기간 지표 Avg(x), TimeWeightedAvg(x), TimeWeightedAvg(x, "linear"), stDev(x), timeWeightedStDev(x), timeWeightedStDev(x, 'p')에 대한 샘플 입력 및 출력이 나와 있습니다.

1분 집계 기간에 대한 샘플 입력:

Note

이러한 데이터 포인트의 품질은 모두 GOOD입니다.

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

집계 결과 출력:

Note

없음 - 이 기간에 대한 결과가 생성되지 않았습니다.

Time	Avg(x)	TimeWeightedAvg(x)	TimeWeightedAvg(X, "linear")	stDev(X)	timeWeightedStDev(x)	timeWeightedStDev(x, 'p')
3:00:00	4	없음	없음	0	없음	없음
3:01:00	2	4	3	0	0	0
3:02:00	14	9	13	6	5.430610041581775	5.385164807134504
3:03:00	11	13	12.875	8.54400374531753	7.724054437220943	7.659416862050705
3:04:00	0	10	2.5	0	10.084389681792215	10
3:05:00	없음	0	0	없음	0	0

변환에서 시간 함수 사용

[변환](#)의 경우에만 `pretrigger()` 함수를 사용하여 현재 변환 계산을 시작한 속성 업데이트 이전에 변수에 대한 GOOD 품질 값을 검색할 수 있습니다.

제조업체가를 사용하여 시스템 상태를 AWS IoT SiteWise 모니터링하는 예를 생각해 보십시오. 제조업체는 다음 측정값과 변환을 사용하여 프로세스를 나타냅니다.

- 측정값 `current_state`는 0 또는 1이 될 수 있습니다.
 - 머신이 청소 상태인 경우, `current_state`는 1과 같습니다.
 - 머신이 제조 상태인 경우 `current_state`는 0과 같습니다.
- `if(pretrigger(current_state) == 1, timestamp(current_state) - timestamp(pretrigger(current_state)), none)`와 동일한 변환 `cleaning_state_duration`입니다. 이 변환은 머신이 청소 상태에 있었던 시간을 초 단위의 Unix epoch 형식으로 반환합니다. 자세한 내용은 [수식 표현식에서 조건부 함수 사용](#) 및 [timestamp\(\)](#) 함수를 참조하세요.

머신이 예상보다 오래 청소 상태로 유지되는 경우 제조업체에서 머신을 조사할 수 있습니다.

또한 `pretrigger()` 함수를 다변량 변환에서 사용할 수 있습니다. 예를 들어, 이름이 x 및 y 인 측정값 두 개와 z 와 같은 변환 $x + y + \text{pretrigger}(y)$ 가 있다고 가정해 보겠습니다. 다음 표에는 오전 9시부터 오전 9시 15분까지의 x , y , z 에 대한 값이 나와 있습니다.

Note

- 이 예에서는 측정값이 시간순으로 도착한다고 가정합니다. 예를 들어 오전 9시의 x 값이 오전 9시 5분의 x 값보다 먼저 도착합니다.
- 오전 9시 5분의 데이터 포인트가 오전 9시의 데이터 포인트보다 먼저 도착하는 경우 z 는 오전 9시 5분에 계산되지 않습니다.
- 오전 9시 5분의 x 값이 오전 9시의 x 값보다 먼저 도달하고 y 값이 시간순으로 도달하면 오전 9시 5분에 z 는 $22 = 20 + 1 + 1$ 와 같습니다.

	오전 9시	오전 9시 5분	오전 9시 10분	오전 9시 15분
x	10	20		30
y	1	2	3	
$z = x + y + \text{pretrigger}(y)$	y 는 오전 9시 이전에는 데이터 포인트를 수신하지 않습니다. 따라서 오전 9시에 z 는 계산되지 않습니다.	$23 = 20 + 2 + 1$ $\text{pretrigger}(y)$ 는 1과 같습니다.	$25 = 20 + 3 + 2$ x 는 새 데이터 포인트를 받지 않습니다. $\text{pretrigger}(y)$ 는 2와 같습니다.	$36 = 30 + 3 + 3$ y 는 새 데이터 포인트를 받지 않습니다. 따라서 $\text{pretrigger}(y)$ 는 오전 9시 15분에는 3이 됩니다.

수식 표현식에서 날짜 및 시간 함수 사용

[변환](#) 및 [지표](#)에서 날짜 및 시간 함수를 다음과 같은 방식으로 사용할 수 있습니다.

- 데이터 포인트의 현재 타임스탬프를 UTC 또는 현지 시간대로 검색합니다.

- year, month, day_of_month와 같은 인수를 사용하여 타임스탬프를 생성합니다.
- unix_time 인수를 사용하여 연 또는 월과 같은 시간 기간을 추출합니다.

함수	설명
now()	Unix epoch 형식으로 현재 날짜와 시간을 초 단위로 반환합니다.
timestamp()	• 변환에서 함수는 Unix epoch 형식으로 입력 메시지의 타임스탬프(초)를 반환합니다. 변환에서만 다음 중 하나를 수행할 수 있습니다. • 변수를 함수에 인수로 제공합니다. 이 timestamp(<i>variable-name</i>) 함수는 Unix epoch 형식의 지정된 변수에 대한 최신 GOOD 품질 값의 타임스탬프(초)를 반환합니다. 예를 들어 각 온도 데이터 포인트를 섭씨에서 화씨로 변환하는 $9/5 * \text{Temperature\_C}$ 공식을 사용하는 Temperature_F 라는 변환 속성이 자산에 있는 경우 timestamp(Temperature_F) 함수를 사용하여 Temperature_F 속성에 대한 최신 GOOD 품질 값의 타임스탬프를 가져올 수 있습니다. • pretrigger() 함수를 함수의 인수로 사용합니다. timestamp(pretrigger(<i>variable-name</i>)) 함수는 Unix epoch 형식으로 현재 변환 계산을 시작한 속성 업데이트 이전의 지정된 변수에 대한 GOOD 품질 값의 타임스탬프(초)를 반환합니다. 자세한 내용은 변환에서 시간 함수 사용 단원을 참조하십시오.

함수	설명
	지표에서 함수는 현재 기간 끝에서 검색된 타임스탬프(초)를 Unix epoch 형식으로 반환합니다.

함수	설명
<code>mktime(time_zone, year, month, day_of_month, hour, minute, second)</code>	입력 시간을 Unix epoch 형식으로 초 단위로 반환합니다. 이 함수 사용에 적용되는 요구 사항은 다음과 같습니다. - 시간대 인수는 따옴표로 묶인 문자열('UTC')이어야 합니다. 지정하지 않은 경우 기본 시간대는 UTC입니다. - 시간대 인수는 첫 번째 또는 마지막 인수일 수 있습니다. - 년, 월, 일, 시, 분, 두 번째 인수는 순서대로 입력해야 합니다. - 년, 월, 일 인수는 필수입니다. 이 함수 사용에는 다음 제한이 적용됩니다. - year - 유효한 값은 1970~2250입니다. - month - 유효한 값은 1~12입니다. - day-of-month - 유효한 값은 1~31입니다. - hour - 유효한 값은 0~23입니다. - minute - 유효한 값은 0~59입니다. - second - 유효한 값은 0~60입니다. 부동 소수점 숫자일 수 있습니다. 예시: <code>mktime(2020, 2, 29)</code> <code>mktime('UTC+3', 2021, 12, 31, 22)</code> <code>mktime(2022, 10, 13, 2, 55, 13.68, 'PST')</code>

함수	설명
<code>localtime(unix_time, time_zone)</code>	Unix 시간을 기준으로 지정된 시간대의 연도, 월의 일, 요일, 연도의 일, 시간, 분 또는 초를 반환합니다. 이 함수 사용에 적용되는 요구 사항은 다음과 같습니다. - 시간대 인수는 따옴표로 묶인 문자열('UTC')이어야 합니다. 지정하지 않은 경우 기본 시간대는 UTC입니다. - Unix 시간 인수는 Unix epoch 형식의 시간(초)입니다. 유효한 범위는 1~31556889864403199입니다. 부동 소수점 숫자일 수 있습니다. 응답의 예: 2007-12-03T10:15:30+01:00[Europe/Paris] <code>localtime(unix_time, time_zone)</code> 은 독립형 함수가 아닙니다. <code>year()</code>, <code>mon()</code>, <code>mday</code>, <code>wday()</code>, <code>yday()</code>, <code>hour()</code>, <code>minute()</code>, <code>sec()</code> 함수는 <code>localtime(unix_time, time_zone)</code> 를 인수로 사용합니다. 예시: <code>year(localtime('GMT', 1605898608.8113723))</code> <code>now().localtime().year()</code> <code>timestamp().localtime('PST').year()</code> <code>localtime(1605289736, 'Europe/London').year()</code>
<code>year(localtime(unix_time, time_zone))</code>	<code>localtime(unix_time, time_zone)</code> 에서 연도를 반환합니다.

함수	설명
<code>mon(localtime(unix_time, time_zone))</code>	<code>localtime(unix_time, time_zone)</code> 에서 월을 반환합니다.
<code>mday(localtime(unix_time, time_zone))</code>	<code>localtime(unix_time, time_zone)</code> 에서 일을 반환합니다.
<code>wday(localtime(unix_time, time_zone))</code>	<code>localtime(unix_time, time_zone)</code> 에서 요일을 반환합니다.
<code>yday(localtime(unix_time, time_zone))</code>	<code>localtime(unix_time, time_zone)</code> 에서 연도의 일을 반환합니다.
<code>hour(localtime(unix_time, time_zone))</code>	<code>localtime(unix_time, time_zone)</code> 에서 시를 반환합니다.
<code>minute(localtime(unix_time, time_zone))</code>	<code>localtime(unix_time, time_zone)</code> 에서 분을 반환합니다.
<code>sec(localtime(unix_time, time_zone))</code>	<code>localtime(unix_time, time_zone)</code> 에서 초를 반환합니다.

지원되는 시간대 형식

다음과 같은 방법으로 시간대 인수를 지정할 수 있습니다.

- 시간대 오프셋 - UTC 또는 오프셋('+2' 또는 '-5')에 대해 'Z'를 지정합니다.
- 오프셋 ID - 시간대 약어와 오프셋을 결합합니다. 예: 'GMT+2' 및 'UTC-01:00'. 시간대 약어는 세 글자만 포함해야 합니다.
- 리전 기반 ID - 예: 'Etc/GMT+12' 및 'Pacific/Pago_Pago'.

지원되는 시간대 약어

날짜 및 시간 함수는 다음과 같은 세 글자로 된 시간대 약어를 지원합니다.

- EST - -05:00
- HST - -10:00

- MST - -07:00
- ACT - Australia/Darwin
- AET - Australia/Sydney
- AGT - America/Argentina/Buenos_Aires
- ART - Africa/Cairo
- AST - America/Anchorage
- BET - America/Sao_Paulo
- BST - Asia/Dhaka
- CAT - Africa/Harare
- CET - Europe/Paris
- CNT - America/St_Johns
- CST - America/Chicago
- CTT - Asia/Shanghai
- EAT - Africa/Addis_Ababa
- IET - America/Indiana/Indianapolis
- IST - Asia/Kolkata
- JST - Asia/Tokyo
- MIT - Pacific/Apia
- NET - Asia/Yerevan
- NST - Pacific/Auckland
- PLT - Asia/Karachi
- PRT - America/Puerto_Rico
- PST - America/Los_Angeles
- SST - Pacific/Guadalcanal
- VST - Asia/Ho_Chi_Minh

지원되는 리전 기반 ID

날짜 및 시간 함수는 UTC+00:00과의 관계를 기준으로 구성된 다음 리전 기반 ID를 지원합니다.

- Etc/GMT+12(UTC-12:00)

- Pacific/Pago_Pago(UTC-11:00)
- Pacific/Samoa(UTC-11:00)
- Pacific/Niue(UTC-11:00)
- US/Samoa(UTC-11:00)
- Etc/GMT+11(UTC-11:00)
- Pacific/Midway(UTC-11:00)
- Pacific/Honolulu(UTC-10:00)
- Pacific/Rarotonga(UTC-10:00)
- Pacific/Tahiti(UTC-10:00)
- Pacific/Johnston(UTC-10:00)
- US/Hawaii(UTC-10:00)
- SystemV/HST10(UTC-10:00)
- Etc/GMT+10(UTC-10:00)
- Pacific/Marquesas(UTC-09:30)
- Etc/GMT+9(UTC-09:00)
- Pacific/Gambier(UTC-09:00)
- America/Atka(UTC-09:00)
- SystemV/YST9(UTC-09:00)
- America/Adak(UTC-09:00)
- US/Aleutian(UTC-09:00)
- Etc/GMT+8(UTC-08:00)
- US/Alaska(UTC-08:00)
- America/Juneau(UTC-08:00)
- America/Metlakatla(UTC-08:00)
- America/Yakutat(UTC-08:00)
- Pacific/Pitcairn(UTC-08:00)
- America/Sitka(UTC-08:00)
- America/Anchorage(UTC-08:00)
- SystemV/PST8(UTC-08:00)

- America/Nome(UTC-08:00)
- SystemV/YST9YDT(UTC-08:00)
- Canada/Yukon(UTC-07:00)
- US/Pacific-New(UTC-07:00)
- Etc/GMT+7(UTC-07:00)
- US/Arizona(UTC-07:00)
- America/Dawson_Creek(UTC-07:00)
- Canada/Pacific(UTC-07:00)
- PST8PDT(UTC-07:00)
- SystemV/MST7(UTC-07:00)
- America/Dawson(UTC-07:00)
- Mexico/BajaNorte(UTC-07:00)
- America/Tijuana(UTC-07:00)
- America/Creston(UTC-07:00)
- America/Hermosillo(UTC-07:00)
- America/Santa_Isabel(UTC-07:00)
- America/Vancouver(UTC-07:00)
- America/Ensenada(UTC-07:00)
- America/Phoenix(UTC-07:00)
- America/Whitehorse(UTC-07:00)
- America/Fort_Nelson(UTC-07:00)
- SystemV/PST8PDT(UTC-07:00)
- America/Los_Angele (UTC-07:00)
- US/Pacific(UTC-07:00)
- America/El_Salvador(UTC-06:00)
- America/Guatemala(UTC-06:00)
- America/Belize(UTC-06:00)
- America/Managua(UTC-06:00)
- America/Tegucigalpa(UTC-06:00)

- Etc/GMT+6(UTC-06:00)
- Pacific/Easter(UTC-06:00)
- Mexico/BajaSur(UTC-06:00)
- America/Regina(UTC-06:00)
- America/Denver(UTC-06:00)
- Pacific/Galapagos(UTC-06:00)
- America/Yellowknife(UTC-06:00)
- America/Swift_Current(UTC-06:00)
- America/Inuvik(UTC-06:00)
- America/Mazatlan(UTC-06:00)
- America/Boise(UTC-06:00)
- America/Costa_Rica(UTC-06:00)
- MST7MDT(UTC-06:00)
- SystemV/CST6(UTC-06:00)
- America/Chihuahua(UTC-06:00)
- America/Ojinaga(UTC-06:00)
- Chile/EasterIsland(UTC-06:00)
- US/Mountain(UTC-06:00)
- America/Edmonton(UTC-06:00)
- Canada/Mountain(UTC-06:00)
- America/Cambridge_Bay(UTC-06:00)
- Navajo(UTC-06:00)
- SystemV/MST7MDT(UTC-06:00)
- Canada/Saskatchewan(UTC-06:00)
- America/Shiprock(UTC-06:00)
- America/Panama(UTC-05:00)
- America/Chicago(UTC-05:00)
- America/Eirunepe(UTC-05:00)
- Etc/GMT+5(UTC-05:00)

- Mexico/General(UTC-05:00)
- America/Porto_Acre(UTC-05:00)
- America/Guayaquil(UTC-05:00)
- America/Rankin_Inlet(UTC-05:00)
- US/Central(UTC-05:00)
- America/Rainy_River(UTC-05:00)
- America/Indiana/Knox(UTC-05:00)
- America/North_Dakota/Beulah(UTC-05:00)
- America/Monterrey(UTC-05:00)
- America/Jamaica(UTC-05:00)
- America/Atikokan(UTC-05:00)
- America/Coral_Harbour(UTC-05:00)
- America/North_Dakota/Center(UTC-05:00)
- America/Cayman(UTC-05:00)
- America/Indiana/Tell_City(UTC-05:00)
- America/Mexico_City(UTC-05:00)
- America/Matamoros(UTC-05:00)
- CST6CDT(UTC-05:00)
- America/Knox_IN(UTC-05:00)
- America/Bogota(UTC-05:00)
- America/Menominee(UTC-05:00)
- America/Resolute(UTC-05:00)
- SystemV/EST5(UTC-05:00)
- Canada/Central(UTC-05:00)
- Brazil/Acre(UTC-05:00)
- America/Cancun(UTC-05:00)
- America/Lima(UTC-05:00)
- America/Bahia_Banderas(UTC-05:00)
- US/Indiana-Starke(UTC-05:00)
- America/Rio_Branco(UTC-05:00)

- SystemV/CST6CDT(UTC-05:00)
- Jamaica(UTC-05:00)
- America/Merida(UTC-05:00)
- America/North_Dakota/New_Salem(UTC-05:00)
- America/Winnipeg(UTC-05:00)
- America/Cuiaba(UTC-04:00)
- America/Marigot(UTC-04:00)
- America/Indiana/Petersburg(UTC-04:00)
- Chile/Continental(UTC-04:00)
- America/Grand_Turk(UTC-04:00)
- Cuba(UTC-04:00)
- Etc/GMT+4(UTC-04:00)
- America/Manaus(UTC-04:00)
- America/Fort_Wayne(UTC-04:00)
- America/St_Thomas(UTC-04:00)
- America/Anguilla(UTC-04:00)
- America/Havana(UTC-04:00)
- US/Michigan(UTC-04:00)
- America/Barbados(UTC-04:00)
- America/Louisville(UTC-04:00)
- America/Curacao(UTC-04:00)
- America/Guyana(UTC-04:00)
- America/Martinique(UTC-04:00)
- America/Puerto_Rico(UTC-04:00)
- America/Port_of_Spain(UTC-04:00)
- SystemV/AST4(UTC-04:00)
- America/Indiana/Vevay(UTC-04:00)
- America/Indiana/Vincennes(UTC-04:00)
- America/Kralendijk(UTC-04:00)
- America/Antigua(UTC-04:00)

- America/Indianapolis(UTC-04:00)
- America/Iqaluit(UTC-04:00)
- America/St_Vincent(UTC-04:00)
- America/Kentucky/Louisville(UTC-04:00)
- America/Dominica(UTC-04:00)
- America/Asuncion(UTC-04:00)
- EST5EDT(UTC-04:00)
- America/Nassa(UTC-04:00)
- America/Kentucky/Monticello(UTC-04:00)
- Brazil/West(UTC-04:00)
- America/Aruba(UTC-04:00)
- America/Indiana/Indianapolis(UTC-04:00)
- America/Santiago(UTC-04:00)
- America/La_Paz(UTC-04:00)
- America/Thunder_Bay(UTC-04:00)
- America/Indiana/Marengo(UTC-04:00)
- America/Blanc-Sablon(UTC-04:00)
- America/Santo_Domingo(UTC-04:00)
- US/Eastern(UTC-04:00)
- Canada/Eastern(UTC-04:00)
- America/Port-au-Prince(UTC-04:00)
- America/St_Barthelemy(UTC-04:00)
- America/Nipigon(UTC-04:00)
- US/East-Indiana(UTC-04:00)
- America/St_Lucia(UTC-04:00)
- America/Montserrat(UTC-04:00)
- America/Lower_Princes(UTC-04:00)
- America/Detroit(UTC-04:00)
- America/Tortola(UTC-04:00)
- America/Porto_Velho(UTC-04:00)

- America/Campo_Grande(UTC-04:00)
- America/Virgin(UTC-04:00)
- America/Pangnirtung(UTC-04:00)
- America/Montreal(UTC-04:00)
- America/Indiana/Winamac(UTC-04:00)
- America/Boa_Vista(UTC-04:00)
- America/Grenada(UTC-04:00)
- America/New_York(UTC-04:00)
- America/St_Kitts(UTC-04:00)
- America/Caracas(UTC-04:00)
- America/Guadeloupe(UTC-04:00)
- America/Toronto(UTC-04:00)
- SystemV/EST5EDT(UTC-04:00)
- America/Argentina/Catamarca(UTC-03:00)
- Canada/Atlantic(UTC-03:00)
- America/Argentina/Cordoba(UTC-03:00)
- America/Araguaina(UTC-03:00)
- America/Argentina/Salta(UTC-03:00)
- Etc/GMT+3(UTC-03:00)
- America/Montevideo(UTC-03:00)
- Brazil/Eas(UTC-03:00)
- America/Argentina/Mendoza(UTC-03:00)
- America/Argentina/Rio_Gallegos(UTC-03:00)
- America/Catamarca(UTC-03:00)
- America/Cordoba(UTC-03:00)
- America/Sao_Paulo(UTC-03:00)
- America/Argentina/Jujuy(UTC-03:00)
- America/Cayenne(UTC-03:00)
- America/Recife(UTC-03:00)
- America/Buenos_Aires(UTC-03:00)

- America/Paramaribo(UTC-03:00)
- America/Moncton(UTC-03:00)
- America/Mendoza(UTC-03:00)
- America/Santarem(UTC-03:00)
- Atlantic/Bermuda(UTC-03:00)
- America/Maceio(UTC-03:00)
- Atlantic/Stanley(UTC-03:00)
- America/Halifax(UTC-03:00)
- Antarctica/Rothera(UTC-03:00)
- America/Argentina/San_Luis(UTC-03:00)
- America/Argentina/Ushuaia(UTC-03:00)
- Antarctica/Palmer(UTC-03:00)
- America/Punta_Arenas(UTC-03:00)
- America/Glace_Bay(UTC-03:00)
- America/Fortaleza(UTC-03:00)
- America/Thule(UTC-03:00)
- America/Argentina/La_Rioja(UTC-03:00)
- America/Belem(UTC-03:00)
- America/Jujuy(UTC-03:00)
- America/Bahia(UTC-03:00)
- America/Goose_Bay(UTC-03:00)
- America/Argentina/San_Juan(UTC-03:00)
- America/Argentina/ComodRivadavia(UTC-03:00)
- America/Argentina/Tucuman(UTC-03:00)
- America/Rosario(UTC-03:00)
- SystemV/AST4ADT(UTC-03:00)
- America/Argentina/Buenos_Aires(UTC-03:00)
- America/St_Johns(UTC-02:30)
- Canada/Newfoundland(UTC-02:30)
- America/Miquelon(UTC-02:00)

- Etc/GMT+2(UTC-02:00)
- America/Godthab(UTC-02:00)
- America/Noronha(UTC-02:00)
- Brazil/DeNoronha(UTC-02:00)
- Atlantic/South_Georgia(UTC-02:00)
- Etc/GMT+1(UTC-01:00)
- Atlantic/Cape_Verde(UTC-01:00)
- Pacific/Kiritimati(UTC+14:00)
- Etc/GMT-14(UTC+14:00)
- Pacific/Fakaofu(UTC+13:00)
- Pacific/Enderbury(UTC+13:00)
- Pacific/Apia(UTC+13:00)
- Pacific/Tongatapu(UTC+13:00)
- Etc/GMT-13(UTC+13:00)
- NZ-CHAT(UTC+12:45)
- Pacific/Chatham(UTC+12:45)
- Pacific/Kwajalein(UTC+12:00)
- Antarctica/McMurdo(UTC+12:00)
- Pacific/Wallis(UTC+12:00)
- Pacific/Fiji(UTC+12:00)
- Pacific/Funafuti(UTC+12:00)
- Pacific/Nauru(UTC+12:00)
- Kwajalein(UTC+12:00)
- NZ(UTC+12:00)
- Pacific/Wake(UTC+12:00)
- Antarctica/South_Pole(UTC+12:00)
- Pacific/Tarawa(UTC+12:00)
- Pacific/Auckland(UTC+12:00)
- Asia/Kamchatka(UTC+12:00)
- Etc/GMT-12(UTC+12:00)

- Asia/Anadyr(UTC+12:00)
- Pacific/Majuro(UTC+12:00)
- Pacific/Ponape(UTC+11:00)
- Pacific/Bougainville(UTC+11:00)
- Antarctica/Macquarie(UTC+11:00)
- Pacific/Pohnpei(UTC+11:00)
- Pacific/Efate(UTC+11:00)
- Pacific/Norfolk(UTC+11:00)
- Asia/Magadan(UTC+11:00)
- Pacific/Kosrae(UTC+11:00)
- Asia/Sakhalin(UTC+11:00)
- Pacific/Noumea(UTC+11:00)
- Etc/GMT-11(UTC+11:00)
- Asia/Srednekolymsk(UTC+11:00)
- Pacific/Guadalcanal(UTC+11:00)
- Australia/Lord_Howe(UTC+10:30)
- Australia/LHI(UTC+10:30)
- Australia/Hobart(UTC+10:00)
- Pacific/Yap(UTC+10:00)
- Australia/Tasmania(UTC+10:00)
- Pacific/Port_Moresby(UTC+10:00)
- Australia/ACT(UTC+10:00)
- Australia/Victoria(UTC+10:00)
- Pacific/Chuuk(UTC+10:00)
- Australia/Queensland(UTC+10:00)
- Australia/Canberra(UTC+10:00)
- Australia/Currie(UTC+10:00)
- Pacific/Guam(UTC+10:00)
- Pacific/Truk(UTC+10:00)
- Australia/NSW(UTC+10:00)

- Asia/Vladivostok(UTC+10:00)
- Pacific/Saipan(UTC+10:00)
- Antarctica/DumontD'Urville(UTC+10:00)
- Australia/Sydney(UTC+10:00)
- Australia/Brisbane(UTC+10:00)
- Etc/GMT-10(UTC+10:00)
- Asia/Ust-Nera(UTC+10:00)
- Australia/Melbourne(UTC+10:00)
- Australia/Lindeman(UTC+10:00)
- Australia/North(UTC+09:30)
- Australia/Yancowinna(UTC+09:30)
- Australia/Adelaide(UTC+09:30)
- Australia/Broken_Hill(UTC+09:30)
- Australia/South(UTC+09:30)
- Australia/Darwin(UTC+09:30)
- Etc/GMT-9(UTC+09:00)
- Pacific/Palau(UTC+09:00)
- Asia/Chita(UTC+09:00)
- Asia/Dili(UTC+09:00)
- Asia/Jayapura(UTC+09:00)
- Asia/Yakutsk(UTC+09:00)
- Asia/Pyongyang(UTC+09:00)
- ROK(UTC+09:00)
- Asia/Seoul(UTC+09:00)
- Asia/Khandyga(UTC+09:00)
- Japan(UTC+09:00)
- Asia/Tokyo(UTC+09:00)
- Australia/Eucla(UTC+08:45)
- Asia/Kuching(UTC+08:00)
- Asia/Chungking(UTC+08:00)

- Etc/GMT-8(UTC+08:00)
- Australia/Perth(UTC+08:00)
- Asia/Macao(UTC+08:00)
- Asia/Maca(UTC+08:00)
- Asia/Choibalsan(UTC+08:00)
- Asia/Shanghai(UTC+08:00)
- Antarctica/Casey(UTC+08:00)
- Asia/Ulan_Bator(UTC+08:00)
- Asia/Chongqing(UTC+08:00)
- Asia/Ulaanbaatar(UTC+08:00)
- Asia/Taipei(UTC+08:00)
- Asia/Manila(UTC+08:00)
- PRC(UTC+08:00)
- Asia/Ujung_Pandang(UTC+08:00)
- Asia/Harbin(UTC+08:00)
- Singapore(UTC+08:00)
- Asia/Brunei(UTC+08:00)
- Australia/West(UTC+08:00)
- Asia/Hong_Kong(UTC+08:00)
- Asia/Makassar(UTC+08:00)
- Hongkong(UTC+08:00)
- Asia/Kuala_Lumpur(UTC+08:00)
- Asia/Irkutsk(UTC+08:00)
- Asia/Singapore(UTC+08:00)
- Asia/Pontianak(UTC+07:00)
- Etc/GMT-7(UTC+07:00)
- Asia/Phnom_Penh(UTC+07:00)
- Asia/Novosibirsk(UTC+07:00)
- Antarctica/Davis(UTC+07:00)
- Asia/Tomsk(UTC+07:00)

- Asia/Jakarta(UTC+07:00)
- Asia/Barnaul(UTC+07:00)
- Indian/Christmas(UTC+07:00)
- Asia/Ho_Chi_Minh(UTC+07:00)
- Asia/Hovd(UTC+07:00)
- Asia/Bangkok(UTC+07:00)
- Asia/Vientiane(UTC+07:00)
- Asia/Novokuznetsk(UTC+07:00)
- Asia/Krasnoyarsk(UTC+07:00)
- Asia/Saigon(UTC+07:00)
- Asia/Yangon(UTC+06:30)
- Asia/Rangoon(UTC+06:30)
- Indian/Cocos(UTC+06:30)
- Asia/Kashgar(UTC+06:00)
- Etc/GMT-6(UTC+06:00)
- Asia/Almaty(UTC+06:00)
- Asia/Dacca(UTC+06:00)
- Asia/Omsk(UTC+06:00)
- Asia/Dhaka(UTC+06:00)
- Indian/Chagos(UTC+06:00)
- Asia/Qyzylorda(UTC+06:00)
- Asia/Bishkek(UTC+06:00)
- Antarctica/Vostok(UTC+06:00)
- Asia/Urumqi(UTC+06:00)
- Asia/Thimbu(UTC+06:00)
- Asia/Thimphu(UTC+06:00)
- Asia/Kathmandu(UTC+05:45)
- Asia/Katmandu(UTC+05:45)
- Asia/Kolkata(UTC+05:30)
- Asia/Colombo(UTC+05:30)

- Asia/Calcutta(UTC+05:30)
- Asia/Aqtau(UTC+05:00)
- Etc/GMT-5(UTC+05:00)
- Asia/Samarkand(UTC+05:00)
- Asia/Karachi(UTC+05:00)
- Asia/Yekaterinburg(UTC+05:00)
- Asia/Dushanbe(UTC+05:00)
- Indian/Maldives(UTC+05:00)
- Asia/Oral(UTC+05:00)
- Asia/Tashkent(UTC+05:00)
- Antarctica/Mawson(UTC+05:00)
- Asia/Aqtobe(UTC+05:00)
- Asia/Ashkhabad(UTC+05:00)
- Asia/Ashgabat(UTC+05:00)
- Asia/Atyrau(UTC+05:00)
- Indian/Kerguelen(UTC+05:00)
- Iran(UTC+04:30)
- Asia/Tehran(UTC+04:30)
- Asia/Kabul(UTC+04:30)
- Asia/Yerevan(UTC+04:00)
- Etc/GMT-4(UTC+04:00)
- Etc/GMT-4(UTC+04:00)
- Asia/Dubai(UTC+04:00)
- Indian/Reunion(UTC+04:00)
- Europe/Saratov(UTC+04:00)
- Europe/Samara(UTC+04:00)
- Indian/Mahe(UTC+04:00)
- Asia/Baku(UTC+04:00)
- Asia/Muscat(UTC+04:00)
- Europe/Volgograd(UTC+04:00)

- Europe/Astrakhan(UTC+04:00)
- Asia/Tbilisi(UTC+04:00)
- Europe/Ulyanovsk(UTC+04:00)
- Asia/Aden(UTC+03:00)
- Africa/Nairobi(UTC+03:00)
- Europe/Istanbul(UTC+03:00)
- Etc/GMT-3(UTC+03:00)
- Europe/Zaporozhye(UTC+03:00)
- Israel(UTC+03:00)
- Indian/Comoro(UTC+03:00)
- Antarctica/Syowa(UTC+03:00)
- Africa/Mogadishu(UTC+03:00)
- Europe/Bucharest(UTC+03:00)
- Africa/Asmera(UTC+03:00)
- Europe/Mariehamn(UTC+03:00)
- Asia/Istanbul(UTC+03:00)
- Europe/Tiraspol(UTC+03:00)
- Europe/Moscow(UTC+03:00)
- Europe/Chisinau(UTC+03:00)
- Europe/Helsinki(UTC+03:00)
- Asia/Beirut(UTC+03:00)
- Asia/Tel_Aviv(UTC+03:00)
- Africa/Djibouti(UTC+03:00)
- Europe/Simferopol(UTC+03:00)
- Europe/Sofia(UTC+03:00)
- Asia/Gaza(UTC+03:00)
- Africa/Asmara(UTC+03:00)
- Europe/Riga(UTC+03:00)
- Asia/Baghdad(UTC+03:00)
- Asia/Damascus(UTC+03:00)

- Africa/Dar_es_Salaam(UTC+03:00)
- Africa/Addis_Ababa(UTC+03:00)
- Europe/Uzhgorod(UTC+03:00)
- Asia/Jerusalem(UTC+03:00)
- Asia/Riyadh(UTC+03:00)
- Asia/Kuwait(UTC+03:00)
- Europe/Kirov(UTC+03:00)
- Africa/Kampala(UTC+03:00)
- Europe/Minsk(UTC+03:00)
- Asia/Qatar UTC+03:00)
- Europe/Kiev(UTC+03:00)
- Asia/Bahrain(UTC+03:00)
- Europe/Vilnius(UTC+03:00)
- Indian/Antananarivo(UTC+03:00)
- Indian/Mayotte(UTC+03:00)
- Europe/Tallinn(UTC+03:00)
- Turkey(UTC+03:00)
- Africa/Juba(UTC+03:00)
- Asia/Nicosia(UTC+03:00)
- Asia/Famagusta(UTC+03:00)
- W-SU(UTC+03:00)
- EET(UTC+03:00)
- Asia/Hebron(UTC+03:00)
- Asia/Amman(UTC+03:00)
- Europe/Nicosia(UTC+03:00)
- Europe/Athens(UTC+03:00)
- Africa/Cairo(UTC+02:00)
- Africa/Mbabane(UTC+02:00)
- Europe/Brussels(UTC+02:00)
- Europe/Warsaw(UTC+02:00)

- CET(UTC+02:00)
- Europe/Luxembourg(UTC+02:00)
- Etc/GMT-2(UTC+02:00)
- Libya(UTC+02:00)
- Africa/Kigali(UTC+02:00)
- Africa/Tripoli(UTC+02:00)
- Europe/Kaliningrad(UTC+02:00)
- Africa/Windhoek(UTC+02:00)
- Europe/Malta(UTC+02:00)
- Europe/Busingen(UTC+02:00)
-
- Europe/Skopje(UTC+02:00)
- Europe/Sarajevo(UTC+02:00)
- Europe/Rome(UTC+02:00)
- Europe/Zurich(UTC+02:00)
- Europe/Gibraltar(UTC+02:00)
- Africa/Lubumbashi(UTC+02:00)
- Europe/Vaduz(UTC+02:00)
- Europe/Ljubljana(UTC+02:00)
- Europe/Berlin(UTC+02:00)
- Europe/Stockholm(UTC+02:00)
- Europe/Budapest(UTC+02:00)
- Europe/Zagreb(UTC+02:00)
- Europe/Paris(UTC+02:00)
- Africa/Ceuta(UTC+02:00)
- Europe/Prague(UTC+02:00)
- Antarctica/Troll(UTC+02:00)
- Africa/Gaborone(UTC+02:00)
- Europe/Copenhagen(UTC+02:00)
- Europe/Vienna(UTC+02:00)

- Europe/Tirane(UTC+02:00)
- MET(UTC+02:00)
- Europe/Amsterdam(UTC+02:00)
- Africa/Maputo(UTC+02:00)
- Europe/San_Marino(UTC+02:00)
- Poland(UTC+02:00)
- Europe/Andorra(UTC+02:00)
- Europe/Oslo(UTC+02:00)
- Europe/Podgorica(UTC+02:00)
- Africa/Bujumbura(UTC+02:00)
- Atlantic/Jan_Mayen(UTC+02:00)
- Africa/Maseru(UTC+02:00)
- Europe/Madrid(UTC+02:00)
- Africa/Blantyre(UTC+02:00)
- Africa/Lusaka(UTC+02:00)
- Africa/Harare(UTC+02:00)
- Africa/Khartoum(UTC+02:00)
- Africa/Johannesburg(UTC+02:00)
- Europe/Belgrade(UTC+02:00)
- Europe/Bratislava(UTC+02:00)
- Arctic/Longyearbyen(UTC+02:00)
- Egypt(UTC+02:00)
- Europe/Vatican(UTC+02:00)
- Europe/Monaco(UTC+02:00)
- Europe/London(UTC+01:00)
- Etc/GMT-1(UTC+01:00)
- Europe/Jersey(UTC+01:00)
- Europe/Guernsey(UTC+01:00)
- Europe/Isle_of_Man(UTC+01:00)
- Africa/Tunis(UTC+01:00)

- Africa/Malabo(UTC+01:00)
- GB-Eire(UTC+01:00)
- Africa/Lagos(UTC+01:00)
- Africa/Algiers(UTC+01:00)
- GB(UTC+01:00)
- Portugal(UTC+01:00)
- Africa/Sao_Tome(UTC+01:00)
- Africa/Ndjamena(UTC+01:00)
- Atlantic/Faeroe(UTC+01:00)
- Eire(UTC+01:00)
- Atlantic/Faroe(UTC+01:00)
- Europe/Dublin(UTC+01:00)
- Africa/Libreville(UTC+01:00)
- Africa/El_Aaiun(UTC+01:00)
- Africa/El_Aaiun(UTC+01:00)
- Africa/Douala(UTC+01:00)
- Africa/Brazzaville(UTC+01:00)
- Africa/Porto-Novo(UTC+01:00)
- Atlantic/Madeira(UTC+01:00)
- Europe/Lisbon(UTC+01:00)
- Atlantic/Canary(UTC+01:00)
- Africa/Casablanca(UTC+01:00)
- Europe/Belfast(UTC+01:00)
- Africa/Luanda(UTC+01:00)
- Africa/Kinshasa(UTC+01:00)
- Africa/Bangui(UTC+01:00)
- WET(UTC+01:00)
- Africa/Niamey(UTC+01:00)
- GMT(UTC+00:00)
- Etc/GMT-0(UTC+00:00)

- Atlantic/St_Helena(UTC+00:00)
- Etc/GMT+0(UTC+00:00)
- Africa/Banjul(UTC+00:00)
- Etc/GMT(UTC+00:00)
- Africa/Freetown(UTC+00:00)
- Africa/Bamako(UTC+00:00)
- Africa/Conakry(UTC+00:00)
- Universal(UTC+00:00)
- Africa/Nouakchott(UTC+00:00)
- UTC(UTC+00:00)
- Etc/Universal(UTC+00:00)
- Atlantic/Azores(UTC+00:00)
- Africa/Abidjan(UTC+00:00)
- Africa/Accra(UTC+00:00)
- Etc/UCT(UTC+00:00)
- GMT0(UTC+00:00)
- Zulu(UTC+00:00)Zulu (UTC+00:00)
- Africa/Ouagadougou(UTC+00:00)
- Atlantic/Reykjavik(UTC+00:00)
- Etc/Zulu(UTC+00:00)
- Iceland(UTC+00:00)
- Africa/Lome(UTC+00:00)
- Greenwich(UTC+00:00)
- Etc/GMT0(UTC+00:00)
- America/Danmarkshavn UTC+00:00)
- Africa/Dakar(UTC+00:00)
- Africa/Bissau(UTC+00:00)
- Etc/Greenwich(UTC+00:00)
- Africa/Timbuktu(UTC+00:00)
- UCT(UTC+00:00)

- Africa/Monrovia(UTC+00:00)
- Etc/UTC(UTC+00:00)

공식 표현식 튜토리얼

이 튜토리얼을 따라 AWS IoT SiteWise에서 공식 표현식을 사용할 수 있습니다.

주제

- [수식에서 문자열 사용](#)
- [데이터 포인트 필터링](#)
- [조건과 일치하는 데이터 포인트를 계산하려면 다음을 수행하세요.](#)
- [공식의 지연 데이터](#)
- [공식의 데이터 품질](#)
- [정의되지 않은 값, 무한 값 및 오버플로우 값](#)

수식에서 문자열 사용

공식 표현식의 문자열에 대해 작업을 할 수 있습니다. 속성 및 측정 속성을 참조하는 변수에서 문자열을 입력할 수도 있습니다.

Important

공식 표현식은 실수 또는 문자열 값만 출력할 수 있습니다. 중첩 표현식은 문자열과 같은 다른 데이터 유형을 출력할 수 있지만 공식 전체는 숫자나 문자열로 평가되어야 합니다. [jp 함수](#)를 사용하여 문자열을 숫자로 변환할 수 있습니다. 부울 값은 1(참) 또는 0(거짓)이어야 합니다. 자세한 내용은 [정의되지 않은 값, 무한 값 및 오버플로우 값](#) 단원을 참조하십시오.

AWS IoT SiteWise 는 문자열에서 작동하는 데 사용할 수 있는 다음과 같은 수식 표현식 기능을 제공합니다.

- [문자열 리터럴](#)
- [인덱스 연산자](#)(s[index])
- [조각 연산자](#)(s[start:end:step])
- [비교 함수](#)는 문자열을 [사전순으로](#) 비교하는 데 사용할 수 있습니다.

- [문자열 함수](#)는 직렬화된 JSON 객체를 파싱하고 문자열을 숫자로 변환할 수 있는 `jp` 함수를 포함합니다.

데이터 포인트 필터링

[if 함수](#)를 사용하여 조건을 충족하지 않는 데이터 요소를 필터링할 수 있습니다. `if` 함수는 조건을 평가하여 `true` 및 `false` 결과에 대해 서로 다른 값을 반환합니다. [none 상수](#)를 `if` 함수의 한 케이스에 대한 출력값으로 사용하여 해당 케이스의 데이터 포인트를 버릴 수 있습니다.

조건과 일치하는 데이터 포인트를 필터링하려면 다음을 수행하세요.

- `if` 함수를 사용하여 조건이 충족되는지 확인하는 조건을 정의하고 `result_if_true` 또는 `result_if_false` 값으로 `none`를 반환하는 변환을 만드세요.

Example 예: 물이 끓지 않는 데이터 포인트 필터링

머신에 있는 물의 온도(섭씨)를 제공하는 측정값(`temp_c`)이 있는 시나리오를 생각해 보세요. 다음 변환을 정의하여 물이 끓지 않는 데이터 포인트를 필터링할 수 있습니다.

- 변환: `boiling_temps = if(gte(temp_c, 100), temp_c, none)` - 섭씨 100도 이상이면 온도를 반환하고, 그렇지 않으면 데이터 포인트를 반환하지 않습니다.

조건과 일치하는 데이터 포인트를 계산하려면 다음을 수행하세요.

[비교 함수](#)와 [sum\(\)](#)을 사용하여 조건이 참인 데이터 포인트 수를 계산할 수 있습니다.

조건과 일치하는 데이터 포인트를 계산하려면 다음을 수행하세요.

1. 비교 함수를 사용하여 다른 속성에 대한 필터 조건을 정의하는 변환을 생성합니다.
2. 해당 조건이 충족되는 데이터 포인트를 합산하는 지표를 생성합니다.

Example 예: 물이 끓는 데이터 포인트 수 계산

머신에 있는 물의 온도(섭씨)를 제공하는 측정값(`temp_c`)이 있는 시나리오를 생각해 보세요. 다음과 같은 변환 및 지표 속성을 정의하여 물이 끓는 데이터 포인트 수를 계산할 수 있습니다.

- 변환: `is_boiling = gte(temp_c, 100)` - 온도가 섭씨 100도보다 높거나 같으면 1을 반환하고, 그렇지 않으면 0을 반환합니다.
- 지표: `boiling_count = sum(is_boiling)` - 물이 끓는 데이터 포인트 수를 계산합니다.

공식의 지연 데이터

AWS IoT SiteWise 는 최대 7일 이전 데이터의 늦은 데이터 수집을 지원합니다. 가 지연 데이터를 AWS IoT SiteWise 수신하면 과거 기간에 지연 데이터를 입력하는 지표의 기존 값을 다시 계산합니다. 이러한 재계산으로 인해 데이터 처리 요금이 발생합니다.

Note

가 지연 데이터를 입력하는 속성을 AWS IoT SiteWise 계산할 때 각 속성의 현재 수식 표현식을 사용합니다.

는 지표에 대한 과거 기간을 AWS IoT SiteWise 다시 계산한 후 해당 기간의 이전 값을 대체합니다. 해당 지표에 대한 알림을 활성화한 경우는 속성 값 알림 AWS IoT SiteWise 도 내보냅니다. 즉, 이전에 알림을 받은 동일한 속성 및 타임스탬프에 대해 새 속성 값 업데이트 알림을 받을 수 있습니다. 애플리케이션이나 데이터 레이크가 속성 값 알림을 사용하는 경우, 데이터가 정확하도록 이전 값을 새 값으로 업데이트해야 합니다.

공식의 데이터 품질

에서 AWS IoT SiteWise 각 데이터 포인트에는 다음 중 하나일 수 있는 품질 코드가 있습니다.

- GOOD – 데이터가 어떤 문제의 영향도 받지 않습니다.
- BAD – 데이터가 센서 고장과 같은 문제의 영향을 받습니다.
- UNCERTAIN – 데이터가 센서 부정확과 같은 문제의 영향을 받습니다.

AWS IoT SiteWise 는 변환 및 지표를 계산할 때 GOOD 품질 데이터만 사용합니다. 성공적인 계산을 위해 GOOD 품질 데이터만 AWS IoT SiteWise 출력합니다. 계산에 실패하면는 해당 계산에 대한 데이터 포인트를 출력 AWS IoT SiteWise 하지 않습니다. 이 문제는 계산의 결과로 정의되지 않은 값, 무한 값 또는 오버플로우 값이 나오는 경우에 발생할 수 있습니다.

데이터를 쿼리하고 데이터 품질별로 필터링하는 방법에 대한 자세한 내용은 [에서 데이터 쿼리 AWS IoT SiteWise](#) 단원을 참조하세요.

정의되지 않은 값, 무한 값 및 오버플로우 값

일부 수식 표현식(예: $x / 0$, $\sqrt{-1}$ 또는 $\log(0)$)은 실수 시스템에서 정의되지 않은 값, 무한 또는에서 지원하는 범위를 벗어나는 값을 계산합니다 AWS IoT SiteWise. 자산 속성의 표현식이 정의되

지 않은 값, 무한 값 또는 오버플로 값을 계산하는 경우 AWS IoT SiteWise 는 해당 계산에 대한 데이터 포인트를 출력하지 않습니다.

AWS IoT SiteWise 또한 수식 표현식의 결과로 숫자가 아닌 값을 계산하는 경우 데이터 포인트를 출력하지 않습니다. 즉, 문자열, 배열 또는 [none 상수](#)를 계산하는 공식을 정의하면 해당 계산에 대한 데이터 포인트가 AWS IoT SiteWise 에서 출력되지 않습니다.

Example 예시

다음 각 수식 표현식은 숫자로 표현할 AWS IoT SiteWise 수 없는 값을 생성합니다. AWS IoT SiteWise 는 이러한 수식 표현식을 계산할 때 데이터 포인트를 출력하지 않습니다.

- $x / 0$ 은 정의되지 않았습니다.
- $\log(0)$ 은 정의되지 않았습니다.
- $\text{sqrt}(-1)$ 는 실수 시스템에서는 정의되지 않습니다.
- "hello" + " world"는 문자열입니다.
- `jp({'values':[3,6,7]}, '$.values')`는 배열입니다.
- temp가 300보다 작을 때 `if(gte(temp, 300), temp, none)`는 none입니다.

사용자 지정 복합 모델(구성 요소) 생성

사용자 지정 복합 모델 또는 구성 요소(콘솔을 사용하는 경우)는 자산 모델 및 구성 요소 모델에 대한 다른 수준의 구성을 제공합니다. 속성을 그룹화하거나 다른 모델을 참조하여 모델을 구조화하는 데 사용할 수 있습니다. 사용자 지정 복합 모델 작업에 대한 자세한 내용은 [사용자 지정 복합 모델\(구성 요소\)](#) 섹션을 참조하세요.

기존 자산 모델 또는 구성 요소 모델 내에서 사용자 지정 복합 모델을 생성합니다. 사용자 지정 복합 모델에는 두 가지 유형이 있습니다. 모델 내에서 관련 속성을 그룹화하려면 인라인 사용자 지정 복합 모델을 생성할 수 있습니다. 자산 모델 또는 구성 요소 모델 내에서 구성 요소 모델을 참조하려면 구성 요소 모델 기반 사용자 지정 복합 모델을 생성할 수 있습니다.

다음 섹션에서는 AWS IoT SiteWise API를 사용하여 사용자 지정 복합 모델을 생성하는 방법을 설명합니다.

주제

- [인라인 구성 요소 생성\(콘솔\)](#)
- [인라인 사용자 지정 복합 모델 생성\(AWS CLI\)](#)
- [구성 요소 모델 기반 구성 요소 생성\(콘솔\)](#)

- [구성 요소 모델 기반 사용자 지정 복합 모델 생성\(AWS CLI\)](#)

인라인 구성 요소 생성(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자체 속성을 정의하는 인라인 구성 요소를 생성할 수 있습니다.

Note

이는 인라인 구성 요소이므로 이러한 속성은 현재 자산 모델에만 적용되며 다른 곳에 공유되지 않습니다.

재사용 가능한 모델을 생성해야 하는 경우(예: 여러 자산 모델 간에 공유하거나 한 자산 모델 내에 여러 인스턴스를 포함) 대신 구성 요소 모델을 기반으로 구성 요소를 생성해야 합니다. 자세한 내용은 다음 섹션을 참조하시기 바랍니다.

구성 요소 생성(콘솔)

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 구성 요소를 추가하고자 하는 자산 모델을 선택합니다.
4. 속성 탭에서 구성 요소를 선택합니다.
5. 구성 요소 생성을 선택합니다.
6. 구성 요소 생성 페이지에서 다음을 수행합니다.
 - a. **ServoMotor** 또는 **ServoMotor Model**과 같은 구성 요소의 이름을 입력합니다. 이 이름은 이 리전의 계정에 있는 모든 구성 요소에서 고유해야 합니다.
 - b. (선택 사항) 모델의 속성 정의를 추가합니다. 속성은 거의 변경되지 않는 정보를 나타냅니다. 자세한 내용은 [정적 데이터\(속성\) 정의](#) 단원을 참조하십시오.
 - c. (선택 사항) 모델의 측정 정의를 추가합니다. 측정값은 장비의 데이터 스트림을 나타냅니다. 자세한 내용은 [장비의 데이터 스트림 정의\(측정값\)](#) 단원을 참조하십시오.
 - d. (선택 사항) 모델의 변환 정의를 추가합니다. 변환은 데이터를 한 양식에서 다른 양식으로 매핑하는 공식입니다. 자세한 내용은 [데이터 변환\(변환\)](#) 단원을 참조하십시오.
 - e. (선택 사항) 모델의 지표 정의를 추가합니다. 지표는 일정 간격으로 데이터를 집계하는 공식입니다. 지표는 연결된 자산에서 데이터를 입력할 수 있으므로 작업 또는 작업의 하위 집합을 나타내는 값을 계산할 수 있습니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하십시오.

- f. 구성 요소 생성을 선택합니다.

인라인 사용자 지정 복합 모델 생성(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자체 속성을 정의하는 인라인 사용자 지정 복합 모델을 생성할 수 있습니다.

[CreateAssetModelCompositeModel](#) 작업을 사용하여 속성이 포함된 인라인 모델을 생성합니다. 이 작업에는 다음과 같은 구조의 페이로드가 필요합니다.

Note

이는 인라인 복합 모델이므로 이러한 속성은 현재 자산 모델에만 적용되며 다른 곳에 공유되지 않습니다. '인라인'인 이유는 `composedAssetModelId` 필드에 값을 제공하지 않기 때문입니다.

재사용 가능한 모델을 생성해야 하는 경우(예: 여러 자산 모델 간에 공유하거나 한 자산 모델 내에 여러 인스턴스를 포함) 대신 구성 요소 모델 기반 복합 모델을 생성해야 합니다. 자세한 내용은 다음 섹션을 참조하시기 바랍니다.

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

```
    ]
  }
```

구성 요소 모델 기반 구성 요소 생성(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 구성 요소 모델을 기반으로 구성 요소를 생성할 수 있습니다.

구성 요소 모델 기반 구성 요소를 생성하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 구성 요소를 추가하고자 하는 자산 모델을 선택합니다.
4. 속성 탭에서 구성 요소를 선택합니다.
5. 구성 요소 생성을 선택합니다.
6. 구성 요소 생성 페이지에서 다음을 수행합니다.
 - a. 구성 요소를 기반으로 할 구성 요소 모델을 선택합니다.
 - b. **ServoMotor** 또는 **ServoMotor Model**과 같은 구성 요소의 이름을 입력합니다. 이 이름은 이 리전의 계정에 있는 모든 구성 요소에서 고유해야 합니다.
 - c. 구성 요소 생성을 선택합니다.

구성 요소 모델 기반 사용자 지정 복합 모델 생성(AWS CLI)

AWS CLI 를 사용하여 자산 모델 내에 component-model-based 사용자 지정 복합 모델을 생성할 수 있습니다. 구성 요소 모델 기반 사용자 지정 복합 모델은 다른 곳에서 이미 정의한 구성 요소 모델에 대한 참조입니다.

[CreateAssetModelCompositeModel](#) 작업을 사용하여 구성 요소 모델 기반 사용자 지정 복합 모델을 생성합니다. 이 작업에는 다음과 같은 구조의 페이로드가 필요합니다.

Note

이 예시에서 `composedAssetModelId`의 값은 기존 구성 요소 모델의 자산 모델 ID 또는 외부 ID입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요. 구성 요소 모델을 생성하는 방법의 예시는 [구성 요소 모델 생성\(AWS CLI\)](#) 섹션을 참조하세요.

```
{
  "assetModelCompositeModelName": "CNCLathe_ServoMotorA",
  "assetModelCompositeModelType": "CUSTOM",
  "composedAssetModelId": component model ID
}
```

단순한 참조이므로 구성 요소 모델 기반 사용자 지정 복합 모델에는 이름 외에는 고유한 속성이 없습니다.

자산 모델에 동일한 구성 요소의 인스턴스를 여러 개 추가하려는 경우(예: 여러 서보 모터가 있는 CNC 기계), 각각 고유한 이름이 있지만 모두 동일한 `composedAssetModelId`를 참조하는 여러 구성 요소 모델 기반 사용자 지정 복합 모델을 추가할 수 있습니다.

다른 구성 요소 내에 구성 요소를 중첩할 수 있습니다. 이렇게 하려면 이 예시와 같이 구성 요소 모델 중 하나에 구성 요소 모델 기반 복합 모델을 추가할 수 있습니다.

에서 자산 모델에 대한 자산 생성 AWS IoT SiteWise

자산 모델에서 자산을 생성할 수 있습니다. 자산을 생성하려면 먼저 자산 모델이 있어야 합니다. 자산 모델을 생성하지 않은 경우 [에서 자산 모델 생성 AWS IoT SiteWise](#) 단원을 참조하세요.

Note

ACTIVE 모델에서만 자산을 만들 수 있습니다. 모델의 상태가 ACTIVE가 아닌 경우 해당 모델에서 자산을 만들기 전에 최대 몇 분 정도 기다려야 할 수 있습니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

주제

- [자산 생성\(콘솔\)](#)
- [자산 생성\(AWS CLI\)](#)
- [새 자산 구성](#)

자산 생성(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자산을 생성할 수 있습니다.

자산을 생성하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 자산 생성을 선택합니다.
4. 자산 생성 페이지에서 다음을 수행합니다.
 - a. 모델에서 자산을 생성할 자산 모델을 선택합니다.

Note

모델이 ACTIVE(활성) 상태가 아닌 경우 활성 상태가 될 때까지 기다리거나, 모델이 FAILED(실패) 상태인 경우 문제를 해결해야 합니다.

- b. 자산의 이름을 입력합니다.
- c. (선택 사항) 자산에 대한 태그를 추가합니다. 자세한 내용은 [AWS IoT SiteWise 리소스에 태그 지정](#) 단원을 참조하십시오.
- d. 자산 생성을 선택합니다.

자산을 생성하면 AWS IoT SiteWise 콘솔이 새 자산 페이지로 이동합니다. 이 페이지에서 자산의 상태를 볼 수 있습니다. 상태는 처음에 CREATING(생성 중)입니다. 이 페이지는 자동으로 업데이트되므로 자산 상태가 업데이트될 때까지 기다릴 수 있습니다.

Note

자산 생성 프로세스에는 최대 1분이 소요될 수 있습니다. 상태가 활성이 된 후에는 자산에 대한 업데이트 작업을 수행할 수 있습니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

자산을 생성한 후 [새 자산 구성](#) 단원을 참조하세요.

자산 생성(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산 모델에서 자산을 생성할 수 있습니다.

자산을 만들려면 `assetModelId`가 있어야 합니다. 자산 모델을 생성했지만 `assetModelId`를 모르는 경우 [ListAssetModels](#) API를 사용하여 모든 자산 모델을 볼 수 있습니다.

자산 모델에서 자산을 생성하려면 다음 파라미터와 함께 [CreateAsset](#) API를 사용합니다.

- **assetName** - 새 자산의 이름입니다. 자산을 식별하는 데 도움이 되는 이름을 자산에 지정합니다.
- **assetModelId** - 자산의 ID입니다. 이는 UUID 형식의 실제 ID이거나 **externalId:myExternalId**(있는 경우)입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

자산을 생성하려면(AWS CLI) 다음을 수행하세요.

- 다음 명령을 실행하여 자산을 생성합니다. **asset-name**을 자산의 이름으로 바꾸고 **asset-model-id**를 자산 모델의 ID 또는 외부 ID로 바꿉니다.

```
aws iotsitewise create-asset \
  --asset-name asset-name \
  --asset-model-id asset-model-id
```

이 작업은 다음과 같은 형식으로 새 자산의 세부 정보와 상태가 포함된 응답을 반환합니다.

```
{
  "assetId": "String",
  "assetArn": "String",
  "assetStatus": {
    "state": "String",
    "error": {
      "code": "String",
      "message": "String"
    }
  }
}
```

자산이 생성될 때까지 자산의 state는 CREATING입니다.

Note

자산 생성 프로세스에는 최대 1분이 소요될 수 있습니다. 자산의 상태를 확인하려면 자산의 ID를 **assetId** 파라미터로 사용하는 [DescribeAsset](#) 작업을 사용하세요. 자산의 state가 ACTIVE이면 자산에 대한 업데이트 작업을 수행할 수 있습니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

자산을 생성한 후 [새 자산 구성](#) 단원을 참조하세요.

새 자산 구성

에서 자산을 생성한 후 자산과 해당 데이터를 완전히 활용하기 위해 취할 수 있는 몇 AWS IoT SiteWise가지 다음 단계가 있습니다. 이러한 단계에는 자산에서 데이터를 수집하도록 데이터 스트림 구성, 자산의 성능을 모니터링하도록 경고 및 알림 설정, 자산의 데이터를 표시하도록 시각화 및 대시보드 생성, 추가 분석 또는 자동화를 위해 자산을 다른 AWS 서비스 또는 타사 애플리케이션과 통합 등이 포함될 수 있습니다.

다음 선택적 작업을 사용하여 자산 구성을 완료합니다.

- [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 자산에 측정 속성이 있는 경우.
- [속성 값 업데이트](#) 자산에 고유한 속성 값이 있는 경우.
- [자산 연결 및 연결 해제](#) 자산이 상위 자산인 경우.

에서 자산 검색 AWS IoT SiteWise Console

AWS IoT SiteWise Console 검색 기능을 사용하여 메타데이터 및 실시간 속성 값 필터를 기반으로 자산을 찾을 수 있습니다.

사전 조건

AWS IoT SiteWise 는 산업 데이터를 더 잘 구성하고 모델링하기 AWS IoT TwinMaker 위해와 통합할 수 있는 권한이 필요합니다. 에 권한을 부여한 경우 [ExecuteQuery](#) API를 AWS IoT SiteWise사용합니다. 에 대한 권한을 부여하지 않았고 시작하는 데 도움이 AWS IoT SiteWise필요한 경우 섹션을 참조하세요 [AWS IoT SiteWise 및 통합 AWS IoT TwinMaker](#).

에 대한 고급 검색 AWS IoT SiteWise Console

메타데이터 검색

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창의 자산 아래에서 고급 검색을 선택합니다.
3. 고급 검색 아래에서 메타데이터 검색 옵션을 선택합니다.
4. 파라미터를 입력합니다. 효율적인 검색을 위해 가능한 한 많은 필드를 입력합니다.

- a. 자산 이름 - 전체 자산 이름 또는 광범위한 검색을 위한 부분 이름을 입력합니다.
 - b. 속성 이름 - 전체 속성 이름 또는 광범위한 검색을 위한 부분 이름을 입력합니다.
 - c. 연산자 - 다음 중 연산자를 선택합니다.
 - =
 - <
 - >
 - <=
 - >=
 - d. 속성 값 - 이 값은 속성의 최신 값과 비교됩니다.
 - e. 속성 값 유형 - 속성의 데이터 유형입니다. 다음 중에서 선택합니다.
 - 배정밀도 실수
 - 정수
 - 문자열
 - 부울
5. 검색을 선택합니다.
6. 검색 결과 테이블의 이름 열에서 자산을 선택합니다. 그러면 해당 자산에 대한 세부 자산 페이지로 이동합니다.

Assets

Assets represent Industrial devices and processes that send data streams to SiteWise. Models are structures that enforce a specific model of properties and hierarchies for all instances of each asset. You must create every asset from a model.

Advanced search
Use advanced search to find assets based on specific metadata. In addition, you can enter SQL queries directly in the query builder.

Metadata search | Query builder

Asset name: Level-2 | Property name: power_max | Operator: > | Property value: 20 | Property value type: Double

Clear | Search

Search results (2)

Name	Asset id	Description
Level-2-asset-1	d0e9019b-9c38-4316-b574-38317aa38143	
Level-2-asset-2	b9c0d2fc-1527-42ce-8ba2-d1a4e8ff43de	Example description

부분 검색

자산 검색을 위해 모든 파라미터를 제공할 필요는 없습니다. 메타데이터 검색 옵션을 사용한 부분 검색의 몇 가지 예는 다음과 같습니다.

- 이름으로 자산 찾기:
 - 자산 이름 필드에만 값을 입력합니다.
 - 속성 이름 및 속성 값 필드가 비어 있습니다.
- 특정 이름의 속성이 포함된 자산 찾기:
 - 속성 이름 필드에만 값을 입력합니다.
 - 자산 이름 및 속성 값 필드가 비어 있습니다.
- 속성의 최신 값을 기반으로 자산 찾기:
 - 속성 이름 및 속성 값 필드에 값을 입력합니다.
 - 연산자 및 속성 값 유형을 선택합니다.

쿼리 빌더 검색

1. AWS IoT SiteWise Console로 이동합니다.

2. 탐색 창 의 자산 아래에서 고급 검색을 선택합니다.
3. 고급 검색에서 쿼리 빌더 옵션을 선택합니다.
4. 쿼리 빌더 창에서 SQL 쿼리를 작성하여 `asset_name`, `asset_id` 및 `asset_description`을 검색합니다.
5. 검색을 선택합니다.
6. 검색 결과 테이블의 이름 열에서 자산을 선택합니다. 그러면 해당 자산에 대한 세부 자산 페이지로 이동합니다.

The screenshot shows the AWS IoT SiteWise console interface. At the top, there's a navigation bar with the AWS logo, 'Services' menu, a search bar, and a region dropdown set to 'N. Virginia'. Below this, the 'Assets' page is displayed. It includes a 'Create asset' button and a description of assets. The 'Advanced search' section is active, with the 'Query builder' tab selected. A SQL query is entered in the query builder: `SELECT a.asset_id, a.asset_name, a.asset_description FROM asset a, asset_property p, latest_value_time_series ts WHERE a.asset_name LIKE '%asset-2%' AND a.property_name = 'temperature_f' AND ts.double_value > 50.0`. Below the query, there are 'Clear' and 'Search' buttons. The search results show two items in a table:

Name	Asset id	Description
Level-2a-asset-2	4fed596d-e903-4338-86db-34ca9301233a	Generator #3
Level-2b-asset-2	b4ac2b24-4fce-4a72-9fea-ef6d0f741e8d	Generator #2

Note

- 검색 결과 테이블에 유효한 자산이 표시되도록 하려면 SQL 쿼리의 SELECT 절에 `asset_name` 및 `asset_id` 필드가 포함되어야 합니다.
- 쿼리 빌더는 결과 테이블에 이름, 자산 ID 및 설명만 표시합니다. SELECT 절에 필드를 더 추가해도 결과 테이블에 열이 더 추가되지는 않습니다.

속성 값 업데이트

자산은 속성의 기본 값을 포함하여 자산 모델의 속성을 상속합니다. 일부 경우에는 자산 제조업체 속성과 같이 자산 모델의 기본 속성을 유지해야 할 수 있습니다. 다른 일부 경우에는 자산의 위도 및 경도와 같이 상속된 속성을 업데이트해야 할 수 있습니다.

Updating an attribute value (console)

AWS IoT SiteWise 콘솔을 사용하여 속성 자산 속성의 값을 업데이트할 수 있습니다.

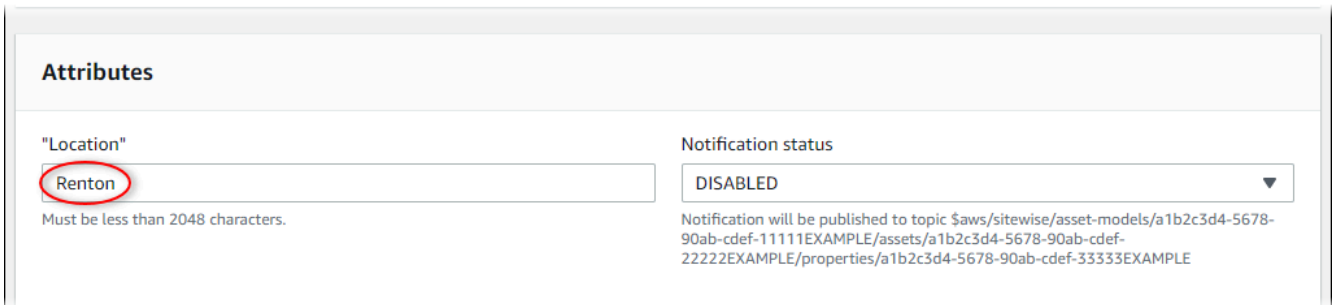
속성의 값을 업데이트하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 속성을 업데이트할 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 편집을 선택합니다.
5. 업데이트할 속성을 찾은 다음, 새 값을 입력합니다.



Attributes

"Location"

Renton

Must be less than 2048 characters.

Notification status

DISABLED

Notification will be published to topic \$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE

6. 저장을 선택합니다.

Updating an attribute value (AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 속성 값을 업데이트할 수 있습니다.

이 절차를 완료하려면 자산의 `assetId` 및 속성의 `propertyId`를 알고 있어야 합니다. 외부 ID를 사용할 수도 있습니다. 자산을 생성했지만 해당 `assetId`를 모르고 있는 경우에는 [ListAssets](#) API

를 사용하여 특정 모델에 대한 모든 자산을 표시합니다. [DescribeAsset](#) 작업을 사용하여 속성 ID를 포함한 해당 자산의 속성을 봅니다.

[BatchPutAssetPropertyValue](#) 작업을 사용하여 자산에 속성 값을 할당합니다. 이 작업을 사용하여 한 번에 여러 속성을 설정할 수 있습니다. 이 작업의 페이로드에는 각각 자산 ID, 속성 ID, 속성 값을 포함하는 항목 목록이 포함되어 있습니다.

속성의 값을 업데이트하려면(AWS CLI) 다음을 수행하세요.

1. `batch-put-payload.json`이라는 파일을 만들고 다음 JSON 개체를 파일에 복사합니다. 이 페이로드 예제에서는 풍력 터빈의 위도와 경도를 설정하는 방법을 보여줍니다. ID, 값 및 타임스탬프를 업데이트하여 사용 사례에 대한 페이로드를 수정합니다.

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

```
    ]
  }
]
}
```

- 페이로드의 각 항목은 고유한 문자열로 정의할 수 있는 entryId를 포함합니다. 요청 항목이 실패하면 각 오류에 해당 요청의 entryId가 포함되므로 다시 시도할 요청을 알 수 있습니다.
- 속성 값을 설정하기 위해 각 특성 속성의 propertyValues 목록에 timestamp-quality-value(TQV) 구조 하나를 포함시킬 수 있습니다. 이 구조에는 새 value와 현재 timestamp가 포함되어야 합니다.
 - value - 설정되는 속성 유형에 따라 다음 필드 중 하나를 포함하는 구조입니다.
 - booleanValue
 - doubleValue
 - integerValue
 - stringValue
 - nullValue
 - timestamp - 현재 Unix 에POCH 시간을 초 단위로 포함하는 구조로, timeInSeconds.는 과거 7일 이상 또는 향후 5분 이상 존재한 타임스탬프가 있는 모든 데이터 포인트를 AWS IoT SiteWise 거부합니다.

[BatchPutAssetPropertyValue](#)에 대한 페이로드를 준비하는 방법에 대한 자세한 내용은 [AWS IoT SiteWise APIs 사용하여 데이터 수집](#) 단원을 참조하세요.

2. 다음 명령을 실행하여 속성 값을 AWS IoT SiteWise에 전송합니다.

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

자산 연결 및 연결 해제

자산의 모델이 하위 자산 모델 계층 구조를 정의하는 경우, 하위 자산을 자산에 연결할 수 있습니다. 상위 자산은 연결된 자산에서 데이터에 액세스하고 집계할 수 있습니다. 계층적 자산 모델에 대한 자세한 내용은 [자산 모델 계층 정의](#) 단원을 참조하세요.

주제

- [자산 연결 및 연결 해제\(콘솔\)](#)
- [자산 연결 및 연결 해제\(AWS CLI\)](#)

자산 연결 및 연결 해제(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자산을 연결 및 연결 해제할 수 있습니다.

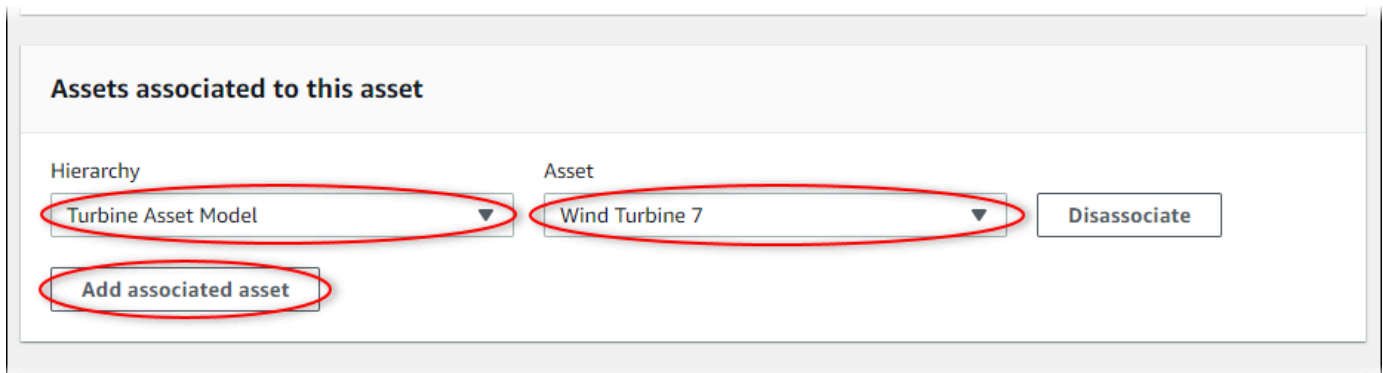
자산을 연결하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 하위 자산을 연결할 상위 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 편집을 선택합니다.
5. 이 자산에 연결된 자산에서 연결된 자산 추가를 선택합니다.



6. 계층 구조에서 상위 자산과 하위 자산 간의 관계를 정의하는 계층 구조를 선택합니다.
7. 자산에서 연결할 하위 자산을 선택합니다.
8. 저장을 선택합니다.

자산의 연결을 해제하려면(콘솔) 다음을 수행하세요.

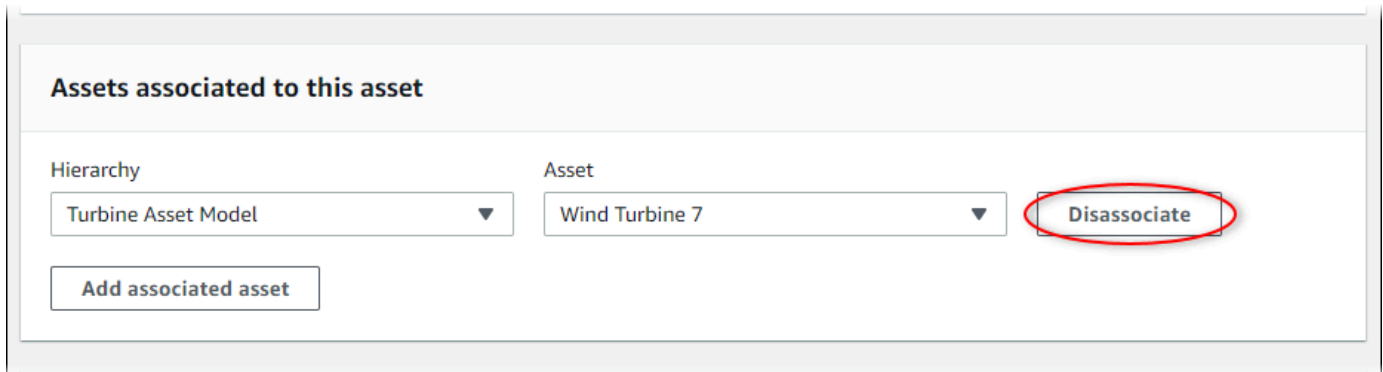
1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.

3. 하위 자산의 연결을 해제할 상위 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 편집을 선택합니다.
5. 이 자산에 연결된 자산에서 자산에 대한 연결 해제를 선택합니다.



6. 저장을 선택합니다.

자산 연결 및 연결 해제(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산을 연결 및 연결 해제할 수 있습니다.

이 절차를 수행하려면 상위 자산 모델에서 하위 자산 모델에 대한 관계를 정의하는 계층 구조 (hierarchyId)의 ID를 알고 있어야 합니다. [DescribeAsset](#) 작업을 사용하여 응답에서 계층 구조 ID를 찾습니다.

계층 구조 ID를 찾으려면

- 다음 명령을 실행하여 상위 자산을 설명합니다. *parent-asset-id*를 상위 자산의 ID 또는 외부 ID로 바꿉니다.

```
aws iotsitewise describe-asset --asset-id parent-asset-id
```

이 작업은 자산의 세부 정보가 포함된 응답을 반환합니다. 응답에는 다음과 같은 구조의 `assetHierarchies` 목록이 포함됩니다.

```
{
```

```

...
"assetHierarchies": [
  {
    "id": "String",
    "name": "String"
  }
],
...
}

```

계층 구조 ID는 자산 계층 구조 목록에 있는 계층 구조의 id 값입니다.

계층 구조 ID를 확인한 후에는 자산을 해당 계층 구조와 연결하거나 연결 해제할 수 있습니다.

하위 자산을 상위 자산에 연결하려면 [AssociateAssets](#) 작업을 사용합니다. 상위 자산에서 하위 자산의 연결을 해제하려면 [DisassociateAssets](#) 작업을 사용합니다. 다음 파라미터를 지정합니다. 이 파라미터는 두 작업에 모두 동일합니다.

- `assetId` - 상위 자산의 ID 또는 외부 ID입니다.
- `hierarchyId` - 상위 자산의 계층 구조 ID 또는 외부 ID입니다.
- `childAssetId` - 하위 자산의 ID 또는 외부 ID입니다.

자산을 연결하려면(AWS CLI) 다음을 수행하세요.

- 다음 명령을 실행하여 하위 자산을 상위 자산에 연결합니다. *parent-asset-id*, *hierarchy-id* 및 *child-asset-id*를 해당 ID로 바꿉니다.

```

aws iotsitewise associate-assets \
  --asset-id parent-asset-id \
  --hierarchy-id hierarchy-id \
  --child-asset-id child-asset-id

```

자산의 연결을 해제하려면(AWS CLI) 다음을 수행하세요.

- 다음 명령을 실행하여 상위 자산에서 하위 자산의 연결을 해제합니다. *parent-asset-id*, *hierarchy-id* 및 *child-asset-id*를 해당 ID로 바꿉니다.

```

aws iotsitewise disassociate-assets \

```

```
--asset-id parent-asset-id \  
--hierarchy-id hierarchy-id \  
--child-asset-id child-asset-id
```

자산 및 모델 업데이트

에서 자산, 자산 모델 및 구성 요소 모델을 업데이트 AWS IoT SiteWise 하여 이름과 정의를 수정할 수 있습니다. 이러한 업데이트 작업은 비동기식이며 전파하는 데 시간이 걸립니다 AWS IoT SiteWise. 추가로 변경하기 전에 자산 또는 모델의 상태를 확인합니다. 업데이트된 자산이나 모델을 계속 사용하려면 변경 사항이 전파될 때까지 기다려야 합니다.

주제

- [에서 자산 업데이트 AWS IoT SiteWise](#)
- [자산 모델 및 구성 요소 모델 업데이트](#)
- [사용자 지정 복합 모델 업데이트\(구성 요소\)](#)
- [자산 모델 쓰기를 위한 낙관적 잠금](#)

에서 자산 업데이트 AWS IoT SiteWise

AWS IoT SiteWise 콘솔 또는 API를 사용하여 자산 이름을 업데이트할 수 있습니다.

자산을 업데이트할 때 자산의 상태는 변경 사항이 전파될 때까지 UPDATING입니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

주제

- [자산 업데이트\(콘솔\)](#)
- [자산 업데이트\(AWS CLI\)](#)

자산 업데이트(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자산 세부 정보를 업데이트할 수 있습니다.

자산을 업데이트하려면(콘솔)

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.

3. 업데이트할 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 편집을 선택합니다.

5. 자산의 이름을 업데이트합니다.

6. (선택 사항) 이 페이지에서 자산에 대한 기타 정보를 업데이트합니다. 자세한 내용은 다음 자료를 참조하세요.

- [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#)
- [속성 값 업데이트](#)
- [다른 AWS 서비스와 상호 작용](#)

7. 저장을 선택합니다.

자산 업데이트(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산 이름을 업데이트할 수 있습니다.

[UpdateAsset](#) 작업을 사용하여 자산을 업데이트합니다. 다음 파라미터를 지정합니다.

- `assetId` - 자산의 ID입니다. 이는 UUID 형식의 실제 ID이거나 `externalId:myExternalId`(있는 경우)입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.
- `assetName` - 자산의 새 이름입니다.

자산의 이름을 업데이트하려면(AWS CLI) 다음을 수행하세요.

- 다음 명령을 실행하여 자산의 이름을 업데이트합니다. `asset-id`를 자산의 ID 또는 외부 ID로 바꿉니다. `asset-name`을 자산의 새 이름으로 업데이트합니다.

```
aws iotsitewise update-asset \
  --asset-id asset-id \
  --asset-name asset-name
```

자산 모델 및 구성 요소 모델 업데이트

AWS IoT SiteWise 콘솔 또는 API를 사용하여 자산 모델 또는 구성 요소 모델을 업데이트합니다.

기존 속성의 유형 또는 데이터 유형과 기존 지표의 기간은 변경할 수 없습니다. 또한 모델의 유형을 자산 모델에서 구성 요소 모델로 또는 그 반대 방향으로 변경할 수 없습니다.

Important

- 자산 모델 또는 구성 요소 모델에서 속성을 제거하면 해당 속성에 대한 모든 이전 데이터를 AWS IoT SiteWise 삭제합니다. 구성 요소 모델의 경우, 이는 해당 구성 요소 모델을 사용하는 모든 자산 모델에 영향을 미치므로 변경 사항이 얼마나 광범위하게 적용될 수 있는지 특히 신중하게 고려해야 합니다.
- 자산 모델에서 계층 구조 정의를 제거하면 해당 계층 구조의 모든 자산을 연결 AWS IoT SiteWise 해제합니다.

자산 모델을 업데이트하면 해당 모델을 기반으로 하는 모든 자산에 기본 모델의 변경 사항이 반영됩니다. 변경 사항이 전파될 때까지 각 자산에는 UPDATING 상태가 있습니다. 이러한 자산과 상호 작용하기 전에 자산의 상태가 ACTIVE 상태로 돌아올 때까지 기다려야 합니다. 이 기간 동안 업데이트된 자산 모델의 상태는 PROPAGATING가 됩니다.

구성 요소 모델을 업데이트하면 해당 구성 요소 모델을 통합하는 모든 자산 모델이 변경 사항을 반영합니다. 구성 요소 모델의 변경 사항이 전파될 때까지 영향을 받는 각 자산 모델에는 이전 단락에 설명된 것처럼 UPDATING 상태가 표시된 후, 연결된 자산이 업데이트되면서 PROPAGATING 상태가 표시됩니다. 이러한 자산 모델과 상호 작용하기 전에 자산 모델이 ACTIVE 상태로 돌아올 때까지 기다려야 합니다. 이 기간 동안 업데이트된 구성 요소 모델의 상태는 PROPAGATING이 됩니다.

자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오.

주제

- [자산 또는 구성 요소 모델 업데이트\(콘솔\)](#)
- [자산 또는 구성 요소 모델 업데이트\(AWS CLI\)](#)

자산 또는 구성 요소 모델 업데이트(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자산 모델 또는 구성 요소 모델을 업데이트할 수 있습니다.

자산 모델 또는 구성 요소 모델을 업데이트하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 업데이트할 자산 모델 또는 구성 요소 모델을 선택합니다.
4. 편집을 선택합니다.
5. 모델 편집 페이지에서 다음 중 하나를 수행합니다.
 - 모델 세부 정보에서 모델의 이름을 변경합니다.
 - 속성 정의를 변경합니다. 기존 속성의 데이터 형식은 변경할 수 없습니다. 자세한 내용은 [정적 데이터\(속성\) 정의](#) 단원을 참조하십시오.
 - 측정 정의를 변경합니다. 기존 측정의 데이터 형식은 변경할 수 없습니다. 자세한 내용은 [장비의 데이터 스트림 정의\(측정값\)](#) 단원을 참조하십시오.
 - 변환 정의를 변경합니다. 자세한 내용은 [데이터 변환\(변환\)](#) 단원을 참조하십시오.
 - 지표 정의를 변경합니다. 기존 지표의 시간 간격은 변경할 수 없습니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하십시오.
 - (자산 모델만 해당) 계층 구조 정의를 변경합니다. 기존 계층 구조의 계층 구조 모델은 변경할 수 없습니다. 자세한 내용은 [자산 모델 계층 정의](#) 단원을 참조하십시오.
6. 저장을 선택합니다.

Note

마지막으로 모델 편집 페이지를 연 이후 다른 사용자가 자산 모델을 성공적으로 업데이트하면 콘솔에서 수행된 업데이트 요청이 거부됩니다. 콘솔은 사용자에게 모델 편집 페이지를 새로 고쳐 업데이트된 모델을 가져오라는 메시지를 표시합니다. 다시 업데이트하고 저장을 다시 시도해야 합니다. 자세한 내용은 [자산 모델 쓰기를 위한 낙관적 잠금](#) 섹션을 참조하세요.

자산 또는 구성 요소 모델 업데이트(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산 모델 또는 구성 요소 모델을 업데이트합니다.

[UpdateAssetModel](#) API를 사용하여 자산 모델 또는 구성 요소 모델의 이름, 설명 및 속성을 업데이트합니다. 자산 모델의 경우에만 계층을 업데이트할 수 있습니다. 다음 파라미터를 지정합니다.

- `assetModelId` – 자산의 ID입니다. 이는 UUID 형식의 실제 ID이거나 `externalId:myExternalId`(있는 경우)입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

페이로드에 업데이트된 모델을 지정합니다. 자산 모델 또는 구성 요소 모델에 필요한 형식에 대해 자세히 알아보려면 [에서 자산 모델 생성 AWS IoT SiteWise](#) 섹션을 참조하세요.

Warning

[UpdateAssetModel](#) API는 페이로드에서 제공하는 모델로 기존 모델을 덮어씁니다. 모델의 속성이나 계층 구조가 삭제되지 않도록 하려면 업데이트된 모델 페이로드에 해당 ID와 정의를 포함해야 합니다. 모델의 기존 구조를 쿼리하는 방법을 알아보려면 [DescribeAssetModel](#) 작업을 참조하세요.

Note

다음 절차는 AWS/ALARM 유형의 복합 모델만 업데이트할 수 있습니다. CUSTOM 복합 모델을 업데이트하려면 대신 [UpdateAssetModelCompositeModel](#)을 사용합니다. 자세한 내용은 [사용자 지정 복합 모델 업데이트\(구성 요소\)](#) 단원을 참조하십시오.

자산 모델 또는 구성 요소 모델을 업데이트하려면(AWS CLI) 다음을 수행하세요.

1. 다음 명령을 실행하여 기존 자산 모델 정의를 검색합니다. *asset-model-id*를 업데이트할 자산 모델 또는 구성 요소 모델의 ID 또는 외부 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

위 명령은 모델의 최신 버전에 해당하는 모델 정의를 반환합니다.

자산 모델이 FAILED 상태에 있는 사용 사례의 경우 활성 버전에 해당하는 유효한 모델 정의를 검색하여 업데이트 요청을 빌드합니다. 세부 정보는 [자산 모델 버전](#) 섹션을 참조하세요. 다음 명령을 실행하여 활성 모델 정의를 검색합니다.

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id --asset-model-version ACTIVE
```

이 작업은 자산 모델의 세부 사항이 포함된 응답을 반환합니다. 응답의 구조는 다음과 같습니다.

```
{
  "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"
}
```

자세한 내용은 [DescribeAssetModel](#) 작업을 참조하세요.

2. update-asset-model.json이라는 파일을 만들고 이전 명령의 응답을 파일에 복사합니다.
3. update-asset-model.json의 JSON 객체에서 다음 키-값 페어를 제거합니다.

- assetModelId
- assetModelArn
- assetModelCompositeModelSummaries
- assetModelCreationDate
- assetModelLastUpdateDate
- assetModelStatus
- assetModelType
- assetModelVersion
- eTag

[UpdateAssetModel](#) 작업에는 다음과 같은 구조의 페이로드가 필요합니다.

```
{
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition,
  "assetModelCompositeModels": Array of AssetModelCompositeModel
}
```

4. update-asset-model.json에서 다음을 수행합니다.

- 자산 모델의 이름(assetModelName)을 변경합니다.
- 자산 모델의 설명(assetModelDescription)을 변경, 추가 또는 제거합니다.
- 자산 모델의 속성(assetModelProperties)을 변경, 추가 또는 제거합니다. 기본 속성의 dataType 또는 기존 지표의 window는 변경할 수 없습니다. 자세한 내용은 [데이터 속성 정의](#) 단원을 참조하십시오.
- 자산 모델의 계층 구조(assetModelHierarchies)를 변경, 추가 또는 제거합니다. 기존 계층 구조의 childAssetModelId는 변경할 수 없습니다. 자세한 내용은 [자산 모델 계층 정의](#) 단원을 참조하십시오.
- AWS/ALARM(assetModelCompositeModels) 유형 자산 모델의 복합 모델을 변경, 추가 또는 제거합니다. 경보는 다른 속성을 모니터링하므로 장비나 프로세스에 주의가 필요한 시점을 식별할 수 있습니다. 각 경고 정의는 경보가 사용하는 속성 집합을 표준화하는 복합 모델입니다. 자세한 내용은 [에서 경보를 사용하여 데이터 모니터링 AWS IoT SiteWise](#) 및 [에서 자산 모델에 대한 경고 정의 AWS IoT SiteWise](#) 섹션을 참조하세요.

5. 다음 명령을 실행하여 update-asset-model.json에 저장된 정의로 자산 모델을 업데이트합니다. *asset-model-id*를 자산 모델의 ID로 바꿉니다.

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://model-payload.json
```

⚠ Important

여러 사용자가 자산 모델을 동시에 업데이트하면 한 사용자가 다른 사용자의 변경 사항을 실수로 덮어쓸 수 있습니다. 이를 방지하려면 조건부 업데이트 요청을 정의해야 합니다. [자산 모델 쓰기를 위한 낙관적 잠금](#)(들) 참조하세요.

사용자 지정 복합 모델 업데이트(구성 요소)

AWS IoT SiteWise API를 사용하여 사용자 지정 복합 모델을 업데이트하거나 AWS IoT SiteWise 콘솔을 사용하여 구성 요소를 업데이트할 수 있습니다.

주제

- [구성 요소 업데이트\(콘솔\)](#)
- [사용자 지정 복합 모델 업데이트\(AWS CLI\)](#)

구성 요소 업데이트(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 구성 요소를 업데이트할 수 있습니다.

구성 요소를 업데이트하려면(콘솔)

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 구성 요소가 있는 자산 모델을 선택합니다.
4. 속성 탭에서 구성 요소를 선택합니다.
5. 업데이트할 구성 요소를 선택합니다.
6. 편집을 선택합니다.
7. 구성 요소 편집 페이지에서 다음 중 하나를 수행합니다.
 - 모델 세부 정보에서 모델의 이름을 변경합니다.
 - 속성 정의를 변경합니다. 기존 속성의 데이터 형식은 변경할 수 없습니다. 자세한 내용은 [정적 데이터\(속성\) 정의](#) 단원을 참조하십시오.
 - 측정 정의를 변경합니다. 기존 측정의 데이터 형식은 변경할 수 없습니다. 자세한 내용은 [장비의 데이터 스트림 정의\(측정값\)](#) 단원을 참조하십시오.
 - 변환 정의를 변경합니다. 자세한 내용은 [데이터 변환\(변환\)](#) 단원을 참조하십시오.

- 지표 정의를 변경합니다. 기존 지표의 시간 간격은 변경할 수 없습니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하십시오.

8. 저장을 선택합니다.

사용자 지정 복합 모델 업데이트(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 사용자 지정 복합 모델을 업데이트합니다.

이름 또는 설명을 업데이트하려면 [UpdateAssetModelCompositeModel](#) 작업을 사용합니다. 인라인 사용자 지정 복합 모델의 경우에만 속성도 업데이트할 수 있습니다. 참조된 구성 요소 모델은 관련 속성을 제공하므로 구성 요소 모델 기반 사용자 지정 복합 모델의 속성을 업데이트할 수 없습니다.

Important

사용자 지정 복합 모델에서 속성을 제거하면 해당 속성에 대한 모든 이전 데이터를 AWS IoT SiteWise 삭제합니다. 기존 속성의 유형 또는 데이터 유형은 변경할 수 없습니다. 기존 복합 모델 속성을 동일한 name의 새 속성으로 바꾸려면 다음을 수행합니다.

- 기존 속성 전체가 제거된 상태로 UpdateAssetModelCompositeModel 요청을 제출합니다.
- 새 속성이 포함된 두 번째 UpdateAssetModelCompositeModel 요청을 제출합니다. 새 자산 속성은 이전 자산 속성과 name 동일하며 고유한 새를 AWS IoT SiteWise 생성합니다id.

사용자 지정 복합 모델을 업데이트하려면(AWS CLI) 다음을 수행하세요.

- 다음 명령을 실행하여 기존 복합 모델 정의를 검색합니다. *composite-model-id*를 업데이트할 사용자 지정 복합 모델의 ID 또는 외부 ID로 바꾸고 *asset-model-id*를 사용자 지정 복합 모델과 연결된 자산 모델로 바꿉니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서를 참조하십시오.
 - 아래 명령을 실행합니다.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-composite-model-id composite-model-id \
  --asset-model-id asset-model-id
```


- b. 위 명령은 연결된 모델의 최신 버전에 해당하는 복합 모델 정의를 반환합니다. 자산 모델이 FAILED 상태에 있는 사용 사례의 경우 활성 버전에 해당하는 유효한 모델 정의를 검색하여 업데이트 요청을 빌드합니다. 세부 정보는 [자산 모델 버전](#) 섹션을 참조하세요.
- c. 다음 명령을 실행하여 활성 모델 정의를 검색합니다.

```
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. 자세한 내용은 [DescribeAssetModelCompositeModel](#) 작업을 참조하세요.
2. update-custom-composite-model.json이라는 파일을 만들고 이전 명령의 응답을 파일에 복사합니다.
 3. 다음 필드를 제외하고 update-custom-composite-model.json의 JSON 객체에서 다음 카-값 페어를 제거합니다.
 - assetModelCompositeModelName
 - assetModelCompositeModelDescription(있는 경우)
 - assetModelCompositeModelProperties(있는 경우)
 4. update-custom-composite-model.json에서 다음을 수행합니다.
 - assetModelCompositeModelName의 값을 변경합니다.
 - assetModelCompositeModelDescription을 추가 또는 제거하거나 값을 변경합니다.
 - 인라인 사용자 지정 복합 모델만 해당: assetModelCompositeModelProperties에서 자산 모델의 속성을 변경, 추가 또는 제거합니다.

이 파일의 필수 형식에 대한 자세한 내용은 [UpdateAssetModelCompositeModel](#)에 대한 요청 구문을 참조하세요.

5. 다음 명령을 실행하여 update-custom-composite-model.json에 저장된 정의로 사용자 지정 복합 모델을 업데이트합니다. *composite-model-id*를 복합 모델의 ID로 바꾸고 *asset-model-id*를 해당 자산 모델의 ID로 바꿉니다.

```
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

여러 사용자가 자산 모델을 동시에 업데이트하면 한 사용자가 다른 사용자의 변경 사항을 실수로 덮어쓸 수 있습니다. 이를 방지하려면 조건부 업데이트 요청을 정의해야 합니다. [자산 모델 쓰기를 위한 낙관적 잠금](#)(를) 참조하세요.

자산 모델 쓰기를 위한 낙관적 잠금

자산 모델을 업데이트할 때 사용자는 다음을 수행합니다.

1. 현재 자산 모델 정의를 읽습니다.
2. 필요한 변경 사항으로 자산 모델 정의를 편집합니다.
3. 자산 모델을 새 정의로 업데이트합니다.

두 사용자가 모델을 업데이트하는 시나리오에서는 다음이 가능합니다.

- 사용자 A는 자산 모델 X 정의를 읽습니다.
- 사용자 B는 자산 모델 X 정의를 읽고 변경 사항을 커밋하여 X의 정의를 수정합니다.
- 사용자 A는 사용자 B의 변경 사항을 확인하거나 통합하지 않고 자산 모델 X에 대한 사용자 B의 변경 사항을 커밋하고 덮어씁니다.

낙관적 잠금은 위 시나리오와 같이 실수로 덮어쓰는 것을 방지하기 AWS IoT SiteWise 위해에서 사용하는 메커니즘입니다. 낙관적 잠금은 업데이트 또는 삭제되는 자산 모델의 현재 버전이 AWS IoT SiteWise의 현재 버전과 동일하도록 하기 위한 전략입니다. 이렇게 하면 자산 모델 쓰기가 실수로 업데이트되어 덮어쓰여지지 않습니다.

낙관적 잠금을 사용하여 자산 모델 쓰기를 수행하려면 다음 단계를 따르세요.

주제

- [낙관적 잠금을 사용하여 자산 모델 쓰기 수행\(콘솔\)](#)
- [낙관적 잠금을 사용하여 자산 모델 쓰기 수행\(AWS CLI\)](#)

낙관적 잠금을 사용하여 자산 모델 쓰기 수행(콘솔)

아래 절차에서는 콘솔에서 자산 모델의 활성 버전에 대해 낙관적 잠금을 사용하여 자산 모델 쓰기를 수행하는 방법을 설명합니다.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 업데이트할 자산 모델 또는 구성 요소 모델을 선택합니다.
4. 편집을 선택합니다.
5. 모델 편집 페이지에서 변경 사항을 적용합니다.
6. 저장을 선택합니다.

Note

때때로 사용자가 모델 편집을 시작할 때와 편집한 내용을 모델에 저장할 때 사이에 하나 이상의 모델 업데이트가 성공하기도 합니다.

사용자가 성공한 새 업데이트를 실수로 덮어쓰지 않도록 하기 위해 사용자의 쓰기가 거부됩니다. 콘솔은 저장 버튼을 비활성화하고 사용자에게 모델 편집 페이지를 새로 고치라는 메시지를 표시합니다. 사용자는 모델의 새 활성 버전을 다시 업데이트해야 합니다. 사용자는 다음 추가 단계를 수행해야 합니다.

7. [Refresh]를 선택합니다.
8. 5단계와 6단계를 다시 따릅니다.

낙관적 잠금을 사용하여 자산 모델 쓰기 수행(AWS CLI)

아래 절차는 AWS CLI에서 낙관적 잠금을 사용하여 자산 모델 쓰기를 수행하는 방법을 설명합니다.

1. 현재 모델 정의와 연결된 ETag 가져오기

ETag는 자산 모델의 각 새 표현에 대해 생성된 고유한 토큰입니다. [DescribeAssetModel](#) API를 직접적으로 호출하여 응답에서 현재 자산 모델 정의 및 연결된 ETag를 가져옵니다.

동시 업데이트 중에 사용자는 성공적인 업데이트(ACTIVE 상태의 모델) 또는 실패한 업데이트(FAILED 상태의 모델)를 수행합니다. 사용자가 성공적인 업데이트를 실수로 덮어쓰지 않도록 하려면 [자산 모델 버전](#)에서 자산 모델의 활성 버전을 검색하고 ETag 값을 가져와야 합니다.

다음 명령 실행:

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id \
--asset-model-version ACTIVE
```

응답은 다음 구조를 반환합니다.

```
{
  "assetModelId": "String",
  "assetModelArn": "String",
  "assetModelName": "String",
  ...
  "eTag": "String"
}
```

Note

업데이트를 덮어쓰지 않으려면 자산 모델 및 해당 ETag의 최신 버전을 검색해야 합니다.

2. 쓰기 조건으로 UPDATE 및 DELETE 작업 수행

다음 자산 모델 API는 낙관적 잠금을 지원합니다.

- [UpdateAssetModel](#)
- [DeleteAssetModel](#)
- [CreateAssetModelCompositeModel](#)
- [UpdateAssetModelCompositeModel](#)
- [DeleteAssetModelCompositeModel](#)

Note

아래 시나리오에서는 UpdateAssetModel API를 참조로 사용합니다. 조건은 위에 나열된 모든 작업에 적용됩니다.

다음 시나리오에서는 동시성 제어 요구 사항에 따라 다양한 쓰기 조건을 설명합니다.

- 성공적인 업데이트를 덮어쓰지 않도록 다음 명령을 실행합니다. 마지막으로 읽은 활성 버전 이후 새 활성 버전이 없어야 합니다. 활성 버전의 읽기에 사용된 API 작업에서 반환된 ETag로 e-tag를 바꿉니다.

```
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
```

- 모델 생성에 실패하면 이에 대한 활성 버전이 FAILED 상태이므로 아직 존재하지 않습니다. 변경 사항이 커밋되기 전에 현재 있는 새 활성 버전을 덮어쓸 수 있습니다. 마지막 읽기 중에 활성 버전이 없는 경우 새 활성 버전을 덮어쓰지 않기 위해 다음 명령을 실행합니다.

```
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
```

- 성공하거나 실패한 업데이트를 덮어쓰지 않기 위해 다음 명령을 실행합니다. 이 명령은 마지막으로 최신 버전을 읽은 이후 최신 버전이 생성되지 않도록 하는 쓰기 조건을 정의합니다. 활성 버전의 읽기에 사용된 API 작업에서 반환된 ETag로 e-tag를 바꿉니다.

```
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
```

쓰기 조건이 FALSE로 평가되면 `PreconditionFailedException`과 함께 쓰기 요청이 실패합니다.

에서 자산 및 모델 삭제 AWS IoT SiteWise

자산 및 모델 작업을 마치 AWS IoT SiteWise 면에서 자산 및 모델을 삭제할 수 있습니다. 삭제 작업은 비동기식이며 전파하는 데 시간이 걸립니다 AWS IoT SiteWise.

주제

- [에서 자산 삭제 AWS IoT SiteWise](#)
- [에서 자산 모델 삭제 AWS IoT SiteWise](#)

에서 자산 삭제 AWS IoT SiteWise

AWS IoT SiteWise 콘솔 또는 API를 사용하여 환경에 더 이상 필요하지 않은 자산을 삭제할 수 있습니다. 자산 모델을 삭제하면 연결된 모든 자산 및 구성 요소 모델도 삭제됩니다. 그러나 자산 또는 모델은 영구적으로 삭제되며, 삭제된 리소스와 연결된 모든 데이터도 제거된다는 점에 유의해야 합니다. 자산 또는 모델을 삭제하기 전에 영향을 받을 수 있는 종속성 또는 통합을 검토하고 중요한 데이터를 백업했는지 확인하는 것이 좋습니다.

자산을 삭제하려면 먼저 하위 자산의 연결을 해제하고 상위 자산에서 해당 자산의 연결을 해제해야 합니다. 자세한 내용은 [자산 연결 및 연결 해제](#) 단원을 참조하십시오. AWS Command Line Interface (AWS CLI)를 사용하는 경우 [ListAssociatedAssets](#) 작업을 사용하여 자산의 하위 항목을 나열할 수 있습니다.

자산을 삭제할 때 자산의 상태는 변경 사항이 전파될 때까지 DELETING입니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오. 자산을 삭제한 후에는 해당 자산을 쿼리할 수 없습니다. 이 경우 API는 HTTP 404 응답을 반환합니다.

Important

AWS IoT SiteWise 는 삭제된 자산에 대한 모든 속성 데이터를 삭제합니다.

주제

- [자산 삭제\(콘솔\)](#)
- [자산 삭제\(AWS CLI\)](#)

자산 삭제(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자산을 삭제할 수 있습니다.

자산을 삭제하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 삭제할 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 자산에 연결된 자산이 있는 경우 각 자산을 삭제합니다. 자산의 이름을 선택하여 자산 페이지로 이동한 후, 이 페이지에서 자산을 삭제할 수 있습니다.
5. 자산 페이지에서 삭제를 선택합니다.
6. 자산 삭제 대화 상자에서 다음을 수행합니다.
 - a. **Delete**를 입력하여 삭제를 확인합니다.
 - b. 삭제를 선택합니다.

자산 삭제(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산을 삭제할 수 있습니다.

[DeleteAsset](#) 작업을 사용하여 자산을 삭제합니다. 다음 파라미터를 지정하세요.

- **assetId** – 자산의 ID입니다. 이는 UUID 형식의 실제 ID이거나 `externalId:myExternalId`(있는 경우)입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

자산을 삭제하려면(AWS CLI) 다음을 수행하세요.

1. 다음 명령을 실행하여 자산의 계층 구조를 나열합니다. **asset-id**를 자산의 ID 또는 외부 ID로 바꿉니다.

```
aws iotsitewise describe-asset --asset-id asset-id
```

이 작업은 자산의 세부 정보가 포함된 응답을 반환합니다. 응답에는 다음과 같은 구조의 `assetHierarchies` 목록이 포함됩니다.

```
{
  ...
  "assetHierarchies": [
    {
      "id": "String",
      "name": "String"
    }
  ],
  ...
}
```

자세한 내용은 [DescribeAsset](#) 작업을 참조하세요.

2. 각 계층 구조에 대해 다음 명령을 실행하여 해당 계층 구조와 연결된 자산의 하위 항목을 나열합니다. `asset-id`를 자산의 ID 또는 외부 ID로 바꾸고 `hierarchy-id`를 계층 구조의 ID 또는 외부 ID로 바꿉니다.

```
aws iotsitewise list-associated-assets \
  --asset-id asset-id \
  --hierarchy-id hierarchy-id
```

자세한 내용은 [ListAssociatedAssets](#) 작업을 참조하세요.

3. 다음 명령을 실행하여 연결된 각 자산을 삭제한 다음 자산을 삭제합니다. `asset-id`를 자산의 ID 또는 외부 ID로 바꿉니다.

```
aws iotsitewise delete-asset --asset-id asset-id
```

에서 자산 모델 삭제 AWS IoT SiteWise

AWS IoT SiteWise 콘솔 또는 API를 사용하여 자산 모델을 삭제할 수 있습니다.

자산 모델을 삭제하려면 먼저 자산 모델에서 생성되었던 모든 자산을 삭제해야 합니다.

자산 모델을 삭제할 때 자산 모델의 상태는 변경 사항이 전파될 때까지 DELETING입니다. 자세한 내용은 [자산 및 모델 상태](#) 단원을 참조하십시오. 자산 모델을 삭제한 후에는 해당 자산 모델을 쿼리할 수 없습니다. 이 경우 API는 HTTP 404 응답을 반환합니다.

주제

- [자산 모델 삭제\(콘솔\)](#)
- [자산 모델 삭제\(AWS CLI\)](#)

자산 모델 삭제(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자산 모델을 삭제할 수 있습니다.

자산 모델을 삭제하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 삭제할 자산 모델을 선택합니다.
4. 모델에 자산이 있는 경우 각 자산을 삭제합니다. 자산의 이름을 선택하여 자산 페이지로 이동한 후, 이 페이지에서 자산을 삭제할 수 있습니다. 자세한 내용은 [자산 삭제\(콘솔\)](#) 단원을 참조하십시오.
5. 모델 페이지에서 삭제를 선택합니다.
6. 모델 삭제 대화 상자에서 다음을 수행합니다.
 - a. **Delete**를 입력하여 삭제를 확인합니다.
 - b. 삭제를 선택합니다.

자산 모델 삭제(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산 모델을 삭제할 수 있습니다.

[DeleteAssetModel](#) 작업을 사용하여 자산 모델을 삭제합니다. 다음 파라미터를 지정하세요.

- `assetModelId` – 자산의 ID입니다. 이는 UUID 형식의 실제 ID이거나 `externalId:myExternalId`(있는 경우)입니다. 자세한 내용은 AWS IoT SiteWise 사용 설명서의 [외부 ID가 있는 참조 객체](#)를 참조하세요.

자산 모델을 삭제하려면(AWS CLI) 다음을 수행하세요.

1. 다음 명령을 실행하여 모델에서 생성된 모든 자산을 나열합니다. `asset-model-id`를 자산 모델의 ID 또는 외부 ID로 바꿉니다.

```
aws iotsitewise list-assets --asset-model-id asset-model-id
```

자세한 내용은 [ListAssets](#) 작업을 참조하세요.

- 이전의 명령이 모델에서 자산을 반환하는 경우 각 자산을 삭제합니다. 자세한 내용은 [자산 삭제 \(AWS CLI\)](#) 단원을 참조하십시오.
- 다음 명령을 실행하여 자산 모델을 삭제합니다. *asset-model-id*를 자산 모델의 ID 또는 외부 ID로 바꿉니다.

```
aws iotsitewise delete-asset-model --asset-model-id asset-model-id
```

Important

마지막 읽기 작업 이후 동시에 업데이트된 자산 모델을 삭제하지 않으려면 조건부 삭제 요청을 정의해야 합니다. [자산 모델 쓰기를 위한 낙관적 잠금](#)을(를) 참조하세요.

자산 및 모델을 사용한 일괄 작업

많은 수의 자산 또는 자산 모델을 사용하여 작업하려면 일괄 작업을 사용하여 리소스를 다른 위치로 일괄적으로 가져오고 내보냅니다. 예를 들어 Amazon S3 버킷에서 자산 또는 자산 모델을 정의하는 데이터 파일을 생성하고 AWS IoT SiteWise에서 일괄 가져오기를 사용하여 이들을 생성하거나 업데이트할 수 있습니다. 또는 자산 또는 자산 모델이 많은 경우 Amazon S3로 내보낼 AWS IoT SiteWise 수 있습니다.

Note

AWS IoT TwinMaker API에서 작업을 호출 AWS IoT SiteWise 하에서 대량 작업을 수행합니다. 워크 AWS IoT TwinMaker 스페이스를 설정 AWS IoT TwinMaker 하거나 생성하지 않고도 이 작업을 수행할 수 있습니다. AWS IoT SiteWise 콘텐츠를 배치할 수 있는 Amazon S3 버킷만 있으면 됩니다.

주제

- [주요 개념 및 용어](#)

- [지원되는 기능](#)
- [일괄 작업 사전 조건](#)
- [일괄 가져오기 작업 실행](#)
- [일괄 내보내기 작업 실행](#)
- [작업 진행 상황 추적 및 오류 처리](#)
- [메타데이터 가져오기 예시](#)
- [메타데이터 예시 내보내기](#)
- [AWS IoT SiteWise 메타데이터 전송 작업 스키마](#)

주요 개념 및 용어

AWS IoT SiteWise 대량 가져오기 및 내보내기 기능은 다음 개념과 용어에 의존합니다.

- 가져오기: Amazon S3 버킷의 파일에서 AWS IoT SiteWise로 자산 또는 자산 모델을 이동하는 작업입니다.
- 내보내기: 자산 또는 자산 모델에서 Amazon S3 버킷 AWS IoT SiteWise 으로 이동하는 작업입니다.
- 소스: 콘텐츠를 이동할 시작 위치입니다.

예를 들어 Amazon S3 버킷은 가져오기 소스이고 AWS IoT SiteWise 는 내보내기 소스입니다.

- 대상: 콘텐츠를 이동할 도착 위치입니다.

예를 들어 Amazon S3 버킷은 내보내기 대상이고는 가져오기 대상 AWS IoT SiteWise 입니다.

- AWS IoT SiteWise 스키마: 이 스키마는 메타데이터를 가져오고 내보내는 데 사용됩니다 AWS IoT SiteWise.
- 최상위 리소스: 자산 또는 자산 모델과 같이 개별적으로 생성하거나 업데이트할 수 있는 AWS IoT SiteWise 리소스입니다.
- 하위 리소스: 최상위 AWS IoT SiteWise 리소스 내의 중첩된 리소스입니다. 예를 들어 속성, 계층 구조 및 복합 모델이 있습니다.
- 메타데이터 : 리소스를 성공적으로 가져오거나 내보내는 데 필요한 주요 정보입니다. 메타데이터의 예로는 자산 및 자산 모델의 정의가 있습니다.
- metadataTransferJob: CreateMetadataTransferJob을 실행할 때 생성된 객체입니다.

지원되는 기능

이 주제에서는 일괄 작업을 실행할 때 수행할 수 있는 작업에 대해 설명합니다. 일괄 작업은 다음 기능을 지원합니다.

- 최상위 리소스 생성: ID를 정의하지 않거나 ID가 기존 ID와 일치하지 않는 자산 또는 자산 모델을 가져오면 새 리소스로 생성됩니다.
- 최상위 리소스 대체: ID가 이미 존재하는 ID와 일치하는 자산 또는 자산 모델을 가져오면 기존 리소스가 대체됩니다.
- 하위 리소스 생성, 대체 또는 삭제: 가져오기가 자산 또는 자산 모델과 같은 최상위 리소스를 대체하면 속성, 계층 구조 또는 복합 모델과 같은 모든 하위 리소스를 새 정의로 대체합니다.

예를 들어 일괄 가져오기 중에 자산 모델을 업데이트하고 업데이트된 버전이 원본에 없는 속성을 정의하면 새 속성이 생성됩니다. 이미 존재하는 속성을 정의하면 기존 속성이 업데이트됩니다. 업데이트된 자산 모델이 원본에 있는 속성을 생략하면 속성이 삭제됩니다.

- 최상위 리소스가 삭제되지 않음: 일괄 작업은 자산 또는 자산 모델을 삭제하지 않습니다. 일괄 작업은 자산 또는 자산 모델을 생성 또는 업데이트하기만 합니다.

일괄 작업 사전 조건

이 섹션에서는 AWS 서비스와 로컬 시스템 간에 리소스를 교환하기 위한 AWS Identity and Access Management (IAM) 권한을 포함하여 대량 작업 사전 조건을 설명합니다. 일괄 작업을 시작하기 전에 다음 필수 조건을 충족해야 합니다.

- 리소스를 저장할 Amazon S3 버킷을 생성합니다. Amazon S3에 대한 자세한 정보는 [Amazon S3란 무엇입니까?](#)에서 확인하세요.

IAM 권한

대량 작업을 수행하려면 Amazon S3 AWS IoT SiteWise와 로컬 시스템 간의 AWS 리소스 교환을 허용하는 권한이 있는 AWS Identity and Access Management (IAM) 정책을 생성해야 합니다. IAM 정책을 만드는 방법에 대한 자세한 내용은 [IAM 정책 만들기](#) 섹션을 참조하세요.

일괄 작업을 수행하려면 다음 정책이 필요합니다.

AWS IoT SiteWise 정책

이 정책은 대량 작업에 필요한 AWS IoT SiteWise API 작업에 대한 액세스를 허용합니다.

```
{
  "Sid": "SiteWiseApiAccess",
  "Effect": "Allow",
  "Action": [
    "iotsitewise:CreateAsset",
    "iotsitewise:CreateAssetModel",
    "iotsitewise:UpdateAsset",
    "iotsitewise:UpdateAssetModel",
    "iotsitewise:UpdateAssetProperty",
    "iotsitewise:ListAssets",
    "iotsitewise:ListAssetModels",
    "iotsitewise:ListAssetProperties",
    "iotsitewise:ListAssetModelProperties",
    "iotsitewise:ListAssociatedAssets",
    "iotsitewise:DescribeAsset",
    "iotsitewise:DescribeAssetModel",
    "iotsitewise:DescribeAssetProperty",
    "iotsitewise:AssociateAssets",
    "iotsitewise:DisassociateAssets",
    "iotsitewise:AssociateTimeSeriesToAssetProperty",
    "iotsitewise:DisassociateTimeSeriesFromAssetProperty",
    "iotsitewise:BatchPutAssetPropertyValue",
    "iotsitewise:BatchGetAssetPropertyValue",
    "iotsitewise:TagResource",
    "iotsitewise:UntagResource",
    "iotsitewise:ListTagsForResource",
    "iotsitewise:CreateAssetModelCompositeModel",
    "iotsitewise:UpdateAssetModelCompositeModel",
    "iotsitewise:DescribeAssetModelCompositeModel",
    "iotsitewise>DeleteAssetModelCompositeModel",
    "iotsitewise:ListAssetModelCompositeModels",
    "iotsitewise:ListCompositionRelationships",
    "iotsitewise:DescribeAssetCompositeModel"
  ],
  "Resource": "*"
}
```

AWS IoT TwinMaker 정책

이 정책은 대량 작업을 수행하는 데 사용하는 AWS IoT TwinMaker API 작업에 대한 액세스를 허용합니다.

```
{
```

```

    "Sid": "MetadataTransferJobApiAccess",
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:CreateMetadataTransferJob",
        "iottwinmaker:CancelMetadataTransferJob",
        "iottwinmaker:GetMetadataTransferJob",
        "iottwinmaker:ListMetadataTransferJobs"
    ],
    "Resource": "*"
}

```

Amazon S3 정책

이 정책은 일괄 작업을 위한 메타데이터 전송을 위해 Amazon S3 버킷에 대한 액세스를 제공합니다.

For a specific Amazon S3 bucket

일괄 작업 메타데이터 작업에 특정 버킷 하나를 사용하는 경우 이 정책은 해당 버킷에 대한 액세스를 제공합니다.

```

{
    "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

여러 다른 버킷을 사용하여 일괄 작업 메타데이터 작업을 하는 경우 이 정책은 모든 버킷에 대한 액세스를 제공합니다.

```

{

```

```

    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": "*"
}

```

가져오기 및 내보내기 작업 문제 해결에 대한 자세한 내용은 [일괄 가져오기 및 내보내기 문제 해결](#) 섹션을 참조하세요.

일괄 가져오기 작업 실행

대량 가져오기는 메타데이터를 AWS IoT SiteWise 워크스페이스로 이동하는 작업입니다. 예를 들어 대량 가져오기는 로컬 파일 또는 Amazon S3 버킷의 파일에서 AWS IoT SiteWise 워크스페이스로 메타데이터를 이동할 수 있습니다.

1단계: 가져올 파일 준비

AWS IoT SiteWise 네이티브 형식 파일을 다운로드하여 자산과 자산 모델을 가져옵니다. 자세한 내용은 [AWS IoT SiteWise 메타데이터 전송 작업 스키마](#) 섹션을 참조하세요.

2단계: 준비된 파일을 Amazon S3에 업로드

Amazon S3로 파일을 업로드합니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 [Amazon S3에 파일 업로드](#)를 참조하세요.

메타데이터 가져오기(콘솔)

를 사용하여 메타데이터 AWS IoT SiteWise Console 를 대량으로 가져올 수 있습니다. [1단계: 가져올 파일 준비](#) 및 [2단계: 준비된 파일을 Amazon S3에 업로드](#)를 따라 가져올 준비가 된 파일을 준비합니다.

Amazon S3에서 로 데이터 가져오기 AWS IoT SiteWise Console

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.

2. 탐색 창에서 일괄 작업 새로 만들기를 선택합니다.
3. 새 가져오기를 선택하여 가져오기 프로세스를 시작합니다.
4. 메타데이터 가져오기 페이지에서 다음을 수행합니다.
 - Amazon S3 버킷 및 파일을 보려면 Amazon S3 찾아보기를 선택합니다.
 - 준비된 가져오기 파일이 포함된 Amazon S3 버킷으로 이동합니다.
 - 가져올 파일을 선택합니다.
 - 선택한 파일을 검토하고 가져오기를 선택합니다.
5. 의 SiteWise 메타데이터에 대한 대량 작업 페이지에는 새로 생성된 가져오기 작업이 작업 진행률 테이블에 AWS IoT SiteWise Console 표시됩니다.

메타데이터 가져오기(AWS CLI)

가져오기 작업을 수행하려면 다음 절차를 따르세요.

Amazon S3에서 로 데이터 가져오기 AWS CLI

1. [AWS IoT SiteWise 메타데이터 전송 작업 스키마](#)에 따라 가져오려는 리소스를 지정하는 메타데이터 파일을 생성합니다. Amazon S3 버킷에 이 파일을 저장합니다.

가져올 메타데이터 파일의 예시는 [메타데이터 가져오기 예시](#) 섹션을 참조하세요.

2. 이제 요청 본문이 포함된 JSON 파일을 생성합니다. 요청 본문은 전송 작업의 소스와 대상을 지정합니다. 이 파일은 이전 단계의 파일과는 별개입니다. Amazon S3 버킷을 소스로, iotsitewise를 대상으로 지정해야 합니다.

다음 예시는 요청 본문을 보여 줍니다.

```
{
  "metadataTransferJobId": "your-transfer-job-Id",
  "sources": [{
    "type": "s3",
    "s3Configuration": {
      "location": "arn:aws:s3:::amzn-s3-demo-bucket/"
    }
  }],
  "destination": {
    "type": "iotsitewise"
  }
}
```


}

3. 다음 AWS CLI 명령을 실행 `CreateMetadataTransferJob` 하여 결과를 호출합니다. 이 예시에서는 이전 단계의 요청 본문 파일 이름이 `createMetadataTransferJobExport.json`입니다.

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
  --cli-input-json file://createMetadataTransferJobImport.json
```

이렇게 하면 메타데이터 전송 작업이 생성되고 선택한 리소스 전송 프로세스가 시작됩니다.

일괄 내보내기 작업 실행

대량 내보내는 AWS IoT SiteWise 워크스페이스에서 Amazon S3 버킷으로 메타데이터를 이동하는 작업입니다.

Amazon S3로 AWS IoT SiteWise 콘텐츠를 대량으로 내보낼 때 내보낼 특정 자산 모델 및 자산을 제한하는 필터를 지정할 수 있습니다.

필터는 JSON 요청의 소스 섹션 내 `iotSiteWiseConfiguration` 섹션에 지정되어야 합니다.

Note

요청에 여러 필터를 포함할 수 있습니다. 일괄 작업은 어느 필터와든 일치하는 자산 모델과 자산을 내보냅니다.

필터를 제공하지 않으면 일괄 작업이 모든 자산 모델과 자산을 내보냅니다.

Example 필터가 있는 요청 본문

```
{
  "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"
  }
}
}
}

```

메타데이터 내보내기(콘솔)

다음 절차에서는 콘솔 내보내기 작업을 설명합니다.

에서 내보내기 작업 생성 AWS IoT SiteWise Console

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 일괄 작업 새로 만들기를 선택합니다.
3. 새 내보내기를 선택하여 내보내기 프로세스를 시작합니다.
4. 메타데이터 내보내기 페이지에서 다음을 수행합니다.
 - 내보내기 작업의 이름을 입력합니다. Amazon S3 버킷의 내보낸 파일에 사용되는 이름입니다.
 - 내보낼 리소스를 선택하면 작업에 대한 필터가 설정됩니다.
 - 모든 자산 및 자산 모델을 내보냅니다. 자산 및 자산 모델에 필터를 사용합니다.
 - 자산을 내보냅니다. 자산을 필터링합니다.

- 내보내기 필터에 사용할 자산을 선택합니다.
 - (선택 사항) 파생 자산 또는 관련 자산 모델을 추가합니다.
 - 자산 모델을 내보냅니다. 자산 모델을 필터링합니다.
 - 내보내기 필터에 사용할 자산 모델을 선택합니다.
 - (선택 사항) 파생 자산, 관련 자산 또는 둘 다를 추가합니다.
 - 다음을 선택합니다.
 - Amazon S3 버킷으로 이동합니다.
 - Amazon S3 버킷 및 파일을 보려면 Amazon S3 찾아보기를 선택합니다.
 - 파일을 배치해야 하는 Amazon S3 버킷으로 이동합니다.
 - 다음을 선택합니다.
 - 내보내기 작업을 검토하고 내보내기를 선택합니다.
5. 의 SiteWise 메타데이터에 대한 대량 작업 페이지에는 새로 생성된 가져오기 작업이 작업 진행률 테이블에 AWS IoT SiteWise Console 표시됩니다.

메타데이터를 내보낼 때 필터를 사용하는 다양한 방법은 [메타데이터 예시 내보내기](#) 섹션을 참조하세요.

메타데이터 내보내기(AWS CLI)

다음 절차에서는 AWS CLI 내보내기 작업을 설명합니다.

에서 Amazon S3 AWS IoT SiteWise 로 데이터 내보내기

1. 요청 본문이 포함된 JSON 파일을 생성합니다. 요청 본문은 전송 작업의 소스와 대상을 지정합니다. 다음 예시는 요청 본문 예시를 보여 줍니다.

```
{
  "metadataTransferJobId": "your-transfer-job-Id",
  "sources": [{
    "type": "iotsitewise"
  }],
  "destination": {
    "type": "s3",
    "s3Configuration": {
      "location": "arn:aws:s3:::amzn-s3-demo-bucket"
    }
  }
}
```

```
}
}
```

Amazon S3 버킷을 메타데이터 전송 작업의 대상으로 지정해야 합니다.

Note

이 예시에서는 모든 자산 모델과 자산을 내보냅니다. 특정 자산 모델 또는 자산으로 내보내기를 제한하려면 요청 본문에 필터를 포함할 수 있습니다. 내보내기 필터 적용에 대한 자세한 내용은 [메타데이터 예시 내보내기](#) 섹션을 참조하세요.

2. 다음 단계에서 사용할 요청 본문 파일을 저장합니다. 이 예시에서 파일의 이름은 `createMetadataTransferJobExport.json`으로 지정됩니다.
3. 다음 AWS CLI 명령을 실행 `CreateMetadataTransferJob`하여를 호출합니다.

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
    --cli-input-json file://createMetadataTransferJobExport.json
```

입력 JSON 파일을 고유한 전송 파일 이름이 포함된 `createMetadataTransferJobExport.json`으로 바꿉니다.

작업 진행 상황 추적 및 오류 처리

일괄 프로세스 작업은 처리하는 데 시간이 걸립니다. 각 작업은 요청을 AWS IoT SiteWise 수신하는 순서대로 처리됩니다. 각 계정에 대해 한 번에 하나씩 처리됩니다. 작업이 완료되면 대기열의 다음가 자동으로 처리를 시작합니다.는 작업을 비동기적으로 AWS IoT SiteWise 해결하고 진행 상황에 따라 각각의 상태를 업데이트합니다. 각 작업에는 리소스 상태 및 오류 메시지(해당되는 경우)가 포함된 상태 필드가 있습니다.

상태는 다음 값 중 하나일 수 있습니다.

- **VALIDATING** - 제출된 파일 형식 및 내용을 포함한 작업을 검증하는 중입니다.
- **PENDING** - 작업이 대기열에 있습니다. AWS IoT SiteWise 콘솔에서 이 상태의 작업을 취소할 수 있지만 다른 모든 상태는 종료될 때까지 계속됩니다.
- **RUNNING** - 작업을 처리하는 중입니다. 가져오기 파일에 정의된 대로 리소스를 생성 및 업데이트하거나 선택한 내보내기 작업 필터를 기반으로 리소스를 내보내는 중입니다. 취소하면 이 작업에서 가

저은 리소스는 삭제되지 않습니다. 자세한 정보는 [작업 진행 상황 및 세부 정보 검토\(콘솔\)](#)을 참조하세요.

- CANCELLING - 작업이 적극적으로 취소되고 있습니다.
- ERROR - 하나 이상의 리소스를 처리하지 못했습니다. 자세한 내용은 세부 작업 보고서를 참조하세요. 자세한 정보는 [오류 세부 정보 검사\(콘솔\)](#)을 참조하세요.
- COMPLETED - 오류 없이 작업이 완료되었습니다.
- CANCELLED - 작업이 취소되고 대기열에 추가되지 않습니다. RUNNING 작업을 취소한 경우 취소 시 이 작업에서 이미 가져온 리소스는 삭제되지 않습니다 AWS IoT SiteWise.

주제

- [작업 진행 상황 추적](#)
- [오류 검사 AWS IoT SiteWise](#)

작업 진행 상황 추적

작업 진행 상황 및 세부 정보 검토(콘솔)

일괄 작업을 시작하려면 [메타데이터 가져오기\(콘솔\)](#) 또는 [메타데이터 내보내기\(콘솔\)](#) 섹션을 참조하세요.

AWS IoT SiteWise 콘솔의 작업 진행 상황 개요:

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 일괄 작업 새로 만들기를 선택합니다.
3. AWS IoT SiteWise 콘솔의 작업 진행 상황 테이블에는 대량 작업 작업 목록이 표시됩니다.
4. 작업 유형 열은 내보내기 또는 가져오기 작업인지 설명합니다. 가져온 날짜 열에는 작업이 시작된 날짜가 표시됩니다.
5. 상태 열에는 작업 상태가 표시됩니다. 작업을 선택하여 작업에 대한 세부 정보를 볼 수 있습니다.
6. 선택한 작업은 성공 시 성공 또는 작업 실패 시 실패 목록을 표시합니다. 오류 설명도 각 리소스 유형과 함께 표시됩니다.

AWS IoT SiteWise 콘솔의 작업 세부 정보 개요:

AWS IoT SiteWise 콘솔의 작업 진행률 테이블에는 대량 작업 작업 목록이 표시됩니다.

1. 더 자세한 정보를 보려면 작업을 선택합니다.
2. 가져오기 작업의 경우 Data source ARN은 가져오기 파일의 Amazon S3 위치를 나타냅니다.
3. 내보내기 작업의 경우 Data destination ARN은 내보내기 후 파일의 Amazon S3 위치를 나타냅니다.
4. Status 및 Status reason은 현재 작업에 대한 추가 세부 정보를 제공합니다. 자세한 내용은 [작업 진행 상황 추적 및 오류 처리](#) 섹션을 참조하세요.
5. Queued position은 프로세스 대기열에서 작업의 위치를 나타냅니다. 작업은 한 번에 하나씩 처리됩니다. 대기열 위치 1은 작업이 다음으로 처리될 것임을 나타냅니다.
6. 작업 세부 정보 페이지에는 작업 진행 상황 개수도 표시됩니다.
 - 작업 진행 상황 수 유형은 다음과 같습니다.
 - i. Total resources - 전송 프로세스에 있는 총 자산 수를 나타냅니다.
 - ii. Succeeded - 프로세스 중에 성공적으로 전송된 자산 수를 나타냅니다.
 - iii. Failed - 프로세스 중에 실패한 자산 수를 나타냅니다.
 - iv. Skipped - 프로세스 중에 건너뛴 자산 수를 나타냅니다.
7. PENDING 또는 VALIDATING의 작업 상태는 모든 작업 진행 상황 수를 - 기호로 표시합니다. 이는 작업 진행 상황 수가 평가 중임을 나타냅니다.
8. RUNNING의 작업 상태는 처리를 위해 제출된 작업인 Total resources 수를 표시합니다. 세부 개수(Succeeded, Failed 및 Skipped)는 처리된 리소스에 적용됩니다. 작업 상태가 COMPLETED 또는 ERROR가 될 때까지 세부 개수의 합계는 Total resources 개수보다 작습니다.
9. 작업의 상태가 COMPLETED 또는 ERROR인 경우 Total resources 개수는 세부 개수(Succeeded, Failed 및 Skipped)의 합계와 같습니다.
10. 작업 상태가 ERROR인 경우 작업 실패 테이블에서 특정 오류 및 실패에 대한 세부 정보를 확인합니다. 자세한 내용은 [오류 세부 정보 검사\(콘솔\)](#) 섹션을 참조하세요.

작업 진행 상황 및 세부 정보 검토(AWS CLI)

일괄 작업을 시작한 후 다음 API 작업을 사용하여 상태를 확인하거나 업데이트할 수 있습니다.

- 특정 작업에 대한 정보를 검색하려면 [GetMetadataTransferJob](#) API 작업을 사용합니다.

GetMetadataTransferJob API를 사용하여 정보 검색:

1. 전송 작업 생성 및 실행 GetMetadataTransferJob API를 호출하세요.

Example AWS CLI 명령:

```
aws iottwinmaker get-metadata-transfer-job \
    --metadata-transfer-job-id your_metadata_transfer_job_id \
    --region your_region
```

2. GetMetadataTransferJob API는 다음 파라미터가 있는 MetadataTransferJobProgress 객체를 반환합니다.

- succeededCount - 프로세스에서 성공적으로 전송된 자산 수를 나타냅니다.
- failedCount - 프로세스 중에 실패한 자산 수를 나타냅니다.
- skippedCount - 프로세스 중에 건너뛴 자산 수를 나타냅니다.
- totalCount - 전송 프로세스의 총 자산 수를 나타냅니다.

이러한 파라미터는 작업 진행 상태를 나타냅니다. 상태가 RUNNING인 경우 아직 처리해야 할 리소스 수를 추적하는 데 도움이 됩니다.

스키마 검증 오류가 발생하거나 failedCount가 1보다 크거나 같으면 작업 진행 상태가 ERROR로 바뀝니다. 작업에 대한 전체 오류 보고서는 Amazon S3 버킷에 배치됩니다. 자세한 내용은 [오류 검사 AWS IoT SiteWise](#) 섹션을 참조하세요.

- 현재 작업을 나열하려면 [ListMetadataTransferJobs](#) API 작업을 사용합니다.

JSON 파일을 사용하여 반환된 작업을 현재 상태에 따라 필터링합니다. 다음 절차를 참조하세요.

1. 사용할 필터를 지정하려면 AWS CLI 입력 JSON 파일을 생성합니다.는 다음을 사용합니다.

```
{
  "sourceType": "s3",
  "destinationType": "iottwinmaker",
  "filters": [{
    "state": "COMPLETED"
  }]
}
```

유효한 state 값 목록은 AWS IoT TwinMaker API 참조 가이드의 [ListMetadataTransferJobsFilter](#)를 참조하세요.

2. 다음 AWS CLI 예제 명령에서 JSON 파일을 인수로 사용합니다.

```
aws iottwinmaker list-metadata-transfer-job --region your_region \
  --cli-input-json file://ListMetadataTransferJobsExample.json
```

- 작업을 취소하려면 [CancelMetadataTransferJob](#) API 작업을 사용합니다. 이 API는 이미 내보내거나 가져온 리소스에 영향을 주지 않고 특정 메타데이터 전송 작업을 취소합니다.

```
aws iottwinmaker cancel-metadata-transfer-job \
  --region your_region \
  --metadata-transfer-job-id job-to-cancel-id
```

오류 검사 AWS IoT SiteWise

오류 세부 정보 검사(콘솔)

AWS IoT SiteWise 콘솔의 오류 세부 정보:

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 대량 작업 목록은의 작업 진행률 테이블 AWS IoT SiteWise Console 을 참조하세요.
3. 작업을 선택하여 작업 세부 정보를 봅니다.
4. 작업의 상태가 COMPLETED 또는 ERROR인 경우 Total resources 개수는 세부 개수 (Succeeded, Failed 및 Skipped)의 합계와 같습니다.
5. 작업 상태가 ERROR인 경우 작업 실패 테이블에서 특정 오류 및 실패에 대한 세부 정보를 확인합니다.
6. 작업 실패 테이블에는 작업 보고서의 내용이 표시됩니다. Resource type 필드는 다음과 같은 오류 또는 실패의 위치를 나타냅니다.
 - 예를 들어 Resource type 필드의 Bulk operations template에서 발생한 유효성 검사 오류는 가져오기 템플릿과 메타데이터 스키마 파일 형식이 일치하지 않음을 나타냅니다. 자세한 정보는 [AWS IoT SiteWise 메타데이터 전송 작업 스키마](#)을 참조하세요.
 - Resource type 필드의 Asset 실패는 다른 자산과의 충돌로 인해 자산이 생성되지 않았음을 나타냅니다. AWS IoT SiteWise 리소스 [오류 및 충돌에 대한 자세한 내용은 일반](#) 오류를 참조하세요.

오류 세부 정보 검사(AWS CLI)

전송 작업 중에 발생한 오류를 처리하고 진단하려면 GetMetadataTransferJob API 작업 사용에 대한 다음 절차를 참조하세요.

1. 전송 작업을 생성하고 실행한 후 [GetMetadataTransferJob](#)을 호출합니다.

```
aws iottwinmaker get-metadata-transfer-job \
    --metadata-transfer-job-id your_metadata_transfer_job_id \
    --region us-east-1
```

2. 작업 상태가 COMPLETED로 전환되는 것을 확인하면 작업 결과 확인을 시작할 수 있습니다.
3. GetMetadataTransferJob을 호출하면 [MetadataTransferJobProgress](#)라는 객체가 반환됩니다.

MetadataTransferJobProgress 객체는 다음 파라미터를 포함합니다.

- failedCount: 전송 프로세스 중에 실패한 자산 수를 나타냅니다.
- skippedCount: 전송 프로세스 중에 건너뛴 자산 수를 나타냅니다.
- succeededCount: 전송 프로세스 중에 성공한 자산 수를 나타냅니다.
- totalCount: 전송 프로세스와 관련된 총 자산 수를 나타냅니다.

4. 또한 API 직접 호출은 미리 서명된 URL이 포함된 reportUrl 요소를 반환합니다. 전송 작업에 추가 조사가 필요한 문제가 있는 경우 이 URL을 참조하세요.

메타데이터 가져오기 예시

이 섹션에서는 단일 일괄 가져오기 작업으로 자산 모델 및 자산을 가져올 메타데이터 파일을 생성하는 방법을 보여 줍니다.

일괄 가져오기의 예

단일 일괄 가져오기 작업으로 많은 자산 모델과 자산을 가져올 수 있습니다. 다음 예시에서는 이를 위해 메타데이터 파일을 생성하는 방법을 보여 줍니다.

이 예시 시나리오에는 작업 셀에 산업 로봇이 포함된 다양한 작업 현장이 있습니다.

이 예시에서는 다음의 두 가지 자산 모델을 정의합니다.

- RobotModel1: 이 자산 모델은 작업 현장에 있는 특정 유형의 로봇을 나타냅니다. 로봇에는 측정 속성인 Temperature가 있습니다.

- **WorkCell**: 이 자산 모델은 작업 현장 중 하나에 있는 로봇 모음을 나타냅니다. 자산 모델은 작업 셀에 로봇이 포함된 관계를 나타내기 위해 계층 구조 `robotHierarchy0EM1`를 정의합니다.

이 예시에서는 일부 자산도 정의합니다.

- **WorkCell1**: Boston 현장 내의 작업 셀
- **RobotArm123456**: 해당 작업 셀 내의 로봇
- **RobotArm987654**: 해당 작업 셀 내의 다른 로봇

다음 JSON 메타데이터 파일은 이러한 자산 모델 및 자산을 정의합니다. 이 메타데이터로 대량 가져오기를 실행하면 계층적 관계를 AWS IoT SiteWise 포함하여 자산 모델과 자산이 생성됩니다.

가져올 메타데이터 파일

```
{
  "assetModels": [
    {
      "assetModelExternalId": "Robot.0EM1.3536",
      "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"
            }
        ]
    }
]
}

```

모델 및 자산의 초기 온보딩 예시

이 예시 시나리오에서는 회사에 산업 로봇이 포함된 다양한 작업 현장이 있습니다.

이 예시는 다음의 여러 자산 모델을 정의합니다.

- **Sample_Enterprise** - 이 자산 모델은 현장이 속한 회사를 나타냅니다. 자산 모델은 계층 구조인 Enterprise to Site를 정의하여 현장과 엔터프라이즈의 관계를 나타냅니다.
- **Sample_Site** - 이 자산 모델은 회사 내 제조 현장을 나타냅니다. 자산 모델은 계층 구조인 Site to Line을 정의하여 현장과 라인의 관계를 나타냅니다.
- **Sample_Welding Line** - 이 자산 모델은 작업 현장 내의 조립 라인을 나타냅니다. 자산 모델은 계층 구조인 Line to Robot을 정의하여 사이트와 라인의 관계를 나타냅니다.
- **Sample_Welding Robot** - 이 자산 모델은 작업 현장에 있는 특정 유형의 로봇을 나타냅니다.

이 예시에서는 또한 자산 모델을 기반으로 자산을 정의합니다.

- **Sample_AnyCompany Motor** - 이 자산은 Sample_Enterprise 자산 모델에서 생성됩니다.
- **Sample_Chicago** - 이 자산은 Sample_Site 자산 모델에서 생성됩니다.
- **Sample_Welding Line 1** - 이 자산은 Sample_Welding Line 자산 모델에서 생성됩니다.
- **Sample_Welding Robot 1** - 이 자산은 Sample_Welding Robot 자산 모델에서 생성됩니다.
- **Sample_Welding Robot 2** - 이 자산은 Sample_Welding Robot 자산 모델에서 생성됩니다.

다음 JSON 메타데이터 파일은 이러한 자산 모델 및 자산을 정의합니다. 이 메타데이터로 대량 가져오기를 실행하면 계층적 관계를 AWS IoT SiteWise 포함하여 자산 모델과 자산이 생성됩니다.

가져올 자산 및 모델을 온보딩하기 위한 JSON 파일

```
{
  "assetModels": [
    {
      "assetModelExternalId": "External_Id_Welding_Robot",
      "assetModelName": "Sample_Welding Robot",
      "assetModelProperties": [
        {
          "dataType": "STRING",
          "externalId": "External_Id_Welding_Robot_Serial_Number",
          "name": "Serial Number",
          "type": {
            "attribute": {
              "defaultValue": "-"
            }
          },
          "unit": "-"
        }
      ]
    }
  ]
}
```

```

    },
    {
      "dataType": "DOUBLE",
      "externalId": "External_Id_Welding_Robot_Cycle_Count",
      "name": "CycleCount",
      "type": {
        "measurement": {}
      },
    },
    "unit": "EA"
  },
  {
    "dataType": "DOUBLE",
    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
    "name": "Joint 1 Current",
    "type": {
      "measurement": {}
    },
  },
  "unit": "Amps"
},
{
  "dataType": "DOUBLE",
  "externalId": "External_Id_Welding_Robot_Joint_1_Max_Current",
  "name": "Max Joint 1 Current",
  "type": {
    "metric": {
      "expression": "max(joint1current)",
      "variables": [
        {
          "name": "joint1current",
          "value": {
            "propertyExternalId":
"External_Id_Welding_Robot_Joint_1_Current"
          }
        }
      ],
      "window": {
        "tumbling": {
          "interval": "5m"
        }
      }
    }
  },
  "unit": "Amps"
}
}

```

```

    ]
  },
  {
    "assetModelExternalId": "External_Id_Welding_Line",
    "assetModelName": "Sample_Welding Line",
    "assetModelProperties": [
      {
        "dataType": "DOUBLE",
        "externalId": "External_Id_Welding_Line_Availability",
        "name": "Availability",
        "type": {
          "measurement": {}
        },
        "unit": "%"
      }
    ],
    "assetModelHierarchies": [
      {
        "externalId": "External_Id_Welding_Line_T0_Robot",
        "name": "Line to Robot",
        "childAssetModelExternalId": "External_Id_Welding_Robot"
      }
    ]
  },
  {
    "assetModelExternalId": "External_Id_Site",
    "assetModelName": "Sample_Site",
    "assetModelProperties": [
      {
        "dataType": "STRING",
        "externalId": "External_Id_Site_Street_Address",
        "name": "Street Address",
        "type": {
          "attribute": {
            "defaultValue": "-"
          }
        },
        "unit": "-"
      }
    ],
    "assetModelHierarchies": [
      {
        "externalId": "External_Id_Site_T0_Line",
        "name": "Site to Line",

```

```

        "childAssetModelExternalId": "External_Id_Welding_Line"
    }
]
},
{
    "assetModelExternalId": "External_Id_Enterprise",
    "assetModelName": "Sample_Enterprise",
    "assetModelProperties": [
        {
            "dataType": "STRING",
            "name": "Company Name",
            "externalId": "External_Id_Enterprise_Company_Name",
            "type": {
                "attribute": {
                    "defaultValue": "-"
                }
            },
            "unit": "-"
        }
    ],
    "assetModelHierarchies": [
        {
            "externalId": "External_Id_Enterprise_T0_Site",
            "name": "Enterprise to Site",
            "childAssetModelExternalId": "External_Id_Site"
        }
    ]
}
],
"assets": [
    {
        "assetExternalId": "External_Id_Welding_Robot_1",
        "assetName": "Sample_Welding Robot 1",
        "assetModelExternalId": "External_Id_Welding_Robot",
        "assetProperties": [
            {
                "externalId": "External_Id_Welding_Robot_Serial_Number",
                "attributeValue": "S1000"
            },
            {
                "externalId": "External_Id_Welding_Robot_Cycle_Count",
                "alias": "AnyCompany/Chicago/Welding Line/S1000/Count"
            }
        ]
    }
]

```

```

        "externalId": "External_Id_Welding_Robot_Joint_1_Current",
        "alias": "AnyCompany/Chicago/Welding Line/S1000/1/Current"
    }
]
},
{
    "assetExternalId": "External_Id_Welding_Robot_2",
    "assetName": "Sample_Welding Robot 2",
    "assetModelExternalId": "External_Id_Welding_Robot",
    "assetProperties": [
        {
            "externalId": "External_Id_Welding_Robot_Serial_Number",
            "attributeValue": "S2000"
        },
        {
            "externalId": "External_Id_Welding_Robot_Cycle_Count",
            "alias": "AnyCompany/Chicago/Welding Line/S2000/Count"
        },
        {
            "externalId": "External_Id_Welding_Robot_Joint_1_Current",
            "alias": "AnyCompany/Chicago/Welding Line/S2000/1/Current"
        }
    ]
},
{
    "assetExternalId": "External_Id_Welding_Line_1",
    "assetName": "Sample_Welding Line 1",
    "assetModelExternalId": "External_Id_Welding_Line",
    "assetProperties": [
        {
            "externalId": "External_Id_Welding_Line_Availability",
            "alias": "AnyCompany/Chicago/Welding Line/Availability"
        }
    ],
    "assetHierarchies": [
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_1"
        },
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_2"
        }
    ]
}
]

```



```

    },
    {
      "assetExternalId": "External_Id_Site_Chicago",
      "assetName": "Sample_Chicago",
      "assetModelExternalId": "External_Id_Site",
      "assetHierarchies": [
        {
          "externalId": "External_Id_Site_T0_Line",
          "childAssetExternalId": "External_Id_Welding_Line_1"
        }
      ]
    },
  ],
  {
    "assetExternalId": "External_Id_Enterprise_AnyCompany",
    "assetName": "Sample_AnyEnterprise Motor",
    "assetModelExternalId": "External_Id_Enterprise",
    "assetHierarchies": [
      {
        "externalId": "External_Id_Enterprise_T0_Site",
        "childAssetExternalId": "External_Id_Site_Chicago"
      }
    ]
  }
]
}

```

다음 스크린샷은 이전 코드 예제를 실행한 AWS IoT SiteWise Console 후에 표시되는 모델의 스크린샷입니다.

IoT SiteWise > Models

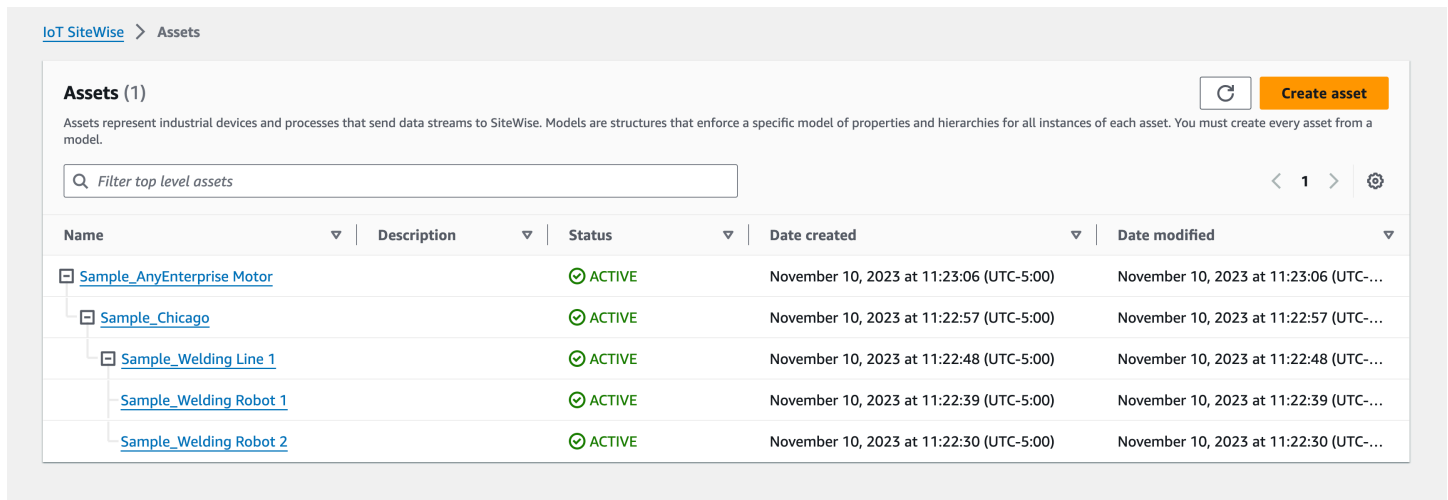
Models (4) ↻ Create component model Create asset model

Assets represent industrial devices and processes that send data streams to SiteWise. Models are structures that enforce a specific model of properties and hierarchies for all instances of each asset. You must create every asset from a model.

< 1 > ⚙

Name	Status	Model type	Date created	Date modified
Sample_Enterprise	🟢 ACTIVE	Asset model	November 10, 2023 at 11:22:13 (UT...)	November 10, 202...
Sample_Site	🟢 ACTIVE	Asset model	November 10, 2023 at 11:21:57 (UT...)	November 10, 202...
Sample_Welding Line	🟢 ACTIVE	Asset model	November 10, 2023 at 11:21:40 (UT...)	November 10, 202...
Sample_Welding Robot	🟢 ACTIVE	Asset model	November 10, 2023 at 11:21:24 (UT...)	November 10, 202...

다음 스크린샷은 이전 코드 예제를 실행한 AWS IoT SiteWise Console 후에 표시되는 모델, 자산 및 계층 구조입니다.



추가 자산 온보딩 예시

이 예시에서는 계정의 기존 자산 모델로 가져올 추가 자산을 정의합니다.

- Sample_Welding Line 2 – 이 자산은 Sample_Welding Line 자산 모델에서 생성됩니다.
- Sample_Welding Robot 3 – 이 자산은 Sample_Welding Robot 자산 모델에서 생성됩니다.
- Sample_Welding Robot 4 – 이 자산은 Sample_Welding Robot 자산 모델에서 생성됩니다.

이 예시의 초기 자산을 생성하려면 [모델 및 자산의 초기 온보딩 예시](#) 섹션을 참조하세요.

다음 JSON 메타데이터 파일은 이러한 자산 모델 및 자산을 정의합니다. 이 메타데이터로 대량 가져오기를 실행하면 계층적 관계를 AWS IoT SiteWise 포함하여 자산 모델과 자산이 생성됩니다.

추가 자산을 온보딩하기 위한 JSON 파일

```
{
  "assets": [
    {
      "assetExternalId": "External_Id_Welding_Robot_3",
      "assetName": "Sample_Welding Robot 3",
      "assetModelExternalId": "External_Id_Welding_Robot",
      "assetProperties": [
        {
          "externalId": "External_Id_Welding_Robot_Serial_Number",
          "attributeValue": "S3000"
        }
      ]
    }
  ]
}
```

```

        },
        {
            "externalId": "External_Id_Welding_Robot_Cycle_Count",
            "alias": "AnyCompany/Chicago/Welding Line/S3000/Count"
        },
        {
            "externalId": "External_Id_Welding_Robot_Joint_1_Current",
            "alias": "AnyCompany/Chicago/Welding Line/S3000/1/Current"
        }
    ]
},
{
    "assetExternalId": "External_Id_Welding_Robot_4",
    "assetName": "Sample_Welding Robot 4",
    "assetModelExternalId": "External_Id_Welding_Robot",
    "assetProperties": [
        {
            "externalId": "External_Id_Welding_Robot_Serial_Number",
            "attributeValue": "S4000"
        },
        {
            "externalId": "External_Id_Welding_Robot_Cycle_Count",
            "alias": "AnyCompany/Chicago/Welding Line/S4000/Count"
        },
        {
            "externalId": "External_Id_Welding_Robot_Joint_1_Current",
            "alias": "AnyCompany/Chicago/Welding Line/S4000/1/Current"
        }
    ]
},
{
    "assetExternalId": "External_Id_Welding_Line_1",
    "assetName": "Sample_Welding Line 1",
    "assetModelExternalId": "External_Id_Welding_Line",
    "assetHierarchies": [
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_1"
        },
        {
            "externalId": "External_Id_Welding_Line_T0_Robot",
            "childAssetExternalId": "External_Id_Welding_Robot_2"
        }
    ]
}

```

```

        "externalId": "External_Id_Welding_Line_T0_Robot",
        "childAssetExternalId": "External_Id_Welding_Robot_3"
      }
    ],
  },
  {
    "assetExternalId": "External_Id_Welding_Line_2",
    "assetName": "Sample_Welding Line 2",
    "assetModelExternalId": "External_Id_Welding_Line",
    "assetHierarchies": [
      {
        "externalId": "External_Id_Welding_Line_T0_Robot",
        "childAssetExternalId": "External_Id_Welding_Robot_4"
      }
    ]
  },
  {
    "assetExternalId": "External_Id_Site_Chicago",
    "assetName": "Sample_Chicago",
    "assetModelExternalId": "External_Id_Site",
    "assetHierarchies": [
      {
        "externalId": "External_Id_Site_T0_Line",
        "childAssetExternalId": "External_Id_Welding_Line_1"
      },
      {
        "externalId": "External_Id_Site_T0_Line",
        "childAssetExternalId": "External_Id_Welding_Line_2"
      }
    ]
  }
]
}

```

다음 스크린샷은 이전 코드 예제를 실행한 AWS IoT SiteWise Console 후에 표시되는 모델, 자산 및 계층 구조입니다.

IoT SiteWise > Assets

Assets (1) ↻ Create asset

Assets represent industrial devices and processes that send data streams to SiteWise. Models are structures that enforce a specific model of properties and hierarchies for all instances of each asset. You must create every asset from a model.

🔍 Filter top level assets < 1 > ⚙️

Name	Description	Status	Date created	Date modified
Sample_AnyCompany Motor		✔️ ACTIVE	November 09, 2023 at 19:18:05 (UTC-5:00)	November 09, 2023 at 19:18:05 (UTC-5:00)
Sample_Chicago		✔️ ACTIVE	November 09, 2023 at 19:17:56 (UTC-5:00)	November 09, 2023 at 19:17:56 (UTC-5:00)
Sample_Welding Line 1		✔️ ACTIVE	November 09, 2023 at 19:17:48 (UTC-5:00)	November 09, 2023 at 19:17:48 (UTC-5:00)
Sample_Welding Robot 2		✔️ ACTIVE	November 09, 2023 at 19:17:39 (UTC-5:00)	November 09, 2023 at 19:51:05 (UTC-5:00)
Sample_Welding Robot 3		✔️ ACTIVE	November 09, 2023 at 20:40:02 (UTC-5:00)	November 09, 2023 at 20:40:02 (UTC-5:00)
Sample_Welding Robot 1		✔️ ACTIVE	November 09, 2023 at 19:17:30 (UTC-5:00)	November 09, 2023 at 19:51:05 (UTC-5:00)
Sample_Welding Line 2		✔️ ACTIVE	November 09, 2023 at 20:40:20 (UTC-5:00)	November 09, 2023 at 20:40:20 (UTC-5:00)
Sample_Welding Robot 4		✔️ ACTIVE	November 09, 2023 at 20:40:11 (UTC-5:00)	November 09, 2023 at 20:40:11 (UTC-5:00)

새 속성 온보딩 예시

이 예시에서는 기존 자산 모델의 새 속성을 정의합니다. 추가 자산 및 모델을 온보딩하려면 [추가 자산 온보딩 예시](#) 섹션을 참조하세요.

- Joint 1 Temperature - 이 속성은 Sample_Welding Robot 자산 모델에 추가됩니다. 이 새 속성은 Sample_Welding Robot 자산 모델에서 생성된 각 자산에도 전파됩니다.

기존 자산 모델에 새 속성을 추가하려면 다음 JSON 메타데이터 파일 예시를 참조하세요. JSON에 표시된 것처럼 기존 Sample_Welding Robot 자산 모델 정의 전체를 새 속성과 함께 제공해야 합니다. 기존 정의의 전체 속성 목록이 제공되지 않으면 생략된 속성이 AWS IoT SiteWise 삭제됩니다.

새 속성을 온보딩하기 위한 JSON 파일

이 예시에서는 Joint 1 Temperature 자산 모델에 새 속성을 추가합니다.

```
{
  "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"
}
]
}
}
}

```

데이터 스트림 관리 예시

이 예시에서는 자산 속성과 연결된 데이터 스트림을 관리하는 두 가지 방법을 보여 줍니다. 자산 속성 별칭의 이름을 바꿀 때는 자산 속성의 데이터 스트림에 현재 저장된 기록 데이터에 대한 두 가지 옵션이 있습니다.

- 옵션 1 - 현재 데이터 스트림을 유지하고 별칭만 이름을 바꿔 새 별칭으로 기록 데이터에 액세스할 수 있도록 합니다.

JSON 메타데이터 파일 예시에서 ID가 External_Id_Welding_Robot_Cycle_Count인 자산 속성은 별칭이 AnyCompany/Chicago/Welding Line/S3000/Count-Updated로 변경됩니다. 이 자산 속성의 기록 데이터는 이 변경 후에도 동일하게 유지됩니다.

- 옵션 2 - 새 별칭으로 액세스할 수 있는 자산 속성에 새 데이터 스트림을 할당합니다. 이전 데이터 스트림은 과거 데이터와 함께 이전 별칭으로도 액세스할 수 있지만 자산 속성과는 연결되지 않습니다.

JSON 메타데이터 파일 예시에서 ID가 External_Id_Welding_Robot_Joint_1_Current인 자산 속성은 별칭이 AnyCompany/Chicago/Welding Line/S4999/1/Current로 변경됩니다. 이번에는 추가 값 retainDataOnAliasChange가 존재하고 False로 설정됩니다. 이 설정을 사용하

면 원본 데이터 스트림이 자산 속성과 연결 해제되고 기록 데이터가 포함되지 않은 새 데이터 스트림이 생성됩니다.

원래 기록 데이터를 사용하여 이전 데이터 스트림에 액세스하려면에서 데이터 스트림 페이지로 AWS Console Home이동하여 이전 별칭을 검색합니다AnyCompany/Chicago/Welding Line/S3000/1/Current.

속성 별칭을 업데이트하기 위한 JSON 파일

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

메타데이터 예시 내보내기

Amazon S3로 AWS IoT SiteWise 콘텐츠를 대량으로 내보낼 때 내보낼 특정 자산 모델 및 자산을 제한하는 필터를 지정할 수 있습니다.

요청 본문의 sources 섹션에 있는 iotSiteWiseConfiguration 섹션에 필터를 지정합니다.

Note

필터는 여러 개 설정할 수 있습니다. 일괄 작업은 그 어느 필터와든 일치하는 모든 자산 모델 또는 자산을 내보냅니다.

필터를 제공하지 않으면 작업이 모든 자산 모델과 자산을 내보냅니다.

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

자산 모델별 필터링

특정 자산 모델을 필터링할 수 있습니다. 해당 모델을 사용하는 모든 자산 또는 계층 구조 내의 모든 자산 모델을 포함할 수도 있습니다. 자산과 계층 구조를 모두 포함할 수는 없습니다.

계층 구조에 대한 자세한 정보는 [자산 모델 계층 정의](#) 단원을 참조하세요.

Asset model

이 필터에는 다음의 지정된 자산 모델이 포함됩니다.

```
"filterByAssetModel": {
  "assetModelId": "asset model ID"
}
```

Asset model and its assets

이 필터에는 지정된 자산 모델과 해당 자산 모델을 사용하는 모든 자산이 포함됩니다.

```
"filterByAssetModel": {
  "assetModelId": "asset model ID",
  "includeAssets": true
}
```

Asset model and its hierarchy

이 필터에는 해당 계층 구조에 있는 모든 관련 자산 모델과 함께 지정된 자산 모델이 포함됩니다.

```
"filterByAssetModel": {
  "assetModelId": "asset model ID",
  "includeOffspring": true
}
```

자산별 필터링

특정 자산을 필터링할 수 있습니다. 자산 모델 또는 해당 계층 구조 내에 연결된 모든 자산을 포함할 수도 있습니다. 자산 모델과 계층 구조를 모두 포함할 수는 없습니다.

계층 구조에 대한 자세한 정보는 [자산 모델 계층 정의](#) 단원을 참조하세요.

Asset

이 필터에는 다음의 지정된 자산이 포함됩니다.

```
"filterByAsset": {
  "assetId": "asset ID"
}
```

Asset and its asset model

이 필터에는 다음의 지정된 자산과 함께 해당 자산이 사용하는 자산 모델이 포함됩니다.

```
"filterByAsset": {
  "assetId": "asset ID",
  "includeAssetModel": true
}
```

Asset and its hierarchy

이 필터에는 해당 계층 구조에 있는 모든 관련 자산과 함께 지정된 자산이 포함됩니다.

```
"filterByAsset": {
  "assetId": "asset ID",
  "includeOffspring": true
}
```

AWS IoT SiteWise 메타데이터 전송 작업 스키마

자체 대량 가져오기 및 내보내기 작업을 수행할 때 AWS IoT SiteWise 메타데이터 전송 작업 스키마를 참조용으로 사용합니다.

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "IoTSiteWise",
  "description": "Metadata transfer job resource schema for IoTSiteWise",
  "definitions": {
    "Name": {
      "type": "string",
      "minLength": 1,
      "maxLength": 256,
      "pattern": "^[^\\u0000-\\u001F\\u007F]+"
    },
    "Description": {
      "type": "string",
      "minLength": 1,
      "maxLength": 2048,
      "pattern": "^[^\\u0000-\\u001F\\u007F]+"
    },
    "ID": {
      "type": "string",
      "minLength": 36,
      "maxLength": 36,
      "pattern": "^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$"
    },
    "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"
    }
  }
}
```

에서 경보를 사용하여 데이터 모니터링 AWS IoT SiteWise

장비 또는 프로세스가 최적화되지 않았을 때 \팀원들에게 경고 알림을 보내도록 데이터에 대한 경보를 구성할 수 있습니다. 특정 메트릭 값이 상한 범위와 하한 범위 내에 있어야 컴퓨터 또는 프로세스 성능이 최적화됩니다. 메트릭이 작동 범위를 벗어나면 장비 운영자가 문제를 해결할 수 있도록 알림을 보내야 합니다. 경보를 사용하여 문제를 신속하게 식별하고 운영자에게 알려 장비 및 프로세스 성능을 극대화할 수 있습니다.

주제

- [경보 유형](#)
- [경보 상태](#)
- [경보 상태 속성](#)
- [에서 자산 모델에 대한 경보 정의 AWS IoT SiteWise](#)
- [에서 자산에 대한 경보 구성 AWS IoT SiteWise](#)
- [의 경보에 대응 AWS IoT SiteWise](#)
- [에서 외부 경보 상태 수집 AWS IoT SiteWise](#)

경보 유형

AWS 클라우드에서 감지하는 경보와 외부 프로세스에서 감지하는 경보를 정의할 수 있습니다.는 다음 유형의 경보를 AWS IoT SiteWise 지원합니다.

- AWS IoT Events 경보

Note

지원 종료 공지: 2026 AWS 년 5월 20일에에 대한 지원이 종료됩니다 AWS IoT Events. 2026년 5월 20일 이후에는 AWS IoT Events 콘솔 또는 AWS IoT Events 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 [AWS IoT Events 지원 종료를 참조하세요](#).

AWS IoT Events 경보는 in AWS IoT Events. AWS IoT SiteWise sends 자산 속성 값들의 경보 모델에 감지하는 경보입니다 AWS IoT Events. 그런 다음 경보 상태를 AWS IoT Events 로 보냅니다 AWS IoT SiteWise. 경보 감지 시기, 경보 상태 변경 시 알림 대상 등의 옵션을 구성할 수 있습니다. 경보 상태가 변경될 때 발생하는 [AWS IoT Events 작업](#)을 정의할 수도 있습니다.

의 경보 AWS IoT Events 는 경보 모델의 인스턴스입니다. 경보 모델은 경보의 임계값 및 심각도, 경보 상태가 변경될 때 수행할 조치 등을 지정합니다. 경보 모델의 각 특성을 구성할 때는 경보가 모니터링하는 자산 모델의 특성(attribute) 속성을 지정합니다. 자산 모델을 기반으로 하는 모든 자산은 경보의 해당 특성을 AWS IoT Events 평가할 때 속성 값을 사용합니다. 자세한 정보는 AWS IoT Events 개발자 안내서의 [경보 사용](#)을 참조하세요.

AWS IoT Events 경보가 상태를 변경할 때 경보에 응답할 수 있습니다. 예를 들어 경보가 활성화되면 경보를 확인하거나 일시 중지할 수 있습니다. 또한 경보를 활성화, 비활성화, 재설정할 수 있습니다.

SiteWise Monitor 사용자는 SiteWise Monitor 포털에서 AWS IoT Events 경보를 시각화, 구성 및 대응할 수 있습니다. 자세한 내용을 알아보려면 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [경보를 통한 모니터링](#)을 참조하세요.

Note

AWS IoT Events 이러한 경보를 평가하고 AWS IoT SiteWise 및 간에 데이터를 전송하는 데 요금이 부과됩니다 AWS IoT Events. 자세한 내용은 [AWS IoT Events 요금](#)을 참조하십시오.

• 외부 경보

외부 경보는 외부에서 평가하는 경보입니다 AWS IoT SiteWise. 경보 상태를 보고하는 데이터 소스가 있는 경우 외부 경보를 사용합니다. 외부 경보에는 경보 상태 데이터를 수집하는 데 사용하는 측정값 속성이 포함되어 있습니다.

상태가 바뀌면 외부 경보를 확인하거나 일시 중지할 수 없습니다.

SiteWise Monitor 사용자는 SiteWise Monitor 포털에서 외부 경보의 상태를 볼 수 있지만, 이러한 경보를 구성하거나 이에 응답할 수는 없습니다.

AWS IoT SiteWise 는 외부 경보의 상태를 평가하지 않습니다.

경보 상태

산업용 경보에는 모니터링하는 장비 또는 프로세스의 상태에 대한 정보와 경보 상태에 대한 작업자의 대응에 대한 정보(선택 사항)가 포함됩니다.

AWS IoT Events 경보를 정의할 때 승인 흐름을 활성화할지 여부를 지정합니다. 확인 흐름은 기본적으로 활성화되어 있습니다. 이 옵션을 활성화하면 작업자는 경보를 확인하고 경보에 대한 세부 정보 또는

경보를 해결하기 위해 취한 조치를 메모로 남길 수 있습니다. 작업자가 활성 경보가 비활성 상태가 되기 전에 확인하지 않으면 경보가 래치 상태가 됩니다. 래치 상태는 경보가 활성화되었지만 인식되지 않았음을 나타내므로 작업자는 장비 또는 프로세스를 점검하고 래치 경보를 확인해야 합니다.

경보의 상태는 다음과 같습니다:

- 정상(Normal) - 경보가 활성화되었지만 비활성 상태입니다. 산업 프로세스 또는 장비가 예상대로 작동합니다.
- 활성(Active) - 경보가 활성 상태입니다. 산업 공정 또는 장비가 작동 범위를 벗어났으므로 주의가 필요합니다.
- 확인됨(Acknowledged) - 작업자가 경보 상태를 확인했습니다.

이 상태는 확인 흐름을 활성화한 경보에만 적용됩니다.

- 래치 상태(Latched) - 경보가 정상으로 돌아왔지만 활성 상태였으며 작업자가 이를 확인하지 않았습니다. 산업 공정 또는 장비에서 경보를 정상으로 재설정하려면 작업자의 주의가 필요합니다.

이 상태는 확인 흐름을 활성화한 경보에만 적용됩니다.

- 일시 중지됨(SnoozeDisabled) - 작업자가 경보를 일시 중지했기 때문에 경보가 비활성화되었습니다. 운영자는 경보가 일시 정지되는 기간을 정의합니다. 이 기간이 지나면 경보는 정상 상태로 돌아옵니다.
- 비활성화됨(Disabled) - 경보가 비활성화되어 감지되지 않습니다.

경보 상태 속성

AWS IoT SiteWise 는 경보 상태 데이터를 문자열에 직렬화된 JSON 객체로 저장합니다. 이 객체에는 작업자 대응 조치, 경보가 평가하는 규칙 등 경보에 대한 상태 및 추가 정보가 포함됩니다.

이름 및 구조 유형 AWS/ALARM_STATE로 경보 상태 속성을 식별할 수 있습니다. 자세한 내용은 [에서 자산 모델에 대한 경보 정의 AWS IoT SiteWise](#) 단원을 참조하십시오.

경보 상태 데이터 객체에는 다음과 같은 정보가 포함되어 있습니다.

stateName

경보 상태 자세한 내용은 [경보 상태](#) 단원을 참조하십시오.

데이터 유형: STRING

customerAction

(선택 사항) 경보에 대한 작업자의 대응 정보가 들어 있는 객체입니다. 작업자는 경보를 활성화, 비활성화, 확인, 일시 중지할 수 있습니다. 이러한 작업을 하면 경보 상태 데이터에 응답과 응답 시 남길 수 있는 메모가 포함됩니다. 이 객체에는 다음 정보가 포함되어 있어야 합니다.

actionName

작업자가 경보에 대응하기 위해 취하는 조치의 이름입니다. 다음 스트링 중 하나를 포함합니다.

- ENABLE
- DISABLE
- SNOOZE
- ACKNOWLEDGE
- RESET

데이터 유형: STRING

enable

(선택 사항) 작업자가 경보를 활성화할 때 customerAction에 표시되는 객체입니다. 작업자가 경보를 활성화하면 경보 상태가 Normal로 바뀝니다. 이 객체에는 다음 정보가 포함되어 있어야 합니다.

note

(선택 사항) 고객이 경보를 활성화할 때 남기는 메모입니다.

데이터 유형: STRING

최대 길이: 128자

disable

(선택 사항) 작업자가 경보를 비활성화할 때 customerAction에 표시되는 객체입니다. 작업자가 경보를 비활성화하면 경보 상태가 Disabled로 바뀝니다. 이 객체에는 다음 정보가 포함되어 있어야 합니다.

note

(선택 사항) 고객이 경보를 비활성화할 때 남기는 메모입니다.

데이터 유형: STRING

최대 길이: 128자

acknowledge

(선택 사항) 작업자가 경보를 확인할 때 `customerAction`에 표시되는 객체입니다. 작업자가 경보를 활성화하면 경보 상태가 `Acknowledged`로 바뀝니다. 이 객체에는 다음 정보가 포함되어 있어야 합니다.

note

(선택 사항) 고객이 경보를 확인할 때 남기는 메모입니다.

데이터 유형: STRING

최대 길이: 128자

snooze

(선택 사항) 작업자가 경보를 일시 중지할 때 `customerAction`에 표시되는 객체입니다. 작업자가 경보를 활성화하면 경보 상태가 `SnoozeDisabled`로 바뀝니다. 이 객체에는 다음 정보가 포함되어 있어야 합니다.

snoozeDuration

작업자가 경보를 일시 중지하는 시간(초)입니다. 이 시간이 지나면 경보가 `Normal` 상태로 바뀝니다.

데이터 유형: INTEGER

note

(선택 사항) 고객이 경보를 일시 중지할 때 남기는 메모입니다.

데이터 유형: STRING

최대 길이: 128자

ruleEvaluation

(선택 사항) 경보를 평가하는 규칙에 대한 정보가 포함된 객체입니다. 이 객체에는 다음 정보가 포함되어 있어야 합니다.

simpleRule

비교 연산자를 사용하여 속성 값을 임계값과 비교하는 간단한 규칙에 대한 정보가 들어 있는 객체입니다. 이 객체에는 다음 정보가 포함되어 있어야 합니다.

inputProperty

경보에서 평가하는 속성의 값입니다.

데이터 유형: DOUBLE

operator

이 경보가 속성을 임계값과 비교하는 데 사용하는 비교 연산자입니다. 다음 스트링 중 하나를 포함합니다.

- < - 미만
- <= - 이하
- == - 같음
- != - 같지 않음
- >= - 이상
- > - 초과

데이터 유형: STRING

threshold

이 경보가 속성값과 비교하는 임계값입니다.

데이터 유형: DOUBLE

에서 자산 모델에 대한 경보 정의 AWS IoT SiteWise

자산 모델은 산업 데이터 및 경보의 표준화를 촉진합니다. 자산 모델에 경보 정의를 정의하여 자산 모델을 기반으로 모든 자산에 대한 경보를 표준화할 수 있습니다.

복합 자산 모델을 사용하여 이 자산 모델에 대한 경보를 정의할 수 있습니다. 복합 자산 모델은 다른 자산 모델의 특정 속성 집합을 표준화하는 자산 모델입니다. 복합 자산 모델은 자산 모델에 특정 속성이 존재하도록 합니다. 경보에는 유형, 상태, 소스 속성(선택 사항)이 있으므로, 경보 복합 모델은 이러한 속성이 존재하도록 강제합니다.

각 복합 자산 모델에는 복합 모델이 지원하는 속성을 정의하는 유형이 있습니다. 경보 복합 모델은 경보 유형, 경보 상태, 경보 소스(선택 사항)의 속성을 정의합니다. 복합 모델이 포함된 자산 모델에서 자산을 생성하는 경우, 자산에는 자산 모델에서 지정한 속성과 함께 복합 모델의 속성이 포함됩니다.

복합 모델의 각 속성에는 해당 복합 모델 유형을 식별하는 이름이 있어야 합니다. 복합 모델 속성은 복잡한 데이터 유형의 속성을 지원합니다. 이러한 속성에는 속성의 복합 데이터 유형을 지정하는 STRUCT 데이터 유형과 dataTypeSpec 특성이 있습니다. 복합 데이터 유형 속성에는 문자열로 직렬화된 JSON 데이터가 포함됩니다.

경보 복합 모델에는 다음과 같은 속성이 있습니다. 각 속성에는 이 유형의 복합 모델에 대해 이를 식별하는 이름이 있어야 합니다.

경보 유형

경보 유형. 다음 중 하나를 지정하세요.

- IOT_EVENTS - AWS IoT Events alarm. AWS IoT SiteWise sends 데이터를 AWS IoT Events에 보내어 경보의 상태를 평가합니다. 경보 소스 속성을 지정하여이 AWS IoT Events 경보 정의에 대한 경보 모델을 정의해야 합니다.

Note

지원 종료 공지: 2026 AWS 년 5월 20일에 대한 지원이 종료됩니다 AWS IoT Events. 2026년 5월 20일 이후에는 AWS IoT Events 콘솔 또는 AWS IoT Events 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 [AWS IoT Events 지원 종료를 참조하세요](#).

- EXTERNAL - 외부 경보입니다. 경보 상태를 측정값으로 수집합니다.

속성 이름: AWS/ALARM_TYPE

속성 유형: [attribute](#)

데이터 유형: STRING

경보 상태

경보 상태에 대한 시계열 데이터입니다. 경보에 대한 상태 및 기타 정보를 포함하는 문자열로 직렬화된 객체입니다. 자세한 내용은 [경보 상태 속성](#) 단원을 참조하십시오.

속성 이름: AWS/ALARM_STATE

속성 유형: [measurement](#)

데이터 유형: STRUCT

데이터 구조 유형: AWS/ALARM_STATE

경보 소스

(선택 사항) 경보 상태를 평가하는 리소스의 Amazon 리소스 이름(ARN)입니다. AWS IoT Events 경보의 경우 경보 모델의 ARN입니다.

속성 이름: AWS/ALARM_SOURCE

속성 유형: [attribute](#)

데이터 유형: STRING

Example 경보 복합 모델 예시

다음 자산 모델은 온도를 모니터링하는 경보가 있는 보일러를 나타냅니다. 온도 데이터를 로 AWS IoT SiteWise 전송 AWS IoT Events 하여 경보를 감지합니다.

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

```

주제

- [의 경보 알림 요구 사항 AWS IoT SiteWise](#)
- [에 대한 AWS IoT Events 경보 정의 AWS IoT SiteWise](#)
- [에서 외부 경보 정의 AWS IoT SiteWise](#)

의 경보 알림 요구 사항 AWS IoT SiteWise

AWS IoT Events 는 AWS 계정의 AWS Lambda 함수를 사용하여 경보 알림을 보냅니다. 경보 알림을 활성화하려면 경보와 동일한 AWS 리전에서 이 Lambda 함수를 생성해야 합니다. 이 Lambda 함수는 [Amazon Simple Notification Service\(SNS\)](#)를 사용하여 문자 알림을 보내고, [Amazon Simple Email](#)

[Service\(Amazon SES\)](#)를 사용하여 이메일 알림을 보냅니다. AWS IoT Events 경보를 생성할 때 경보가 알림을 보내는 데 사용하는 프로토콜과 설정을 구성합니다.

Note

지원 종료 공지: 2026 AWS 년 5월 20일에에 대한 지원이 종료됩니다 AWS IoT Events. 2026 년 5월 20일 이후에는 AWS IoT Events 콘솔 또는 AWS IoT Events 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 [AWS IoT Events 지원 종료를 참조하세요](#).

AWS IoT Events 는 계정에서이 Lambda 함수를 생성하는 데 사용할 수 있는 AWS CloudFormation 스택 템플릿을 제공합니다. 자세한 정보는 AWS IoT Events 개발자 안내서의 [Lambda 함수 오류](#)를 참조하세요.

에 대한 AWS IoT Events 경보 정의 AWS IoT SiteWise

Note

지원 종료 공지: 2026 AWS 년 5월 20일에에 대한 지원이 종료됩니다 AWS IoT Events. 2026 년 5월 20일 이후에는 AWS IoT Events 콘솔 또는 AWS IoT Events 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 [AWS IoT Events 지원 종료를 참조하세요](#).

AWS IoT Events 경보를 생성할 때는 자산 속성 값을 로 AWS IoT SiteWise 전송 AWS IoT Events 하여 경보의 상태를 평가합니다. AWS IoT Events 경보 정의에서 정의한 경보 모델에 따라 달라집니다 AWS IoT Events. 자산 모델에 AWS IoT Events 경보를 정의하려면 경보 모델을 AWS IoT Events 경보 소스 속성으로 지정하는 경보 복합 모델을 정의합니다.

AWS IoT Events 경보는 경보 임계값 및 경보 알림 설정과 같은 입력에 따라 달라집니다. 이러한 입력을 자산 모델의 속성(attribute)으로 정의합니다. 그런 다음 모델에 따라 각 자산에서 이러한 입력을 사용자 지정할 수 있습니다. AWS IoT SiteWise 콘솔에서 이러한 속성을 생성할 수 있습니다. AWS CLI 또는 API를 사용하여 경보를 정의하는 경우 자산 모델에서 이러한 속성을 수동으로 정의해야 합니다.

또한 사용자 지정 경보 알림 동작과 같이 경보가 감지될 때 발생하는 다른 작업을 정의할 수 있습니다. 예를 들어, Amazon SNS 주제에 푸시 알림을 보내는 작업을 구성할 수 있습니다. 정의할 수 있는 작업에 대한 자세한 내용은 AWS IoT Events 개발자 안내서의 [다른 AWS 서비스 작업을 참조하세요](#).

자산 모델을 업데이트하거나 삭제할 때의 AWS IoT Events 경보 모델이이 자산 모델과 연결된 자산 속성을 모니터링하는지 확인할 AWS IoT SiteWise 수 있습니다. 이렇게 하면 AWS IoT

Events 경보가 현재 사용 중인 자산 속성을 삭제할 수 없습니다. 에서이 기능을 활성화하려면 `iotevents:ListInputRoutings` 권한이 있어야 AWS IoT SiteWise입니다. 이 권한을 통해 AWS IoT SiteWise 는에서 지원하는 [ListInputRoutings](#) API 작업을 호출할 수 있습니다 AWS IoT Events. 자세한 내용은 [\(선택 사항\) ListInputRoutings 권한](#) 단원을 참조하십시오.

Note

경보 알림 기능은 중국(베이징) 리전에서 사용할 수 없습니다.

주제

- [AWS IoT Events 경보 정의\(AWS IoT SiteWise 콘솔\)](#)
- [AWS IoT Events 경보 정의\(AWS IoT Events 콘솔\)](#)
- [AWS IoT Events 경보 정의\(AWS CLI\)](#)

AWS IoT Events 경보 정의(AWS IoT SiteWise 콘솔)

AWS IoT SiteWise 콘솔을 사용하여 기존 자산 모델에 대한 AWS IoT Events 경보를 정의할 수 있습니다. 새 자산 모델에 대한 AWS IoT Events 경보를 정의하려면 자산 모델을 생성한 다음 다음 다음 단계를 완료합니다. 자세한 내용은 [에서 자산 모델 생성 AWS IoT SiteWise](#) 단원을 참조하십시오.

Important

각 경보에는 경보와 비교할 임계값을 지정하는 속성(attribute)이 필요합니다. 경보를 정의하려면 먼저 자산 모델에서 임계값 속성(attribute)을 정의해야 합니다.

풍력 터빈이 최대 풍속 등급인 50mph를 초과할 때 이를 감지하는 경보를 정의하려는 경우를 예로 들어 보겠습니다. 경보를 정의하기 전에 기본값 50으로 속성(attribute)(최대 풍속)을 정의해야 합니다.

자산 모델에 AWS IoT Events 경보를 정의하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 경보를 정의할 자산 모델을 선택합니다.
4. 경보 탭을 선택합니다.
5. 경보 추가를 선택합니다.

6. 경보 유형 옵션 섹션에서 AWS IoT Events 경보를 선택합니다.
7. 경보 세부 정보 섹션에서 다음을 수행합니다.
 - a. 경보의 이름을 입력합니다.
 - b. (선택 사항) 경보에 대한 설명을 입력합니다.
8. 임계값 정의 섹션에서는 경보가 탐지되는 시기와 경보의 심각도를 정의합니다. 다음을 수행합니다.
 - a. 경보가 탐지되는 속성을 선택합니다. 이 속성이 새 값을 수신할 때마다는 경보 상태를 평가하기 AWS IoT Events 위해 값에 AWS IoT SiteWise 전송합니다.
 - b. 속성을 임계값과 비교하는 데 사용할 연산자를 선택합니다. 다음 옵션 중 하나를 선택합니다.
 - < 미만
 - <= 이하
 - == 같음
 - != 같지 않음
 - >= 이상
 - > 초과
 - c. 값에서 임계값으로 사용할 속성 속성을 선택합니다.는 속성의 값이 속성의 값과 AWS IoT Events 비교합니다.
 - d. 경보의 심각도를 입력합니다. 팀에서 이해할 수 있는 숫자를 사용하여 이 경보의 심각도를 반영합니다.
9. (선택 사항) 알림 설정 - 선택 사항 섹션에서 다음을 수행합니다.
 - a. 활성화를 선택합니다.

 Note

비활성을 선택하면 사용자와 팀이 경보 알림을 받지 않게 됩니다.

- b. 수신자에서 수신자를 선택합니다.

 Important

AWS IAM Identity Center 사용자에게 경보 알림을 보낼 수 있습니다. 이 기능을 사용하려면 IAM Identity Center를 활성화해야 합니다. 한 번에 한 AWS 리전에서 IAM

Identity Center를 활성화할 수 있습니다. 즉, IAM Identity Center를 활성화한 리전에 서만 경보 알림을 정의할 수 있습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 [시작하기](#)를 참조하세요.

- c. 프로토콜에서 다음 옵션 중에 선택합니다.
- 이메일 및 문자 메시지 - IAM Identity Center 사용자에게 SMS 메시지와 이메일 메시지를 통해 알림을 전달합니다.
 - 이메일 - 경보가 IAM Identity Center 사용자에게 이메일 메시지를 통해 알림을 전달합니다.
 - 텍스트 - 경보가 IAM Identity Center 사용자에게 SMS 메시지를 통해 알림을 전달합니다.
- d. 발신자에서 발신자를 선택합니다.

 Important

Amazon Simple Email Service(Amazon SES)에서 발신자 이메일 주소를 확인해야 합니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드의 [이메일 주소 자격 증명 확인](#)을 참조하세요.

10. 기본 자산 상태 섹션에서 이 자산 모델에서 생성된 경보의 기본 상태를 설정할 수 있습니다.

 Note

이후 단계에서 이 자산 모델에서 생성한 자산에 대해 이 경보를 활성화하거나 비활성화합니다.

11. 고급 설정 섹션에서 권한, 추가 알림 설정, 경보 상태 작업, SiteWise Monitor의 경보 모델, 확인 흐름을 구성할 수 있습니다.

 Note

AWS IoT Events 경보에는 다음과 같은 서비스 역할이 필요합니다.

- 가 경보 상태 값을 전송하도록 AWS IoT Events 수입하는 역할입니다 AWS IoT SiteWise.
- Lambda로 데이터를 전송하기 위해 수 AWS IoT Events 임하는 역할입니다. 경보가 알림을 보내는 경우에만 이 역할이 필요합니다.

권한 섹션에서 다음을 수행합니다.

- a. AWS IoT Events 역할의 경우 기존 역할을 사용하거나 필요한 권한이 있는 역할을 생성합니다. 이 역할에는 `iotevents.amazonaws.com`가 역할을 맡을 수 있도록 허용하는 `iotsitewise:BatchPutAssetPropertyValue` 권한과 신뢰 관계가 필요합니다.
- b. AWS IoT Events Lambda 역할의 경우 기존 역할을 사용하거나 필요한 권한이 있는 역할을 생성합니다. 이 역할에는 `iotevents.amazonaws.com`가 역할을 맡을 수 있도록 허용하는 `lambda:InvokeFunction` 및 `sso-directory:DescribeUser` 권한과 신뢰 관계가 필요합니다.

12. (선택 사항) 추가 알림 설정 섹션에서 다음을 수행합니다.

- a. 수신자 속성(attribute)에서 알림 수신자를 지정하는 값을 가진 속성(attribute)을 정의합니다. IAM Identity Center 사용자를 수신자로 선택할 수 있습니다.

속성(attribute)을 생성하거나 자산 모델에서 기존 속성(attribute)을 사용할 수 있습니다.

- 새 수신자 속성(attribute) 생성을 선택하는 경우 속성(attribute)에 대해 수신자 속성(attribute) 이름과 수신자 기본값 - 선택 사항을 지정합니다.
- 기존 수신자 속성(attribute) 사용을 선택한 경우 수신자 속성(attribute) 이름에서 속성(attribute)을 선택합니다. 경보는 사용자가 선택한 속성(attribute) 기본값을 사용합니다.

이 자산 모델에서 만든 각 자산의 기본값을 재정의할 수 있습니다.

- b. 사용자 지정 메시지 속성(attribute)의 경우 기본 상태 변경 메시지 외에 전송할 사용자 지정 메시지를 지정하는 값을 가진 속성(attribute)을 정의합니다. 예를 들어 팀이 이 경보를 해결하는 방법을 이해하는 데 도움이 되는 메시지를 지정할 수 있습니다.

속성(attribute)을 생성하거나 자산 모델에서 기존 속성(attribute)을 사용하도록 선택할 수 있습니다.

- 새 사용자 지정 메시지 속성(attribute) 생성을 선택한 경우 속성(attribute)에 대해 사용자 지정 메시지 속성(attribute) 이름과 사용자 지정 메시지 기본값 - 선택 사항을 지정합니다.
- 기존 사용자 지정 메시지 속성(attribute) 사용을 선택한 경우 사용자 지정 메시지 속성(attribute) 이름에서 속성(attribute)을 선택합니다. 경보는 사용자가 선택한 속성(attribute) 기본값을 사용합니다.

이 자산 모델에서 만든 각 자산의 기본값을 재정의할 수 있습니다.

c. Lambda 함수 관리에서 다음 중 하나를 수행합니다.

- 새 Lambda 함수를 AWS IoT SiteWise 생성하려면 AWS 관리형 템플릿에서 새 Lambda 생성을 선택합니다.
- 기존 Lambda 함수를 사용하려면 기존 Lambda 사용을 선택하고 함수 이름을 선택합니다.

자세한 정보는 AWS IoT Events 개발자 안내서의 [경보 알림 관리](#)를 참조하세요.

13. (선택 사항) 상태 작업 설정 섹션에서 다음을 수행합니다.

- 작업 편집을 선택합니다.
- 경보 상태 작업 추가에서 작업을 추가한 다음 저장을 선택합니다.

최대 10개의 작업을 추가할 수 있습니다.

AWS IoT Events 는 경보가 활성 상태일 때 작업을 수행할 수 있습니다. 타이머를 사용하거나 변수를 설정하거나 다른 AWS 리소스로 데이터를 전송하도록 기본 제공 작업을 정의할 수 있습니다. 자세한 정보는 AWS IoT Events 개발자 안내서의 [지원되는 작업](#)을 참조하세요.

14. (선택 사항) SiteWise Monitor에서 경보 모델 관리 - 선택 사항, 활성 또는 비활성화를 선택합니다.

이 옵션을 사용하면 SiteWise Monitor에서 경보 모델을 업데이트할 수 있습니다. 이 옵션은 기본적으로 활성화되어 있습니다.

15. 확인 흐름에서 활성 또는 비활성을 선택합니다. 확인 흐름에 대한 자세한 내용은 [경보 상태](#) 섹션을 참조합니다.

16. 경보 추가를 선택합니다.

Note

AWS IoT SiteWise 콘솔은 자산 모델에 경보를 추가하기 위해 여러 API 요청을 수행합니다. 경보 추가를 선택하면 콘솔에서 이러한 API 요청의 진행 상황을 보여주는 대화 상자가 열립니다. 각 API 요청이 성공하거나 API 요청이 실패할 때까지 이 페이지에 머무르세요. 요청이 실패하면 대화 상자를 닫고 문제를 해결한 다음 경보 추가를 선택하여 다시 시도합니다.

AWS IoT Events 경보 정의(AWS IoT Events 콘솔)

AWS IoT Events 콘솔을 사용하여 기존 자산 모델에 대한 AWS IoT Events 경보를 정의할 수 있습니다. 새 자산 모델에 대한 AWS IoT Events 경보를 정의하려면 자산 모델을 생성한 다음 다음 다음 단계를 완료합니다. 자세한 내용은 [에서 자산 모델 생성 AWS IoT SiteWise](#) 단원을 참조하십시오.

Important

각 경보에는 경보와 비교할 임계값을 지정하는 속성(attribute)이 필요합니다. 경보를 정의하려면 먼저 자산 모델에서 임계값 속성(attribute)을 정의해야 합니다. 풍력 터빈이 최대 풍속 등급인 50mph를 초과할 때 이를 감지하는 경보를 정의하려는 경우를 예로 들어 보겠습니다. 경보를 정의하기 전에 기본값 50으로 속성(attribute)(최대 풍속)을 정의해야 합니다.

자산 모델에 AWS IoT Events 경보를 정의하려면

1. [AWS IoT Events 콘솔](#)로 이동합니다.
2. 탐색 창에서 경보 모델을 선택합니다.
3. 경보 모델 생성을 선택하세요.
4. 경보의 이름을 입력합니다.
5. (선택 사항) 경보에 대한 설명을 입력합니다.
6. 경보 대상 섹션에서 다음을 수행합니다.
 - a. 대상 옵션에서 AWS IoT SiteWise 자산 속성을 선택합니다.
 - b. 경보를 추가하고자 하는 자산 모델을 선택합니다.
7. 임계값 정의 섹션에서는 경보가 탐지되는 시기와 경보의 심각도를 정의합니다. 다음을 수행합니다.
 - a. 경보가 탐지되는 속성을 선택합니다. 이 속성이 새 값을 수신할 때마다는 경보 상태를 평가하기 AWS IoT Events 위해 값에 AWS IoT SiteWise 전송합니다.
 - b. 속성을 임계값과 비교하는 데 사용할 연산자를 선택합니다. 다음 옵션 중 하나를 선택합니다.
 - < 미만
 - <= 이하
 - == 같음

- != 같지 않음
 - >= 이상
 - > 초과
- c. 값에서 임계값으로 사용할 속성 속성을 선택합니다.는 속성의 값이 속성의 값과 AWS IoT Events 비교합니다.
 - d. 경보의 심각도를 입력합니다. 팀에서 이해할 수 있는 숫자를 사용하여 이 경보의 심각도를 반영합니다.
8. (선택 사항) 알림 설정 - 선택 사항 섹션에서 다음을 수행합니다.
 - a. 프로토콜에서 다음 옵션 중에 선택합니다.
 - 이메일 및 문자 메시지 - IAM Identity Center 사용자에게 SMS 메시지와 이메일 메시지를 통해 알림을 전달합니다.
 - 이메일 - 경보가 IAM Identity Center 사용자에게 이메일 메시지를 통해 알림을 전달합니다.
 - 텍스트 - 경보가 IAM Identity Center 사용자에게 SMS 메시지를 통해 알림을 전달합니다.
 - b. 발신자에서 발신자를 선택합니다.
- ⚠ Important**

Amazon Simple Email Service(Amazon SES)에서 발신자 이메일 주소를 확인해야 합니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 [Amazon SES에서 이메일 주소 확인](#)을 참조하세요.
- c. 수신자 속성(attribute) - 선택 사항에서 속성(attribute)을 선택합니다. 경보는 사용자가 선택한 속성(attribute) 기본값을 사용합니다.
 - d. 사용자 지정 메시지 속성(attribute) - 선택 사항에서 속성(attribute)을 선택합니다. 경보는 사용자가 선택한 속성(attribute) 기본값을 사용합니다.
9. 인스턴스 섹션에서 이 경보의 기본 상태를 지정합니다. 이후 단계에서 이 자산 모델에서 생성한 모든 자산에 대해 이 경보를 활성화하거나 비활성화할 수 있습니다.
 10. 고급 설정 설정에서 권한, 추가 알림 설정, 경보 상태 동작, SiteWise Monitor의 경보 모델 및 확인 흐름을 구성할 수 있습니다.

i Note

AWS IoT Events 경보에는 다음과 같은 서비스 역할이 필요합니다.

- 가 경보 상태 값을 전송하도록 AWS IoT Events 수임하는 역할입니다 AWS IoT SiteWise.
- Lambda로 데이터를 전송하기 위해 수 AWS IoT Events 임하는 역할입니다. 경보가 알림을 보내는 경우에만 이 역할이 필요합니다.

- 확인 흐름 섹션에서 활성화됨 또는 비활성화됨을 선택합니다. 확인 흐름에 대한 자세한 내용은 [경보 상태](#) 섹션을 참조합니다.
- 권한 섹션에서 다음을 수행합니다.
 - AWS IoT Events 역할의 경우 기존 역할을 사용하거나 필요한 권한이 있는 역할을 생성합니다. 이 역할에는 `iotevents.amazonaws.com`가 역할을 맡을 수 있도록 허용하는 `iotsitewise:BatchPutAssetPropertyValue` 권한과 신뢰 관계가 필요합니다.
 - Lambda 역할의 경우 기존 역할을 사용하거나 필요한 권한이 있는 역할을 생성합니다. 이 역할에는 `iotevents.amazonaws.com`가 역할을 맡을 수 있도록 허용하는 `lambda:InvokeFunction` 및 `sso-directory:DescribeUser` 권한과 신뢰 관계가 필요합니다.
- (선택 사항) 추가 알림 설정 창에서 다음을 수행합니다.
 - (Lambda 함수 관리에 대해 다음 중 하나를 수행합니다.
 - 새 Lambda 함수를 AWS IoT Events 생성하려면 새 Lambda 함수 생성을 선택합니다.
 - 기존 Lambda 함수를 사용하려면 기존 Lambda 함수 사용을 선택하고 함수 이름을 선택합니다.

자세한 정보는 AWS IoT Events 개발자 안내서의 [경보 알림 관리](#)를 참조하세요.

- (선택 사항) 상태 작업 설정 - 선택 사항 섹션에서 다음을 수행합니다.
 - 경보 상태 작업업에서 작업을 추가하고 저장을 선택합니다.

최대 10개의 작업을 추가할 수 있습니다.

AWS IoT Events 는 경보가 활성 상태일 때 작업을 수행할 수 있습니다. 타이머를 사용하거나 변수를 설정하거나 다른 AWS 리소스로 데이터를 전송하도록 기본 제공 작업을 정의할 수 있습니다. 자세한 정보는 AWS IoT Events 개발자 안내서의 [지원되는 작업](#)을 참조하세요.

11. 생성(Create)을 선택합니다.

Note

AWS IoT Events 콘솔은 자산 모델에 경보를 추가하기 위해 여러 API 요청을 수행합니다. 경보 추가를 선택하면 콘솔에서 이러한 API 요청의 진행 상황을 보여주는 대화 상자가 열립니다. 각 API 요청이 성공하거나 API 요청이 실패할 때까지 이 페이지에 머무르세요. 요청이 실패하면 대화 상자를 닫고 문제를 해결한 다음 경보 추가를 선택하여 다시 시도합니다.

AWS IoT Events 경보 정의(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산 속성을 모니터링하는 AWS IoT Events 경보를 정의할 수 있습니다. 새 자산 모델이나 기존 자산 모델에서 경보를 정의할 수 있습니다. 자산 모델에 경보를 정의한 후에서 경보를 생성하고 자산 모델에 AWS IoT Events 연결합니다. 이 프로세스에서는 다음을 수행합니다.

단계

- [1단계: 자산 모델에 대한 경보 정의](#)
- [2단계: AWS IoT Events 경보 모델 정의](#)
- [3단계: AWS IoT SiteWise 와 간의 데이터 흐름 활성화 AWS IoT Events](#)

1단계: 자산 모델에 대한 경보 정의

새 자산 모델 또는 기존 자산 모델에 경보 정의 및 연결된 속성을 추가합니다.

자산 모델(CLI)에서 경보를 정의하려면 다음을 수행하세요.

1. `asset-model-payload.json`이라는 파일을 생성합니다. 다른 섹션의 단계에 따라 자산 모델의 세부 정보를 파일에 추가하되, 자산 모델을 만들거나 업데이트하라는 요청을 제출하지 마세요. 이 섹션에서는 `asset-model-payload.json` 파일의 자산 모델 세부 정보에 경보 정의를 추가합니다.
 - 자산 모델을 만드는 방법에 대한 자세한 내용은 [자산 모델 생성\(AWS CLI\)](#)을 참조하세요.
 - 기존 자산 모델을 업데이트하는 방법에 대한 자세한 내용은 [자산 또는 구성 요소 모델 업데이트\(AWS CLI\)](#) 섹션을 참조합니다.

Note

자산 모델은 경보로 모니터링할 자산 속성을 포함하여 하나 이상의 자산 속성을 정의해야 합니다.

2. 경보 복합 모델(assetModelCompositeModels)을 자산 모델에 추가합니다. AWS IoT Events 경보 복합 모델은 IOT_EVENTS 유형을 지정하고 경보 소스 속성을 지정합니다. 경보 모델을 생성한 후 경보 소스 속성을 추가합니다 AWS IoT Events.

Important

경보 복합 모델은 나중에 생성하는 AWS IoT Events 경보 모델과 이름이 동일해야 합니다. 경보 모델 이름에는 영숫자만 사용할 수 있습니다. 경보 모델에 동일한 이름을 사용할 수 있도록 고유한 영숫자 이름을 지정합니다.

```
{
  ...
  "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. 자산 모델에 경보 임계값 속성(attribute)을 추가합니다. 이 임계값에 사용할 기본값을 지정합니다. 이 모델을 기반으로 각 자산에서 이 기본값을 재정의할 수 있습니다.

Note

경보 임계값 속성(attribute)은 INTEGER 또는 DOUBLE여야 합니다.

```

{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "Temperature Max Threshold",
      "dataType": "DOUBLE",
      "type": {
        "attribute": {
          "defaultValue": "105.0"
        }
      }
    }
  ]
}
```

4. (선택 사항) 자산 모델에 경보 알림 속성(attribute)을 추가합니다. 이러한 속성은 경보 상태가 변경될 때가 알림을 보내는 데 AWS IoT Events 사용하는 IAM Identity Center 수신자 및 기타 입력을 지정합니다. 이 모델을 기반으로 각 자산에서 이러한 기본값을 재정의할 수 있습니다.

Important

AWS IAM Identity Center 사용자에게 경보 알림을 보낼 수 있습니다. 이 기능을 사용하려면 IAM Identity Center를 활성화해야 합니다. 한 번에 한 AWS 리전에서만 IAM Identity Center를 활성화할 수 있습니다. 즉, IAM Identity Center를 활성화한 리전에서만 경보 알림을 정의할 수 있습니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 [시작하기](#)를 참조하세요.

다음을 수행합니다.

- a. IAM Identity Center 아이덴티티 스토어의 ID를 지정하는 속성(attribute)을 추가합니다. IAM Identity Center [ListInstances](#) API 작업을 사용하여 아이덴티티 스토어를 나열할 수 있습니다. 이 작업은 IAM Identity Center를 활성화한 리전에서만 작동합니다.

```
aws sso-admin list-instances
```

그런 다음 아이덴티티 스토어 ID(예:d-123EXAMPLE)를 속성(attribute)의 기본값으로 지정합니다.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "identityStoreId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "d-123EXAMPLE"
        }
      }
    }
  ]
}
```

- b. 알림을 받는 IAM Identity Center 사용자의 ID를 지정하는 속성(attribute)을 추가합니다. 기본 알림 수신자를 정의하려면 IAM Identity Center 사용자 ID를 기본값으로 추가합니다. IAM Identity Center 사용자 ID를 가져오려면 다음 중 하나를 수행합니다.
 - i. IAM Identity Center [ListUsers](#) API를 사용하여 사용자 이름을 알고 있는 사용자의 ID를 가져올 수 있습니다. *d-123EXAMPLE*을 아이덴티티 스토어의 ID로 바꾸고, *Name*을 사용자의 사용자 이름으로 바꿉니다.

```
aws identitystore list-users \
  --identity-store-id d-123EXAMPLE \
  --filters AttributePath=UserName,AttributeValue=Name
```

- ii. [IAM Identity Center 콘솔](#)을 사용하여 사용자를 검색하고 사용자 ID를 찾습니다.

그런 다음 사용자 ID(예: 123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE)를 속성(attribute)의 기본값으로 지정하거나 기본값 없이 속성(attribute)을 정의합니다.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "userId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE"
        }
      }
    }
  ]
}
```

- c. (선택 사항) SMS(텍스트) 메시지 알림의 기본 발신자 ID를 지정하는 속성(attribute)을 추가합니다. 발신자 ID는 Amazon Simple Notification Service(SNS)가 보내는 메시지에 메시지 발신자로 표시됩니다. 자세한 내용은 AWS 최종 사용자 메시징 SMS 사용 설명서의 [에서 발신자 ID 요청을 AWS 최종 사용자 메시징 SMS](#) 참조하세요.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "senderId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "MyFactory"
        }
      }
    }
  ]
}
```

- d. (선택 사항) 이메일 알림에서 발신 주소로 사용할 기본 이메일 주소를 지정하는 속성(attribute)을 추가합니다.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "fromAddress",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "my.factory@example.com"
        }
      }
    }
  ]
}
```

- e. (선택 사항) 이메일 알림에 사용할 기본 제목을 지정하는 속성(attribute)을 추가합니다.

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "emailSubject",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "[ALERT] High boiler temperature"
        }
      }
    }
  ]
}
```

- f. (선택 사항) 알림에 포함할 추가 메시지를 지정하는 속성(attribute)을 추가합니다. 기본적으로 알림 메시지에는 경보에 대한 정보가 포함됩니다. 사용자에게 자세한 정보를 제공하는 추가 메시지를 포함할 수도 있습니다.

```
{
```

```

...
"assetModelProperties": [
  ...
  {
    "name": "additionalMessage",
    "dataType": "STRING",
    "type": {
      "attribute": {
        "defaultValue": "Turn off the power before you check the alarm."
      }
    }
  }
]
}

```

5. 자산 모델을 만들거나 기존 자산 모델을 업데이트합니다. 다음 중 하나를 수행합니다.

- 다음 명령을 실행하여 자산 모델을 생성합니다.

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-payload.json
```

- 다음 명령을 실행하여 기존 자산 모델을 업데이트합니다. *asset-model-id*를 자산 모델의 ID로 바꿉니다.

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://asset-model-payload.json
```

명령을 실행한 후 응답에서 `assetModelId`를 기록합니다.

예: 보일러 자산 모델

다음 자산 모델은 온도 데이터를 보고하는 보일러를 나타냅니다. 이 자산 모델은 보일러 과열을 감지하는 경보를 정의합니다.

```

{
  "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": {}
    }
  }
]
}
]
}

```

2단계: AWS IoT Events 경보 모델 정의

에서 경보 모델을 생성합니다 AWS IoT Events.에서는 표현식을 AWS IoT Events 사용하여 경보 모델의 값을 지정합니다. 표현식을 사용하여의 값을 지정 AWS IoT SiteWise 하여 경보에 대한 입력으로를 평가하고 사용할 수 있습니다. 가 자산 속성 값을 경보 모델로 AWS IoT SiteWise 전송할 때 AWS IoT Events 는 표현식을 평가하여 속성의 값 또는 자산의 ID를 가져옵니다. 경보 모델에서는 다음 표현식을 사용할 수 있습니다.

- 자산 속성 값

자산 속성의 값을 가져오려면 다음 표현식을 사용합니다. *assetModelId*를 자산의 ID로 바꾸고 *propertyId*를 속성의 ID로 바꿉니다.

```
$sitewise.assetModel.`assetModelId`.`propertyId`.propertyValue.value
```

- 자산 ID

자산의 ID를 가져오려면 다음 표현식을 사용합니다. *assetModelId*를 자산의 ID로 바꾸고 *propertyId*를 속성의 ID로 바꿉니다.

```
$sitewise.assetModel.`assetModelId`.`propertyId`.assetId
```

Note

경보 모델을 생성할 때 AWS IoT SiteWise 값으로 평가되는 표현식 대신 리터럴을 정의할 수 있습니다. 이렇게 하면 자산 모델에 정의하는 속성(attribute)의 수를 줄일 수 있습니다. 하지만 값을 리터럴로 정의하면 자산 모델을 기반으로 자산의 해당 값을 사용자 지정할 수 없습니다.

AWS IoT SiteWise Monitor 또한 사용자는 자산에 대해서만 경보 설정을 구성할 수 있으므로 경보를 사용자 지정할 수 없습니다.

AWS IoT Events 경보 모델을 생성하려면(CLI)

1. 에서 경보 모델을 생성할 때 경보가 사용하는 각 속성의 ID를 지정 AWS IoT Events해야 합니다. 여기에는 다음이 포함됩니다.
 - 복합 자산 모델의 경보 상태 속성
 - 경보가 모니터링하는 속성
 - 임계값 속성(attribute)
 - (선택 사항) IAM Identity Center 아이덴티티 스토어 ID 속성(attribute)
 - (선택 사항) IAM Identity Center 사용자 ID 속성(attribute)
 - (선택 사항) SMS 발신자 ID 속성(attribute)
 - (선택 사항) 이메일 발신 주소 속성(attribute)
 - (선택 사항) 이메일 제목 속성(attribute)
 - (선택 사항) 추가 메시지 속성(attribute)

다음 명령을 실행하여 자산 모델에서 이러한 속성의 ID를 검색합니다. *asset-model-id*를 이전 단계의 자산 모델의 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

이 작업은 자산 모델의 세부 사항이 포함된 응답을 반환합니다. 경보가 사용하는 각 속성의 ID를 기록해 둡니다. 다음 단계에서 AWS IoT Events 경보 모델을 생성할 때 이러한 ID를 사용합니다.

2. 에서 경보 모델을 생성합니다 AWS IoT Events. 다음을 수행합니다.
 - a. alarm-model-payload.json이라는 파일을 생성합니다.
 - b. 다음 JSON 객체를 파일에 복사합니다.
 - c. 경보의 이름(alarmModelName), 설명(alarmModelDescription), 그리고 심각도(severity)를 입력합니다. 심각도에서는 회사의 심각도 수준을 반영하는 정수를 지정합니다.

⚠ Important

경보 모델은 이전에 자산 모델에서 정의한 경보 복합 모델과 이름이 같아야 합니다. 경보 모델 이름에는 영숫자만 사용할 수 있습니다.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3
}
```

- d. 비교 규칙(alarmRule)을 경보에 추가합니다. 이 규칙은 모니터링할 속성(inputProperty), 비교할 임계값(threshold), 사용할 비교 연산자(comparisonOperator)를 정의합니다.

- *assetModelId*를 자산 모델의 ID로 바꿉니다.
- *alarmPropertyId*를 경보가 모니터링하는 속성의 ID로 바꿉니다.
- *thresholdAttributeId*를 임계값 특성(attribute) 속성의 ID로 바꿉니다.
- *GREATER*를 속성 값을 임계값과 비교하는 데 사용할 연산자로 바꿉니다. 다음 옵션 중 하나를 선택합니다.
 - 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. 경보 상태가 변경될 때 경보 상태를 AWS IoT SiteWise 로 보내는 작업 (alarmEventActions)을 추가합니다.

 Note

고급 구성에서 경보 상태가 변경될 때 수행할 추가 작업을 정의할 수 있습니다. 예를 들어 AWS Lambda 함수를 호출하거나 MQTT 주제에 게시할 수 있습니다. 자세한 내용은 AWS IoT Events 개발자 안내서의 [다른 AWS 서비스 작업을](#) 참조하세요.

- *assetModelId*를 자산 모델의 ID로 바꿉니다.
- *alarmPropertyId*를 경보가 모니터링하는 속성의 ID로 바꿉니다.
- *alarmStatePropertyId*를 경보 복합 모델의 경보 상태 속성 ID로 바꿉니다.

```

{
  "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. (선택 사항) 경보 알림 설정을 구성합니다. 경보 알림 작업은 계정의 Lambda 함수를 사용하여 경보 알림을 전송합니다. 자세한 내용은 [의 경보 알림 요구 사항 AWS IoT SiteWise](#) 단원을 참조하십시오. 경보 알림 설정에서 IAM Identity Center 사용자에게 전송하도록 SMS 및 이메일 알림을 구성할 수 있습니다. 다음을 수행합니다.
 - i. alarm-model-payload.json의 페이로드에 경보 알림 구성(alarmNotification)을 추가합니다.
 - *alarmNotificationFunctionArn*을 경보 알림을 처리하는 Lambda 함수의 ARN으로 바꿉니다.

```

{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
        "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
        "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
            "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  }
}

```

```

    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        }
      }
    ]
  }
}

```

- ii. (선택 사항) 경보 상태가 변경될 때 IAM Identity Center 사용자에게 전송하도록 SMS 알림(smsConfigurations)을 구성합니다.

- *identityStoreIdAttributeId*를 IAM Identity Center 아이덴티티 스토어의 ID가 포함된 속성(attribute)의 ID로 바꿉니다.
- *userIdAttributeId*를 IAM Identity Center 사용자의 ID가 포함된 속성(attribute)의 ID로 바꿉니다.
- *senderIdAttributeId*를 Amazon SNS 발신자 ID가 포함된 속성(attribute)의 ID로 바꾸거나 페이로드에서 senderId를 제거합니다.
- *additionalMessageAttributeId*를 추가 메시지가 포함된 속성(attribute)의 ID로 바꾸거나 페이로드에서 additionalMessage를 제거합니다.

```

{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
        "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
        "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  }
},

```

```

"alarmEventActions": {
  "alarmActions": [
    {
      "iotSiteWise": {
        "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
        "propertyId": "'alarmStatePropertyId'"
      }
    }
  ],
},
"alarmNotification": {
  "notificationActions": [
    {
      "action": {
        "lambdaAction": {
          "functionArn": "alarmNotificationFunctionArn"
        }
      },
      "smsConfigurations": [
        {
          "recipients": [
            {
              "ssoIdentity": {
                "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value",
                "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
              }
            }
          ],
          "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
          "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.value"
        }
      ]
    }
  ]
}
}

```

- iii. (선택 사항) 경보 상태가 변경될 때 IAM Identity Center 사용자에게 전송하도록 이메일 알림(emailConfigurations)을 구성합니다.

- *identityStoreIdAttributeId*를 IAM Identity Center 아이덴티티 스토어 ID 특성(attribute) 속성의 ID로 바꿉니다.
- *userIdAttributeId*를 IAM Identity Center 사용자 ID 특성(attribute) 속성의 ID로 바꿉니다.
- *fromAddressAttributeId*를 “발신” 주소 특성(attribute) 속성의 ID로 바꾸거나 페이로드에서 from를 제거합니다.
- *emailSubjectAttributeId*를 이메일 제목 특성(attribute) 속성의 ID로 바꾸거나 페이로드에서 subject를 제거합니다.
- *additionalMessageAttributeId*를 추가 메시지 특성(attribute) 속성의 ID로 바꾸거나 페이로드에서 additionalMessage를 제거합니다.

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
```



```

        "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
        }
    },
    "smsConfigurations": [
        {
            "recipients": [
                {
                    "ssoIdentity": {
                        "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId` .propertyValue.value",
                        "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId` .propertyValue.value"
                    }
                }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId` .propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId` .propertyValue.value"
        }
    ],
    "emailConfigurations": [
        {
            "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId` .propertyValue.value",
            "recipients": {
                "to": [
                    {
                        "ssoIdentity": {
                            "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId` .propertyValue.value",
                            "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId` .propertyValue.value"
                        }
                    }
                ]
            },
            "content": {
                "subject":
"$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId` .propertyValue.value",
                "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId` .propertyValue.value"
            }
        }
    ]
}

```

```

    }
  ]
}
}
}

```

- g. (선택 사항) 경보 기능(alarmCapabilities)을 alarm-model-payload.json의 페이로드에 추가합니다. 이 객체에서 자산 모델을 기반으로 확인 흐름의 활성화 여부와 자산의 기본 활성화 상태를 지정할 수 있습니다. 확인 흐름에 대한 자세한 내용은 [경보 상태](#) 섹션을 참조합니다.

```

{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
"$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
"$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        }
      }
    ]
  }
}

```

```

    },
    "smsConfigurations": [
      {
        "recipients": [
          {
            "ssoIdentity": {
              "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value",
              "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
            }
          }
        ],
        "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
        "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.value"
      }
    ],
    "emailConfigurations": [
      {
        "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value",
        "recipients": {
          "to": [
            {
              "ssoIdentity": {
                "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value",
                "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
              }
            }
          ]
        },
        "content": {
          "subject":
"$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value",
          "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.value"
        }
      }
    ]
  }
}

```

```

    ]
  },
  "alarmCapabilities": {
    "initializationConfiguration": {
      "disabledOnInitialization": false
    },
    "acknowledgeFlow": {
      "enabled": true
    }
  }
}

```

- h. 가 데이터를 전송하기 위해 수입할 AWS IoT Events 수 있는 IAM 서비스 역할(roleArn)을 추가합니다 AWS IoT SiteWise. 이 역할에는 iotevents.amazonaws.com가 역할을 맡을 수 있도록 허용하는 iotsitewise:BatchPutAssetPropertyValue 권한과 신뢰 관계가 필요합니다. 알림을 보내려면 이 역할에도 lambda:InvokeFunction 및 sso-directory:DescribeUser 권한이 필요합니다. 자세한 정보는 AWS IoT Events 개발자 안내서에서 [경보 서비스 역할](#)을 참조하세요.
- 를 이러한 작업을 수행하기 위해가 맡을 AWS IoT Events 수 있는 역할의 ARNroleArn으로 바꿉니다.

```

{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
        "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
        "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
            "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",

```

```

        "propertyId": "'alarmStatePropertyId'"
      }
    }
  ],
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        },
        "smsConfigurations": [
          {
            "recipients": [
              {
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value",
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                }
              }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.value"
          }
        ],
        "emailConfigurations": [
          {
            "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value",
            "recipients": {
              "to": [
                {
                  "ssoIdentity": {
                    "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value",
                    "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                  }
                }
              ]
            }
          }
        ]
      }
    ]
  }
}

```

```

        }
      ]
    },
    "content": {
      "subject":
"$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value",
      "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.value"
    }
  }
]
},
"alarmCapabilities": {
  "initializationConfiguration": {
    "disabledOnInitialization": false
  },
  "acknowledgeFlow": {
    "enabled": false
  }
},
"roleArn": "arn:aws:iam::<123456789012>:role/MyIoTEventsAlarmRole"
}

```

- i. 다음 명령을 실행하여의 페이로드에서 AWS IoT Events 경보 모델을 생성합니다alarm-model-payload.json.

```
aws iotevents create-alarm-model --cli-input-json file://alarm-model-payload.json
```

- j. 이 작업은 경보 모델(alarmModelArn)의 ARN이 포함된 응답을 반환합니다. 이 ARN을 복사하여 다음 단계에서 자산 모델의 경보 정의에 설정합니다.

3단계: AWS IoT SiteWise 와 간의 데이터 흐름 활성화 AWS IoT Events

AWS IoT SiteWise 및에서 필요한 리소스를 생성한 후 리소스 간 데이터 흐름을 활성화하여 경보를 활성화 AWS IoT Events할 수 있습니다. 이 섹션에서는 이전 단계에서 만든 경보 모델을 사용하도록 자산 모델의 경보 정의를 업데이트합니다.

AWS IoT SiteWise 와 간의 데이터 흐름을 활성화하려면 AWS IoT Events (CLI)

- 정보 모델을 자산 모델에서 경고 소스로 설정합니다. 다음을 수행합니다.
 - a. 다음 명령을 실행하여 기존 자산 모델 정의를 검색합니다. *asset-model-id*를 자산 모델의 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

이 작업은 자산 모델의 세부 사항이 포함된 응답을 반환합니다.

- b. `update-asset-model-payload.json`이라는 파일을 만들고 이전 명령의 응답을 파일에 복사합니다.
- c. `update-asset-model-payload.json` 파일에서 다음 키-값 페어를 제거합니다.
 - `assetModelId`
 - `assetModelArn`
 - `assetModelCreationDate`
 - `assetModelLastUpdateDate`
 - `assetModelStatus`
- d. 이전에 정의한 경고 복합 모델에 경고 소스 속성(AWS/ALARM_SOURCE)을 추가합니다. *alarmModelArn*을 경고 소스 속성의 값을 설정하는 경고 모델의 ARN으로 바꿉니다.

```
{
  ...
  "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. 다음 명령을 실행하여 `update-asset-model-payload.json` 파일에 저장된 정의로 자산 모델을 업데이트합니다. *asset-model-id*를 자산 모델의 ID로 바꿉니다.

```

aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://update-asset-model-payload.json

```

이제 자산 모델이 AWS IoT Events에서 이를 감지하는 경보를 정의합니다. 경보는 이 자산 모델을 기반으로 모든 자산의 대상 자산을 모니터링합니다. 각 자산에 대한 경보를 구성하여 각 자산의 임계값 또는 IAM Identity Center 수신자와 같은 속성을 사용자 지정할 수 있습니다. 자세한 내용은 [에서 자산에 대한 경보 구성 AWS IoT SiteWise](#) 단원을 참조하십시오.

에서 외부 경보 정의 AWS IoT SiteWise

외부 경보에는 AWS IoT SiteWise외부에서 감지한 경보 상태가 포함됩니다.

외부 경보 정의(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 기존 자산 모델에 대한 외부 경보를 정의할 수 있습니다. 새 자산 모델에 외부 경보를 정의하려면 자산 모델을 만든 다음 다음 단계를 완료합니다. 자세한 내용은 [에서 자산 모델 생성 AWS IoT SiteWise](#) 단원을 참조하십시오.

자산 모델에 경보를 정의하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택합니다.
3. 경보를 정의할 자산 모델을 선택합니다.
4. 경보 정의 탭을 선택합니다.
5. 경보 추가를 선택합니다.
6. 경보 유형 옵션에서 외부 경보를 선택합니다.
7. 경보의 이름을 입력합니다.
8. (선택 사항) 경보에 대한 설명을 입력합니다.
9. 경보 추가를 선택합니다.

외부 경보 정의(CLI)

AWS CLI 를 사용하여 신규 또는 기존 자산 모델에 대한 외부 경보를 정의할 수 있습니다.

자산 모델에 외부 경보를 추가하려면 자산 모델에 경보 복합 모델을 추가합니다. 외부 경보 복합 모델은 EXTERNAL 유형을 지정하며 경보 소스 속성을 지정하지 않습니다. 다음 예제 복합 경보는 외부 온도 경보를 정의합니다.

```
{
  ...
  "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": {}
        }
    }
]
}
]
}

```

신규 또는 기존 자산 모델에 복합 모델을 추가하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- [자산 모델 생성\(AWS CLI\)](#)
- [자산 또는 구성 요소 모델 업데이트\(AWS CLI\)](#)

외부 경보를 정의한 후에는 자산 모델을 기반으로 자산에 경보 상태를 수집할 수 있습니다. 자세한 내용은 [에서 외부 경보 상태 수집 AWS IoT SiteWise](#) 단원을 참조하십시오.

에서 자산에 대한 경보 구성 AWS IoT SiteWise

자산 모델에 AWS IoT Events 경보를 정의한 후 자산 모델을 기반으로 각 자산에 경보를 구성할 수 있습니다. 임계값과 경보의 알림 설정을 편집할 수 있습니다. 각 값은 자산의 속성(attribute)이므로 속성(attribute)의 기본값을 업데이트하여 이러한 값을 구성할 수 있습니다.

Note

AWS IoT Events 경보에 대해 이러한 값을 구성할 수 있지만 외부 경보에는 구성할 수 없습니다.

주제

- [임계값 구성\(콘솔\)](#)
- [임계값 구성\(AWS CLI\)](#)
- [에서 알림 설정 구성 AWS IoT SiteWise](#)

임계값 구성(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 경보의 임계값을 지정하는 속성 값을 업데이트할 수 있습니다.

경보의 임계값 업데이트(콘솔)

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 경보 임계값을 업데이트할 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 편집을 선택합니다.
5. 경보가 임계값으로 사용하는 속성(attribute)을 찾은 다음 새 값을 입력합니다.
6. 저장을 선택합니다.

임계값 구성(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 경보의 임계값을 지정하는 속성 값을 업데이트할 수 있습니다.

이 절차를 완료하려면 자산의 `assetId` 및 속성의 `propertyId`를 알고 있어야 합니다. 외부 ID를 사용할 수도 있습니다. 자산을 생성했지만 해당 `assetId`를 모르고 있는 경우에는 [ListAssets](#) API를 사용하여 특정 모델에 대한 모든 자산을 표시합니다. [DescribeAsset](#) 작업을 사용하여 속성 ID를 포함한 해당 자산의 속성을 봅니다.

[BatchPutAssetPropertyValue](#) 작업을 사용하여 자산에 속성 값을 할당합니다. 이 작업을 사용하여 한 번에 여러 속성을 설정할 수 있습니다. 이 작업의 페이로드에는 각각 자산 ID, 속성 ID, 속성 값을 포함하는 항목 목록이 포함되어 있습니다.

속성의 값을 업데이트하려면(AWS CLI) 다음을 수행하세요.

1. batch-put-payload.json이라는 파일을 만들고 다음 JSON 개체를 파일에 복사합니다. 이 페이로드 예제에서는 풍력 터빈의 위도와 경도를 설정하는 방법을 보여줍니다. ID, 값 및 타임스탬프를 업데이트하여 사용 사례에 대한 페이로드를 수정합니다.

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

- 페이로드의 각 항목은 고유한 문자열로 정의할 수 있는 entryId를 포함합니다. 요청 항목이 실패하면 각 오류에 해당 요청의 entryId가 포함되므로 다시 시도할 요청을 알 수 있습니다.
- 속성 값을 설정하기 위해 각 특성 속성의 propertyValues 목록에 timestamp-quality-value(TQV) 구조 하나를 포함시킬 수 있습니다. 이 구조에는 새 value와 현재 timestamp가 포함되어야 합니다.
- value -설정되는 속성 유형에 따라 다음 필드 중 하나를 포함하는 구조입니다.
 - booleanValue
 - doubleValue
 - integerValue
 - stringValue
 - nullValue
- timestamp - 현재 Unix 에포크 시간을 초 단위로 포함하는 구조로, timeInSeconds.는 과거 7일 이상 또는 향후 5분 이상 존재한 타임스탬프가 있는 모든 데이터 포인트를 AWS IoT SiteWise 거부합니다.

[BatchPutAssetPropertyValue](#)에 대한 페이로드를 준비하는 방법에 대한 자세한 내용은 [AWS IoT SiteWise APIs 사용하여 데이터 수집](#) 단원을 참조하세요.

2. 다음 명령을 실행하여 속성 값을 AWS IoT SiteWise에 전송합니다.

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

에서 알림 설정 구성 AWS IoT SiteWise

AWS IoT SiteWise 콘솔 또는 AWS Command Line Interface ()를 사용하여 경보 알림 설정을 구성할 수 있습니다AWS CLI.

알림 설정 구성(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 경보에 대한 알림 설정을 지정하는 속성의 값을 업데이트할 수 있습니다.

경보의 알림 설정을 업데이트하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.

2. 탐색 창에서 자산을 선택합니다.
3. 경보 설정을 업데이트할 자산을 선택합니다.
4. 편집을 선택합니다.
5. 변경하려는 알림 설정에 경보가 사용하는 속성(attribute)을 찾은 다음 새 값을 입력합니다.
6. 저장을 선택합니다.

알림 설정 구성(CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 경보에 대한 알림 설정을 지정하는 속성의 값을 업데이트할 수 있습니다.

이 절차를 완료하려면 자산의 `assetId` 및 속성의 `propertyId`를 알고 있어야 합니다. 외부 ID를 사용할 수도 있습니다. 자산을 생성했지만 해당 `assetId`를 모르고 있는 경우에는 [ListAssets](#) API를 사용하여 특정 모델에 대한 모든 자산을 표시합니다. [DescribeAsset](#) 작업을 사용하여 속성 ID를 포함한 해당 자산의 속성을 봅니다.

[BatchPutAssetPropertyValue](#) 작업을 사용하여 자산에 속성 값을 할당합니다. 이 작업을 사용하여 한 번에 여러 속성을 설정할 수 있습니다. 이 작업의 페이로드에는 각각 자산 ID, 속성 ID, 속성 값을 포함하는 항목 목록이 포함되어 있습니다.

속성의 값을 업데이트하려면(AWS CLI) 다음을 수행하세요.

1. `batch-put-payload.json`이라는 파일을 만들고 다음 JSON 개체를 파일에 복사합니다. 이 페이로드 예제에서는 풍력 터빈의 위도와 경도를 설정하는 방법을 보여줍니다. ID, 값 및 타임스탬프를 업데이트하여 사용 사례에 대한 페이로드를 수정합니다.

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

```

- 페이로드의 각 항목은 고유한 문자열로 정의할 수 있는 entryId를 포함합니다. 요청 항목이 실패하면 각 오류에 해당 요청의 entryId가 포함되므로 다시 시도할 요청을 알 수 있습니다.
- 속성 값을 설정하기 위해 각 특성 속성의 propertyValues 목록에 timestamp-quality-value(TQV) 구조 하나를 포함시킬 수 있습니다. 이 구조에는 새 value와 현재 timestamp가 포함되어야 합니다.
 - value -설정되는 속성 유형에 따라 다음 필드 중 하나를 포함하는 구조입니다.
 - booleanValue
 - doubleValue
 - integerValue
 - stringValue
 - nullValue
 - timestamp - 현재 Unix 에포크 시간을 초 단위로 포함하는 구조로, timeInSeconds.는 과거 7일 이상 또는 향후 5분 이상 존재한 타임스탬프가 있는 모든 데이터 포인트를 AWS IoT SiteWise 거부합니다.

[BatchPutAssetPropertyValue](#)에 대한 페이로드를 준비하는 방법에 대한 자세한 내용은 [AWS IoT SiteWise APIs 사용하여 데이터 수집](#) 단원을 참조하세요.

2. 다음 명령을 실행하여 속성 값을 AWS IoT SiteWise에 전송합니다.

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

의 경보에 대응 AWS IoT SiteWise

AWS IoT Events 경보 상태가 변경되면 다음을 수행하여 경보에 응답할 수 있습니다.

- 문제를 처리하고 있음을 나타내는 경보를 확인합니다.
- 경보를 일시 중지하여 일시적으로 비활성화합니다.
- 경보를 비활성화하면 다시 활성화할 때까지 영구적으로 비활성화됩니다.
- 경보 상태를 감지하려면 비활성화된 경보를 활성화합니다.
- 경보를 재설정하 상태 및 최신 값을 지울 수 있습니다.

AWS IoT SiteWise 콘솔 또는 AWS IoT Events API를 사용하여 경보에 응답할 수 있습니다.

Note

AWS IoT Events 경보에 응답할 수 있지만 외부 경보에는 응답할 수 없습니다.

주제

- [경보에 대응\(콘솔\)](#)
- [경보에 응답\(API\)](#)

경보에 대응(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 경보를 승인, 일시 중지, 비활성화 또는 활성화할 수 있습니다.

주제

- [경보 확인\(콘솔\)](#)
- [경보 일시 중지\(콘솔\)](#)
- [경보 비활성화\(콘솔\)](#)

- [경보 활성화\(콘솔\)](#)
- [경보 재설정\(콘솔\)](#)

경보 확인(콘솔)

문제를 처리하고 있음을 나타내는 경보를 확인할 수 있습니다.

Note

경보를 확인하려면 경보에서 확인 흐름을 활성화해야 합니다. AWS IoT SiteWise 콘솔에서 경보를 정의하는 경우 이 옵션이 기본적으로 활성화됩니다.

경보를 확인하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 경보를 확인할 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 경보 탭을 선택합니다.
5. 확인할 경보를 선택한 다음 작업을 선택하여 응답 조치 메뉴를 엽니다.
6. 확인을 선택합니다. 경보 상태가 확인됨으로 변경됩니다.

경보 일시 중지(콘솔)

경보를 일시 중지하여 일시적으로 비활성화할 수 있습니다. 경보를 일시 중지할 기간을 지정합니다.

경보를 일시 중지하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 경보를 일시 중지할 자산을 선택합니다.

i Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 경보 탭을 선택합니다.
5. 일시 중지할 경보를 선택한 다음 작업을 선택하여 응답 조치 메뉴를 엽니다.
6. 일시 정지를 선택합니다. 일시 중지 기간을 지정하는 모델이 열립니다.
7. 일시 중지 길이를 선택하거나 사용자 지정 일시 중지 길이를 입력합니다.
8. 저장을 선택합니다. 경보 상태가 일시 정지 상태로 바뀝니다.

경보 비활성화(콘솔)

경보가 더 이상 감지되지 않도록 비활성화할 수 있습니다. 경보를 비활성화한 후 경보가 감지되도록 하려면 다시 활성화해야 합니다.

경보를 비활성화하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 경보를 사용하지 않을 자산을 선택합니다.

i Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 경보 탭을 선택합니다.
5. 비활성화할 경보를 선택한 다음 작업을 선택하여 응답 조치 메뉴를 엽니다.
6. 비활성화를 선택합니다. 경보 상태가 비활성화됨으로 변경됩니다.

경보 활성화(콘솔)

경보를 비활성화하거나 일시 중지한 후에 다시 감지하도록 활성화할 수 있습니다.

경보를 사용하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.

2. 탐색 창에서 자산을 선택합니다.
3. 경보를 사용하고자 하는 자산을 선택합니다.

 Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 경보 탭을 선택합니다.
5. 활성화할 경보를 선택한 다음 경보를 선택하여 응답 조치 메뉴를 엽니다.
6. 활성화를 선택합니다. 경보 상태가 정상으로 변경됩니다.

경보 재설정(콘솔)

경보를 재설정하여 상태 및 최신 값을 지울 수 있습니다.

경보를 재설정하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 경보를 재설정하고자 하는 자산을 선택합니다.

 Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 경보 탭을 선택합니다.
5. 활성화할 경보를 선택한 다음 경보를 선택하여 응답 조치 메뉴를 엽니다.
6. 재설정을 선택합니다. 경보 상태가 정상으로 변경됩니다.

경보에 응답(API)

AWS IoT Events API를 사용하여 경보를 승인, 일시 중지, 비활성화, 활성화 또는 재설정할 수 있습니다. 자세한 내용은 AWS IoT Events API 참조에 있는 다음 운영을 참조하세요.

- [BatchAcknowledgeAlarm](#)
- [BatchSnoozeAlarm](#)

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

자세한 정보는 AWS IoT Events 개발자 안내서에서 [경보에 응답하기](#)를 참조하세요.

에서 외부 경보 상태 수집 AWS IoT SiteWise

외부 경보는 외부에서 평가하는 경보입니다 AWS IoT SiteWise. AWS IoT SiteWise에 수집하려는 경보 상태를 보고하는 데이터 소스가 있는 경우 외부 경보를 사용할 수 있습니다.

경보 상태 속성에는 경보 상태 데이터 값에 대한 특정 형식이 필요합니다. 각 데이터 값은 문자열로 직렬화된 JSON 객체여야 합니다. 그런 다음 직렬화된 문자열을 문자열 값으로 수집합니다. 자세한 내용은 [경보 상태 속성](#) 단원을 참조하십시오.

Example 경보 상태 데이터 값 예시(직렬화되지 않음)

```
{
  "stateName": "Active"
}
```

Example 경보 상태 데이터 값 예시(직렬화)

```
{"stateName\":\"Active\"}
```

Note

데이터 소스에서 이 형식의 데이터를 보고할 수 없거나 데이터를 수집하기 전에 이 형식으로 변환할 수 없는 경우, 경보 속성을 사용하지 않도록 선택할 수 있습니다. 대신, 문자열 데이터 유형을 사용하여 측정 속성으로 데이터를 수집할 수 있습니다. 자세한 내용은 [장비의 데이터 스트림 정의\(측정값\)](#) 및 [데이터 수집 AWS IoT SiteWise](#) 섹션을 참조하세요.

에서 외부 경보 상태 스트림 매핑 AWS IoT SiteWise

속성 별칭을 정의하여 데이터 스트림을 경보 상태 속성에 매핑할 수 있습니다. 이를 통해 데이터를 수집하거나 검색할 때 경보 상태 속성을 쉽게 식별할 수 있습니다. 속성 별칭에 대한 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#)을 참조하세요.

주제

- [외부 경보 상태 스트림 매핑\(콘솔\)](#)
- [외부 경보 상태 스트림 매핑\(AWS CLI\)](#)

외부 경보 상태 스트림 매핑(콘솔)

속성 별칭을 정의하여 데이터 스트림을 경보 상태 속성에 매핑할 수 있습니다. 이를 통해 데이터를 수집하거나 검색할 때 경보 상태 속성을 쉽게 식별할 수 있습니다. 속성 별칭에 대한 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#)을 참조하세요.

AWS IoT SiteWise 콘솔을 사용하여 경보 상태 속성에 대한 별칭을 설정할 수 있습니다.

경보 상태 속성에 속성 별칭을 설정하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 속성 별칭을 설정할 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 편집을 선택합니다.
5. 경보로 스크롤하고 섹션을 확장합니다.
6. 외부 경보 아래의 속성 별칭 – 선택 사항에 별칭을 입력합니다.
7. 저장을 선택합니다.

외부 경보 상태 스트림 매핑(AWS CLI)

속성 별칭을 정의하여 데이터 스트림을 경보 상태 속성에 매핑할 수 있습니다. 이를 통해 데이터를 수집하거나 검색할 때 경보 상태 속성을 쉽게 식별할 수 있습니다. 속성 별칭에 대한 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#)을 참조하세요.

AWS Command Line Interface (AWS CLI)를 사용하여 경보 상태 속성에 대한 별칭을 설정할 수 있습니다.

이 절차를 완료하려면 자산의 `assetId` 및 속성의 `propertyId`를 알고 있어야 합니다. 외부 ID를 사용할 수도 있습니다. 자산을 생성했지만 해당 `assetId`를 모르고 있는 경우에는 [ListAssets](#) API를 사용

하여 특정 모델에 대한 모든 자산을 표시합니다. [DescribeAsset](#) 작업을 사용하여 속성 ID를 포함한 해당 자산의 속성을 봅니다.

Note

[DescribeAsset](#) 응답에는 해당 자산의 복합 자산 모델 목록이 포함됩니다. 각 경보는 복합 모델입니다. `propertyId`를 찾으려면 경보의 복합 모델을 찾은 다음 해당 복합 모델에서 `AWS/ALARM_STATE` 속성을 찾습니다.

속성 별칭을 설정하여 그렇게 하는 방법에 대한 자세한 내용은 [자산 속성 별칭 업데이트](#)를 참조하세요.

에서 경보 상태 데이터 수집 AWS IoT SiteWise

경보 상태 속성은 경보 상태를 직렬화된 JSON 문자열로 예상합니다. 에서 외부 경보에 경보 상태를 수집하려면이 직렬화된 문자열을 타임스탬프 문자열 값으로 AWS IoT SiteWise수집합니다. 다음 예제는 활성 경보의 상태 데이터 값을 보여줍니다.

```
{\"stateName\": \"Active\"}
```

경보 자산 속성을 식별하려면 다음 중 하나를 지정합니다.

- 데이터를 보낼 경보 자산 속성의 `assetId` 및 `propertyId`.
- 데이터 스트림 별칭(예: `/company/windfarm/3/turbine/7/temperature/high`)인 `propertyAlias`. 이 옵션을 사용하려면 먼저 경보 속성의 별칭을 설정해야 합니다. 경보 상태 속성의 속성 별칭을 설정하는 방법을 알아보려면 [에서 외부 경보 상태 스트림 매핑 AWS IoT SiteWise](#) 섹션을 참조합니다.

다음 예제 [BatchPutAssetPropertyValue](#) API 페이로드는 외부 경보의 상태를 포맷하는 방법을 보여줍니다. 이 외부 경보는 풍력 터빈의 분당 회전수(RPM)측정값이 너무 높을 때 보고합니다.

Example 경보 상태 데이터에 대한 BatchPutAssetPropertyValue 페이로드 예시

```
{
  "entries": [
    {
      "entryId": "unique entry ID",
      "propertyAlias": "/company/windfarm/3/turbine/7/temperature/high",
      "propertyValues": [
```

```
{
  "value": {
    "stringValue": "{\"stateName\":\"Active\"}"
  },
  "timestamp": {
    "timeInSeconds": 1607550262
  }
}
]
```

BatchPutAssetPropertyValue API를 사용하여 데이터를 수집하는 방법에 대한 자세한 내용은 [AWS IoT SiteWise APIs 사용하여 데이터 수집](#) 섹션을 참조하세요.

데이터를 수집하는 다른 방법에 대한 자세한 내용은 [예 데이터 수집 AWS IoT SiteWise](#) 섹션을 참조하세요.

AWS IoT SiteWise 어시스턴트

AWS IoT SiteWise 어시스턴트는 생성형 AI 기반 어시스턴트입니다. 이를 통해 공장 관리자, 품질 엔지니어, 유지 관리 기술자와 같은 사용자가 운영 및 엔터프라이즈 데이터에서 직접 인사이트를 얻고, 문제를 해결하고, 조치를 취할 수 있습니다.

AWS IoT SiteWise 어시스턴트는 AWS IoT 데이터, 자산 모델, 설명서 및 설명서의 정보를 중요한 이벤트에 대한 이해하기 쉬운 요약으로 통합합니다. 또한 대화형 심층 질문 및 답변 세션을 활성화하여 쉽게 진단하고 근본 원인을 탐색하고 권장 사항을 안내할 수 있습니다.

주제

- [AWS IoT SiteWise 도우미 구성](#)
- [데이터세트 생성](#)
- [데이터 세트 편집](#)
- [데이터 세트 삭제](#)
- [AWS IoT SiteWise 어시스턴트 질문](#)

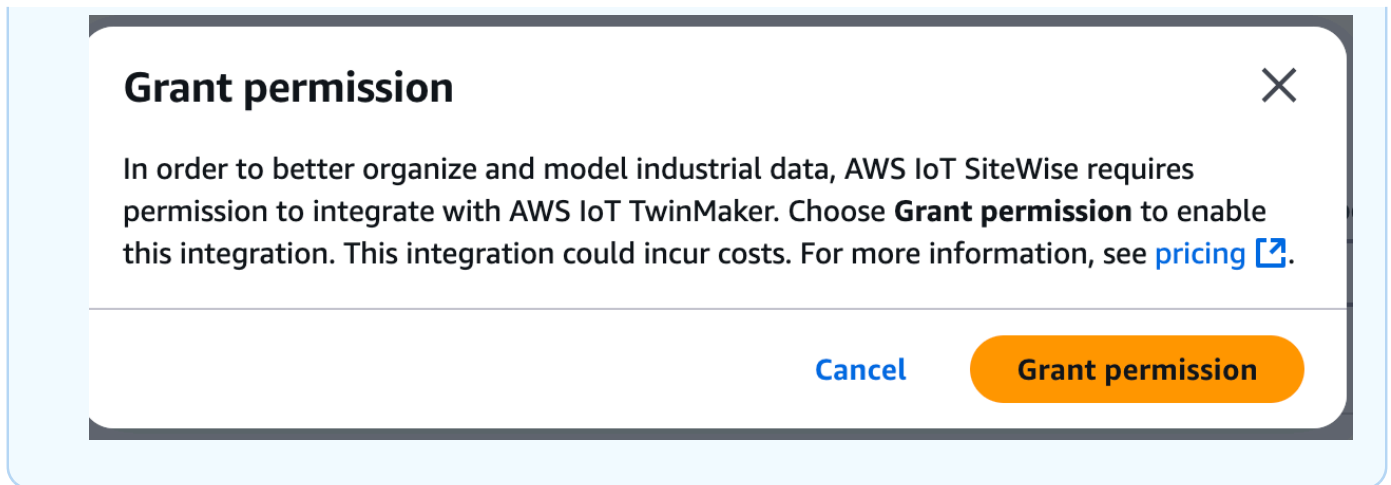
AWS IoT SiteWise 도우미 구성

AWS IoT SiteWise 어시스턴트 구성

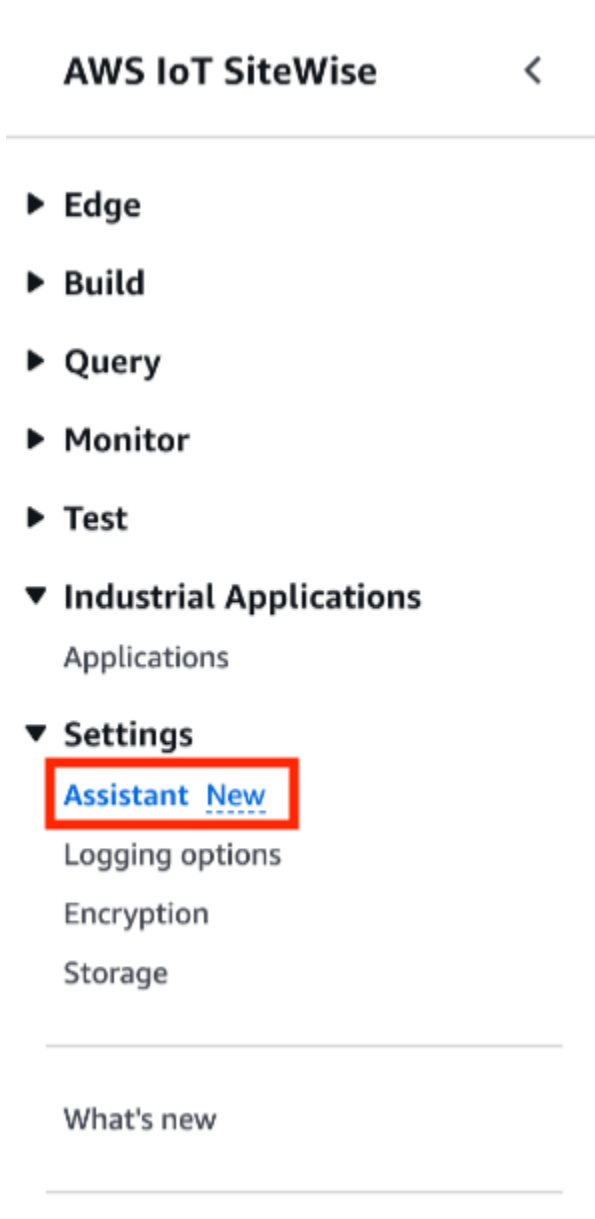
1. [AWS IoT SiteWise 콘솔](#)에 로그인합니다.

Note

AWS IoT TwinMaker 서비스와의 통합을 활성화할 수 있는 권한을 부여합니다. 이는 AWS IoT SiteWise 도우미와 대시보드가 AWS IoT SiteWise 리소스에서 SQL 쿼리를 실행하는 데 필요합니다. [AWS IoT SiteWise 및 통합을 AWS IoT TwinMaker](#) 참조하세요.



2. 왼쪽 탐색 패널에서 도우미를 선택합니다.



데이터세트 생성

Note

AWS IoT SiteWise 도우미는 엔터프라이즈급 지식 및 지침을 위해 [Amazon Kendra](#) 인덱스가 있는 데이터 세트를 사용해야 합니다. Amazon Kendra 인덱스가 없는 경우 [인덱스 생성](#)을 참조하여 인덱스를 생성합니다. [데이터 세트를](#) 추가하면 어시스턴트의 응답 품질이 향상되고 할루시네이션이 최소화됩니다.

Console

AWS IoT SiteWise 콘솔에서 데이터 세트 생성

1. 데이터 세트는 AWS IoT SiteWise 어시스턴트 페이지의 데이터 세트 섹션에 표시됩니다.
2. 데이터 세트가 없는 경우 데이터 세트 생성을 선택합니다.
3. 데이터 세트 세부 정보 페이지의 드롭다운 메뉴에서 데이터 세트와 연결할 Kendra 인덱스를 선택합니다.
4. 데이터 세트 이름은 3단계에서 선택한 Kendra 인덱스로 채워집니다. 필요한 경우 이름을 편집합니다.
5. (선택 사항) 데이터 세트 설명은 3단계에서 선택한 Kendra 인덱스로 채워집니다. 필요한 경우 설명을 편집합니다.
6. 권한 섹션에서 다음 중에서 선택합니다.
 - a. 새 서비스 역할 생성 및 사용을 선택합니다. 기본적으로는 서비스 역할을 AWS IoT SiteWise 자동으로 생성합니다. 이 역할을 통해 AWS IoT SiteWise 도우미는 Kendra 인덱스에 액세스할 수 있습니다.
 - b. 기존 서비스 역할 사용을 선택하고 대상 역할을 선택합니다.
7. 생성(Create)을 선택합니다.

[AWS IoT SiteWise](#) > [Assistant](#) > Create dataset

Create dataset

Create a dataset for the Assistant.

Dataset details [Info](#)

Kendra index

Select a Kendra index for the Assistant dataset.

test-index ▼

[Amazon Kendra](#) 

Dataset name

Dataset name is pre-populated from the Kendra index selected. You can modify the dataset name.

Dataset1

Dataset name must be 1-256 characters. Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen).

Dataset description - optional

Dataset description is pre-populated from the Kendra index selected. You can modify the dataset description.

Dataset for AWS IoT SiteWise Assistant

Dataset description must be 1-2048 characters.

Permissions

Assistant must have permissions to access the data. To create a custom role, visit the [IAM console](#).

Choose a method to authorize Assistant

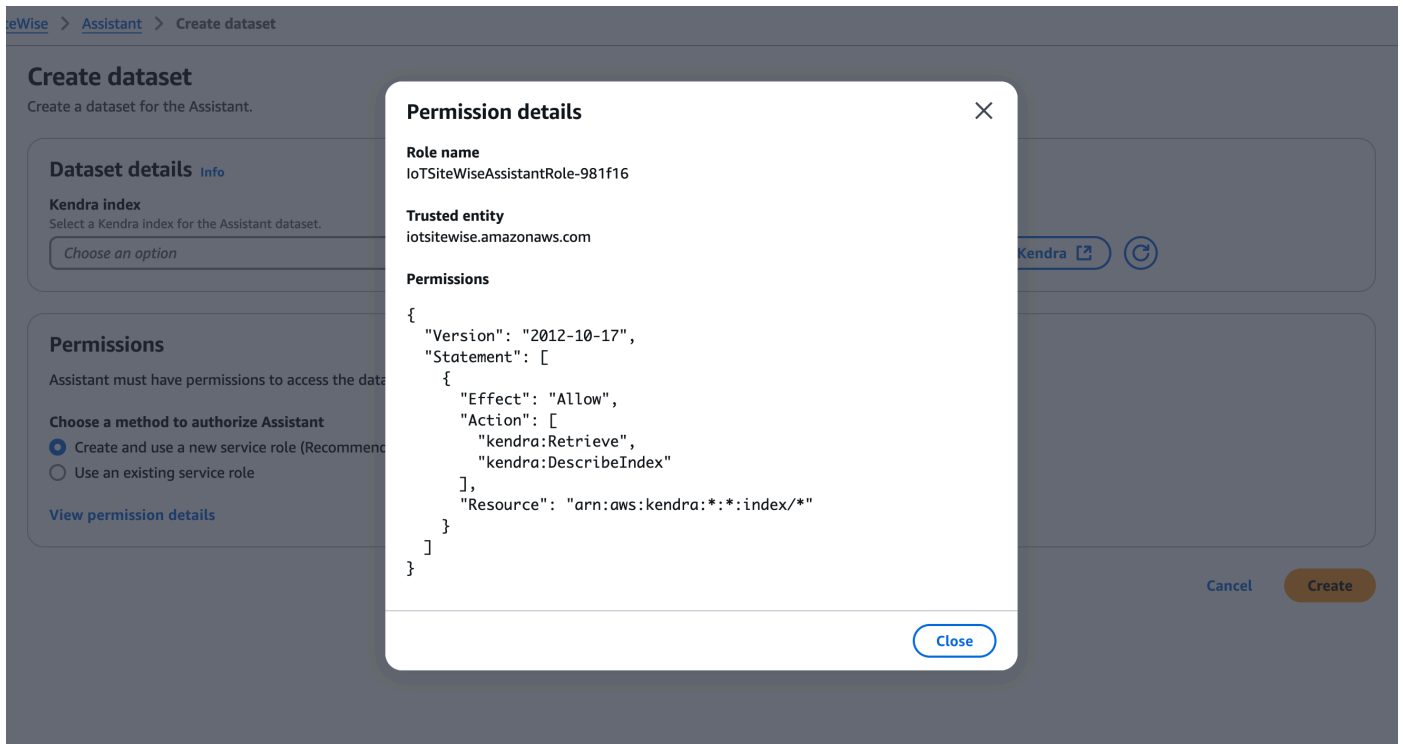
- ☒ Create and use a new service role - *recommended*
- ☐ Use an existing service role

[View permission details](#)

[Cancel](#)

[Create](#)

사용자가 새 서비스 역할 생성 및 사용을 선택한 경우 사용자에게 AWS IoT SiteWise 대해에서 생성한 서비스 역할입니다.



AWS CLI

에서 데이터 세트 생성 AWS CLI

1. 데이터 세트를 생성하는 데 사용되는 IAM 역할을 생성합니다. 다음 권한 정책을 사용합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kendra:Retrieve"
      ],
      "Resource": "arn:aws:kendra:*:*:index/*"
    }
  ]
}
```

다음 신뢰 관계를 사용합니다.

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iotsitewise.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- 예제에 제공된 템플릿을 사용하여 create-dataset.json 파일을 생성합니다. datasetId, kendra knowledgeBaseArn 및 roleArn를 채워이 데이터 세트와 연결합니다.

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

- 다음 명령을 사용하여 데이터 세트를 생성합니다.

```
aws iotsitewise create-dataset --cli-input-json file://create-dataset.json --
region us-east-1
```

데이터 세트 편집

Console

데이터 세트 편집

1. 데이터 세트는 Assistant 페이지의 데이터 세트 섹션에 표시됩니다. 편집할 데이터 세트를 선택합니다. 편집을 선택하여 편집을 시작합니다.
2. 데이터 세트 세부 정보 페이지의 드롭다운 메뉴에서 데이터 세트와 연결할 Kendra 인덱스를 선택합니다.
3. 데이터 세트 이름은 2단계에서 선택한 Kendra 인덱스로 채워집니다. 필요한 경우 이름을 편집합니다.
4. (선택 사항) 데이터 세트 설명은 2단계에서 선택한 Kendra 인덱스로 채워집니다. 필요한 경우 설명을 편집합니다.
5. 권한 섹션에서 다음 중에서 선택합니다.
 - a. 새 서비스 역할 생성 및 사용을 선택합니다. 기본적으로는 서비스 역할을 AWS IoT SiteWise 자동으로 생성합니다. 이 역할을 통해 AWS IoT SiteWise 도우미는 Kendra 인덱스에 액세스할 수 있습니다.
 - b. 기존 서비스 역할 사용을 선택하고 대상 역할을 선택합니다.
6. 변경 사항 저장을 선택하여 선택 사항을 저장합니다.

[AWS IoT SiteWise](#) > [Assistant](#) > Edit dataset

Edit dataset

Edit a dataset for the Assistant.

Dataset details [Info](#)

Kendra index

Select a Kendra index for the Assistant dataset.

test-index ▼

[Amazon Kendra](#) 

Dataset name

Dataset name is pre-populated from the Kendra index selected. You can modify the dataset name.

Dataset1

Dataset name must be 1-256 characters. Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen).

Dataset description - optional

Dataset description is pre-populated from the Kendra index selected. You can modify the dataset description.

Dataset for AWS IoT SiteWise Assistant

Dataset description must be 1-2048 characters.

Permissions

Assistant must have permissions to access the data. To create a custom role, visit the [IAM console](#).

Choose a method to authorize Assistant

- ☐ Create and use a new service role - *recommended*
- ☒ Use an existing service role

Existing role

IoTSiteWiseAssistantRole-40f994 ▼

[View the IoTSiteWiseAssistantRole-40f994 role in the IAM console](#)

[Cancel](#)

[Save changes](#)

AWS CLI

에서 데이터 세트 편집 AWS CLI

- 예제에 제공된 템플릿을 사용하여 update-dataset.json 파일을 생성합니다. datasetId, kendra knowledgeBaseArn 및 roleArn를 채워이 데이터 세트와 연결합니다.

```
{
  "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. 다음 명령을 사용하여 데이터 세트를 업데이트합니다.

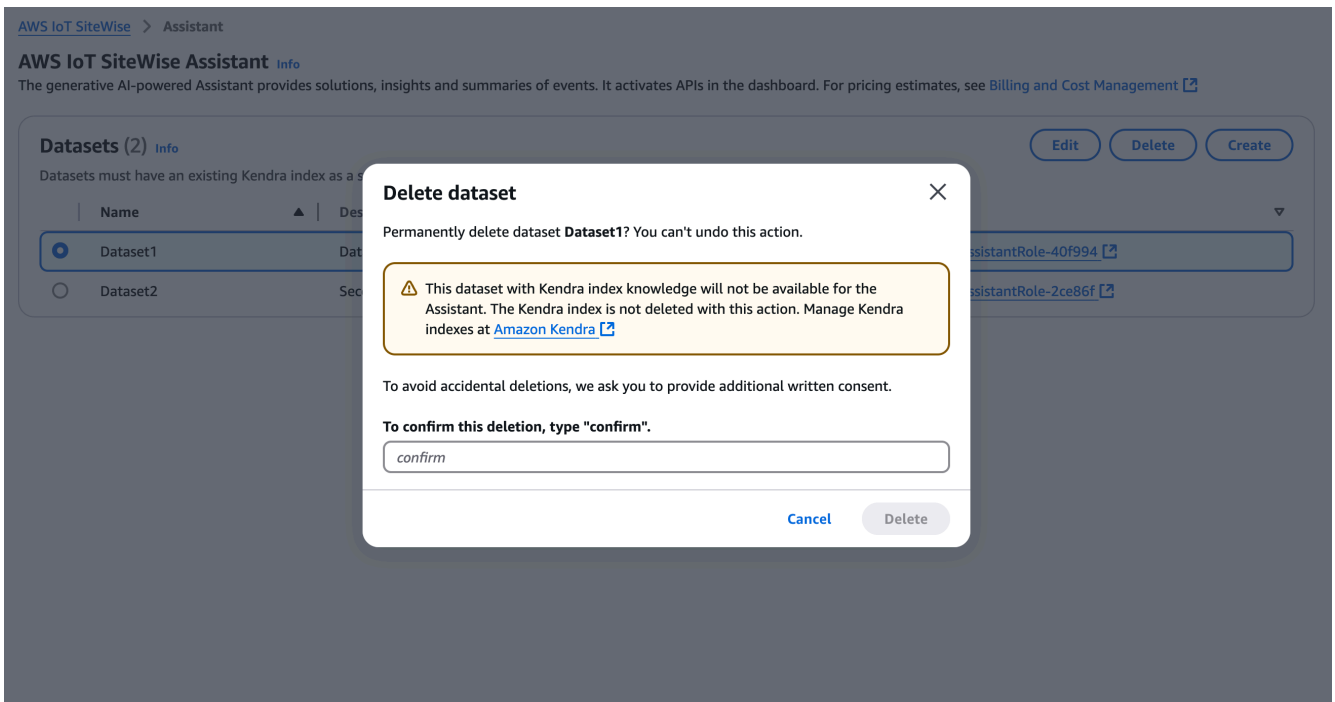
```
aws iotsitewise update-dataset --cli-input-json file://update-dataset.json --
region us-east-1
```

데이터 세트 삭제

Console

데이터 세트 삭제

1. 데이터 세트는 어시스턴트 페이지의 데이터 세트 섹션에 표시됩니다. 데이터 세트를 선택합니다. Delete(삭제)를 선택합니다.
2. 팝업에 확인을 입력하여 삭제를 확인합니다.



3. Delete(삭제)를 선택합니다.

AWS CLI

데이터 세트 삭제

- 를 사용하여 데이터 세트를 삭제합니다datasetId.

```
aws iotsitewise delete-dataset --region us-east-1 --dataset-id <UUID>
```

AWS IoT SiteWise 어시스턴트 질문

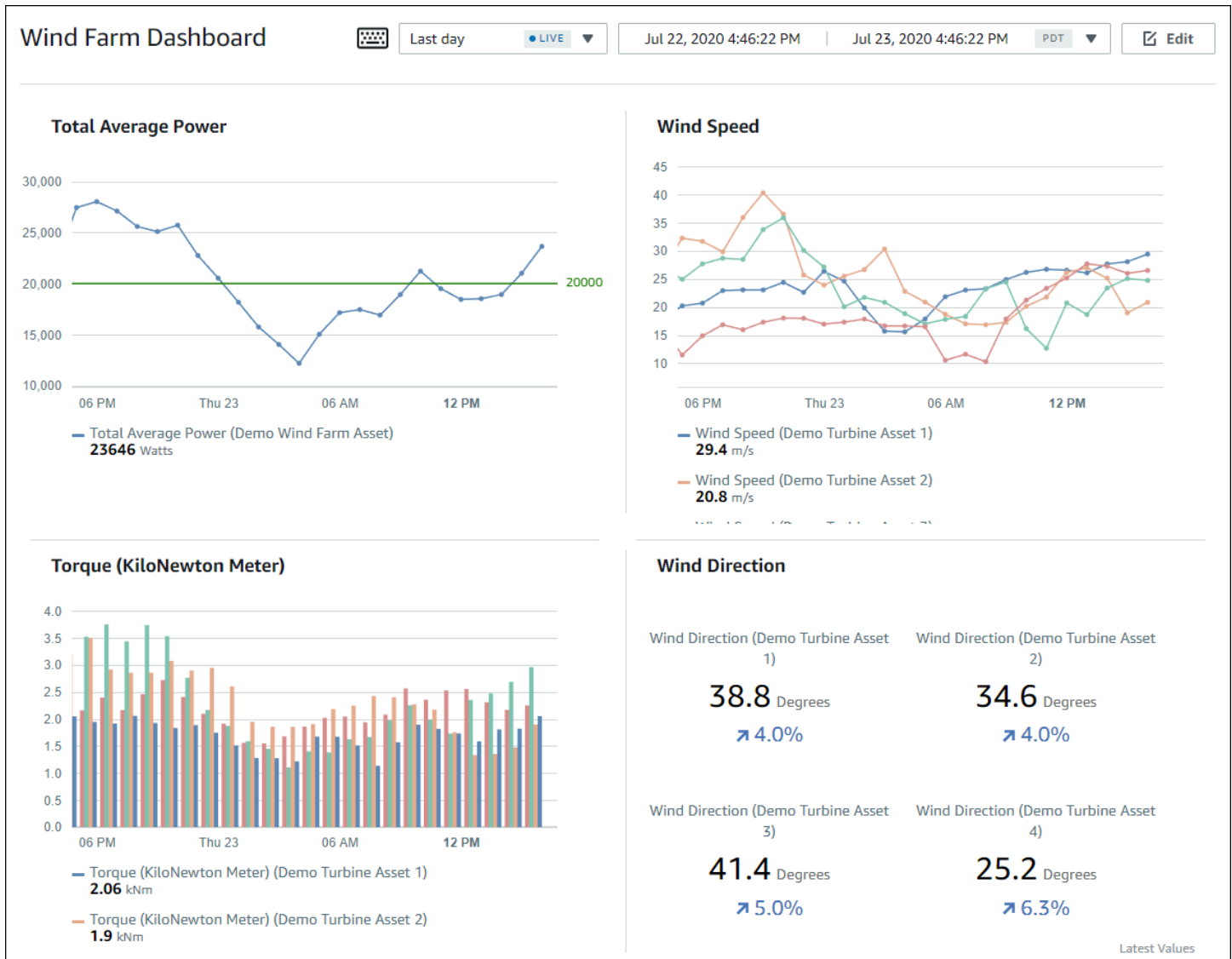
AWS IoT SiteWise Assistant 쿼리[AWS IoT SiteWise 도우미에게 물어볼 샘플 질문](#)에 대한 자세한 내용은 섹션을 참조하세요.

를 사용하여 데이터 모니터링 AWS IoT SiteWise Monitor

AWS IoT SiteWise 를 사용하여 SiteWise Monitor 웹 포털을 생성하여 프로세스, 디바이스 및 장비의 데이터를 모니터링할 수 있습니다. SiteWise Monitor는 관리형 웹 애플리케이션의 형태로 포털을 생성하는 데 사용할 수 있는 AWS IoT SiteWise 있는의 기능입니다. 이러한 포털을 사용하여 운영 데이터를 보고 공유할 수 있습니다. 대시보드로 프로젝트를 만들어, AWS IoT에 연결된 프로세스, 디바이스 및 장비의 데이터를 시각화할 수 있습니다.

프로세스 엔지니어 같은 도메인 전문가는 이러한 포털을 사용해 운영 데이터에 대한 통찰을 빠르게 확보하여 디바이스와 장비의 동작을 파악할 수 있습니다.

다음은 풍력발전단지예 대한 데이터를 표시하는 대시보드의 예입니다.



는 시간 경과에 따른 데이터를 AWS IoT SiteWise 캡처하므로 SiteWise Monitor를 사용하여 시간 경과에 따른 운영 데이터 또는 특정 시점의 마지막으로 보고된 값을 볼 수 있습니다. 이렇게 하면 다른 방법으로는 얻기 힘든 통찰을 얻을 수 있습니다.

SiteWise Monitor 역할

SiteWise Monitor와 상호 작용하는 4가지 역할은 다음과 같습니다.

AWS 관리자

AWS 관리자는 AWS IoT SiteWise 콘솔을 사용하여 포털을 생성합니다. AWS 관리자는 포털 관리자를 할당하고 포털 사용자를 추가할 수도 있습니다. 포털 관리자는 나중에 포털 사용자를 프로젝트에 소유자 또는 뷰어로 할당합니다. 관리자는 AWS AWS 콘솔에서만 작동합니다.

포털 관리자

각 SiteWise Monitor 포털에는 한 명 이상의 포털 관리자가 있습니다. 포털 관리자는 포털을 사용하여 자산 및 대시보드 컬렉션이 포함된 프로젝트를 생성합니다. 그런 다음 포털 관리자가 각 프로젝트에 자산과 소유자를 할당합니다. 프로젝트에 대한 액세스를 제어하면 포털 관리자가 프로젝트 소유자와 뷰어가 볼 수 있는 자산을 제한할 수 있습니다.

프로젝트 소유자

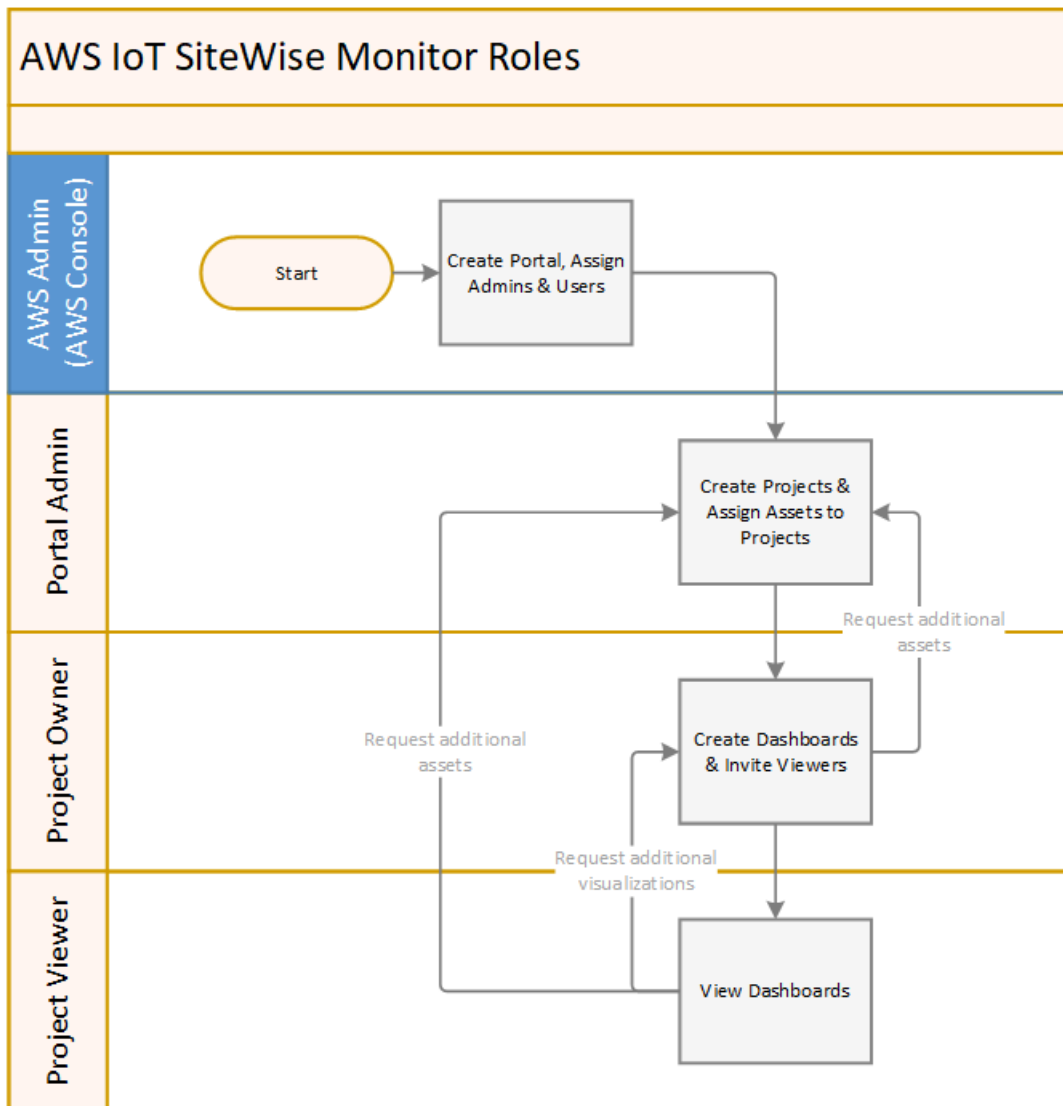
각 SiteWise Monitor 프로젝트에는 소유자가 있습니다. 프로젝트 소유자는 운영 데이터를 일관된 방식으로 나타내기 위해 대시보드 형태로 시각화를 생성합니다. 대시보드를 공유할 준비가 되면 프로젝트 소유자가 프로젝트에 뷰어를 초대할 수 있습니다. 프로젝트 소유자는 프로젝트에 다른 소유자를 할당할 수도 있습니다. 프로젝트 소유자는 경보에 대한 임계값 및 알림 설정을 구성할 수 있습니다.

프로젝트 뷰어

각 SiteWise Monitor 프로젝트에는 뷰어가 있습니다. 프로젝트 뷰어는 포털에 연결하여 프로젝트 소유자가 생성한 대시보드를 볼 수 있습니다. 각 대시보드에서 프로젝트 뷰어는 시간 범위를 조정하여 운영 데이터를 더 잘 이해할 수 있습니다. 프로젝트 뷰어는 액세스 권한이 있는 프로젝트의 대시보드만 볼 수 있습니다. 프로젝트 뷰어는 경보를 확인하고 일시 중지할 수 있습니다.

조직에 따라 같은 사람이 여러 역할을 수행할 수 있습니다.

다음 이미지는 이러한 네 가지 역할이 SiteWise Monitor 포털에서 상호 작용하는 방식을 보여줍니다.



AWS IAM Identity Center 또는 IAM을 사용하여 데이터에 액세스할 수 있는 사용자를 관리할 수 있습니다. 데이터 사용자는 IAM Identity Center 또는 IAM 보안 인증을 사용하여 데스크톱 또는 모바일 브라우저에서 SiteWise Monitor에 로그인할 수 있습니다.

SAML 페더레이션

IAM Identity Center 및 IAM은 [SAML\(Security Assertion Markup Language\) 2.0](#)을 사용한 ID 페더레이션을 지원합니다. SAML 2.0은 많은 외부 ID 제공업체(idP)가 사용자를 인증하고 사용자의 자격 증명 및 보안 정보를 서비스 제공업체(SP)에 전달하는 개방형 표준입니다. SP는 일반적으로 애플리케이션 또는 서비스입니다. SAML 페더레이션을 사용하면 SiteWise Monitor 포털 관리자와 사용자가 회사 사용자 이름 및 암호와 같은 외부 보안 인증을 사용하여 할당된 포털에 로그인할 수 있습니다.

SiteWise Monitor 포털에 액세스하기 위해 SAML 기반 페더레이션을 사용하도록 IAM Identity Center 및 IAM을 구성할 수 있습니다.

IAM Identity Center

포털 관리자와 사용자는 회사 사용자 이름과 암호를 사용하여 AWS 액세스 포털에 로그인할 수 있습니다. 그런 다음 할당된 SiteWise Monitor 포털로 이동할 수 있습니다. IAM Identity Center는 인증서를 사용하여 자격 증명 공급자와 간에 SAML 신뢰 관계를 설정합니다 AWS. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [SCIM 프로파일 및 SAML 2.0 구현](#)을 참조하세요.

IAM

포털 관리자와 사용자는 할당된 SiteWise Monitor 포털에 액세스하기 위한 임시 보안 자격 증명을 요청할 수 있습니다. IAM에서 SAML 자격 증명 공급자 자격 증명을 생성하여 자격 증명 공급자와 간에 신뢰 관계를 설정합니다 AWS. 자세한 내용은 IAM 사용 설명서의 [API 액세스를 위한 SAML 기반 페더레이션 AWS](#) 사용을 참조하세요.

포털 관리자와 사용자는 회사의 포털에 로그인하고 옵션을 선택하여 AWS 관리 콘솔로 이동할 수 있습니다. 그런 다음 할당된 SiteWise Monitor 포털로 이동할 수 있습니다. 회사 포털은 ID 제공업체와 간의 신뢰 교환을 처리합니다 AWS. 자세한 내용은 IAM 사용 설명서의 [SAML 2.0 페더레이션 사용자가 AWS Management Console에 액세스하도록 활성화](#)를 참조하세요.

Note

포털에 사용자나 관리자를 추가할 때 제한된 IP와 같이 사용자 권한을 제한하는 IAM 정책을 생성하지 마세요. 권한이 제한된 연결된 정책은 AWS IoT SiteWise 포털에 연결할 수 없습니다.

SiteWise Monitor 개념

SiteWise Monitor를 사용하려면 다음 개념을 잘 알고 있어야 합니다.

Portal

AWS IoT SiteWise Monitor 포털은 AWS IoT SiteWise 데이터를 시각화하고 공유하는 데 사용할 수 있는 웹 애플리케이션입니다. 포털에는 관리자가 한 명 이상 있으며 프로젝트가 0개 이상 포함되어 있습니다.

Project

각 SiteWise Monitor 포털에는 일련의 프로젝트가 포함되어 있습니다. 각 프로젝트에는 연결된 AWS IoT SiteWise 자산의 하위 집합이 있습니다. 프로젝트 소유자는 하나 이상의 대시보드를 생성하여 해당 자산과 연결된 데이터를 일관된 방식으로 볼 수 있습니다. 프로젝트 소유자는 프로젝트에 뷰어를 초대하여 프로젝트의 자산 및 대시보드를 보도록 허용할 수 있습니다. 이 프로젝트는 SiteWise Monitor 내 공유의 기본 단위입니다. 프로젝트 소유자는 AWS 관리자가 포털에 대한 액세스 권한을 부여한 사용자를 초대할 수 있습니다. 해당 포털에 있는 프로젝트를 해당 사용자와 공유하려면 먼저 사용자에게 포털에 대한 액세스 권한이 있어야 합니다.

자산

산업 장비 AWS IoT SiteWise 에서 데이터를 수집하면 디바이스, 장비 및 프로세스가 각각 자산으로 표시됩니다. 각 자산에는 관련 속성 및 경보가 있습니다. 포털 관리자는 각 프로젝트에 자산 집합을 할당합니다.

속성

속성은 자산과 관련된 시계열 데이터입니다. 예를 들어 장비에는 일련 번호, 위치, 제조업체 및 모델, 설치 날짜가 있을 수 있습니다. 또한 가용성, 성능, 품질, 온도, 압력 등에 대한 시계열 값도 있을 수 있습니다.

경보

경보는 속성을 모니터링하여 장비가 작동 범위를 벗어나는 시기를 식별합니다. 각 경보는 모니터링할 임계값과 속성을 정의합니다. 속성이 임계값을 초과하면 경보가 활성화되어 사용자 또는 팀원이 문제를 해결해야 함을 알립니다. 프로젝트 소유자는 경보의 임계값 및 알림 설정을 사용자 지정할 수 있습니다. 프로젝트 뷰어는 경보를 확인하고 일시 중지할 수 있으며 경보에 대한 세부 정보 또는 경보를 해결하기 위해 취한 조치를 메시지로 남길 수 있습니다.

대시보드

각 프로젝트에는 대시보드 집합이 포함되어 있습니다. 대시보드는 자산 집합의 값에 대한 시각화 집합을 제공합니다. 프로젝트 소유자는 대시보드와 대시보드에 포함된 시각화를 생성합니다. 프로젝트 소유자가 대시보드 집합을 공유할 준비가 되면 소유자는 프로젝트에 뷰어를 초대하여 프로젝트의 모든 대시보드에 대한 액세스 권한을 부여할 수 있습니다. 다른 대시보드에 대해 다른 뷰어 집합을 사용하려면 프로젝트 간에 대시보드를 나누어야 합니다. 뷰어는 대시보드를 볼 때 특정 데이터를 확인할 수 있도록 시간 범위를 사용자 지정할 수 있습니다.

시각화

각 대시보드에서 프로젝트 소유자는 프로젝트와 연결된 속성과 경보를 표시하는 방법을 결정합니다. 가용성은 꺾은선형 차트로 가장 잘 표현되고 다른 값은 막대 차트나 핵심 성능 지표(KPI)로 표시

될 수 있습니다. 경보는 상태 그리드 및 상태 타임라인으로 표시하는 것이 가장 좋습니다. 프로젝트 소유자는 각 시각화를 사용자 지정하여 해당 자산의 데이터를 가장 잘 이해할 수 있도록 합니다.

시작하기 AWS IoT SiteWise Monitor (클래식)

조직의 AWS 관리자인 경우 AWS IoT SiteWise 콘솔에서 포털을 생성합니다. 조직의 구성원이 AWS IoT SiteWise 데이터를 볼 수 있도록 다음 단계를 완료하여 포털을 생성합니다.

1. 포털 구성 및 생성
2. 포털 관리자 추가 및 초대 이메일 보내기
3. 포털 사용자 추가

포털을 생성한 후 포털 관리자는 AWS IoT SiteWise 자산을 보고 포털의 프로젝트에 할당할 수 있습니다. 그러면 프로젝트 소유자가 대시보드를 만들어 자산 속성을 시각화할 수 있는데, 이는 프로젝트 뷰어가 디바이스, 프로세스 및 장비의 작동 방식을 이해하는 데 도움이 됩니다.

Note

포털에 사용자 또는 관리자를 추가할 때는 제한된 IP와 같이 사용자 권한을 제한하는 AWS Identity and Access Management (IAM) 정책을 생성하지 마세요. 권한이 제한된 연결된 정책은 AWS IoT SiteWise 포털에 연결할 수 없습니다.

풍력 발전소 데이터를 사용하여 특정 시나리오의 프로젝트, 대시보드 및 여러 사용자가 포함된 포털을 설정하는 데 필요한 단계를 안내하는 자습서를 따를 수 있습니다. 자세한 내용은 [SiteWise Monitor에서 풍력 발전소 데이터 시각화 및 공유](#) 단원을 참조하십시오.

주제

- [SiteWise Monitor에서 포털 생성](#)
- [SiteWise Monitor에서 포털 구성](#)
- [SiteWise Monitor에서 관리자 초대](#)
- [SiteWise Monitor에서 포털 사용자 추가](#)
- [AWS IoT SiteWise 대시보드 생성\(AWS CLI\)](#)
- [에서 포털에 대한 경보 활성화 AWS IoT SiteWise](#)
- [엣지에서 AWS IoT SiteWise 포털 활성화](#)

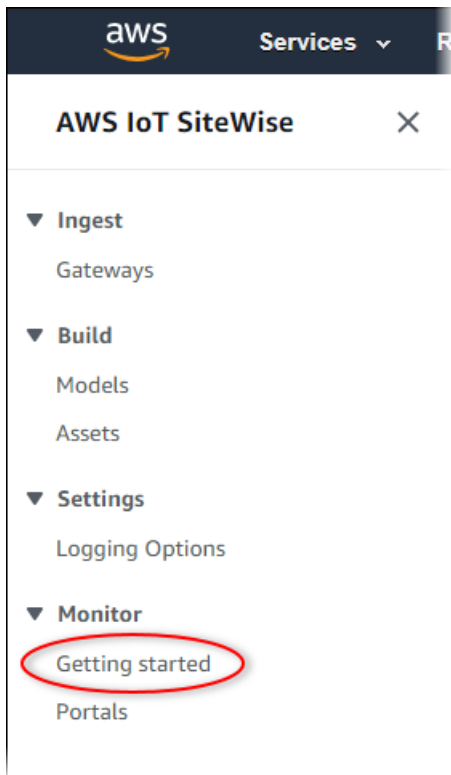
- [SiteWise Monitor 포털 관리](#)

SiteWise Monitor에서 포털 생성

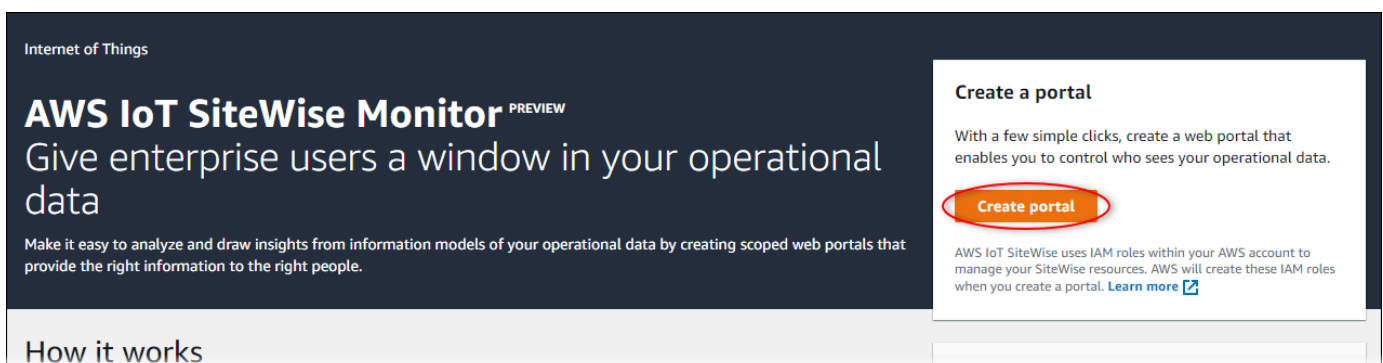
AWS IoT SiteWise 콘솔에서 SiteWise Monitor 포털을 생성합니다.

포털 생성

1. [AWS IoT SiteWise 콘솔](#)에 로그인합니다.
2. 탐색 창에서 모니터, 시작하기를 선택합니다.



3. Create Portal(포털 생성)을 선택합니다.



그런 다음 포털을 구성하기 위한 몇 가지 기본 정보를 입력해야 합니다.

SiteWise Monitor에서 포털 구성

사용자는 포털을 사용하여 데이터를 봅니다. 포털의 이름, 설명, 브랜딩, 사용자 인증, 지원 연락처 이메일 및 권한을 사용자 지정할 수 있습니다.

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configurationStep 2- optional
Additional featuresStep 3
Invite administratorsStep 4
Assign users

Portal configuration

Each web portal provides enterprise users with access to your IoT SiteWise assets. [Learn more](#)

Portal details

Portal name

Choose a portal name to identify the web portal to your users. Company name is recommended.

Name should be 1-128 characters and only contain A-Z a-z 0-9 _ and -.

Description - optional

Create a description of your portal

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.

The file size must be less than 1 MB.

User authentication

Your users can sign in to this portal with their AWS Single Sign-On (AWS SSO) or AWS Identity and Access Management (IAM) credentials. If you choose AWS SSO, you must enable the service for your AWS account.

 You haven't enabled AWS SSO in your account yet. When you create your first portal user, this automatically enables AWS SSO in your AWS account.

☒ AWS SSO

Your users can sign in to the portal with their corporate usernames and passwords.

☐ IAM

Your users can sign in to the portal with their IAM credentials.

Support contact email

You can provide an email address for cases where there's a problem or issue with this portal and your users need to contact support to resolve.

Email

Tags

This resource doesn't have any tags.

You can add up to 50 more tags.

Permissions

SiteWise Monitor assumes this role to give permissions to your federated users to access AWS IoT SiteWise resources. [Learn](#)

포털을 구성하려면 다음을 수행하세요.

1. 프로젝트의 이름을 입력합니다.
2. (선택 사항) 포털에 대한 설명을 입력합니다. 포털이 여러 개인 경우 의미 있는 설명을 사용하여 각 포털에 포함된 내용을 추적할 수 있습니다.
3. (선택 사항) 이미지를 업로드하여 포털에 브랜드를 표시합니다. 정사각형의 PNG 이미지를 선택합니다. 정사각형이 아닌 이미지를 업로드하는 경우 포털에서 이미지의 크기를 정사각형으로 축소합니다.
4. 다음 옵션 중 하나를 선택하세요.
 - 포털 사용자가 회사 사용자 이름과 암호를 사용하여 이 포털에 로그인하는 경우 IAM Identity Center를 선택합니다.

계정에서 IAM Identity Center를 활성화하지 않은 경우, 다음을 수행하세요.

- a. 사용자 생성을 선택합니다.
- b. 사용자 생성 페이지에서 첫 번째 포털을 생성하려면 사용자의 이메일 주소, 이름, 성을 입력한 다음 사용자 생성을 선택합니다.

Create user [X]

When you create your first portal user, this automatically enables AWS SSO in your AWS account.

Email address
janedoe@example.com

First name
Jane

Last name
Doe

Cancel Create user

Note

- AWS 는 첫 번째 포털 사용자를 생성할 때 계정에서 IAM Identity Center를 자동으로 활성화합니다.
- 한 번에 한 리전에서만 IAM Identity Center를 구성할 수 있습니다. SiteWise Monitor는 IAM Identity Center를 위해 구성된 리전에 연결됩니다. 즉, IAM

Identity Center 액세스에는 하나의 리전을 사용하지만 어느 리전에서든 포털을 생성할 수 있습니다.

- 포털 사용자가 IAM 자격 증명으로 이 포털에 로그인하는 경우 IAM을 선택하세요.

Important

사용자 또는 역할에는 포털에 로그인할 수 있는 `iotsitewise:DescribePortal` 권한이 있어야 합니다.

5. 포털에 문제가 있고 이를 해결하는 데 도움이 필요한 경우 포털 사용자가 문의할 수 있는 이메일 주소를 입력합니다.
6. (선택 사항) 포털에 대한 태그를 추가합니다. 자세한 내용은 [AWS IoT SiteWise 리소스에 태그 지정](#) 단원을 참조하십시오.
7. 다음 옵션 중 하나를 선택하세요.
 - 새 서비스 역할 생성 및 사용을 선택합니다. 기본적으로 SiteWise Monitor는 각 포털에 대한 서비스 역할을 자동으로 생성합니다. 이 역할을 통해 포털 사용자는 AWS IoT SiteWise 리소스에 액세스할 수 있습니다. 자세한 내용은 [에 대한 서비스 역할 사용 AWS IoT SiteWise Monitor](#) 단원을 참조하십시오.
 - 기존 서비스 역할 사용을 선택하고 대상 역할을 선택합니다.
8. 다음을 선택합니다.
9. (선택 사항) 포털의 경보를 활성화합니다. 자세한 내용은 [에서 포털에 대한 경보 활성화 AWS IoT SiteWise](#) 단원을 참조하십시오.
10. 생성을 선택합니다. 포털을 AWS IoT SiteWise 생성합니다.

Note

콘솔을 닫으면 관리자와 사용자를 추가하여 설치 프로세스를 완료할 수 있습니다. 자세한 내용은 [에서 포털 관리자 추가 또는 제거 AWS IoT SiteWise](#) 단원을 참조하십시오. 이 포털을 유지용하지 않으려면 리소스를 사용하지 않도록 삭제하세요. 자세한 내용은 [에서 포털 삭제 AWS IoT SiteWise](#) 단원을 참조하십시오.

상태 열은 다음 값 중 하나일 수 있습니다.

- 생성 중 - 포털 생성 요청을 AWS IoT SiteWise 처리하고 있습니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다.
- 업데이트 중 - 포털 업데이트 요청을 AWS IoT SiteWise 처리하고 있습니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다.
- 대기 중 - DNS 레코드 전파가 완료될 때까지 대기 AWS IoT SiteWise 중입니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다. 대기 중 상태일 때 포털을 삭제할 수 있습니다.
- 삭제 중 - 포털 삭제 요청을 AWS IoT SiteWise 처리하고 있습니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다.
- 활성 - 포털이 활성화되면 포털 사용자가 포털에 액세스할 수 있습니다.
- FAILED - 포털 생성, 업데이트 또는 삭제 요청을 처리할 수 AWS IoT SiteWise 없습니다. Amazon CloudWatch Logs AWS IoT SiteWise 로 로그를 전송하도록 활성화한 경우 이러한 로그를 사용하여 문제를 해결할 수 있습니다. 자세한 내용은 [CloudWatch Logs AWS IoT SiteWise 를 사용한 모니터링을 참조하세요](#).

포털이 생성되면 메시지가 나타납니다.

☑ Successfully created portal URL at <https://a1b2c3d4-5678-90ab-cdef-11111EXAMPLE.app.iotsitewise.aws>

×

다음으로, 한 명 이상의 포털 관리자를 포털에 초대해야 합니다. 지금까지 포털을 만들었지만 아무도 액세스할 수 없습니다.

SiteWise Monitor에서 관리자 초대

새 포털에서 시작하려면 포털 관리자를 할당해야 합니다. 포털 관리자는 프로젝트를 만들고, 프로젝트 소유자를 선택하고, 프로젝트에 자산을 할당합니다. 포털 관리자는 모든 AWS IoT SiteWise 자산을 볼 수 있습니다.

사용자 인증 서비스에 따라 다음 옵션 중 하나를 선택합니다.

IAM Identity Center

SiteWise Monitor를 처음 사용하는 경우 이전에 생성한 사용자를 포털 관리자로 선택할 수 있습니다. 다른 사용자를 포털 관리자로 추가하려는 경우 이 페이지에서 IAM Identity Center 사용자를 생성할 수 있습니다. 또는 외부 자격 증명 공급자를 IAM Identity Center에 연결할 수 있습니다. 자세한 내용은 [AWS IAM Identity Center 사용 설명서](#)를 참조하십시오.

관리자를 초대하려면 다음을 수행하세요.

1. 포털 관리자로 초대할 사용자의 확인란을 선택합니다. 그러면 포털 관리자 목록에 사용자가 추가됩니다.

 Note

IAM Identity Center를 자격 증명 스토어로 사용하고 AWS Organizations 관리 계정에 로그인한 경우 사용자 생성을 선택하여 IAM Identity Center 사용자를 생성할 수 있습니다. IAM Identity Center는 새 사용자에게 비밀번호를 설정하라는 이메일을 보냅니다. 그런 다음 사용자를 포털에 관리자로 할당할 수 있습니다. 자세한 내용은 [IAM Identity Center에서 자격 증명 관리](#) 섹션을 참조하세요.

2. (선택 사항) 선택한 사용자에게 초대 보내기를 선택합니다. 이메일 클라이언트가 열리고 메시지 본문에 초대장이 채워집니다.

이메일을 포털 관리자에게 보내기 전에 사용자 지정할 수 있습니다. 나중에 포털 관리자에게 이메일을 보낼 수도 있습니다. IAM Identity Center를 처음 사용하고 새 IAM Identity Center 또는 IAM 사용자 또는 역할을 포털 관리자로 추가하려는 경우 본인에게는 이메일을 보내지 않아도 됩니다.

3. 관리자로 초대하지 않을 사용자를 추가하는 경우 해당 사용자의 확인란 선택을 취소합니다.
4. 포털 관리자 초대를 마쳤으면 다음을 선택합니다.

IAM

포털 관리자가 될 사용자 또는 역할을 선택할 수 있습니다. 다른 사용자나 역할을 포털 관리자로 추가하려는 경우 IAM 콘솔에서 사용자 또는 역할을 생성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [AWS 계정에서 IAM 사용자 생성](#) 및 [IAM 역할 생성](#)을 참조하세요.

관리자를 초대하려면 다음을 수행하세요.

1. 다음을 수행합니다.
 - IAM 사용자를 선택하여 IAM 사용자를 포털 관리자로 추가합니다.
 - IAM 역할을 선택하여 IAM 역할을 포털 관리자로 추가합니다.
2. 포털 관리자로 지정하려는 사용자 또는 확인란을 선택합니다. 그러면 포털 관리자 목록에 사용자 또는 역할이 추가됩니다.

3. 관리자로 지정하지 않으려는 사용자 또는 역할을 추가하는 경우 해당 사용자 또는 역할의 확인란 선택을 취소합니다.
4. 포털 관리자 초대를 마쳤으면 다음을 선택합니다.

Important

사용자 또는 역할에는 포털에 로그인할 수 있는 `iotsitewise:DescribePortal` 권한이 있어야 합니다.

Note

IAM Identity Center를 자격 증명 스토어로 사용하고 AWS Organizations 관리 계정에 로그인한 경우 사용자 생성을 선택하여 IAM Identity Center 사용자를 생성할 수 있습니다. IAM Identity Center는 새 사용자에게 비밀번호를 설정하라는 이메일을 보냅니다. 그런 다음 사용자를 포털에 관리자로 할당할 수 있습니다. 자세한 내용은 [IAM Identity Center에서 자격 증명 관리](#) 섹션을 참조하세요.

나중에 포털 관리자 목록을 변경할 수 있습니다. 자세한 내용은 [에서 포털 관리자 추가 또는 제거 AWS IoT SiteWise](#) 단원을 참조하십시오.

Note

포털 관리자만 프로젝트를 생성하고 프로젝트에 자산을 할당할 수 있으므로 포털 관리자를 한 명 이상 지정해야 합니다.

마지막 단계로, 새 포털에 액세스할 수 있는 사용자를 추가합니다.

SiteWise Monitor에서 포털 사용자 추가

포털에 액세스할 수 있는 사용자를 통제합니다. 각 포털에서 포털 관리자는 하나 이상의 프로젝트를 생성하고 포털 사용자를 각 프로젝트의 소유자 또는 뷰어로 할당합니다. 각 프로젝트 소유자는 프로젝트를 소유하거나 볼 수 있는 추가 포털 사용자를 초대할 수 있습니다.

사용자 인증 서비스에 따라 다음 옵션 중 하나를 선택합니다.

IAM Identity Center

사용자 목록에 사용자를 추가하려면 다음 단계를 완료하세요.

포털 사용자를 추가하려면 다음을 수행하세요.

1. 사용자 목록에서 포털에 추가할 사용자를 선택합니다. 그러면 포털 사용자 목록에 사용자가 추가됩니다. SiteWise Monitor를 처음 사용하는 경우 포털 관리자를 포털 사용자로 추가할 필요가 없습니다.

Note

IAM Identity Center를 자격 증명 스토어로 사용하고 AWS Organizations 관리 계정에 로그인한 경우 사용자 생성을 선택하여 IAM Identity Center 사용자를 생성할 수 있습니다. IAM Identity Center는 새 사용자에게 비밀번호를 설정하라는 이메일을 보냅니다. 그런 다음 사용자를 포털에 사용자로 할당할 수 있습니다. 자세한 내용은 [IAM Identity Center에서 자격 증명 관리](#) 섹션을 참조하세요.

2. 포털에 액세스하지 못하는 사용자를 추가하는 경우 해당 사용자의 확인란 선택을 취소합니다.
3. 사용자 선택을 마치면 사용자 할당을 선택합니다.

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2
[Invite administrators](#)

Step 3
Assign users

Assign users

Select the users you want to be able to access and view this portal. Portal administrators will send invitations to these users at a later date. [Learn more](#)

Users (2) Create user

Find resources

	Display name	Email
☐	Jane Doe	janedoe@example.com
☒	John Doe	johndoe@example.com

Selected users (1)

Cancel Previous **Assign users**

IAM

IAM 사용자 또는 IAM 역할 목록에 추가하려는 사용자 또는 역할이 있으면 다음 단계를 완료하세요.

포털 사용자를 추가하려면 다음을 수행하세요.

1. 다음 옵션이 있습니다.

- IAM 사용자를 선택하여 IAM 사용자를 포털 사용자로 추가합니다.
- IAM 역할을 선택하여 IAM 역할을 포털 사용자로 추가합니다.

SiteWise Monitor를 처음 사용하는 경우 포털 관리자를 포털 사용자로 추가할 필요가 없습니다.

2. 포털 사용자로 지정하려는 사용자 또는 역할의 확인란을 선택합니다. 그러면 포털 사용자 목록에 사용자 또는 역할이 추가됩니다.
3. 포털에 액세스하지 못하는 사용자를 추가하는 경우 해당 사용자의 확인란 선택을 취소합니다.
4. 사용자 선택을 마치면 사용자 할당을 선택합니다.

Important

사용자 또는 역할에는 포털에 로그인할 수 있는 `iotsitewise:DescribePortal` 권한이 있어야 합니다.

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2
[Invite administrators](#)

Step 3
Assign users

Assign users

Select the users you want to be able to access and view this portal. Portal administrators will send invitations to these users at a later date. [Learn more](#)

Users

Roles

IAM users (1) [Manage users in IAM console](#)

< 1 >

☒	Name	Date created
☒	raspberrypi-testing	11-08-2019

► Portal users (1)

Remove

Cancel

Previous

Assign users

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2
Invite administrators

Step 3
Assign users

Assign users

Select the users you want to be able to access and view this portal. Portal administrators will send invitations to these users at a later date. [Learn more](#)

Users **Roles**

IAM roles (66) [Manage roles in IAM console](#)

Find role name

	Name	Date created
☐	[REDACTED]	
☐	[REDACTED]	
☒	AWSIoTSiteWiseMonitorServiceRole_4wZigNpA1	03-16-2021
☐	AWSIoTSiteWiseMonitorServiceRole_ECKT-2Oar	03-11-2021
☐	AWSIoTSiteWiseMonitorServiceRole_GTnd004Wr	03-16-2021
☐	AWSIoTSiteWiseMonitorServiceRole_rHINLNCS-	03-11-2021
☐	AWSIoTSiteWiseMonitorServiceRole_XB330QUIO	03-10-2021
☐	[REDACTED]	
☐	[REDACTED]	
☐	[REDACTED]	

► Portal users (2) Remove

Cancel Previous **Assign users**

축하합니다! 포털을 성공적으로 생성하고 포털 관리자를 할당했으며, 해당 포털을 사용할 수 있는 사용자를 초대하여 할당했습니다. 이제 포털 관리자가 프로젝트를 생성하고 해당 프로젝트에 자산을 할당할 수 있습니다. 그런 다음 프로젝트 소유자는 대시보드를 생성하여 각 프로젝트 자산에 대한 데이터를 시각화할 수 있습니다.

나중에 포털 사용자 목록을 변경할 수 있습니다. 자세한 내용은 [에서 포털 사용자 추가 또는 제거 AWS IoT SiteWise](#) 단원을 참조하십시오.

포털을 변경해야 하는 경우 [SiteWise Monitor 포털 관리](#) 단원을 참조하세요.

포털에서 시작하려면 SiteWise Monitor 애플리케이션 안내서에서 [시작하기](#)를 참조하세요.

AWS IoT SiteWise 대시보드 생성(AWS CLI)

를 사용하여 대시보드에서 시각화(또는 위젯)를 정의할 때 dashboardDefinition JSON 문서에 다음 정보를 지정 AWS CLI해야 합니다. 이 정의는 [CreateDashboard](#) 및 [UpdateDashboard](#) 작업의 파라미터입니다.

widgets

각각 다음 정보를 포함하는 위젯 정의 구조의 목록입니다.

type

위젯 유형은 다음과 같은 위젯 유형을 AWS IoT SiteWise 제공합니다.

- `sc-line-chart` – 선형 차트입니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [선형 차트](#)를 참조하세요.
- `sc-scatter-chart` – 산점도 차트입니다. 자세한 정보는 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [산점도 차트](#)를 참조하세요.
- `sc-bar-chart` – 막대 차트입니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [막대 차트](#)를 참조하세요.
- `sc-status-grid` - 자산 속성의 최신 값을 그리드로 표시하는 상태 위젯입니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [상태 위젯](#)을 참조하세요.
- `sc-status-timeline` - 자산 속성의 과거 값을 타임라인으로 표시하는 상태 위젯입니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [상태 위젯](#)을 참조하세요.
- `sc-kpi` – 핵심 성능 지표(KPI) 시각화입니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [KPI 위젯](#)을 참조하세요.
- `sc-table` - 테이블 위젯입니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [테이블 위젯](#)을 참조하세요.

title

위젯의 제목입니다.

x

그리드의 왼쪽을 기준으로 한 위젯의 수평 위치입니다. 이 값은 대시보드 그리드에서 위젯의 위치를 나타냅니다.

y

그리드의 위쪽을 기준으로 한 위젯의 수직 위치입니다. 이 값은 대시보드 그리드에서 위젯의 위치를 나타냅니다.

width

대시보드 그리드의 공백 수로 표시되는 위젯의 너비입니다.

height

대시보드 그리드의 공백 수로 표시되는 위젯의 높이입니다.

metrics

이 위젯에 대한 데이터 스트림을 각각 정의하는 지표 구조의 목록입니다. 각 목록 구조에는 다음 정보가 포함되어야 합니다.

label

이 지표에 대해 표시할 레이블입니다.

type

이 metric. AWS IoT SiteWise provides에 대한 데이터 소스 유형은 다음과 같은 지표 유형을 제공합니다.

- **iotsitewise** - 대시보드는 AWS IoT SiteWise의 자산 속성에 대한 데이터를 가져옵니다. 이 옵션을 선택하는 경우 이 지표에 대한 **assetId** 및 **propertyId**를 정의해야 합니다.

assetId

(선택 사항) AWS IoT SiteWise의 자산 ID입니다.

이 지표에서 type으로 **iotsitewise**를 선택한 경우 이 필드가 필요합니다.

propertyId

(선택 사항) AWS IoT SiteWise의 자산 속성 ID입니다.

이 지표에서 type으로 **iotsitewise**를 선택한 경우 이 필드가 필요합니다.

analysis

(선택 사항) 위젯에 표시할 분석(예: 추세선)을 정의하는 구조입니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [추세선 구성](#)을 참조하세요. 위젯의 속성별로 각 유형의 추세선을 하나씩 추가할 수 있습니다. 분석 구조에는 다음 정보가 포함되어 있습니다.

trends

(선택 사항) 이 위젯의 추세 분석을 각각 정의하는 추세 구조 목록입니다. 목록의 각 구조에는 다음 정보가 포함되어 있습니다.

type

추세선 유형. 다음 옵션을 선택합니다.

- `linear-regression` - 선형 회귀선을 표시합니다. SiteWise Monitor는 [최소 제곱법](#)을 사용하여 선형 회귀를 계산합니다.

annotations

(선택 사항) 위젯의 임계값을 정의하는 주석 구조입니다. 자세한 내용은 AWS IoT SiteWise Monitor 애플리케이션 안내서의 [임계값 구성](#)을 참조하세요. 위젯당 최대 6개의 주석을 추가할 수 있습니다. 주석 구조에는 다음 정보가 포함되어 있습니다.

y

(선택 사항) 이 위젯의 수평 임계값을 각각 정의하는 주석 구조 목록입니다. 목록의 각 구조에는 다음 정보가 포함되어 있습니다.

comparisonOperator

임계값에 대한 비교 연산자입니다. 다음 중 하나를 선택합니다.

- `LT` - value보다 작은 데이터 포인트가 하나 이상 있는 속성을 강조 표시합니다.
- `GT` - value보다 큰 데이터 포인트가 하나 이상 있는 속성을 강조 표시합니다.
- `LTE` - value보다 작거나 같은 데이터 포인트가 하나 이상 있는 속성을 강조 표시합니다.
- `GTE` - value보다 크거나 같은 데이터 포인트가 하나 이상 있는 속성을 강조 표시합니다.
- `EQ` - value와 같은 데이터 포인트가 하나 이상 있는 속성을 강조 표시합니다.

value

데이터 포인트를 `comparisonOperator`와 비교하기 위한 임계값입니다.

color

(선택 사항) 임계값 색상의 6자리 16진수 코드입니다. 시각화에서는 임계값 규칙을 충족하는 데이터 포인트가 하나 이상 있는 속성에 대한 속성 범례를 이 색상으로 표시합니다. 기본값은 검정색(`#000000`)입니다.

showValue

(선택 사항) 위젯 여백에 임계값을 표시할지 여부입니다. 기본값은 true입니다.

properties

(선택 사항) 위젯 속성의 기본 사전입니다. 이 구조의 멤버는 컨텍스트에 따라 다릅니다.는 properties를 사용하는 다음 위젯을 AWS IoT SiteWise 제공합니다.

- [선형 차트](#), [산점도 차트](#) 및 [막대 차트](#)에는 다음과 같은 속성이 있습니다.

colorDataAcrossThresholds

(선택 사항) 이 위젯의 임계값을 초과하는 데이터의 색상을 변경할지 여부입니다. 이 옵션을 활성화하면 임계값을 초과하는 데이터가 선택한 색상으로 표시됩니다. 기본값은 true입니다.

- [상태 그리드](#)에는 다음과 같은 속성이 있습니다.

labels

(선택 사항) 상태 그리드에 표시할 레이블을 정의하는 구조입니다. 레이블 구조에는 다음 정보가 포함되어 있습니다.

showValue

(선택 사항) 이 위젯의 각 자산 속성에 대한 단위 및 값을 표시할지 여부입니다. 기본값은 true입니다.

Example 대시보드 정의 예

다음 예제에서는 JSON 파일에 저장된 페이로드로 대시보드를 정의합니다.

```
aws iotsitewise create-dashboard \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE \
  --dashboard-name "Wind Farm Dashboard" \
  --dashboard-definition file:///dashboard-definition.json
```

dashboard-definition.json에 대한 다음 JSON 예제는 다음과 같은 시각화 위젯으로 대시보드를 정의합니다.

- 대시보드의 왼쪽 상단에 풍력 발전 단지의 총 전력을 시각화하는 꺾은선형 차트입니다. 이 선형 차트에는 풍력 발전 단지가 최소 예상 출력보다 적은 전력을 출력하는 경우를 나타내는 임계값이 포함되어 있습니다. 이 선형 차트에는 선형 회귀 추세선도 포함되어 있습니다.

- 대시보드의 오른쪽 상단에 네 개의 터빈 풍속을 시각화하는 막대 차트입니다.

Note

이 예제에서는 대시보드의 선형 차트와 막대 차트 시각화를 나타냅니다. 이 대시보드는 [풍력 발전 단지 대시보드의 예](#)와 유사합니다.

```
{
  "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"
      }
    ]
  }
]
}

```

에서 포털에 대한 경보 활성화 AWS IoT SiteWise

포털 관리자가 SiteWise Monitor 포털에서 경보 모델을 생성, 편집 및 삭제할 수 있도록 포털에 AWS IoT Events 대해에서 지원하는 AWS IoT Events 경보 기능을 활성화할 수 있습니다. 프로젝트 소유자

는 경보를 구성할 수 있습니다. 프로젝트 뷰어는 경보 세부 정보를 볼 수 있습니다. 이 섹션에서는 AWS IoT SiteWise 콘솔을 사용하여 포털에 대한 경보 기능을 활성화하는 방법을 설명합니다.

 Important

- 포털에서는 외부 경보를 생성할 수 없습니다.
- 경보 알림을 보내려면 사용자 인증 서비스로 IAM Identity Center를 선택해야 합니다.
- 중국(베이징)에서는 경보 알림 기능을 사용할 수 없습니다 AWS 리전.

포털을 구성하고 생성할 때 2단계 추가 기능에서 경보 및 경보 알림을 활성화할 수 있습니다. 사용자 인증 서비스에 따라 다음 옵션 중 하나를 선택합니다.

IAM Identity Center

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2 - optional
Additional features

Step 3
Invite administrators

Step 4
Assign users

Additional features - optional

Alarms

Your portal users can create alarms in the portal to monitor equipment or processes. They can also get notified when the equipment or processes perform outside specified range.

☒ **Enable alarms**
If enabled, your portal users can define AWS IoT Events alarms in SiteWise Monitor.

AWS IoT SiteWise access role
Choose an IAM role that allows AWS IoT Events to send data to AWS IoT SiteWise. To edit the role, go to the [IAM console](#).

☒ Create a role from an AWS managed template

☐ Use an existing role

☒ **Enable alarm notifications**
If enabled, alarms can send email or SMS notifications.

Sender
Specify the email address that sends alarm notifications. To edit or add a sender, go to the [Amazon SES console](#).

AWS Lambda role
Choose an IAM role that allows AWS Lambda to send data to Amazon SES and Amazon SNS. To edit the role, go to the [IAM console](#).

☒ Create a role from an AWS managed template

☐ Use an existing role

AWS Lambda function
Choose an AWS Lambda function to manage alarm notifications. To edit the function, go to the [AWS Lambda console](#).

☒ Create a lambda from an AWS managed template

☐ Use an existing lambda

Previous **Create**

포털의 경보를 활성화하려면 다음을 수행하세요.

- (선택 사항) 경보 활성화를 선택합니다.
 - AWS IoT SiteWise 액세스 역할의 경우 기존 역할을 사용하거나 필요한 권한이 있는 역할을 생성합니다. 이 역할에는 `iot.amazonaws.com` 및 `iotevents.amazonaws.com`가 역할을 맡을 수 있도록 허용하는 `iotevents:BatchPutMessage` 권한과 신뢰 관계가 필요합니다.
- (선택 사항) 경보 알림 활성화를 선택합니다.
 - 발신자에서 발신자를 선택합니다.

⚠ Important

Amazon SES에서 발신자 이메일 주소를 확인해야 합니다. 자세한 내용은 Amazon Simple Email Service 개발자 가이드에서 [Amazon SES에서 이메일 주소 확인](#)을 참조하십시오.

- b. AWS Lambda 역할의 경우 기존 역할을 사용하거나 필요한 권한이 있는 역할을 생성합니다. 이 역할에는 `iotevents.amazonaws.com` 및 `lambda.amazonaws.com`가 역할을 맡을 수 있도록 허용하는 `lambda:InvokeFunction` 및 `sso-directory:DescribeUser` 권한과 신뢰 관계가 필요합니다.
- c. AWS Lambda 함수의 경우 기존 Lambda 함수를 선택하거나 경보 알림을 관리하는 함수를 생성합니다. 자세한 정보는 AWS IoT Events 개발자 안내서의 [경보 알림 관리](#)를 참조하세요.

IAM

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2- optional
Additional features

Step 3
Invite administrators

Step 4
Assign users

Additional features - optional

Alarms

Your portal users can create alarms in the portal to monitor equipment or processes. They can also get notified when the equipment or processes perform outside specified range.

☒ **Enable alarms**
If enabled, your portal users can define AWS IoT Events alarms in SiteWise Monitor.

AWS IoT SiteWise access role
Choose an IAM role that allows AWS IoT Events to send data to AWS IoT SiteWise. To edit the role, go to the [IAM console](#).

☒ **Create a role from an AWS managed template**

☐ Use an existing role

i Alarms created in the portal can't send notifications. If you want to send alarm notifications, choose **Previous**. Then, on the **Portal configuration** page, choose **AWS SSO for User authentication**.

Previous Create

포털의 경보를 활성화하려면 다음을 수행하세요.

- (선택 사항) 경보 활성화를 선택합니다.

- AWS IoT SiteWise 액세스 역할의 경우 기존 역할을 사용하거나 필요한 권한이 있는 역할을 생성합니다. 이 역할에는 `iot.amazonaws.com` 및 `iotevents.amazonaws.com`가 역할을 맡을 수 있도록 허용하는 `iotevents:BatchPutMessage` 권한과 신뢰 관계가 필요합니다.

SiteWise Monitor의 경보에 대한 자세한 내용은 AWS IoT SiteWise 애플리케이션 안내서에서 [경보로 모니터링](#)을 참조하세요.

엣지에서 AWS IoT SiteWise 포털 활성화

엣지에서 포털을 활성화하면 계정에 데이터 처리 팩이 활성화되어 있는 모든 SiteWise Edge 게이트웨이에서 이 포털을 사용할 수 있습니다.

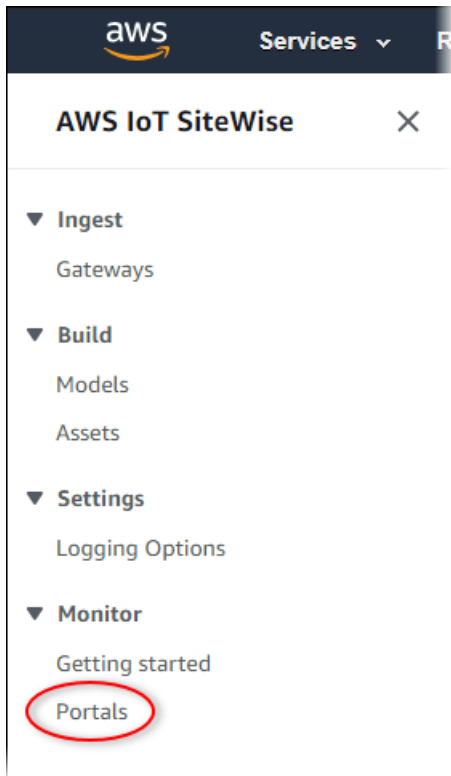
엣지에서 포털을 활성화하려면 다음을 수행하세요.

1. 엣지 구성 섹션에서 엣지에서 이 포털 활성화를 켜세요.
2. 생성(Create)을 선택합니다.

SiteWise Monitor 포털 관리

포털의 다양한 측면을 관리하고 구성할 수 있습니다. 여기에는 사용자 또는 관리자 추가 및 제거, 사용자 권한 및 역할 설정, 포털의 URL, 이름 사용자 지정, 지원 연락처 정보 설정, 포털 관리자에게 이메일 초대 전송이 포함됩니다.

1. [AWS IoT SiteWise 콘솔](#)에 로그인합니다.
2. 탐색 창에서 모니터, 포털을 선택합니다.



3. 포털을 선택한 다음 세부 정보 보기(또는 포털 이름)를 선택합니다.

4. 다음 관리 작업을 수행할 수 있습니다.

- [에서 포털 세부 정보 변경 AWS IoT SiteWise](#)
- [에서 포털 관리자 추가 또는 제거 AWS IoT SiteWise](#)
- [포털 관리자에게 이메일 초대 전송](#)
- [에서 포털 사용자 추가 또는 제거 AWS IoT SiteWise](#)
- [에서 포털 삭제 AWS IoT SiteWise](#)

포털 생성 방법에 대한 자세한 내용은 [시작하기 AWS IoT SiteWise Monitor \(클래식\)](#) 단원을 참조하세요.

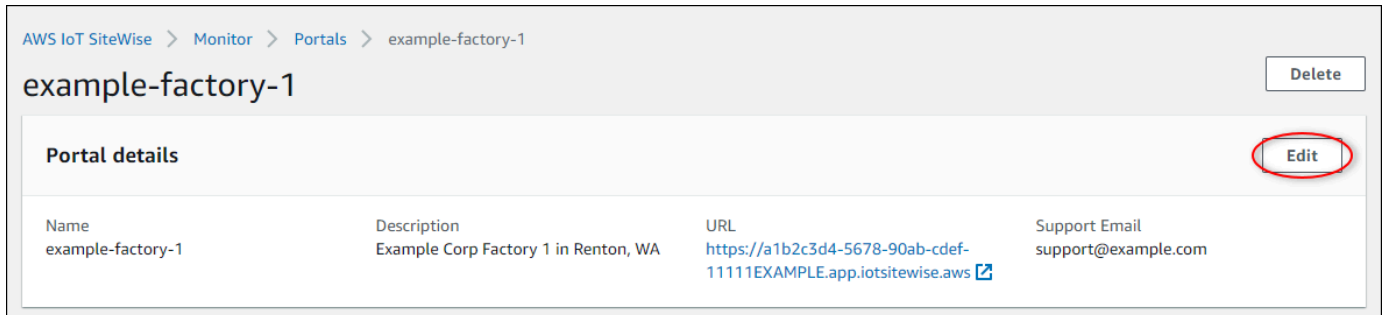
주제

- [에서 포털 세부 정보 변경 AWS IoT SiteWise](#)
- [에서 포털 관리자 추가 또는 제거 AWS IoT SiteWise](#)
- [포털 관리자에게 이메일 초대 전송](#)
- [에서 포털 사용자 추가 또는 제거 AWS IoT SiteWise](#)
- [에서 포털 삭제 AWS IoT SiteWise](#)

에서 포털 세부 정보 변경 AWS IoT SiteWise

포털의 이름, 설명, 브랜딩, 지원 이메일 및 권한을 변경할 수 있습니다.

1. 포털 세부 정보 페이지의 포털 세부 정보 섹션에서 편집을 선택합니다.

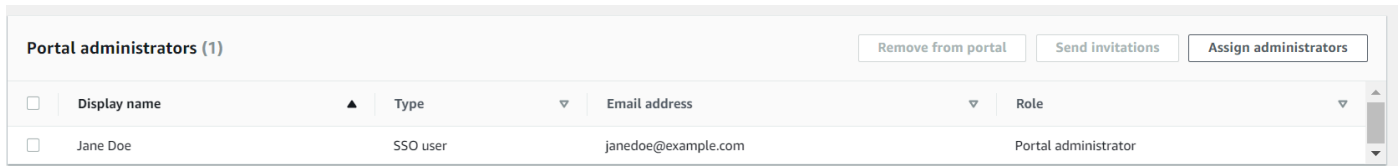


2. 이름, 설명, 포털 브랜딩, 지원 연락처 이메일 또는 권한을 업데이트합니다.
3. 작업을 마쳤으면 저장을 선택합니다.

에서 포털 관리자 추가 또는 제거 AWS IoT SiteWise

몇 단계를 거쳐 포털의 관리자 역할을 할 사용자를 추가하거나 제거할 수 있습니다. 사용자 인증 서비스에 따라 다음 옵션 중 하나를 선택합니다.

IAM Identity Center



포털 관리자를 추가하려면 다음을 수행하세요.

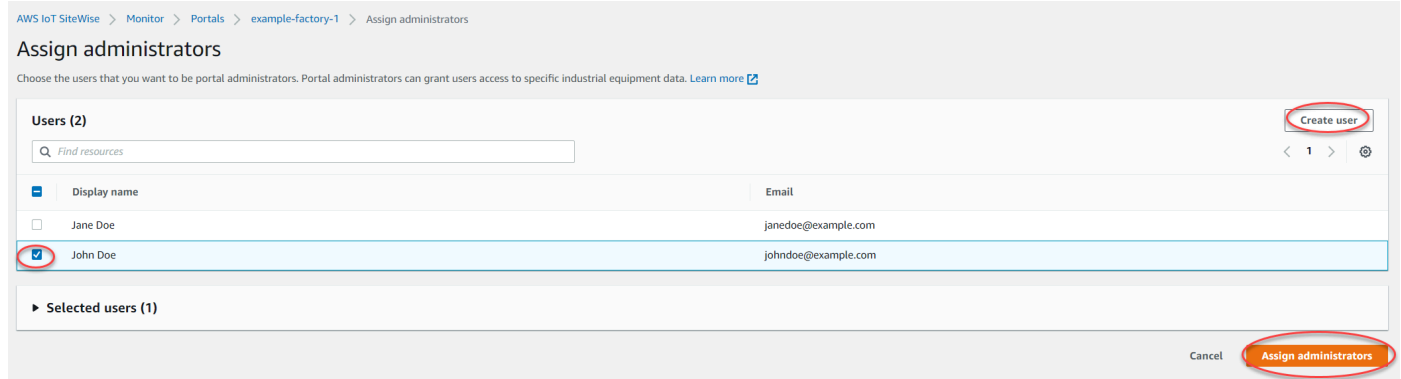
1. 포털 세부 정보 페이지의 포털 관리자 섹션에서 관리자 할당을 선택합니다.
2. 관리자 할당 페이지에서 포털에 관리자로 추가할 사용자의 확인란을 선택합니다.

Note

IAM Identity Center를 자격 증명 스토어로 사용하고 AWS Organizations 관리 계정에 로그인한 경우 사용자 생성을 선택하여 IAM Identity Center 사용자를 생성할 수 있습니다. IAM Identity Center는 새 사용자에게 비밀번호를 설정하라는 이메일을 보냅니다.

니다. 그런 다음 사용자를 포털에 관리자로 할당할 수 있습니다. 자세한 내용은 [IAM Identity Center에서 자격 증명 관리](#) 섹션을 참조하세요.

3. 관리자 할당을 선택합니다.



포털 관리자를 제거하려면 다음을 수행하세요.

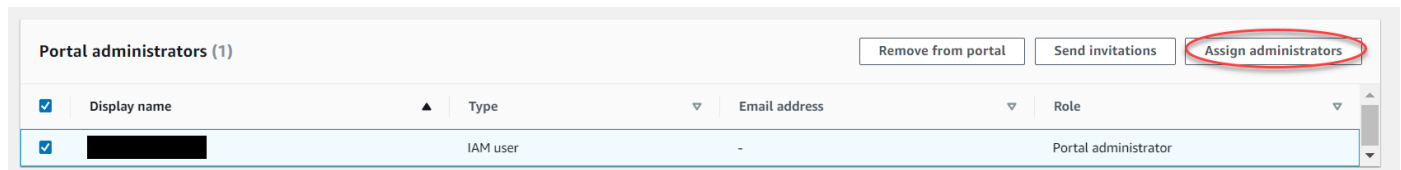
- 포털 세부 정보 페이지의 포털 관리자 섹션에서 제거할 각 사용자의 확인란을 선택한 다음 포털에서 제거를 선택합니다.



Note

포털 관리자를 한 명 이상 선택하는 것이 좋습니다.

IAM



포털 관리자를 추가하려면 다음을 수행하세요.

1. 포털 세부 정보 페이지의 포털 관리자 섹션에서 관리자 할당을 선택합니다.
2. 관리자 할당 페이지에서 다음을 수행하세요.
 - IAM 사용자를 포털 관리자로 추가하려면 IAM 사용자를 선택합니다.
 - IAM 역할을 포털 관리자로 추가하려면 IAM 역할을 선택합니다.

3. 포털 관리자로 초대할 사용자 또는 역할의 확인란을 선택합니다. 그러면 포털 관리자 목록에 사용자 또는 역할이 추가됩니다.
4. 관리자 할당을 선택합니다.

Important

사용자 또는 역할에는 포털에 로그인할 수 있는 `iotsitewise:DescribePortal` 권한이 있어야 합니다.

AWS IoT SiteWise > Monitor > Portals > example-factory-2 > Assign administrators

Assign administrators

Choose the users that you want to be portal administrators. Portal administrators can grant users access to specific industrial equipment data. [Learn more](#)

Users Roles

IAM users (1) [Manage users in IAM console](#)

Find user name

☒	Name	Date created
☒	raspberrypi-testing	11-08-2019

► Portal administrators (1) [Remove](#)

[Cancel](#) [Assign administrators](#)

AWS IoT SiteWise > Monitor > Portals > example-factory-2 > Assign administrators

Assign administrators

Choose the users that you want to be portal administrators. Portal administrators can grant users access to specific industrial equipment data. [Learn more](#)

Users **Roles**

IAM roles (66) [Manage roles in IAM console](#)

Find role name

☐	Name	Date created
☐	[REDACTED]	
☒	AWSIoTSiteWiseMonitorServiceRole_4wZigNpA1	03-16-2021
☐	AWSIoTSiteWiseMonitorServiceRole_ECKT-2Oar	03-11-2021
☐	AWSIoTSiteWiseMonitorServiceRole_GTnd0O4Wr	03-16-2021
☐	AWSIoTSiteWiseMonitorServiceRole_rHINLNCs-	03-11-2021
☐	AWSIoTSiteWiseMonitorServiceRole_XB330QUIO	03-10-2021
☐	[REDACTED]	
☐	[REDACTED]	
☐	[REDACTED]	

► Portal administrators (2) [Remove](#)

[Cancel](#) [Assign administrators](#)

포털 관리자를 제거하려면 다음을 수행하세요.

- 포털 세부 정보 페이지의 포털 관리자 섹션에서 제거할 각 사용자의 확인란을 선택한 다음 포털에서 제거를 선택합니다.

Note

포털 관리자 없이 포털을 종료하는 것은 권장하지 않습니다.

포털 관리자에게 이메일 초대 전송

포털 관리자에게 이메일 초대를 보낼 수 있습니다.

1. 포털 세부 정보 페이지의 포털 관리자 섹션에서 포털 관리자의 확인란을 선택합니다.

Portal administrators (1)				Remove from portal	Send invitations	Assign users
☒	Display name	Email address	Role			
☒	John Doe	john.doe@example.com	Portal administrator			

2. 초대 보내기를 선택합니다. 이메일 클라이언트가 열리고 메시지 본문에 초대장이 채워집니다.

이메일을 포털 관리자에게 보내기 전에 사용자 지정할 수 있습니다.

에서 포털 사용자 추가 또는 제거 AWS IoT SiteWise

포털에 액세스할 수 있는 사용자를 통제합니다. 포털 사용자는 SiteWise Monitor 포털 내의 사용자 목록에 나타납니다. 이 목록에서 포털 관리자는 프로젝트 소유자를 추가할 수 있으며 프로젝트 소유자는 프로젝트 뷰어를 추가할 수 있습니다.

Note

포털 관리자와 포털 사용자는 사용자를 추가하거나 제거해야 하는 경우 포털의 지원 이메일을 통해 사용자에게 연락할 수 있습니다.

사용자 인증 서비스에 따라 다음 옵션 중 하나를 선택합니다.

IAM Identity Center

Portal users (1)					Remove from portal	Assign users
☐	Display name	Type	Email address	Role		
☐	John Doe	SSO user	johndoe@example.com	Portal viewer		

포털 사용자를 추가하려면 다음을 수행하세요.

1. 포털 세부 정보 페이지의 포털 사용자 섹션에서 사용자 할당을 선택합니다.
2. 사용자 할당 페이지에서 포털에 추가할 사용자의 확인란을 선택합니다.

Note

IAM Identity Center를 자격 증명 스토어로 사용하고 AWS Organizations 관리 계정에 로그인한 경우 사용자 생성을 선택하여 IAM Identity Center 사용자를 생성할 수 있습니다. IAM Identity Center는 새 사용자에게 비밀번호를 설정하라는 이메일을 보냅니다. 그런 다음 사용자를 포털에 사용자로 할당할 수 있습니다. 자세한 내용은 [IAM Identity Center에서 자격 증명 관리](#) 섹션을 참조하세요.

3. 사용자 할당을 선택합니다.

AWS IoT SiteWise > Monitor > Portals > example-factory-1 > Assign users

Assign users

Users (2)

Find resources

☐	Display name	Email
☒	John Doe	johndoe@example.com
☐	Jane Doe	janedoe@example.com

Selected users (1)

Cancel Assign users

포털 사용자를 제거하려면 다음을 수행하세요.

- 포털 세부 정보 페이지의 포털 사용자 섹션에서 포털에서 제거할 사용자의 확인란을 선택한 다음 포털에서 제거를 선택합니다.

IAM

Portal users (1)					Remove from portal	Assign users
☐	Display name	Type	Email address	Role		
☐	AWSIoTSiteWiseMonitorServiceRole_4wZigNpA1	IAM role	-	Portal viewer		

포털 사용자를 추가하려면 다음을 수행하세요.

1. 포털 세부 정보 페이지의 포털 사용자 섹션에서 사용자 할당을 선택합니다.
2. 사용자 할당 페이지에서 다음을 수행합니다.
 - IAM 사용자를 선택하여 IAM 사용자를 포털 사용자로 추가합니다.
 - IAM 역할을 선택하여 IAM 역할을 포털 사용자로 추가합니다.
3. 포털 사용자로 추가하려는 사용자 또는 역할의 확인란을 선택합니다. 그러면 포털 사용자 목록에 사용자 또는 역할이 추가됩니다.
4. 사용자 할당을 선택합니다.

AWS IoT SiteWise > Monitor > Portals > example-factory-2 > Assign users

Assign users

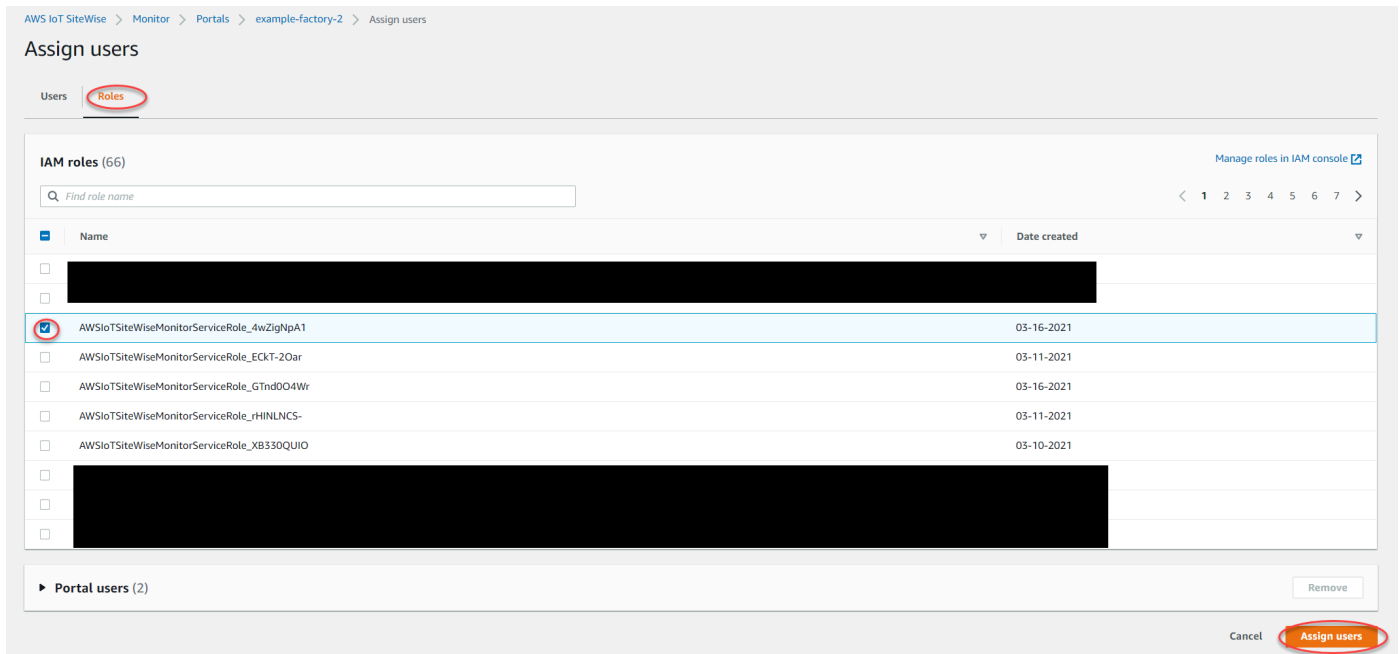
Users Roles

IAM users (1) [Manage users in IAM console](#)

☒	Name	Date created
☒	[REDACTED]	11-08-2019

Portal users (1) [Remove](#)

Cancel **Assign users**



포털 사용자를 제거하려면 다음을 수행하세요.

- 포털 세부 정보 페이지의 포털 사용자 섹션에서 포털에서 제거할 사용자의 확인란을 선택한 다음 포털에서 제거를 선택합니다.

Important

사용자 또는 역할에는 포털에 로그인할 수 있는 `iotsitewise:DescribePortal` 권한이 있어야 합니다.

에서 포털 삭제 AWS IoT SiteWise

테스트 목적으로 포털을 생성했거나 이미 존재하는 포털의 복제본을 생성한 경우 포털을 삭제할 수 있습니다.

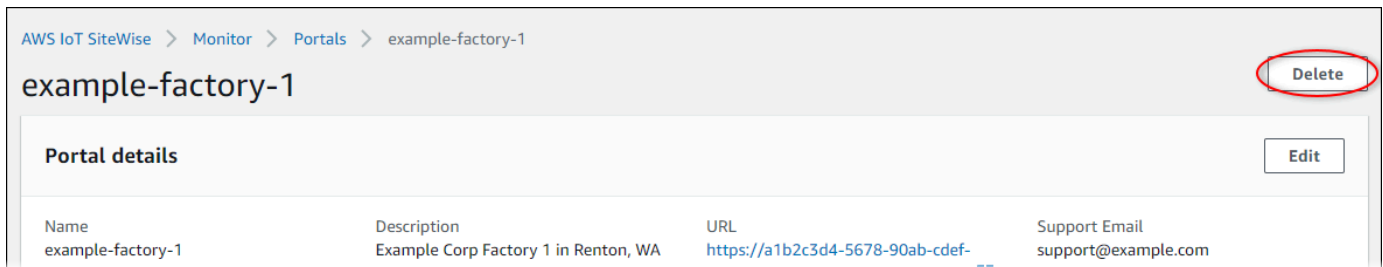
Note

포털을 삭제하려면 먼저 포털의 모든 대시보드와 프로젝트를 수동으로 삭제해야 합니다. 자세한 내용은 SiteWise Monitor 애플리케이션 안내서의 [프로젝트 삭제](#) 및 [대시보드 삭제](#)를 참조하세요.

1. 포털 세부 정보 페이지에서 삭제를 선택합니다.

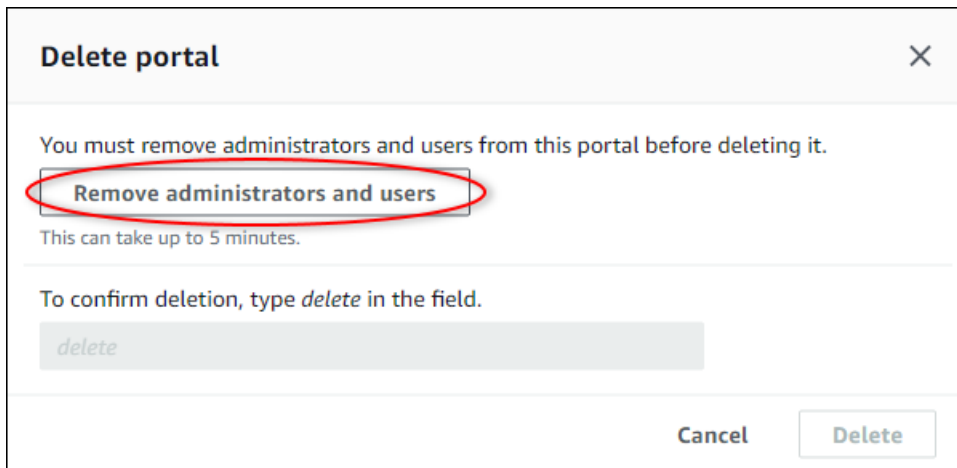
⚠ Important

포털을 삭제하면 포털에 포함된 모든 프로젝트와 각 프로젝트의 모든 대시보드가 손실됩니다. 이 작업은 실행 취소할 수 없습니다. 자산 데이터는 영향을 받지 않습니다.

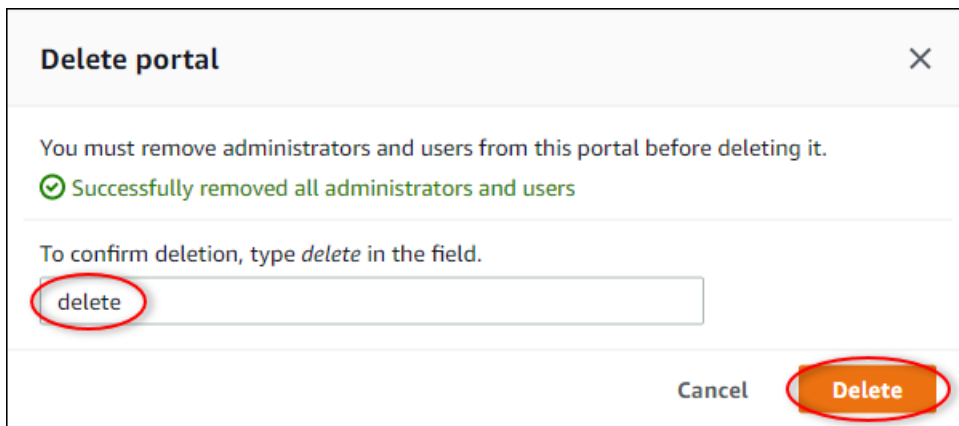


2. 포털 삭제 대화 상자에서 관리자 및 사용자 제거를 선택합니다.

먼저 포털에서 관리자와 사용자를 제거해야 포털을 삭제할 수 있습니다. 포털에 관리자나 사용자가 없는 경우 버튼이 나타나지 않으며 다음 단계로 건너뛸 수 있습니다.



3. 전체 포털을 삭제하려면 필드에 **delete**를 입력하여 삭제를 확인합니다.



4. Delete(삭제)를 선택합니다.

시작하기 AWS IoT SiteWise Monitor (AI 인식) - 미리 보기

조직의 AWS 관리자는 AWS IoT SiteWise 콘솔에서 포털을 생성하여 조직의 구성원이 AWS IoT SiteWise 데이터를 볼 수 있도록 할 수 있습니다. 시작하려면 다음 단계를 완료하세요.

1. 포털을 구성하고 생성합니다.
2. 포털 관리자를 추가하고 초대 이메일을 보냅니다.
3. 포털 사용자를 추가합니다.

포털을 생성한 후 포털 관리자는 프로젝트를 생성하고 프로젝트에 사용자를 추가할 수 있습니다. 그런 다음 프로젝트 멤버는 대시보드를 생성하여 연결된 데이터를 시각화 AWS IoT SiteWise하고 연결된 디바이스, 프로세스 및 장비의 성능을 모니터링할 수 있습니다.

Note

포털에 사용자 또는 관리자를 추가할 때는 제한된 IP와 같이 사용자 권한을 제한하는 AWS Identity and Access Management (IAM) 정책을 생성하지 마세요. 권한이 제한된 연결된 정책은 AWS IoT SiteWise 포털에 연결할 수 없습니다.

팀과 공유할 프로젝트를 생성합니다. 그런 다음 프로젝트 소유자는 대시보드를 생성하여 자산의 속성을 시각화하여 프로젝트 뷰어가 디바이스, 프로세스 및 장비의 성능을 이해하는 데 도움이 됩니다. 또한 팀에 운영에 대한 일관된 보기를 제공합니다.

대시보드는 프로젝트 데이터를 시각화하고 이해하는 데 도움이 됩니다. 이를 통해 비즈니스와 애플리케이션 사용자는 AWS IoT 디바이스와 데이터를 추적할 수 있습니다. 필요에 맞게 데이터를 가장 잘 표시하는 시각화 유형을 선택합니다. 시각화를 재정렬하고 크기를 조정하여 팀에 맞는 레이아웃을 생성합니다. 디바이스, 프로세스 및 장비 자산과 데이터를 탐색하고 문제를 빠르게 식별하고 운영 효율성을 개선합니다.

주제

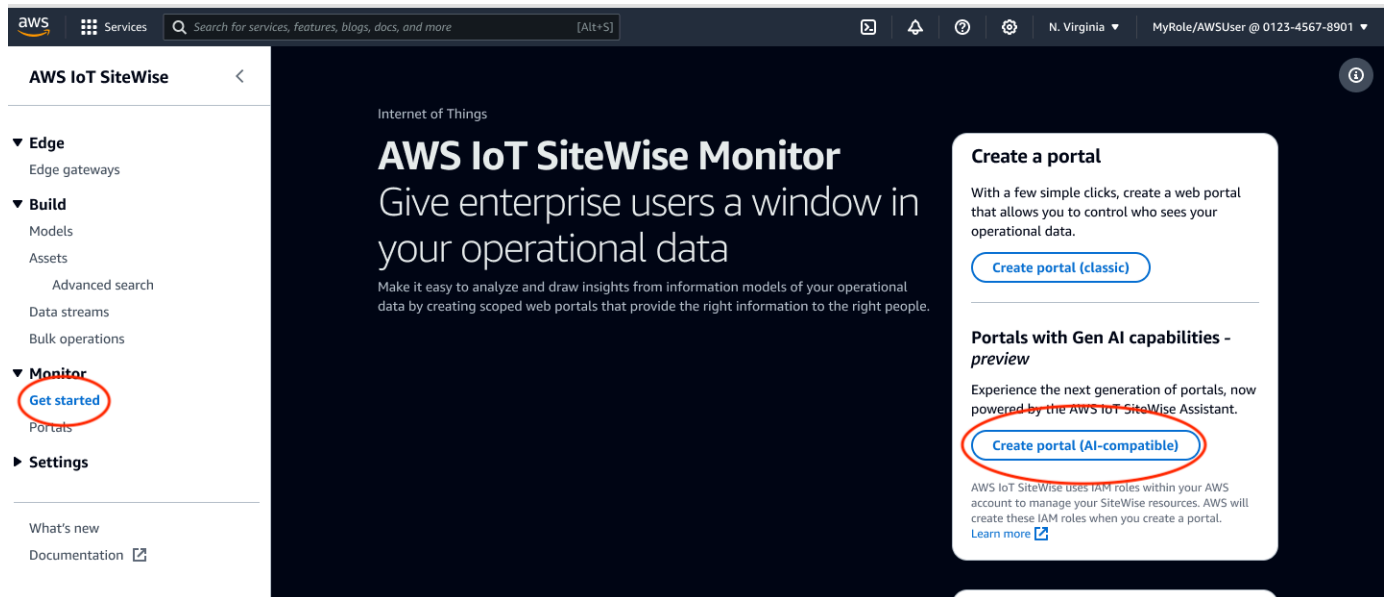
- [포털 생성](#)
- [포털 구성](#)
- [포털 관리](#)
- [포털 삭제](#)
- [를 사용하여 대시보드 생성 AWS CLI](#)
- [포털 로그인](#)
- [프로젝트 생성](#)
- [프로젝트 업데이트](#)
- [프로젝트 삭제](#)
- [대시보드 생성](#)
- [대시보드 업데이트](#)
- [대시보드 삭제](#)
- [대시보드 구성](#)

포털 생성

AWS IoT SiteWise 콘솔에서 SiteWise Monitor 포털을 생성합니다.

포털 생성

1. [AWS IoT SiteWise 콘솔](#)에 로그인합니다.
2. 탐색 창에서 모니터링, 시작하기를 선택합니다.
3. 포털 생성(AI 인식)을 선택합니다.



그런 다음 포털을 구성하기 위한 몇 가지 기본 정보를 입력해야 합니다.

포털 구성

사용자는 포털을 사용하여 데이터를 봅니다. 포털의 이름, 설명, 브랜딩, 사용자 인증, 지원 연락처 이메일 및 권한을 사용자 지정할 수 있습니다.

포털을 구성하는 단계:

1. 프로젝트의 이름을 입력합니다.
2. (선택 사항) 포털에 대한 설명을 입력합니다. 포털이 여러 개인 경우 의미 있는 설명을 사용하여 각 포털에 포함된 내용을 추적할 수 있습니다.
3. (선택 사항) 이미지를 업로드하여 포털에 브랜드를 표시합니다. 정사각형의 PNG 이미지를 선택합니다. 정사각형이 아닌 이미지를 업로드하는 경우 포털에서 이미지의 크기를 정사각형으로 축소합니다.
4. 지원 문의 이메일 상자에 지원 문제에 대한 이메일 주소를 입력합니다.
5. 사용자 인증 상자에서 다음 옵션을 선택합니다.
 - 포털 사용자가 회사 사용자 이름과 암호를 사용하여 이 포털에 로그인하는 경우 IAM Identity Center를 선택합니다.

계정에서 IAM Identity Center를 활성화하지 않은 경우, 다음을 수행하세요.

- a. 사용자 생성을 선택합니다.

- b. 사용자 생성 페이지에서 첫 번째 포털을 생성하려면 사용자의 이메일 주소, 이름, 성을 입력한 다음 사용자 생성을 선택합니다.

Note

IAM 자격 증명에 대한 지원이 곧 제공될 예정입니다.

6. 서비스 액세스 섹션에서 다음 옵션 중 하나를 선택합니다.
 - 새 서비스 역할 생성 및 사용을 선택합니다. 기본적으로 SiteWise Monitor는 각 포털에 대한 서비스 역할을 자동으로 생성합니다. 이 역할을 통해 포털 사용자는 AWS IoT SiteWise 리소스에 액세스할 수 있습니다. 자세한 내용은 [에 대한 서비스 역할 사용 AWS IoT SiteWise Monitor](#) 단원을 참조하십시오.
 - 기존 서비스 역할 사용을 선택하고 대상 역할을 선택합니다.
7. 를 선택하여이 포털에 대해 AWS IoT SiteWise 도우미를 활성화합니다. AWS IoT SiteWise 어시스턴트는 빠른 데이터 분석, 실시간 인사이트 및 안내된 권장 사항을 제공합니다.

Note

AWS IoT SiteWise 어시스턴트를 활성화하면 요금이 발생합니다. 엔터프라이즈급 지식 솔루션 및 지침을 사용하려면 Amazon Kendra 인덱스와 연결된 데이터 세트가 있어야 합니다.

8. (선택 사항) 포털에 대한 태그를 추가합니다. 자세한 내용은 [AWS IoT SiteWise 리소스에 태그 지정](#) 단원을 참조하십시오.
9. 포털 생성을 선택합니다. AWS IoT SiteWise 포털을 생성합니다.

Note

콘솔을 닫으면 관리자와 사용자를 추가하여 설치 프로세스를 완료할 수 있습니다. 자세한 내용은 [포털 관리자 추가 또는 제거](#) 단원을 참조하십시오. 이 포털을 유지용하지 않으려면 리소스를 사용하지 않도록 삭제하세요. 자세한 내용은 [포털 삭제](#) 단원을 참조하십시오.

포털이 생성되면 메시지가 나타납니다.

☑ Successfully created portal "example portal".

×

[AWS IoT SiteWise](#) > [Monitor](#) > [Portals](#) > example portal

example portal

Edit

Delete

Open portal 

Portal details

Name example portal	Status  Active	URL  https://p-jfnlf2d8.gamma.iotsitewise.aws
Type AI-compatible	AWS IoT SiteWise Assistant  Disabled	Portal branding -
Description -	Last updated November 1, 2024, 15:37 (UTC-07:00)	Support contact email myemail@mycompany.com
ID f5fc93a1-011c-4c5a-81a6-e001b50d2547	Date created November 1, 2024, 15:37 (UTC-07:00)	

포털이 생성되면 포털 섹션에 나열됩니다. 포털 세부 정보 섹션에는 각 포털의 이름, 설명, ID, URL, 상태, 마지막으로 업데이트 및 생성된 날짜, 포털 브랜딩 및 지원 이메일이 나열됩니다.

상태 열은 다음 값 중 하나일 수 있습니다.

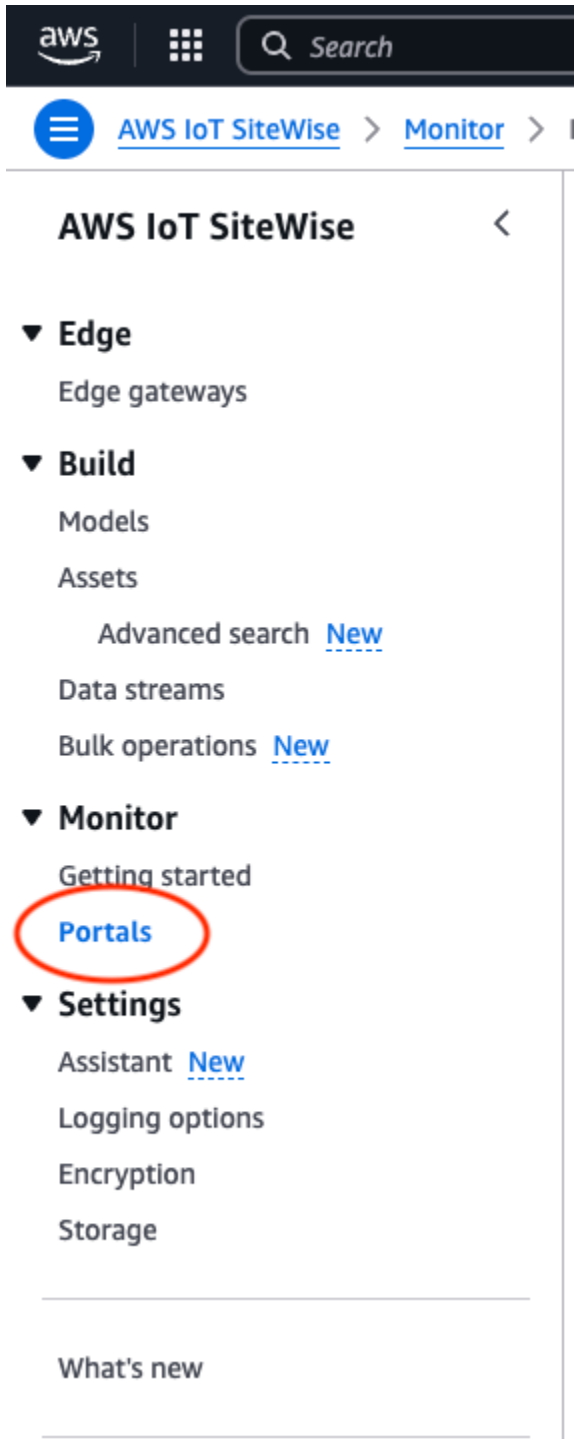
- 생성 AWS IoT SiteWise 중 - 포털 생성 요청을 처리하고 있습니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다.
- 업데이트 중 - 포털 업데이트 요청을 AWS IoT SiteWise 처리하고 있습니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다.
- 대기 중 - DNS 레코드 전파가 완료될 때까지 대기 AWS IoT SiteWise 중입니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다. 대기 중 상태일 때 포털을 삭제할 수 있습니다.
- 삭제 중 - 포털 삭제 요청을 AWS IoT SiteWise 처리하고 있습니다. 이 프로세스는 완료하는 데 몇 분 걸릴 수 있습니다.
- 활성 - 포털이 활성화되면 포털 사용자가 포털에 액세스할 수 있습니다.
- FAILED - 포털 생성, 업데이트 또는 삭제 요청을 처리할 수 AWS IoT SiteWise 없습니다. Amazon CloudWatch Logs AWS IoT SiteWise 로 로그를 전송하도록 활성화한 경우 이러한 로그를 사용하여 문제를 해결할 수 있습니다. 자세한 내용은 [CloudWatch Logs AWS IoT SiteWise 를 사용한 모니터링을 참조하세요](#).

포털 관리

포털의 다양한 측면을 관리하고 구성할 수 있습니다. 여기에는 관리자 추가 및 제거, 권한 및 역할 설정, 이름, 설명 사용자 지정, 지원 이메일 설정, 포털 관리자 초대가 포함됩니다.

1. [AWS IoT SiteWise 콘솔](#)에 로그인합니다.

2. 탐색 창에서 모니터, 포털을 선택합니다.



3. 포털을 선택한 다음 포털 열기(또는 포털의 이름 선택)를 선택합니다.
4. 다음 관리 작업을 수행할 수 있습니다.
 - [포털 속성 편집](#)
 - [포털 관리자 추가 또는 제거](#)

- [포털 관리자에게 이메일 초대 전송](#)
- [에서 포털 삭제 AWS IoT SiteWise](#)

포털 속성 편집

포털의 이름, 설명, 브랜딩, 지원 이메일 및 서비스 액세스를 변경할 수 있습니다.

1. 포털 세부 정보 페이지의 포털 세부 정보 섹션에서 편집을 선택합니다.

☑ Successfully created portal "example portal".

[AWS IoT SiteWise](#) > [Monitor](#) > [Portals](#) > example portal

example portal Edit Delete Open portal

Portal details		
Name example portal	Status Active	URL https://p-jfnlf2d8.gamma.iotsitewise.aws
Type AI-compatible	AWS IoT SiteWise Assistant Disabled	Portal branding -
Description -	Last updated November 1, 2024, 15:37 (UTC-07:00)	Support contact email myemail@mycompany.com
ID f5fc93a1-011c-4c5a-81a6-e001b50d2547	Date created November 1, 2024, 15:37 (UTC-07:00)	

2. 이름, 설명, 포털 브랜딩, 지원 연락처 이메일, AWS IoT SiteWise 도우미 또는 서비스 액세스를 업데이트합니다.
3. 작업을 마쳤으면 변경 내용 저장을 선택합니다.

포털 관리자 추가 또는 제거

몇 단계를 거쳐 포털의 관리자 역할을 할 사용자를 추가하거나 제거할 수 있습니다. 사용자 인증 서비스에 따라 다음 옵션 중 하나를 선택합니다.

IAM Identity Center

Administrators (2) [Info](#) Remove from portal Send invitations Assign administrators

☐	Display name	Email address
☐	Jane Doe	janedoe@amazon.com
☐	John Doe	johndoe@amazon.com

포털 관리자를 추가하려면 다음을 수행하세요.

1. 포털 세부 정보 페이지의 관리자 섹션에서 관리자 할당을 선택합니다.
2. 관리자 할당 페이지에서 포털에 관리자로 추가할 사용자를 선택합니다.

Note

IAM Identity Center를 자격 증명 스토어로 사용하고 AWS Organizations 관리 계정에 로그인한 경우 사용자 생성을 선택하여 IAM Identity Center 사용자를 생성할 수 있습니다. IAM Identity Center는 새 사용자에게 비밀번호를 설정하라는 이메일을 보냅니다. 그런 다음 사용자를 포털에 관리자로 할당할 수 있습니다. 자세한 내용은 [IAM Identity Center에서 자격 증명 관리](#) 섹션을 참조하세요.

3. 관리자 할당을 선택합니다.

Assign administrators [Info](#)

Choose portal administrators from the users list. Portal administrators grant access to specific industrial equipment data. [Learn more](#)

Users (11)

Create user

2 matches

< 1 > ⚙

☐	Name	Email
☒	johndoe@amazon.com	johndoe@amazon.com
☐	janedoe@amazon.com	janedoe@amazon.com

[Cancel](#)
[Assign administrators](#)

포털 관리자를 제거하려면 다음을 수행하세요.

- 포털 세부 정보 페이지의 포털 관리자 섹션에서 제거할 각 사용자의 확인란을 선택한 다음 포털에서 제거를 선택합니다.

Note

관리자(#)에는 포털의 관리자 수가 나열됩니다. 여러 포털 관리자를 추가하여 프로젝트를 관리하고 작업할 수 있습니다.

포털 관리자에게 이메일 초대 전송

포털 관리자에게 이메일 초대를 보낼 수 있습니다.

1. 포털 세부 정보 페이지의 관리자 섹션에서 포털 관리자의 확인란을 선택합니다.
2. 초대 보내기를 선택합니다. 이메일 클라이언트가 열리고 메시지 본문에 초대장이 채워집니다.

이메일을 포털 관리자에게 보내기 전에 사용자 지정할 수 있습니다.

포털 삭제

테스트 목적으로 포털을 생성했거나 이미 존재하는 포털의 복제본을 생성한 경우 포털을 삭제할 수 있습니다.

Note

포털을 삭제하려면 먼저 포털의 모든 대시보드와 프로젝트를 수동으로 삭제해야 합니다.

1. 포털 세부 정보 페이지에서 삭제를 선택합니다.

⚠ Important

포털을 삭제하면 포털에 포함된 모든 프로젝트와 각 프로젝트의 모든 대시보드가 손실됩니다. 이 작업은 실행 취소할 수 없습니다. 자산 데이터는 영향을 받지 않습니다.

The screenshot shows the AWS IoT SiteWise Mezzanine interface. On the left is a navigation menu with sections: Edge (Edge gateways), Build (Models, Assets, Advanced search, Data streams, Bulk operations), and Monitor (Get started). The main content area is titled 'Portals (3)' and includes buttons for 'Open portal', 'Edit', 'Delete' (highlighted with a red box), and 'Create portal'. Below the buttons is a search bar and a table of portals.

	Name	Status	Link	Type	Assistant
☐	Portal1	Updating	https://p-mrr5m4bb.app.iotsitewise.aws	Classic	-
☐	Portal2	Failed	https://p-w6zwmryz.app.iotsitewise.aws	Classic	-
☒	BikeFactory	Active	https://p-f5r29mr5.app.iotsitewise.aws	AI-compatible	Enabled

2. 포털 삭제 대화 상자에서 관리자 및 사용자 제거를 선택합니다.

먼저 포털에서 관리자와 사용자를 제거해야 포털을 삭제할 수 있습니다. 포털에 관리자나 사용자가 없는 경우 버튼이 나타나지 않으며 다음 단계로 건너뛸 수 있습니다.

3. 전체 포털을 삭제하려면 필드에 **confirm**를 입력하여 삭제를 확인합니다.
4. Delete(삭제)를 선택합니다.

를 사용하여 대시보드 생성 AWS CLI

를 사용하여 대시보드에서 시각화(또는 위젯)를 정의할 때 dashboardDefinition JSON 문서에 다음 정보를 지정 AWS CLI해야 합니다. 이 정의는 [CreateDashboard](#) 및 [UpdateDashboard](#) 작업의 파라미터입니다.

displaySettings

다음 파라미터가 포함된 표시 설정:

- numRows - 대시보드 레이아웃의 행 수입니다. 각 행은 cellSize 너비입니다.
- numColumns - 대시보드 레이아웃의 열 수입니다. 각 열은 cellSize 너비입니다.
- cellSize - (선택 사항) 레이아웃에 있는 셀의 크기를 픽셀 단위로 표시합니다. 양수여야 합니다. 기본값은 10입니다.
- significantDigits - (선택 사항) 대시보드에 표시할 유효 자릿수입니다. 기본값은 4입니다.

querySettings

다음 파라미터가 포함된 쿼리 정보입니다.

- refreshRate - (선택 사항) 밀리초 단위로 데이터를 새로 고치는 속도입니다. 1000, 5000, 10000, 60000, 300000 값을 허용합니다.

defaultViewport

입력하지 않으면 기본값은 지난 5분입니다. 다음 파라미터를 포함합니다.

- duration - (선택 사항) 현재 시간부터 데이터를 쿼리할 과거의 거리를 결정합니다.
- start - (선택 사항) Date 유형입니다. 데이터를 쿼리할 시작 시간 범위입니다. end 날짜를 지정해야 합니다.
- end - (선택 사항) Date 유형입니다. 데이터를 쿼리할 종료 시간 범위입니다. start 날짜를 지정해야 합니다.

widgets

다음 정보가 포함된 위젯 정의 구조의 목록입니다.

type

위젯 유형은 다음과 같은 위젯 유형을 AWS IoT SiteWise 제공합니다.

- `xy-plot` - 구성에 따라 선 차트 또는 산점도입니다.
- `bar-chart` - 막대 차트입니다.
- `kpi-chart` - 주요 성과 지표 차트입니다.
- `status-timeline` - 하나 이상의 데이터 소스에서 시계열 데이터를 시각화하고 탐색하는 상태 위젯입니다.
- `text` - 텍스트 위젯입니다.
- `table` - 테이블 위젯입니다.

id

위젯의 고유 식별자입니다.

x

대시보드 왼쪽부터 위젯의 가로 위치입니다. 이 값은 대시보드 그리드에서 위젯의 위치를 나타냅니다.

y

대시보드 상단부터 위젯의 세로 위치입니다. 이 값은 대시보드 그리드에서 위젯의 위치를 나타냅니다.

z

위젯의 상대적 순서입니다. Z 값 위젯이 겹치는 경우 더 큰 Z 값 위젯이 더 낮은 Z 값 위젯 앞에 표시됩니다.

width

대시보드의 셀 수로 표시되는 위젯의 너비입니다.

height

대시보드의 셀 수로 표시되는 위젯의 높이입니다.

properties

위젯의 속성 목록입니다. 위젯 유형에 따라 다릅니다. 자세한 내용은 [IoT 앱 키트를](#) 참조하세요.

Example 대시보드 정의 예

다음 예제에서는 JSON 파일에 저장된 페이로드로 대시보드를 정의합니다.

```
aws iotsitewise create-dashboard \  
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeeEXAMPLE \  
  --dashboard-name "Example Dashboard" \  
  --dashboard-definition file://dashboard-definition.json
```

dashboard-definition.json에 대한 다음 JSON 예제는 다음과 같은 시각화 위젯으로 대시보드를 정의합니다.

```
{  
  "displaySettings": {  
    "numColumns": 200,  
    "numRows": 1000,  
    "cellSize": 20,  
    "significantDigits": 4  
  },  
  "widgets": [{  
    "id": "0t73JcxUoc6oEXAMPLE",  
    "type": "xy-plot",  
    "width": 33,  
    "height": 20,  
    "x": 0,  
    "y": 0,  
    "z": 0,  
    "properties": {  
      "aggregationType": "AVERAGE",  
      "queryConfig": {  
        "source": "iotsitewise",  
        "query": {  
          "assets": [{  
            "assetId": "97c97abf-e883-47bb-a3f4-EXAMPLE",  
            "properties": [{  
              "propertyId": "97cc61f4-57a4-4c5f-a82c-EXAMPLE",  
              "refId": "692ce941-f3d9-4074-a297-EXAMPLE",  
            }]  
          }  
        }  
      }  
    }  
  }  
}]
```

```

        "aggregationType": "AVERAGE",
        "color": "#7d2105",
        "resolution": "1m"
    }
  ],
  "properties": [],
  "assetModels": [],
  "alarms": [],
  "alarmModels": []
}
},
"line": {
  "connectionStyle": "linear",
  "style": "solid"
},
"symbol": {
  "style": "filled-circle"
},
"axis": {
  "yVisible": true,
  "xVisible": true
},
"legend": {
  "visible": true,
  "position": "right",
  "width": "30%",
  "height": "30%",
  "visibleContent": {
    "unit": true,
    "asset": true,
    "latestValue": true,
    "latestAlarmStateValue": true,
    "maxValue": false,
    "minValue": false
  }
}
}
}, {
  "id": "fto7rF40Ny1EXAMPLE-G",
  "type": "bar-chart",
  "width": 33,
  "height": 20,
  "x": 0,
  "y": 20,

```

```

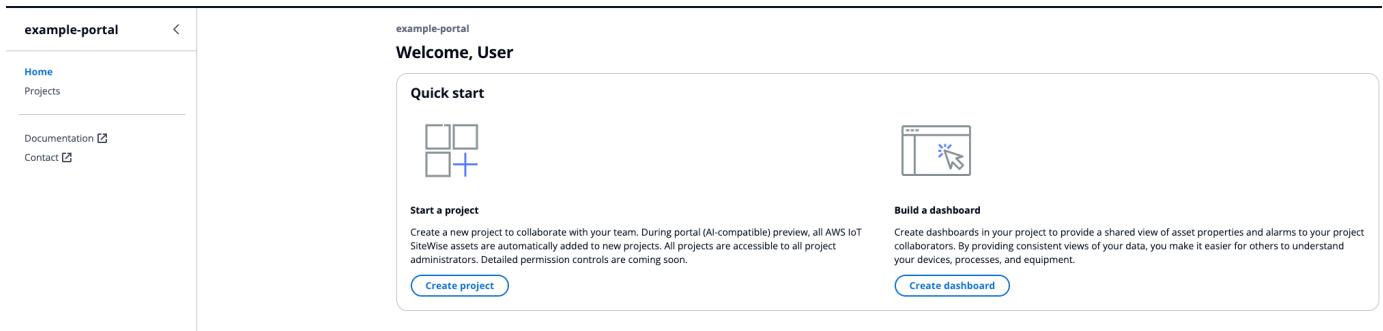
    "z": 0,
    "properties": {
      "aggregationType": "AVERAGE",
      "queryConfig": {
        "source": "iotsitewise",
        "query": {
          "assets": [{
            "assetId": "97c97abf-e883-47bb-a3f4-EXAMPLE",
            "properties": [{
              "propertyId": "c84ca8f3-3dea-478a-afec-EXAMPLE",
              "aggregationType": "AVERAGE",
              "refId": "2960b958-2034-4d6e-bcc2-EXAMPLE"
            }]
          }],
          "properties": [],
          "assetModels": [],
          "alarms": [],
          "alarmModels": [],
          "requestSettings": {
            "aggregation": "AVERAGE"
          }
        }
      },
      "axis": {
        "showX": true,
        "showY": true
      },
      "styleSettings": {
        "2960b958-2034-4d6e-bcc2-360f1f02e505": {
          "color": "#7d2105"
        }
      }
    }
  },
  "querySettings": {
    "refreshRate": 5000
  }
}

```

포털 로그인

사용자 로그인

1. 브라우저에서 애플리케이션 URL을 입력합니다.
2. 사용자 이름과 암호를 입력하고 로그인 버튼을 클릭합니다.
3. 이제 애플리케이션에 로그인했습니다.



프로젝트 생성

프로젝트 생성

1. 프로젝트는 두 가지 방법으로 생성됩니다.
 - a. 홈 페이지의 빠른 시작 아래의 시작 섹션에서 프로젝트 생성을 선택합니다.
 - b. 왼쪽 탐색 창에서 프로젝트를 선택합니다. 오른쪽 상단 모서리에서 생성을 선택하여 프로젝트를 생성합니다.
2. 프로젝트 생성 섹션에서 프로젝트 이름을 입력하고 선택적 설명을 제공합니다.
3. 생성(Create)을 선택합니다.

example-portal

example-portal <

Home
Projects

Documentation
Contact

Home > Projects > Create

Create project

Create a project to host your dashboards.

Project name
Example project
The project name must contain from 1 and 256 characters. Character count: 15/256.

Project description - optional
An example project.
The project description must contain from 1 and 2048 characters. Character count: 19/2048.

Cancel **Create**

프로젝트 업데이트

프로젝트 편집

1. 프로젝트 페이지의 오른쪽 상단에 있는 업데이트 버튼을 선택하여 프로젝트 세부 정보를 편집합니다.
2. 프로젝트 이름을 편집하여 프로젝트 이름을 변경합니다.
3. 설명 세부 정보를 편집하여 프로젝트의 설명을 변경합니다.
4. 업데이트를 선택하여 변경 사항을 저장합니다.

example-portal

example-portal <

Home
Projects

Documentation
Contact

Home > Projects > Example-project

Example-project

Project details

Description	Creation date	Last update date
Example project	10/25/2024, 11:17:28 AM	10/25/2024, 11:17:28 AM

Dashboards

Filter dashboards

Name	Description	Creation date	Last update date
example dashboard	dashboard	10/25/2024, 11:17:52 AM	10/25/2024, 11:17:52 AM

Delete Update **Create**

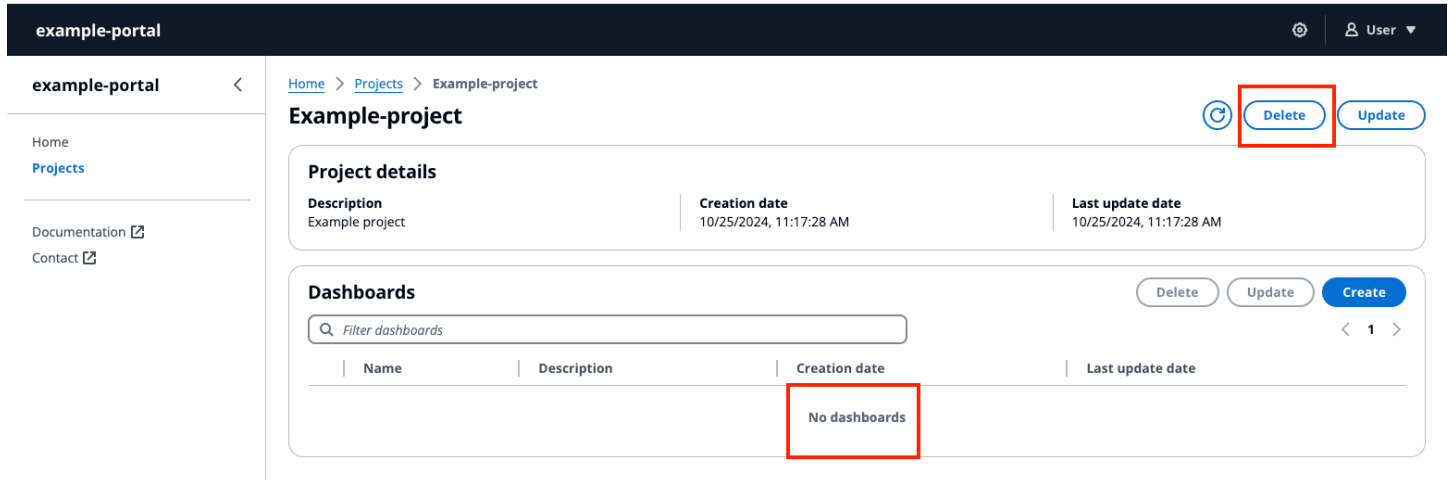
< 1 >

프로젝트 삭제

프로젝트 삭제

1. 프로젝트의 모든 대시보드가 삭제된 후에만 프로젝트를 삭제할 수 있습니다.

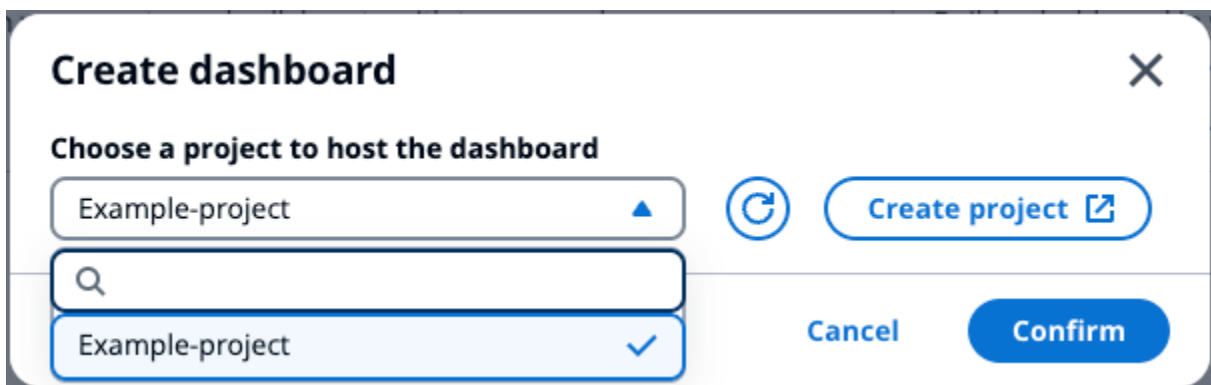
2. 프로젝트 페이지의 오른쪽 상단에서 삭제 버튼을 선택합니다.
3. 프로젝트를 삭제할 것인지 다시 확인합니다.
4. 삭제를 선택하여 프로젝트를 삭제합니다.



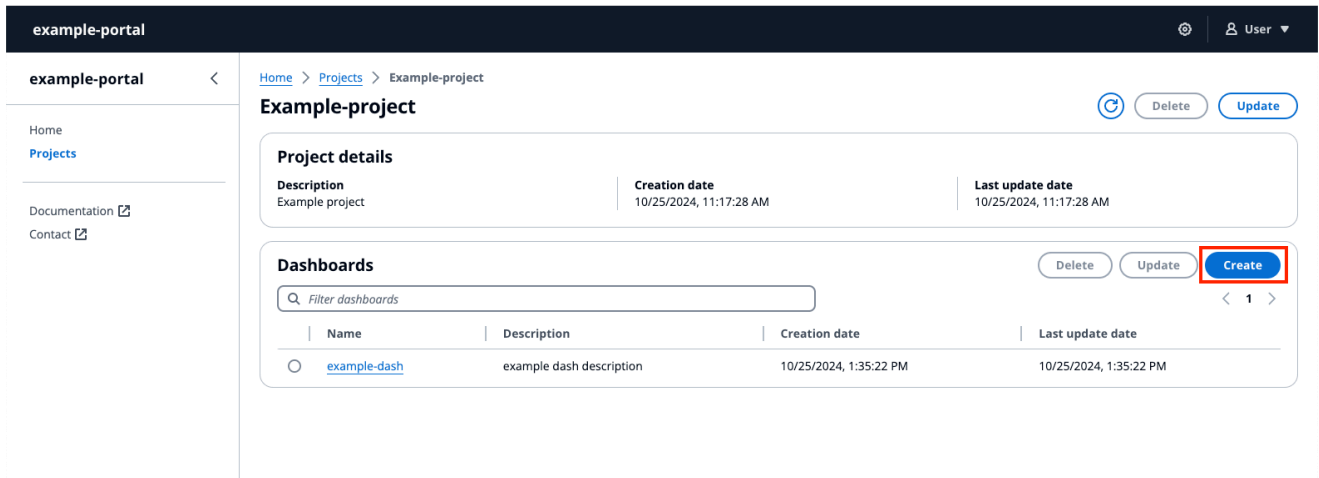
대시보드 생성

대시보드 생성

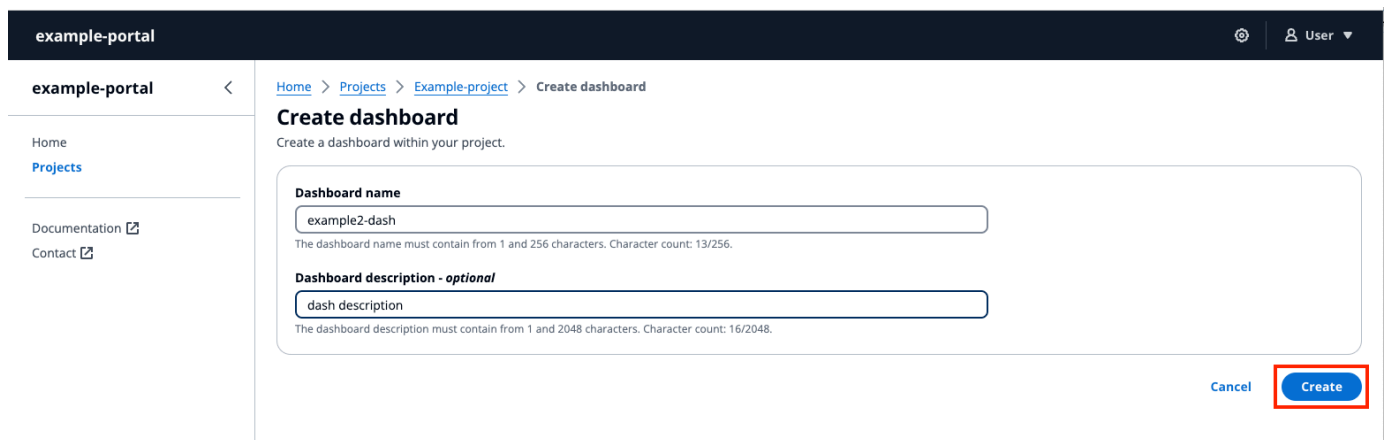
1. 두 가지 방법으로 대시보드를 생성합니다.
 - a. 홈 페이지의 대시보드 구축에서 대시보드를 생성합니다.
 - i. 기존 프로젝트에서 대시보드를 생성하려면 대시보드를 호스팅할 프로젝트 선택의 드롭 다운 메뉴에서 프로젝트 이름을 선택합니다.
 - ii. 프로젝트가 없는 경우 프로젝트 생성을 선택하고 확인을 선택합니다.



- b. 프로젝트 섹션의 대시보드 아래에 있는 프로젝트에서 대시보드를 생성합니다.



- 오른쪽 상단 모서리에서 생성을 선택합니다.
- 대시보드 이름을 입력하고 선택적 대시보드 설명을 제공합니다.
- 생성을 선택합니다.



- 새로 생성된 대시보드를 구성합니다.

대시보드 업데이트

대시보드 섹션에는 프로젝트의 대시보드가 나열됩니다. 목록에서 대시보드를 선택합니다.

대시보드 업데이트

- 업데이트할 대시보드를 선택합니다.

example-portal

Home > Projects > Example-project

Example-project

Project details

Description Example project	Creation date 10/25/2024, 11:17:28 AM	Last update date 10/25/2024, 11:17:28 AM

Dashboards

Filter dashboards

Name	Description	Creation date	Last update date
example-dash	example dash description	10/25/2024, 1:35:22 PM	10/25/2024, 1:35:22 PM

2. 대시보드 이름과 선택적으로 대시보드 설명을 업데이트합니다. 업데이트를 선택하여 변경 사항을 저장합니다.

Home > Projects > example-dash > Update

Update dashboard: example-dash

Dashboard name

example-update-dash-name

The dashboard name must contain from 1 and 256 characters. Character count: 24/256.

Dashboard description - optional

update dash description

The dashboard description must contain from 1 and 2048 characters. Character count: 23/2048.

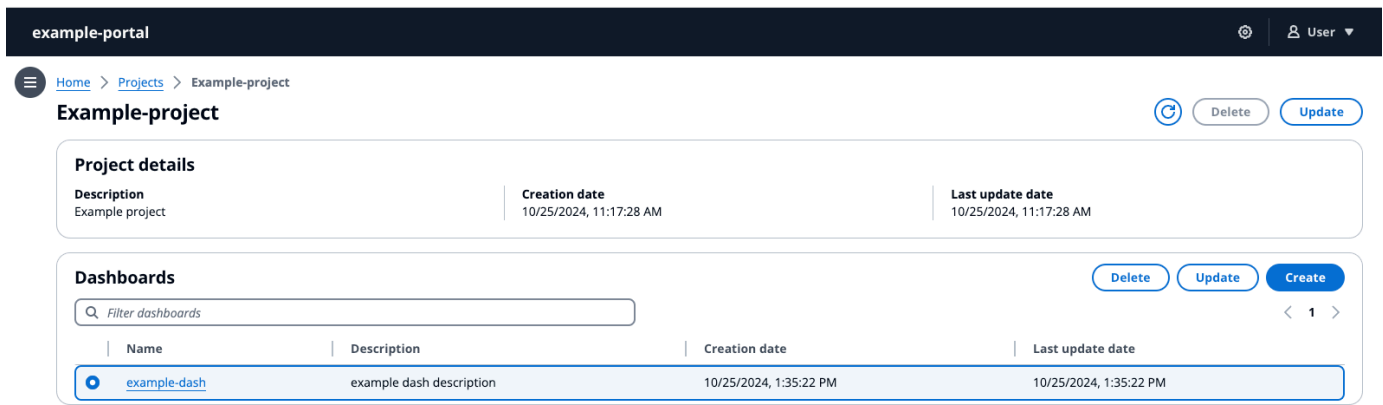
Cancel Update

대시보드 삭제

대시보드 섹션에는 프로젝트의 대시보드가 나열됩니다. 목록에서 대시보드를 선택합니다.

대시보드 삭제

1. 삭제할 대시보드를 선택합니다.



2. 삭제를 선택하여 대시보드를 삭제합니다. 실행 취소할 수 없습니다.

대시보드 구성

대시보드 섹션에는 프로젝트의 대시보드가 나열됩니다. 목록에서 대시보드를 선택합니다. 편집 모드를 사용하면 위젯을 추가하고 구성하여 대시보드를 구성할 수 있습니다. 미리 보기 버튼을 사용하면 변경 사항을 시각화할 수 있습니다.



대시보드를 구성하는 단계:

- 데이터 시각화를 위해 다양한 유형의 데이터 위젯을 대시보드 캔버스로 끌어서 놓습니다.
- 왼쪽의 리소스 탐색기에서 원하는 위젯에 데이터를 추가합니다. 리소스 탐색기는 모델링된 자산, 모델링되지 않은 자산 및 동적 자산 섹션으로 구성됩니다. 자산 이름 또는 속성 이름으로 검색합니다. 추가할 속성을 선택하고 추가를 선택합니다.
- 위젯의 구성을 변경하여 레이아웃과 스타일을 미세 조정합니다. 제목, 임계값 및 기타 구성 세부 정보를 포함한 구성 요소를 구성합니다.
- 데이터가 표시되는 시간 범위를 구성합니다.
 - 데이터가 표시되는 시간 범위를 선택합니다. 오른쪽 상단 모서리에서 시간 범위와 새로 고침 빈도를 선택하고 범위를 개인 설정합니다. 메뉴에서 데이터를 새로 고칠 비율을 선택합니다.

- 트랙볼 마우스 스크롤 휠을 사용하거나 마우스 오른쪽 버튼을 클릭하여 위젯에서 시간 범위를 선택합니다. 이렇게 하면 표시 시간 범위가 이동합니다.
- 저장(Save)을 선택합니다.

주제

- [리소스 탐색기](#)
- [위젯](#)
- [위젯 구성](#)
- [위젯 사용](#)
- [위젯의 경보](#)
- [AWS IoT SiteWise 위젯에서 어시스턴트 사용](#)
- [AWS IoT SiteWise 도우미에게 물어볼 샘플 질문](#)

리소스 탐색기

이 섹션에서는 모델링된 자산, 모델링되지 않은 자산 및 동적 자산을 설명합니다. 세 가지 자산 중 하나를 선택하여 위젯에 추가하고 시각화합니다.

주제

- [모델링됨](#)
- [모델링되지 않음](#)
- [동적 자산](#)

모델링됨

이 섹션에서는 모델링된 자산을 선택하고 시각화하는 프로세스를 설명합니다.

자산 선택

자산은 다음과 같이 쿼리할 수 있습니다.

- 자산 이름을 검색합니다. 와일드카드를 사용합니다*. 예를 들어는 텍스트 로 시작하는 자산 이름을 Wind* 반환합니다Wind. 이 기능을 사용하려면 [와 통합 AWS IoT TwinMaker](#)해야 합니다.
- 기본적으로 모든 자산이 나열됩니다.

나열된 자산에서 이름, 설명, ID 또는 자산 모델 ID를 기준으로 필터링합니다. 자산 하나를 선택하여 속성(데이터 스트림)과 경보를 나열합니다.

데이터 스트림 선택

데이터 스트림은 데이터 스트림 메뉴 아래에 나열됩니다. <https://docs.aws.amazon.com/iot-sitewise/latest/APIReference/> [속성](#) 메타데이터별로 나열된 데이터 스트림을 필터링합니다. 선택한 위젯에 따라 데이터 스트림을 하나 이상 선택합니다.

- KPI 및 게이지는 단일 데이터 스트림만 지원합니다.
- 나머지 위젯은 다중 선택으로 여러 데이터 스트림을 지원합니다.

경보 선택

AWS IoT SiteWise 경보는 경보 데이터 스트림 메뉴 아래에 나열됩니다. 경보 메타데이터별로 나열된 경보 데이터 스트림을 필터링합니다. 이름, 입력 속성 및 복합 모델 ID는 필터링에 사용되는 일부 메타데이터입니다. 선택한 위젯에 따라 데이터 스트림을 하나 이상 선택합니다.

- KPI 및 게이지는 단일 경보만 지원합니다.
- 나머지 위젯은 다중 선택으로 여러 경보를 지원합니다.

모델링된 자산 시각화

1. 위젯을 캔버스로 드래그합니다. 각 위젯 패널의 속성을 선택하여 대시보드를 구성합니다.
2. 필터 옵션은 자산을 필터링하여 시각화할 자산을 선택합니다. 필터링은 텍스트, 속성 또는 값을 기준으로 수행됩니다. 필터링은 백엔드 필터링이 아닌 브라우저에 로드된 자산에 대한 것입니다.
3. 위젯에 추가할 자산을 검색하여 나열합니다.
4. 캔버스의 위젯에 자산을 추가합니다.
5. 재설정을 선택하여 다른 자산을 선택하거나 선택한 자산을 수정합니다.
6. 대시보드를 저장합니다. 미리 보기 모드의 드롭다운 메뉴에서 다른 자산을 선택하여 데이터 패널을 재구성하지 않고 각 자산의 속성을 모니터링합니다.

 Note

오른쪽의 구성 설정 휠에는 페이지 크기, 첫 번째 열 고정, 마지막 열 고정 및 열 기본 설정과 같이 사용자가 선택할 수 있는 기본 설정이 표시됩니다. 기본 설정을 사용자 지정하고 확인을 선택하여 변경 사항을 적용합니다.

Modeled

Unmodeled

Dynamic assets

Assets (1)

Browse through your asset hierarchy and select an asset to view its associated data streams.

Root

Search

Search

Filter

< 1 > ⚙

Name



Description



Demo Wind Farm Asset

Asset properties (8)

Select a modeled datastream to add to a selected widget

Filter

< 1 > ⚙



Name



Unit



Data ty...

Latest ...

Latest value time



Total Aver...

DOUBLE

37478.2303

2024-10-01 09:50:00 p.m.



Total Aver...

DOUBLE

6.0000

2022-10-26 03:42:43 p.m.



Total Aver...

DOUBLE

555.0000

2022-10-13 11:59:49 p.m.



Code

INTEGER

300.0000

2022-10-13 10:59:28 p.m.



Reliability ...

STRING

Mary Major

2022-10-13 10:59:28 p.m.



Location

STRING

Renton

2022-10-13 10:59:28 p.m.



Total Over...

DOUBLE

900.0000

2024-10-01 09:50:00 p.m.



recipient a...

STRING

54a88418-...

2022-10-26 03:42:43 p.m.

모델링되지 않음

이 섹션에서는 모델링되지 않은 데이터 스트림을 검색하고 시각화할 위젯에 추가하는 방법을 설명합니다.

모델링되지 않은 데이터 스트림 시각화

1. 위젯을 캔버스로 드래그합니다. 각 위젯 패널의 속성을 선택하여 대시보드를 구성합니다.
2. 모델링되지 않은 데이터 스트림은 시계열 섹션 아래에 나열됩니다. 사용자 지정 가능한 속성이 있습니다.
3. 필터 옵션은 시각화할 데이터 스트림을 필터링합니다. 필터링은 백엔드 필터링이 아닌 브라우저에 로드된 데이터 스트림에 대한 것입니다.
4. 캔버스의 위젯에 데이터 스트림을 추가합니다.
5. 재설정을 선택하여 데이터 스트림을 선택 취소합니다.
6. 대시보드를 저장합니다. 미리 보기 모드의 드롭다운 메뉴에서 다른 자산을 선택하여 데이터 패널을 재구성하지 않고 각 자산의 속성을 모니터링합니다.

Note

오른쪽의 구성 설정 휠에는 페이지 크기, 첫 번째 열 고정, 마지막 열 고정 및 열 기본 설정과 같이 사용자가 선택할 수 있는 기본 설정이 표시됩니다. 기본 설정을 사용자 지정하고 확인을 선택하여 변경 사항을 적용합니다.

Modeled
Unmodeled
Dynamic assets

Time series (1)

Select a unmodeled datastream to add to a selected widget

Filter
< 1 > ⚙

☒	Alias	ID	Data type	Latest val...	Latest valu...
☒	DemoDisass...	33e78bb9-39...	DOUBLE	3.0283	2024-10-01 0...

동적 자산

새로운 SiteWise Monitor를 사용하면 고객이 선택한 자산 모델의 자산을 동적으로 전환할 수 있습니다. 드롭다운 메뉴에서 선택하여 다양한 자산의 속성을 시각화할 수 있습니다.

동적 자산 시각화

1. 리소스 탐색기에서 동적 자산 탭을 선택합니다.
2. 드롭다운 메뉴에서 자산을 나열할 자산 모델을 선택합니다.
3. 드롭다운 메뉴에서 기본 자산을 선택합니다.
4. 자산 모델 설정을 선택하여 자산 모델을 선택합니다.
5. 대시보드를 저장합니다. 미리 보기 모드의 드롭다운 메뉴에서 다른 자산을 선택하여 데이터 패널을 재구성하지 않고 각 자산의 속성을 모니터링합니다.

Resource explorer

Modeled | Unmodeled | **Dynamic assets**

Dynamic asset visualizations allow you to build one visualization to represent any asset of a specified asset model. [Learn more](#)

Asset model
Select an asset model to add the associated properties into your dynamic display.

Demo Wind Farm Asset Model 2022-Oct-14 05:59:28

Q Filter asset models

Demo Turbine Asset Model 2022-Oct-14 05:59:12
This is an asset model used in the IoT SiteWise Demo for representing a turbine in a wind farm. It will be deleted at the end of the demo.

Demo Wind Farm A Demo Turbine Asset Model 2022-Oct-14 05:59:12
This is an asset model used in the IoT SiteWise Demo for representing a wind farm. It will be deleted at the end of the demo. ✓

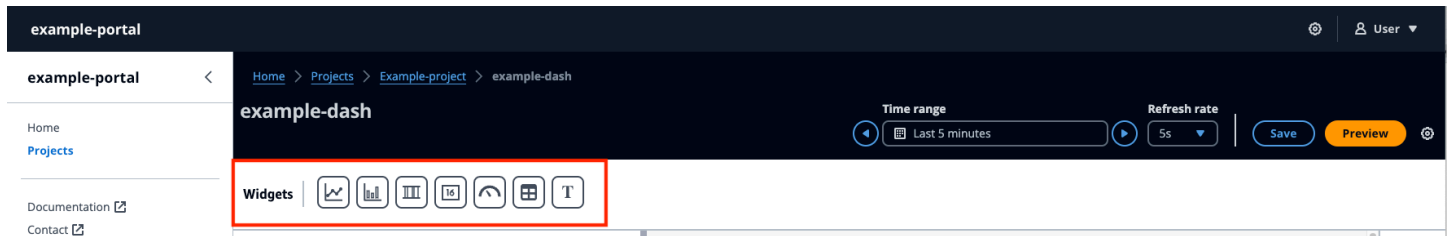
Finished loading asset models.

Set asset model

위젯

위젯은 경보, 고성능 라이브 스트리밍, 다른 IoT App Kit 구성 요소와의 원활한 동기화를 비롯한 다양한 기능을 지원합니다. 대시보드는 다음 위젯을 지원합니다.

- **선** - 선 위젯은 시간 경과에 따른 추세와 변화를 표시하는 시각화 위젯입니다. 이는 일련의 데이터 포인트로 구성되며, 각 데이터 포인트는 점 또는 마커로 표현되고 직선 세그먼트로 연결되어 선 그래프를 생성합니다. 경보, 임계값, 고성능 라이브 스트리밍, 다른 IoT App Kit 구성 요소와의 원활한 동기화 등 다양한 기능을 지원합니다. 이 위젯은 복잡한 데이터를 명확하고 간결하게 전달하도록 사용자 지정할 수 있습니다.
- **막대 차트** - 막대 차트는 시계열 데이터를 표시하는 강력한 시각화 도구입니다. 경보, 고성능 라이브 스트리밍, 다른 IoT App Kit 구성 요소와의 원활한 동기화 등 다양한 기능을 지원합니다.
- **타임라인** - 타임라인 위젯은 데이터 소스에서 시계열 데이터를 시각화하고 탐색하는 방법을 제공합니다. 타임라인에 데이터 스트림 값이 고유한 색상으로 표시되는 데 고유합니다. 경보, 고성능 라이브 스트리밍, 다른 IoT App Kit 구성 요소 간의 원활한 동기화를 비롯한 다양한 기능을 지원합니다. 숫자가 아닌 데이터 형식을 표시하는 데 가장 적합합니다.
- **KPI** - 핵심 성과 지표(KPI) 구성 요소는 자산 속성의 개요를 간결하게 표현합니다. 경보 및 임계값을 지원합니다. 이 개요에서는 디바이스, 장비 및 프로세스의 전반적인 성능에 대한 중요한 인사이트를 제공합니다. KPI는 단일 데이터 스트림 또는 경보만 지원하며 여러 데이터 스트림은 지원하지 않습니다.
- **게이지** - 게이지 구성 요소는 자산 속성의 개요를 간결하게 표현합니다. 디바이스, 장비 또는 프로세스의 전반적인 성능에 대한 중요한 인사이트를 시각화하는 데 사용됩니다. KPI와 기능적으로 동일하지만 시각적으로 다릅니다. 게이지는 데이터 스트림 값, 임계값 및 값 범위를 표시합니다. 게이지를 사용하여 하나 이상의 AWS IoT 데이터 소스의 데이터와 상호 작용할 수 있습니다.
- **테이블** - 테이블 구성 요소는 하나 이상의 시계열 데이터 소스에서 하나 이상의 데이터 스트림을 볼 수 있는 간단한 양식을 제공합니다. 속성, 최신 값 및 단위가 있는 자산을 표 형식으로 표시합니다. AWS IoT SiteWise 경보를 지원합니다.
- **텍스트** - 텍스트 위젯을 사용하면 다양한 색상과 글꼴로 텍스트를 작성할 수 있습니다. 텍스트를 URL과 연결하여 링크를 생성할 수 있습니다. 이 위젯에는 속성 및 임계값 필드가 활성화되지 않습니다.



위젯 구성

위젯이 대시보드에 추가되면 오른쪽 패널에서 구성 아이콘을 선택하여 위젯을 구성할 수 있습니다.

- 스타일 - 위젯 제목에 제목을 추가합니다. 위젯마다 구성이 다릅니다. 다음은 몇 가지 예입니다.

- 막대 위젯:

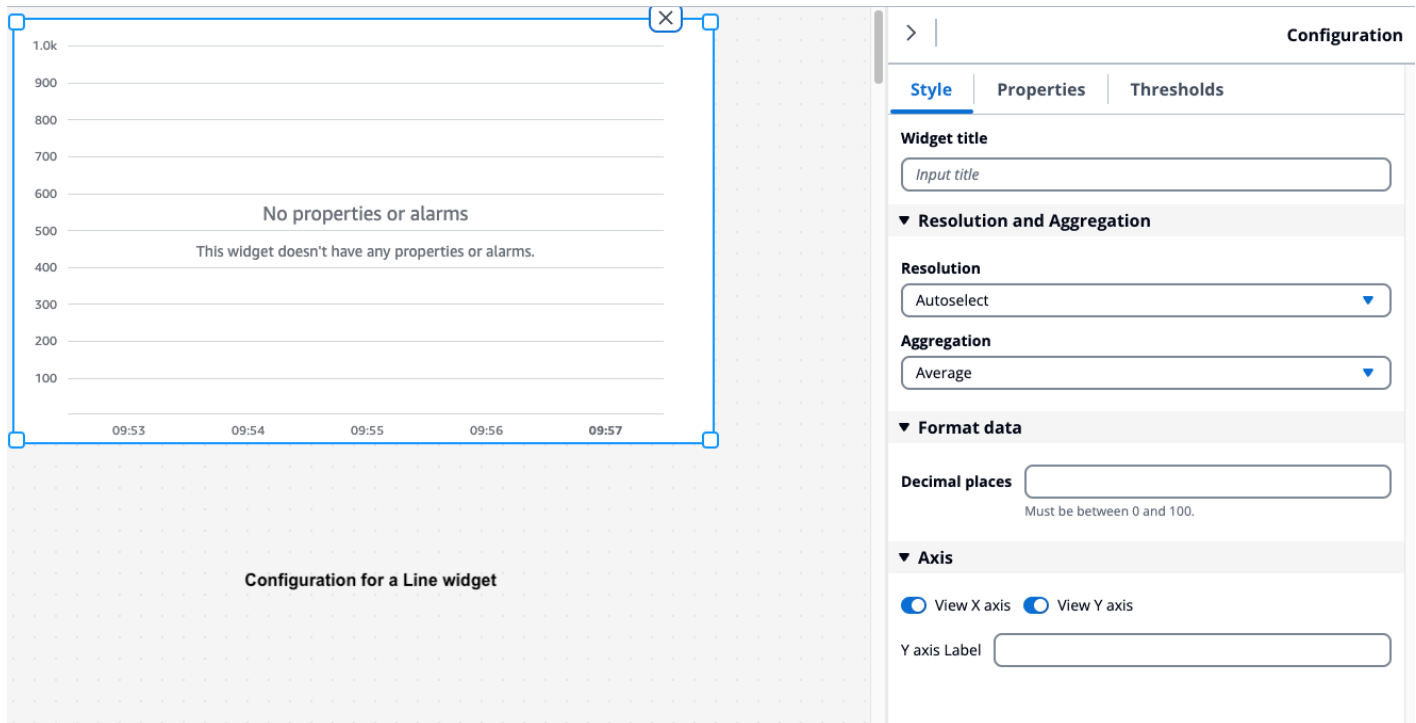
- 해상도 및 집계 - 여기에서 해상도 및 집계 값을 설정합니다.
- 데이터 형식 지정 - 10진수 위치를 표시할 10진수로 설정합니다.
- 표시 스타일 - 표시할 값을 선택합니다.
- 축 - 축을 표시하려면 선택합니다.

- 선 위젯:

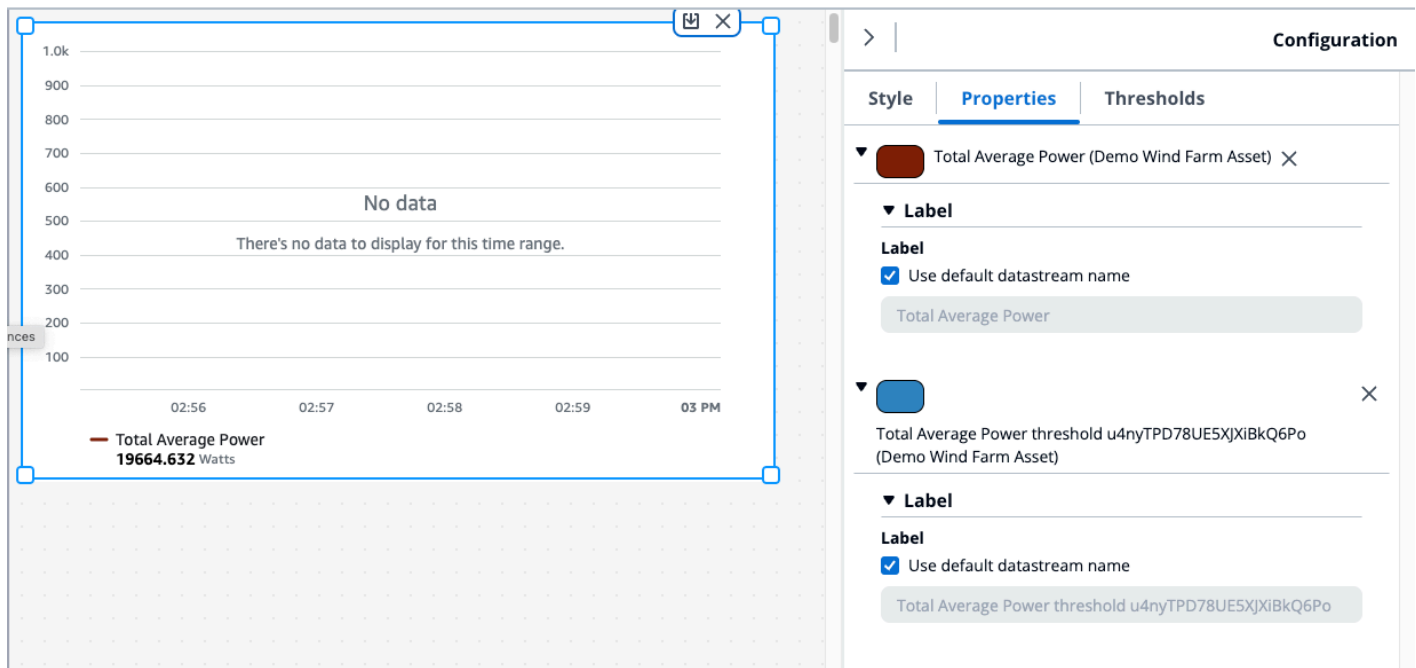
- 해상도 및 집계 - 여기에서 해상도 및 집계 값을 설정합니다.
- 데이터 형식 지정 - 소수점을 표시할 소수점 수로 설정합니다.
- Y축 - 레이블, 최소 및 최대 값을 추가합니다.
- 위젯 스타일 - 선 유형, 선 스타일, 선 두께 및 데이터 포인트 셰이프 값을 선택합니다.
- 범례 - 정렬 및 표시를 선택합니다.

- 게이지 위젯:

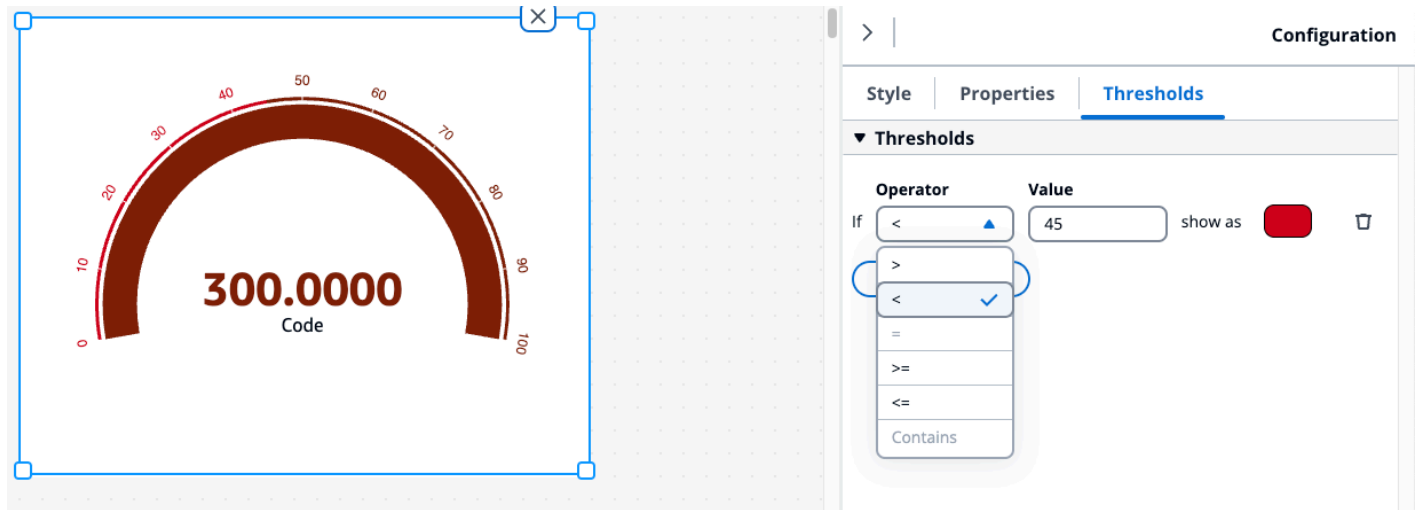
- 해상도 및 집계 - 여기에서 해상도 및 집계 값을 설정합니다.
- 데이터 형식 지정 - 소수점을 표시할 소수점 수로 설정합니다.
- 표시 스타일 - 표시할 값을 선택합니다.
- Y축 - 레이블, 최소 및 최대 값을 추가합니다.
- 글꼴 - 글꼴 크기, 단위 글꼴 크기 및 레이블 글꼴 크기 값을 선택합니다.



- 속성 - 위젯의 모든 속성이이 섹션에 나열됩니다. 위젯마다 속성이 다릅니다. 다음은 몇 가지 예입니다.
 - 선 위젯:
 - 레이블 - 기본 데이터스트림 이름을 사용하거나 새 이름을 지정하도록 선택합니다.
 - 스타일 - 선 유형, 선 스타일을 표시할 소수점 수로 설정합니다.
 - Y축 - 기본 스타일로 값을 선택하고, Y축 컨트롤을 표시하고, 최소 및 최대 값을 설정합니다.
 - 테이블 위젯:
 - 레이블 - 기본 데이터스트림 이름을 사용하거나 새 이름을 지정하도록 선택합니다.
 - 테이블 위젯:
 - 레이블 - 기본 데이터스트림 이름을 사용하거나 새 이름을 지정하도록 선택합니다.



- 임계값 - 위젯에 대한 임계값을 추가합니다. 위젯마다 구성이 다릅니다. 다음은 몇 가지 예입니다.
- 막대 차트 위젯:
 - 위젯에 추가할 임계값 추가를 선택합니다.
 - 연산자를 선택하고 임계값에 값을 지정합니다. 색상 팔레트의 색상으로 임계값을 사용자 지정합니다.
 - 모든 데이터에 임계값을 적용하도록 선택할 수 있습니다.
- 선 위젯:
 - 위젯에 추가할 임계값 추가를 선택합니다.
 - 연산자를 선택하고 임계값에 값을 지정합니다. 색상 팔레트의 색상으로 임계값을 사용자 지정합니다.
 - 드롭다운 메뉴에서 임계값을 표시하는 방법을 선택합니다.
- 게이지 위젯:
 - 위젯에 추가할 임계값 추가를 선택합니다.
 - 연산자를 선택하고 임계값에 값을 지정합니다. 색상 팔레트의 색상으로 임계값을 사용자 지정합니다.



위젯 사용

대시보드에서 위젯을 개별적으로 사용하거나 여러 개 선택하여 위젯을 사용할 수 있습니다.

대시보드에서 위젯 편집

단일 위젯을 선택하고 편집합니다. 대시보드에서 여러 위젯을 편집하려면 Shift + Left 키를 클릭하고 대시보드의 모든 위젯을 선택합니다. 선택하면 사용자는 스타일 구성 설정에서 새 데이터 스트림을 추가하고 위젯 제목을 수정할 수 있습니다. 대시보드의 모든 위젯에 대해 제목이 변경됩니다.

캔버스를 마우스 오른쪽 버튼으로 클릭하고 다음을 수행합니다.

- 복사 - 위젯의 사본을 캔버스에 추가합니다.
- 삭제 - 위젯을 삭제합니다.
- 앞으로 가져오기 - 선택한 위젯을 캔버스 앞으로 가져옵니다.
- 다시 전송 - 선택한 위젯을 캔버스의 후면으로 전송합니다.

위젯 크기 조정

대시보드에서 위젯을 여러 개 선택하여 위젯의 크기를 개별적으로 또는 그룹으로 조정합니다.

위젯의 크기를 변경하려면:

- 단일 위젯의 크기를 변경하려면 위젯을 선택하고 모서리로 드래그하여 크기를 변경합니다.
- 여러 위젯의 크기를 변경하려면 Shift + 마우스 왼쪽 버튼으로 여러 위젯을 선택하고 모서리로 끌어 크기를 변경합니다.

대시보드에서 위젯 삭제

대시보드에서 위젯을 여러 번 선택하여 위젯을 개별적으로 또는 그룹에서 삭제합니다.

위젯을 삭제하려면:

- 단일 위젯을 삭제하려면 위젯을 선택하고 마우스 오른쪽 버튼을 클릭하고 삭제를 선택합니다. 또한 오른쪽 상단 모서리에서 X를 선택하고 클릭하여 위젯을 삭제할 수 있습니다.
- 위젯을 여러 개 삭제하려면 Shift + 마우스 왼쪽 버튼으로 위젯을 여러 개 선택한 다음 마우스 오른쪽 버튼을 클릭하고 삭제를 선택합니다.

위젯의 경보

장비나 프로세스가 최적의 성능이 아닌 경우 경보를 통해 사용자와 팀원에게 알립니다. 특정 메트릭 값이 상한 범위와 하한 범위 내에 있어야 컴퓨터 또는 프로세스 성능이 최적화됩니다. 메트릭이 작동 범위를 벗어난 경우, 장비 운영자에게 경보를 보내 문제를 해결할 수 있도록 합니다. 경보를 통해 문제를 신속하게 식별하고 운영자에게 알려 장비 및 프로세스 성능을 극대화할 수 있습니다.

리소스 탐색기의 모델링된 탭에서 자산과 연결된 경보를 찾을 수 있습니다.

- 자산을 검색하고 선택합니다.
- 데이터 스트림 테이블을 지나 경보 데이터 스트림 섹션으로 스크롤하여 확장합니다.
- 경보 테이블에서 경보를 선택하고 추가를 선택합니다.

주제

- [다양한 위젯의 경보](#)

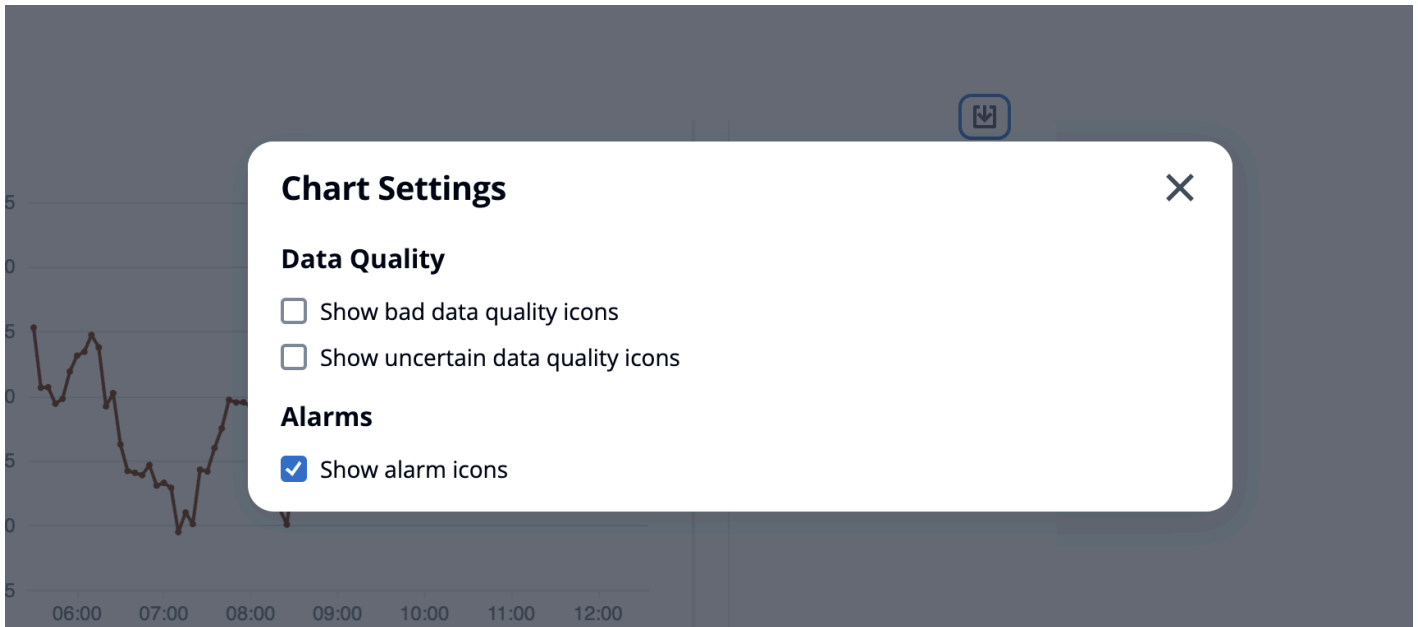
다양한 위젯의 경보

모든 위젯의 경우:

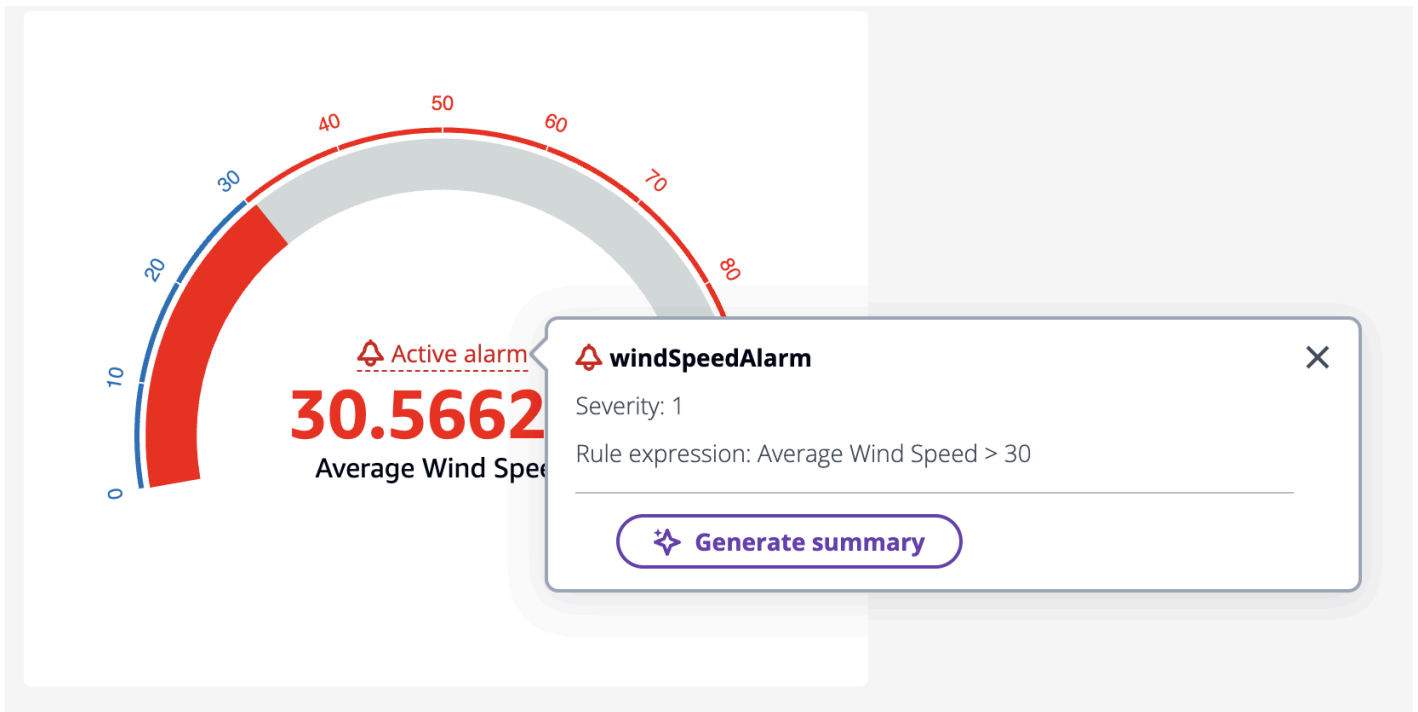
- 데이터 스트림 속성 설정은 위젯에 추가되는 속성 유형에 따라 달라집니다. 데이터 스트림 속성에는 전체 속성 설정 지원이 있지만 경보 속성은 현재 속성 설정 구성을 허용하지 않습니다.
- 경보 데이터 스트림을 추가하면 연결된 입력 속성 데이터 스트림도 차트에 추가됩니다. 경보 데이터 스트림을 제거하면 해당 입력 속성도 제거됩니다.
- 경보의 입력 속성 데이터 스트림을 개별적으로 제어하려면 둘 다 별도로 추가해야 합니다.

아래 예제에서는 일부 위젯이 경보를 사용하는 방법을 설명합니다.

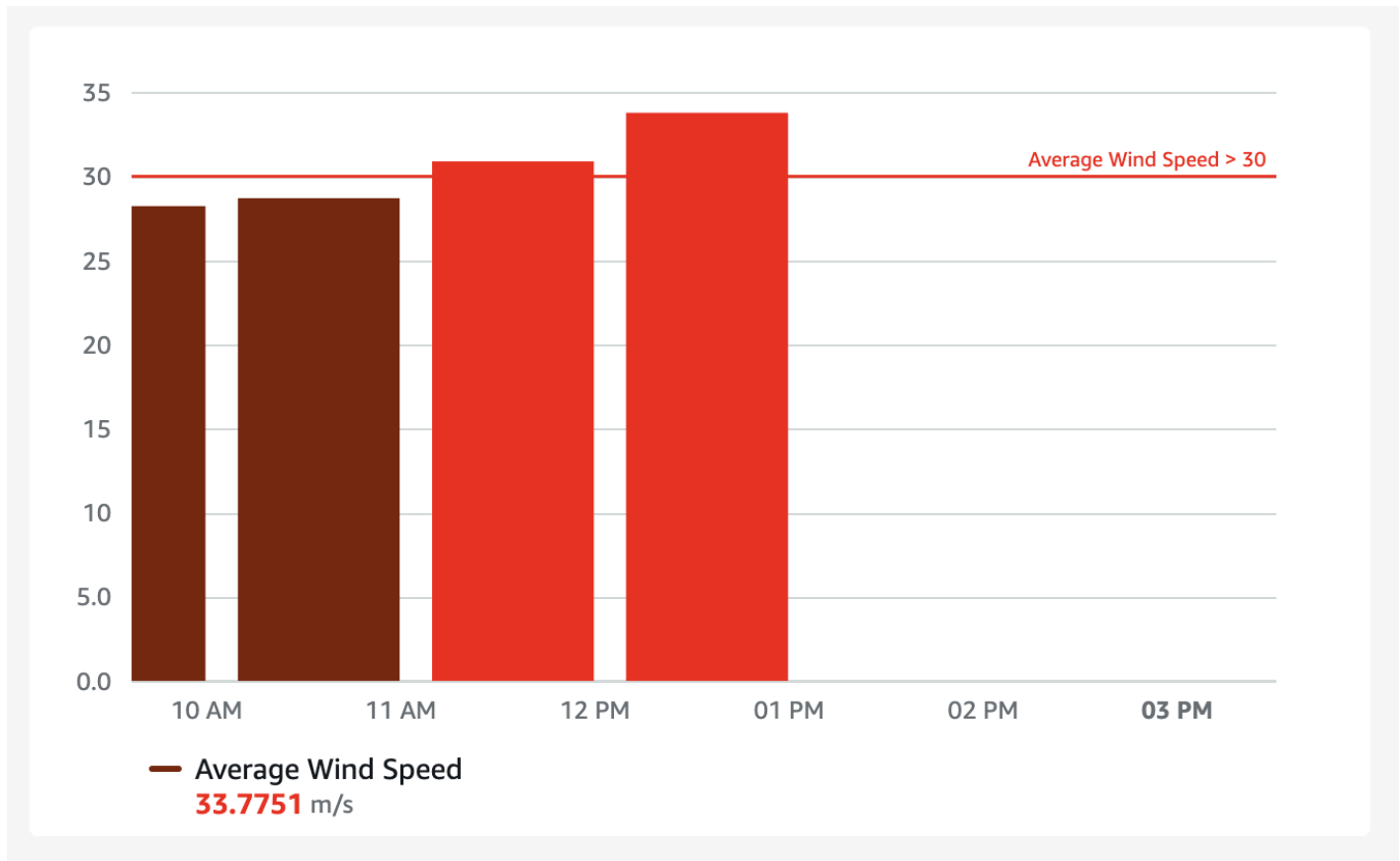
- 선 차트
 - 경보와 해당 입력 속성 데이터 스트림이 차트에 추가됩니다.
 - 차트 범례에서 경보 상태를 볼 수 있으며, 경보 상태가 변경될 때 데이터 스트림을 가리키는 아이콘으로 볼 수 있습니다.
 - 차트 설정에서 경보 아이콘을 끌 수 있습니다.



- KPI 및 게이지
 - 경보와 해당 입력 속성 데이터 스트림이 선택한 위젯에 추가됩니다.
 - 경보 임계값이 위젯에 추가되며, 위젯의 구성에 따라 색상이 변경됩니다.
 - 위젯에서 경보 상태를 선택하고 경보 세부 정보를 확인한 다음 요약 생성을 클릭하여 AWS IoT SiteWise 를 호출하여 경보 요약을 가져올 수 있습니다.



- 표
 - 경보와 해당 입력 속성이 테이블에 행으로 추가됩니다.
- 막대 차트
 - 경보는 차트에 임계값으로 추가되어 임계값을 위반하는 데이터 스트림의 색상을 변경합니다.
 - 연결된 데이터 스트림을 별도로 추가할 수 있습니다.
 - 위젯에서는 AWS IoT SiteWise 도우미와 상호 작용할 수 없습니다.

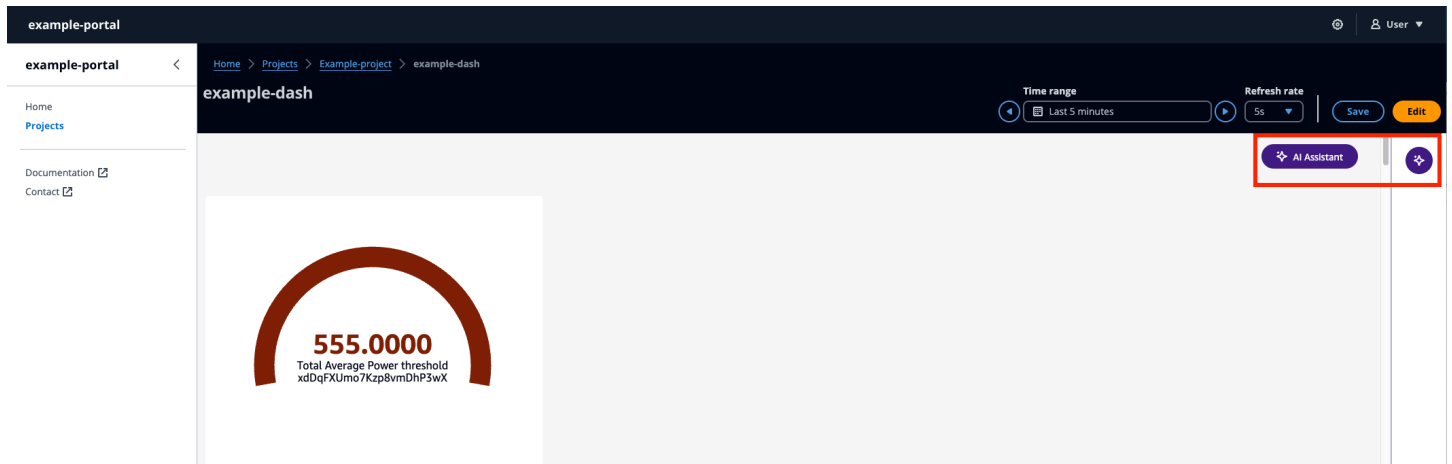


- 상태 타임라인
 - 경보는 타임라인에 임계값으로 추가됩니다.
 - 경보 상태 및 해당 입력 속성 데이터를 타임라인에 추가하는 작업이 진행 중입니다.
 - 위젯에서는 AWS IoT SiteWise 도우미와 상호 작용할 수 없습니다.

AWS IoT SiteWise 위젯에서 어시스턴트 사용

AWS IoT SiteWise 어시스턴트는 생성형 AI 기반 어시스턴트입니다. 이를 통해 공장 관리자, 품질 엔지니어, 유지 관리 기술자와 같은 사용자가 운영 및 엔터프라이즈 데이터에서 직접 인사이트를 얻고, 문제를 해결하고, 조치를 취할 수 있습니다. AWS IoT SiteWise 어시스턴트는 AWS IoT 데이터, 자산 모델, 설명서 및 설명서의 정보를 중요한 이벤트에 대한 이해하기 쉬운 요약으로 통합합니다. 또한 대화형 심층 질문 및 답변 세션을 활성화하여 쉽게 진단하고 근본 원인을 탐색하고 권장 사항을 안내할 수 있습니다.

AWS IoT SiteWise 어시스턴트 버튼은 대시보드의 오른쪽 상단에 있습니다. 이를 클릭하여 어시스턴트를 활성화합니다. 대시보드의 미리 보기 모드에서만 사용할 수 있습니다.



다음 시나리오에서 AWS IoT SiteWise 도우미를 사용합니다.

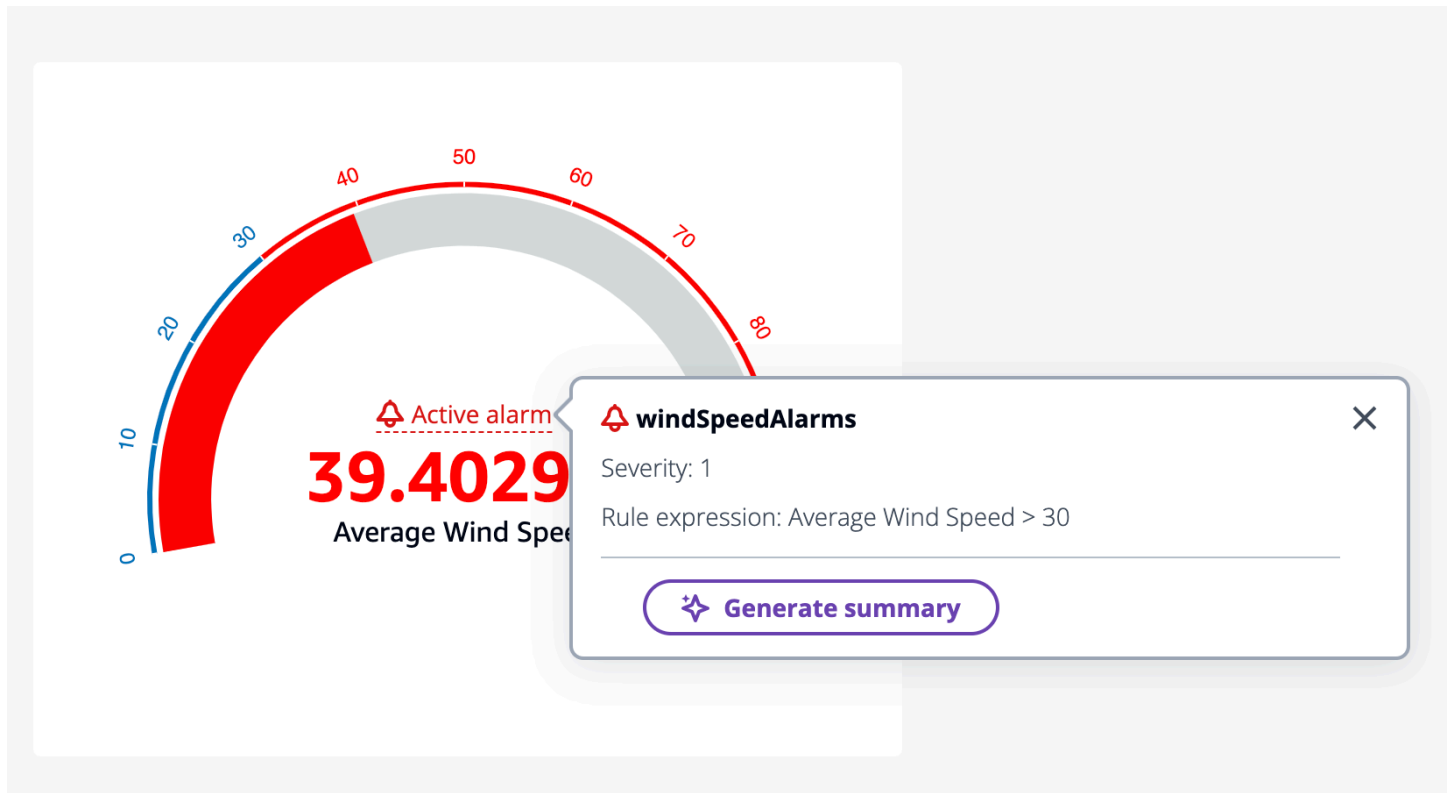
주제

- [사용 사례 - 경보 요약](#)
- [사용 사례 - 상황 요약](#)
- [사용 사례 - 심층 분석 요약](#)

사용 사례 - 경보 요약

대시보드에서 선택한 패널에 대한 현재 경보를 요약합니다. 경보는 라인, KPI, 게이지 및 테이블 위젯에서 지원됩니다. 경보가 있는 위젯을 선택하고 요약합니다.

- 위젯에서 활성 경보를 선택합니다.
- 경보에 대한 심각도 및 규칙 표현식이 표시됩니다.
- 요약 생성을 선택하여 요약을 생성합니다.



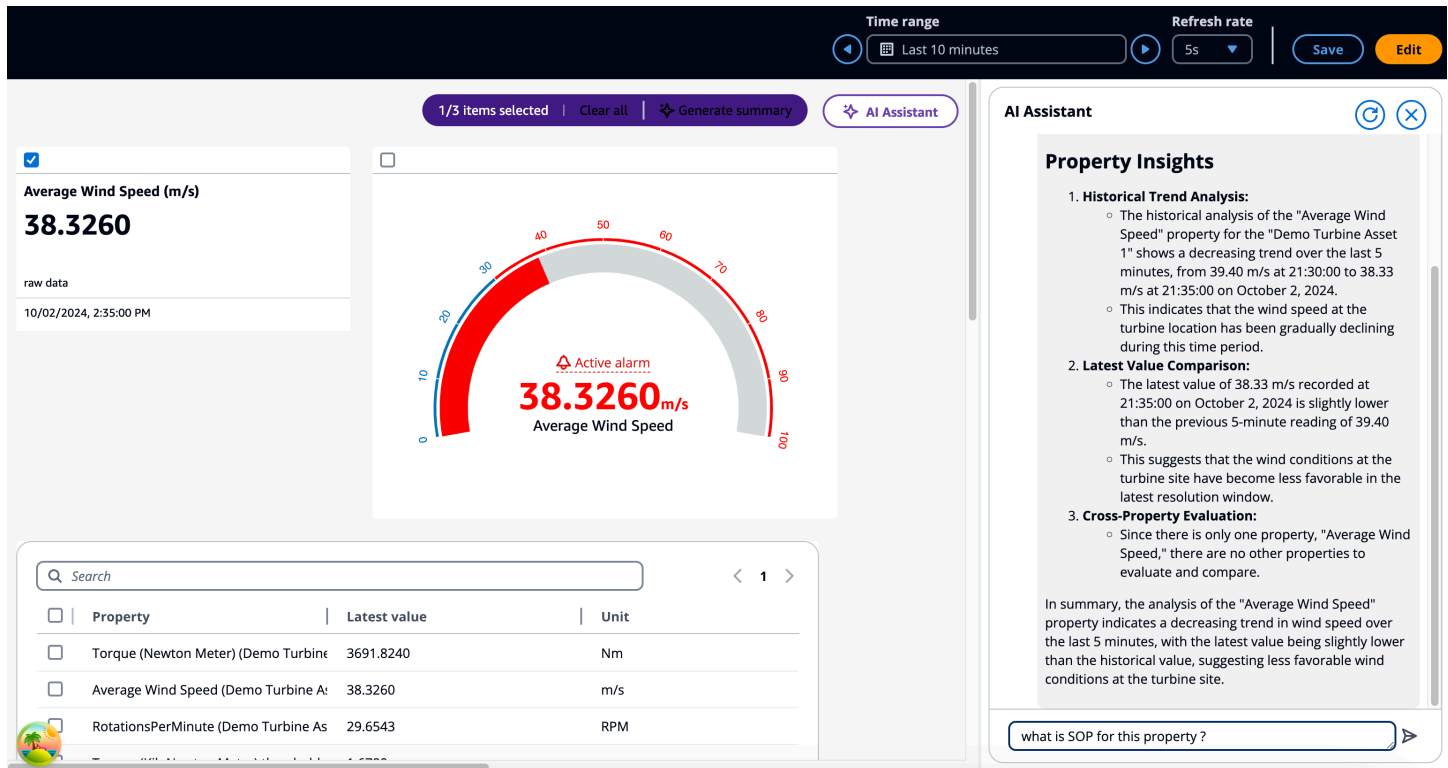
사용 사례 - 상황 요약

요약할 위젯을 최대 3개까지 선택합니다. 위젯과 속성의 조합일 수 있습니다. 3개 이상을 선택하면 도우미가 오류를 반환합니다.

AWS IoT SiteWise Assistant를 사용하여 상황 요약 생성

1. AI 도우미를 클릭합니다. 세 가지 옵션이 있는 메뉴가 표시됩니다.
 - a. 선택한 항목 - 3개만 선택합니다. 3개 이상을 선택할 수 없습니다.
 - b. 모두 지우기 - 선택을 지웁니다.
 - c. 요약 생성 - 선택한 항목에 대한 요약을 생성합니다.
2. 요약 생성을 선택하여 선택한 항목에 대한 요약을 생성합니다.

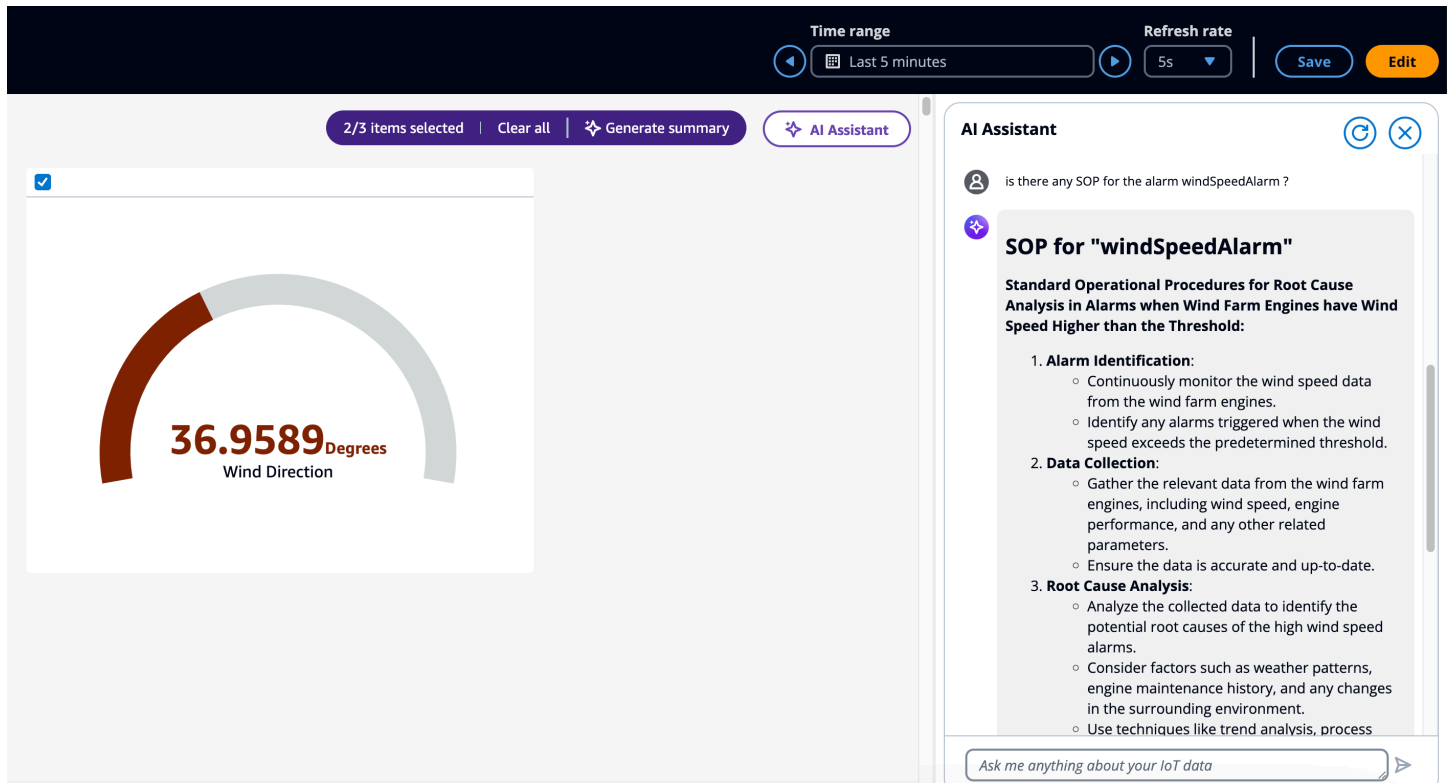
아래 이미지에는 위젯이 선택되어 있고 AWS IoT SiteWise 도우미의 요약이 있습니다.



사용 사례 - 심층 분석 요약

이는 사용자가 심층 분석을 수행하고 SOPs(표준 운영 절차), 설명서, 설명서에 액세스하고 다음 조치 단계를 고려할 수 있는 사용 사례입니다. 이전 섹션의 예제에서 사용자가 속성의 SOP에 대해 자세히 알아보려면 도우미에게 속성의 SOP에 대해 문의하세요. 그러면 SOP에 대한 심층 분석 정보가 사용자에게 표시됩니다.

아래 예제에서는 "WwindSpeedAlarm 경보에 대한 SOP가 있나요?"에 대한 답변을 표시합니다.



AWS IoT SiteWise 도우미에게 물어볼 샘플 질문

Note

- AWS IoT SiteWise 도우미는 엔터프라이즈급 지식 및 지침을 위해 [Amazon Kendra](#) 인덱스가 있는 데이터 세트를 사용해야 합니다. Amazon Kendra 인덱스가 없는 경우 [인덱스 생성을](#) 참조하여 인덱스를 생성합니다. [데이터 세트를](#) 추가하면 어시스턴트의 응답 품질이 향상됩니다. 자세한 내용은 [데이터 세트 생성을](#) 참조하세요.
- 일부 질문에는 AWS IoT TwinMaker 통합이 필요합니다. 자세한 내용은 [AWS IoT TwinMaker 및 통합 AWS IoT SiteWise](#)를 참조하세요.

대시보드에서 경보 요약을 가져온 후 동일한 대화의 일부로 도우미에게 물어볼 몇 가지 후속 질문입니다.

- 위 요약에서 자산의 세부 정보를 표시하시겠습니까?
- 루트에서 언급된 자산까지의 계층적 경로는 무엇입니까?
- 언급된 자산의 종속 하위 자산은 무엇입니까?
- 활성 경보가 있는 언급된 자산의 종속 자산은 무엇입니까?

- 활성 경보가 있는 모든 자산을 찾습니다.

대시보드에서 속성 요약을 가져온 후 동일한 대화의 일부로 도우미에게 물어볼 몇 가지 후속 질문입니다.

- 지난 24시간 동안 동일한 분석을 수행합니다.
- 위에서 언급한 속성과 관련된 설명서를 찾습니다.
- 자산 ID 1da67d28-14f8-4f71-a06a-386f0425a21d/asset 이름 데모 터빈 자산 1의 세부 정보를 제공합니다.

API에서 AWS IoT SiteWise 도우미를 호출합니다.

- 자산 ID에서 경보 이름 windSpeedAlarm에 대한 경보 요약을 생성합니다d591e153-e5cf-4206-96bb-ce3c119d9d2d.
- 자산 ID에서 경보 이름 windSpeedAlarm에 대한 지난 12시간/2일/1주 동안의 경보 요약을 생성합니다d591e153-e5cf-4206-96bb-ce3c119d9d2d.
- 자산 ID의 속성 IDab187fb7-d74b-44d9-bd9b-f2f19a9137cc에 대한 속성 요약 생성d591e153-e5cf-4206-96bb-ce3c119d9d2d
- 자산 ID ab187fb7-d74b-44d9-bd9b-f2f19a9137cc의 속성 ID에 대해 지난 12시간/2일/1주 동안의 속성 요약을 생성합니다d591e153-e5cf-4206-96bb-ce3c119d9d2d.
- 자산 이름 Turbine으로 자산을 찾습니다.
- 자산 ID d591e153-e5cf-4206-96bb-ce3c119d9d2d, 5356168c-3390-456f-802c-9f6e047810d4에서 속성 ID의 현재 속성 값을 지정합니다3cbb084e-1ded-4b08-9f21-1b47b2fb86fd.
- 자산 IDd591e153-e5cf-4206-96bb-ce3c119d9d2d와 자산 ID 간의 관계는 무엇입니까3cbb084e-1ded-4b08-9f21-1b47b2fb86fd?
- 풍력 터빈 낮은 RPM 문제를 해결하는 방법에 대한 설명서를 찾습니다.
- 속성 별칭 WindSpeed에 대한 속성 요약을 생성합니다.
- 지식 기반에 따른 작업 전 점검은 무엇입니까?

에서 데이터 쿼리 AWS IoT SiteWise

AWS IoT SiteWise API 작업을 사용하여 특정 시간 간격 동안 자산 속성의 현재 값, 기록 값 및 집계를 쿼리할 수 있습니다.

이러한 기능을 사용하여 데이터에 대한 인사이트를 얻으세요. 예를 들어 지정된 속성 값을 가진 모든 자산을 검색하거나 데이터에 대한 사용자 지정 표현을 구축합니다. API 작업을 사용하여 AWS IoT SiteWise 자산에 저장된 산업 데이터와 통합되는 소프트웨어 솔루션을 개발할 수도 있습니다. AWS IoT SiteWise Monitor에서 라이브 자산 데이터를 탐색할 수도 있습니다. SiteWise Monitor를 구성하는 방법에 대해 자세히 알아보려면 [를 사용하여 데이터 모니터링 AWS IoT SiteWise Monitor](#) 섹션을 참조하세요.

이 섹션에 설명된 작업은 타임스탬프, 품질, 값(TQV) 구조를 포함하는 속성 값 객체를 반환합니다.

- timestamp는 현재 Unix epoch 시간(초)과 오프셋(나노초)을 포함합니다.
- quality는 데이터 요소의 품질을 나타내는 다음 문자열 중 하나를 포함합니다.
 - GOOD – 데이터가 어떤 문제의 영향도 받지 않습니다.
 - BAD – 데이터가 센서 고장과 같은 문제의 영향을 받습니다.
 - UNCERTAIN – 데이터가 센서 부정확과 같은 문제의 영향을 받습니다.
- value는 속성 유형에 따라 다음 필드 중 하나를 포함합니다.
 - booleanValue
 - doubleValue
 - integerValue
 - stringValue
 - nullValue

주제

- [에서 현재 자산 속성 값 쿼리 AWS IoT SiteWise](#)
- [에서 과거 자산 속성 값 쿼리 AWS IoT SiteWise](#)
- [에서 자산 속성 집계 쿼리 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 쿼리 언어](#)

에서 현재 자산 속성 값 쿼리 AWS IoT SiteWise

이 자습서에서는 자산 속성의 현재 값을 가져오는 두 가지 방법을 보여줍니다. AWS IoT SiteWise 콘솔을 사용하거나 AWS Command Line Interface ()에서 API를 사용할 수 있습니다AWS CLI.

주제

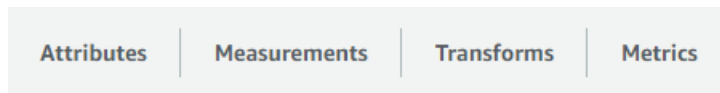
- [자산 속성의 현재 값 쿼리\(콘솔\)](#)
- [자산 속성의 현재 값 쿼리\(AWS CLI\)](#)

자산 속성의 현재 값 쿼리(콘솔)

AWS IoT SiteWise 콘솔을 사용하여 자산 속성의 현재 값을 볼 수 있습니다.

자산 속성의 현재 값을 가져오려면(콘솔)

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 쿼리할 속성이 있는 자산을 선택합니다.
4. 화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾습니다.
5. 속성 유형에 대한 탭을 선택합니다. 예를 들어 측정 속성의 현재 값을 보려면 측정을 선택합니다.



6. 보려는 속성을 찾습니다. 현재 값이 최신 값 옆에 나타납니다.

자산 속성의 현재 값 쿼리(AWS CLI)

AWS Command Line Interface (AWS CLI)를 사용하여 자산 속성의 현재 값을 쿼리할 수 있습니다.

[GetAssetPropertyValue](#) 작업을 사용하여 자산 속성의 현재 값을 쿼리합니다.

자산 속성을 식별하려면 다음 중 하나를 지정합니다.

- 데이터를 보낼 자산 속성의 assetId 및 propertyId
- 데이터 스트림 별칭(예: /company/windfarm/3/turbine/7/temperature)인 propertyAlias. 이 옵션을 사용하려면 먼저 자산 속성의 별칭을 설정해야 합니다. 속성 별칭을 설정하려면 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 섹션을 참조하세요.

자산 속성의 현재 값을 가져오려면(AWS CLI)

- 다음 명령을 실행하여 자산 속성의 현재 값을 가져옵니다. *asset-id*를 자산의 ID로 바꾸고 *property-id*를 속성의 ID로 바꿉니다.

```
aws iotsitewise get-asset-property-value \
  --asset-id asset-id \
  --property-id property-id
```

이 작업은 다음 형식으로 속성의 현재 TQV를 포함하는 응답을 반환합니다.

```
{
  "propertyValue": {
    "value": {
      "booleanValue": Boolean,
      "doubleValue": Number,
      "integerValue": Number,
      "stringValue": "String",
      "nullValue": {
        "valueType": "String"
      }
    }
  },
  "timestamp": {
    "timeInSeconds": Number,
    "offsetInNanos": Number
  },
  "quality": "String"
}
```

에서 과거 자산 속성 값 쿼리 AWS IoT SiteWise

AWS IoT SiteWise API [GetAssetPropertyValueHistory](#) 작업을 사용하여 자산 속성의 기록 값을 쿼리할 수 있습니다.

자산 속성을 식별하려면 다음 중 하나를 지정합니다.

- 데이터를 보낼 자산 속성의 *assetId* 및 *propertyId*

- 데이터 스트림 별칭(예: /company/windfarm/3/turbine/7/temperature)인 `propertyAlias`. 이 옵션을 사용하려면 먼저 자산 속성의 별칭을 설정해야 합니다. 속성 별칭을 설정하려면 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 섹션을 참조하세요.

다음 파라미터를 전달하여 결과를 구체화할 수도 있습니다.

- `startDate` – Unix Epoch 시간(초)으로 표시되며 과거 데이터를 쿼리하기 시작할 범위의 시작(제외)입니다.
- `endDate` – Unix Epoch 시간(초)으로 표시되며 과거 데이터를 쿼리하기 시작할 범위의 끝(포함)입니다.
- `maxResults` – 하나의 요청에서 반환할 최대 결과 수입니다. 기본값은 20개의 결과입니다.
- `nextToken` – 이 작업의 이전 호출에서 반환된 페이지 매김 토큰입니다.
- `timeOrdering` – 반환된 값에 적용할 정렬은 ASCENDING 또는 DESCENDING 입니다.
- `qualities` – 결과 필터링 기준이 되는 품질은 GOOD, BAD, 또는 UNCERTAIN입니다.

자산 속성에 대한 값 기록을 쿼리하려면(AWS CLI) 다음을 수행하세요.

1. 다음 명령을 실행하여 자산 속성에 대한 값 기록을 가져옵니다. 이 명령은 특정한 10분 간격 동안 속성의 기록을 쿼리합니다. *asset-id*를 자산의 ID로 바꾸고 *property-id*를 속성의 ID로 바꿉니다. 날짜 매개 변수를 쿼리할 간격으로 바꿉니다.

```
aws iotsitewise get-asset-property-value-history \
  --asset-id asset-id \
  --property-id property-id \
  --start-date 1575216000 \
  --end-date 1575216600
```

이 작업은 다음 형식으로 속성의 기록 TQV를 포함하는 응답을 반환합니다.

```
{
  "assetPropertyValueHistory": [
    {
      "value": {
        "booleanValue": Boolean,
        "doubleValue": Number,
        "integerValue": Number,
        "stringValue": "String",
        "nullValue": {
```

```

        "valueType": "String"
    },
    },
    "timestamp": {
        "timeInSeconds": Number,
        "offsetInNanos": Number
    },
    "quality": "String"
}
],
"nextToken": "String"
}

```

2. 값 항목이 더 있는 경우, nextToken 필드의 페이지 매김 토큰을 [GetAssetPropertyValueHistory](#)에 대한 후속 직접 호출에 전달할 수 있습니다.

에서 자산 속성 집계 쿼리 AWS IoT SiteWise

AWS IoT SiteWise 는 여러 시간 간격으로 계산된 기본 지표 집합인 집계된 자산 속성 값을 자동으로 계산합니다.는 자산 속성에 대해 매분, 시간 및 일마다 다음 집계를 AWS IoT SiteWise 계산합니다.

- average(평균) – 시간 간격 동안 속성 값의 평균입니다.
- count(개수) – 일정 기간에 걸친 속성의 데이터 포인트 수입니다.
- maximum(최대) – 시간 간격 동안 속성 값의 최대값입니다.
- minimum(최소) – 시간 간격 동안 속성 값의 최소값입니다.
- standard deviation(표준 편차) – 시간 간격 동안 속성 값의 표준 편차입니다.
- sum(합계) – 시간 간격 동안 속성 값의 합계입니다.

문자열 및 부울과 같은 숫자가 아닌 속성의 경우는 개수 집계만 AWS IoT SiteWise 계산합니다.

자산 데이터에 대한 사용자 지정 지표를 계산할 수도 있습니다. 지표 속성을 사용하여 그 작업에만 해당되는 집계를 정의할 수 있습니다. 지표 속성은 AWS IoT SiteWise API에 대해 사전 계산되지 않은 추가 집계 함수와 시간 간격을 제공합니다. 자세한 내용은 [속성과 기타 자산에서 데이터 집계\(지표\)](#) 단원을 참조하십시오.

주제

- [자산 속성에 대한 집계\(API\)](#)
- [자산 속성에 대한 집계를 쿼리하려면\(AWS CLI\) 다음을 수행하세요.](#)

자산 속성에 대한 집계(API)

AWS IoT SiteWise API를 사용하여 자산 속성에 대한 집계를 가져옵니다.

[GetAssetPropertyAggregates](#) 작업을 사용하여 자산 속성의 집계를 쿼리합니다.

자산 속성을 식별하려면 다음 중 하나를 지정합니다.

- 데이터를 보낼 자산 속성의 `assetId` 및 `propertyId`
- 데이터 스트림 별칭(예: `/company/windfarm/3/turbine/7/temperature`)인 `propertyAlias`. 이 옵션을 사용하려면 먼저 자산 속성의 별칭을 설정해야 합니다. 속성 별칭을 설정하려면 [예 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 섹션을 참조하세요.

다음 필수 파라미터를 전달해야 합니다.

- `aggregateTypes` – 검색할 집계 목록입니다. AVERAGE, COUNT, MAXIMUM, MINIMUM, STANDARD_DEVIATION, SUM 중 하나를 지정할 수 있습니다.
- `resolution` – 지표를 검색할 시간 간격: 1m (1분), 15m (15분), 1h (1시간) 또는 1d (1일).
- `startDate` – Unix Epoch 시간(초)으로 표시되며 과거 데이터를 쿼리하기 시작할 범위의 시작(제외)입니다.
- `endDate` – Unix Epoch 시간(초)으로 표시되며 과거 데이터를 쿼리하기 시작할 범위의 끝(포함)입니다.

다음 파라미터 중 하나를 전달하여 결과를 구체화할 수도 있습니다.

- `maxResults` – 하나의 요청에서 반환할 최대 결과 수입니다. 기본값은 20개의 결과입니다.
- `nextToken` – 이 작업의 이전 호출에서 반환된 페이지 매김 토큰입니다.
- `timeOrdering` – 반환된 값에 적용할 정렬은 ASCENDING 또는 DESCENDING 입니다.
- `qualities` – 결과 필터링 기준이 되는 품질은 GOOD, BAD, 또는 UNCERTAIN입니다.

Note

[GetAssetPropertyAggregates](#) 작업은 이 단원에서 설명하는 다른 작업과 다른 형식의 TQV를 반환합니다. `value` 구조에는 요청의 각 `aggregateTypes`에 대한 필드가 포함되어 있습니다. `timestamp`에는 집계 발생 시간(초)이 포함됩니다.

자산 속성에 대한 집계를 쿼리하려면(AWS CLI) 다음을 수행하세요.

자산 속성에 대한 집계를 쿼리하려면(AWS CLI) 다음을 수행하세요.

1. 다음 명령을 실행하여 자산 속성에 대한 집계를 가져옵니다. 이 명령은 특정한 1시간 간격에 대해 1시간 해상도로 평균과 합계를 쿼리합니다. *asset-id*를 자산의 ID로 바꾸고 *property-id*를 속성의 ID로 바꿉니다. 파라미터를 쿼리할 집계 및 간격으로 바꿉니다.

```
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
```

이 작업은 다음 형식으로 속성의 과거 TQV를 포함하는 응답을 반환합니다. 응답에는 요청된 집계만 포함됩니다.

```
{
  "aggregatedValues": [
    {
      "timestamp": Number,
      "quality": "String",
      "value": {
        "average": Number,
        "count": Number,
        "maximum": Number,
        "minimum": Number,
        "standardDeviation": Number,
        "sum": Number
      }
    }
  ],
  "nextToken": "String"
}
```

2. 값 항목이 더 있는 경우 nextToken 필드의 페이지 매김 토큰을 [GetAssetPropertyAggregates](#)에 대한 후속 직접 호출에 전달할 수 있습니다.

Note

쿼리 범위에 TQV null 값이 포함된 경우 [AssetPropertyValue](#) API를 참조하세요. TQVs 개수를 제외한 모든 통계는 문자열 TQVs 대한 통계와 마찬가지로 null 응답을 생성합니다. 쿼리 범위에 이중 유형 TQVs에 Double.NaN 대항가 포함된 경우 개수를 제외한 모든 계산에서가 생성됩니다Double.NaN.

AWS IoT SiteWise 쿼리 언어

AWS IoT SiteWise 데이터 검색 [ExecuteQuery](#) API 작업을 사용하면 다음에서 선언적 구조 정의 및 이와 연결된 시간 데이터에 대한 정보를 검색할 수 있습니다.

- 모델
- 자산
- 측정값
- 지표
- 변환
- 집계

이는 쿼리 문과 같은 SQL을 사용하여 단일 API 요청으로 수행할 수 있습니다.

Note

이 기능은 AWS GovCloud(미국 서부)를 제외하고 AWS IoT SiteWise 및 AWS IoT TwinMaker를 모두 사용할 수 있는 모든 리전에서 사용할 수 있습니다.

주제

- [사전 조건](#)
- [에 대한 쿼리 언어 참조 AWS IoT SiteWise](#)

사전 조건

AWS IoT SiteWise 는 산업 데이터를 구성하고 모델링할 수 AWS IoT TwinMaker 있도록와 통합할 수 있는 권한이 필요합니다.

모델, 자산, 측정값, 지표, 변환 및 집계에 대한 정보를 검색하기 전에 다음 사전 조건을 충족하는지 확인하세요.

- AWS 계정의 AWS IoT SiteWise 및 AWS IoT TwinMaker 설정에 대한 서비스 연결 역할입니다. 서비스 연결 역할에 대한 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 생성](#)을 참조하세요
- IAM 역할에 대해 활성화된 AWS IoT SiteWise 통합입니다. 자세한 내용은 [AWS IoT SiteWise 및 통합 AWS IoT TwinMaker](#) 단원을 참조하십시오.
- 리전의 계정IoTSiteWiseDefaultWorkspace에서 ID가 인 AWS IoT TwinMaker 워크스페이스입니다. 자세한 내용은 AWS IoT TwinMaker 사용 설명서의 [IoTSiteWiseDefaultWorkspace 사용](#)을 참조하세요.
- 에 대한 표준 또는 계층형 번들 요금 모드가 AWS IoT TwinMaker 활성화되어 있습니다. 자세한 내용은 AWS IoT TwinMaker 사용 설명서의 [AWS IoT TwinMaker 요금 모드 전환](#)을 참조하세요.

에 대한 쿼리 언어 참조 AWS IoT SiteWise

AWS IoT SiteWise 는 데이터 작업을 위한 풍부한 쿼리 언어를 지원합니다. 사용 가능한 데이터 유형, 연산자, 함수 및 구문은 다음 주제에 설명되어 있습니다.

쿼리 언어로 AWS IoT SiteWise 쿼리를 작성 [쿼리 예제](#) 하려면 섹션을 참조하세요.

주제

- [쿼리 참조 보기 이해](#)
- [지원되는 데이터 유형](#)
- [SELECT 문을 사용하여 데이터 검색](#)
- [논리 연산자](#)
- [비교 연산자](#)
- [쿼리 예제](#)

쿼리 참조 보기 이해

이 섹션에서는 프로세스 메타데이터 및 원격 측정 데이터와 AWS IoT SiteWise같은의 뷰를 이해하는데 도움이 되는 정보를 제공합니다.

다음 표에는 뷰의 이름과 설명이 나와 있습니다.

데이터 모델

뷰 이름	보기 설명
asset	자산 및 모델 파생에 대한 정보를 포함합니다.
asset_property	자산 속성의 구조에 대한 정보를 포함합니다.
raw_time_series	시계열의 기록 데이터를 포함합니다.
latest_value_time_series	시계열의 최신 값을 포함합니다.
precomputed_aggregates	자동으로 계산된 집계된 자산 속성 값을 포함합니다. 여러 시간 간격으로 계산된 기본 지표 세트입니다.

다음 뷰에는 샘플 데이터와 함께 쿼리의 열 이름이 나열됩니다.

View:asset

asset_id	asset_name	asset_description	asset_model_id
88898498-0b8b-42b5-bf57-16180bc3d3a0	WindTurbine A	WindTurbine Asset A	17847250-5bf0-4f74-b775-cc03f05e7cb8
17847250-5bf0-4f74-b775-cc03f05e7cb8	풍력 터빈 자산 모델	풍력 발전소의 터빈을 나타냅니다.	

View:asset_property

property_id	asset_id	property_name	property_alias	asset_composite_model_id	
b29be434-b000-4d74-b809-75287d83bcd6	88898498-0b8b-42b5-bf57-16180bc3d3a0	모터 온도	Rochester 2/44///Line-5/Bus-2/Machine		

property_id	asset_id	property_name	property_alias	asset_composite_model_id	
			-5/Temperature		
3b458f00-24e7-458a-b4e8-c6026eff654a	88898498-0b8b-42b5-bf57-16180bc3d3a0	바람 방향	/company/windfarm/3/turbine/7/winddirection	2f458n00-56e7-458h-b4e8-c6026eff985g	

View:raw_time_series

asset_id	property_id	property_alias	event_timestamp	quality	boolean_value	int_value	double_value	string_value
88898498-0b8b-42b5-bf57-16180bc3d3a0	b29be434-b000-4d72-2/44///-Li-b809-75287d83bcd	Rochester-5/Bus-2/Machine-5/Temperature	157521960	좋음			115.0	
88898498-0b8b-42b5-bf57-16180bc3d3a0	3b458f00-24e7-458a-b4e8-c6026eff654a	/company/windfarm/3/turbine/7/winddirection	157521967	좋음			348.75	

Note

raw_time_series 뷰를 쿼리하려면 event_timestamp 열에 필터 절을 포함해야 합니다. 이는 필수 필터이며 쿼리가 없으면 실패합니다.

Example 쿼리

```
SELECT event_timestamp, double_value FROM raw_time_series WHERE event_timestamp > 1234567890
```

View:latest_value_time_series

asset_id	property_id	property_alias	event_timestamp	quality	boolean_value	int_value	double_value	string_value
888984980b8b-42b1-b4e8-bf57-16180bc3d3a	3b458f00-24e7-458c-b4e8-c6026eff654a	/ company, windfarm 3/ turbine /7/ winddirection	15752196	좋은			355.39	

View:precomputed_aggregates

asset_id	property_id	property_alias	event_timestamp	quality	resolution	sum_value	count_value	average_value	maximum_value	minimum_value	stdev_value
888984980b8b-42b1-b4e8-bf57-16180bc3d3a	b29be4b000-42b1-b4e8-bf57-16180bc3d3a	Roche 2/44/, Li ne-5/ Bus-2/ Machi	15752196	좋은	15m	1105.4	15	73.4	80.6	68	3.64

asset_id	property_id	property_alias	event_timestamp	quality	resolution	sum_value	count_value	average_value	maximum_value	minimum_value	stdev_value
		-5/ Temperature									

지원되는 데이터 유형

AWS IoT SiteWise 쿼리 언어는 다음 데이터 형식을 지원합니다.

스칼라 값

데이터 유형	설명
STRING	최대 길이가 1,024바이트인 문자열입니다.
INTEGER	범위가 -2,147,483,648 to 2,147,483,647 인 서명된 32비트 정수입니다.
DOUBLE	범위가 -10^{100} to 10^{100} 이거나 IEEE 754 배정밀 Nan도인 부동 소수점 숫자입니다.
BOOLEAN	true 또는 false

Null 값: 정의된 데이터가 없음을 true 나타내는 부울입니다.

Note

배정밀도 데이터는 정확하지 않습니다. 일부 값은 정확하게 변환되지 않으며 제한된 정밀도로 인해 모든 실수를 나타내지 않습니다. 쿼리의 부동 소수점 데이터는 내부적으로 표시되는 값과 같지 않을 수 있습니다. 입력 숫자의 정밀도가 너무 높으면 값이 반올림됩니다.

SELECT 문을 사용하여 데이터 검색

SELECT 문은 하나 이상의 뷰에서 데이터를 검색하는 데 사용됩니다. AWS IoT SiteWise 는 뷰의 암시적 JOIN을 지원합니다. 쉼표로 구분하여 조인할 뷰를 나열할 수 있습니다(SELECT 문의 FROM 절에서).

Example

다음 SELECT 문을 사용합니다.

```
SELECT select_expr [, ...]
[ FROM from_item [, ...] ]
[ WHERE [LIKE condition ESCAPE condition] ]
```

이전 예제에서 LIKE 절은 wild card. AWS IoT SiteWise supports를 와일드카드 문자percentage (%)로 사용하여 검색 및 필터링 조건을 지정합니다.

Example 조건에서 %를 사용하려면:

```
Prefix search: String%
Infix search: %String%
Suffix search: %String
```

Example 자산을 검색하려면:

```
SELECT asset_name, asset_description FROM asset WHERE asset_name LIKE 'Wind%'
```

Example ESCAPE 조건을 사용하여 자산을 검색하려면:

```
SELECT asset_name, asset_description FROM asset WHERE asset_name LIKE 'room\%' ESCAPE
'\'
```

논리 연산자

AWS IoT SiteWise 는 다음과 같은 논리 연산자를 지원합니다.

논리 연산자

연산자	설명	예제
AND	TRUE 두 값이 모두 true인 경우	a AND b

a 또는 b가 FALSE이면 이전 표현식은 false로 평가됩니다. AND 연산자가 true로 평가되려면 a와 b가 모두 true여야 합니다.

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
```

비교 연산자

AWS IoT SiteWise 는 다음 비교 연산자를 지원합니다. 모든 비교 작업은 기본 제공 데이터 유형에 사용할 수 있으며 부울로 평가됩니다.

논리 연산자

연산자	설명
<	보다 작음
>	보다 큼
<=	작거나 같음
>=	크거나 같음
=	같음
!=	같지 않음

숫자가 아닌 값에 대한 비교 작업 실제 테이블

유형	유형 >= x	유형 <= x	유형 > x	유형 < x	유형 = x	유형 != x
NaN	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

연산자처럼 동작하지만 특별한 구문이 있는 일부 조건자가 있습니다. 예제는 아래를 참조하세요.

비교 조건자

연산자	설명
IS NULL	값이 인지 테스트합니다NULL.
IS NOT NULL	값이가 아닌지 테스트합니다NULL.
IS NaN	값이 인지 테스트합니다NaN.
IS NOT NaN	값이가 아닌지 테스트합니다NaN.

쿼리 예제

메타데이터 필터링

다음 예제는 AWS IoT SiteWise 쿼리 언어가 있는 SELECT 문을 사용한 메타데이터 필터링을 위한 것입니다.

```
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%'
```

값 필터링

다음은 AWS IoT SiteWise 쿼리 언어가 있는 SELECT 문을 사용한 값 필터링의 예입니다.

```
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
```

다른 AWS 서비스와 상호 작용

AWS IoT SiteWise 는 자산 데이터를 AWS IoT MQTT 게시-구독 메시지 브로커에 게시할 수 있으므로 다른 AWS 서비스의 자산 데이터와 상호 작용할 수 있습니다.는 AWS IoT 코어 규칙을 사용하여 자산 데이터를 다른 AWS 서비스로 라우팅하는 데 사용할 수 있는 고유한 MQTT 주제를 각 자산 속성에 AWS IoT SiteWise 할당합니다. 예를 들어 다음 작업을 수행하도록 AWS IoT 코어 규칙을 구성할 수 있습니다.

- 장비 결함을 식별하고 [AWS IoT Events](#)에 데이터를 전송하여 담당자에게 알립니다.
- 데이터를 [Amazon DynamoDB](#)로 전송하여 외부 소프트웨어 솔루션에서 사용할 특정 자산 데이터를 기록합니다.
- [AWS Lambda](#) 함수를 트리거하여 주간 보고서를 생성합니다.

DynamoDB에 속성 값을 저장하는 규칙의 설정 단계를 안내하는 자습서를 따를 수 있습니다. 자세한 내용은 [Amazon DynamoDB에 속성 값 업데이트 게시](#) 섹션을 참조하세요.

규칙을 구성하는 방법에 대한 자세한 내용은 AWS IoT 개발자 가이드의 [규칙](#)을 참조하세요.

다른 AWS 서비스의 데이터를 다시 사용할 수도 있습니다 AWS IoT SiteWise. AWS IoT SiteWise 규칙 작업을 통해 데이터를 수집하려면 섹션을 참조하세요 [AWS IoT Core 규칙을 AWS IoT SiteWise 사용하여 데이터 수집](#).

주제

- [MQTT 주제의 자산 속성 이해](#)
- [에서 자산 속성 알림 켜기 AWS IoT SiteWise](#)
- [에서 자산 속성 알림 쿼리 AWS IoT SiteWise](#)
- [자산 속성 알림을 사용하여 Amazon S3로 데이터 내보내기](#)
- [Grafana AWS IoT SiteWise 와 통합](#)
- [AWS IoT SiteWise 및 통합 AWS IoT TwinMaker](#)
- [Lookout for Equipment로 이상 감지](#)

MQTT 주제의 자산 속성 이해

모든 자산 속성에는 다음과 같은 형식의 고유한 MQTT 주제 경로가 있습니다.


```
$aws/sitewise/asset-models/assetModelId/assets/assetId/properties/propertyId
```

Note

AWS IoT SiteWise 는 AWS IoT 코어 규칙 엔진에서 # (다중 수준) 주제 필터 와일드카드를 지원하지 않습니다. +(단일 레벨) 와일드카드를 사용할 수 있습니다. 예를 들어 다음 주제 필터를 사용하여 특정 자산 모델의 모든 업데이트를 일치시킬 수 있습니다.

```
$aws/sitewise/asset-models/assetModelId/assets/+/properties/+
```

주제 필터 와일드카드에 대한 자세한 내용은 AWS IoT Core 개발자 가이드의 [주제](#)를 참조하세요.

에서 자산 속성 알림 켜기 AWS IoT SiteWise

속성 알림을 활성화하여 자산 데이터 업데이트를 게시 AWS IoT Core한 다음 데이터에 대한 쿼리를 실행할 수 있습니다. 자산 속성 알림을 통해서는 Amazon S3로 AWS IoT SiteWise 데이터를 내보내는 데 사용할 수 있는 AWS CloudFormation 템플릿을 AWS IoT SiteWise 제공합니다.

Note

자산 데이터는 값이 변경되었는지 여부에 AWS IoT SiteWise관계없이에서 수신할 AWS IoT Core 때마다 로 전송됩니다.

주제

- [자산 속성 알림 켜기\(콘솔\)](#)
- [자산 속성 알림 켜기\(AWS CLI\)](#)

자산 속성 알림 켜기(콘솔)

기본적으로 AWS IoT SiteWise 는 속성 값 업데이트를 게시하지 않습니다. AWS IoT SiteWise 콘솔을 사용하여 자산 속성에 대한 알림을 활성화할 수 있습니다.

자산 속성에 대한 알림을 활성화하거나 비활성화하려면(콘솔) 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택합니다.
3. 속성의 알림을 활성화할 자산을 선택합니다.

Tip

화살표 아이콘을 선택하여 자산 계층 구조를 확장하고 자산을 찾을 수 있습니다.

4. 편집을 선택합니다.
5. 자산 속성의 알림 상태에서 ENABLED(활성)를 선택합니다.

The screenshot shows the AWS IoT SiteWise console interface. On the left, there is a text input field for a property alias, labeled '"Wind Speed"' and containing the placeholder text 'Enter a property alias'. Below the input field, it says 'Must be less than 2048 characters.' On the right, there is a 'Notification status' dropdown menu. The dropdown is currently set to 'ENABLED', which is circled in red. Below the dropdown, there is a text block stating: 'Notification will be published to topic \$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-cdef-33333EXAMPLE'.

DISABLED(비활성)를 선택하여 자산 속성에 대한 알림을 비활성화할 수도 있습니다.

6. 저장(Save)을 선택합니다.

자산 속성 알림 켜기(AWS CLI)

기본적으로 AWS IoT SiteWise 는 속성 값 업데이트를 게시하지 않습니다. AWS Command Line Interface (AWS CLI)를 사용하여 자산 속성에 대한 알림을 활성화하거나 비활성화할 수 있습니다.

이 절차를 완료하려면 자산의 `assetId` 및 속성의 `propertyId`를 알고 있어야 합니다. 외부 ID를 사용할 수도 있습니다. 자산을 생성했지만 해당 `assetId`를 모르고 있는 경우에는 [ListAssets](#) API를 사용하여 특정 모델에 대한 모든 자산을 표시합니다. [DescribeAsset](#) 작업을 사용하여 속성 ID를 포함한 해당 자산의 속성을 봅니다.

[UpdateAssetProperty](#) 작업을 사용하여 자산 속성에 대한 알림을 활성화하거나 비활성화합니다. 다음 파라미터를 지정합니다.

- `assetId` - 자산의 ID입니다.
- `propertyId` - 자산 속성의 ID입니다.
- `propertyNotificationState` - 속성 값 알림 상태는 ENABLED 또는 DISABLED입니다.

- `propertyAlias` – 속성의 별칭입니다. 알림 상태를 업데이트할 때 속성의 기존 별칭을 지정합니다. 이 파라미터를 생략하면 속성의 기존 별칭이 제거됩니다.

자산 속성에 대한 알림을 활성화하거나 비활성화하려면(CLI) 다음을 수행하세요.

1. 다음 명령을 실행하여 자산 속성의 별칭을 검색합니다. `asset-id`를 자산의 ID로 바꾸고 `property-id`를 속성의 ID로 바꿉니다.

```
aws iotsitewise describe-asset-property \
  --asset-id asset-id \
  --property-id property-id
```

이 작업은 다음과 같은 형식으로 자산 속성의 세부 정보가 포함된 응답을 반환합니다. 속성 별칭은 JSON 객체에 있는 `assetProperty.alias`입니다.

```
{
  "assetId": "a1b2c3d4-5678-90ab-cdef-2222EXAMPLE",
  "assetName": "Wind Turbine 7",
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-1111EXAMPLE",
  "assetProperty": {
    "id": "a1b2c3d4-5678-90ab-cdef-3333EXAMPLE",
    "name": "Wind Speed",
    "alias": "/company/windfarm/3/turbine/7/windspeed",
    "notification": {
      "topic": "$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-1111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-2222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-3333EXAMPLE",
      "state": "DISABLED"
    },
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
      "measurement": {}
    }
  }
}
```

2. 다음 명령을 실행하여 자산 속성에 대한 알림을 활성화합니다. `property-alias`를 이전 명령의 응답에서 얻은 속성 별칭으로 바꾸거나, `--property-alias`를 생략하여 별칭 없이 속성을 업데이트합니다.

```
aws iotsitewise update-asset-property \
  --asset-id asset-id \
  --property-id property-id \
  --property-notification-state ENABLED \
  --property-alias property-alias
```

--property-notification-state DISABLED를 전달하여 자산 속성에 대한 알림을 비활성화할 수도 있습니다.

에서 자산 속성 알림 쿼리 AWS IoT SiteWise

자산 속성 알림을 쿼리하려면 SQL 문으로 구성된 AWS IoT Core 규칙을 생성합니다.

AWS IoT SiteWise 는 자산 속성 데이터 업데이트를 다음 형식으로 AWS IoT 코어에 게시합니다.

```
{
  "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 목록의 각 구조는 타임스탬프-품질-값(TQV) 구조입니다.

- timestamp는 현재 Unix epoch 시간(초)과 오프셋(나노초)을 포함합니다.
- quality는 데이터 요소의 품질을 나타내는 다음 문자열 중 하나를 포함합니다.
 - GOOD – 데이터가 어떤 문제의 영향도 받지 않습니다.
 - BAD – 데이터가 센서 고장과 같은 문제의 영향을 받습니다.
 - UNCERTAIN – 데이터가 센서 부정확과 같은 문제의 영향을 받습니다.
- value는 속성 유형에 따라 다음 필드 중 하나를 포함합니다.
 - booleanValue
 - doubleValue
 - integerValue
 - stringValue
 - nullValue

nullValue - 값이 Null이고 품질이 BAD 또는 인 속성 값의 유형을 나타내는 다음 필드가 있는 구조입니다UNCERTAIN.

- valueType - {"B", "D", "S", "I"}의 열거형

values 배열의 값을 구문 분석하려면 규칙의 SQL 문에서 복잡한 중첩 객체 쿼리를 사용해야 합니다. 자세한 내용은 AWS IoT 개발자 가이드의 [중첩 객체 쿼리](#)를 참조하거나 [Amazon DynamoDB에 속성 값 업데이트 게시](#) 자습서에서 자산 속성 알림 메시지 구문 분석에 대한 구체적인 예를 참조하세요.

Example 값 배열을 추출하는 쿼리 예제

다음 명령문은 해당 속성을 가진 모든 자산의 특정 이중 유형 속성을 확인하기 위해 업데이트된 속성 값 배열을 쿼리하는 방법을 보여 줍니다.

```
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'
```

이전 규칙 쿼리 문은 다음과 같은 형식으로 데이터를 출력합니다.

```
{
  "windspeed": [
    26.32020195042838,
    26.282584572975477,
    26.352566977372508,
    26.283084346171442,
    26.571883739599322,
    26.60684140743005,
    26.628738636715045,
    26.273486932802125,
    26.436379105473964,
    26.600590095377303
  ]
}
```

Example 단일 값을 추출하는 쿼리 예제

다음 명령문은 해당 속성을 가진 모든 자산의 특정 이중 유형 속성을 확인하기 위해 속성 값 배열의 첫 번째 값을 쿼리하는 방법을 보여 줍니다.

```
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'
```

이전 규칙 쿼리 문은 다음과 같은 형식으로 데이터를 출력합니다.

```
{
  "windspeed": 26.32020195042838
}
```

Important

이 규칙 쿼리 문은 각 배치에 있는 첫 번째 값 이외의 값 업데이트를 무시합니다. 각 배치에는 최대 10개의 값이 포함될 수 있습니다. 나머지 값을 포함해야 하는 경우 자산 속성 값을 다른 서비스로 출력하는 보다 복잡한 솔루션을 설정해야 합니다. 예를 들어, 배열의 각 값을 다른 주

제에 다시 게시하는 AWS Lambda 작업이 포함된 규칙을 설정하고, 해당 주제를 쿼리하고 각 값을 원하는 규칙 작업에 게시하는 다른 규칙을 설정할 수 있습니다.

자산 속성 알리를 사용하여 Amazon S3로 데이터 내보내기

에서 수신되는 데이터를 계정의 Amazon S3 버킷 AWS IoT SiteWise 으로 내보낼 수 있습니다. 기록 보고서 생성하거나 복잡한 데이터 분석에 사용할 수 있는 형식으로 데이터를 백업할 수 있습니다.

시계열 데이터를 내보내려면 콜드 티어 기능을 AWS IoT SiteWise 활성화하여 데이터를 Amazon S3 버킷에 저장합니다. 자세한 내용은 [의 데이터 스토리지 관리를 AWS IoT SiteWise](#) 참조하세요.

자산 모델 및 자산 메타데이터를 내보내려면 대량 작업 기능을 AWS IoT SiteWise 사용하여 메타데이터를 Amazon S3 버킷으로 내보냅니다. 자세한 내용은 [자산 및 모델을 사용한 대량 작업을](#) 참조하세요.

Grafana AWS IoT SiteWise 와 통합

Grafana는 대시보드의 데이터를 시각화하고 모니터링하는 데 사용되는 데이터 시각화 플랫폼입니다. Grafana 버전 10.4.0 이상에서는 AWS IoT SiteWise 플러그인을 사용하여 Grafana 대시보드에서 AWS IoT SiteWise 자산 데이터를 시각화합니다. 사용자는 단일 Grafana 대시보드를 사용하여 여러 AWS 소스(예: AWS IoT SiteWise Amazon Timestream, Amazon CloudWatch) 및 기타 데이터 소스의 데이터를 시각화할 수 있습니다.

AWS IoT SiteWise 플러그인을 사용할 수 있는 두 가지 옵션이 있습니다.

- 로컬 Grafana 서버

관리하는 Grafana 서버에서 AWS IoT SiteWise 플러그인을 설정할 수 있습니다. 플러그인 추가 및 사용 방법에 대한 자세한 내용은 GitHub 웹 사이트의 [AWS IoT SiteWise 데이터 소스 README](#) 파일을 참조하세요.

- AWS Grafana용 관리형 서비스

AWS Managed Service for Grafana(AMG)에서 AWS IoT SiteWise 플러그인을 사용할 수 있습니다. AMG는 사용자를 대신하여 Grafana 서버를 관리하므로 사용자는 하드웨어나 기타 Grafana 인프라를 구축, 패키징 또는 배포하지 않고도 데이터를 시각화할 수 있습니다. 자세한 내용은 AWS Grafana용 관리형 서비스 사용 설명서의 다음 주제를 참조하세요.

- [Amazon Grafana용 관리형 서비스\(AMG\)란?](#)
- [AWS IoT SiteWise 데이터 소스 사용](#)

Example Grafana 대시보드 예시

다음 Grafana 대시보드는 [데모 풍력 발전 단지](#)를 시각화합니다. [Grafana Play](#) 웹 사이트에서 이 데모 대시보드에 액세스할 수 있습니다.



AWS IoT SiteWise 및 통합 AWS IoT TwinMaker

와 통합하면 AWS IoT SiteWise 콘솔에서 AWS IoT SiteWise 데이터 검색 ExecuteQuery API 및 고급 자산 검색 AWS IoT SiteWise와 같은 강력한 기능에 AWS IoT TwinMaker 액세스할 수 있습니다. 서비스를 통합하고 이러한 기능을 사용하려면 먼저 통합을 활성화해야 합니다.

주제

- [통합 활성화](#)

- [AWS IoT SiteWise 및 통합 AWS IoT TwinMaker](#)

통합 활성화

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다. JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. AWS IoT SiteWise 지원되는 작업에 대한 자세한 내용은 서비스 승인 참조의에서 [정의한 작업을 AWS IoT SiteWise](#) 참조하세요.

AWS IoT TwinMaker 서비스 연결 역할에 대한 자세한 내용은 AWS IoT TwinMaker 사용 설명서의에 [대한 서비스 연결 역할을 AWS IoT TwinMaker](#) 참조하세요.

AWS IoT SiteWise 및를 통합하려면 먼저가 AWS IoT TwinMaker 연결된 워크스페이스와 AWS IoT SiteWise 통합할 수 있도록 허용하는 다음 권한을 부여 AWS IoT TwinMaker해야 합니다.

- `iotsitewise:EnableSiteWiseIntegration` - AWS IoT SiteWise 를 연결된 AWS IoT TwinMaker 워크스페이스와 통합할 수 있습니다. 이 통합 AWS IoT TwinMaker 을 통해서는 AWS IoT TwinMaker 서비스 연결 역할을 AWS IoT SiteWise 통해의 모든 모델링 정보를 읽을 수 있습니다. 이 권한을 활성화하려면 사용자의 IAM 역할에 다음 정책을 추가합니다.

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

AWS IoT SiteWise 및 통합 AWS IoT TwinMaker

AWS IoT SiteWise 및를 통합하려면 다음이 있어야 AWS IoT TwinMaker합니다.

- AWS IoT SiteWise 계정에 설정된 서비스 연결 역할

- AWS IoT TwinMaker 계정에 설정된 서비스 연결 역할
- AWS IoT TwinMaker 리전IoTSiteWiseDefaultWorkspace의 계정에 ID가 인 워크스페이스입니다.

AWS IoT SiteWise 콘솔을 사용하여 통합하려면

콘솔에 AWS IoT TwinMaker와 통합 배너가 표시되면 권한 부여를 선택합니다. 사전 조건은 계정에 생성됩니다.

를 사용하여 통합하려면 AWS CLI

AWS IoT TwinMaker 를 사용하여 AWS IoT SiteWise 및를 통합하려면 다음 명령을 AWS CLI입력합니다.

1. `iotsitewise.amazonaws.com`의 `AWSServiceName`을 사용하여 `CreateServiceLinkedRole`을 직접적으로 호출합니다.

```
aws iam create-service-linked-role --aws-service-name iotsitewise.amazonaws.com
```

2. `iottwinmaker.amazonaws.com`의 `AWSServiceName`을 사용하여 `CreateServiceLinkedRole`을 직접적으로 호출합니다.

```
aws iam create-service-linked-role --aws-service-name iottwinmaker.amazonaws.com
```

3. `IoTSiteWiseDefaultWorkspace`의 ID를 사용하여 `CreateWorkspace`을 직접적으로 호출합니다.

```
aws iottwinmaker create-workspace --workspace-id IoTSiteWiseDefaultWorkspace
```

Lookout for Equipment로 이상 감지

Note

이상 감지는 Amazon Lookout for Equipment가 제공되는 리전에서만 사용할 수 있습니다.

Amazon Lookout for Equipment AWS IoT SiteWise 와 통합하여 산업 장비의 이상 탐지 및 예측 유지 관리를 통해 산업 장비에 대한 인사이트를 얻을 수 있습니다. Lookout for Equipment는 비정상적

인 장비 동작을 감지하고 잠재적 장애를 식별하는 산업용 장비 모니터링용 기계 학습(ML) 서비스입니다. Lookout for Equipment를 사용하면 예측 유지 보수 프로그램을 구현하고 최적화되지 않은 장비 프로세스를 식별할 수 있습니다. Lookout for Equipment에 대한 자세한 내용은 Amazon Lookout for Equipment 사용 설명서의 [Amazon Lookout for Equipment란 무엇인가요?](#)를 참조하세요.

ML 모델을 훈련하여 비정상적인 장비 동작을 감지하도록 예측을 생성하면 Lookout for Equipment에 자산 속성 값을 AWS IoT SiteWise 전송하여 비정상적인 장비 동작을 감지하도록 ML 모델을 훈련합니다. 자산 모델에 대한 예측 정의를 정의하려면 Lookout for Equipment가 데이터에 액세스하는 데 필요한 IAM 역할과 Lookout for Equipment로 보내고 처리된 데이터를 Amazon S3로 전송할 속성을 지정합니다. 자세한 내용은 [에서 자산 모델 생성 AWS IoT SiteWise](#) 단원을 참조하십시오.

AWS IoT SiteWise 및 Lookout for Equipment를 통합하려면 다음과 같은 상위 수준 단계를 수행합니다.

- 추적하려는 속성을 개괄적으로 설명하는 자산 모델에 예측 정의를 추가합니다. 예측 정의는 해당 자산 모델을 기반으로 자산에 대한 예측을 생성하는 데 사용되는 측정, 변환 및 지표의 재사용 가능한 모음입니다.
- 제공한 기록 데이터를 기반으로 예측을 훈련합니다.
- 특정 예측을 실행할 AWS IoT SiteWise 빈도를 알려주는 추론을 예약합니다.

추론이 예약되면 Lookout for Equipment 모델은 장비에서 수신하는 데이터를 모니터링하고 장비 동작의 이상을 찾습니다. AWS IoT SiteWise GET API 작업 또는 Lookout for Equipment 콘솔을 사용하여 SiteWise Monitor에서 결과를 보고 분석할 수 있습니다. 자산 모델의 경보 감지기로 경보를 생성하여 이상 장비 동작에 대한 알림을 받을 수도 있습니다.

주제

- [예측 정의 추가\(콘솔\)](#)
- [예측 훈련\(콘솔\)](#)
- [예측에 대한 추론 시작 또는 중지\(콘솔\)](#)
- [예측 정의 추가\(CLI\)](#)
- [예측 훈련 및 추론 시작\(CLI\)](#)
- [예측 훈련\(CLI\)](#)
- [예측에 대한 추론 시작 또는 중지\(CLI\)](#)

예측 정의 추가(콘솔)

에서 수집한 데이터를 Lookout for Equipment AWS IoT SiteWise 로 전송하려면 자산 모델에 AWS IoT SiteWise 예측 정의를 추가해야 합니다.

AWS IoT SiteWise 자산 모델에 예측 정의를 추가하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 모델을 선택하고 예측 정의를 추가할 자산 모델을 선택합니다.
3. 예측을 선택합니다.
4. 예측 정의 추가를 선택합니다.
5. 예측 정의에 대한 세부 정보를 정의합니다.
 - a. 고유한 이름과 예측 정의에 대한 설명을 입력합니다. 예측 정의를 생성한 후에는 이름을 변경할 수 없으므로 이름을 신중하게 선택합니다.
 - b. AWS IoT SiteWise 가 Amazon Lookout for Equipment와 자산 데이터를 공유하도록 허용하는 IAM 권한 역할을 생성하거나 선택합니다. 역할에는 다음과 같은 IAM 및 신뢰 정책이 있어야 합니다. 역할을 생성하는 데 도움이 필요하다면 [사용자 지정 신뢰 정책을 사용하여 역할 생성](#)을 참조하세요.

IAM 정책

```
{
  "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"
    }
]
}

```

신뢰 정책

```
{
```

```

"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. Next(다음)를 선택합니다.
6. Lookout for Equipment로 전송할 데이터 속성(측정값, 변환 및 지표)을 선택합니다.
 - a. (선택 사항) 측정값을 선택합니다.
 - b. (선택 사항) 변환을 선택합니다.
 - c. (선택 사항) 지표를 선택합니다.

- d. Next(다음)를 선택합니다.
7. 선택한 내용을 검토합니다. 자산 모델에 예측 정의를 추가하려면 요약 페이지에서 예측 정의 추가를 선택합니다.

활성 예측이 연결된 기존 예측 정의를 편집하거나 삭제할 수도 있습니다.

예측 훈련(콘솔)

자산 모델에 예측 정의를 추가한 후 자산에 있는 예측을 훈련할 수 있습니다.

에서 예측을 훈련하려면 AWS IoT SiteWise

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택하고 모니터링할 자산을 선택합니다.
3. 예측을 선택합니다.
4. 훈련하려는 예측을 선택합니다.
5. 작업에서 훈련 시작을 선택하고 다음을 수행합니다.
 - a. 예측 세부 정보에서가 자산 데이터를 Lookout for Equipment와 AWS IoT SiteWise 공유할 수 있도록 허용하는 IAM 권한 역할을 선택합니다. 새 역할을 생성해야 하는 경우, 새 역할 생성을 선택합니다.
 - b. 훈련 데이터 설정에서 훈련 데이터 시간 범위를 입력하여 예측 훈련에 사용할 데이터를 선택합니다.
 - c. (선택 사항) 사후 처리 후 데이터의 샘플링 속도를 선택합니다.
 - d. (선택 사항) 데이터 레이블에 Amazon S3 버킷 및 레이블 지정 데이터를 포함하는 접두사를 제공합니다. 데이터 레이블 지정에 대한 자세한 내용은 Amazon Lookout for Equipment 사용 설명서의 [데이터 레이블 지정](#)을 참조하세요.
 - e. Next(다음)를 선택합니다.
6. (선택 사항) 훈련을 완료하는 즉시 예측을 활성화하려면 고급 설정에서 훈련 후 예측 자동 활성화를 선택한 다음 다음을 수행합니다.
 - a. 입력 데이터의 데이터 업로드 빈도에서 데이터가 업로드되는 빈도를 정의하고 오프셋 지연 시간에서는 사용할 버퍼의 양을 정의합니다.
 - b. Next(다음)를 선택합니다.
7. 예측 세부 정보를 검토하고 저장 및 시작을 선택합니다.

예측에 대한 추론 시작 또는 중지(콘솔)

Note

Lookout for Equipment 요금은 AWS IoT SiteWise 및 Lookout for Equipment 간에 전송된 데이터를 사용하여 예약된 추론에 적용됩니다. 자세한 내용은 [Amazon Lookout for Equipment 요금](#)을 참조하세요.

예측 `lookoutequipment:CreateDataset`를 추가했지만 훈련 후 활성화하지 않은 경우 자산 모니터링을 시작하려면 활성화해야 합니다.

예측에 대한 추론을 시작하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택하고 예측이 추가될 자산을 선택합니다.
3. 예측을 선택합니다.
4. 활성화하려는 예측을 선택합니다.
5. 작업에서 추론 시작을 선택하고 다음을 수행합니다.
 - a. 입력 데이터의 데이터 업로드 빈도에서 데이터가 업로드되는 빈도를 정의하고 오프셋 지연 시간에서는 사용할 버퍼의 양을 정의합니다.
 - b. 저장 및 테스트를 선택합니다.

예측에 대한 추론을 중지하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 자산을 선택하고 예측이 추가될 자산을 선택합니다.
3. 예측을 선택합니다.
4. 중지하려는 예측을 선택합니다.
5. 작업에서 추론 중지를 선택합니다.

예측 정의 추가(CLI)

신규 또는 기존 자산 모델에 대한 예측 정의를 정의하려면 AWS Command Line Interface ()를 사용할 수 있습니다. 자산 모델에서 예측 정의를 정의한 후 Lookout for Equipment를 사용하여 이상 탐지를 수행하기 AWS IoT SiteWise 위해의 자산에 대한 예측을 훈련하고 추론을 예약합니다.

사전 조건

이 단계를 완료하려면 자산 모델과 하나 이상의 자산을 생성해야 합니다. 자세한 내용은 [자산 모델 생성\(AWS CLI\)](#) 및 [자산 생성\(AWS CLI\)](#) 단원을 참조하세요.

처음 사용하는 경우 CreateBulkImportJob API 작업을 호출하여 자산 속성 값을 가져와 AWS IoT SiteWise야 합니다. AWS IoT SiteWise이 값은 모델을 훈련하는 데 사용됩니다. 자세한 내용은 [AWS IoT SiteWise 대량 가져오기 작업 생성\(AWS CLI\)](#) 단원을 참조하십시오.

예측 정의를 추가하려면 다음을 수행하세요.

1. `asset-model-payload.json`이라는 파일을 생성합니다. 다른 섹션의 단계에 따라 자산 모델의 세부 정보를 파일에 추가하되, 자산 모델을 만들거나 업데이트하라는 요청을 제출하지 마세요.
 - 자산 모델을 만드는 방법에 대한 자세한 내용은 [자산 모델 생성\(AWS CLI\)](#) 섹션을 참조하세요.
 - 기존 자산 모델을 업데이트하는 방법에 대한 자세한 내용은 [자산 또는 구성 요소 모델 업데이트\(AWS CLI\)](#) 섹션을 참조하세요.
2. 다음 코드를 추가하여 Lookout for Equipment 복합 모델(`assetModelCompositeModels`)을 자산 모델에 추가합니다.
 - **Property**를 포함하려는 속성의 ID로 바꿉니다. 이러한 ID를 가져오려면 [DescribeAssetModel](#)을 직접적으로 호출합니다.
 - **RoleARN**를 Lookout for Equipment가 AWS IoT SiteWise 데이터에 액세스할 수 있도록 허용하는 IAM 역할의 ARN으로 바꿉니다.

```
{
  ...
  "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. 자산 모델을 만들거나 기존 자산 모델을 업데이트합니다. 다음 중 하나를 수행합니다.

- 다음 명령을 실행하여 자산 모델을 생성합니다.

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-
payload.json
```

- 다음 명령을 실행하여 기존 자산 모델을 업데이트합니다. *asset-model-id*를 모니터링하고자 하는 작업의 ID로 바꿉니다.

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
```

```
--cli-input-json file://asset-model-payload.json
```

명령을 실행한 후 응답에서 `assetModelId`를 기록합니다.

예측 훈련 및 추론 시작(CLI)

이제 예측 정의가 정의되었으므로 이를 기반으로 자산을 훈련하고 추론을 시작할 수 있습니다. 예측을 훈련하지만 추론을 시작하지 않으려면 [예측 훈련\(CLI\)](#) 섹션으로 건너뛰세요. 예측을 훈련하고 자산에 대한 추론을 시작하려면 대상 리소스의 `assetId`가 필요합니다.

예측 추론을 훈련하고 시작하려면 다음을 수행하세요.

1. 다음 명령을 실행하여 `assetModelCompositeModelSummaries` 아래의 `assetModelCompositeModelId`를 찾습니다. *asset-model-id*를 [자산 또는 구성 요소 모델 업데이트\(AWS CLI\)](#)에서 생성한 자산 모델의 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 다음 명령을 실행하여 `TrainingWithInference` 작업의 `actionDefinitionId`를 찾습니다. *asset-model-id*를 이전 단계에서 사용된 ID로 바꾸고 *asset-model-composite-model-id*를 이전 단계에서 반환된 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. `train-start-inference-prediction.json`이라는 파일을 생성하고 다음을 대체하여 다음 코드를 추가합니다.

- *asset-id*를 대상 자산의 ID로
- *action-definition-id*를 `TrainingWithInference` 작업의 ID로
- *StartTime*을 Epoch 초 단위로 제공되는 훈련 데이터 시작 시점으로
- *EndTime*을 Epoch 초 단위로 제공되는 훈련 데이터 종료 시점으로
- *TargetSamplingRate*를 Lookout for Equipment에 의해 사후 처리된 후의 데이터 샘플링 속도로 허용되는 값은 PT1S | PT5S | PT10S | PT15S | PT30S | PT1M | PT5M | PT10M | PT15M | PT30M | PT1H입니다.

```
{
  "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. 다음 명령을 실행하여 훈련 및 추론을 시작합니다.

```
aws iotsitewise execute-action --cli-input-json file://train-start-inference-prediction.json
```

예측 훈련(CLI)

이제 예측 정의가 정의되었으므로 이를 기반으로 자산을 훈련할 수 있습니다. 자산에 대한 예측을 훈련하려면 대상 리소스의 `assetId`가 필요합니다.

예측을 훈련하려면 다음을 수행하세요.

1. 다음 명령을 실행하여 `assetModelCompositeModelSummaries` 아래의 `assetModelCompositeModelId`를 찾습니다. *asset-model-id*를 [자산 또는 구성 요소 모델 업데이트\(AWS CLI\)](#)에서 생성한 자산 모델의 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 다음 명령을 실행하여 Training 작업의 `actionDefinitionId`를 찾습니다. *asset-model-id*를 이전 단계에서 사용된 ID로 바꾸고 *asset-model-composite-model-id*를 이전 단계에서 반환된 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model-composite-model \
```

```
--asset-model-id asset-model-id \
--asset-model-composite-model-id asset-model-composite-model-id \
```

3. train-prediction.json이라는 파일을 생성하고 다음을 대체하여 다음 코드를 추가합니다.

- *asset-id*를 대상 자산의 ID로
- *action-definition-id*를 훈련 작업의 ID로
- *StartTime*을 Epoch 초 단위로 제공되는 훈련 데이터 시작 시점으로
- *EndTime*을 Epoch 초 단위로 제공되는 훈련 데이터 종료 시점으로
- (선택 사항) *BucketName*을 레이블 데이터를 저장하는 Amazon S3 버킷의 이름으로
- (선택 사항) *Prefix*를 Amazon S3 버킷과 연결된 접두사로
- *TargetSamplingRate*를 Lookout for Equipment에 의해 사후 처리된 후의 데이터 샘플링 속도로 허용되는 값은 PT1S | PT5S | PT10S | PT15S | PT30S | PT1M | PT5M | PT10M | PT15M | PT30M | PT1H입니다.

Note

버킷 이름과 접두사를 둘 다 포함하거나 둘 다 포함하지 않습니다.

```
{
  "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. 훈련을 시작하려면 다음 명령을 실행합니다.

```
aws iotsitewise execute-action --cli-input-json file://train-prediction.json
```

추론을 시작하려면 먼저 훈련을 완료해야 합니다. 훈련 상태를 확인하려면 다음 중 하나를 수행하세요.

- 콘솔에서 예측이 켜져 있는 자산으로 이동합니다.
- 에서 `trainingStatus` 속성 `propertyId`의를 `BatchGetAssetPropertyValue` 사용하여 AWS CLI호출합니다.

예측에 대한 추론 시작 또는 중지(CLI)

예측이 훈련되면 추론을 시작하여 Lookout for Equipment에 자산 모니터링을 시작하도록 지시할 수 있습니다. 추론을 시작하거나 중지하려면 대상 리소스의 `assetId`가 필요합니다.

추론을 시작하려면 다음을 수행하세요.

1. 다음 명령을 실행하여 `assetModelCompositeModelSummaries` 아래의 `assetModelCompositeModelId`를 찾습니다. *asset-model-id*를 [자산 또는 구성 요소 모델 업데이트\(AWS CLI\)](#)에서 생성한 자산 모델의 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 다음 명령을 실행하여 Inference 작업의 `actionDefinitionId`를 찾습니다. *asset-model-id*를 이전 단계에서 사용된 ID로 바꾸고 *asset-model-composite-model-id*를 이전 단계에서 반환된 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. `start-inference.json`이라는 파일을 생성하고 다음을 대체하여 다음 코드를 추가합니다.

- *asset-id*를 대상 자산의 ID로
- *action-definition-id*를 시작 추론 작업의 ID로
- *Offset*을 사용할 버퍼의 양으로
- *Frequency*를 데이터가 업로드되는 빈도로

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
```

```
"actionPayload":{ "stringValue": "{\\"l4EInference\\": {\\"inferenceMode\\":\\"START\\",\\"dataDelayOffsetInMinutes\\": Offset, \\"dataUploadFrequency\\": \\"Frequency\\"}}}
```

4. 추론을 실행하려면 다음 명령을 실행합니다.

```
aws iotsitewise execute-action --cli-input-json file://start-inference.json
```

추론을 중지하려면 다음을 수행하세요.

1. 다음 명령을 실행하여 `assetModelCompositeModelSummaries` 아래의 `assetModelCompositeModelId`를 찾습니다. *asset-model-id*를 [자산 또는 구성 요소 모델 업데이트\(AWS CLI\)](#)에서 생성한 자산 모델의 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 다음 명령을 실행하여 Inference 작업의 `actionDefinitionId`를 찾습니다. *asset-model-id*를 이전 단계에서 사용된 ID로 바꾸고 *asset-model-composite-model-id*를 이전 단계에서 반환된 ID로 바꿉니다.

```
aws iotsitewise describe-asset-model-composite-model \
  --asset-model-id asset-model-id \
  --asset-model-composite-model-id asset-model-composite-model-id \
```

3. `stop-inference.json`이라는 파일을 생성하고 다음을 대체하여 다음 코드를 추가합니다.

- *asset-id*를 대상 자산의 ID로
- *action-definition-id*를 시작 추론 작업의 ID로

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-id",
  "actionPayload":{ "stringValue": "{\\"l4EInference\\":{\\"inferenceMode\\":\\"STOP\\",\\"dataDelayOffsetInMinutes\\": Offset, \\"dataUploadFrequency\\": \\"Frequency\\"}}}
```


4. 추론을 실행하려면 다음 명령을 실행합니다.

```
aws iotsitewise execute-action --cli-input-json file://stop-inference.json
```

에서 데이터 스토리지 관리 AWS IoT SiteWise

다음 스토리지 계층에 데이터를 저장 AWS IoT SiteWise 하도록을 구성할 수 있습니다.

핫 티어

핫 스토리지 계층은 AWS IoT SiteWise 관리형 시계열 스토리지입니다. 핫 티어는 쓰기 후 읽기 지연 시간이 낮고 자주 액세스하는 데이터에 가장 효과적입니다. 핫 티어에 저장된 데이터는 장비의 최신 측정값에 빠르게 액세스해야 하는 산업 애플리케이션에서 사용됩니다. 여기에는 대화형 대시 보드를 사용하여 실시간 지표를 시각화하는 애플리케이션 또는 작업을 모니터링하고 성능 문제를 식별하기 위해 경보를 실행하는 애플리케이션이 포함됩니다.

기본적으로 수집된 데이터는 핫 티어에 AWS IoT SiteWise 저장됩니다. 핫 티어의 보존 기간을 정의할 수 있습니다. 그 후에는 구성에 따라 핫 티어의 데이터를 워م 또는 콜드 티어 스토리지로 AWS IoT SiteWise 이동합니다. 최상의 성능과 비용 효율성을 위해 핫 티어 보존 기간을 데이터를 자주 검색하는 데 걸리는 시간보다 길게 설정합니다. 이는 실시간 지표, 경고 및 모니터링 시나리오에 사용됩니다. 보존 기간이 설정되지 않은 경우 데이터는 핫 티어에 무기한 저장됩니다.

웜 티어

웜 스토리지 계층은 기록 데이터의 비용 효율적인 스토리지에 효과적인 AWS IoT SiteWise 관리형 계층입니다. 중간 정도의 쓰기 후 읽기 지연 시간으로 대용량 데이터를 검색하는 데 가장 적합합니다. 웜 티어를 사용하여 대규모 워크로드에 필요한 기록 데이터를 저장합니다. 예를 들어 분석, 비즈니스 인텔리전스 애플리케이션(BI), 보고 도구 및 기계 학습(ML) 모델의 훈련을 위한 데이터 검색에 사용됩니다. 콜드 스토리지 티어를 활성화하면 웜 티어 보존 기간을 정의할 수 있습니다. 보존 기간이 끝나면 웜 티어에서 데이터를 AWS IoT SiteWise 삭제합니다.

콜드 티어

콜드 스토리지 티어는 Amazon S3 버킷을 사용하여 거의 사용되지 않는 데이터를 저장합니다. 콜드 티어가 활성화된 상태에서 AWS IoT SiteWise 는 측정값, 지표, 변환 및 집계, 자산 모델 정의를 포함하여 시계열을 6시간마다 복제합니다. 콜드 티어는 기록 보고서 및 백업에 대한 긴 읽기 지연 시간을 허용하는 데이터를 저장하는 데 사용됩니다.

주제

- [에서 스토리지 설정 구성 AWS IoT SiteWise](#)
- [의 스토리지 설정 문제 해결 AWS IoT SiteWise](#)
- [콜드 티어에 저장된 데이터의 파일 경로 및 스키마](#)

에서 스토리지 설정 구성 AWS IoT SiteWise

서비스 관리형 워م 티어 스토리지에 옵트인하고 쿨드 티어에 데이터를 복제하도록 스토리지 설정을 구성할 수 있습니다. 워م 티어와 핫 티어의 보존 기간에 대한 자세한 내용은 [데이터 보존 영향](#) 섹션을 참조하세요. 스토리지 설정을 구성하는 동안 다음을 수행합니다.

- 핫 티어 보존 - 데이터가 삭제되기 전에 핫 티어에 저장되고 스토리지 설정에 따라 서비스 관리형 워م 티어 스토리지 또는 쿨드 티어 스토리지로 이동하는 기간에 대한 보존 기간을 설정합니다. AWS IoT SiteWise 는 보존 기간이 끝나기 전에 핫 티어에 있던 모든 데이터를 삭제합니다. 보존 기간을 설정하지 않으면 데이터가 무기한 핫 티어에 저장됩니다.
- 워م 티어 보존 - 스토리지에서 데이터를 삭제하고 고객 관리형 쿨드 티어 AWS IoT SiteWise 스토리지로 이동하기 전에 데이터가 워م 티어에 저장되는 기간의 보존 기간을 설정합니다. 보존 기간이 끝나기 전에 존재했던 워م 티어의 모든 데이터를 AWS IoT SiteWise 삭제합니다. 보존 기간이 설정되지 않은 경우 데이터는 워م 티어에 무기한 저장됩니다.

Note

쿼리 성능을 개선하려면 워م 티어 스토리지에 핫 티어 보존 기간을 설정합니다.

핫 티어 스토리지와 워م 티어 스토리지의 데이터 보존 영향

- 핫 티어 스토리지의 보존 기간을 줄이면 데이터가 핫 티어에서 워م 티어 또는 쿨드 티어로 영구적으로 이동됩니다. 워م 티어의 보존 기간을 줄이면 데이터가 쿨드 티어로 이동하고 워م 티어에서 영구적으로 삭제됩니다.
- 핫 또는 워م 티어 스토리지의 보존 기간을 늘리면 변경 사항이 그때 AWS IoT SiteWise 부터 로 전송되는 데이터에 영향을 미칩니다.는 핫 티어를 채우기 위해 워م 또는 쿨드 스토리지에서 데이터를 검색하지 AWS IoT SiteWise 않습니다. 예를 들어 핫 티어 스토리지의 보존 기간이 처음에 30일 동안 설정된 다음 60일로 증가하면 핫 티어 스토리지에 60일 분량의 데이터가 포함되려면 30일이 걸립니다.

주제

- [웜 티어에 대한 스토리지 설정 구성\(콘솔\)](#)
- [웜 티어에 대한 스토리지 설정 구성\(AWS CLI\)](#)
- [쿨드 티어에 대한 스토리지 설정 구성\(콘솔\)](#)
- [쿨드 티어에 대한 스토리지 설정 구성\(AWS CLI\)](#)

웜 티어에 대한 스토리지 설정 구성(콘솔)

다음 절차에서는 AWS IoT SiteWise 콘솔의 웜 티어에 데이터를 복제하도록 스토리지 설정을 구성하는 방법을 보여줍니다.

콘솔을 사용하여 저장 설정을 구성하는 방법

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창의 설정 아래에서 저장을 선택합니다.
3. 오른쪽 상단 모서리에서 편집을 선택합니다.
4. 작업 편집 페이지에서 다음을 수행합니다.
5. 핫 티어 설정에서 다음을 수행합니다.
 - 데이터가 삭제되기 전에 핫 티어에 저장되고 서비스 관리형 웜 티어 스토리지로 이동되는 기간인 보존 기간을 설정하려면 보존 기간 활성화를 선택합니다.
 - 보존 기간을 구성하려면 정수를 입력하고 단위를 선택합니다. 보존 기간은 30일 이상이어야 합니다.
6. (권장) 웜 티어 설정에서 다음을 수행합니다.
 - 웜 티어 스토리지에 옵트인하려면 웜 티어 스토리지의 옵트인 확인을 선택하고 웜 티어 스토리지에 옵트인합니다.
 - (선택 사항) 보존 기간을 구성하려면 정수를 입력하고 단위를 선택합니다. 보존 기간은 365일 이상이어야 합니다.

AWS IoT SiteWise 는 보존 기간 이전에 존재했던 웜 티어의 데이터를 삭제합니다. 보존 기간을 설정하지 않으면 데이터가 무기한 저장됩니다.

Note

- 웜 티어에 옵트인하면 구성이 한 번만 표시됩니다.
- 핫 티어 보존을 설정하려면 웜 티어 또는 콜드 티어 스토리지가 있어야 합니다. 비용 효율성 및 과거 데이터 검색을 위해서는 웜 티어에 장기 데이터를 저장할 것을 AWS IoT SiteWise 권장합니다.

- 워م 티어 보존을 설정하려면 콜드 티어 스토리지가 있어야 합니다.

7. 설정을 저장하려면 저장을 선택합니다.

AWS IoT SiteWise 스토리지 섹션에서 워م 티어 스토리지는 다음 상태 중 하나에 있습니다.

- 활성화됨 - 데이터가 핫 티어 보존 기간 이전에 존재한 경우 AWS IoT SiteWise 는 데이터를 워م 티어로 이동합니다.
- 비활성화됨 - 워م 티어 스토리지가 비활성화되었습니다.

웜 티어에 대한 스토리지 설정 구성(AWS CLI)

AWS CLI 및 다음 명령을 사용하여 데이터를 워م 티어로 이동하도록 스토리지 설정을 구성할 수 있습니다.

기존 구성을 재정의하지 않으려면 다음 명령을 실행하여 현재 스토리지 구성 정보를 검색합니다.

```
aws iotsitewise describe-storage-configuration
```

Example 기존 콜드 티어 구성이 없는 응답

```
{
  "storageType": "SITEWISE_DEFAULT_STORAGE",
  "disassociatedDataStorage": "ENABLED",
  "configurationStatus": {
    "state": "ACTIVE"
  },
  "lastUpdateDate": "2021-10-14T15:53:35-07:00",
  "warmTier": "DISABLED"
}
```

Example 기존 콜드 티어 구성이 있는 응답

```
{
  "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"
}

```

를 사용하여 워밍 티어에 대한 스토리지 설정 구성 AWS CLI

스토리지 설정을 구성하려면 다음 명령을 실행합니다. 를 AWS IoT SiteWise 스토리지 구성에 포함된 파일의 이름으로 file-name 바꿉니다.

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise 핫 티어와 워밍 티어를 사용한 구성

```

{
  "storageType": "SITEWISE_DEFAULT_STORAGE",
  "disassociatedDataStorage": "ENABLED",
  "warmTier": "ENABLED",
  "retentionPeriod": {
    "numberOfDays": hot-tier-retention-in-days
  }
}

```

hot-tier-retention-in-days는 30일 이상의 정수여야 합니다.

Example 응답

```

{
  "storageType": "SITEWISE_DEFAULT_STORAGE",
  "configurationStatus": {
    "state": "UPDATE_IN_PROGRESS"
  }
}

```

콜드 티어 스토리지가 활성화되어 있는 경우 [AWS CLI 및 기존 콜드 티어를 사용하여 스토리지 설정 구성](#) 섹션을 참조하세요.

AWS CLI 및 기존 콜드 티어를 사용하여 스토리지 설정 구성

기존 콜드 티어 스토리지 AWS CLI 와 함께 사용하여 스토리지 설정 구성

- 스토리지 설정을 구성하려면 다음 명령을 실행합니다. *file-name*을 AWS IoT SiteWise 스토리지 구성이 포함된 파일의 이름으로 바꿉니다.

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise 스토리지 구성

- amzn-s3-demo-bucket*을 Amazon S3 버킷 이름으로 바꿉니다.
- prefix*를 Amazon S3 접두사로 바꿉니다.
- aws-account-id*를 AWS 계정 ID로 바꿉니다.
- role-name*을가 Amazon S3로 데이터를 전송하도록 허용하는 Amazon S3 액세스 역할의 이름으로 바꿉 AWS IoT SiteWise 니다.
- hot-tier-retention-in-days*를 30일 이상의 정수로 바꿉니다.
- warm-tier-retention-in-days*를 365일 이상의 정수로 바꿉니다.

Note

AWS IoT SiteWise 는 콜드 티어의 보존 기간보다 오래된 웜 티어의 모든 데이터를 삭제합니다. 보존 기간을 설정하지 않으면 데이터가 무기한 저장됩니다.

```
{
  "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 응답

```

{
  "storageType": "MULTI_LAYER_STORAGE",
  "configurationStatus": {
    "state": "UPDATE_IN_PROGRESS"
  }
}

```

콜드 티어에 대한 스토리지 설정 구성(콘솔)

다음 절차에서는 AWS IoT SiteWise 콘솔의 콜드 티어에 데이터를 복제하도록 스토리지 설정을 구성하는 방법을 보여줍니다.

콘솔을 사용하여 저장 설정을 구성하는 방법

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창의 설정 아래에서 저장을 선택합니다.
3. 오른쪽 상단 모서리에서 편집을 선택합니다.
4. 작업 편집 페이지에서 다음을 수행합니다.
 - a. 스토리지 설정에서 콜드 티어 스토리지 활성화를 선택합니다. 기본적으로 콜드 티어 스토리지는 비활성화되어 있습니다.
 - b. S3 버킷 위치의 경우 기존 Amazon S3 버킷의 이름과 접두사를 입력합니다.

Note

- Amazon S3는 접두사를 Amazon S3 버킷의 폴더 이름으로 사용합니다. 접두사는 1~255자로 구성되어야 하며 슬래시(/)로 끝나야 합니다. AWS IoT SiteWise 데이터는 이 폴더에 저장됩니다.

- Amazon S3 버킷이 없는 경우에는 보기를 선택한 후 Amazon S3 콘솔에서 새 버킷을 생성합니다. 자세한 내용은 Amazon S3 사용 설명서에서 [첫 S3 버킷 생성](#)을 참조하세요.

c. S3 액세스 역할의 경우 다음 중 하나를 수행합니다.

- AWS 관리형 템플릿에서 역할 생성을 선택하면 Amazon S3 AWS IoT SiteWise 로 데이터를 전송할 수 있는 IAM 역할이 AWS 자동으로 생성됩니다.
- 기존 역할 사용을 선택하고 목록에서 생성한 에 역할을 선택합니다.

Note

- 이전 단계 및 IAM 정책에서 사용한 것과 동일한 Amazon S3 버킷 이름을 S3 버킷 위치에 사용해야 합니다.
- 역할에 다음 예에 표시된 것과 같은 권한이 있는지 확인합니다.

Example 권한 정책:

```
{
  "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/*"
      ]
    }
  ]
}
```

amzn-s3-demo-bucket을 Amazon S3 버킷 이름으로 바꿉니다.

- Amazon S3 버킷이 고객 관리형 KMS 키를 사용하여 암호화된 경우 KMS 키에는 kms:Decrypt 및 kms:GenerateDataKey 작업에 대한 IAM 역할이 있는 액세스 정책이 있어야 합니다.

- d. 핫 티어를 설정하려면 [웜 티어에 대한 스토리지 설정 구성\(콘솔\)](#)의 5단계를 참조하세요.
- e. (선택 사항) AWS IoT Analytics 통합의 경우 다음을 수행합니다.

Note

지원 종료 공지: 2025년 12월 15 AWS 일에는에 대한 지원을 종료합니다 AWS IoT Analytics. 2025년 12월 15일 이후에는 AWS IoT Analytics 콘솔 또는 AWS IoT Analytics 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 [AWS IoT Analytics 지원 종료를 참조하세요](#).

- i. 를 사용하여 데이터를 AWS IoT Analytics 쿼리하려면 활성화된 AWS IoT Analytics 데이터 스토어를 선택합니다.
- ii. AWS IoT SiteWise 는 데이터 스토어의 이름을 생성하거나 다른 이름을 입력할 수 있습니다.

AWS IoT SiteWise 는에 데이터 스토어를 자동으로 생성 AWS IoT Analytics 하여 데이터를 저장합니다. 데이터를 쿼리하려면를 사용하여 데이터 세트를 AWS IoT Analytics 생성할 수 있습니다. 자세한 내용은 AWS IoT Analytics 사용 설명서의 [AWS IoT SiteWise 데이터 작업을 참조하세요](#).

- f. 저장을 선택합니다.

AWS IoT SiteWise 스토리지 섹션에서 콜드 티어 스토리지는 다음 값 중 하나일 수 있습니다.

- 활성화됨 - 지정된 Amazon S3 버킷에 데이터를 AWS IoT SiteWise 복제합니다.
- 활성화 - AWS IoT SiteWise 가 콜드 티어 스토리지 활성화 요청을 처리하고 있습니다. 이 프로세스는 완료하는 데 몇 분 정도 걸립니다.
- Enable_Failed – 콜드 티어 스토리지 활성화 요청을 처리할 수 AWS IoT SiteWise 없습니다. Amazon CloudWatch Logs AWS IoT SiteWise 로 로그를 전송하도록 활성화한 경우 이러한 로그를 사용하여 문제를 해결할 수 있습니다. 자세한 내용은 [Amazon CloudWatch Logs를 사용하여 모니터링 단원을 참조하십시오](#).

- Disabled - 콜드 티어 스토리지가 비활성화되었습니다.

콜드 티어에 대한 스토리지 설정 구성(AWS CLI)

다음 절차에서는 AWS CLI를 사용하여 콜드 티어에 데이터를 복제하도록 스토리지 설정을 구성하는 방법을 보여줍니다.

를 사용하여 스토리지 설정을 구성하려면 AWS CLI

1. 계정의 Amazon S3 버킷으로 데이터를 내보내려면 다음 명령을 실행하여 스토리지 설정을 구성합니다. *file-name*을 AWS IoT SiteWise 스토리지 구성에 포함된 파일의 이름으로 바꿉니다.

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise 스토리지 구성

- *amzn-s3-demo-bucket*을 Amazon S3 버킷 이름으로 바꿉니다.
- *prefix*를 Amazon S3 접두사로 바꿉니다.
- *aws-account-id*를 AWS 계정 ID로 바꿉니다.
- *role-name*을가 Amazon S3로 데이터를 전송하도록 허용하는 Amazon S3 액세스 역할의 이름으로 바꿉 AWS IoT SiteWise 니다.
- *retention-in-days*를 30일 이상의 정수로 바꿉니다.

```
{
  "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

- AWS IoT SiteWise 스토리지 구성 및 IAM 정책에 동일한 Amazon S3 버킷 이름을 사용해야 합니다.
- 역할에 다음 예에 표시된 것과 같은 권한이 있는지 확인합니다.

Example 권한 정책:

```
{
  "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/*"
      ]
    }
  ]
}
```

amzn-s3-demo-bucket을 Amazon S3 버킷 이름으로 바꿉니다.

- Amazon S3 버킷이 고객 관리형 KMS 키를 사용하여 암호화된 경우 KMS 키에는 kms:Decrypt 및 kms:GenerateDataKey 작업에 대한 IAM 역할이 있는 액세스 정책이 있어야 합니다.

Example 응답

```
{
  "storageType": "MULTI_LAYER_STORAGE",
  "retentionPeriod": {
```

```

        "numberOfDays": 100,
        "unlimited": false
    },
    "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
    }
}

```

Note

에서 스토리지 구성을 업데이트 AWS IoT SiteWise 하는 데 몇 분 정도 걸릴 수 있습니다.

2. 스토리지 구성 정보를 검색하려면 다음 명령을 실행합니다.

```
aws iotsitewise describe-storage-configuration
```

Example 응답

```

{
    "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. Amazon S3 버킷으로 데이터 내보내기를 중지하려면 다음 명령을 실행하여 스토리지 설정을 구성합니다.

```
aws iotsitewise put-storage-configuration --storage-type SITEWISE_DEFAULT_STORAGE
```

Note

기본적으로 데이터는 핫 티어에만 저장됩니다 AWS IoT SiteWise.

Example 응답

```
{
  "storageType": "SITEWISE_DEFAULT_STORAGE",
  "configurationStatus": {
    "state": "UPDATE_IN_PROGRESS"
  }
}
```

4. 스토리지 구성 정보를 검색하려면 다음 명령을 실행합니다.

```
aws iotsitewise describe-storage-configuration
```

Example 응답

```
{
  "storageType": "SITEWISE_DEFAULT_STORAGE",
  "configurationStatus": {
    "state": "ACTIVE"
  },
  "lastUpdateDate": "2021-03-30T15:57:14-07:00"
}
```

(선택 사항) AWS IoT Analytics 데이터 스토어 생성(AWS CLI)**Note**

지원 종료 공지: 2025년 12월 15 AWS 일에는에 대한 지원을 종료합니다 AWS IoT Analytics. 2025년 12월 15일 이후에는 AWS IoT Analytics 콘솔 또는 AWS IoT Analytics 리소스에 더 이상 액세스할 수 없습니다. 자세한 내용은 [AWS IoT Analytics 지원 종료를 참조하세요](#).

AWS IoT Analytics 데이터 스토어는 데이터를 수신하고 저장하는 확장 가능하고 쿼리 가능한 리포지토리입니다. AWS IoT SiteWise 콘솔 또는 AWS IoT Analytics APIs를 사용하여 AWS IoT Analytics 데이터 스토어를 생성하여 AWS IoT SiteWise 데이터를 저장할 수 있습니다. 데이터를 쿼리하려면 사용하여 데이터 세트를 생성합니다 AWS IoT Analytics. 자세한 내용을 알아보려면 AWS IoT Analytics 사용 설명서의 [AWS IoT SiteWise 데이터를 사용하여 작업하기](#)를 참조하세요.

다음 단계에서는 AWS CLI 를 사용하여 데이터 스토어를 생성합니다 AWS IoT Analytics.

다음 명령을 실행하여 데이터 스토어를 생성합니다. *file-name*을 데이터 스토어 구성이 포함된 파일의 이름으로 바꿉니다.

```
aws iotanalytics create-datastore --cli-input-json file://file-name.json
```

Note

- 기존 Amazon S3 버킷의 이름을 지정해야 합니다. Amazon S3 버킷이 없는 경우에는 먼저 하나를 만들 수 있습니다. 자세한 내용은 Amazon S3 사용 설명서에서 [첫 S3 버킷 생성](#)을 참조하세요.
- AWS IoT SiteWise 스토리지 구성, IAM 정책 및 AWS IoT Analytics 데이터 스토어 구성에 동일한 Amazon S3 버킷 이름을 사용해야 합니다.

Example AWS IoT Analytics 데이터 스토어 구성

data-store-name 및 *amzn-s3-demo-bucket*을 AWS IoT Analytics 데이터 스토어 이름 및 Amazon S3 버킷 이름으로 바꿉니다.

```
{
  "datastoreName": "data-store-name",
  "datastoreStorage": {
    "iotSiteWiseMultiLayerStorage": {
      "customerManagedS3Storage": {
        "bucket": "amzn-s3-demo-bucket"
      }
    }
  },
  "retentionPeriod": {
    "numberOfDays": 90
  }
}
```

```
}
```

Example 응답

```
{
  "datastoreName": "datastore_IoTSiteWise_demo",
  "datastoreArn": "arn:aws:iotanalytics:us-west-2:123456789012:datastore/
datastore_IoTSiteWise_demo",
  "retentionPeriod": {
    "numberOfDays": 90,
    "unlimited": false
  }
}
```

의 스토리지 설정 문제 해결 AWS IoT SiteWise

다음 정보를 사용하여 스토리지 구성 관련 문제를 해결하세요.

문제

- [오류: 버킷이 존재하지 않습니다](#)
- [오류: Amazon S3 경로에 대한 액세스 거부됨](#)
- [오류: 역할 ARN을 맡을 수 없습니다](#)
- [오류: 리전 간 S3 버킷에 액세스하지 못함](#)

오류: 버킷이 존재하지 않습니다

솔루션: AWS IoT SiteWise Amazon S3 버킷을 찾을 수 없습니다. 현재 리전의 기존 Amazon S3 버킷의 이름을 입력해야 합니다.

오류: Amazon S3 경로에 대한 액세스 거부됨

솔루션: AWS IoT SiteWise Amazon S3 버킷에 액세스할 수 없습니다. 다음을 수행합니다.

- IAM 정책에서 지정한 것과 동일한 Amazon S3 버킷을 사용해야 합니다.
- 역할에 다음 예에 표시된 것과 같은 권한이 있는지 확인합니다.

Example 권한 정책

```
{
```



```

    "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/*"
        ]
      }
    ]
  }
}

```

amzn-s3-demo-bucket을 Amazon S3 버킷 이름으로 바꿉니다.

오류: 역할 ARN을 맡을 수 없습니다

Solution: AWS IoT SiteWise couldn't은 사용자를 대신하여 IAM 역할을 수임하지 않습니다. 역할이 다음 서비스를 신뢰하는지 확인합니다. [iotsitewise.amazonaws.com](https://docs.aws.amazon.com/iam/latest/userguide/iam-roles.html) 자세한 내용은 IAM 사용자 가이드의 [역할을 가정할 수 없습니다](#)를 참조하세요.

오류: 리전 간 S3 버킷에 액세스하지 못함

솔루션: 지정한 Amazon S3 버킷이 다른 AWS 리전에 있습니다. Amazon S3 버킷과 AWS IoT SiteWise 자산이 동일한 리전에 있는지 확인합니다.

콜드 티어에 저장된 데이터의 파일 경로 및 스키마

AWS IoT SiteWise 는 측정값, 지표, 변환 및 집계, 자산 및 자산 모델 정의를 포함하여 시계열을 복제하여 콜드 티어에 데이터를 저장합니다. 다음은 콜드 티어로 전송되는 데이터의 파일 경로와 스키마를 설명합니다.

주제

- [장비 데이터\(측정값\)](#)

- [지표, 변환, 집계](#)
- [자산 메타데이터](#)
- [자산 계층 구조 메타데이터](#)
- [스토리지 데이터 인덱스 파일](#)

장비 데이터(측정값)

AWS IoT SiteWise 는 6시간마다 한 번씩 장비 데이터(측정값)를 콜드 티어로 내보냅니다. 원시 데이터는 [Apache AVRO](#)(.avro) 형식으로 콜드 티어에 저장됩니다.

파일 경로

AWS IoT SiteWise 는 다음 템플릿을 사용하여 장비 데이터(측정)를 콜드 티어에 저장합니다.

```
{keyPrefix}/raw/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/raw_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Amazon S3의 원시 데이터에 대한 모든 파일 경로에는 다음과 같은 구성 요소가 포함되어 있습니다.

경로 구성 요소	설명
keyPrefix	AWS IoT SiteWise 스토리지 구성에서 지정한 Amazon S3 접두사입니다. Amazon S3에서는 접두사를 버킷의 폴더 이름으로 사용합니다.
raw	장비의 시계열 데이터(측정값)를 저장하는 폴더. raw 폴더는 접두사 폴더에 저장됩니다.
seriesBucket	00~ff 사이의 16진수입니다. 이 숫자는 timeSeriesId 에서 파생되었습니다. 이 파티션은가 콜드 티어에 AWS IoT SiteWise 쓸 때 처리량을 높이는 데 사용됩니다. Amazon Athena 를 사용하여 쿼리를 실행하면 세분화된 분할에 파티션을 사용하여 쿼리 성능을 개선할 수 있습니다. 자산 메타데이터의 seriesBucket 과 timeSeriesBucket 은 같은 숫자입니다.

경로 구성 요소	설명
startYear	시계열 데이터와 연결된 배타적 시작 시간의 연도.
startMonth	시계열 데이터와 연결된 배타적 시작 시간의 달.
startDay	시계열 데이터와 연결된 배타적 시작 시간의 달의 날짜.
fileName	파일 이름은 밑줄 (_)문자를 구분 기호로 사용하여 다음을 구분합니다. - raw 접두사입니다. - timeSeriesId 값입니다. - 시계열 데이터와 관련된 배타적 시작 시간의 epoch 타임스탬프. - 데이터 품질. 유효한 값: GOOD, BAD, 및 UNCERTAIN . 자세한 내용을 알아보려면 AWS IoT SiteWise API 참조의 AssetProperty를 참조하세요. 파일은 Snappy 압축을 사용하여 .avro 형식으로 저장됩니다.

Example 콜드 티어의 원시 데이터 파일 경로

```
keyPrefix/raw/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
raw_7020c8e2-e6db-40fa-9845-ed0ddddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

필드

콜드 티어로 내보내는 원시 데이터의 스키마에는 다음 필드가 포함됩니다.

AWS IoT SiteWise 는 향후 추가 필드가 도입될 수 있으므로 콜드 티어에서 원시 데이터를 읽는 시스템에서 스키마 진화 지원을 구현하도록 고객에게 조언합니다.

Null 데이터는 null인 모든 값 필드로 표시됩니다. 그러나 AWS IoT SiteWise APIs.

필드 이름	지원되는 유형	기본 형식	설명
seriesId	string	N/A	장비의 시계열 데이터 (측정값)를 식별하는 ID. 이 필드를 사용하여 쿼리에서 원시 데이터와 자산 메타데이터를 결합할 수 있습니다.
timeInSeconds	long	N/A	Unix epoch 형식의 타임스탬프 날짜(초)입니다. offsetInNanos 에서 나노초 단위의 데이터를 제공합니다.
offsetInNanos	long	N/A	timeInSeconds 의 나노초 오프셋.
quality	string	N/A	시계열 값의 품질.
doubleValue	double 또는 null	null	더블 유형의 시계열 데이터(이동 소숫점 수).
stringValue	string 또는 null	null	문자열 유형의 시계열 데이터(문자 시퀀스).
integerValue	int 또는 null	null	정수 유형의 시계열 데이터(정수).
booleanValue	boolean 또는 null	null	부울 유형의 시계열 데이터(true 또는 false).
jsonValue	string 또는 null	null	JSON 유형의 시계열 데이터(문자열로 저장

필드 이름	지원되는 유형	기본 형식	설명
			된 복잡한 데이터 유형).
recordVersion	long 또는 null	null	기록을 위한 버전 번호. 버전 번호를 사용하여 최신 레코드를 선택할 수 있습니다. 최신 레코드는 버전 번호가 더 큼니다.

Example 콜드 티어의 원시 데이터

```
{
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675887,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": 0.75,
  "stringValue": null,
  "integerValue": null,
  "booleanValue": null,
  "jsonValue": null,
  "recordVersion": 1
}, {
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675889,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": 0.69,
  "stringValue": null,
  "integerValue": null,
  "booleanValue": null,
  "jsonValue": null,
  "recordVersion": 2
}, {
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675890,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": 0.66,
  "stringValue": null,
  "integerValue": null,
  "booleanValue": null,
  "jsonValue": null,
  "recordVersion": 3
}, {
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675891,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": 0.92,
  "stringValue": null,
  "integerValue": null,
  "booleanValue": null,
  "jsonValue": null,
  "recordVersion": 4
}, {
  "seriesId": "e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeInSeconds": 1625675892,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "doubleValue": 0.73,
  "stringValue": null,
  "integerValue": null,
  "booleanValue": null,
  "jsonValue": null,
  "recordVersion": 5
}
```

지표, 변환, 집계

AWS IoT SiteWise 는 6시간마다 한 번씩 지표, 변환 및 집계를 콜드 티어로 내보냅니다. 지표, 변환 및 집계는 [Apache AVRO](#)(.avro) 형식으로 콜드 티어에 저장됩니다.

파일 경로

AWS IoT SiteWise 는 다음 템플릿을 사용하여 콜드 티어에 지표, 변환 및 집계를 저장합니다.

```
{keyPrefix}/agg/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/agg_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Amazon S3의 지표, 변환 및 집계에 대한 모든 파일 경로에는 다음 구성 요소가 포함되어 있습니다.

경로 구성 요소	설명
keyPrefix	AWS IoT SiteWise 스토리지 구성에서 지정한 Amazon S3 접두사입니다. Amazon S3에서는 접두사를 버킷의 폴더 이름으로 사용합니다.
agg	지표의 시계열 데이터를 저장하는 폴더. agg 폴더는 접두사 폴더에 저장됩니다.
seriesBucket	00~ff 사이의 16진수입니다. 이 숫자는 timeSeriesId 에서 파생되었습니다. 이 파티션은가 쿼리 처리에 AWS IoT SiteWise 쓸 때 처리량을 높이는 데 사용됩니다. Amazon Athena 를 사용하여 쿼리를 실행하면 세분화된 분할에 파티션을 사용하여 쿼리 성능을 개선할 수 있습니다. 자산 메타데이터의 seriesBucket 과 timeSeriesBucket 은 같은 숫자입니다.
startYear	시계열 데이터와 연결된 배타적 시작 시간의 연도.
startMonth	시계열 데이터와 연결된 배타적 시작 시간의 달.
startDay	시계열 데이터와 연결된 배타적 시작 시간의 달의 날짜.
fileName	파일 이름은 밑줄 (_)문자를 구분 기호로 사용하여 다음을 구분합니다. - raw 접두사입니다. - timeSeriesId 값입니다.

경로 구성 요소	설명
	- 시계열 데이터와 관련된 배타적 시작 시간의 epoch 타임스탬프. - 데이터 품질. 유효한 값: GOOD, BAD, 및 UNCERTAIN . 자세한 내용을 알아보려면 AWS IoT SiteWise API 참조의 AssetProperty를 참조하세요. 파일은 Snappy 압축을 사용하여 .avro 형식으로 저장됩니다.

Example 콜드 티어의 지표에 대한 파일 경로

```
keyPrefix/agg/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/agg_7020c8e2-e6db-40fa-9845-ed0ddddd4c77d_95e63da7-d34e-43e1-bc6f-1b490154b07a_1609577700_GOOD.avro
```

필드

콜드 티어로 내보내는 지표, 변환 및 집계 스키마에는 다음 필드가 포함됩니다.

필드 이름	지원되는 유형	기본 형식	설명
seriesId	string	N/A	장비, 지표 또는 변환의 시계열 데이터를 식별하는 ID. 이 필드를 사용하여 쿼리에서 원시 데이터와 자산 메타데이터를 결합할 수 있습니다.
timeInSeconds	long	N/A	Unix epoch 형식의 타임스탬프 날짜(초)입니다. offsetInNanos 에서 나노초 단

필드 이름	지원되는 유형	기본 형식	설명
			위의 데이터를 제공합니다.
offsetInNanos	long	N/A	timeInSeconds 의 나노초 오프셋.
quality	string	N/A	자산 데이터를 필터링하는 데 사용되는 품질.
resolution	string	N/A	데이터를 집계하는 데 걸리는 시간 간격.
count	double 또는 null	null	현재 시간 간격 동안 해당 변수에 대한 총 데이터 포인트 수.
average	double 또는 null	null	현재 시간 간격 동안 해당 변수 값의 평균.
min	double 또는 null	null	현재 시간 간격 동안 해당 변수 값의 최소 값.
max	boolean 또는 null	null	현재 시간 간격 동안 해당 변수 값의 최대 값.
sum	string 또는 null	null	현재 시간 간격 동안 해당 변수 값의 합계.
recordVersion	long 또는 null	null	기록을 위한 버전 번호. 버전 번호를 사용하여 최신 레코드를 선택할 수 있습니다. 최신 레코드는 버전 번호가 더 큼니다.

Example 콜드 티어의 지표 데이터

```
{
  "seriesId": "f74c2828-5317-4df3-ba16-6d41b5bcb531",
  "timeInSeconds": 1637334060,
  "offsetInNanos": 0,
  "quality": "GOOD",
  "resolution": "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": "16.0",
 "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": "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": "16.0",
 "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": "16.0",
 "min": {
   "double": 1.0
 },
 "max": {
   "double": 31.0
 },
 "sum": {
   "double": 496.0
 },
 "recordVersion": null
}
```

자산 메타데이터

AWS IoT SiteWise 가 데이터를 콜드 티어로 처음 내보내도록 활성화하면 자산 메타데이터가 콜드 티어로 내보내집니다. 초기 구성 후는 자산 모델 정의 또는 자산 정의를 변경할 때만 자산 메타데이터를 계층으로 내 AWS IoT SiteWise 보냅니다. 자산 메타데이터는 콜드 티어에 새 줄 구분 JSON(.ndjson) 형식으로 저장됩니다.

파일 경로

AWS IoT SiteWise 는 다음 템플릿을 사용하여 콜드 티어에 자산 메타데이터를 저장합니다.

```
{keyPrefix}/asset_metadata/asset_{assetId}.ndjson
```

콜드 티어의 자산 메타데이터에 대한 모든 파일 경로에는 다음 구성 요소가 포함됩니다.

경로 구성 요소	설명
keyPrefix	스토리지 구성에서 지정한 Amazon S3 접두사 AWS IoT SiteWise입니다. Amazon S3에서는 접두사를 버킷의 폴더 이름으로 사용합니다.
asset_metadata	자산 메타데이터를 저장하는 폴더. asset_metadata 폴더는 접두사 폴더에 저장됩니다.
fileName	파일 이름은 밑줄 (_)문자를 구분 기호로 사용하여 다음을 구분합니다. - asset 접두사입니다. - assetId 값입니다. 파일은 .ndjson 형식으로 저장됩니다.

Example 더 콜드 티어의 자산 메타데이터에 대한 파일 경로

keyPrefix/asset_metadata/asset_35901915-d476-4dca-8637-d9ed4df939ed.ndjson

필드

콜드 티어로 내보내는 자산 메타데이터의 스키마에는 다음 필드가 포함됩니다.

필드 이름	설명
assetId	자산의 ID입니다.
assetName	자산 이름.
assetExternalId	자산의 외부 ID입니다.
assetModelId	자산을 생성할 때 사용된 자산 모델의 ID입니다.
assetModelName	자산 모델의 이름입니다.
assetModelExternalId	자산의 외부 ID입니다.

필드 이름	설명
<code>assetPropertyId</code>	자산 속성의 ID입니다.
<code>assetPropertyName</code>	자산 특성의 이름입니다.
<code>assetPropertyExternalId</code>	자산 속성의 ID입니다.
<code>assetPropertyDataType</code>	자산 속성의 데이터 유형입니다.
<code>assetPropertyUnit</code>	자산 속성의 단위(예: Newtons 및 RPM).
<code>assetPropertyAlias</code>	OPC UA 서버 데이터 스트림 경로(예: /company/windfarm/3/turbine/7/temperature)와 같이 자산 속성을 식별하는 별칭입니다.
<code>timeSeriesId</code>	장비, 지표 또는 변환의 시계열 데이터를 식별하는 ID. 이 필드를 사용하여 쿼리에서 원시 데이터와 자산 메타데이터를 결합할 수 있습니다.
<code>timeSeriesBucket</code>	00~ff 사이의 16진수입니다. 이 숫자는 <code>timeSeriesId</code> 에서 파생되었습니다. 이 파티션은가 콜드 티어에 AWS IoT SiteWise 쓸 때 처리량을 높이는 데 사용됩니다. Amazon Athena 를 사용하여 쿼리를 실행하면 세분화된 분할에 파티션을 사용하여 쿼리 성능을 개선할 수 있습니다. 원시 데이터에 대한 파일 경로의 <code>timeSeriesBucket</code> 와 <code>seriesBucket</code> 는 같은 숫자입니다.
<code>assetCompositeModelId</code>	복합 모델의 ID입니다.
<code>assetCompositeModelExternalId</code>	복합 모델의 외부 ID입니다.
<code>assetCompositeModelDescription</code>	복합 모델에 대한 설명입니다.
<code>assetCompositeModelName</code>	복합 모델의 이름입니다.

필드 이름	설명
assetCompositeModelType	복합 모델의 유형입니다. 경보 복합 모델의 경우 이 유형은 AWS/ALARM 입니다.
assetCreationDate	UNIX epoch 시간 형식으로 표시된 자산이 생성된 날짜.
assetLastUpdateDate	Unix Epoch 시간으로 표시된 자산이 마지막으로 업데이트된 날짜.
assetStatusErrorCode	오류 코드입니다.
assetStatusErrorMessage	오류 메시지입니다.
assetStatusState	자산의 현재 상태입니다.

Example 콜드 티어의 자산 메타데이터

```
{
  "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",
  "assetPropertyDescription": "Temperature in Washington/Seattle/WT2/temp",
  "timeSeriesId": "7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-bc6f-1b490154b07a",
  "timeSeriesBucket": "f6",
  "assetArn": null,
  "assetCompositeModelDescription": null
},
{
  "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,
  "assetPropertyName": "Pressure",
  "assetPropertyDescription": "Pressure in Washington/Seattle/WT2/pressure",
  "timeSeriesId": "7020c8e2-e6db-40fa-9845-ed0dddd4c77d_c706d54d-4c11-42dc-9a01-63662fc697b4",
  "timeSeriesBucket": "1e",
  "assetArn": null,
  "assetCompositeModelDescription": null
},
{
  "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-ed0ddd4c77d_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
```

자산 계층 구조 메타데이터

AWS IoT SiteWise 가 처음으로 콜드 티어에 데이터를 저장하도록 활성화하면 자산 계층 메타데이터가 콜드 티어로 내보내집니다. 초기 구성 후는 자산 모델 또는 자산 정의를 변경할 때만 자산 계층 구조 메타데이터를 콜드 티어로 내 AWS IoT SiteWise 보냅니다. 자산 계층 구조 메타데이터는 콜드 티어에 새 줄 구분 JSON(.ndjson) 형식으로 저장됩니다.

계층 구조, 대상 자산 또는 소스 자산의 외부 식별자는 [DescribeAsset](#) API를 직접적으로 호출하여 검색됩니다.

파일 경로

AWS IoT SiteWise 는 다음 템플릿을 사용하여 자산 계층 구조 메타데이터를 콜드 티어에 저장합니다.

```
{keyPrefix}/asset_hierarchy_metadata/{parentAssetId}_{hierarchyId}.ndjson
```

콜드 티어의 자산 계층 구조 메타데이터에 대한 모든 파일 경로에는 다음 구성 요소가 포함됩니다.

경로 구성 요소	설명
keyPrefix	AWS IoT SiteWise 스토리지 구성에서 지정한 Amazon S3 접두사입니다. Amazon S3에서는 접두사를 버킷의 폴더 이름으로 사용합니다.

경로 구성 요소	설명
asset_hierarchy_metadata	자산 계층 구조 메타데이터를 저장하는 폴더. asset_hierarchy_metadata 폴더는 접두사 폴더에 저장됩니다.
fileName	파일 이름은 밑줄 (_)문자를 구분 기호로 사용하여 다음을 구분합니다. - parentAssetId 값입니다. - hierarchyId 값입니다. 파일은 .ndjson 형식으로 저장됩니다.

Example 콜드 티어의 자산 계층 구조 메타데이터에 대한 파일 경로

keyPrefix/asset_hierarchy_metadata/35901915-d476-4dca-8637-d9ed4df939ed_c5b3ced8-589a-48c7-9998-cdcccfc9747a0.ndjson

필드

콜드 티어로 내보내는 자산 계층 구조 메타데이터의 스키마에는 다음 필드가 포함됩니다.

필드 이름	설명
sourceAssetId	이 자산 관계에 있는 소스 자산의 ID.
targetAssetId	이 자산 관계에 있는 대상 자산의 ID.
hierarchyId	해당 계층 구조의 ID.
associationType	이 자산 관계의 연결 유형. 값은 CHILD여야 합니다. 대상 자산은 소스 자산의 하위 자산입니다.

Example 콜드 티어의 자산 계층 구조 메타데이터

```
{
  "sourceAssetId": "80388e72-2284-44fb-9c89-bfbaf0dfedd2",
  "targetAssetId": "2b866c25-0c74-4750-bdf5-b73683c8a2a2",
  "hierarchyId": "bbcd9f59-0412-4585-a61d-6044db526aee",
  "associationType": "CHILD"
}
{
  "sourceAssetId": "80388e72-2284-44fb-9c89-bfbaf0dfedd2",
  "targetAssetId": "6b51246e-984d-460d-bc0b-470ea47d1e31",
  "hierarchyId": "bbcd9f59-0412-4585-a61d-6044db526aee",
  "associationType": "CHILD"
}
```

콘솔 티어에서 데이터를 보려면

1. [Amazon S3 콘솔](#)로 이동합니다.
2. 탐색 패널에서, 버킷을 선택하고 Amazon S3 버킷을 선택합니다.
3. 원시 데이터, 자산 메타데이터 또는 자산 계층 구조 메타데이터가 포함된 폴더로 이동합니다.
4. 파일을 선택한 다음 작업에서 다운로드를 선택합니다.

스토리지 데이터 인덱스 파일

AWS IoT SiteWise 는 이러한 파일을 사용하여 데이터 쿼리 성능을 최적화합니다. Amazon S3 버킷에 표시되지만 사용할 필요는 없습니다.

파일 경로

AWS IoT SiteWise 는 다음 템플릿을 사용하여 데이터 인덱스 파일을 콜드 티어에 저장합니다.

```
keyPrefix/index/series=timeseriesId/startYear=startYear/startMonth=startMonth/startDay=startDay/index_timeseriesId_startTimestamp_quality
```

Example 데이터 스토리지 인덱스 파일의 파일 경로

```
keyPrefix/index/series=7020c8e2-e6db-40fa-9845-ed0ddddd4c77d_95e63da7-d34e-43e1-bc6f-1b490154b07a/startYear=2022/startMonth=02/startDay=03/index_7020c8e2-e6db-40fa-9845-ed0ddddd4c77d_95e63da7-d34e-43e1-bc6f-1b490154b07a_1643846400_G00D
```

AWS SDKs AWS IoT SiteWise 사용에 대한 코드 예제

다음 코드 예제에서는 AWS 소프트웨어 개발 키트(SDK)와 AWS IoT SiteWise 함께를 사용하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 관련 시나리오의 컨텍스트에 따라 표시되며, 개별 서비스 함수를 직접적으로 호출하는 방법을 보여줍니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서 이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작

안녕하세요 AWS IoT SiteWise

다음 코드 예제에서는 AWS IoT SiteWise의 사용을 시작하는 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
}
}

```

- API 세부 정보는 API 참조의 [ListAssetModels](#) AWS SDK for Java 2.x 를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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();
}
```

- API 세부 정보는 API 참조의 [ListAssetModels](#) AWS SDK for JavaScript 를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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"))
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [ListAssetModels](#)를 참조하세요. AWS

코드 예제

- [AWS SDKs AWS IoT SiteWise 사용에 대한 기본 예제](#)
 - [안녕하세요 AWS IoT SiteWise](#)
 - [AWS SDK를 AWS IoT SiteWise 사용하여의 기본 사항 알아보기](#)
 - [AWS SDKs AWS IoT SiteWise 를 사용하기 위한 작업](#)
 - [AWS SDK 또는 CLI와 BatchPutAssetPropertyValue 함께 사용](#)
 - [AWS SDK 또는 CLI와 CreateAsset 함께 사용](#)
 - [AWS SDK 또는 CLI와 CreateAssetModel 함께 사용](#)

- [AWS SDK 또는 CLI와 CreateGateway 함께 사용](#)
- [AWS SDK 또는 CLI와 CreatePortal 함께 사용](#)
- [AWS SDK 또는 CLI와 DeleteAsset 함께 사용](#)
- [AWS SDK 또는 CLI와 DeleteAssetModel 함께 사용](#)
- [AWS SDK 또는 CLI와 DeleteGateway 함께 사용](#)
- [AWS SDK 또는 CLI와 DeletePortal 함께 사용](#)
- [AWS SDK 또는 CLI와 DescribeAssetModel 함께 사용](#)
- [AWS SDK 또는 CLI와 DescribeGateway 함께 사용](#)
- [AWS SDK 또는 CLI와 DescribePortal 함께 사용](#)
- [AWS SDK 또는 CLI와 GetAssetPropertyValue 함께 사용](#)
- [AWS SDK 또는 CLI와 ListAssetModels 함께 사용](#)

AWS SDKs AWS IoT SiteWise 사용에 대한 기본 예제

다음 코드 예제에서는 AWS IoT SiteWise SDKs에서의 기본 사항을 AWS 사용하는 방법을 보여줍니다.

예시

- [안녕하세요 AWS IoT SiteWise](#)
- [AWS SDK를 AWS IoT SiteWise 사용하여의 기본 사항 알아보기](#)
- [AWS SDKs AWS IoT SiteWise 를 사용하기 위한 작업](#)
 - [AWS SDK 또는 CLI와 BatchPutAssetPropertyValue 함께 사용](#)
 - [AWS SDK 또는 CLI와 CreateAsset 함께 사용](#)
 - [AWS SDK 또는 CLI와 CreateAssetModel 함께 사용](#)
 - [AWS SDK 또는 CLI와 CreateGateway 함께 사용](#)
 - [AWS SDK 또는 CLI와 CreatePortal 함께 사용](#)
 - [AWS SDK 또는 CLI와 DeleteAsset 함께 사용](#)
 - [AWS SDK 또는 CLI와 DeleteAssetModel 함께 사용](#)
 - [AWS SDK 또는 CLI와 DeleteGateway 함께 사용](#)
 - [AWS SDK 또는 CLI와 DeletePortal 함께 사용](#)

- [AWS SDK 또는 CLI와 DescribeGateway 함께 사용](#)
- [AWS SDK 또는 CLI와 DescribePortal 함께 사용](#)
- [AWS SDK 또는 CLI와 GetAssetPropertyValue 함께 사용](#)
- [AWS SDK 또는 CLI와 ListAssetModels 함께 사용](#)

안녕하세요 AWS IoT SiteWise

다음 코드 예제에서는 AWS IoT SiteWise의 사용을 시작하는 방법을 보여 줍니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
}
}

```

- API 세부 정보는 API 참조의 [ListAssetModels](#) AWS SDK for Java 2.x 를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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();
}

```

- API 세부 정보는 API 참조의 [ListAssetModels](#) AWS SDK for JavaScript 를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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"))

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [ListAssetModels](#)를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 AWS IoT SiteWise 사용하여의 기본 사항 알아보기

다음 코드 예제는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- AWS IoT SiteWise 자산 모델을 생성합니다.
- AWS IoT SiteWise 자산을 생성합니다.
- 속성 ID 값을 검색합니다.
- AWS IoT SiteWise 애셋으로 데이터를 전송합니다.
- AWS IoT SiteWise Asset 속성의 값을 검색합니다.
- AWS IoT SiteWise 포털을 생성합니다.
- AWS IoT SiteWise 게이트웨이를 생성합니다.
- AWS IoT SiteWise 게이트웨이를 설명합니다.
- AWS IoT SiteWise 자산을 삭제합니다.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS IoT SiteWise 기능을 보여주는 대화형 시나리오를 실행합니다.

```
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!");
    }
}

```

AWS IoT SiteWise SDK 메서드의 래퍼 클래스입니다.

```

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
 *      property ID. The calling code can attach callbacks, then handle the
result or exception by calling
 *      {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
By calling
 *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
    ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
    return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
        .handle((response, throwable) -> {
            if (response != null) {
                return response.assetModelPropertySummaries().stream()
                    .collect(Collectors
                        .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
            } else {
                logger.error("Error occurred while fetching property IDs:
{}.", throwable.getCause().getMessage());
                throw (CompletionException) throwable;
            }
        });
}
}

```

```

/**
 * Deletes an asset.
 *
 * @param assetId the ID of the asset to be deleted.
 * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
 *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
 *         {@link CompletableFuture#get()}.
 *         <p>
 *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *         it available to the calling code as a {@link CompletionException}.
By calling
 *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String
assetId) {
    DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
        .assetId(assetId)
        .build();

    return getAsyncClient().deleteAsset(deleteAssetRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("An error occurred deleting asset with id: {}",
assetId);
            }
        });
}

/**
 * Deletes an Asset Model with the specified ID.
 *
 * @param assetModelId the ID of the Asset Model to delete.
 * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
 *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
 *         {@link CompletableFuture#get()}.
 *         <p>

```

```

        *           If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
        *           it available to the calling code as a {@link CompletionException}.
By calling
        *           {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<DeleteAssetModelResponse>
deleteAssetModelAsync(String assetModelId) {
        DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
            .assetModelId(assetModelId)
            .build();

        return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete asset model with ID:{}",
exception.getMessage());
                }
            });
    }

/**
 * Creates a new IoT SiteWise portal.
 *
 * @param portalName the name of the portal to create.
 * @param iamRole the IAM role ARN to use for the portal.
 * @param contactEmail the email address of the portal contact.
 * @return a {@link CompletableFuture} that represents a {@link String}
result of the portal ID. The calling code
 * can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
 * {@link CompletableFuture#get()}.
 * <p>
 * If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 * it available to the calling code as a {@link CompletionException}.
By calling
 * {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<String> createPortalAsync(String portalName, String
iamRole, String contactEmail) {

```

```

        CreatePortalRequest createPortalRequest = CreatePortalRequest.builder()
            .portalName(portalName)
            .portalDescription("This is my custom IoT SiteWise portal.")
            .portalContactEmail(contactEmail)
            .roleArn(iamRole)
            .build();

        return getAsyncClient().createPortal(createPortalRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create portal: {} ",
exception.getCause().getMessage());
                    throw (CompletionException) exception;
                }
                return response.portalId();
            });
    }

    /**
     * Deletes a portal.
     *
     * @param portalId the ID of the portal to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeletePortalResponse}. The calling code can attach
     *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *
     * <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<DeletePortalResponse> deletePortalAsync(String
portalId) {
        DeletePortalRequest deletePortalRequest = DeletePortalRequest.builder()
            .portalId(portalId)
            .build();

        return getAsyncClient().deletePortal(deletePortalRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {

```

```

        logger.error("Failed to delete portal with ID: {}. Error:
        {}", portalId, exception.getCause().getMessage());
    }
    });
}

/**
 * Retrieves the asset model ID for the given asset model name.
 *
 * @param assetModelName the name of the asset model for the ID.
 * @return a {@link CompletableFuture} that represents a {@link String}
result of the asset model ID or null if the
 *      asset model cannot be found. The calling code can attach
callbacks, then handle the result or exception
 *      by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
By calling
 *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<String> getAssetModelIdAsync(String assetModelName)
{
    ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
    return getAsyncClient().listAssetModels(listAssetModelsRequest)
        .handle((listAssetModelsResponse, exception) -> {
            if (exception != null) {
                logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                throw (CompletionException) exception;
            }
            for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                if (assetModelSummary.name().equals(assetModelName)) {
                    return assetModelSummary.id();
                }
            }
            return null;
        });
}

```

```

/**
 * Retrieves a portal's description.
 *
 * @param portalId the ID of the portal to describe.
 * @return a {@link CompletableFuture} that represents a {@link String}
result of the portal's start URL
 *      (see: {@link DescribePortalResponse#portalStartUrl()}). The
calling code can attach callbacks, then handle the
 *      result or exception by calling {@link CompletableFuture#join()} or
{@link CompletableFuture#get()}.
 *      <p>
 *      If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
 *      it available to the calling code as a {@link CompletionException}.
By calling
 *      {@link CompletionException#getCause()}, the calling code can
access the original exception.
 */
public CompletableFuture<String> describePortalAsync(String portalId) {
    DescribePortalRequest request = DescribePortalRequest.builder()
        .portalId(portalId)
        .build();

    return getAsyncClient().describePortal(request)
        .handle((response, exception) -> {
            if (exception != null) {
                logger.error("An exception occurred retrieving the portal
description: {}", exception.getCause().getMessage());
                throw (CompletionException) exception;
            }
            return response.portalStartUrl();
        });
}

/**
 * Creates a new IoT SiteWise gateway.
 *
 * @param gatewayName The name of the gateway to create.
 * @param myThing      The name of the core device thing to associate with the
gateway.
 * @return a {@link CompletableFuture} that represents a {@link String}
result of the gateways ID. The calling code

```

```

    *      can attach callbacks, then handle the result or exception by
    calling {@link CompletableFuture#join()} or
    *      {@link CompletableFuture#get()}.
    *      <p>
    *      If any completion stage in this method throws an exception, the
    method logs the exception cause and keeps
    *      it available to the calling code as a {@link CompletionException}.
    By calling
    *      {@link CompletionException#getCause()}, the calling code can
    access the original exception.
    */
    public CompletableFuture<String> createGatewayAsync(String gatewayName,
String myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest =
        CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }

    /**

```



```

    * Deletes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
    public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
        DeleteGatewayRequest deleteGatewayRequest =
DeleteGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().deleteGateway(deleteGatewayRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
    * Describes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to describe.
    * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps

```

```

        *           it available to the calling code as a {@link CompletionException}.
By calling
        *           {@link CompletionException#getCause()}, the calling code can
access the original exception.
        */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
                }
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}

```

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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,
            },
          },
        }),
      );
    } catch (caught) {
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

```

```

        },
        },
    )),
);
console.log(
    `Gateway creation completed successfully. ID is
    ${createGatewayResponse.gatewayId}`,
);
state.gateway.gatewayId = createGatewayResponse.gatewayId;
} catch (caught) {
    if (caught.name === "IoTSiteWiseException") {
        console.log(
            `There was a problem creating the gateway: ${caught.message}.`,
        );
        throw caught;
    }
    console.error(`${caught.message}`);
    throw caught;
}
},
);

const displayDescribeIoTSiteWiseGateway = new ScenarioOutput(
    "displayDescribeIoTSiteWiseGateway",
    "9. Describe the IoT SiteWise Gateway",
);

const sdkDescribeIoTSiteWiseGateway = new ScenarioAction(
    "sdkDescribeIoTSiteWiseGateway",
    async (/** @type {State} */ state) => {
        try {
            const describeGatewayResponse = await state.iotSiteWiseClient.send(
                new DescribeGatewayCommand({
                    gatewayId: state.gateway.gatewayId,
                }),
            );
            console.log("Gateway creation completed successfully.");
            console.log(`Gateway Name: ${describeGatewayResponse.gatewayName}`);
            console.log(`Gateway ARN: ${describeGatewayResponse.gatewayArn}`);
            console.log(
                `Gateway Platform:
                ${Object.keys(describeGatewayResponse.gatewayPlatform)}`,
            );
            console.log(

```

```

        `Gateway Creation Date: ${describeGatewayResponse.creationDate}`,
    );
} catch (caught) {
    if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
        throw caught;
    }
    console.error(`${caught.message}`);
    throw caught;
}
},
);

const askToDeleteResources = new ScenarioInput(
    "askToDeleteResources",
    `10. Delete the AWS IoT SiteWise Assets

Before you can delete the Asset Model, you must delete the assets.`
    , { type: "confirm" }
);

const displayConfirmDeleteResources = new ScenarioAction(
    "displayConfirmDeleteResources",
    async (** @type {State} */ state) => {
        if (state.askToDeleteResources) {
            return "You selected to delete the SiteWise assets.";
        }
        return "The resources will not be deleted. Please delete them manually to avoid charges.";
    },
);

const sdkDeleteResources = new ScenarioAction(
    "sdkDeleteResources",
    async (** @type {State} */ state) => {
        await wait(10); // Give the portal status time to catch up.
        try {
            await state.iotSiteWiseClient.send(
                new DeletePortalCommand({
                    portalId: state.portal.portalId,
                })
            );
            console.log(
                `Portal ${state.portal.portalName} was deleted successfully.`
            );

```

```
    );
  } catch (caught) {
    if (caught.name === "ResourceNotFoundException") {
      console.log(`The Portal ${state.portal.portalName} was not found.`);
    } else {
      console.log(`When trying to delete the portal: ${caught.message}`);
    }
  }
}

try {
  await state.iotSiteWiseClient.send(
    new DeleteGatewayCommand({
      gatewayId: state.gateway.gatewayId,
    }),
  );
  console.log(
    `Gateway ${state.gateway.gatewayName} was deleted successfully.`
  );
} catch (caught) {
  if (caught.name === "ResourceNotFoundException") {
    console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
  } else {
    console.log(`When trying to delete the gateway: ${caught.message}`);
  }
}

try {
  await state.iotSiteWiseClient.send(
    new DeleteAssetCommand({
      assetId: state.asset.assetId,
    }),
  );
  await wait(5); // Allow the delete to finish.
  console.log(`Asset ${state.asset.assetName} was deleted successfully.`);
} catch (caught) {
  if (caught.name === "ResourceNotFoundException") {
    console.log(`The Asset ${state.asset.assetName} was not found.`);
  } else {
    console.log(`When deleting the asset: ${caught.message}`);
  }
}

await wait(30); // Allow asset deletion to finish.
try {
```



```

    await state.iotSiteWiseClient.send(
      new DeleteAssetModelCommand({
        assetModelId: state.assetModel.assetModelId,
      }),
    );
    console.log(
      `Asset Model ${state.assetModel.assetModelName} was deleted
successfully.`,
    );
  } catch (caught) {
    if (caught.name === "ResourceNotFoundException") {
      console.log(
        `The Asset Model ${state.assetModel.assetModelName} was not found.`,
      );
    } else {
      console.log(`When deleting the asset model: ${caught.message}`);
    }
  }

  try {
    await state.cloudFormationClient.send(
      new DeleteStackCommand({
        StackName: stackName,
      }),
    );
    await waitUntilStackDeleteComplete(
      { client: state.cloudFormationClient },
      { StackName: stackName },
    );
    console.log("The stack was deleted successfully.");
  } catch (caught) {
    console.log(
      `${caught.message}. The stack was NOT deleted. Please clean up the
resources manually.`,
    );
  }
},
{ skipWhen: (/** @type {} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the IoT Sitewise Basics scenario for the AWS Javascript SDK v3.
Thank you!",

```

```
);

const myScenario = new Scenario(
  "IoTSiteWise Basics",
  [
    greet,
    pressEnter,
    displayBuildCloudFormationStack,
    sdkBuildCloudFormationStack,
    pressEnter,
    displayCreateAWSSiteWiseAssetModel,
    sdkCreateAWSSiteWiseAssetModel,
    displayCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    waitThirtySeconds,
    sdkCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    displayRetrievePropertyId,
    sdkRetrievePropertyId,
    pressEnter,
    displaySendDataToIoTSiteWiseAsset,
    sdkSendDataToIoTSiteWiseAsset,
    pressEnter,
    displayRetrieveValueOfIoTSiteWiseAsset,
    sdkRetrieveValueOfIoTSiteWiseAsset,
    pressEnter,
    displayCreateIoTSiteWisePortal,
    sdkCreateIoTSiteWisePortal,
    pressEnter,
    displayDescribePortal,
    sdkDescribePortal,
    pressEnter,
    displayCreateIoTSiteWiseGateway,
    sdkCreateIoTSiteWiseGateway,
    pressEnter,
    displayDescribeIoTSiteWiseGateway,
    sdkDescribeIoTSiteWiseGateway,
    pressEnter,
    askToDeleteResources,
    displayConfirmDeleteResources,
    sdkDeleteResources,
    goodbye,
  ],
  {
```

```

    iotSiteWiseClient: new IoTSiteWiseClient({}),
    cloudFormationClient: new CloudFormationClient({}),
    asset: { assetName: "MyAsset1" },
    assetModel: { assetModelName: "MyAssetModel1" },
    portal: { portalName: "MyPortal1" },
    gateway: { gatewayName: "MyGateway1" },
    propertyIds: [],
    contactEmail: "user@mydomain.com",
    thing: "MyThing1",
    sampleData: { temperature: 23.5, humidity: 65.0 },
  },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
  });
  main({ confirmAll: values.yes });
}

```

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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

```

AWS IoT SiteWise 작업을 래핑하는 IoTSitewiseWrapper 클래스입니다.

```

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

```

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 [섹션을 참조하세요](#) [AWS SDK에서 이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDKs AWS IoT SiteWise 를 사용하기 위한 작업

다음 코드 예제에서는 AWS SDKs를 사용하여 개별 AWS IoT SiteWise 작업을 수행하는 방법을 보여줍니다. 각 예제에는 GitHub에 대한 링크가 포함되어 있습니다. 여기에서 코드 설정 및 실행에 대한 지침을 찾을 수 있습니다.

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 [AWS IoT SiteWise API 참조](#)를 참조하세요.

예시

- [AWS SDK 또는 CLI와 BatchPutAssetPropertyValue 함께 사용](#)
- [AWS SDK 또는 CLI와 CreateAsset 함께 사용](#)
- [AWS SDK 또는 CLI와 CreateAssetModel 함께 사용](#)
- [AWS SDK 또는 CLI와 CreateGateway 함께 사용](#)
- [AWS SDK 또는 CLI와 CreatePortal 함께 사용](#)
- [AWS SDK 또는 CLI와 DeleteAsset 함께 사용](#)
- [AWS SDK 또는 CLI와 DeleteAssetModel 함께 사용](#)
- [AWS SDK 또는 CLI와 DeleteGateway 함께 사용](#)
- [AWS SDK 또는 CLI와 DeletePortal 함께 사용](#)
- [AWS SDK 또는 CLI와 DescribeAssetModel 함께 사용](#)
- [AWS SDK 또는 CLI와 DescribeGateway 함께 사용](#)
- [AWS SDK 또는 CLI와 DescribePortal 함께 사용](#)
- [AWS SDK 또는 CLI와 GetAssetPropertyValue 함께 사용](#)
- [AWS SDK 또는 CLI와 ListAssetModels 함께 사용](#)

AWS SDK 또는 CLI와 **BatchPutAssetPropertyValue** 함께 사용

다음 코드 예시는 BatchPutAssetPropertyValue의 사용 방법을 보여 줍니다.

CLI

AWS CLI

자산 속성으로 데이터 보내기

다음 batch-put-asset-property-value 예시에서는 속성 별칭으로 식별된 자산 속성으로 전력 및 온도 데이터를 전송합니다.

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

batch-put-asset-property-value.json의 콘텐츠:

```
{
  "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
    }
  }
]
}

```

출력:

```

{
  "errorEntries": []
}

```

자세한 내용은 [AWS IoT SiteWise 사용 설명서의 IoT SiteWise API를 사용하여 데이터 수집](#)을 참조하세요. AWS IoT SiteWise

- API 세부 정보는 AWS CLI 명령 참조의 [BatchPutAssetPropertyValue](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/**
 * 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();

    return getAsyncClient().batchPutAssetPropertyValue(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("An exception occurred: {}",
                    exception.getCause().getMessage());
            }
        });
}

```

- API 세부 정보는 API 참조의 [BatchPutAssetPropertyValue](#) AWS SDK for Java 2.x 를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
  }
};

```

- API 세부 정보는 API 참조의 [BatchPutAssetPropertyValue](#) AWS SDK for JavaScript 를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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

```

값 목록에서 항목 파라미터를 생성하는 헬퍼 함수입니다.

```

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

```

다음은 헬퍼 함수에 전달할 값 목록의 예입니다.

```

values = [
    {
        "propertyId": humidity_property_id,
        "valueType": "doubleValue",
        "value": 65.0,
    },
    {
        "propertyId": temperature_property_id,
        "valueType": "doubleValue",
        "value": 23.5,
    },
]

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [BatchPutAssetPropertyValue](#)를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **CreateAsset** 함께 사용

다음 코드 예시는 CreateAsset의 사용 방법을 보여 줍니다.

CLI

AWS CLI

자산 생성

다음 create-asset 예시에서는 풍력 터빈 자산 모델에서 풍력 터빈 자산을 생성합니다.

```
aws iotsitewise create-asset \  
  --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \  
  --asset-name "Wind Turbine 1"
```

출력:

```
{  
  "assetId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",  
  "assetArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/  
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",  
  "assetStatus": {  
    "state": "CREATING"  
  }  
}
```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Creating assets](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [CreateAsset](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**  
 * 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());
                }
            });
    }
}

```

- API 세부 정보는 API 참조의 [CreateAsset](#) AWS SDK for Java 2.x 을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
  }
};
```

- API 세부 정보는 API 참조의 [CreateAsset](#) AWS SDK for JavaScript 을 참조하세요.

Python

SDK for Python (Boto3)

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [CreateAsset](#)을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **CreateAssetModel** 함께 사용

다음 코드 예시는 CreateAssetModel의 사용 방법을 보여 줍니다.

CLI

AWS CLI

자산 모델 생성

다음 create-asset-model 예시에서는 다음 속성을 가진 풍력 터빈을 정의하는 자산 모델을 생성합니다.

일련 번호 - 풍력 터빈의 일련 번호발전 전력 - 풍력 터빈의 발전 전력 데이터 스트림온도 C - 풍력 터빈의 온도 데이터 스트림(섭씨 단위)온도 F - 섭씨에서 화씨로 매핑된 온도 데이터 포인트

```
aws iotsitewise create-asset-model \
  --cli-input-json file://create-wind-turbine-model.json
```

create-wind-turbine-model.json의 콘텐츠:

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

```

출력:

```

{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelStatus": {

```

```

    "state": "CREATING"
  }
}

```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Defining asset models](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [CreateAssetModel](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/**
 * 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());
        }
    });
}
```

- API 세부 정보는 API 참조의 [CreateAssetModel](#) AWS SDK for Java 2.x 을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
  }
};
```

- API 세부 정보는 API 참조의 [CreateAssetModel](#) AWS SDK for JavaScript 을 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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

```

다음은 함수에 전달할 속성 목록의 예입니다.

```

properties = [
    {
        "name": temperature_property_name,
        "dataType": "DOUBLE",
        "type": {
            "measurement": {},
        },
    },
    {

```

```

        "name": humidity_property_name,
        "dataType": "DOUBLE",
        "type": {
            "measurement": {},
        },
    },
]

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [CreateAssetModel](#)을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **CreateGateway** 함께 사용

다음 코드 예시는 CreateGateway의 사용 방법을 보여 줍니다.

CLI

AWS CLI

게이트웨이 생성

다음 create-gateway 예제에서는 AWS IoT Greengrass에서 실행되는 게이트웨이를 생성합니다.

```

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}

```

출력:

```

{
  "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
  "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
}

```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Configuring a gateway](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [CreateGateway](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * 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();
            });
    }
}

```

- API 세부 정보는 API 참조의 [CreateGateway](#) AWS SDK for Java 2.x 를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
  }
};

```

- API 세부 정보는 API 참조의 [CreateGateway](#) AWS SDK for JavaScript 를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [CreateGateway](#)를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **CreatePortal** 함께 사용

다음 코드 예시는 CreatePortal의 사용 방법을 보여 줍니다.

CLI

AWS CLI

포털 생성

다음 create-portal 예시에서는 풍력 발전 단지 회사의 웹 포털을 생성합니다. AWS Single Sign-On을 활성화한 리전과 동일한 리전에서만 포털을 생성할 수 있습니다.

```

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

```

출력:

```

{
  "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "portalArn": "arn:aws:iot: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"
}

```

자세한 내용은 [AWS IoT SiteWise 사용 설명서의 IoT SiteWise Monitor 시작하기](#) AWS 및 IoT SiteWise AWS IoT SiteWise 사용 설명서의 [AWS SSO 활성화](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [CreatePortal](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/**
 * 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();
        });
}

```

- API 세부 정보는 API 참조의 [CreatePortal](#) AWS SDK for Java 2.x 을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
  }
};

```

- API 세부 정보는 API 참조의 [CreatePortal](#) AWS SDK for JavaScript 을 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [CreatePortal](#)을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeleteAsset** 함께 사용

다음 코드 예시는 DeleteAsset의 사용 방법을 보여 줍니다.

CLI

AWS CLI

자산 삭제

다음 delete-asset 예시에서는 풍력 터빈 자산을 삭제합니다.

```

aws iotsitewise delete-asset \
  --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE

```

출력:

```

{
  "assetStatus": {
    "state": "DELETING"
  }
}

```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Deleting assets](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteAsset](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * 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);
            }
        });
}
```



```

        }
    });
}

```

- API 세부 정보는 API 참조의 [DeleteAsset](#) AWS SDK for Java 2.x 을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
}
};

```

- API 세부 정보는 API 참조의 [DeleteAsset](#) AWS SDK for JavaScript 을 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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

```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 [DeleteAsset](#)을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 [섹션을 참조하세요](#) [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeleteAssetModel** 함께 사용

다음 코드 예시는 DeleteAssetModel의 사용 방법을 보여 줍니다.

CLI

AWS CLI

자산 모델 삭제

다음 delete-asset-model 예시에서는 풍력 터빈 자산 모델을 삭제합니다.

```

aws iotsitewise delete-asset-model \
  --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE

```

출력:

```
{
  "assetModelStatus": {
    "state": "DELETING"
  }
}
```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Deleting asset models](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteAssetModel](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * 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());
        }
    });
}

```

- API 세부 정보는 API 참조의 [DeleteAssetModel](#) AWS SDK for Java 2.x 을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
}
};

```

- API 세부 정보는 API 참조의 [DeleteAssetModel](#) AWS SDK for JavaScript 을 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [DeleteAssetModel](#)을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 [섹션을 참조하세요](#) [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeleteGateway** 함께 사용

다음 코드 예시는 DeleteGateway의 사용 방법을 보여 줍니다.

CLI

AWS CLI

게이트웨이 삭제

다음 delete-gateway 예시에서는 게이트웨이를 삭제합니다.

```
aws iotsitewise delete-gateway \
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

이 명령은 출력을 생성하지 않습니다.

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Ingesting data using a gateway](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [DeleteGateway](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * 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());
                }
            });
    }
}

```

- API 세부 정보는 API 참조의 [DeleteGateway](#) AWS SDK for Java 2.x 를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
  }
};
```

- API 세부 정보는 API 참조의 [DeleteGateway](#) AWS SDK for JavaScript 를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [DeleteGateway](#)를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 [섹션을 참조하세요](#)[AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DeletePortal** 함께 사용

다음 코드 예시는 DeletePortal의 사용 방법을 보여 줍니다.

CLI

AWS CLI

포털 삭제

다음 delete-portal 예시에서는 풍력 발전 단지 회사의 웹 포털을 삭제합니다.

```
aws iotsitewise delete-portal \
  --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

출력:

```
{
  "portalStatus": {
    "state": "DELETING"
  }
}
```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Deleting a portal](#)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [DeletePortal](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
```

```

    * 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());
                }
            });
    }
}

```

- API 세부 정보는 API 참조의 [DeletePortal](#) AWS SDK for Java 2.x 을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
  }
};
```

- API 세부 정보는 API 참조의 [DeletePortal](#) AWS SDK for JavaScript 을 참조하세요.

Python

SDK for Python (Boto3)

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [DeletePortal](#)을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeAssetModel** 함께 사용

다음 코드 예시는 DescribeAssetModel의 사용 방법을 보여 줍니다.

CLI

AWS CLI

자산 모델을 설명하는 방법

다음 describe-asset-model 예시에서는 풍력 발전 단지 자산 모델을 설명합니다.

```

aws iotsitewise describe-asset-model \
  --asset-model-id a1b2c3d4-5678-90ab-cdef-2222EXAMPLE

```

출력:

```

{
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-2222EXAMPLE",

```



```

    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-2222EXAMPLE",
    "assetModelName": "Wind Farm Model",
    "assetModelDescription": "Represents a wind farm that comprises many wind
turbines",
    "assetModelProperties": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",
            "name": "Total Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "metric": {
                    "expression": "sum(power)",
                    "variables": [
                        {
                            "name": "power",
                            "value": {
                                "propertyId": "a1b2c3d4-5678-90ab-
cdef-66666EXAMPLE",
                                "hierarchyId": "a1b2c3d4-5678-90ab-
cdef-77777EXAMPLE"
                            }
                        }
                    ]
                },
                "window": {
                    "tumbling": {
                        "interval": "1h"
                    }
                }
            }
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-88888EXAMPLE",
            "name": "Region",
            "dataType": "STRING",
            "type": {
                "attribute": {
                    "defaultValue": " "
                }
            }
        }
    ]
},

```

```

    "assetModelHierarchies": [
      {
        "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",
        "name": "Wind Turbines",
        "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
      }
    ],
    "assetModelCreationDate": 1575671284.0,
    "assetModelLastUpdateDate": 1575671988.0,
    "assetModelStatus": {
      "state": "ACTIVE"
    }
  }
}

```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Describing a specific asset model](#)을 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [DescribeAssetModel](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/**
 * Retrieves the property IDs associated with a specific asset model.
 *
 * @param assetModelId the ID of the asset model that defines the properties.
 * @return a {@link CompletableFuture} that represents a {@link Map} result
 * that associates the property name to the
 *         property ID. The calling code can attach callbacks, then handle the
 * result or exception by calling
 *         {@link CompletableFuture#join()} or {@link
 * CompletableFuture#get()}.
 *
 * <p>
 * If any completion stage in this method throws an exception, the
 * method logs the exception cause and keeps

```

```

    *           it available to the calling code as a {@link CompletionException}.
    By calling
    *           {@link CompletionException#getCause()}, the calling code can
    access the original exception.
    */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs:
{}.", throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
}

```

- API 세부 정보는 API 참조의 [DescribeAssetModel](#) AWS SDK for Java 2.x 을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
  }
};

```

- API 세부 정보는 API 참조의 [DescribeAssetModel](#) AWS SDK for JavaScript 을 참조하세요.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribeGateway** 함께 사용

다음 코드 예시는 DescribeGateway의 사용 방법을 보여 줍니다.

CLI

AWS CLI

게이트웨이를 설명하는 방법

다음 describe-gateway 예시에서는 게이트웨이를 설명합니다.

```
aws iotsitewise describe-gateway \
  --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

출력:

```
{
  "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
}
```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Ingesting data using a gateway](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [DescribeGateway](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/**
 * 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());
            }
        });
}
}

```

- API 세부 정보는 API 참조의 [DescribeGateway](#) AWS SDK for Java 2.x 를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
  }
};
```

- API 세부 정보는 API 참조의 [DescribeGateway](#) AWS SDK for JavaScript 를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [DescribeGateway](#)를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 [섹션을 참조하세요](#)[AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **DescribePortal** 함께 사용

다음 코드 예시는 DescribePortal의 사용 방법을 보여 줍니다.

CLI

AWS CLI

포털을 설명하는 방법

다음 describe-portal 예시에서는 풍력 발전 단지 회사의 웹 포털에 대해 설명합니다.

```

aws iotsitewise describe-portal \
    --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE

```

출력:

```
{
  "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"
}
```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Administering your portals](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [DescribePortal](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * 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();
            });
    }
}

```

- API 세부 정보는 API 참조의 [DescribePortal](#) AWS SDK for Java 2.x 을 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
  }
};

```

- API 세부 정보는 API 참조의 [DescribePortal](#) AWS SDK for JavaScript 을 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [DescribePortal](#)을 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **GetAssetPropertyValue** 함께 사용

다음 코드 예시는 GetAssetPropertyValue의 사용 방법을 보여 줍니다.

CLI

AWS CLI

자산 속성의 현재 값을 검색하는 방법

다음 get-asset-property-value 예시에서는 풍력 터빈 자산의 현재 총 전력을 검색합니다.

```

aws iotsitewise get-asset-property-value \
  --asset-id a1b2c3d4-5678-90ab-cdef-3333EXAMPLE \
  --property-id a1b2c3d4-5678-90ab-cdef-6666EXAMPLE

```

출력:

```
{
```

```

    "propertyValue": {
      "value": {
        "doubleValue": 6890.8677520453875
      },
      "timestamp": {
        "timeInSeconds": 1580853000,
        "offsetInNanos": 0
      },
      "quality": "GOOD"
    }
  }
}

```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Querying current asset property values](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [GetAssetPropertyValue](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/**
 * 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();
            });
    }
}

```

- API 세부 정보는 API 참조의 [GetAssetPropertyValue](#) AWS SDK for Java 2.x 를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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;
    }
  }
};

```

- API 세부 정보는 API 참조의 [GetAssetPropertyValue](#) AWS SDK for JavaScript 를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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

```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [GetAssetPropertyValue](#)를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 CLI와 **ListAssetModels** 함께 사용

다음 코드 예시는 ListAssetModels의 사용 방법을 보여 줍니다.

CLI

AWS CLI

모든 자산 모델 나열

다음 list-asset-models 예시에서는 현재 리전의 AWS 계정에 정의된 모든 자산 모델을 나열합니다.

```
aws iotsitewise list-asset-models
```

출력:

```

{
  "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"
        }
    }
]
}

```

자세한 내용은 AWS IoT SiteWise 사용 설명서의 [Listing all asset models](#)를 참조하세요.

- API 세부 정보는 AWS CLI 명령 참조의 [ListAssetModels](#) 섹션을 참조하세요.

Java

SDK for Java 2.x

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/**
 * 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;
    });
}

```

- API 세부 정보는 API 참조의 [ListAssetModels](#) AWS SDK for Java 2.x 를 참조하세요.

JavaScript

SDK for JavaScript (v3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
  }
};
```

- API 세부 정보는 API 참조의 [ListAssetModels](#) AWS SDK for JavaScript 를 참조하세요.

Python

SDK for Python (Boto3)

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 SDK for Python (Boto3) API 참조의 [ListAssetModels](#)를 참조하세요. AWS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 섹션을 참조하세요 [AWS SDK에서이 서비스 사용](#). 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

의 보안 AWS IoT SiteWise

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 규정 [AWS 준수 프로그램](#) 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 [AWS 프로그램 제공 범위 내 서비스규정 준수](#) AWS IoT SiteWise참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다 AWS IoT SiteWise. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 AWS IoT SiteWise 를 구성하는 방법을 보여줍니다. 또한 AWS IoT SiteWise 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

- [의 데이터 보호 AWS IoT SiteWise](#)
- [의 데이터 암호화 AWS IoT SiteWise](#)
- [에 대한 자격 증명 및 액세스 관리 AWS IoT SiteWise](#)
- [에 대한 규정 준수 검증 AWS IoT SiteWise](#)
- [의 복원력 AWS IoT SiteWise](#)
- [의 인프라 보안 AWS IoT SiteWise](#)
- [의 구성 및 취약성 분석 AWS IoT SiteWise](#)
- [용 VPC 엔드포인트 AWS IoT SiteWise](#)
- [에 대한 보안 모범 사례 AWS IoT SiteWise](#)

의 데이터 보호 AWS IoT SiteWise

AWS [공동 책임 모델](#)의 데이터 보호에 적용됩니다 AWS IoT SiteWise. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 [데이터 프라이버시 FAQ](#)를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 [AWS 공동 책임 모델 및 GDPR](#) 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 [CloudTrail 추적 작업을 참조하세요](#).
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 [Federal Information Processing Standard\(FIPS\) 140-3](#)을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 AWS IoT SiteWise 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

주제

- [에 대한 Internetwork 트래픽 개인 정보 보호 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 어시스턴트 비즈니스 서비스 개선](#)

에 대한 Internetwork 트래픽 개인 정보 보호 AWS IoT SiteWise

SiteWise Edge 게이트웨이와 같은 AWS IoT SiteWise 와 온프레미스 애플리케이션 간의 연결은 전송 계층 보안(TLS) 연결을 통해 보호됩니다. 자세한 내용은 [에 대한 전송 중 데이터 암호화 AWS IoT SiteWise](#) 단원을 참조하십시오.

AWS IoT SiteWise 는 AWS 리전 내 가용 영역 간 연결 또는 AWS 계정 간 연결을 지원하지 않습니다.

한 번에 한 리전에서만 IAM Identity Center를 구성할 수 있습니다. SiteWise Monitor는 IAM Identity Center를 위해 구성된 리전에 연결됩니다. 즉, IAM Identity Center 액세스에는 하나의 리전을 사용하지만 어느 리전에서든 포털을 생성할 수 있습니다.

AWS IoT SiteWise 어시스턴트 비즈니스 서비스 개선

AWS IoT SiteWise Assistant는 서비스 개선 또는 기본 LLMs 개선을 위해 고객 데이터를 사용하지 않습니다.

의 데이터 암호화 AWS IoT SiteWise

데이터 암호화는 전송 중(와 주고받을 때 AWS IoT SiteWise, SiteWise Edge 게이트웨이와 서버 간에) 및 저장 중(로컬 디바이스 또는 AWS 서비스에 저장되는 동안) 데이터를 보호하는 것을 말합니다. 전송 계층 보안(TLS)을 사용하여 전송 중인 데이터를 보호하거나 클라이언트 측 암호화를 사용하여 유훈 상태인 데이터를 보호할 수 있습니다.

Note

AWS IoT SiteWise 엣지 처리는 SiteWise Edge 게이트웨이 내에서 호스팅되고 로컬 네트워크를 통해 액세스할 수 있는 APIs를 노출합니다. 이러한 APIs는 AWS IoT SiteWise Edge 커넥터가 소유한 서버 인증서가 지원하는 TLS 연결을 통해 노출됩니다. 클라이언트 인증의 경우 이러한 API는 액세스 제어 암호를 사용합니다. 서버 인증서 프라이빗 키와 액세스 제어 암호는 모두 디스크에 저장됩니다. AWS IoT SiteWise 엣지 처리는 저장 시 이러한 자격 증명의 보안을 위해 파일 시스템 암호화에 의존합니다.

서버 측 암호화 및 클라이언트 측 암호화에 대한 자세한 내용은 아래 나열된 주제를 검토하세요.

주제

- [에서 저장 시 암호화 AWS IoT SiteWise](#)
- [에 대한 전송 중 데이터 암호화 AWS IoT SiteWise](#)

• [의 키 관리 AWS IoT SiteWise](#)

에서 저장 시 암호화 AWS IoT SiteWise

AWS IoT SiteWise 는 AWS 클라우드 및 AWS IoT SiteWise Edge 게이트웨이에 데이터를 저장합니다.

AWS 클라우드에 저장된 데이터

AWS IoT SiteWise 는 기본적으로 저장 데이터를 암호화하는 다른 AWS 서비스에 데이터를 저장합니다. 유틸리티 시 암호화는 자산 속성 값과 집계 값을 암호화하는 데 사용되는 암호화 키를 관리하기 위해 AWS Key Management Service (AWS KMS)와 통합됩니다 AWS IoT SiteWise. AWS IoT SiteWise에서 자산 속성 값과 집계 값을 암호화하기 위해 고객 관리형 키를 사용하도록 선택할 수 있습니다. AWS KMS를 통해 KMS 키 암호화 키를 직접 생성하고 관리할 수 있습니다.

AWS 소유 키 를 선택하여 데이터를 암호화하거나 고객 관리형 키를 선택하여 자산 속성 값과 집계 값을 암호화할 수 있습니다.

작동 방법

유틸리티 시 암호화는 데이터를 암호화하는 데 사용되는 암호화 키를 관리하기 위해와 통합됩니다.

- **AWS 소유 키** - 기본 암호화 key. AWS IoT SiteWise owns 이 키입니다. AWS 계정에서 이 키를 볼 수 없습니다. 또한 AWS CloudTrail 로그에서 키에 대한 작업을 볼 수 없습니다. 추가 비용 없이 이 키를 사용할 수 있습니다.
- **고객 관리형 키** - 키는 사용자가 생성, 소유 및 관리하는 계정에 저장됩니다. KMS 키를 완전히 제어할 수 있습니다. 추가 AWS KMS 요금이 적용됩니다.

AWS 소유 키

AWS 소유 키 는 계정에 저장되지 않습니다. 여러 AWS 계정에서 사용하기 위해를 AWS 소유하고 관리하는 KMS 키 모음의 일부입니다. AWS 서비스는를 사용하여 AWS 소유 키 데이터를 보호할 수 있습니다.

사용을 확인, 관리 AWS 소유 키, 사용 또는 감사할 수 없습니다. 하지만 사용자는 데이터를 암호화하는 키를 보호하기 위해 어떤 작업을 수행하거나 어떤 프로그램을 변경할 필요가 없습니다.

를 사용하는 경우 월별 요금 또는 사용 요금이 부과되지 AWS 소유 키 않으며 계정의 AWS KMS 할당량에 포함되지 않습니다.

고객 관리형 키

고객 관리형 키는 사용자가 생성, 소유 및 관리하는 계정의 KMS 키입니다. 다음과 같애 | KMS 키를 완전히 제어할 수 있습니다.

- 키 정책, IAM 정책 및 권한 부여 수립 및 유지
- 활성화 및 비활성화
- 암호화 자료 교체
- 태그 추가
- 이를 참조하는 별칭 생성
- KMS 키 삭제 예약

CloudTrail 및 Amazon CloudWatch Logs를 사용하여가 사용자를 대신하여에 AWS IoT SiteWise 보내는 요청을 추적할 수도 AWS KMS 있습니다.

고객 관리형 키를 사용하는 경우 계정에 저장된 KMS 키에 대한 AWS IoT SiteWise 액세스 권한을 부여해야 합니다.는 봉투 암호화 및 키 계층 구조를 AWS IoT SiteWise 사용하여 데이터를 암호화합니다. AWS KMS 암호화 키는이 키 계층 구조의 루트 키를 암호화하는 데 사용됩니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 [봉투 암호화](#)를 참조하세요.

다음 예제 정책은 사용자를 대신하여 고객 관리형 키 생성에 AWS IoT SiteWise 권한을 부여합니다. 키를 생성할 때는 kms:CreateGrant 및 kms:DescribeKey 작업을 허용해야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1603902045292",
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

생성한 권한 부여의 암호화 컨텍스트는 사용자 `aws:iotsitewise:subscriberId` 및 계정 ID를 사용합니다.

SiteWise Edge 게이트웨이의 저장 데이터

AWS IoT SiteWise 게이트웨이는 로컬 파일 시스템에 다음 데이터를 저장합니다.

- OPC UA 소스 구성 정보
- 연결된 OPC UA 소스에서 오는 OPC UA 데이터 스트림 경로 집합
- SiteWise Edge 게이트웨이의 인터넷 연결이 끊길 때 캐싱된 산업 데이터

SiteWise Edge 게이트웨이는 on. AWS IoT Greengrass 에서 실행됩니다 AWS IoT Greengrass.는 Unix 파일 권한 및 전체 디스크 암호화(활성화된 경우)를 사용하여 코어에 저장된 데이터를 보호합니다. 파일 시스템 및 디바이스의 보안을 유지하는 것은 사용자의 책임입니다.

그러나 AWS IoT Greengrass 는 Secrets Manager에서 검색된 OPC UA 서버 보안 암호의 로컬 사본을 암호화합니다. 자세한 내용은 AWS IoT Greengrass Version 1 개발자 안내서의 [암호 암호화](#)를 참조하세요.

AWS IoT Greengrass 코어의 저장 데이터 암호화에 대한 자세한 내용은 AWS IoT Greengrass Version 1 개발자 안내서의 [저장 데이터 암호화](#)를 참조하세요.

에 대한 전송 중 데이터 암호화 AWS IoT SiteWise

AWS IoT SiteWise 는 전송 중 암호화를 사용하여 디바이스, 게이트웨이 및 AWS 클라우드 간에 전송되는 데이터를 보호합니다. 와의 통신 AWS IoT SiteWise 은 HTTPS 및 TLS 1.2를 사용하여 암호화되므로 데이터가 기밀로 유지되고 무단 액세스 또는 가로채기로부터 보호됩니다.

세 가지 통신 모드로 데이터를 전송할 수 있습니다.

- [인터넷을 통해](#) - 로컬 디바이스(SiteWise Edge 게이트웨이 포함)와 간의 통신 AWS IoT SiteWise 이 암호화됩니다.
- [로컬 네트워크를 통해 전송](#) - SiteWise용 OpsHub 애플리케이션과 SiteWise Edge 게이트웨이 간의 통신은 항상 암호화됩니다. 브라우저 내에서 실행되는 SiteWise Monitor 애플리케이션과 SiteWise Edge 게이트웨이 간의 통신은 항상 암호화됩니다. SiteWise Edge 게이트웨이와 OPC UA 소스 간의 통신을 암호화할 수 있습니다.
- [SiteWise Edge 게이트웨이의 구성 요소 간](#) - SiteWise Edge 게이트웨이의 AWS IoT Greengrass 구성 요소 간 통신은 암호화되지 않습니다.

주제

- [인터넷을 통해 전송 중인 데이터](#)
- [로컬 네트워크를 통해 전송 중인 데이터](#)
- [SiteWise Edge 게이트웨이 로컬 구성 요소 간의 전송 중 데이터](#)

인터넷을 통해 전송 중인 데이터

AWS IoT SiteWise 는 TLS(전송 계층 보안)를 사용하여 인터넷을 통한 모든 통신을 암호화합니다. AWS 클라우드로 전송되는 모든 데이터는 MQTT 또는 HTTPS 프로토콜을 사용하여 TLS 연결을 통해 전송되므로 기본적으로 안전합니다. 에서 실행되는 SiteWise Edge 게이트웨이 AWS IoT Greengrass 와 속성 값 알림은 AWS IoT 전송 보안 모델을 사용합니다. 자세한 정보는 AWS IoT 개발자 안내서의 [전송 보안](#)을 참조하세요.

로컬 네트워크를 통해 전송 중인 데이터

SiteWise Edge 게이트웨이는 로컬 OPC UA 소스와의 통신을 위해 OPC UA 사양을 따릅니다. 사용자는 메시지 보안 모드를 사용하여 전송 중인 데이터를 암호화하도록 소스를 구성할 책임이 있습니다.

서명 메시지 보안 모드를 선택하면 SiteWise Edge 게이트웨이와 소스 간의 전송 중 데이터는 서명되지만 암호화되지 않습니다. 서명 및 암호화 메시지 보안 모드를 선택하면 SiteWise Edge 게이트웨이와 소스 간의 전송 중 데이터가 서명되고 암호화됩니다. 소스 구성에 대한 자세한 내용은 [AWS IoT SiteWise Edge 게이트웨이에 데이터 소스 추가](#)을 참조하세요.

엣지 콘솔 애플리케이션과 SiteWise Edge 게이트웨이 간의 통신은 항상 TLS로 암호화됩니다. SiteWise Edge 게이트웨이의 SiteWise Edge 커넥터는 AWS IoT SiteWise 애플리케이션용 엣지 콘솔과의 TLS 연결을 설정할 수 있도록 자체 서명된 인증서를 생성하고 저장합니다. AWS IoT SiteWise 애플리케이션을 SiteWise Edge 게이트웨이에 연결하기 전에 SiteWise Edge 게이트웨이에서 애플리케이션용 엣지 콘솔로 인증서를 복사해야 합니다. 이렇게 하면 AWS IoT SiteWise 애플리케이션용 엣지 콘솔이 신뢰할 수 있는 SiteWise Edge 게이트웨이에 연결되었는지 확인할 수 있습니다.

SiteWise Edge는 보안 및 서버 인증을 위한 TLS 외에도 SigV4 프로토콜을 사용하여 엣지 콘솔 애플리케이션의 신뢰성을 설정합니다. SiteWise Edge 게이트웨이의 SiteWise Edge 커넥터는 AWS IoT SiteWise SDK를 기반으로 엣지 콘솔 애플리케이션, 브라우저 내에서 실행되는 SiteWise Monitor 애플리케이션 및 기타 클라이언트에서 들어오는 연결을 확인할 수 있는 암호를 수락하고 저장합니다.

비밀번호 및 서버 인증서 생성에 대한 자세한 내용은 [the section called “게이트웨이 관리”](#)을 참조하세요.

SiteWise Edge 게이트웨이 로컬 구성 요소 간의 전송 중 데이터

SiteWise Edge 게이트웨이는에서 실행되며 AWS IoT Greengrass, 데이터가 디바이스를 벗어나지 않기 때문에 코어에서 AWS IoT Greengrass 로컬로 교환되는 데이터를 암호화하지 않습니다. 여기에는 AWS IoT SiteWise 커넥터와 같은 구성 요소 간의 AWS IoT Greengrass 통신이 포함됩니다. 자세한 정보는 AWS IoT Greengrass Version 1 개발자 안내서의 [코어 디바이스의 데이터](#)를 참조하세요.

의 키 관리 AWS IoT SiteWise

AWS IoT SiteWise 클라우드 키 관리

기본적으로는 AWS 관리형 키를 AWS IoT SiteWise 사용하여 AWS 클라우드의 데이터를 보호합니다. 설정을 업데이트하여 고객 관리형 키를 사용하여 AWS IoT SiteWise에서 일부 데이터를 암호화할 수 있습니다. AWS Key Management Service (AWS KMS)을 통해 암호화 키를 만들고, 관리하고, 볼 수 있습니다.

AWS IoT SiteWise 는 다음 데이터를 암호화 AWS KMS 하기 위해에 저장된 고객 관리형 키를 사용한 서버 측 암호화를 지원합니다.

- 자산 속성 값
- 값 집계

Note

다른 데이터 및 리소스에서 관리하는 키가 있는 기본 암호화를 사용하여 암호화됩니다 AWS IoT SiteWise. 이 키는 AWS IoT SiteWise 계정에 저장됩니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 [What is AWS Key Management Service?](#)를 참조하세요.

고객 관리형 키를 사용하여 암호화를 활성화합니다.

에서 고객 관리형 키를 사용하려면 설정을 업데이트 AWS IoT SiteWise AWS IoT SiteWise 해야 합니다.

KMS 키를 사용하여 암호화를 활성화하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.

2. 계정 설정을 선택하고 편집을 선택하여 계정 설정 편집 페이지를 엽니다.
3. 암호화 키 유형에서 다른 AWS KMS 키 선택을 선택합니다. 이렇게 하면 고객 관리형 키를 저장하여 AWS KMS에 암호화할 수 있습니다.

Note

현재는 자산 속성 값 및 집계 값에 대해서만 고객이 관리하는 키 암호화를 사용할 수 있습니다.

4. 다음 옵션 중 하나로 KMS 키를 선택합니다.
 - 기존 KMS 키를 사용하려면 - 목록에서 KMS 키 별칭을 선택합니다.
 - 새 KMS 키를 생성하려면 - AWS KMS 키 생성을 선택합니다.

Note

AWS KMS 대시보드가 열립니다. KMS 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 [키 생성](#)을 참조하세요.

5. 설정을 저장하려면 저장을 선택합니다.

SiteWise Edge 게이트웨이 키 관리

SiteWise Edge 게이트웨이는에서 실행 AWS IoT Greengrass되며 AWS IoT Greengrass 코어 디바이스는 퍼블릭 및 프라이빗 키를 사용하여 AWS 클라우드로 인증하고 OPC UA 인증 보안 암호와 같은 로컬 보안 암호를 암호화합니다. 자세한 정보는 AWS IoT Greengrass Version 1 개발자 설명서의 [키 관리](#)를 참조하세요.

에 대한 자격 증명 및 액세스 관리 AWS IoT SiteWise

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와주는입니다. IAM 관리자는 누가 AWS IoT SiteWise 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는입니다.

주제

- [AWS IoT SiteWise 보안 대상](#)

- [에서 자격 증명으로 인증 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 에서 IAM을 사용하는 방법](#)
- [AWS 에 대한 관리형 정책 AWS IoT SiteWise](#)
- [에 대한 서비스 연결 역할 사용 AWS IoT SiteWise](#)
- [에서 이벤트 경보에 대한 권한 설정 AWS IoT SiteWise](#)
- [의 교차 서비스 혼동된 대리자 방지 AWS IoT SiteWise](#)
- [AWS IoT SiteWise 자격 증명 및 액세스 문제 해결](#)

AWS IoT SiteWise 보안 대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 수행하는 작업에 따라 다릅니다 AWS IoT SiteWise.

서비스 사용자 - AWS IoT SiteWise 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 AWS IoT SiteWise 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. AWS IoT SiteWise의 기능에 액세스할 수 없는 경우 [AWS IoT SiteWise 자격 증명 및 액세스 문제 해결](#)을 참조하세요.

서비스 관리자 - 회사에서 AWS IoT SiteWise 리소스를 책임지고 있는 경우에 대한 전체 액세스 권한을 가지고 있을 것입니다 AWS IoT SiteWise. 서비스 관리자는 서비스 사용자가 액세스해야 하는 AWS IoT SiteWise 기능과 리소스를 결정합니다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가에서 IAM을 사용하는 방법에 대한 자세한 내용은 섹션을 AWS IoT SiteWise참조하세요 [AWS IoT SiteWise 에서 IAM을 사용하는 방법](#).

IAM 관리자 - IAM 관리자라면 AWS IoT SiteWise에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 자격 AWS IoT SiteWise 증명 기반 정책 예제를 보려면 섹션을 참조하세요 [AWS IoT SiteWise 자격 증명 기반 정책 예제](#).

에서 자격 증명으로 인증 AWS IoT SiteWise

인증은 자격 증명 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자 또는 AWS 계정의 루트 사용자 IAM 역할을 수임하여 로 인증(로그인 AWS)되어야 합니다.

자격 증명 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 자격 증명 AWS 으로 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증 및 Google 또는 Facebook 자격 증명은 페더레이션 자격 증명의 예입니다. 페더레이션형 ID로 로그인

할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임하게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 [로그인하는 방법을 AWS참조하세요. AWS 계정](#)

AWS 프로그래밍 방식으로 액세스하는 경우는 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK)와 명령줄 인터페이스(CLI)를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 자세한 방법은 IAM 사용 설명서에서 [API 요청용 AWS Signature Version 4](#)를 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어는 멀티 팩터 인증(MFA)을 사용하여 계정의 보안을 강화할 것을 AWS 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 [다중 인증](#) 및 IAM 사용 설명서에서 [IAM의 AWS 다중 인증](#)을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 계정의 모든 AWS 서비스 및 리소스에 대한 전체 액세스 권한이 있는 하나의 로그인 자격 증명으로 AWS 계정 시작합니다. 이 자격 증명을 AWS 계정 루트 사용자라고 하며 계정을 생성하는 데 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명을 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 [루트 사용자 자격 증명이 필요한 작업](#)을 참조하세요.

IAM 사용자 및 그룹

[IAM 사용자](#)는 단일 사용자 또는 애플리케이션에 대한 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 [장기 보안 인증이 필요한 사용 사례의 경우, 정기적으로 액세스 키 교체](#)를 참조하세요.

[IAM 그룹](#)은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 자격 증명을 가지고 있

지만, 역할은 임시 보안 인증만 제공합니다. 자세한 내용은 IAM 사용 설명서에서 [IAM 사용자 사용 사례](#)를 참조하세요.

IAM 역할

[IAM 역할](#)은 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 에서 IAM 역할을 일시적으로 수입하려면 사용자에서 IAM 역할(콘솔)로 전환할 AWS Management Console을 사용합니다. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-console.html 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 AWS CLI 사용하여 역할을 수입할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 [역할 수입 방법](#)을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 관련 역할에 대한 자세한 내용은 IAM 사용 설명서의 [Create a role for a third-party identity provider \(federation\)](#)를 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연관짓습니다. 권한 집합에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 [권한 집합](#)을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수입하여 특정 작업에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 교차 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부에서는 정책을 리소스에 직접 연결할 AWS 서비스수 있습니다(역할을 프록시로 사용하는 대신). 교차 계정 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 [IAM의 교차 계정 리소스 액세스](#)를 참조하세요.
- 교차 서비스 액세스 - 일부는 다른에서 기능을 AWS 서비스 사용합니다 AWS 서비스. 예를 들어, 서비스에서 호출하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 직접적으로 호출하는 위탁자의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
- 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하기 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. 이 경우, 두 작업을 모두

수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 [전달 액세스 세션](#)을 참조하세요.

- 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 맡는 [IAM 역할](#)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [Create a role to delegate permissions to an AWS 서비스](#)를 참조하세요.
- 서비스 연결 역할 - 서비스 연결 역할은 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.
- Amazon EC2에서 실행되는 애플리케이션 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여](#)를 참조하세요.

AWS IoT SiteWise 에서 IAM을 사용하는 방법

AWS Identity and Access Management (IAM)를 사용하여 액세스를 관리하기 전에 어떤 IAM 기능을 사용할 수 있는지 이해해야 AWS IoT SiteWise합니다 AWS IoT SiteWise.

IAM 기능

예
서
지
원
하
나
요
AWS
IoT
SiteWise

[리소스 수준 권한이 있는 자격 증명 기반 정책](#)

예

[정책 작업](#)

예

IAM 기능	예
정책 리소스	예
정책 조건 키	예
리소스 기반 정책	아니요
액세스 제어 목록(ACL)	아니요
태그 기반 권한 부여(ABAC)	예
임시 보안 인증	예
전달 액세스 세션(FAS)	예
서비스 연결 역할	예
서비스 역할	예

AWS IoT SiteWise 및 기타 AWS 서비스에서 IAM을 사용하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 [AWS IAM으로 작업하는 서비스를](#) 참조하세요.

목차

- [AWS IoT SiteWise IAM 역할](#)

- [에서 임시 자격 증명 사용 AWS IoT SiteWise](#)
- [에 대한 전달 액세스 세션\(FAS\) AWS IoT SiteWise](#)
- [서비스 연결 역할](#)
- [서비스 역할](#)
- [AWS IoT SiteWise에서 IAM 역할 선택](#)
- [AWS IoT SiteWise 태그 기반 인증](#)
- [AWS IoT SiteWise 자격 증명 기반 정책](#)
 - [정책 작업](#)
 - [BatchPutAssetPropertyValue 권한 부여](#)
 - [정책 리소스](#)
 - [정책 조건 키](#)
 - [예시](#)
- [AWS IoT SiteWise 자격 증명 기반 정책 예제](#)
 - [정책 모범 사례](#)
 - [AWS IoT SiteWise 콘솔을 사용합니다.](#)
 - [사용자가 자신의 고유한 권한을 볼 수 있도록 허용](#)
 - [사용자가 하나의 계층에서 자산에 데이터를 수집할 수 있도록 허용](#)
 - [태그를 기반으로 AWS IoT SiteWise 자산 보기](#)
- [의 정책을 사용하여 액세스 관리 AWS IoT SiteWise](#)
 - [ID 기반 정책](#)
 - [리소스 기반 정책](#)
 - [액세스 제어 목록\(ACL\)](#)
 - [기타 정책 타입](#)
 - [여러 정책 유형](#)

AWS IoT SiteWise IAM 역할

[IAM 역할](#)은 특정 권한을 가지고 있는 AWS 계정 내 엔터티입니다.

에서 임시 자격 증명 사용 AWS IoT SiteWise

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. [AssumeRole](#) 또는 [GetFederationToken](#) 같은 AWS STS API 작업을 호출하여 임시 보안 인증을 가져옵니다.

AWS IoT SiteWise 는 임시 자격 증명 사용을 지원합니다.

SiteWise Monitor에서는 페더레이션 사용자가 포털에 액세스할 수 있도록 지원합니다. 포털 사용자는 IAM Identity Center 또는 IAM 보안 인증으로 인증합니다.

Important

사용자 또는 역할에는 포털에 로그인할 수 있는 `iotsitewise:DescribePortal` 권한이 있어야 합니다.

사용자가 포털에 로그인하면 SiteWise Monitor에서는 다음 권한을 제공하는 세션 정책을 생성합니다.

- 해당 포털의 역할이 액세스를 제공하는 계정의 AWS IoT SiteWise 에 있는 자산 및 자산 데이터에 대한 읽기 전용 액세스입니다.
- 사용자에게 관리자(프로젝트 소유자) 또는 읽기 전용(프로젝트 뷰어) 액세스 권한이 있는 포털의 프로젝트에 액세스합니다.

페더레이션 포털 사용자 권한에 대한 자세한 내용은 [에 대한 서비스 역할 사용 AWS IoT SiteWise Monitor](#)을 참조하세요.

에 대한 전달 액세스 세션(FAS) AWS IoT SiteWise

전달 액세스 세션(FAS) 지원: 예

IAM 사용자 또는 역할을 사용하여에서 작업을 수행하는 경우 AWS보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우, 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청은 서비스가 완료하기 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. 이 경우, 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 [전달 액세스 세션](#)을 참조하세요.

서비스 연결 역할

[서비스 연결 역할](#)을 사용하면 AWS 서비스가 다른 서비스의 리소스에 액세스하여 사용자를 대신하여 작업을 완료할 수 있습니다. 서비스 연결 역할은 AWS 계정에 표시되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수 없습니다.

AWS IoT SiteWise 는 서비스 연결 역할을 지원합니다. AWS IoT SiteWise 서비스 연결 역할을 생성 또는 관리하는 방법에 대한 자세한 내용은 [에 대한 서비스 연결 역할 사용 AWS IoT SiteWise](#)을 참조하세요.

서비스 역할

이 기능을 사용하면 서비스가 사용자를 대신하여 [서비스 역할](#)을 수입할 수 있습니다. 이 역할을 사용하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 작업을 완료할 수 있습니다. 서비스 역할은 AWS 계정에 나타나며 계정이 소유합니다. 즉, IAM 관리자가 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

AWS IoT SiteWise 는 서비스 역할을 사용하여 SiteWise Monitor 포털 사용자가 사용자를 AWS IoT SiteWise 대신하여 일부 리소스에 액세스할 수 있도록 허용합니다. 자세한 내용은 [에 대한 서비스 역할 사용 AWS IoT SiteWise Monitor](#) 단원을 참조하십시오.

AWS IoT Events 경보 모델을 생성하려면 먼저 필요한 권한이 있어야 합니다 AWS IoT SiteWise. 자세한 내용은 [에서 이벤트 경보에 대한 권한 설정 AWS IoT SiteWise](#) 단원을 참조하십시오.

AWS IoT SiteWise에서 IAM 역할 선택

에서 portal 리소스를 생성할 때 SiteWise Monitor 포털의 페더레이션 사용자가 AWS IoT SiteWise 사용자를 대신하여 액세스할 수 있도록 허용하는 역할을 선택해야 AWS IoT SiteWise합니다. 이전에 서비스 역할을 생성한 경우는 선택할 수 있는 역할 목록을 AWS IoT SiteWise 제공합니다. 그렇지 않으면 포털을 만들 때 필요한 권한이 있는 역할을 만들 수 있습니다. 자산 및 자산 데이터에 액세스할 수 있는 역할을 선택하는 것이 중요합니다. 자세한 내용은 [에 대한 서비스 역할 사용 AWS IoT SiteWise Monitor](#) 단원을 참조하십시오.

AWS IoT SiteWise 태그 기반 인증

AWS IoT SiteWise 리소스에 태그를 연결하거나 요청에 태그를 전달할 수 있습니다 AWS IoT SiteWise. 태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 [조건 요소](#)에 태그 정보를 제공합니다. AWS IoT SiteWise 리소스 태깅에 대한 자세한 내용은 [AWS IoT SiteWise 리소스에 태그 지정](#)을 참조하세요.

리소스의 태그를 기반으로 리소스에 대한 액세스를 제한하는 자격 증명 기반 정책의 예시는 [태그를 기반으로 AWS IoT SiteWise 자산 보기](#)에서 확인할 수 있습니다.

AWS IoT SiteWise 자격 증명 기반 정책

IAM 정책을 사용하면 AWS IoT SiteWise에서 누가 어떤 작업을 수행할 수 있는지 제어할 수 있습니다. 허용되는 작업을 결정하고 이러한 작업에 대한 구체적인 조건을 설정할 수 있습니다. 예를 들어에서 정보를 보거나 변경할 수 있는 사용자에게 대한 규칙을 만들 수 있습니다 AWS IoT SiteWise.는 특정 작업, 리소스 및 조건 키를 AWS IoT SiteWise 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 [IAM JSON 정책 요소 참조](#)를 참조하세요.

정책 작업

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 정책 작업은 일반적으로 연결된 AWS API 작업과 이름이 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

의 정책 작업은 작업 앞에 접두사를 AWS IoT SiteWise 사용합니다 `iotsitewise:`. 예를 들어 API `BatchPutAssetPropertyValue` 작업을 사용하여 자산 속성 데이터를 업로드할 수 AWS IoT SiteWise 있는 권한을 부여하려면 해당 정책에 `iotsitewise:BatchPutAssetPropertyValue` 작업을 포함합니다. 정책 설명에는 Action 또는 NotAction 요소가 포함되어야 합니다. 이 서비스로 수행할 수 있는 작업을 설명하는 고유한 작업 세트를 AWS IoT SiteWise 정의합니다.

명령문 하나에 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
  "iotsitewise:action1",
  "iotsitewise:action2"
]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Describe라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "iotsitewise:Describe*"
```

AWS IoT SiteWise 작업 목록을 보려면 IAM 사용 설명서의에서 [정의한 작업을 AWS IoT SiteWise](#) 참조하세요.

BatchPutAssetPropertyValue 권한 부여

AWS IoT SiteWise 는 비정상적인 방식으로 [BatchPutAssetPropertyValue](#) 작업에 대한 액세스를 승인합니다. 대부분의 작업에서 액세스를 허용하거나 거부하면 권한이 부여되지 않은 경우 해당 작업에서 오류가 반환됩니다. BatchPutAssetPropertyValue를 사용하면 단일 API 요청에서 여러 데이터 항목을 서로 다른 자산 및 자산 속성으로 보낼 수 있으며, 각 데이터 항목에 독립적으로 AWS IoT SiteWise 권한을 부여할 수 있습니다. 요청에서 권한 부여에 실패한 개별 항목의 경우는 반환된 오류 목록에 AccessDeniedException를 AWS IoT SiteWise 포함합니다.는 동일한 요청의 다른 항목이 실패하더라도 권한을 부여하고 성공하는 모든 항목에 대한 데이터를 AWS IoT SiteWise 수신합니다.

Important

데이터 스트림으로 데이터를 수집하기 전에 다음을 수행하세요.

- 데이터 스트림을 식별하기 위해 속성 별칭을 사용하는 경우 time-series 리소스에 권한을 부여합니다.
- 연결된 자산 속성이 포함된 자산을 식별하기 위해 자산 ID를 사용하는 경우 asset 리소스에 권한을 부여합니다.

정책 리소스

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. 모범 사례에 따라 [Amazon 리소스 이름\(ARN\)](#)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"

```

각 IAM 정책 명령문은 ARN을 사용하여 지정한 리소스에 적용됩니다. ARN의 일반적인 구문은 다음과 같습니다.

```
arn:${Partition}:${Service}:${Region}:${Account}:${ResourceType}/${ResourcePath}
```

ARN 형식에 대한 자세한 내용은 [Amazon AWS 리소스 이름\(ARNs\)을 사용하여 리소스 식별을 참조하세요](#).

예를 들어 명령문에서 ID가 a1b2c3d4-5678-90ab-cdef-2222EXAMPLE인 자산을 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/a1b2c3d4-5678-90ab-cdef-2222EXAMPLE"
```

특정 계정에 속하는 모든 데이터 스트림을 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:iotsitewise:region:123456789012:time-series/*"
```

특정 계정에 속하는 모든 자산을 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/*"
```

리소스를 생성하기 위한 작업과 같은 일부 AWS IoT SiteWise 작업은 특정 리소스에서 수행할 수 없습니다. 이러한 경우, 와일드카드(*)를 사용해야 합니다.

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

단일 문에서 여러 리소스를 지정하려면 ARN을 쉼표로 구분합니다.

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

AWS IoT SiteWise 리소스 유형 및 해당 ARNs 목록을 보려면 IAM 사용 설명서의에서 [정의한 리소스 유형을 AWS IoT SiteWise](#) 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 [AWS IoT SiteWise가 정의한 작업](#)을 참조하십시오.

정책 조건 키

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Condition 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 옵션입니다. 같거나 작음과 같은 [조건 연산자](#)를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 Condition 요소를 지정하거나 단일 Condition 요소에서 여러 키를 지정하는 경우, AWS는 논리적 AND 작업을 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우는 논리적 OR 작업을 사용하여 조건을 AWS 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [IAM 정책 요소: 변수 및 태그](#)를 참조하세요.

AWS는 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 [AWS 전역 조건 컨텍스트 키](#)를 참조하세요.

Important

여러 조건 키들이 하나의 리소스에 딸려 있고, 일부 API 작업은 다수의 리소스를 사용합니다. 조건 키로 정책 설명문을 작성하려면 설명문의 Resource 요소를 사용해 조건 키가 적용되는 리소스를 지정합니다. 그렇게 하지 않으면, 조건 키가 해당되지 않는 리소스에 대해서는 조건 검사가 실패하여 정책이 사용자로 하여금 작업을 전혀 수행하지 못하게 막을 수도 있습니다. 리소스를 지정하고 싶지 않거나 다수의 API 작업을 포함하도록 정책의 Action 요소를 작성했다면, 반드시 ...IfExists 조건 유형을 사용해 조건 키가 그것을 사용하지 않는 리소스에 대해서는 무시되도록 해야 합니다. 자세한 내용은 IAM 사용 설명서의 [...IfExists 조건](#)을 참조하세요.

AWS IoT SiteWise는 자체 조건 키 세트를 정의하고 일부 전역 조건 키 사용을 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 [AWS 전역 조건 컨텍스트 키](#)를 참조하세요.

AWS IoT SiteWise 조건 키

조건 키	설명	유형
<code>iotsitewise:isAssociatedWithAssetProperty</code>	데이터 스트림이 자산 속성과 연결되어 있는지 여부입니다. 이 조건 키를 사용하여 데이터 스트림에 대한 관련 자산 속성의 존재 여부에 따라 권한을 정의할 수 있습니다. 예제 값: <code>true</code>	String
<code>iotsitewise:assetHierarchyPath</code>	슬래시로 구분된 자산 ID의 문자열인 자산의 계층 구조 경로입니다. 이 조건 키를 사용하여 계층에 있는 모든 자산의 계층 구조 하위 집합을 기반으로 권한을 정의할 수 있습니다. 예제 값: <code>/a1b2c3d4-5678-90ab-cdef-2222EXAMPLE/a1b2c3d4-5678-90ab-cdef-6666EXAMPLE</code>	String
<code>iotsitewise:propertyId</code>	자산 속성의 ID입니다. 자산 모델의 지정된 속성을 기반으로 사용 권한을 정의하려면 이 조건 키를 사용합니다. 이 조건 키는 해당 모델의 모든 자산에 적용됩니다. 예제 값: <code>a1b2c3d4-5678-90ab-cdef-3333EXAMPLE</code>	String
<code>iotsitewise:childAssetId</code>	다른 자산에 대한 하위 자산으로 연결된 자산 ID입니다. 이 조	String

조건 키	설명	유형
	건 키를 사용하여 하위 자산을 기반으로 사용 권한을 정의합니다. 상위 자산을 기반으로 권한을 정의하려면 정책 설명의 리소스 섹션을 사용합니다. 예제 값: a1b2c3d4-5678-90ab-cdef-66666EXAMPLE	
iotsitewise:iam	액세스 정책을 나열할 때 IAM ID의 ARN입니다. 이 조건 키를 사용하여 IAM ID에 대한 액세스 정책 권한을 정의합니다. 예제 값: arn:aws:iam::123456789012:user/JohnDoe	String, Null
iotsitewise:propertyAlias	자산 속성 또는 데이터 스트림을 식별하는 별칭입니다. 이 조건 키를 사용하여 별칭을 기반으로 사용 권한을 정의합니다.	String
iotsitewise:user	액세스 정책을 나열할 때 IAM Identity Center 사용자의 ID입니다. 이 조건 키를 사용하여 IAM Identity Center 사용자에게 대한 액세스 정책 권한을 정의합니다. 예제 값: a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE	String, Null

조건 키	설명	유형
iotsitewise:group	액세스 정책을 나열할 때 IAM Identity Center 그룹의 ID입니다. 이 조건 키를 사용하여 IAM Identity Center 그룹에 대한 액세스 정책 권한을 정의합니다. 예제 값: a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbbEXAMPLE	String, Null
iotsitewise:portal	액세스 정책의 포털 ID입니다. 포털을 기반으로 액세스 정책 권한을 정의하려면 이 조건 키를 사용합니다. 예제 값: a1b2c3d4-5678-90ab-cdef-7777EXAMPLE	String, Null
iotsitewise:project	액세스 정책의 프로젝트 ID 또는 대시보드에 대한 프로젝트 ID입니다. 이 조건 키를 사용하여 프로젝트를 기반으로 대시보드 또는 액세스 정책 권한을 정의할 수 있습니다. 예제 값: a1b2c3d4-5678-90ab-cdef-8888EXAMPLE	String, Null

조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 [에서 정의한 작업을 AWS IoT SiteWise](#) 참조하세요.

예시

자격 AWS IoT SiteWise 증명 기반 정책의 예를 보려면 섹션을 참조하세요 [AWS IoT SiteWise 자격 증명 기반 정책 예제](#).

AWS IoT SiteWise 자격 증명 기반 정책 예제

기본적으로 엔터티(사용자 및 역할)에는 AWS IoT SiteWise 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 권한을 조정하려면 AWS Identity and Access Management (IAM) 관리자가 다음을 수행해야 합니다.

1. 필요한 리소스에서 특정 API 작업을 수행할 수 있는 권한을 사용자와 역할에 부여하는 IAM 정책을 생성합니다.
2. 해당 권한이 필요한 사용자 또는 그룹에 이러한 정책을 연결합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 [JSON 탭에서 정책 생성](#)을 참조하세요.

주제

- [정책 모범 사례](#)
- [AWS IoT SiteWise 콘솔을 사용합니다.](#)
- [사용자가 자신의 고유한 권한을 볼 수 있도록 허용](#)
- [사용자가 하나의 계층에서 자산에 데이터를 수집할 수 있도록 허용](#)
- [태그를 기반으로 AWS IoT SiteWise 자산 보기](#)

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 AWS IoT SiteWise 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따릅니다.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 [AWS 관리형 정책](#) 또는 [AWS 직무에 대한 관리형 정책](#)을 참조하세요.
- 최소 권한 적용 - IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있

는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 [IAM의 정책 및 권한](#)을 참조하세요.

- IAM 정책의 조건을 사용하여 액세스 추가 제한 - 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정을 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 AWS CloudFormation. 자세한 정보는 IAM 사용 설명서의 [IAM JSON 정책 요소: 조건](#)을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 [IAM Access Analyzer에서 정책 검증](#)을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정입니다. API 작업을 직접 호출할 때 MFA가 필요하다면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 [MFA를 통한 보안 API 액세스](#)를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 [IAM의 보안 모범 사례](#)를 참조하세요.

AWS IoT SiteWise 콘솔을 사용합니다.

AWS IoT SiteWise 콘솔에 액세스하려면 기본 권한 집합이 필요합니다. 이러한 권한을 통해 AWS 계정의 AWS IoT SiteWise 리소스에 대한 세부 정보를 보고 관리할 수 있습니다.

너무 제한적인 정책을 만들면 콘솔이 해당 정책의 사용자 또는 역할(엔터티)에 대해 예상대로 작동하지 않을 수 있습니다. 이러한 엔터티가 AWS IoT SiteWise 콘솔을 계속 사용할 수 있도록 하려면 [AWSIoTSiteWiseConsoleFullAccess](#) 관리형 정책을 해당 엔터티에 연결하거나 해당 엔터티에 대해 동등한 권한을 정의합니다. 자세한 내용은 IAM 사용 설명서의 [사용자에게 권한 추가](#)를 참조하세요.

엔터티가 콘솔이 아닌 AWS Command Line Interface (CLI) 또는 AWS IoT SiteWise API만 사용하는 경우 이러한 최소 권한이 필요하지 않습니다. 이 경우 API 작업에 필요한 특정 작업에 대한 액세스 권한을 부여하기만 하면 됩니다.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
```

```

"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": "*"
  }
]
}

```

사용자가 하나의 계층에서 자산에 데이터를 수집할 수 있도록 허용

이 예제에서는 AWS 계정의 사용자에게 루트 자산 부터 시작하여 특정 자산 계층 구조의 모든 자산 속성에 데이터를 쓸 수 있는 액세스 권한을 부여하려고 합니다a1b2c3d4-5678-90ab-cdef-22222EXAMPLE. 정책은 사용자에게 `iotsitewise:BatchPutAssetPropertyValue` 권한을 부여합니다. 이 정책은 `iotsitewise:assetHierarchyPath` 조건 키를 사용하여 계층 경로가 자산 또는 하위 항목과 일치하는 자산에 대한 액세스를 제한합니다.

```

{
  "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/*"
      ]
    }
  }
}
```

태그를 기반으로 AWS IoT SiteWise 자산 보기

자격 증명 기반 정책의 조건을 사용하여 태그를 기반으로 AWS IoT SiteWise 리소스에 대한 액세스를 제어합니다. 이 예시에서는 자산 보기를 허용하는 정책을 생성할 수 있는 방법을 보여 줍니다. 그러나 자산 태그 Owner에 해당 사용자의 사용자 이름 값이 있는 경우에만 권한이 부여됩니다. 이 정책은 콘솔에서 이 작업을 완료하는 데 필요한 권한도 부여합니다.

```
{
  "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}"
  }
}
]
}

```

이 정책을 계정의 사용자에게 연결합니다. 라는 사용자가 AWS IoT SiteWise 자산을 보려고 richard-roe 하면 자산에 Owner=richard-roe 또는 태그를 지정해야 합니다 owner=richard-roe. 그러지 않으면 Richard는 액세스가 거부됩니다. 조건 태그 키 이름은 대/소문자를 구분하지 않습니다. Owner는 Owner와 owner 둘 다와 일치합니다. 자세한 정보는 IAM 사용 설명서의 [IAM JSON 정책 요소: 조건](#)을 참조하세요.

의 정책을 사용하여 액세스 관리 AWS IoT SiteWise

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결 AWS 될 때 권한을 정의하는의 객체입니다.는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청할 때 이러한 정책을 AWS 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 [JSON 정책 개요](#)를 참조하세요.

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, iam:GetRole 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console AWS CLI, 또는 API에서 역할 정보를 가져올 수 있습니다 AWS .

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 [고객 관리형 정책으로 사용자 지정 IAM 권한 정의](#)를 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은의 여러 사용자, 그룹 및 역할에 연결할 수 있는

독립 실행형 정책입니다 AWS 계정. 관리형 정책에는 AWS 관리형 정책 및 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 [관리형 정책 및 인라인 정책 중에서 선택](#)을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 위탁자가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 [위탁자를 지정](#)해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF 및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 [액세스 제어 목록\(ACL\) 개요](#)를 참조하세요.

기타 정책 타입

AWS 는 덜 일반적인 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔티티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 ID 기반 정책과 그 권한 경계의 교집합입니다. Principal 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 [IAM 엔티티에 대한 권한 경계](#)를 참조하세요.
- 서비스 제어 정책(SCPs) - SCPs는 조직 또는 조직 단위(OU)에 대한 최대 권한을 지정하는 JSON 정책입니다 AWS Organizations. AWS Organizations 는 비즈니스가 소유 AWS 계정 한 여러를 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각각을 포함하여 멤버 계정의 엔

터티에 대한 권한을 제한합니다 AWS 계정 루트 사용자. 조직 및 SCP에 대한 자세한 내용은 AWS Organizations 사용 설명서에서 [Service control policies](#)을 참조하세요.

- 리소스 제어 정책(RCP) - RCP는 소유한 각 리소스에 연결된 IAM 정책을 업데이트하지 않고 계정의 리소스에 대해 사용 가능한 최대 권한을 설정하는 데 사용할 수 있는 JSON 정책입니다. RCP는 멤버 계정의 리소스에 대한 권한을 제한하며 조직에 속하는지 여부에 AWS 계정 루트 사용자관계없이 포함 자격 증명에 대한 유효 권한에 영향을 미칠 수 있습니다. RCP를 AWS 서비스 지원하는 목록을 포함하여 조직 및 RCPs에 대한 자세한 내용은 AWS Organizations 사용 설명서의 [리소스 제어 정책\(RCPs\)](#)을 참조하세요.
- 세션 정책 - 세션 정책은 역할 또는 페더레이션 사용자에게 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 [세션 정책](#)을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 [정책 평가 로직](#)을 참조하세요.

AWS 에 대한 관리형 정책 AWS IoT SiteWise

정책을 직접 작성하는 대신 AWS 관리형 정책을 사용하여 사용자, 그룹 및 역할에 권한을 추가하는 작업을 간소화합니다. 팀에 정확한 권한을 제공하는 [IAM 고객 관리형 정책을 생성](#)하기 위해서는 시간과 전문 지식이 필요합니다. 더 빠른 설정을 위해 일반적인 사용 사례에 관리 AWS 형 정책을 사용하는 것이 좋습니다. AWS 계정에서 AWS 관리형 정책을 찾습니다. AWS 관리형 정책에 대한 자세한 정보는 [IAM 사용 설명서](#)에서 AWS 관리형 정책을 참조하세요.

AWS 서비스는 AWS 관리형 정책을 업데이트하고 유지 관리합니다. 즉, 이러한 정책의 권한을 수정할 수 없습니다. 경우에 따라 새 기능을 수용할 AWS IoT SiteWise 수 있는 권한을 추가하여 정책이 연결된 모든 자격 증명에 영향을 미칠 수 있습니다. 이러한 업데이트는 새로운 서비스 또는 기능을 도입할 때 흔히 발생합니다. 그러나 권한은 제거되지 않으므로 설정이 그대로 유지됩니다.

또한 여러 서비스에 걸쳐 있는 직무에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 ReadOnlyAccess AWS 관리형 정책은 모든 AWS 서비스 및 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면 새 작업 및 리소스에 대한 읽기 전용 권한을 AWS 추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 [직무에 관한AWS 관리형 정책](#)을 참조하세요.

AWS 관리형 정책: AWSIoTSiteWiseReadOnlyAccess

AWSIoTSiteWiseReadOnlyAccess AWS 관리형 정책을 사용하여에 대한 읽기 전용 액세스를 허용합니다 AWS IoT SiteWise.

AWSIoTSiteWiseReadOnlyAccess 정책을 IAM 보안 인증에 연결할 수 있습니다.

서비스 수준 권한

이 정책은에 대한 읽기 전용 액세스를 제공합니다 AWS IoT SiteWise. 이 정책에 다른 서비스 권한은 포함되지 않습니다.

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

AWS 관리형 정책: AWSServiceRoleForIoTSiteWise

AWSServiceRoleForIoTSiteWise 역할은 다음 권한을 가진 AWSServiceRoleForIoTSiteWise 정책을 사용합니다. 이 정책은 다음과 같습니다.

- AWS IoT SiteWise 가 SiteWise Edge 게이트웨이(에서 실행됨)를 배포하도록 허용합니다AWS IoT Greengrass.
- 가 로깅 AWS IoT SiteWise 을 수행하도록 허용합니다.
- 가 AWS IoT TwinMaker 데이터베이스에 대해 메타데이터 검색 쿼리를 실행 AWS IoT SiteWise 하도록 허용합니다.

단일 사용자 계정과 AWS IoT SiteWise 함께를 사용하는 경우 AWSServiceRoleForIoTSiteWise 역할은 IAM 계정에 AWSServiceRoleForIoTSiteWise 정책을 생성하고의 AWSServiceRoleForIoTSiteWise 서비스 연결 역할에 연결합니다. [AWS IoT SiteWise](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSiteWiseReadGreenGrass",
      "Effect": "Allow",
      "Action": [
        "greengrass:GetAssociatedRole",
        "greengrass:GetCoreDefinition",
        "greengrass:GetCoreDefinitionVersion",
        "greengrass:GetGroup",
        "greengrass:GetGroupVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowSiteWiseAccessLogGroup",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:DescribeLogGroups"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*"
    },
    {
      "Sid": "AllowSiteWiseAccessLog",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
    },
    {
      "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
      "Effect": "Allow",
      "Action": [
```

```

    "iottwinmaker:GetWorkspace",
    "iottwinmaker:ExecuteQuery"
  ],
  "Resource": "arn:aws:iottwinmaker:*:*:workspace/*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "iottwinmaker:linkedServices": [
        "IOTSITWISE"
      ]
    }
  }
}
]
}

```

AWS IoT SiteWise AWS 관리형 정책에 대한 업데이트

이 서비스가 변경 사항을 추적하기 시작한 AWS IoT SiteWise 시점부터의 AWS 관리형 정책 업데이트에 대한 세부 정보를 볼 수 있습니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 AWS IoT SiteWise 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AWSServiceRoleForIoTSiteWise - 기존 정책 업데이트	AWS IoT SiteWise 이제 데터베이스에 대해 메타데이터 검색 쿼리를 AWS IoT TwinMaker 실행할 수 있습니다.	2023년 11월 6일
AWSIoTSiteWiseReadOnlyAccess - 기존 정책 업데이트	AWS IoT SiteWise 에 배치 읽기 작업을 수행할 수 BatchGet* 있는 새 정책 접두사가 추가되었습니다.	2022년 9월 16일
AWSIoTSiteWiseReadOnlyAccess - 새 정책	AWS IoT SiteWise 는에 대한 읽기 전용 액세스 권한을 부여하는 새 정책을 추가했습니다 AWS IoT SiteWise.	2021년 11월 24일

변경 사항	설명	날짜
AWS IoT SiteWise 에서 변경 내용 추적 시작	AWS IoT SiteWise 가 AWS 관리형 정책에 대한 변경 내용 추적을 시작했습니다.	2021년 11월 24일

에 대한 서비스 연결 역할 사용 AWS IoT SiteWise

AWS IoT SiteWise 는 AWS Identity and Access Management (IAM) [서비스 연결 역할](#)을 사용합니다. 서비스 연결 역할은 직접 연결된 고유한 유형의 IAM 역할입니다 AWS IoT SiteWise. 서비스 연결 역할은에서 사전 정의 AWS IoT SiteWise 하며 서비스가 사용자를 대신하여 다른 AWS 서비스를 호출하는데 필요한 모든 권한을 포함합니다.

서비스 연결 역할은 필요한 모든 권한을 AWS IoT SiteWise 자동으로 포함하여 구성을 간소화합니다. 는 서비스 연결 역할의 권한을 AWS IoT SiteWise 정의하며, 달리 정의되지 않은 한 만 해당 역할을 수 임 AWS IoT SiteWise 할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함됩니다. 또한 해당 권한 정책은 다른 어떤 IAM 엔터티에도 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 AWS IoT SiteWise 리소스에 대한 액세스 권한을 실수로 제거할 수 없기 때문에 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 다른 서비스에 대한 자세한 내용은 [AWS IAM으로 작업하는 서비스를](#) 참조하고 서비스 연결 역할 열에서 예인 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 링크가 있는 예를 선택합니다.

주제

- [AWS IoT SiteWise에 대한 서비스 링크 역할 권한](#)
- [AWS IoT SiteWise용 서비스 연결 역할 생성](#)
- [AWS IoT SiteWise에 대한 서비스 연결 역할 업데이트](#)
- [AWS IoT SiteWise에 대한 서비스 연결 역할 삭제](#)
- [AWS IoT SiteWise 서비스 연결 역할에 지원되는 리전](#)
- [에 대한 서비스 역할 사용 AWS IoT SiteWise Monitor](#)

AWS IoT SiteWise에 대한 서비스 링크 역할 권한

AWS IoT SiteWise 는 AWSServiceRoleForIoTSiteWise라는 서비스 연결 역할을 사용합니다. AWS IoT SiteWise 는이 서비스 연결 역할을 사용하여 SiteWise Edge 게이트웨이(에서 실행됨 AWS IoT Greengrass)를 배포하고 로깅을 수행합니다.

AWSServiceRoleForIoTSiteWise 서비스 연결 역할은 다음 권한이 있는 AWSServiceRoleForIoTSiteWise 정책을 사용합니다. 이 정책은 다음과 같습니다.

- AWS IoT SiteWise 가 SiteWise Edge 게이트웨이(에서 실행됨)를 배포하도록 허용합니다AWS IoT Greengrass.
- 가 로깅 AWS IoT SiteWise 을 수행하도록 허용합니다.
- 가 AWS IoT TwinMaker 데이터베이스에 대해 메타데이터 검색 쿼리를 실행 AWS IoT SiteWise 하도록 허용합니다.

AWSServiceRoleForIoTSiteWise에서 허용된 작업에 대한 자세한 내용은 [AWS IoT SiteWise에 대한AWS 관리형 정책](#)을 참조하세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSiteWiseReadGreenGrass",
      "Effect": "Allow",
      "Action": [
        "greengrass:GetAssociatedRole",
        "greengrass:GetCoreDefinition",
        "greengrass:GetCoreDefinitionVersion",
        "greengrass:GetGroup",
        "greengrass:GetGroupVersion"
      ],
      "Resource": "*"
    },
    {
      "Sid": "AllowSiteWiseAccessLogGroup",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:DescribeLogGroups"
      ],
      "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*"
    }
  ]
}
```

```

},
{
  "Sid": "AllowSiteWiseAccessLog",
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogStream",
    "logs:DescribeLogStreams",
    "logs:PutLogEvents"
  ],
  "Resource": "arn:aws:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
},
{
  "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
  "Effect": "Allow",
  "Action": [
    "iottwinmaker:GetWorkspace",
    "iottwinmaker:ExecuteQuery"
  ],
  "Resource": "arn:aws:iottwinmaker:*:*:workspace/*",
  "Condition": {
    "ForAnyValue:StringEquals": {
      "iottwinmaker:linkedServices": [
        "IOTSITewise"
      ]
    }
  }
}
]
}

```

로그를 사용하여 SiteWise Edge 게이트웨이를 모니터링하고 문제를 해결할 수 있습니다. 자세한 내용은 [SiteWise Edge 게이트웨이 로그 모니터링](#) 단원을 참조하십시오.

IAM 엔티티(예: 사용자, 그룹 또는 역할)가 서비스 연결 역할을 생성, 편집 또는 삭제할 수 있도록 하려면 먼저 권한을 구성합니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 권한](#)을 참조하세요.

AWS IoT SiteWise용 서비스 연결 역할 생성

AWS IoT SiteWise에서는 특정 작업을 수행하고 사용자를 대신하여 리소스에 액세스하려면 서비스 연결 역할이 필요합니다. 서비스 연결 역할은 직접 연결된 고유한 유형의 AWS Identity and Access Management(IAM) 역할입니다 AWS IoT SiteWise. 이 역할을 생성하면 데이터 스토리지 또는 디바이스 통신용 Amazon S3와 같이 작업에 필요한 다른 AWS 서비스 및 리소스에 액세스하는 AWS IoT 데 AWS IoT SiteWise 필요한 권한을 부여합니다.

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS IoT SiteWise 콘솔에서 다음 작업을 수행하면 서비스 연결 역할을 AWS IoT SiteWise 생성합니다.

- Greengrass V1 게이트웨이를 생성합니다.
- 로깅 옵션을 구성합니다.
- 쿼리 실행 배너에서 옵트인 버튼을 선택합니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. AWS IoT SiteWise 콘솔에서 작업을 수행하면 서비스 연결 역할을 다시 AWS IoT SiteWise 생성합니다.

또한 IAM 콘솔 또는 API를 사용하여 AWS IoT SiteWise를 위한 서비스 연결 역할을 만들 수도 있습니다.

- IAM 콘솔에서 이 작업을 수행하려면 `AWSServiceRoleForIoTSiteWise` 정책 및 `iotsitewise.amazonaws.com`와의 신뢰 관계를 사용하여 역할을 생성하세요.
- AWS CLI 또는 IAM API를 사용하여 이렇게 하려면 `arn:aws:iam::aws:policy/aws-service-role/AWSServiceRoleForIoTSiteWise` 정책 및 `iotsitewise.amazonaws.com`와의 신뢰 관계를 사용하여 역할을 생성합니다.

자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 생성](#)을 참조하세요.

이 서비스 연결 역할을 삭제하면 동일한 프로세스를 사용하여 역할을 다시 생성할 수 있습니다.

AWS IoT SiteWise에 대한 서비스 연결 역할 업데이트

AWS IoT SiteWise에서는 `AWSServiceRoleForIoTSiteWise` 서비스 연결 역할을 편집할 수 없습니다. 서비스 연결 역할을 생성한 후에는 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [서비스 연결 역할 업데이트](#)를 참조하세요.

AWS IoT SiteWise에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스를 더 이상 사용하지 않는 경우 연결된 역할을 삭제하는 것이 좋습니다. 이는 모니터링되거나 유지 관리되지 않는 비활성 엔터티가 발생하지 않도록 하기 위한 것입니다. 단, 서비스 링크 역할에 대한 리소스를 먼저 정리해야 수동으로 삭제할 수 있습니다.

Note

리소스를 삭제하려고 할 때 AWS IoT SiteWise 서비스가 역할을 사용하는 경우 삭제에 실패할 수 있습니다. 그 경우 몇 분 동안 기다렸다가 다시 시도하세요.

AWSServiceRoleForIoTSiteWise에서 사용하는 AWS IoT SiteWise 리소스를 삭제하려면

1. 에 대한 로깅을 비활성화합니다 AWS IoT SiteWise. 자세한 내용은 [로깅 수준 변경](#) 섹션을 참조하세요.
2. 활성 SiteWise Edge 게이트웨이를 삭제합니다.

IAM을 사용하여 수동으로 서비스 연결 역할을 삭제하려면 다음을 수행하세요.

IAM 콘솔 AWS CLI, 또는 AWS API를 사용하여 AWSServiceRoleForIoTSiteWise 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 [역할 또는 인스턴스 프로파일 삭제](#)를 참조하세요.

AWS IoT SiteWise 서비스 연결 역할에 지원되는 리전

AWS IoT SiteWise 는 서비스를 사용할 수 있는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용은 [AWS IoT SiteWise 엔드포인트 및 할당량](#)을 참조하세요.

에 대한 서비스 역할 사용 AWS IoT SiteWise Monitor

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 [IAM 역할](#)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 [Create a role to delegate permissions to an AWS 서비스](#)를 참조하세요.

페더레이션 SiteWise Monitor 포털 사용자가 AWS IoT SiteWise 및 AWS IAM Identity Center 리소스에 액세스할 수 있도록 하려면 생성한 각 포털에 서비스 역할을 연결해야 합니다. 이 서비스 역할은 SiteWise Monitor를 신뢰할 수 있는 엔터티로 지정하고 [AWSIoTSiteWiseMonitorPortalAccess](#) 관리형 정책을 포함하거나 [동등한 권한](#)을 정의해야 합니다. 이 정책은에서 유지 AWS 관리하며 SiteWise Monitor가 및 IAM Identity Center 리소스에 액세스하는 AWS IoT SiteWise 데 사용하는 권한 세트를 정의합니다.

SiteWise Monitor 포털을 생성할 때 해당 포털의 사용자가 사용자의 AWS IoT SiteWise 및 IAM Identity Center 리소스에 액세스할 수 있도록 허용하는 역할을 선택해야 합니다. AWS IoT SiteWise 콘솔에서 역할을 생성하고 구성할 수 있습니다. 나중에 IAM에서 역할을 편집할 수 있습니다. 이 역할에서 필요한

권한을 제거하거나 역할을 삭제하면 포털 사용자가 SiteWise Monitor 포털을 사용하는 데 문제가 발생합니다.

Note

2020년 4월 29일 이전에 생성된 포털에는 서비스 역할이 필요 없었습니다. 이 날짜 이전에 포털을 만든 경우, 계속 사용하려면 서비스 역할을 연결해야 합니다. 이렇게 하려면 [AWS IoT SiteWise 콘솔](#)의 포털 페이지로 이동한 다음 IAM 역할을 사용하도록 모든 포털 마이그레이션을 선택합니다.

다음 섹션에서는 AWS Management Console 또는 AWS Command Line Interface에서 SiteWise Monitor 서비스 역할을 생성 및 관리하는 방법에 대해 설명합니다.

목차

- [SiteWise Monitor\(클래식\)에 대한 서비스 역할 권한](#)
- [SiteWise Monitor\(AI 인식\)에 대한 서비스 역할 권한](#)
- [SiteWise Monitor 서비스 역할 관리\(콘솔\)](#)
 - [포털의 서비스 역할 찾기\(콘솔\)](#)
 - [SiteWise Monitor 서비스 역할 생성](#)
 - [SiteWise Monitor 서비스 역할 생성\(IAM 콘솔\)](#)
 - [포털의 서비스 역할 변경\(콘솔\)](#)
- [SiteWise Monitor 서비스 역할 관리\(CLI\)](#)
 - [포털의 서비스 역할 찾기\(CLI\)](#)
 - [SiteWise Monitor 서비스 역할 생성\(CLI\)](#)
- [SiteWiseMonitor에서 AWSIoTSiteWiseMonitorServiceRole로 업데이트](#)

SiteWise Monitor(클래식)에 대한 서비스 역할 권한

포털을 생성할 때 이름이 AWSIoTSiteWiseMonitorServiceRole로 시작하는 역할을 생성할 수 AWS IoT SiteWise 있습니다. 페더레이션 SiteWise Monitor 사용자는 이 역할을 사용하여 포털 구성, 자산, 자산 데이터, IAM Identity Center 구성에 액세스할 수 있습니다.

이 역할은 역할을 맡기 위해 다음 서비스를 신뢰합니다.

- `monitor.iotsitewise.amazonaws.com`

역할은 `AWSIoTSiteWiseMonitorServicePortalPolicy`로 시작하는 다음 권한 정책을 사용하여 SiteWise Monitor 사용자가 계정의 리소스에 대한 작업을 완료할 수 있도록 허용합니다. [AWSIoTSiteWiseMonitorPortalAccess](#) 관리형 정책은 이와 동등한 권한을 정의해야 합니다.

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

```

경보에 필요한 사용 권한에 대한 자세한 내용은 [에서 이벤트 경보에 대한 권한 설정 AWS IoT SiteWise](#)을 참조하세요.

포털 사용자가 로그인하면 SiteWise Monitor에서는 서비스 역할과 해당 사용자의 액세스 정책의 공통점을 기반으로 [세션 정책](#)을 생성합니다. 액세스 정책은 포털 및 프로젝트에 대한 ID의 액세스 수준을 정의합니다. 포털 권한 및 액세스 정책에 대한 자세한 내용은 [SiteWise Monitor 포털 관리](#) 및 [CreateAccessPolicy](#)를 참조하세요.

SiteWise Monitor(AI 인식)에 대한 서비스 역할 권한

포털을 생성할 때 이름이 IoTSiteWisePortalRole로 시작하는 역할을 생성할 수 AWS IoT SiteWise 있습니다. 페더레이션 SiteWise Monitor 사용자는 이 역할을 사용하여 포털 구성, 자산, 자산 데이터, IAM Identity Center 구성에 액세스할 수 있습니다.

Warning

SiteWise Monitor(AI 인식)에서는 프로젝트 소유자 및 프로젝트 뷰어 역할이 지원되지 않습니다.

이 역할은 역할을 맡기 위해 다음 서비스를 신뢰합니다.

- `monitor.iotsitewise.amazonaws.com`

역할은 `IoTSiteWiseAPIPortalAccessPolicy`로 시작하는 다음 권한 정책을 사용하여 SiteWise Monitor 사용자가 계정의 리소스에 대한 작업을 완료할 수 있도록 허용합니다.

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

```

포털 사용자가 로그인하면 SiteWise Monitor에서는 서비스 역할과 해당 사용자의 액세스 정책의 공통 점을 기반으로 [세션 정책](#)을 생성합니다.

SiteWise Monitor 서비스 역할 관리(콘솔)

를 사용하면 포털에 대한 SiteWise Monitor 서비스 역할을 AWS IoT SiteWise Console 쉽게 관리할 수 있습니다. 포털을 생성하면 콘솔은 연결에 적합한 기존 역할을 확인합니다. 사용 가능한 역할이 없으면 콘솔이 사용자를 대신하여 서비스 역할을 생성 및 구성할 수 있습니다. 자세한 내용은 [SiteWise Monitor에서 포털 생성](#) 단원을 참조하십시오.

주제

- [포털의 서비스 역할 찾기\(콘솔\)](#)
- [SiteWise Monitor 서비스 역할 생성](#)
- [SiteWise Monitor 서비스 역할 생성\(IAM 콘솔\)](#)
- [포털의 서비스 역할 변경\(콘솔\)](#)

포털의 서비스 역할 찾기(콘솔)

다음 단계에 따라 SiteWise Monitor 포털에 연결된 서비스 역할을 찾을 수 있습니다.

포털의 서비스 역할을 찾으려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 포털을 선택합니다.

3. 서비스 역할을 찾으려는 포털을 선택합니다.

포털에 연결된 역할이 권한, 서비스 역할 아래에 나타납니다.

SiteWise Monitor 서비스 역할 생성

SiteWise Monitor 포털을 만들 때 포털에 대한 새 서비스 역할을 만들 수 있습니다. 자세한 내용은 [SiteWise Monitor에서 포털 생성](#) 단원을 참조하십시오.

AWS IoT SiteWise 콘솔에서 기존 포털에 대한 서비스 역할을 생성할 수도 있습니다. 이는 포털의 기존 서비스 역할을 대체합니다.

기존 포털에 대한 서비스 역할을 만들려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 포털을 선택합니다.
3. 새 서비스 역할을 만들려는 포털을 선택합니다.
4. 포털 세부 정보에서 편집을 선택합니다.
5. 권한 아래에서 새 서비스 역할 생성 및 사용을 선택합니다.
6. 새 역할의 이름을 입력합니다.
7. 저장을 선택합니다.

SiteWise Monitor 서비스 역할 생성(IAM 콘솔)

IAM 콘솔의 서비스 역할 템플릿에서 서비스 역할을 생성할 수 있습니다. 이 역할 템플릿에는 [AWSIoTSiteWiseMonitorPortalAccess](#) 관리형 정책이 포함되어 있으며 SiteWise Monitor를 신뢰할 수 있는 엔티티로 지정합니다.

포털 서비스 역할 템플릿에서 서비스 역할을 만들려면

1. [IAM 콘솔](#)로 이동합니다.
2. 탐색 창에서 Roles를 선택합니다.
3. 역할 생성(Create role)을 선택합니다.
4. 사용 사례 선택에서 IoT SiteWise를 선택합니다.
5. 사용 사례 선택에서 IoT SiteWise Monitor - Portal를 선택합니다.
6. 다음: 권한을 선택합니다.

7. 다음: 태그를 선택합니다.
8. 다음: 검토를 선택합니다.
9. 역할 이름에 새 서비스 역할의 이름을 입력합니다.
10. 역할 생성을 선택합니다.

포털의 서비스 역할 변경(콘솔)

다음 절차에 따라 포털에 대해 다른 SiteWise Monitor 서비스 역할을 선택할 수 있습니다.

포털의 서비스 역할을 변경하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 포털을 선택합니다.
3. 서비스 역할을 변경할 포털을 선택합니다.
4. 포털 세부 정보에서 편집을 선택합니다.
5. 권한에서 기존 역할 사용을 선택합니다.
6. 이 포털에 연결할 기존 역할을 선택합니다.
7. 저장을 선택합니다.

SiteWise Monitor 서비스 역할 관리(CLI)

다음 포털 서비스 역할 관리 작업에 AWS CLI 를 사용할 수 있습니다.

주제

- [포털의 서비스 역할 찾기\(CLI\)](#)
- [SiteWise Monitor 서비스 역할 생성\(CLI\)](#)

포털의 서비스 역할 찾기(CLI)

SiteWise Monitor 포털에 연결된 서비스 역할을 찾으려면 다음 명령을 실행하여 현재 리전의 모든 포털을 나열합니다.

```
aws iotsitewise list-portals
```

이 작업은 포털 요약 정보가 포함된 응답을 다음 형식으로 반환합니다.

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

포털의 ID를 알고 있는 경우, [DescribePortal](#) 작업을 사용하여 포털의 역할을 찾을 수도 있습니다.

SiteWise Monitor 서비스 역할 생성(CLI)

다음 단계에 따라 새 SiteWise Monitor 서비스 역할을 만들 수 있습니다.

SiteWise Monitor 서비스 역할을 만들려면

1. 이 역할을 수임하도록 허용하는 SiteWise Monitor 신뢰 정책으로 역할을 생성합니다. 이 예제에서는 JSON 문자열에 저장된 신뢰 정책으로 **MySiteWiseMonitorPortalRole**이라는 역할을 만듭니다.

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. 출력의 역할 메타데이터에서 역할 ARN을 복사합니다. 포털을 생성할 때 이 ARN을 사용하여 역할을 포털에 연결합니다. 포털 생성에 대한 자세한 내용은 AWS IoT SiteWise API 참조의 [CreatePortal](#)을 참조하세요.
3. a. SiteWise Monitor(Classic)의 경우 - AWSIoTSiteWiseMonitorPortalAccess 정책을 역할에 연결하거나 동등한 권한을 정의하는 정책을 연결합니다.

```
aws iam attach-role-policy --role-name MySiteWiseMonitorPortalRole --policy-arn arn:aws:iam::aws:policy/service-role/AWSIoTSiteWiseMonitorPortalAccess
```

- b. SiteWise Monitor(AI 인식)의 경우 - IoTSiteWiseAIPortalAccessPolicy 정책을 역할에 연결하거나 동등한 권한을 정의하는 정책을 연결합니다. 예를 들어 포털 액세스 권한이 있는 정책을 생성합니다. 다음 예제에서는 라는 정책을 생성합니다 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": "*"
}
]
}'

```

서비스 역할을 기존 포털에 연결하려면 다음을 수행하세요.

1. 포털의 기존 세부 정보를 검색하려면 다음 명령을 실행합니다. *portal-id*를 해당 포털의 ID로 바꿉니다.

```
aws iotsitewise describe-portal --portal-id portal-id
```

이 작업은 포털의 세부 정보가 포함된 응답을 다음 형식으로 반환합니다.

```
{
  "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. 서비스 역할을 포털에 연결하려면 다음 명령을 실행합니다. *role-arn*을 서비스 역할 ARN으로 바꾸고 나머지 파라미터를 해당 포털의 기존 값으로 바꿉니다.

```
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
```

SiteWiseMonitor에서 AWSIoTSiteWiseMonitorServiceRole로 업데이트

이 서비스가 변경 사항을 추적하기 시작한 시점부터 SiteWise Monitor용 AWSIoTSiteWiseMonitorServiceRole 업데이트에 대한 세부 정보를 볼 수 있습니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 AWS IoT SiteWise 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AWSIoTSiteWiseMonitorPortalAccess – 업데이트된 정책	AWS IoT SiteWise 에서 경보 기능에 대한 AWSIoTSiteWiseMonitorPortalAccess 관리형 정책을 업데이트했습니다.	2021년 5월 27일
AWS IoT SiteWise 에서 변경 내용 추적 시작	AWS IoT SiteWise 가 서비스 역할에 대한 변경 내용 추적을 시작했습니다.	2020년 12월 15일

에서 이벤트 경보에 대한 권한 설정 AWS IoT SiteWise

AWS IoT Events 경보 모델을 사용하여 AWS IoT SiteWise 자산 속성을 모니터링하는 경우 다음 IAM 권한이 있어야 합니다.

- 가 데이터를 전송할 수 있도록 허용하는 AWS IoT Events 서비스 역할 AWS IoT Events 입니다 AWS IoT SiteWise. 자세한 내용은 AWS IoT Events 개발자 안내서에서 [AWS IoT Events의 자격 증명 및 액세스 관리](#)를 참조하세요.
- `iotsitewise:DescribeAssetModel` 및 AWS IoT SiteWise 작업 권한이 있어야 합니다 `iotsitewise:UpdateAssetModelPropertyRouting`. 이러한 권한을 통해 AWS IoT SiteWise 는 자산 속성 값을 AWS IoT Events 경보 모델로 전송할 수 있습니다.

자세한 내용은 IAM 사용 설명서의 [리소스 기반 정책](#)을 참조하세요.

필요한 작업 권한

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 위탁자가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다. JSON 정책의 Action 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다.

AWS IoT Events 경보 모델을 정의하기 전에가 자산 속성 값을 경보 모델 AWS IoT SiteWise 로 전송하도록 허용하는 다음 권한을 부여해야 합니다.

- `iotsitewise:DescribeAssetModel`, `iotsitewise:ListAssetModels`- 자산 속성이 존재하는지 AWS IoT Events 확인할 수 있습니다.

- `iotsitewise:UpdateAssetModelPropertyRouting` - AWS IoT SiteWise 가 데이터를 전송할 수 있는 구독 AWS IoT SiteWise 을 자동으로 생성할 수 있습니다 AWS IoT Events.

AWS IoT SiteWise 지원되는 작업에 대한 자세한 내용은 서비스 승인 참조의에서 [정의한 작업을 AWS IoT SiteWise](#) 참조하세요.

Example 권한 정책 예 1

다음 정책은 AWS IoT SiteWise 가 모든 AWS IoT Events 경보 모델에 자산 속성 값을 전송하도록 허용합니다.

```
{
  "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 권한 정책 예 2

다음 정책은 AWS IoT SiteWise 가 지정된 자산 속성의 값을 지정된 AWS IoT Events 경보 모델로 전송하도록 허용합니다.

```
{
  "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"
        }
      }
    }
  ]
}

```

(선택 사항) ListInputRoutings 권한

자산 모델을 업데이트하거나 삭제할 때의 AWS IoT Events 경보 모델이 자산 모델과 연결된 자산 속성을 모니터링하는지 확인할 AWS IoT SiteWise 수 있습니다. 이렇게 하면 AWS IoT Events 경보가 현재 사용 중인 자산 속성을 삭제할 수 없습니다. 에서이 기능을 활성화하려면 `iotevents:ListInputRoutings` 권한이 있어야 AWS IoT SiteWise합니다. 이 권한을 통해 AWS IoT SiteWise 는에서 지원하는 [ListInputRoutings](#) API 작업을 호출할 수 있습니다 AWS IoT Events.

Note

ListInputRoutings 권한을 추가하는 것이 좋습니다.

Example 권한 정책 예

다음 정책은 자산 모델을 업데이트 및 삭제하고에서 ListInputRoutings API를 사용할 수 있도록 허용합니다 AWS IoT SiteWise.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:UpdateAssetModel",
        "iotsitewise>DeleteAssetModel",
        "iotevents:ListInputRoutings"
      ],
      "Resource": "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/*"
    }
  ]
}
```

SiteWise Monitor에 필요한 권한

SiteWise Monitor 포털에서 경보 기능을 사용하려면 [SiteWise Monitor 서비스 역할](#)을 다음 정책으로 업데이트해야 합니다.

```
{
  "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"
            ]
        }
    }
}
}

```

```
]
}
```

의 교차 서비스 혼동된 대리자 방지 AWS IoT SiteWise

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접적으로 호출하는 서비스)가 다른 서비스(직접적으로 호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 [aws:SourceArn](#) 및 [aws:SourceAccount](#) 전역 조건 컨텍스트 키를 사용하여 리소스에 다른 서비스를 AWS IoT SiteWise 제공하는 권한을 제한하는 것이 좋습니다. 만약 [aws:SourceArn](#) 값에 Amazon S3 버킷 Amazon 리소스 이름(ARN)과 같은 계정 ID가 포함되어 있지 않은 경우 권한을 제한하려면 두 전역 조건 컨텍스트 키를 모두 사용해야 합니다. 두 전역 조건 컨텍스트 키와 계정을 포함한 [aws:SourceArn](#) 값을 모두 사용하는 경우, [aws:SourceAccount](#) 값 및 [aws:SourceArn](#) 값의 계정은 동일한 정책 명령문에서 사용할 경우 반드시 동일한 계정 ID를 사용해야 합니다.

- 하나의 리소스만 교차 서비스 액세스와 연결되도록 허용하려는 경우 [aws:SourceArn](#)를 사용하세요.
- 해당 계정의 모든 리소스가 교차 서비스 사용과 연결되도록 허용하려는 경우 [aws:SourceAccount](#)를 사용하세요.

의 값은 `sts:AssumeRole` 요청과 연결된 AWS IoT SiteWise 고객 리소스여야 [aws:SourceArn](#) 합니다.

혼동된 대리인 문제로부터 보호하는 가장 효과적인 방법은 리소스의 전체 ARN이 포함된 [aws:SourceArn](#) 글로벌 조건 컨텍스트 키를 사용하는 것입니다. 리소스의 전체 ARN을 모를 경우 또는 여러 리소스를 지정하는 경우, ARN의 알 수 없는 부분에 대해 와일드카드(*)를 포함한 [aws:SourceArn](#) 글로벌 조건 컨텍스트 키를 사용합니다. 예를 들어 `arn:aws:servicename:*:123456789012:*`입니다.

Example – 혼동된 대리자 방지

다음 예제에서는 [aws:SourceArn](#) 및 [aws:SourceAccount](#) 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 AWS IoT SiteWise 방지하는 방법을 보여줍니다.

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

AWS IoT SiteWise 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 AWS IoT SiteWise 및 AWS Identity and Access Management (IAM)로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- [에서 작업을 수행할 권한이 없음 AWS IoT SiteWise](#)
- [iam:PassRole 작업을 수행할 권한이 없음](#)
- [내 AWS 계정 외부의 사람이 내 AWS IoT SiteWise 리소스에 액세스하도록 허용하고 싶습니다.](#)

에서 작업을 수행할 권한이 없음 AWS IoT SiteWise

에서 작업을 수행할 권한이 없다는 AWS Management Console 메시지가 표시되면 관리자에게 문의하여 도움을 받아야 합니다. 관리자는 사용자 이름과 비밀번호를 제공한 사람입니다.

다음 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 자산의 세부 정보를 보려고 하지만 iotsitewise:DescribeAsset 권한이 없는 경우에 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
iotsitewise:DescribeAsset on resource: a1b2c3d4-5678-90ab-cdef-2222EXAMPLE
```

이 경우 Mateo는 `iotsitewise:DescribeAsset` 작업을 사용하여 `a1b2c3d4-5678-90ab-cdef-2222EXAMPLE` ID가 있는 자산 리소스에 액세스할 수 있도록 정책을 업데이트할 것을 관리자 에게 요청합니다.

iam:PassRole 작업을 수행할 권한이 없음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 AWS IoT SiteWise에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 AWS IoT SiteWise에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

이 경우, Mary가 `iam:PassRole` 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사람이 내 AWS IoT SiteWise 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- 에서 이러한 기능을 AWS IoT SiteWise 지원하는지 여부를 알아보려면 섹션을 참조하세요 [AWS IoT SiteWise에서 IAM을 사용하는 방법](#).

- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 [IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요](#).
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [타사가 AWS 계정 소유한에 대한 액세스 권한 제공을](#) AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [외부에서 인증된 사용자에게 액세스 권한 제공\(ID 페더레이션\)](#)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 [IAM의 크로스 계정 리소스 액세스](#)를 참조하세요.

에 대한 규정 준수 검증 AWS IoT SiteWise

AWS IoT SiteWise 는 규정 AWS 준수 프로그램의 범위에 속하지 않습니다.

특정 규정 준수 프로그램 범위의 AWS 서비스 목록은 규정 준수 프로그램 [AWS 제공 범위 내 서비스 규정 준수 프로그램](#). 일반 정보는 [AWS 규정 준수 프로그램](#).

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 [에서 보고서 다운로드 AWS Artifact](#)에서 .

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS IoT SiteWise 는 규정 준수를 지원할 다음과 같은 리소스를 AWS 제공합니다.

- [보안 및 규정 준수 킷 스타트 가이드](#) -이 배포 가이드에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수 중심 기준 환경을 배포하기 위한 단계를 제공합니다 AWS.
- [HIPAA 보안 및 규정 준수를 위한 설계 백서](#) -이 백서에서는 기업이 AWS 를 사용하여 HIPAA 준수 애플리케이션을 생성하는 방법을 설명합니다.
- [AWS 규정 준수 리소스](#) -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- AWS Config 개발자 안내서의 [규칙을 사용하여 리소스 평가](#) -이 AWS Config 서비스는 리소스 구성 이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- [AWS Security Hub](#) -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.
- [산업용 IoT 솔루션을 위한 10가지 보안 황금률](#) — 이 블로그 게시물에서는 산업용 제어 시스템(ICS), 산업용 사물 인터넷(IIoT) 및 클라우드 환경을 보호하는 데 도움이 되는 10가지 황금률을 소개합니다.
- [Security Best Practices for Manufacturing OT](#) -이 백서에서는 AWS 클라우드용 온프레미스 하이브리드 제조 워크로드를 설계, 배포 및 설계하기 위한 보안 모범 사례를 설명합니다.

의 복원력 AWS IoT SiteWise

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며, 이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS IoT SiteWise 는 완전 관리형이며 Amazon S3 및 Amazon EC2와 같은 가용성과 내구성이 뛰어난 AWS 서비스를 사용합니다. 가용 영역 중단 시 가용성을 보장하기 위해서는 여러 가용 영역에서 AWS IoT SiteWise 작동합니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 [AWS 글로벌 인프라](#)를 참조하세요.

AWS 글로벌 인프라 외에도 데이터 복원성과 백업 요구 사항을 지원하는 몇 가지 기능을 AWS IoT SiteWise 제공합니다.

- MQTT 메시지를 AWS IoT Core 통해 속성 값 업데이트를 게시한 다음 해당 데이터에 따라 작동하도록 규칙을 구성할 수 있습니다. 이 기능을 사용하면 Amazon S3 및 Amazon DynamoDB와 같은 다른 AWS 서비스에 데이터를 백업할 수 있습니다. 자세한 내용은 [다른 AWS 서비스와 상호 작용 및 자산 속성 알림을 사용하여 Amazon S3로 데이터 내보내기](#) 섹션을 참조하세요.
- API를 AWS IoT SiteWise Get* 사용하여 과거 자산 속성 데이터를 검색하고 백업할 수 있습니다. APIs 자세한 내용은 [에서 과거 자산 속성 값 쿼리 AWS IoT SiteWise](#) 섹션을 참조하세요.
- APIs를 AWS IoT SiteWise Describe* 사용하여 자산 및 모델과 같은 리소스에 대한 정의를 검색할 수 있습니다. 이러한 정의를 백업해 두었다가 나중에 이것으로 리소스를 다시 만들 수 있습니다. 자세한 내용은 [AWS IoT SiteWise API 참조](#)를 참조하세요.

의 인프라 보안 AWS IoT SiteWise

관리형 서비스인 AWS 글로벌 네트워크 보안으로 보호 AWS IoT SiteWise 됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 [AWS 클라우드 보안을](#) 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 [인프라 보호](#)를 참조하세요.

AWS 에서 게시한 API 호출을 사용하여 네트워크를 AWS IoT SiteWise 통해 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.

- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 보안 암호 액세스 키를 사용하여 서명해야 합니다. 또는 [AWS Security Token Service](#)(AWS STS)를 사용하여 임시 자격 증명을 생성하여 요청에 서명할 수 있습니다.

에서 실행되는 SiteWise Edge 게이트웨이는 X.509 인증서 및 암호화 키를 AWS IoT Greengrass 사용하여 AWS 클라우드에 연결하고 인증합니다. 자세한 내용은 AWS IoT Greengrass Version 1 개발자 안내서의 [AWS IoT Greengrass에 대한 디바이스 인증 및 권한 부여](#)를 참조하세요.

의 구성 및 취약성 분석 AWS IoT SiteWise

IoT 플릿은 다양한 기능을 수행하고 장기적으로 사용되며 지리적으로 분산된 다수의 디바이스로 구성될 수 있습니다. 이러한 특성으로 인해 플릿 설정이 복잡해지고 오류가 발생하기 쉬워집니다. 디바이스의 처리 능력, 메모리 및 스토리지는 일반적으로 제한적이므로 암호화 및 기타 보안 조치를 항상 지원할 수는 없습니다. 또한 장치에서 알려진 취약성이 있는 소프트웨어를 사용하는 경우도 있습니다. 이러한 요소로 인해 IoT 플릿이 해커의 매력적인 대상이 되며, 디바이스 플릿을 지속적으로 보호하기 어렵게 됩니다.

AWS IoT Device Defender 는 보안 문제와 모범 사례와의 편차를 식별하는 도구를 제공하여 이러한 문제를 해결합니다. AWS IoT Device Defender 를 사용하여 연결된 디바이스를 분석, 감사 및 모니터링하여 비정상적인 동작을 감지하고 보안 위험을 완화합니다. AWS IoT Device Defender 는 디바이스 플릿을 감사하여 보안 모범 사례를 준수하고 디바이스에서 비정상적인 동작을 감지할 수 있도록 합니다. 이를 통해 AWS IoT 디바이스 플릿에 일관된 보안 정책을 적용하고 디바이스가 손상되면 신속하게 대응할 수 있습니다. 자세한 내용은 AWS IoT Device Defender 개발자 가이드의 [AWS IoT Device Defender이란 무엇입니까?](#)를 참조하세요.

SiteWise Edge 게이트웨이를 사용하여 서비스에 데이터를 수집하는 경우 SiteWise Edge 게이트웨이 환경을 구성하고 유지 관리할 책임은 사용자에게 있습니다. 이 책임에는 SiteWise Edge 게이트웨이의 시스템 소프트웨어, AWS IoT Greengrass 소프트웨어 및 AWS IoT SiteWise 커넥터의 최신 버전으로 업그레이드하는 것이 포함됩니다. 자세한 내용은 AWS IoT Greengrass Version 1 개발자 안내서의 [AWS IoT Greengrass 코어 구성](#) 및 [섹션을 참조하세요](#) [SiteWise Edge 게이트웨이 관리](#).

용 VPC 엔드포인트 AWS IoT SiteWise

인터페이스 VPC 엔드포인트는 Virtual Private Cloud(VPC)와 간에 프라이빗 연결을 설정합니다 AWS IoT SiteWise.는 인터페이스 엔드포인트에 [AWS PrivateLink](#) AWS IoT SiteWise 전원을 공급하여 API 작업에 대한 프라이빗 액세스를 활성화합니다. 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect의 필요성을 우회할 수 있습니다. VPC의 인스턴스는 AWS IoT SiteWise API 작업과 통신하는 데 퍼블릭 IP 주소를 필요로 하지 않습니다. VPC와 간의 트래픽 AWS IoT SiteWise 은 네트워크를 벗어나 AWS 지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 [탄력적 네트워크 인터페이스](#)로 표현됩니다.

에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 AWS PrivateLink 가이드의 [인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스 액세스를](#) AWS IoT SiteWise 검토합니다.

의 VPC 엔드포인트에 대한 API 작업 AWS IoT SiteWise

AWS IoT SiteWise 는 VPC에서 다음 AWS IoT SiteWise API 작업을 호출할 수 있도록 지원합니다.

- 모든 데이터 플레인 API 작업에 다음 엔드포인트를 사용하세요. *region*을 사용 중인 AWS 리전으로 대체합니다.

```
data.iotsitewise.region.amazonaws.com
```

데이터 영역 API 작업에는 다음이 포함됩니다.

- [BatchGetAssetPropertyValue](#)
- [BatchGetAssetPropertyValueHistory](#)
- [BatchPutAssetPropertyValue](#)
- [GetAssetPropertyAggregates](#)
- [GetAssetPropertyValue](#)
- [GetAssetPropertyValueHistory](#)
- [GetInterpolatedAssetPropertyValues](#)
- 자산 모델, 자산, SiteWise Edge 게이트웨이, 태그 및 계정 구성을 관리하는 데 사용하는 컨트롤 플레인 API 작업의 경우 다음 엔드포인트를 사용하세요. *region*를 사용자의 AWS 리전으로 교체합니다.

```
api.iotsitewise.region.amazonaws.com
```

지원되는 컨트롤 플레인 API 작업에는 다음이 포함됩니다.

- [AssociateAssets](#)
- [CreateAsset](#)
- [CreateAssetModel](#)
- [DeleteAsset](#)
- [DeleteAssetModel](#)
- [DeleteDashboard](#)
- [DescribeAsset](#)
- [DescribeAssetModel](#)
- [DescribeAssetProperty](#)
- [DescribeDashboard](#)
- [DescribeLoggingOptions](#)
- [DisassociateAssets](#)
- [ListAssetModels](#)
- [ListAssetRelationships](#)
- [ListAssets](#)
- [ListAssociatedAssets](#)
- [PutLoggingOptions](#)
- [UpdateAsset](#)
- [UpdateAssetModel](#)
- [UpdateAssetProperty](#)
- [CreateGateway](#)
- [DeleteGateway](#)
- [DescribeDefaultEncryptionConfiguration](#)
- [DescribeGateway](#)
- [DescribeGatewayCapabilityConfiguration](#)
- [DescribeStorageConfiguration](#)
- [ListGateways](#)
- [ListTagsForResource](#)
- [UpdateGateway](#)
- [UpdateGatewayCapabilityConfiguration](#)

- [PutDefaultEncryptionConfiguration](#)
- [PutStorageConfiguration](#)
- [TagResource](#)
- [UntagResource](#)

 Note

컨트롤 플레인 API 작업에 대한 인터페이스 VPC 엔드포인트는 현재 다음 SiteWise Monitor API 작업에 대한 직접 호출을 지원하지 않습니다.

- [BatchAssociateProjectAssets](#)
- [BatchDisassociateProjectAssets](#)
- [CreateAccessPolicy](#)
- [CreateDashboard](#)
- [CreatePortal](#)
- [CreateProject](#)
- [DeleteAccessPolicy](#)
- [DeletePortal](#)
- [DeleteProject](#)
- [DescribeAccessPolicy](#)
- [DescribePortal](#)
- [DescribeProject](#)
- [ListAccessPolicies](#)
- [ListDashboards](#)
- [ListPortals](#)
- [ListProjects](#)
- [ListProjectAssets](#)
- [UpdateAccessPolicy](#)
- [UpdateDashboard](#)
- [UpdatePortal](#)
- [UpdateProject](#)

AWS IoT SiteWise용 인터페이스 VPC 엔드포인트 생성

AWS IoT SiteWise 서비스에 대한 VPC 엔드포인트를 생성하려면 Amazon VPC 콘솔 또는 AWS Command Line Interface ()를 사용합니다. 자세한 내용은 AWS PrivateLink 가이드의 [인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스 액세스](#)를 참조하세요.

다음 서비스 이름 중 하나를 AWS IoT SiteWise 사용하여에 대한 VPC 엔드포인트를 생성합니다.

- 데이터 플레인 API 작업의 경우 다음 서비스 이름을 사용하세요.

```
com.amazonaws.region.iotsitewise.data
```

- 컨트롤 플레인 API 작업의 경우 다음 서비스 이름을 사용하세요.

```
com.amazonaws.region.iotsitewise.api
```

인터페이스 VPC 엔드포인트를 AWS IoT SiteWise 통한 액세스

인터페이스 엔드포인트를 생성하면 통신에 사용할 수 있는 엔드포인트별 DNS 호스트 이름이 생성됩니다. AWS IoT SiteWise. 프라이빗 DNS 옵션은 기본적으로 활성화되어 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 [프라이빗 호스팅 영역 사용](#)을 참조하세요.

엔드포인트에 대해 프라이빗 DNS를 활성화하면 다음 VPC 엔드포인트 중 하나를 AWS IoT SiteWise 통해에 API 요청을 할 수 있습니다.

- 데이터 플레인 API 작업의 경우, 다음 엔드포인트를 사용하세요. *region*을 사용 중인 AWS 리전으로 대체합니다.

```
data.iotsitewise.region.amazonaws.com
```

- 컨트롤 플레인 API 작업의 경우, 다음 엔드포인트를 사용하세요. *region*을 사용 중인 AWS 리전으로 대체합니다.

```
api.iotsitewise.region.amazonaws.com
```

엔드포인트에 대해 프라이빗 DNS를 비활성화하는 경우 엔드포인트를 AWS IoT SiteWise 통해에 액세스하려면 다음을 수행해야 합니다.

1. API 요청에서 VPC 엔드포인트 URL을 지정합니다.

- 데이터 플레인 API 작업의 경우 다음 엔드포인트 URL을 사용하세요. *vpc-endpoint-id*와 *#*을 VPC 엔드포인트 ID 및 리전으로 바꾸세요.

```
vpc-endpoint-id.data.iotsitewise.region.vpce.amazonaws.com
```

- 컨트롤 플레인 API 작업의 경우 다음 엔드포인트 URL을 사용하세요. *vpc-endpoint-id*와 *#*을 VPC 엔드포인트 ID 및 리전으로 바꾸세요.

```
vpc-endpoint-id.api.iotsitewise.region.vpce.amazonaws.com
```

2. 호스트 접두사 삽입을 비활성화합니다. AWS CLI 및 AWS SDKs는 각 API 작업을 호출할 때 서비스 엔드포인트 앞에 다양한 호스트 접두사를 추가합니다. 이 기능을 사용하면 VPC 엔드포인트를 AWS CLI 지정할 AWS IoT SiteWise 때 및 AWS SDKs가 유효하지 않은 URLs을 생성합니다.

Important

AWS CLI 또는 예서는 호스트 접두사 삽입을 비활성화할 수 없습니다 AWS Tools for PowerShell. 즉, 프라이빗 DNS를 비활성화하면 이러한 도구를 사용하여 VPC 엔드포인트를 AWS IoT SiteWise 통해 액세스할 수 없습니다. 프라이빗 DNS를 활성화하여 AWS CLI 또는를 사용하여 엔드포인트 AWS Tools for PowerShell 를 AWS IoT SiteWise 통해 액세스 합니다.

AWS SDKs

- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Go v2](#)
- [AWS SDK for Java](#)
- [AWS SDK for Java 2.x](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for .NET](#)
- [AWS SDK for PHP](#)
- [AWS SDK for Python \(Boto3\)](#)
- [AWS SDK for Ruby](#)

자세한 내용은 AWS PrivateLink 가이드의 [인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스 액세스](#)를 참조하세요.

에 대한 VPC 엔드포인트 정책 생성 AWS IoT SiteWise

AWS IoT SiteWise에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 있는 리소스.

자세한 내용은 Amazon VPC 사용 설명서의 [엔드포인트 정책을 사용하여 VPC 엔드포인트에 대한 액세스 제어](#)를 참조하세요.

예: AWS IoT SiteWise 작업에 대한 VPC 엔드포인트 정책

다음은에 대한 엔드포인트 정책의 예입니다 AWS IoT SiteWise. 엔드포인트에 연결되면이 정책은 지정된 자산의 AWS 계정 *123456789012iotsitewiseadmin*에 있는 사용자에게 나열된 AWS IoT SiteWise 작업에 대한 액세스 권한을 부여합니다.

```
{
  "Statement": [
    {
      "Action": [
        "iotsitewise:CreateAsset",
        "iotsitewise:ListGateways",
        "iotsitewise:ListTagsForResource"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:iotsitewise:us-west-2:123456789012:asset/a1b2c3d4-5678-90ab-cdef-3333EXAMPLE",
      "Principal": {
        "AWS": [
          "123456789012:user/iotsitewiseadmin"
        ]
      }
    ]
  ]
}
```

에 대한 보안 모범 사례 AWS IoT SiteWise

이 주제에는에 대한 보안 모범 사례가 포함되어 있습니다 AWS IoT SiteWise.

OPC UA 서버에서 인증 자격 증명 사용

OPC UA 서버에 연결할 때 인증 자격 증명을 요구합니다. 방법은 서버 설명서를 참조하세요. 그런 다음 SiteWise Edge 게이트웨이가 OPC UA 서버에 연결할 수 있도록 하기 위해 SiteWise Edge 게이트웨이에 서버 인증 암호를 추가합니다. 자세한 내용은 [SiteWise Edge에 대한 데이터 소스 인증 구성](#) 단원을 참조하십시오.

OPC UA 서버에 암호화된 통신 모드 사용

SiteWise Edge 게이트웨이에 OPC UA 소스를 구성할 때는 더 이상 사용되지 않는 암호화된 메시지 보안 모드를 선택하세요. 이는 OPC UA 서버에서 SiteWise Edge 게이트웨이로 이동하는 산업 데이터를 보호하는 데 도움이 됩니다. 자세한 내용은 [로컬 네트워크를 통해 전송 중인 데이터](#) 및 [SiteWise Edge에서 OPC UA 소스 설정](#) 섹션을 참조하세요.

구성 요소를 최신 상태로 유지

SiteWise Edge 게이트웨이를 사용하여 서비스에 데이터를 수집하는 경우 SiteWise Edge 게이트웨이 환경을 구성하고 유지 관리할 책임은 사용자에게 있습니다. 이러한 책임에는 게이트웨이의 시스템 소프트웨어, AWS IoT Greengrass 소프트웨어 및 커넥터를 최신 버전으로 업그레이드하는 것이 포함됩니다.

Note

AWS IoT SiteWise Edge 커넥터는 파일 시스템에 보안 암호를 저장합니다. 이러한 암호는 SiteWise Edge 게이트웨이 내에 캐시된 데이터를 볼 수 있는 사용자를 제어합니다. SiteWise Edge 게이트웨이를 실행하는 시스템의 디스크 또는 파일 시스템 암호화를 켜는 것이 좋습니다.

AWS IoT SiteWise 콘솔에서 구성 요소를 업그레이드하는 방법에 대한 자세한 내용은 섹션을 참조하세요 [SiteWise Edge 게이트웨이 구성 요소 팩 버전 변경](#).

SiteWise Edge 게이트웨이의 파일 시스템 암호화

SiteWise Edge 게이트웨이를 암호화하고 보호하므로 SiteWise Edge 게이트웨이를 통과하는 산업 데이터가 안전하게 보호됩니다. SiteWise Edge 게이트웨이에 하드웨어 보안 모듈이 있는 경우 SiteWise Edge 게이트웨이를 보호 AWS IoT Greengrass 하도록을 구성할 수 있습니다. 자세한 정보는 AWS IoT Greengrass Version 1 개발자 안내서에서 [하드웨어 보안 통합](#)을 참조하세요. 그렇지 않으면 운영 체제 설명서를 참조하여 파일 시스템을 암호화하고 보호하는 방법을 알아보세요.

엣지 구성에 대한 보안 액세스

엣지 콘솔 애플리케이션 비밀번호나 SiteWise Monitor 애플리케이션 비밀번호를 공유하지 마세요. 이 암호를 다른 사람이 볼 수 있는 곳에 두지 마세요. 암호에 적절한 만료 기간을 설정하여 건전한 암호 교체 정책을 구현하세요.

Siemens Industrial Edge Management의 데이터 보안

AWS IoT SiteWise Edge와 공유하도록 선택한 디바이스 데이터는 Siemens IEM Databus 구성 주제에서 결정됩니다. SiteWise Edge와 공유할 주제를 선택하면 주제 수준 데이터를 AWS IoT SiteWise와 공유할 수 있습니다. Siemens Industrial Edge Marketplace는 별개의 독립 마켓플레이스입니다 AWS. 공유 데이터를 보호하기 위해 Siemens Secured Storage를 사용하지 않으면 SiteWise Edge 애플리케이션이 실행되지 않습니다. 자세한 내용을 알아보려면 Siemens 설명서의 [스토리지 등급](#)을 참조하세요.

SiteWise Monitor 사용자에게 가능한 최소 권한 부여

포털 사용자에게 대한 최소 액세스 정책 권한 집합을 사용하여 최소 권한의 원칙을 따르세요.

- 포털을 만들 때 해당 포털에 필요한 최소한의 자산 집합을 허용하는 역할을 정의합니다. 자세한 내용은 [에 대한 서비스 역할 사용 AWS IoT SiteWise Monitor](#) 섹션을 참조하세요.
- 사용자와 포털 관리자가 프로젝트를 만들고 공유할 때 해당 프로젝트에 필요한 최소한의 자산 집합을 사용합니다.
- 포털 또는 프로젝트에 더 이상 액세스할 필요가 없는 ID는 해당 리소스에서 제거합니다. 해당 ID를 조직에 더 이상 적용할 수 없는 경우 ID 저장소에서 해당 ID를 삭제하세요.

최소 원칙의 모범 사례는 IAM 역할에도 적용됩니다. 자세한 내용은 [정책 모범 사례](#) 섹션을 참조하세요.

민감한 정보 노출 금지

보안 인증 및 개인 식별 정보(PII)와 같은 기타 민감한 정보의 로깅을 방지해야 합니다. SiteWise Edge 게이트웨이의 로컬 로그에 액세스하려면 루트 권한이 필요하고 CloudWatch Logs에 액세스하려면 IAM 권한이 필요한 경우에도 다음과 같은 보호 조치를 구현하는 것이 좋습니다.

- 자산 또는 모델의 이름, 설명 또는 속성에 민감한 정보를 사용하지 마세요.
- SiteWise Edge 게이트웨이 또는 소스 이름에 민감한 정보를 사용하지 마세요.
- 포털, 프로젝트 또는 대시보드의 이름이나 설명에 민감한 정보를 사용하지 마세요.

AWS IoT Greengrass 보안 모범 사례 준수

SiteWise Edge 게이트웨이의 AWS IoT Greengrass 보안 모범 사례를 따릅니다. 자세한 내용은 AWS IoT Greengrass Version 1 개발자 안내서의 [보안 모범 사례](#)를 참조하세요.

다음 사항도 참조하세요.

- AWS IoT 개발자 안내서의 [보안 모범 사례](#)
- [산업 IoT 솔루션을 위한 10가지 보안 항목](#)

에서 로그 및 모니터링 AWS IoT SiteWise

모니터링은 AWS IoT SiteWise 및 다른 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다.는 서비스를 모니터링하고, 이상이 있을 때 보고하고, 적절한 경우 자동 조치를 취할 수 있도록 다음 모니터링 도구를 AWS IoT SiteWise 지원합니다.

- Amazon CloudWatch는 AWS 리소스와 AWS 실행 중인 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성하고, 지표가 특정 임계값에 도달하면 사용자에게 알리거나 조치를 하도록 경보를 설정합니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 [Amazon CloudWatch 사용 설명서](#)를 참조하세요.
- Amazon CloudWatch Logs는 SiteWise Edge 게이트웨이, CloudTrail 및 기타 소스에서 로그 파일을 모니터링, 저장 및 액세스합니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임계값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 [Amazon CloudWatch Logs 사용 설명서](#)를 참조하세요.
- AWS CloudTrail는 AWS 계정에서 또는 계정을 대신하여 수행된 API 호출 및 관련 이벤트를 캡처합니다. 그리고 나서 CloudTrail은 지정된 Amazon S3 버킷에 로그 파일을 전송합니다. 호출한 사용자 및 계정 AWS, 호출이 수행된 소스 IP 주소, 호출이 발생한 시기를 식별할 수 있습니다. 자세한 내용은 [AWS CloudTrail 사용 설명서](#)를 참조하십시오.

주제

- [Amazon CloudWatch Logs를 사용하여 모니터링](#)
- [SiteWise Edge 게이트웨이 로그 모니터링](#)
- [Amazon CloudWatch 지표 AWS IoT SiteWise 로 모니터링](#)
- [를 사용하여 AWS IoT SiteWise API 호출 로깅 AWS CloudTrail](#)

Amazon CloudWatch Logs를 사용하여 모니터링

CloudWatch Logs에 정보를 기록 AWS IoT SiteWise 하도록 구성하여 서비스를 모니터링하고 문제를 해결합니다.

AWS IoT SiteWise 콘솔을 사용할 때는 서비스가 사용자를 대신하여 정보를 로깅할 수 있도록 서비스 연결 역할을 AWS IoT SiteWise 생성합니다. AWS IoT SiteWise 콘솔을 사용하지 않는 경우 로그를 수신하려면 서비스 연결 역할을 수동으로 생성해야 합니다. 자세한 내용은 [AWS IoT SiteWise용 서비스 연결 역할 생성](#) 단원을 참조하십시오.

가 CloudWatch 스트림에 로그 이벤트를 넣을 AWS IoT SiteWise 수 있도록 허용하는 리소스 정책이 있어야 합니다. CloudWatch Logs에 대한 리소스 정책을 생성하고 업데이트하려면 다음 명령을 실행합니다. **logging-policy-name**을 생성할 정책 이름으로 대체하세요.

```
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 Logs는 [aws:SourceArn](#) 및 [aws:SourceAccount](#) 조건 컨텍스트 키도 지원합니다. 이러한 조건 컨텍스트 키는 선택 사항입니다.

가 지정된 리소스와 연결된 로그 AWS IoT SiteWise 만 CloudWatch 스트림에 넣도록 허용하는 AWS IoT SiteWise 리소스 정책을 생성하거나 업데이트하려면 명령을 실행하고 다음을 수행합니다.

- **logging-policy-name**을 생성할 정책 이름으로 대체하세요.
- **source-ARN**을 자산 모델 또는 자산과 같은 AWS IoT SiteWise 리소스의 ARN으로 바꿉니다. 각 AWS IoT SiteWise 리소스 유형에 대한 ARN을 찾으려면 서비스 승인 참조의에서 [정의한 리소스 유형을 AWS IoT SiteWise](#) 참조하세요.
- **account-ID**를 지정된 AWS IoT SiteWise 리소스와 연결된 AWS 계정 ID로 바꿉니다.

```
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\"] } } } ] }"
```

기본적으로 AWS IoT SiteWise 는 CloudWatch Logs에 정보를 기록하지 않습니다. 로깅을 활성화하려면 비활성화() 이외의 로깅 수준을 선택합니다. OFF는 다음 로깅 수준을 AWS IoT SiteWise 지원합니다.

- OFF – 로깅이 해제되어 있습니다.
- ERROR – 오류를 기록합니다.
- INFO – 오류 및 정보 메시지를 기록합니다.

AWS IoT Greengrass를 통해 CloudWatch Logs에 정보를 기록하도록 SiteWise Edge 게이트웨이를 구성할 수 있습니다. 자세한 내용은 [SiteWise Edge 게이트웨이 로그 모니터링](#) 단원을 참조하십시오.

AWS IoT SiteWise 규칙 작업의 문제를 해결하는 경우 CloudWatch Logs에 정보를 AWS IoT Core 로깅하도록 구성할 수도 있습니다. 자세한 내용은 [AWS IoT SiteWise 규칙 작업 문제 해결](#) 단원을 참조하십시오.

목차

- [로그인 관리 AWS IoT SiteWise](#)
 - [로깅 수준 찾기](#)
 - [로깅 수준 변경](#)
- [예: AWS IoT SiteWise 로그 파일 항목](#)

로그인 관리 AWS IoT SiteWise

다음 로깅 구성 작업에는 AWS IoT SiteWise 콘솔 또는 AWS CLI 를 사용합니다.

로깅 수준 찾기

Console

AWS IoT SiteWise 콘솔에서 현재 로깅 수준을 찾아보려면 다음 절차를 따르세요.

현재 AWS IoT SiteWise 로깅 수준을 찾으려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 로깅 옵션을 선택합니다.

현재의 로깅 상태가 로깅 상태 아래에 나타납니다. 로깅을 활성화하면 현재 로깅 수준이 세부 사항 수준 아래에 나타납니다.

AWS CLI

다음 명령을 실행하여에서 현재 AWS IoT SiteWise 로깅 수준을 찾습니다 AWS CLI.

```
aws iotsitewise describe-logging-options
```

이 작업은 로깅 수준이 포함된 응답을 다음 형식으로 반환합니다.

```
{
  "loggingOptions": {
    "level": "String"
  }
}
```

로깅 수준 변경

다음 절차에 따라 AWS IoT SiteWise 콘솔에서 또는를 사용하여 로깅 수준을 변경합니다 AWS CLI.

Console

AWS IoT SiteWise 로깅 수준을 변경하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 로깅 옵션을 선택합니다.
3. 편집을 선택합니다.
4. 활성화할 세부 사항 수준을 선택합니다.
5. 저장(Save)을 선택합니다.

AWS CLI

다음 AWS CLI 명령을 실행하여 AWS IoT SiteWise 로깅 수준을 변경합니다. *logging-level*을 원하는 로깅 수준으로 바꿉니다.

```
aws iotsitewise put-logging-options --logging-options level=logging-level
```

예: AWS IoT SiteWise 로그 파일 항목

각 AWS IoT SiteWise 로그 항목에는 이벤트 정보와 해당 이벤트에 대한 관련 리소스가 포함되어 있으므로 로그 데이터를 이해하고 분석할 수 있습니다.

다음 예제는 자산 모델을 성공적으로 생성할 때가 AWS IoT SiteWise 기록하는 CloudWatch Logs 항목을 보여줍니다.

```
{
  "eventTime": "2020-05-05T00:10:22.902Z",
```

```

{
  "logLevel": "INFO",
  "eventType": "AssetModelCreationSuccess",
  "message": "Successfully created asset model.",
  "resources": {
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
  }
}

```

SiteWise Edge 게이트웨이 로그 모니터링

Amazon CloudWatch Logs 또는 로컬 파일 시스템에 정보를 기록하도록 AWS IoT SiteWise Edge 게이트웨이를 구성할 수 있습니다.

주제

- [Amazon CloudWatch Logs 사용](#)
- [에서 서비스 로그 사용 AWS IoT SiteWise](#)

Amazon CloudWatch Logs 사용

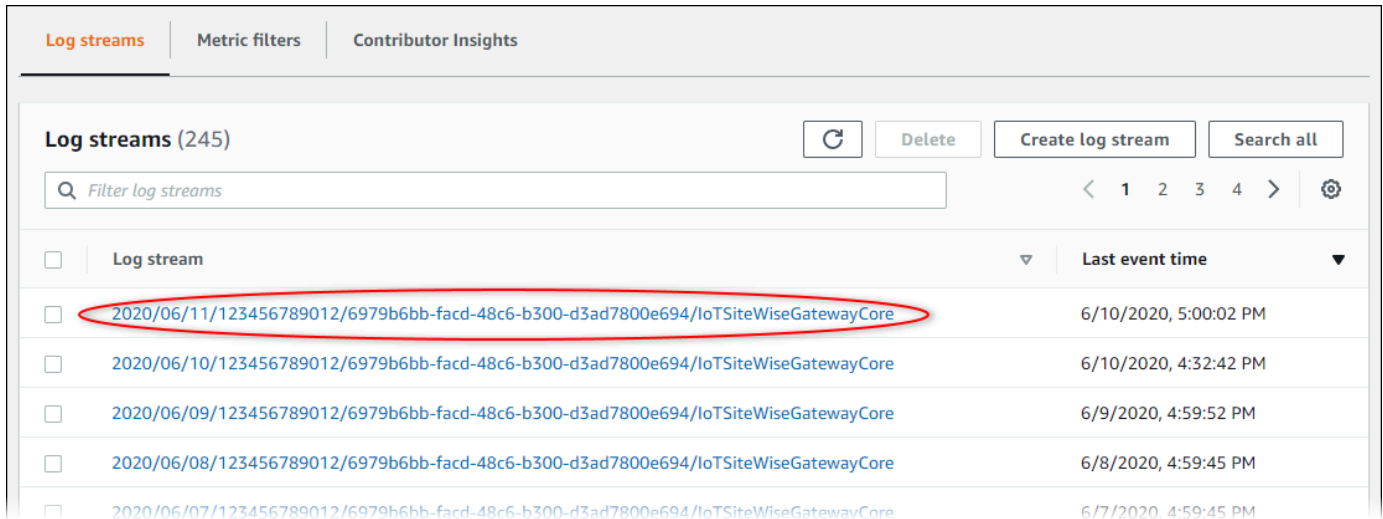
CloudWatch Logs에 로그를 전송하도록 SiteWise Edge 게이트웨이를 구성할 수 있습니다. 자세한 내용은 AWS IoT Greengrass Version 2 개발자 안내서의 [CloudWatch Logs의 로깅 활성화](#)를 참조하세요.

CloudWatch Logs를 구성하고 액세스하려면(콘솔) 다음을 수행하세요.

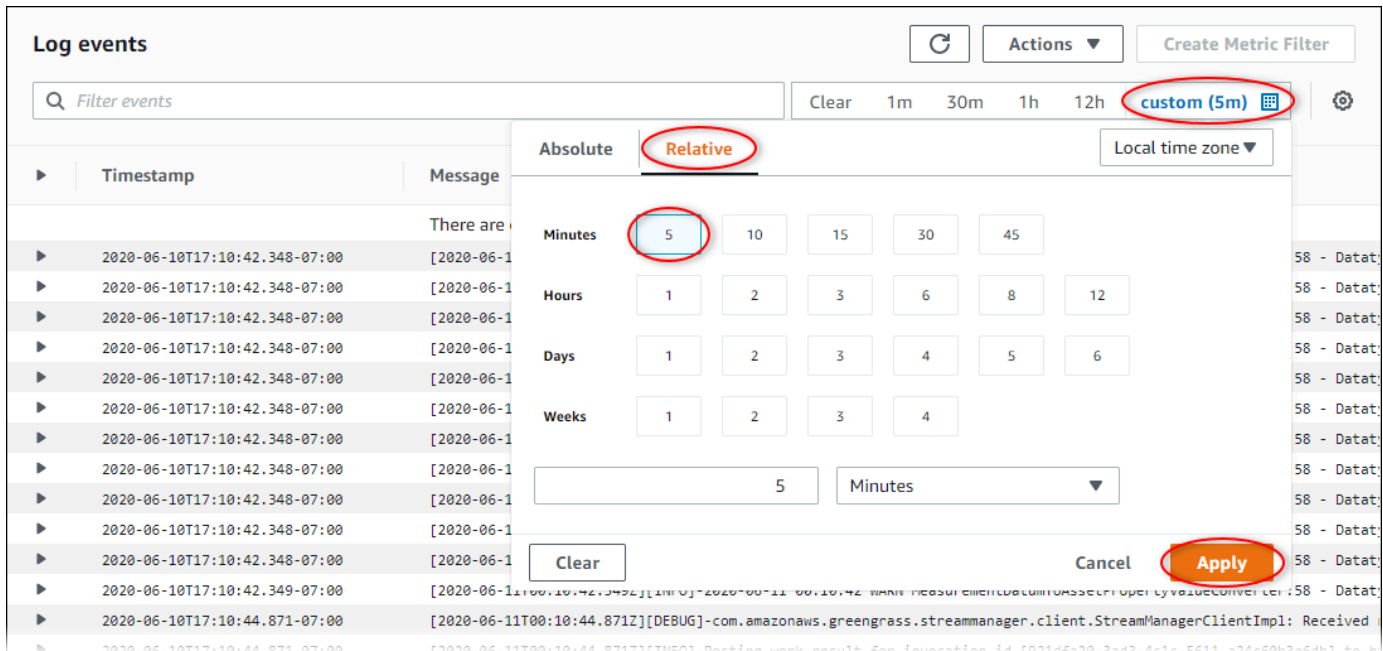
1. [CloudWatch 콘솔](#)로 이동합니다.
2. 탐색 창에서 로그 그룹을 선택합니다.
3. 구성 AWS IoT SiteWise 요소 로그는 다음 로그 그룹에서 찾을 수 있습니다.
 - `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgeCollectorOpcua` - SiteWise Edge 게이트웨이의 OPC UA 소스에서 데이터를 수집하는 SiteWise Edge 게이트웨이의 구성 요소에 대한 로그입니다.
 - `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgePublisher` - OPC UA 데이터 스트림을 게시하는 SiteWise Edge 게이트웨이의 구성 요소에 대한 로그입니다 AWS IoT SiteWise.

디버깅할 함수의 로그 그룹을 선택합니다.

4. AWS IoT Greengrass 그룹 이름으로 끝나는 이름이 있는 로그 스트림을 선택합니다. 기본적으로 CloudWatch는 가장 최근 로그 스트림을 먼저 표시합니다.



5. 최근 5분 동안의 로그를 표시하려면 다음을 수행합니다.
- 오른쪽 상단 모서리에서 사용자 지정을 선택합니다.
 - 상대를 선택합니다.
 - 5분을 선택합니다.
 - 적용을 선택합니다.



6. (선택 사항) 더 적은 수의 로그를 보려면 오른쪽 상단 모서리에서 1분을 선택하면 됩니다.

7. 가장 최근 로그를 표시하려면 로그 항목의 맨 아래로 스크롤합니다.

에서 서비스 로그 사용 AWS IoT SiteWise

SiteWise Edge 게이트웨이 디바이스에는 문제를 디버깅하는 데 도움이 되는 서비스 로그 파일이 포함되어 있습니다. 다음 섹션에서는 AWS IoT SiteWise OPC UA Collector 및 AWS IoT SiteWise Publisher 구성 요소에 대한 서비스 로그 파일을 찾고 활용하는 데 도움이 됩니다.

AWS IoT SiteWise OPC UA Collector 서비스 로그 파일

AWS IoT SiteWise OPC UA Collector 구성 요소는 다음 로그 파일을 사용합니다.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeCollector0pcua.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollector0pcua.log
```

이 구성 요소의 로그를 보려면

- Core 디바이스에서 다음과 같은 명령을 실행하여 실시간으로 이 구성 요소의 로그 파일을 봅니다. */greengrass/v2* 또는 *C:\greengrass\v2*를 AWS IoT Greengrass 루트 폴더의 경로로 바꿉니다.

Linux

```
sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeCollector0pcua.log
```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollector0pcua.log -Tail  
10 -Wait
```

AWS IoT SiteWise 게시자 서비스 로그 파일

AWS IoT SiteWise 게시자 구성 요소는 다음 로그 파일을 사용합니다.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log
```

이 구성 요소의 로그를 보려면

- Core 디바이스에서 다음과 같은 명령을 실행하여 실시간으로 이 구성 요소의 로그 파일을 봅니다. `/greengrass/v2` 또는 `C:\greengrass\v2`를 AWS IoT Greengrass 루트 폴더의 경로로 바꿉니다.

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
```

Amazon CloudWatch 지표 AWS IoT SiteWise 로 모니터링

원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표로 처리하는 CloudWatch를 AWS IoT SiteWise 사용하여 모니터링할 수 있습니다. 이러한 통계는 15개월간 보관되므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 특정 임계값을 주시하다가 해당 임계값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 [Amazon CloudWatch 사용 설명서](#)를 참조하세요.

AWS IoT SiteWise 는 아래 섹션에 나열된 지표와 차원을 AWS/IoTSiteWise 네임스페이스에 게시합니다.

Tip

AWS IoT SiteWise 는 1분 간격으로 지표를 게시합니다. CloudWatch 콘솔에서 이러한 지표를 그래프로 볼 때 기간을 1분으로 선택하는 것이 좋습니다. 이렇게 하면 지표 데이터를 가능한 최고 해상도로 볼 수 있습니다.

주제

- [AWS IoT Greengrass Version 2 게이트웨이 지표](#)

AWS IoT Greengrass Version 2 게이트웨이 지표

AWS IoT SiteWise 는 Classic 스트림, V2 게이트웨이 및 MQTT 지원 V3 게이트웨이에 대한 게이트웨이 지표를 게시합니다. 달리 명시되지 않는 한 각 지표는 두 자체 호스팅 게이트웨이 버전 모두에 적용됩니다. 모든 SiteWise Edge 게이트웨이 지표는 1분 간격으로 게시됩니다.

SiteWise Edge 게이트웨이 지표

지표	설명
Gateway.AvailableMemory	SiteWise Edge 게이트웨이의 사용 가능한 메모리. 단위: 바이트 측정 기준: 없음
Gateway.AvailableDiskSpace	SiteWise Edge 게이트웨이의 사용 가능한 디스크 공간. 단위: 바이트 측정 기준: 없음
Gateway.CloudConnectivity	SiteWise Edge 게이트웨이의 클라우드 연결 상태. 단위: 없음 측정 기준: GatewayId

지표	설명
Gateway.CpuUsage	SiteWise Edge 게이트웨이의 CPU 사용량. 단위: 퍼센트 측정 기준: 없음
Gateway.TotalDiskSpace	SiteWise Edge 게이트웨이의 총 디스크 공간. 단위: 바이트 측정 기준: 없음
Gateway.TotalMemory	SiteWise Edge 게이트웨이의 총 메모리. 단위: 바이트 측정 기준: 없음
Gateway.UsedDiskSpace	SiteWise Edge 게이트웨이의 사용된 디스크 공간. 단위: 바이트 측정 기준: 없음
Gateway.UsedMemory	SiteWise Edge 게이트웨이의 사용된 메모리. 단위: 바이트 측정 기준: 없음
Gateway.UsedPercentageDiskSpace	SiteWise Edge 게이트웨이의 디스크 공간 사용률. 단위: 바이트 측정 기준: 없음

지표	설명
Gateway.UsedPercentageMemory	SiteWise Edge 게이트웨이의 메모리 사용률. 단위: 바이트 측정 기준: 없음

AWS IoT SiteWise 게시자 지표

지표	설명
IoTSiteWisePublisher.ComponentBuildVersion	이 지표는 게이트웨이에서 실행되는 IoT SiteWise 게시자 구성 요소의 빌드 버전을 나타냅니다. 값이 1이면 게이트웨이가 Component BuildVersion 차원에 해당하는 게시자 버전을 실행 중임을 나타냅니다. 단위: 1 차원: GatewayId, ComponentBuildVersion
IoTSiteWisePublisher.DroppedCount	SiteWise Edge 게이트웨이(GatewayId)에 의해 삭제되고 클라우드에 게시되지 않은 데이터 포인트의 수로, 1분마다 생성됩니다. 단위: 수 측정 기준: GatewayId
IoTSiteWisePublisher.Heartbeat	SiteWise Edge 게이트웨이의 게시자에 의해 1분마다 생성됩니다. 단위: 1(게시자가 실행 중임을 나타내는 1 및 게시자가 실행 중이 아님을 나타내는 데이터 포인트 누락) 측정 기준: GatewayId

지표	설명
<code>IoTSiteWisePublisher.IsConnectedToMqttBroker</code>	SiteWise Edge 게이트웨이의 게시자에 의해 1분마다 생성됩니다. 단위: 1(게시자가 MQTT 브로커에 연결되어 있음을 나타내는 1) 측정 기준: GatewayId
<code>IoTSiteWisePublisher.MessageCheckpointPersistenceErrorCount</code>	지표는 게이트웨이가 게시자가 처리한 데이터를 추적하는 데 사용되는 체크포인트 파일에서 문제를 감지했음을 나타냅니다. 값이 1이면 오류가 발생했다는 의미입니다. 단위: 없음 차원: AccountId, GatewayId
<code>IoTSiteWisePublisher.MqttMessageReceivedSuccessCount</code>	게시자가 MQTT 브로커로부터 성공적으로 수신한 메시지 수로, 1분마다 생성됩니다. 단위: 수 측정 기준: GatewayId
<code>IoTSiteWisePublisher.MqttReceivedSuccessBytes</code>	게시자가 MQTT 브로커로부터 성공적으로 수신한 메시지 데이터의 바이트 수로, 1분마다 생성됩니다. 단위: 수 측정 기준: GatewayId
<code>IoTSiteWisePublisher.NumberOfSubscriptionsToMqttBroker</code>	게시자에 의해 MQTT 브로커에 구독된 주제의 수로, 1분마다 생성됩니다. 다단계 와일드카드 주제는 1로 계산됩니다. 단위: 수 측정 기준: GatewayId

지표	설명
<code>IoTSiteWisePublisher.NumberOfUniqueMqttTopicsReceived</code>	게시자가 MQTT 브로커로부터 수신한 고유한 주제 수로, 1분마다 생성됩니다. 단위: 수 측정 기준: GatewayId
<code>IoTSiteWisePublisher.PublisherFailureCount</code>	SiteWise Edge 게이트웨이(GatewayId)가 게시하지 못한 데이터 포인트 수로, 1분마다 생성됩니다. 단위: 수 측정 기준: GatewayId
<code>IoTSiteWisePublisher.PublisherRejectedCount</code>	SiteWise Edge 게이트웨이(GatewayId)가 클라우드 측에서 거부한 데이터 포인트 수로, 1분마다 생성됩니다. 단위: 수 측정 기준: GatewayId
<code>IoTSiteWisePublisher.PublisherSuccessCount</code>	SiteWise Edge 게이트웨이(GatewayId)가 클라우드에 성공적으로 게시한 데이터 포인트 수로, 1분마다 생성됩니다. 단위: 수 측정 기준: GatewayId
<code>IoTSiteWisePublisher.PublisherToS3FailureCount</code>	게이트웨이(GatewayId)가 Amazon S3 버킷에 게시하지 못한 데이터 포인트 수입니다. 단위: 수 측정 기준: GatewayId

지표	설명
IoTSiteWisePublisher.Publi shToS3SuccessCount	게이트웨이(GatewayId)가 Amazon S3 버킷에 성공적으로 게시한 데이터 포인트 수입니다. 단위: 수 측정 기준: GatewayId

OPC UA 수집기 지표

지표	설명
OpcUaCollector.ActiveDataSt reamCount	SiteWise Edge 게이트웨이(gatewayId)가 OPC UA 소스(sourceName)에 대해 구독한 데이터 스트림 수입니다. 단위: 수 크기: 게이트웨이 ID, 소스 이름, 속성 그룹
OpcUaCollector.ComponentBui ldVersion (클래식 스트림, V2 게이트웨이에서는 사용할 수 없음)	이 지표는 게이트웨이에서 실행되는 IoT SiteWise OPC UA 수집기 구성 요소의 빌드 버전을 나타냅니다. 값이 1이면 게이트웨이가 ComponentBuildVersion 차원에 해당하는 수집기 버전을 실행 중임을 나타냅니다. 단위: 1 차원: GatewayId, ComponentBuildVersion
OpcUaCollector.ConversionErrors	SiteWise Edge 게이트웨이(gatewayId)가 OPC UA 소스(sourceName)에 대해 수신하여 데이터를 보내는 동안 변환 오류가 발생한 데이터 포인트 수입니다 AWS IoT SiteWise. 이러한 데이터 포인트는 OPC UA 수집기가 수집하지 않습니다. 단위: 수

지표	설명
	크기: 게이트웨이 ID, 소스 이름
<code>OpcUaCollector.Heartbeat</code>	SiteWise Edge 게이트웨이(<code>gatewayId</code>)에 연결된 각 OPC UA 소스(<code>sourceName</code>)에 대해 1분마다 생성됩니다. 단위: 1(소스가 연결되었음을 나타내는 1 및 소스 연결이 끊겼음을 나타내는 0) 크기: 게이트웨이 ID, 소스 이름
<code>OpcUaCollector.IncomingValuesCount</code>	SiteWise Edge 게이트웨이(<code>gatewayId</code>)가 OPC UA 소스(<code>sourceName</code>)에 대해 수신한 데이터 포인트 수로, 1분마다 생성됩니다. 단위: 수 크기: 게이트웨이 ID, 소스 이름, 속성 그룹
<code>OpcUaCollector.IncomingValueErrors</code>	SiteWise Edge 게이트웨이(<code>gatewayId</code>)가 OPC UA 소스(<code>sourceName</code>)에 대해 수신한 잘못된 값인 데이터 포인트 수입니다. 이러한 데이터 포인트는 OPC UA 수집기가 수집하지 않으며 1분마다 생성됩니다. 단위: 수 크기: 게이트웨이 ID, 소스 이름, 속성 그룹
<code>OpcUaCollector.IsConnectedToMqttBroker</code> (클래식 스트림, V2 게이트웨이에서는 사용할 수 없음)	SiteWise Edge 게이트웨이의 IoT SiteWise OPC UA 수집기 구성 요소에 의해 1분마다 생성됩니다. 단위: 1(IoT SiteWise OPC UA 수집기 구성 요소가 MQTT 브로커에 연결되어 있음을 나타내는 1) 측정 기준: <code>GatewayId</code>

지표	설명
<code>OpcUaCollector.MqttMessagesDroppedCount</code> (클래식 스트림, V2 게이트웨이에서는 사용할 수 없음)	IoT SiteWise OPC UA 수집기 구성 요소에서 삭제된 MQTT 메시지 수입니다. 단위: 수 크기: 게이트웨이 ID, 소스 이름
<code>OpcUaCollector.MqttMessagesPublishedBytes</code> (클래식 스트림, V2 게이트웨이에서는 사용할 수 없음)	IoT SiteWise OPC UA 수집기 구성 요소가 MQTT 브로커에 성공적으로 게시한 MQTT 메시지 데이터의 바이트 수입니다. 단위: 수 크기: 게이트웨이 ID, 소스 이름
<code>OpcUaCollector.MqttMessagesPublishedCount</code> (클래식 스트림, V2 게이트웨이에서는 사용할 수 없음)	IoT SiteWise OPC UA 수집기 구성 요소가 MQTT 브로커에 성공적으로 게시한 MQTT 메시지 수입니다. 단위: 수 크기: 게이트웨이 ID, 소스 이름
<code>OpcUaCollector.NullValueCount</code> (클래식 스트림, V2 게이트웨이에서는 사용할 수 없음)	OPC UA 서버에서 IoT SiteWise OPC UA 수집기 구성 요소가 수신한 null 값의 수입니다. 단위: 수 크기: 게이트웨이 ID, 소스 이름, 속성 그룹
<code>OpcUaCollector.NumberOfUniqueMqttTopicsPublished</code> (클래식 스트림, V2 게이트웨이에서는 사용할 수 없음)	IoT SiteWise OPC UA 수집기가 MQTT 브로커에 게시한 고유한 MQTT 주제 수입니다. 단위: 수 크기: 게이트웨이 ID, 소스 이름

AWS IoT SiteWise 프로세서 지표

지표	설명
<code>Gateway.DataProcessor.IngestionThrottled</code> (MQTT 지원 V3 게이트웨이에서는 사용할 수 없음)	스로틀링된 데이터 포인트 수로, 1분마다 생성됩니다. 단위: 수 차원: <code>ThrottledAt</code>
<code>Gateway.DataProcessor.MeasurementRejected</code> (MQTT 지원 V3 게이트웨이에서는 사용할 수 없음)	거부된 측정 수로, 1분마다 생성됩니다. 단위: 수 차원: <code>Reason</code>
<code>Gateway.DataProcessor.MessagesRemaining</code> (MQTT 지원 V3 게이트웨이에서는 사용할 수 없음)	스트림에 남아 있는 메시지 수로, 1분마다 생성됩니다. 단위: 수 측정 기준: <code>StreamName</code>
<code>Gateway.DataProcessor.ProcessingError</code> (MQTT 지원 V3 게이트웨이에서는 사용할 수 없음)	처리 오류 수로, 1분마다 생성됩니다. 단위: 수 차원: <code>Reason</code>

를 사용하여 AWS IoT SiteWise API 호출 로깅 AWS CloudTrail

AWS IoT SiteWise 는 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다 AWS IoT SiteWise. CloudTrail은 AWS IoT SiteWise 에 대한 API 직접 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 AWS IoT SiteWise 콘솔의 호출과 AWS IoT SiteWise API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 이벤트를 포함하여 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다 AWS IoT SiteWise. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 수행된 요청, 요청이 수행된 AWS IoT SiteWise IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 [AWS CloudTrail 사용 설명서](#)를 참조하세요.

AWS IoT SiteWise CloudTrail의 정보

CloudTrail은 AWS 계정을 생성할 때 계정에서 활성화됩니다. 지원되는 이벤트 활동이에서 발생하면 AWS IoT SiteWise 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 내용은 [CloudTrail 이벤트 기록을 사용하여 이벤트 보기](#)를 참조하십시오.

에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 AWS IoT SiteWise 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 지역에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 추가적으로, CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- [추적 생성 개요](#)
- [CloudTrail 지원 서비스 및 통합](#)
- [CloudTrail에 대한 Amazon SNS 알림 구성](#)
- [여러 리전에서 CloudTrail 로그 파일 수신 및 여러 계정에서 CloudTrail 로그 파일 수신](#)

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자에게 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에 의해 이루어졌는지 여부입니다.

자세한 내용은 [CloudTrail userIdentity 요소](#)를 참조하세요.

AWS IoT SiteWise CloudTrail의 데이터 이벤트

[데이터 이벤트](#)는 리소스 기반 또는 리소스에서 수행된 리소스 작업에 대한 정보를 제공합니다(예: Amazon S3 객체 읽기 또는 쓰기). 이를 데이터 영역 작업이라고도 합니다. 데이터 이벤트가 대량 활동인 경우도 있습니다. 기본적으로 CloudTrail은 데이터 이벤트를 로깅하지 않습니다. CloudTrail 이벤트 기록은 데이터 이벤트를 기록하지 않습니다.

데이터 이벤트에는 추가 요금이 적용됩니다. CloudTrail 요금에 대한 자세한 내용은 [AWS CloudTrail 요금](#)을 참조하세요.

CloudTrail 콘솔 AWS CLI 또는 CloudTrail API 작업을 사용하여 AWS IoT SiteWise 리소스 유형에 대한 데이터 이벤트를 로깅할 수 있습니다. 이 섹션의 [표](#)에는 사용 가능한 리소스 유형이 나와 있습니다 AWS IoT SiteWise.

- CloudTrail 콘솔을 사용하여 데이터 이벤트를 로깅하려면 [추적](#) 또는 [이벤트 데이터 스토어](#)를 생성하여 데이터 이벤트를 기록하거나 [기존 추적 또는 이벤트 데이터 스토어를 업데이트](#)하여 데이터 이벤트를 기록합니다.
 - 데이터 이벤트를 로깅하려면 데이터 이벤트를 선택합니다.
 - 데이터 이벤트 유형에서 데이터 이벤트를 로깅할 리소스 유형을 선택합니다.
 - 사용할 템플릿에서 로그 선택기를 선택합니다. 리소스 유형에 대한 모든 데이터 이벤트를 로그하거나, 모든 readOnly 이벤트를 로그하거나, 모든 writeOnly 이벤트를 로그하거나, 사용자 지정 로그 선택기 템플릿을 생성하여 readOnly, eventName 및 resources.ARN 필드를 필터링할 수 있습니다.
- 를 사용하여 데이터 이벤트를 로깅하려면 eventCategory 필드를 로 AWS CLI 설정하고 Data resources.type 필드를 리소스 유형 값과 로 설정하도록 --advanced-event-selectors 파라미터를 구성합니다([표](#) 참조). 조건을 추가하여 readOnly, eventName 및 resources.ARN 필드의 값을 필터링할 수 있습니다.
 - 데이터 이벤트를 로깅하도록 추적을 구성하려면 [AWS CloudTrail put-event-selectors](#) 명령을 실행합니다. 자세한 내용은 [AWS CLI를 사용하여 추적에 대한 데이터 이벤트 로깅](#)을 참조하세요.
 - 데이터 이벤트를 로깅하도록 이벤트 데이터 스토어를 구성하려면 [AWS CloudTrail create-event-data-store](#) 명령을 실행하여 새 이벤트 데이터 스토어를 생성함으로써 데이터 이벤트를 로깅하거나 [AWS CloudTrail update-event-data-store](#) 명령을 실행하여 기존 이벤트 데이터 스토어를 업데이트합니다. 자세한 내용은 [AWS CLI를 사용해 데이터 이벤트 스토어에 대한 데이터 이벤트 로깅](#)을 참조하세요.

다음 표에는 AWS IoT SiteWise 리소스 유형이 나열되어 있습니다. 데이터 이벤트 유형(콘솔) 열에는 CloudTrail 콘솔의 데이터 이벤트 유형 목록에서 선택할 값이 표시됩니다. resources.type 값 열에는 AWS CLI 또는 CloudTrail APIs를 사용하여 고급 이벤트 선택기를 구성할 때 지정하는 resources.type 값이 표시됩니다. CloudTrail에 로깅되는 데이터 API 열에는 리소스 유형에 대해 CloudTrail에 로깅된 API 호출이 표시됩니다.

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API*
AWS IoT SiteWise 애셋	AWS::IoTSiteWise::Asset	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory • BatchGetAssetPropertyAggregates
AWS IoT SiteWise 시계열	AWS::IoTSiteWise::TimeSeries	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory

데이터 이벤트 유형(콘솔)	resources.type 값	CloudTrail에 로깅되는 데이터 API*
		BatchGetAssetPropertyAggregates
AWS IoT SiteWise Assistant	AWS::SitewiseAssistant::Conversation	InvokeAssistant

Note

Cloudtrail 이벤트에 로깅된 resources.type은 API 요청에 사용되는 식별자에 따라 달라집니다. 요청에 자산 ID가 지정되면 Asset resources.type이 로깅되고, 그렇지 않으면 TimeSeries resources.type이 로깅됩니다.

*eventName, readOnly 및 resources.ARN 필드를 필터링하여 중요한 이벤트만 로깅하도록 고급 이벤트 선택기를 구성할 수 있습니다. 필드에 대한 자세한 내용은 [AdvancedFieldSelector](#) 섹션을 참조하세요.

AWS IoT SiteWise CloudTrail의 관리 이벤트

[관리 이벤트 로깅](#)은 AWS 계정의 리소스에서 수행되는 관리 작업에 대한 정보를 제공합니다. 이를 컨트롤 플레인 작업이라고도 합니다. 기본적으로 CloudTrail은 관리 이벤트를 로깅합니다.

AWS IoT SiteWise 는 모든 AWS IoT SiteWise 컨트롤 플레인 작업을 관리 이벤트로 기록합니다. CloudTrail에 AWS IoT SiteWise 로깅하는 AWS IoT SiteWise 컨트롤 플레인 작업 목록은 [AWS IoT SiteWise API 참조](#)를 참조하세요.

예: AWS IoT SiteWise 로그 파일 항목

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예는 CreateAsset작업을 보여주는 CloudTrail 로그 항목입니다.

```
{
  "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-bbbbbbEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

AWS IoT SiteWise 리소스에 태그 지정

AWS IoT SiteWise 리소스에 태그를 지정하면 조직 자산을 효율적으로 분류, 관리 및 검색할 수 있는 강력한 방법을 제공합니다. 키-값 페어로 구성된 태그를 할당하면 리소스에 설명 메타데이터를 연결할 수 있습니다. 태그의 메타데이터를 사용하여 작업을 간소화할 수 있습니다. 예를 들어 풍력 발전소 시나리오에서 태그를 사용하면 위치, 용량 및 운영 상태와 같은 특정 속성으로 터빈에 레이블을 지정하여 AWS IoT SiteWise 내에서 빠르게 식별하고 관리할 수 있습니다.

태그를 AWS Identity and Access Management (IAM) 정책과 통합하면 조건부 액세스 규칙을 정의하여 보안 및 운영 제어가 향상됩니다. 즉, 특정 태그가 있는 사용자만 지정할 수 있습니다. 예를 들어 특정 역할 또는 부서로 태그가 지정된 사람만 특정 리소스에 액세스하거나 이를 수정할 수 있습니다.

에서 태그 사용 AWS IoT SiteWise

태그를 사용하여 용도, 소유자, 환경 또는 사용 사례에 맞는 기타 분류별로 AWS IoT SiteWise 리소스를 분류합니다. 동일한 유형의 리소스가 많은 경우, 할당한 태그를 기준으로 특정 리소스를 빠르게 식별할 수 있습니다.

각 태그는 키와 사용자가 지정하는 선택적 값으로 구성됩니다. 예를 들어 지원하는 산업 프로세스에 따라 자산 모델을 추적하기 위해 자산 모델에 대한 일련의 태그를 설정할 수 있습니다. 관리하는 각 유형의 리소스에 맞는 태그 키 세트를 개발하는 것이 좋습니다. 일관된 태그 키 세트를 사용하면 리소스를 보다 쉽게 관리할 수 있습니다.

로 태그 지정 AWS Management Console

의 태그 편집기는 모든 AWS 서비스의 리소스에 대한 태그를 생성하고 관리할 수 있는 중앙 통합 방법을 AWS Management Console 제공합니다. 자세한 내용은 AWS 리소스 [태깅 및 Tag Editor 사용 설명서](#)의 [Tag Editor 시작하기](#)를 참조하세요.

AWS IoT SiteWise API로 태그 지정

AWS IoT SiteWise API는 태그도 사용합니다. 태그를 생성하기 전에 태그 지정에 대한 제한 사항을 유의하세요. 자세한 내용은 AWS 일반 참조의 [태그 이름 지정 및 사용 규칙](#)을 참조하세요.

- 리소스를 만들 때 태그를 추가하려면 리소스의 tags 속성에서 태그를 정의합니다.
- 기존 리소스에 태그를 추가하거나 태그 값을 업데이트하려면 [TagResource](#) 작업을 사용합니다.
- 리소스에서 태그를 제거하려면 [UntagResource](#) 작업을 사용합니다.

- 리소스와 연결된 태그를 검색하려면 [ListTagsForResource](#) 작업을 사용하거나, 해당 리소스를 설명하고 tags 속성을 검사합니다.

다음 표에는 AWS IoT SiteWise API를 사용하여 태그를 지정할 수 있는 리소스와 해당 Create 및 Describe 작업이 나와 있습니다.

태그 지정 가능한 AWS IoT SiteWise 리소스

리소스	작업 생성	작업 설명
자산 모델 또는 구성 요소 모델	CreateAssetModel	DescribeAssetModel
자산	CreateAsset	DescribeAsset
SiteWise Edge 게이트웨이	CreateGateway	DescribeGateway
Portal	CreatePortal	DescribePortal
Project	CreateProject	DescribeProject
대시보드	CreateDashboard	DescribeDashboard
액세스 정책	CreateAccessPolicy	DescribeAccessPolicy
시계열	BatchPutAssetPropertyValue	DescribeTimeSeries

의 [BatchPutAssetPropertyValue](#) 경우 자산 모델 및 자산을 생성하기 AWS IoT SiteWise 전에 산업 데이터를 로 보내도록 데이터 소스를 구성할 수 있습니다. AWS IoT SiteWise 는 자동으로 데이터 스트림을 생성하여 장비에서 원시 데이터 스트림을 수신합니다. 자세한 내용은 [데이터 수집 관리](#)를 참조하세요.

다음 작업을 사용하여 태그 지정이 가능한 리소스의 태그를 보고 관리합니다.

- [TagResource](#) – 리소스에 태그를 추가하거나 기존 태그의 값을 업데이트합니다.
- [ListTagsForResource](#) – 리소스의 태그를 나열합니다.
- [UntagResource](#) – 리소스에서 태그를 제거합니다.

언제든지 리소스에서 태그를 추가하거나 제거할 수 있습니다. 기존 태그 키의 값을 업데이트하려면 동일한 키와 원하는 새로운 값을 포함한 태그를 리소스에 추가합니다. 이 작업은 이전 값을 새 값으로 바꿉니다. 빈 문자열을 태그 값으로 할당할 수 있지만 null 값은 할당할 수 없습니다.

리소스를 삭제하면 리소스에 연결된 모든 태그도 제거됩니다.

IAM 정책에 태그 사용

IAM 정책에서 리소스 태그를 사용하여 사용자 액세스 및 권한을 제어합니다. 예를 들어, 정책은 사용자가 특정 태그가 연결된 리소스만 생성하도록 허용할 수 있습니다. 또한 정책은 사용자가 특정 태그가 있는 리소스를 생성하거나 수정하는 것을 제한할 수 있습니다.

Note

태그를 사용하여 리소스에 대한 사용자의 액세스를 허용하거나 거부하는 경우, 동일한 리소스에 대한 사용자의 태그 추가 또는 제거 권한을 거부해야 합니다. 그러지 않으면 사용자가 제한을 우회하여 태그를 수정함으로써 리소스에 대한 액세스 권한을 얻을 수 있습니다.

정책 설명의 Condition 요소(Condition 블록이라고도 함)에 다음 조건 컨텍스트 키와 값을 사용할 수 있습니다.

`aws:ResourceTag/tag-key: tag-value`

특정 태그가 있는 리소스에 대한 작업을 허용하거나 거부합니다.

`aws:RequestTag/tag-key: tag-value`

태그 지정이 가능한 리소스를 만들거나 수정할 때 특정 태그를 사용하거나 사용하지 않도록 요구합니다.

`aws:TagKeys: [tag-key, ...]`

태그 지정이 가능한 리소스를 만들거나 수정할 때 특정 태그 키 집합을 사용하거나 사용하지 않도록 요구합니다.

Note

IAM 정책의 조건 컨텍스트 키 및 값은 태그 지정 가능한 리소스가 필수 파라미터인 작업에만 적용됩니다. 예를 들어 [ListAssets](#)에 대해 태그 기반의 조건부 액세스를 설정할 수 있습니다. 이

요청에서 태그 지정 가능한 리소스를 참조하지 않으므로 [PutLoggingOptions](#)에 대해 태그 기반의 조건부 액세스를 설정할 수 없습니다.

자세한 내용은 IAM 사용 설명서의 [AWS 리소스 태그를 사용하여 리소스에 대한 액세스 제어](#) 및 IAM JSON 정책 참조를 참조하세요. https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies.html

태그를 사용하는 IAM 정책의 예

- [태그를 기반으로 AWS IoT SiteWise 자산 보기](#)

문제 해결 AWS IoT SiteWise

다음 정보를 사용하여 문제를 해결합니다 AWS IoT SiteWise.

주제

- [SiteWise Edge 게이트웨이 문제 해결](#)
- [AWS IoT SiteWise 포털 문제 해결](#)
- [AWS IoT SiteWise 규칙 작업 문제 해결](#)
- [일괄 가져오기 및 내보내기 작업 문제 해결](#)

SiteWise Edge 게이트웨이 문제 해결

관련 주제를 탐색하여 일반적인 AWS IoT SiteWise Edge 게이트웨이 문제를 해결합니다.

SiteWise Edge 게이트웨이가 보고한 CloudWatch 지표를 보고 연결 또는 데이터 스트림 관련 문제를 해결할 수도 있습니다. 자세한 내용은 [Amazon CloudWatch 지표 AWS IoT SiteWise 로 모니터링](#) 단원을 참조하십시오.

주제

- [SiteWise Edge 게이트웨이 로그 구성 및 액세스](#)
- [SiteWise Edge 게이트웨이 문제 해결](#)
- [에서 AWS IoT SiteWise Edge 애플리케이션 문제 해결 Siemens Industrial Edge](#)
- [Edge에서 오픈 소스 통합 문제 해결](#)
- [AWS IoT Greengrass 문제 해결](#)

SiteWise Edge 게이트웨이 로그 구성 및 액세스

SiteWise Edge 게이트웨이 로그를 보려면 먼저 Amazon CloudWatch Logs로 로그를 보내거나 로컬 파일 시스템에 로그를 저장하도록 SiteWise Edge 게이트웨이를 구성해야 합니다.

- 를 사용하여 SiteWise Edge 게이트웨이의 로그 파일을 AWS Management Console 보려면 CloudWatch Logs를 사용합니다. SiteWise 자세한 내용은 [Amazon CloudWatch Logs 사용](#) 단원을 참조하십시오.

- 명령줄 또는 로컬 소프트웨어를 사용하여 SiteWise Edge 게이트웨이의 로그 파일을 보려면 로컬 파일 시스템 로그를 사용합니다. 자세한 내용은 [에서 서비스 로그 사용 AWS IoT SiteWise](#) 단원을 참조하십시오.

SiteWise Edge 게이트웨이 문제 해결

다음 정보를 사용하여 SiteWise Edge 게이트웨이 문제를 해결합니다.

문제

- [SiteWise Edge 게이트웨이에 팩을 배포할 수 없음](#)
- [AWS IoT SiteWise OPC UA 서버에서 데이터를 수신하지 않음](#)
- [대시보드에 데이터가 표시되지 않음](#)
- [/greengrass/v2/logs 오류에서 aws.iot.siteWiseEdgePublisher 로그에 “메인 클래스를 찾거나 로드할 수 없음”이라는 메시지가 표시됨.](#)
- ['SESSION_TAKEN_OVER' 또는 'com.aws.greengrass.mqttclient.MqttClient 스폰서를 통해 메시지를 게시하지 못했으며 다시 시도할 것입니다.' 메시지가 표시됨 로그에서](#)
- ['com.aws.greengrass.deployment.IotJobsHelper: 배포 작업을 찾을 수 없습니다.' 또는 '배포 결과가 이미 보고되었습니다.' 메시지가 표시됨 로그에서](#)
- [OPC UA 데이터 소스의 속성 그룹에서 타임스탬프 설정을 구성하려고 하면 'SYNC_FAILED' 상태가 표시됩니다.](#)
- [변환된 데이터 유형이 포함되지 않음](#)
- [트러스트 스토어 문제](#)
- [프록시 지원 설치 문제](#)

SiteWise Edge 게이트웨이에 팩을 배포할 수 없음

AWS IoT Greengrass nucleus 구성 요소(aws.greengrass.Nucleus)가 오래된 경우 SiteWise Edge 게이트웨이에 팩을 배포하지 못할 수 있습니다. AWS IoT Greengrass V2 콘솔을 사용하여 AWS IoT Greengrass nucleus 구성 요소를 업그레이드할 수 있습니다.

AWS IoT Greengrass nucleus 구성 요소를 업그레이드하려면(콘솔)

1. [AWS IoT Greengrass 콘솔](#)로 이동합니다.
2. 탐색 창의 AWS IoT Greengrass에서 배포를 선택합니다.

3. 배포 목록에서 수정하려는 배포를 선택합니다.
4. 수정을 선택합니다.
5. 대상 지정 페이지에서 다음을 선택합니다.
6. 구성 요소 선택 페이지의 퍼블릭 구성 요소에서 검색 상자에 **aws.greengrass.Nucleus**를 입력한 다음 aws.greengrass.Nucleus를 선택합니다.
7. 다음을 선택합니다.
8. 구성 요소 구성 페이지에서 다음을 선택합니다.
9. 고급 설정 구성 페이지에서 다음을 선택합니다.
10. 검토 페이지에서 배포를 선택합니다.

AWS IoT SiteWise OPC UA 서버에서 데이터를 수신하지 않음

AWS IoT SiteWise 자산이 OPC UA 서버에서 전송한 데이터를 수신하지 않는 경우 SiteWise Edge 게이트웨이의 로그를 검색하여 문제를 해결할 수 있습니다. 다음 메시지가 포함된 정보 수준 swPublisher 로그를 찾습니다.

```
Emitting diagnostic name=PublishError.SomeException
```

로그의 *SomeException* 유형에 따라 다음 예외 유형 및 해당 문제를 사용하여 SiteWise Edge 게이트웨이 문제를 해결합니다.

- **ResourceNotFoundException** – OPC UA 서버에서 자산의 속성 별칭과 일치하지 않는 데이터를 보냅니다. 이 예외는 두 가지 경우에 발생할 수 있습니다.
 - 속성 별칭은 정의한 원본 접두사를 포함해 OPC UA 변수와 정확히 일치하지 않습니다. 속성 별칭 및 원본 접두사가 올바른지 확인합니다.
 - OPC UA 변수를 자산 속성에 매핑하지 않았습니다. 자세한 내용은 [에 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 단원을 참조하십시오.

원하는 모든 OPC UA 변수를 이미 매핑한 경우 SiteWise Edge 게이트웨이가 전송하는 OPC UA 변수를 필터링 AWS IoT SiteWise할 수 있습니다. 자세한 내용은 [SiteWise Edge에서 OPC UA 노트 필터 사용](#) 단원을 참조하십시오.

- **InvalidRequestException** – OPC UA 변수 데이터 형식이 자산 속성 데이터 형식과 일치하지 않습니다. 예를 들어 OPC UA 변수가 정수 데이터 형식인 경우, 해당 자산 속성은 정수 데이터 유형이어야 합니다. 배정밀도 형식의 자산 속성은 OPC UA 정수 값을 받을 수 없습니다. 이 문제를 해결하려면 새 속성을 올바른 데이터 형식으로 정의합니다.

- `TimestampOutOfRangeException` - SiteWise Edge 게이트웨이가 AWS IoT SiteWise 허용 범위를 벗어나는 데이터를 보내고 있습니다. AWS IoT SiteWise 는 타임스탬프가 과거 7일 이전이거나 향후 5분 이전인 모든 데이터 포인트를 거부합니다. SiteWise Edge 게이트웨이의 전원이 끊어지거나 AWS 클라우드에 연결되지 않은 경우 SiteWise Edge 게이트웨이의 캐시를 지워야 할 수 있습니다.
- `ThrottlingException` 또는 `LimitExceededException` - 요청이 수집된 데이터 포인트 비율 또는 자산 속성 데이터 API 작업에 대한 요청 비율과 같은 AWS IoT SiteWise 서비스 할당량을 초과했습니다. 구성이 [AWS IoT SiteWise 할당량](#)를 초과하지 않는지 확인합니다.

대시보드에 데이터가 표시되지 않음

대시보드에 데이터가 표시되지 않으면 SiteWise Edge 게이트웨이의 게시자 구성과 데이터 소스가 동기화되지 않을 수 있습니다. 동기화되지 않는 경우 데이터 소스 이름을 변경하면 클라우드에서 엣지로의 동기화가 빨라져 동기화되지 않음 오류가 수정될 수 있습니다.

데이터 소스의 이름을 업데이트하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 대시보드에 연결된 SiteWise Edge 게이트웨이를 선택합니다.
4. 데이터 소스에서 편집을 선택합니다.
5. 새 소스 이름을 선택하고 저장을 선택하여 변경 내용을 확인합니다.
6. 데이터 원본 테이블에서 데이터 원본 이름이 업데이트되었는지 확인하여 변경 내용을 확인합니다.

`/greengrass/v2/logs` 오류에서 `aws.iot.siteWiseEdgePublisher` 로그에 “메인 클래스를 찾거나 로드할 수 없음”이라는 메시지가 표시됨.

이 오류가 표시되면 SiteWise Edge 게이트웨이의 Java 버전을 업데이트해야 할 수 있습니다.

- 터미널에서 다음 명령을 실행합니다.

```
java -version
```

SiteWise Edge 게이트웨이가 실행 중인 Java 버전이 OpenJDK Runtime Environment 아래에 표시됩니다. 다음과 같은 응답이 표시됩니다.

```
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)
```

자바 버전 11.0.20.8.1을 실행하는 경우 IoT SiteWise 퍼블리셔 팩을 버전 2.4.1 이상으로 업데이트해야 합니다. 자바 버전 11.0.20.8.1만 영향을 받으며, 다른 자바 버전을 사용하는 환경에서는 IoT SiteWise Publisher 구성 요소의 이전 버전을 계속 사용할 수 있습니다. 구성 요소 팩 업데이트에 대한 자세한 내용은 [SiteWise Edge 게이트웨이 구성 요소 팩 버전 변경](#) 단원을 참조하세요.

'SESSION_TAKEN_OVER' 또는 'com.aws.greengrass.mqttclient.MqttClient 스폰서를 통해 메시지를 게시하지 못했으며 다시 시도할 것입니다.' 메시지가 표시됨 로그에서

/greengrass/v2/logs/greengrass.log의 로그에 SESSION_TAKEN_OVER가 포함된 경고 또는 com.aws.greengrass.mqttclient.MqttClient: Failed to publish the message via Spooler and will retry.가 포함된 오류가 표시되면 여러 디바이스의 여러 SiteWise Edge 게이트웨이에 동일한 구성 파일을 사용하려고 하는 것일 수 있습니다. 각 SiteWise Edge 게이트웨이는 AWS 계정에 연결하기 위해 고유한 구성 파일이 필요합니다.

'com.aws.greengrass.deployment.IotJobsHelper: 배포 작업을 찾을 수 없습니다.' 또는 '배포 결과가 이미 보고되었습니다.' 메시지가 표시됨 로그에서

/greengrass/v2/logs/greengrass.log의 로그에 com.aws.greengrass.deployment.IotJobsHelper: No deployment job found. 또는 Deployment result already reported.가 표시되면 동일한 구성 파일을 재사용하려고 하는 것일 수 있습니다.

다음과 같은 여러 해결 방법이 있습니다.

- 구성 파일을 재사용하려면 다음을 수행하세요.

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 재사용하려는 SiteWise Edge 게이트웨이를 선택합니다.
4. 업데이트 탭을 선택합니다.
5. 다른 게시자 버전을 선택하고 배포를 선택합니다.

[Siemens Industrial Edge의 게이트웨이 생성](#)의 단계에 따라 새 구성 파일을 생성합니다.

OPC UA 데이터 소스의 속성 그룹에서 타임스탬프 설정을 구성하려고 하면 'SYNC_FAILED' 상태가 표시됩니다.

버전 2.5.0 AWS IoT Greengrass 에서에 대한 OPC UA 수집기 구성 요소가 AWS IoT SiteWise 업데이트되면 새로운 타임스탬프 구성 옵션이 도입되었습니다. 디바이스의 타임스탬프 또는 서버의 타임스탬프를 사용할 수 있습니다. OPC UA 수집기 구성 요소의 이전 버전은이 옵션을 지원하지 않으며 동기화에 실패합니다.

두 가지 방법으로 실패한 데이터 소스 동기화 상태를 확인할 수 있습니다. 권장되는 방법은 IoT SiteWise OPC UA 수집기 구성 요소를 버전 2.5.0 이상으로 업그레이드하는 것입니다. 또는 타임스탬프를 로 설정한 경우 이전 OPC UA 수집기 구성 요소 버전을 계속 사용할 수 있습니다Source. IoT SiteWise OPC UA 수집기 구성 요소를 업그레이드하는 방법은 [섹션을 참조하세요구성 요소의 AWS IoT SiteWise 버전 업데이트](#). 모든 구성 요소의 최신 버전을 사용하는 것이 좋습니다.

Note

데이터 소스 동기화 상태가 실패해도 데이터 중단은 없습니다. 소스 데이터는 계속 들어갑니다 AWS IoT SiteWise. 구성이 AWS IoT Greengrass V2 배포의 IoT SiteWise OPC UA 수집기 구성 요소와 동기화되지 않습니다.

속성 그룹의 타임스탬프 구성을 변경하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.
2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 편집할 게이트웨이를 선택합니다.
4. 데이터 소스 섹션에서 동기화 실패 상태의 데이터 소스를 선택하고 편집을 선택합니다.
5. 고급 구성을 확장한 다음 그룹 설정을 확장합니다.
6. 타임스탬프에서 소스를 선택합니다. 소스를 선택하면 구성에서 timestampToReturn 속성이 제거됩니다. 이 설정을 사용하면 기본적으로 디바이스에서 데이터 소스 타임스탬프를 수집할 수 있으므로 데이터 소스가 IoT SiteWise OPC UA 수집기 구성 요소와 동기화될 수 있습니다.
7. 저장을 선택합니다.

변환된 데이터 유형이 포함되지 않음

지원되지 않는 OPC UA 데이터 형식을 문자열로 변환할 때 오류가 표시되는 경우 몇 AWS IoT SiteWise가 가능한 이유가 있습니다.

- 변환하려는 데이터 유형이 복잡한 데이터 유형입니다. 복잡한 데이터 유형은 지원되지 않습니다.
- Amazon S3를 사용한AWS IoT SiteWise Buffered를 대상으로 사용하면 전체 문자열 값이 Amazon S3 버킷에 푸시된 파일에 보존됩니다. 나중에 데이터를 수집하면 1 AWS IoT SiteWise 024바이트보다 긴 전체 문자열 값이 거부됩니다.

트러스트 스토어 문제

SiteWise Edge의 트러스트 스토어와 관련된 문제가 발생하면 다음 문제 해결 단계를 고려하세요.

- AWS IoT Greengrass 루트 CA 인증서가 있고 적절한 트러스트 스토어에 올바른 형식이 지정되어 있는지 확인합니다.
- Java KeyStore 암호가 올바르게 설정되고 SiteWise Edge 구성 요소에 액세스할 수 있는지 확인합니다.
- 사용자 지정 인증서(예: HTTPS 프록시용)가 올바른 형식(일반적으로 PEM)이고 트러스트 스토어로 올바르게 가져오는지 확인합니다.
- 트러스트 스토어에 올바른 파일 권한이 있고 SiteWise Edge 프로세스에 액세스할 수 있는지 확인합니다.
- SiteWise Edge 로그에서 트러스트 스토어 문제를 나타낼 수 있는 SSL/TLS 관련 오류를 검토합니다.
- 와 같은 도구를 사용하여 SSL/TLS 연결을 독립적으로 테스트openssl하여 트러스트 스토어 기능 확인

프록시 지원 설치 문제

프록시 구성 프로세스 중에 문제가 발생하면 다음 문제 해결 단계를 고려하세요.

- 프록시 URL의 형식이 올바르고 적절한 체계(http:// 또는 https://)가 포함되어 있는지 확인합니다.
- 특수 문자가 포함된 프록시 자격 증명이 URL로 인코딩되었는지 확인합니다.
- 비프록시 목록에 필요한 모든 로컬 주소 및 AWS 서비스 엔드포인트가 포함되어 있는지 확인합니다.
- HTTPS 프록시의 경우 제공된 CA 인증서가 PEM 형식인지 확인합니다.

- 설치 로그에서 문제의 원인을 나타낼 수 있는 특정 오류 메시지를 검토합니다.
- 프록시 연결을 독립적으로 테스트하여 올바르게 작동하는지 확인합니다.

에서 AWS IoT SiteWise Edge 애플리케이션 문제 해결 Siemens Industrial Edge

Siemens Industrial Edge 디바이스에서 AWS IoT SiteWise Edge 애플리케이션의 문제를 해결하려면 Siemens Industrial Edge Management 또는 Siemens Industrial Edge 디바이스(IED) 포털을 통해 애플리케이션의 로그에 액세스할 수 있습니다. 자세한 내용은 Siemens 설명서의 [로그 다운로드](#)를 참조하십시오.

내 데이터가에 표시되지 않음 AWS IoT SiteWise

- Databus 사용자에게 문제가 없고 Databus_Configuration의 체크 표시 아이콘이 회색이 아닌 녹색인지 확인합니다.
- Secure Storage가 포함된 버전에서 Siemens Industrial Edge Management를 실행하지 않을 수 있습니다. Siemens OS 버전을 업그레이드합니다. 자세한 내용은 [Siemens Secure Storage 및 AWS IoT SiteWise Edge 애플리케이션](#) 단원을 참조하십시오.

로그에 'Config file missing AWS_REGION'이 표시됩니다.

Siemens 로그에 Config file missing AWS_REGION이 표시되면 구성 파일의 JSON이 손상된 것입니다. 새 구성 파일을 생성해야 합니다. [Siemens Industrial Edge의 게이트웨이 생성](#)의 단계에 따라 새 구성 파일을 생성합니다.

엣지 게이트웨이 구성에 '동기화되지 않음' 오류 메시지가 표시됩니다.

배포가 완료된 후 Siemens Industrial Edge 게이트웨이에 Out of sync 오류 메시지가 표시되면 IoT SiteWise 게시자 구성 요소가 게이트웨이와 동기화되지 않은 것입니다. IoT SiteWise 게시자 구성 요소는 Siemens Industrial Edge 게이트웨이의 백그라운드에서 작동하여 MQTT 주제 기능을 제공합니다. iotsitewise:publisher:3 대신 기능 네임스페이스를 사용하도록 Siemens Industrial Edge 게이트웨이를 업그레이드했습니다 iotsitewise:publisher:2. 게시자의 최신 버전으로 업데이트하여 이 문제를 해결할 수 있습니다.

IoT SiteWise 게시자의 최신 버전으로 업그레이드하려면

1. [AWS IoT SiteWise 콘솔](#)로 이동합니다.

2. 탐색 창에서 엣지 게이트웨이를 선택합니다.
3. 편집할 Siemens Industrial Edge 게이트웨이를 선택합니다.
4. 엣지 기능 섹션에서 소프트웨어 버전 보기를 선택합니다.
5. 게시자 드롭다운 메뉴에서 IoT SiteWise 게시자의 최신 버전을 선택합니다.
6. 완료를 선택합니다.

Edge에서 오픈 소스 통합 문제 해결

이 섹션에서는 오픈 소스 도구를 SiteWise Edge와 통합할 때 발생할 수 있는 일반적인 문제에 대한 솔루션을 제공합니다.

Note

Node-RED®, InfluxDB® 및 Grafana®는 SiteWise Edge의 공급업체 또는 공급업체가 아닙니다.

연결 문제

Node-RED가 MQTT 브로커에 연결할 수 없음

MQTT 브로커가 실행 중이고 지정된 포트에서 액세스할 수 있는지 확인합니다. 네트워크 구성을 확인하고 브로커 주소가 올바른지 확인합니다.

MQTT 브로커 상태를 확인하려면 다음을 실행합니다.

```
docker ps | grep emqx
```

InfluxDB 연결 오류

인증 토큰이 유효하고 올바른 조직 및 버킷 이름을 지정했는지 확인합니다. InfluxDB가 실행 중이고 액세스할 수 있는지 확인합니다.

InfluxDB 상태를 확인하려면 다음을 실행합니다.

```
curl -I http://localhost:8086
```

Grafana가 InfluxDB에 연결할 수 없음

URL, 인증 토큰, 조직 및 버킷을 포함하여 Grafana의 InfluxDB 데이터 소스 구성이 올바른지 확인합니다.

데이터 흐름 문제

에 데이터가 표시되지 않음 AWS IoT SiteWise

Node-RED 흐름의 속성 별칭이 예상 형식과 일치하는지 확인합니다. MQTT 주제 구조가 올바른지, SiteWise Edge 게이트웨이가 MQTT 브로커로부터 데이터를 수신하도록 올바르게 구성되어 있는지 확인합니다.

InfluxDB에 저장된 SiteWise Edge 데이터 없음

Node-RED 보존 흐름이 올바르게 구성되었고 InfluxDB 라이터 노드에 적절한 버킷 및 측정 설정이 있는지 확인합니다. Node-RED 디버그 출력에서 오류가 있는지 확인합니다.

데이터 형식 오류

데이터 변환 함수가 형식 간에 데이터를 올바르게 변환하는지 확인합니다. Node-RED 디버그 노드를 사용하여 흐름의 각 단계에서 데이터를 검사합니다.

성능 문제

높은 CPU 또는 메모리 사용량

리소스 사용량을 모니터링하고 필요에 따라 구성 요소의 구성을 조정합니다. 데이터 수집 빈도를 줄이거나 데이터 필터링을 구현하여 처리 부하를 줄이는 것이 좋습니다.

리소스 사용량을 모니터링하려면 다음을 실행합니다.

```
docker stats
```

느린 Grafana 대시보드 로드

InfluxDB 쿼리를 최적화하고 대시보드 패널에 시간 범위 제한을 추가하는 것이 좋습니다. 적절한 집계 함수를 사용하여 표시되는 데이터 포인트 수를 줄입니다.

로깅 및 진단

문제를 해결하려면 각 구성 요소의 로그를 확인합니다.

노드-빨간색 로그

Node-RED 콘솔에서 로그를 보거나 다음을 실행합니다.

```
docker logs node-red
```

InfluxDB 로그

다음을 실행하여 로그에 액세스합니다.

```
docker logs influxdb
```

Grafana 로그

다음을 실행하여 로그를 봅니다.

```
docker logs grafana
```

SiteWise Edge 로그

SiteWise Edge 게이트웨이 로그에서 MQTT 연결 및 데이터 처리 문제를 확인합니다. 자세한 내용은 [SiteWise Edge 게이트웨이 문제 해결](#) 단원을 참조하십시오.

AWS IoT Greengrass 문제 해결

SiteWise Edge 게이트웨이를 구성하거나 배포하는 데 발생하는 여러 문제에 대한 해결 방법은 AWS IoT Greengrass 개발자 안내서의 [문제 해결을 AWS IoT Greengrass](#) AWS IoT Greengrass참조하세요.

AWS IoT SiteWise 포털 문제 해결

AWS IoT SiteWise 포털의 일반적인 문제를 해결합니다.

사용자와 관리자가 AWS IoT SiteWise 포털에 접근할 수 없음

사용자 또는 관리자가 AWS IoT SiteWise 포털에 액세스할 수 없는 경우 로그인 성공을 방해하는 연결된 AWS Identity and Access Management (IAM) 정책에서 제한된 권한이 있을 수 있습니다.

로그인 실패로 이어지는 IAM 정책의 다음 예를 참조하세요.

 Note

"Condition" 요소가 포함된 IAM 정책을 첨부하면 로그인에 실패합니다.

예 1: 이 조건은 제한된 IP이며, 이로 인해 로그인에 실패합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:DescribePortal"
      ],
      "Resource": "*",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": [
            "REPLACESAMPLEIP"
          ]
        }
      }
    }
  ]
}
```

예 2: 이 조건은 포함된 태그이며, 이로 인해 로그인에 실패합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:DescribePortal"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
```

```

    "aws:ResourceTag/project": "*"
  }
}
]
}

```

포털에 사용자나 관리자를 추가할 때 제한된 IP와 같이 사용자 권한을 제한하는 IAM 정책을 생성하지 마세요. 권한이 제한된 연결된 정책은 AWS IoT SiteWise 포털에 연결할 수 없습니다.

AWS IoT SiteWise 규칙 작업 문제 해결

에서 AWS IoT SiteWise 규칙 작업의 문제를 해결하려면 다음 절차 중 하나를 수행할 AWS IoT Core 수 있습니다.

- Amazon CloudWatch Logs 구성
- 규칙에 대한 재게시 오류 작업 구성

그 다음 오류 메시지를 이 주제의 오류와 비교하여 문제를 해결합니다.

주제

- [AWS IoT Core 로그 구성](#)
- [재게시 오류 작업 구성](#)
- [규칙 문제 해결](#)
- [규칙 문제 해결\(AWS IoT SiteWise\)](#)
- [규칙 문제 해결\(DynamoDB\)](#)

AWS IoT Core 로그 구성

CloudWatch Logs에 다양한 수준의 정보를 로깅 AWS IoT 하도록을 구성할 수 있습니다.

CloudWatch Logs를 구성하고 액세스하려면 다음을 수행하세요.

1. 에 대한 로깅을 구성하려면 AWS IoT 개발자 안내서의 [CloudWatch Logs를 사용한 모니터링을](#) AWS IoT Core참조하세요.
2. [CloudWatch 콘솔](#)로 이동합니다.

3. 탐색 창에서 로그 그룹을 선택합니다.
4. AWSIoTLogs 그룹을 선택합니다.
5. 최근 로그 스트림을 선택합니다. 기본적으로 CloudWatch는 가장 최근 로그 스트림을 먼저 표시합니다.
6. 로그 항목을 선택하여 로그 메시지를 확장합니다. 로그 항목은 다음 스크린샷처럼 보일 수 있습니다.

CloudWatch > Log Groups > AWSIoTLogs > 9ca6614a-00fc-4f9e-8100-5c2a34918e90_123456789012_0

Expand all ☒ Row ☐ Text   

Filter events all 2020-02-10 (19:36:11) ▾

Time (UTC +00:00)	Message
2020-02-11	No older events found at the moment. Retry .
19:36:11	2020-02-11 19:36:11.823 TRACEID:d4cd3bd0-ac41-cd4a-4f59-74a242ec70e6 PRINCIPALID:AIDAZ2YMUHYHIEDEL3VA3 [ERROR] EVENT:IotSiteWiseActionFailure TOPICNAME:/tutorial/device/SiteWiseTutorialDevice1/cpu CLIENTID:iotconsole-1581444173801-0 MESSAGE:Failed to send message data to IoT SiteWise asset properties. [Code: InvalidRequestException, Message: Property value does not match data type DOUBLE]. Message arrived on: /tutorial/device/SiteWiseTutorialDevice1/cpu, Action: iotSiteWise
	No newer events found at the moment. Retry .

7. 오류 메시지를 이 주제의 오류와 비교하여 문제를 해결합니다.

재게시 오류 작업 구성

오류 메시지를 처리하도록 규칙에 대한 오류 작업을 구성할 수 있습니다. 이 절차에서는 MQTT 테스트 클라이언트에서 오류 메시지를 보기 위해 재게시 규칙 작업을 오류 작업으로 구성합니다.

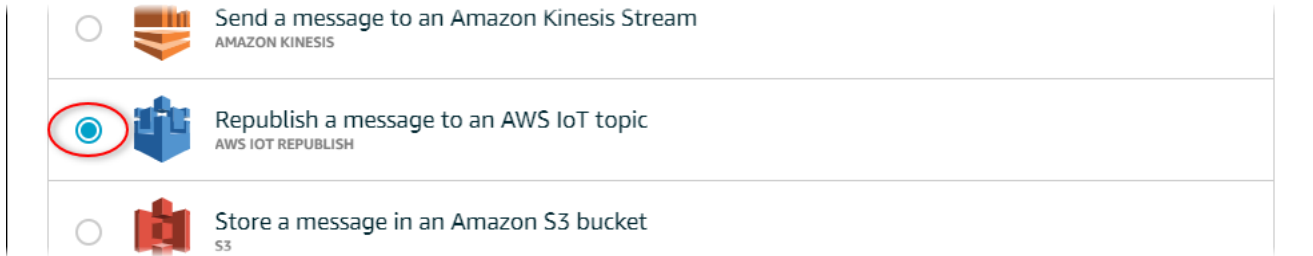
Note

재게시 오류 작업은 ERROR 수준 로그와 동일한 로그만 출력합니다. 상세 로그를 보려면 [CloudWatch Logs를 구성](#)해야 합니다.

규칙에 재게시 오류 작업 추가

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 작업을 선택한 다음 규칙을 선택합니다.
3. 규칙을 선택합니다.
4. 오류 작업에서 작업 추가를 선택합니다.

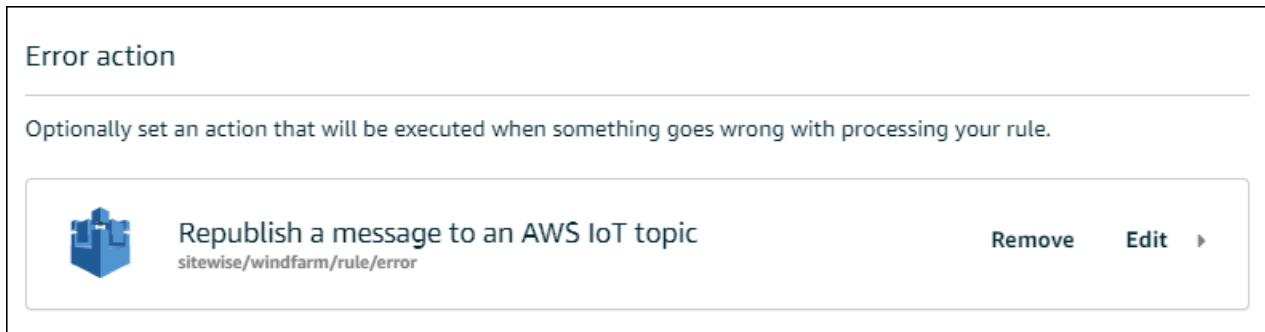
5. AWS IoT 주제에 메시지 다시 게시를 선택합니다.



6. 페이지 하단에서 작업 구성을 선택합니다.

- 주제에서 고유한 주제(예: **sitewise/windfarm/rule/error**)를 입력합니다. AWS IoT Core 는이 주제에 오류 메시지를 다시 게시합니다.
- 선택을 선택하여 오류 작업을 수행할 수 있는 AWS IoT Core 액세스 권한을 부여합니다.
- 규칙에 대해 생성한 역할 옆에 있는 선택을 선택합니다.
- 역할 업데이트를 선택하여 역할에 권한을 더 추가합니다.
- 작업 추가를 선택합니다.

규칙의 오류 작업은 다음 스크린샷과 비슷해야 합니다.



12. 콘솔 왼쪽 상단에서 뒤로 화살표를 선택하여 AWS IoT 콘솔 홈으로 돌아갑니다.

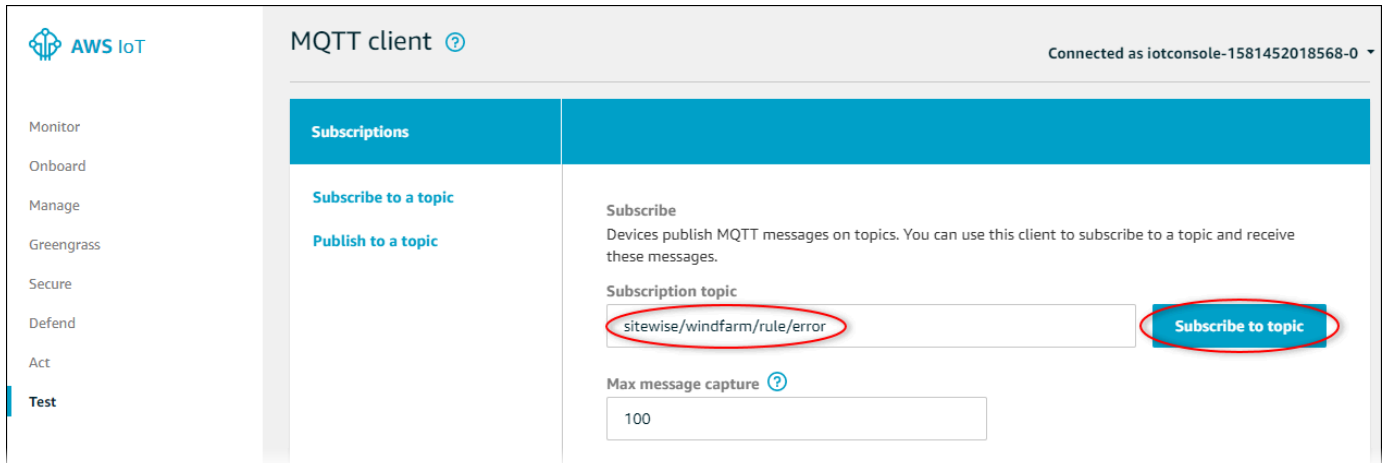
재게시 오류 작업을 설정한 후에는 AWS IoT Core의 MQTT 테스트 클라이언트에서 오류 메시지를 볼 수 있습니다.

다음 절차에서는 MQTT 테스트 클라이언트에서 오류 주제를 구독합니다. MQTT 테스트 클라이언트에서 규칙의 오류 메시지를 받아 문제를 해결할 수 있습니다.

오류 작업 주제 구독

- [AWS IoT 콘솔](#)로 이동합니다.
- 왼쪽 탐색 페이지에서 테스트를 선택하여 MQTT 테스트 클라이언트를 엽니다.

- 구독 주제 필드에 이전에 구성한 오류 주제(예: **sitewise/windfarm/rule/error**)를 입력하고 구독을 선택합니다.



- 오류 메시지가 나타나는지 확인한 다음 오류 메시지에서 failures 배열을 확장합니다.
그런 다음 오류 메시지를 이 주제의 오류와 비교하여 문제를 해결합니다.

규칙 문제 해결

다음 정보를 사용하여 규칙 문제를 해결합니다.

문제

- 오류: 멤버는 현재 타임스탬프 전 604800초, 후 300초 이내에 있어야 합니다.
- 오류: 속성 값이 데이터 유형 <type>이 일치하지 않습니다.
- 오류: 사용자 <role-arn>이 수행할 권한이 없음: iotsitewise:BatchPutAssetPropertyValue on resource
- 오류: iot.amazonaws.com이 수행할 수 없음: sts:AssumeRole on resource: <role-arn>
- 정보: 요청이 전송되지 않았습니다. 대체 템플릿을 수행한 후 PutAssetPropertyValueEntries가 비어 있습니다.

오류: 멤버는 현재 타임스탬프 전 604800초, 후 300초 이내에 있어야 합니다.

타임스탬프는 현재 Unix 에포크 시간과 비교하여 7일 이전 또는 5분 이후입니다. 다음을 시도해 보세요.

- 타임스탬프가 Unix 에포크 시간(UTC)인지 확인합니다. 다른 시간대로 타임스탬프를 제공하면 이 오류가 표시됩니다.

- 타임스탬프가 초 단위인지 확인합니다. 타임스탬프가 초 단위의 시간(Unix epoch 시간)으로 분할되고 나노초 단위로 오프셋될 것으로 AWS IoT SiteWise 예상합니다.
- 지난 7일 이내로 타임스탬프가 지정된 데이터를 업로드하고 있는지 확인합니다.

오류: 속성 값이 데이터 유형 <type>이 일치하지 않습니다.

규칙 작업의 항목에는 대상 자산 속성이 아닌 다른 데이터 유형이 있습니다. 예를 들어 대상 자산 속성이 DOUBLE이고 선택한 데이터 유형이 정수이거나 integerValue에서 값을 전달했습니다. 다음을 시도해 보세요.

- AWS IoT 콘솔에서 규칙을 구성하는 경우 각 항목에 대해 올바른 데이터 유형을 선택했는지 확인합니다.
- API 또는 AWS Command Line Interface (AWS CLI)에서 규칙을 구성하는 경우 value 객체가 올바른 유형 필드(예: DOUBLE 속성doubleValue)를 사용하는지 확인합니다.

오류: 사용자 <role-arn>이 수행할 권한이 없음:

iotsitewise:BatchPutAssetPropertyValue on resource

규칙이 대상 자산 속성에 액세스할 권한이 없거나 대상 자산 속성이 존재하지 않습니다. 다음을 시도해 보세요.

- 속성 별칭이 올바른지와 지정된 속성 별칭이 있는 자산 속성이 있는지 확인합니다. 자세한 내용은 [예 대한 데이터 스트림 관리 AWS IoT SiteWise](#) 단원을 참조하십시오.
- 규칙에 역할이 있는지와 역할이 대상 자산 속성(예: 대상 자산 계층 구조)에 대한 iotsitewise:BatchPutAssetPropertyValue 권한을 허용하는지 확인합니다. 자세한 내용은 [AWS IoT 필요한 액세스 권한 부여](#) 단원을 참조하세요.

오류: iot.amazonaws.com이 수행할 수 없음: sts:AssumeRole on resource: <role-arn>

사용자는 AWS Identity and Access Management (IAM)에서 규칙의 역할을 수임할 권한이 없습니다.

사용자에게 규칙의 역할에 대한 iam:PassRole 권한이 허용되는지 확인합니다. 자세한 내용은 AWS IoT 개발자 가이드의 [역할 전달 권한](#) 단원을 참조하세요.

정보: 요청이 전송되지 않았습니다. 대체 템플릿을 수행한 후 PutAssetPropertyValueEntries가 비어 있습니다.

 Note

이 메시지는 INFO 수준 로그입니다.

요청에 모든 필수 파라미터가 포함된 항목이 하나 이상 있어야 합니다.

대체 템플릿을 포함한 규칙의 파라미터가 비어 있지 않은 값을 생성하는지 확인합니다. 대체 템플릿은 규칙 쿼리 문의 AS 절에 정의된 값에 액세스할 수 없습니다. 자세한 내용은 AWS IoT 개발자 가이드의 [대체 템플릿](#)을 참조하세요.

규칙 문제 해결(AWS IoT SiteWise)

CPU 및 메모리 사용량 데이터가 예상 AWS IoT SiteWise 대로에 표시되지 않는 경우 이 절차의 단계에 따라 규칙 문제를 해결합니다. 이 절차에서는 MQTT 테스트 클라이언트에서 오류 메시지를 보기 위해 재게시 규칙 작업을 오류 작업으로 구성합니다. 문제 해결을 위해 CloudWatch 로깅에 로깅을 구성할 수도 있습니다. 자세한 내용은 [AWS IoT SiteWise 규칙 작업 문제 해결](#) 단원을 참조하세요.

규칙에 재게시 오류 작업 추가

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 메시지 라우팅을 선택한 다음 규칙을 선택합니다.
3. 이전에 생성한 규칙을 선택하고 편집을 선택합니다.
4. 오류 작업 - 선택 사항에서 오류 작업 추가를 선택합니다.
5. AWS IoT 주제에 메시지 다시 게시를 선택합니다.
6. 주제에서 오류 경로(예: **sitewise/rule/tutorial/error**)를 입력합니다. AWS IoT Core 는 이 주제에 오류 메시지를 다시 게시합니다.
7. 앞서 생성한 역할(예: SiteWiseTutorialDeviceRuleRole)을 선택합니다.
8. 업데이트를 선택합니다.

재게시 오류 작업을 설정한 후에는 AWS IoT Core의 MQTT 테스트 클라이언트에서 오류 메시지를 볼 수 있습니다.

다음 절차에서는 MQTT 테스트 클라이언트에서 오류 주제를 구독합니다.

오류 작업 주제 구독

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 페이지에서 MQTT 테스트 클라이언트를 선택하여 MQTT 테스트 클라이언트를 엽니다.
3. 주제 필터 필드에서 **sitewise/rule/tutorial/error**를 입력한 다음 구독을 선택합니다.

오류 메시지가 나타나면 오류 메시지에서 failures 배열을 보고 문제를 진단합니다. 가능한 문제 및 해결 방법에 대한 자세한 내용은 [AWS IoT SiteWise 규칙 작업 문제 해결](#) 단원을 참조하세요.

오류가 나타나지 않는 경우 규칙이 활성화되어 있고 오류 재게시 작업에서 구성한 것과 동일한 주제를 구독했는지 확인합니다. 그런 후에도 오류가 계속 나타나지 않으면 디바이스 스크립트가 실행 중이고 디바이스 새도우를 성공적으로 업데이트하는지 확인하세요.

Note

디바이스의 새도우 업데이트 주제를 구독하여 AWS IoT SiteWise 작업에서 구문 분석하는 페이로드를 볼 수도 있습니다. 이렇게 하려면 다음 주제를 구독하세요.

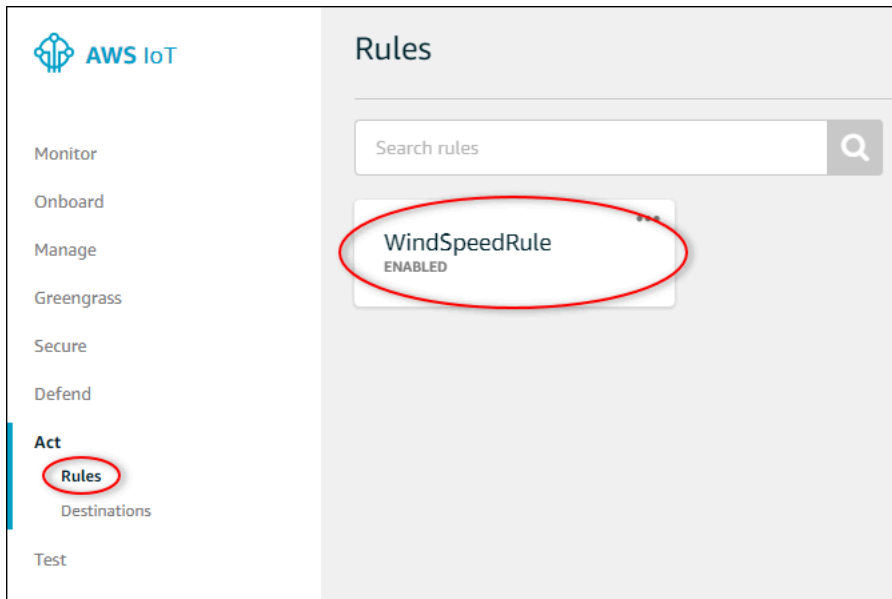
```
$aws/things/+/shadow/update/accepted
```

규칙 문제 해결(DynamoDB)

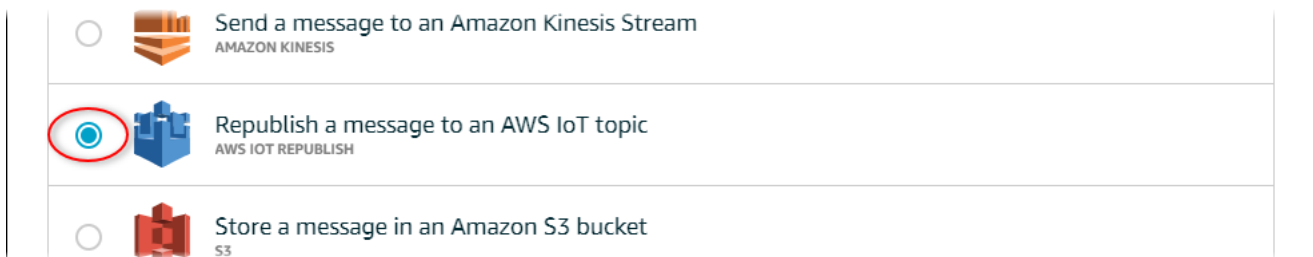
데모 자산 데이터가 예상대로 DynamoDB 테이블에 표시되지 않는 경우 이 절차의 단계에 따라 규칙 문제를 해결합니다. 이 절차에서는 MQTT 테스트 클라이언트에서 오류 메시지를 보기 위해 재게시 규칙 작업을 오류 작업으로 구성합니다. 문제 해결을 위해 CloudWatch 로깅에 로깅을 구성할 수도 있습니다. 자세한 내용을 알아보려면 AWS IoT 개발자 가이드의 [CloudWatch Logs를 통해 모니터링](#)을 참조하세요.

규칙에 재게시 오류 작업 추가

1. [AWS IoT 콘솔](#)로 이동합니다.
2. 왼쪽 탐색 창에서 작업을 선택한 다음 규칙을 선택합니다.
3. 앞서 생성한 규칙을 선택합니다.



4. 오류 작업에서 작업 추가를 선택합니다.
5. AWS IoT 주제에 메시지 다시 게시를 선택합니다.



6. 페이지 하단에서 작업 구성을 선택합니다.
7. 주제에 **windspeed/error**를 입력합니다. AWS IoT Core는이 주제에 오류 메시지를 다시 게시합니다.

Configure action


Republish a message to an AWS IoT topic
AWS IOT REPUBLISH

This action will republish the message to another AWS IoT topic.

*Topic 

windspeed/error

Quality of Service 

☒ 0 - The message is delivered zero or more times.

☐ 1 - The message is delivered one or more times.

Choose or create a role to grant AWS IoT access to perform this action.

No role selected

Create Role

Select

Cancel

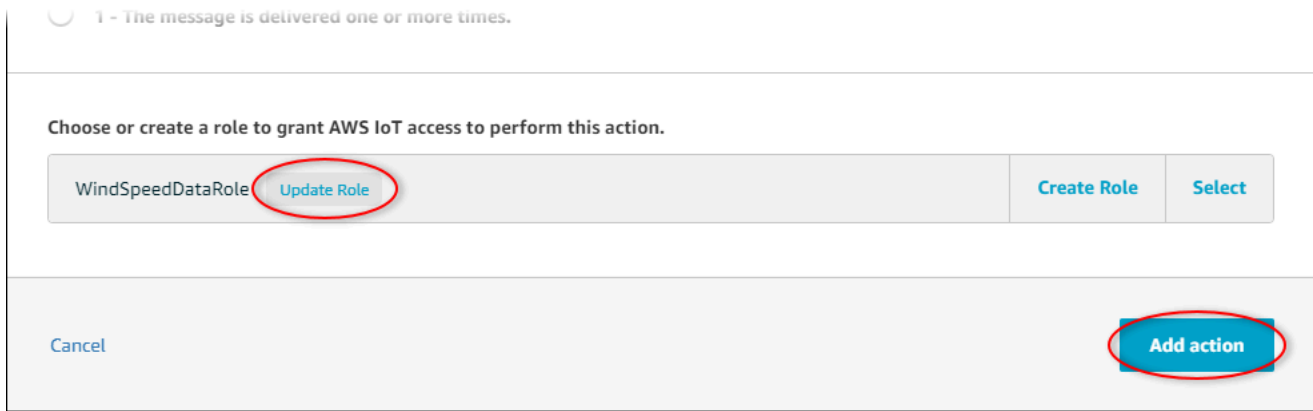
Add action

- 선택을 선택하여 이전에 생성한 역할을 사용하여 오류 작업을 수행할 수 있는 AWS IoT 코어 액세스 권한을 부여합니다.
- 해당 역할 옆에 있는 선택을 선택합니다.

Choose or create a role to grant AWS IoT access to perform this action.

No role selected	Refresh	Create Role	Close
Search for IAM roles			
WindSpeedDataRole	Select		

- 역할 업데이트를 선택하여 역할에 권한을 더 추가합니다.



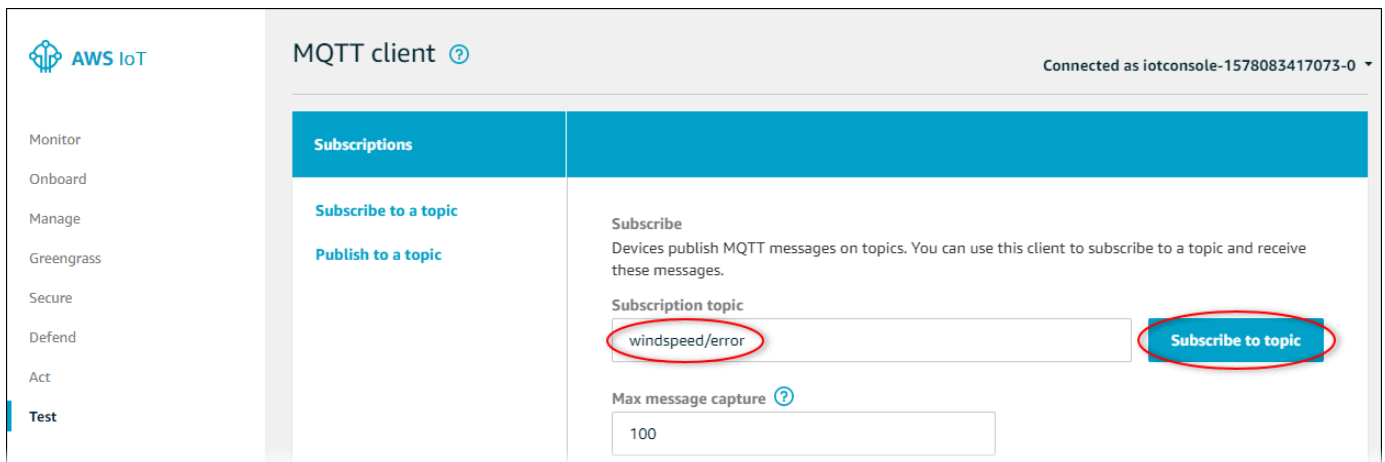
11. 작업 추가를 선택하여 오류 작업 추가를 완료합니다.
12. 콘솔 왼쪽 상단의 뒤로 화살표를 선택하여 AWS IoT 코어 콘솔 홈으로 돌아갑니다.

재게시 오류 작업을 설정한 후 AWS IoT 코어의 MQTT 테스트 클라이언트에서 오류 메시지를 볼 수 있습니다.

다음 절차에서는 MQTT 테스트 클라이언트에서 오류 주제를 구독합니다.

오류 작업 주제 구독

1. AWS IoT 코어 콘솔의 왼쪽 탐색 페이지에서 테스트를 선택합니다.
2. 구독 주제 필드에 **windspeed/error**를 입력하고 주제 구독을 선택합니다.



3. 오류 메시지가 나타나는지 확인하고 오류 메시지에서 failures 배열을 탐색하여 다음과 같은 일반적인 문제를 진단합니다.
 - 규칙 쿼리 문의 오타
 - 역할 권한 부족

오류가 나타나지 않는 경우 규칙이 활성화되어 있고 오류 재게시 작업에서 구성한 것과 동일한 주제를 구독했는지 확인합니다. 그래도 오류가 나타나지 않으면 데모 풍력 발전소 자산이 여전히 존재하고 풍속 속성에 대한 알림을 활성화했는지 확인합니다. 데모 자산이 만료되고 사라진 경우 새 데모를 생성하고 규칙 쿼리 문을 업데이트하여 업데이트된 자산 모델 및 속성 ID를 반영 AWS IoT SiteWise할 수 있습니다. IDs

일괄 가져오기 및 내보내기 작업 문제 해결

전송 작업 중에 발생한 오류를 처리하고 진단하려면 AWS IoT TwinMaker GetMetadataTransferJob API를 참조하세요.

1. 전송 작업을 생성하고 실행한 후 GetMetadataTransferJob API를 직접적으로 호출합니다.

```
aws iottwinmaker get-metadata-transfer-job \  
--metadata-transfer-job-id your_metadata_transfer_job_id \  
--region us-east-1
```

2. 작업 상태가 다음 상태 중 하나로 변경됩니다.

- COMPLETED
- 취소됨
- ERROR

3. GetMetadataTransferJob API는 [MetadataTransferJobProgress](#) 객체를 반환합니다.

4. MetadataTransferJobProgress 객체는 다음 파라미터를 포함합니다.

- failedCount: 전송 프로세스 중에 실패한 자산 수를 나타냅니다.
- skippedCount: 전송 프로세스 중에 건너뛴 자산 수를 나타냅니다.
- succeededCount: 전송 프로세스 중에 성공한 자산 수를 나타냅니다.
- totalCount: 전송 프로세스와 관련된 총 자산 수를 나타냅니다.

5. 또한 미리 서명된 URL이 포함된 API 직접 호출을 통해 reportUrl 요소가 반환됩니다. 전송 작업에 조사가 필요한 오류가 있는 경우 이 URL에서 전체 오류 보고서를 다운로드할 수 있습니다.

AWS IoT SiteWise 엔드포인트 및 할당량

다음 섹션에서는 엔드포인트 및 할당량에 대해 설명합니다 AWS IoT SiteWise.

주제

- [AWS IoT SiteWise 엔드포인트](#)
- [AWS IoT SiteWise 할당량](#)

AWS IoT SiteWise 엔드포인트

AWS 일반 참조 가이드에는의 AWS IoT SiteWise 엔드포인트가 나열되어 있습니다 AWS 계정. 자세한 내용은 AWS 일반 참조 사용 설명서의 [AWS IoT SiteWise 엔드포인트 및 할당량](#)을 참조하세요.

AWS IoT SiteWise 할당량

다음 표에서는의 할당량을 설명합니다 AWS IoT SiteWise. 할당량과 할당량 증가를 요청하는 방법에 대한 자세한 내용은 AWS 일반 참조의 [AWS Service Quotas](#)을 참조하세요. 할당 AWS IoT SiteWise 량에 대한 자세한 내용은 [AWS IoT SiteWise 의 서비스 할당량을 참조하세요](#) AWS 일반 참조.

AWS IoT SiteWise 자산 및 자산 모델의 할당량

리소스	설명	할당량	조정 가능
각의 자산 모델 수 AWS 리전 AWS 계정	에 AWS 리전 대해에 서 생성할 수 있는 최대 자산 모델 수입니다 AWS 계정.	1000	예
각 자산 모델의 자산 수	각 자산 모델에 대해 생성할 수 있는 최대 자산 수입니다.	10,000	예
각 상위 자산의 하위 자산 수	상위 자산과 연결할 수 있는 최대 하위 자산 수입니다.	2000	예

리소스	설명	할당량	조정 가능
자산 모델 계층 트리 깊이	자산 모델의 최대 자산 계층 구조 트리 깊이입니다.	30	예
각 자산 모델의 계층 구조 정의 수	자산 모델에 포함할 수 있는 최대 계층 정의 수입니다.	30	예
각 자산 모델의 루트 수준에 있는 속성 수	각 자산 모델의 최대 수 <code>assetModelProperties</code> 입니다. 이 개수에는 <code>compositeModelProperties</code> 가 포함되지 않습니다. 이 할당량은 이 자산 모델에서 생성된 모든 고유 자산에도 적용됩니다.	500	예
자산 모델의 속성 수	<code>ASSET_MODEL</code> 또는 <code>COMPONENT_MODEL</code> 유형 자산 모델의 최대 속성 수입니다. 이 수는 루트 자산 모델의 속성과 포함된 구성 요소 모델 기반 또는 인라인 복합 모델을 결합하여 결정됩니다. 이 할당량은 이 자산 모델에서 생성된 모든 고유 자산에도 적용됩니다.	5000	예

리소스	설명	할당량	조정 가능
각 복합 모델의 속성 수	복합 모델에 허용되는 최대 속성 수입니다. 또한 COMPONENT_MODEL 유형의 자산 모델에 허용되는 최대 속성 수입니다.	100	예
자산 모델의 속성 트리 깊이	예를 들어 측정 속성 A를 사용하는 변환 속성 B를 사용하는 변환 속성 C가 있는 모델의 깊이는 3입니다.	10	아니요
각 계층 구조 트리의 자산 모델 수	단일 계층 구조 트리에 포함할 수 있는 최대 자산 모델 수입니다.	100	예
자산 모델의 직접 종속 속성 수	이 할당량은 속성 공식 표현식에 정의된 대로 단일 속성에 직접 종속될 수 있는 속성의 수를 제한합니다. 종속 속성 수는 자산 모델의 직접 종속 속성 수보다 커야 합니다. 자산 모델의 종속 속성보다 직접 종속 속성이 더 많은 경우 두 할당량에 대한 증가를 요청합니다.	20	예

리소스	설명	할당량	조정 가능
자산 모델의 종속 속성 수	이 할당량은 속성 공식 표현식에 정의된 대로 단일 속성에 직접 또는 간접적으로 종속될 수 있는 속성의 수를 제한합니다.	30	아니요
자산 모델의 복합 모델 수	단일 자산 모델에 포함할 수 있는 최대 복합 모델 수입입니다.	50	예
복합 모델 깊이	인라인 및 component-model-based 복합 모델을 포함하여 각 자산 모델에서 복합 모델 트리의 최대 깊이입니다.	2	예
동일한 구성 요소 모델을 사용하는 고유한 자산 모델 수	COMPONENT_MODEL 유형의 특정 자산 모델을 직접 참조하는 구성 요소 모델 기반 복합 모델이 하나 이상 있는 고유 자산 모델의 최대 수입입니다.	20	예
속성 수식 표현식의 속성 변수 수	예를 들어, avg(power) + max(temp) 표현식에는 power 및 temp의 두 개의 속성 변수가 있습니다. 이는 변환 계산 결과에도 적용됩니다.	10	아니요

리소스	설명	할당량	조정 가능
속성 수식 표현식의 함수 수	예를 들어, <code>avg(power) + max(temp)</code> 표현식에는 <code>avg</code> 및 <code>max</code> 의 두 개의 함수가 있습니다.	10	아니요

AWS IoT SiteWise 자산 속성 데이터의 할당량

리소스	설명	할당량	조정 가능
자산 속성 데이터 API 작업에 대한 요청 속도	각의 각에서 수행할 수 있는 초당 최대 자산 속성 데이터 API 요청 수 AWS 리전입니다 AWS 계정. 이 할당량은 <code>GetAssetPropertyValue</code> 및 <code>BatchPutAssetPropertyValue</code> 와 같은 API 작업에 적용됩니다.	1000	예
각 자산 속성의 각 데이터 품질에 대한 초당 데이터 포인트 수	이 할당량은 각 자산 속성의 각 데이터 품질에 대해 초 단위로 타임스탬프가 동일한 최대 타임스탬프 <code>timestamp-quality-value(TQV)</code> 데이터 포인트 수에 적용됩니다. 각 자산 속성에 대해 1 초 동안 품질이 우수하거나 불확실한 데이터, 품질이 좋지 않은 데이터	10	아니요

리소스	설명	할당량	조정 가능
	터 포인트를 이 수까지 저장할 수 있습니다.		
의 각 자산 속성에서 초당 수집된 BatchPutAssetProperty 항목 수 AWS 리전입니다 AWS 계정.	SiteWise Edge 게이트웨이, AWS IoT Core 규칙 및 API 호출을 포함한 모든 소스 BatchPutAssetProperty 의에 대한 각 자산 속성의 최대 항목 수입니다.	10	아니요
수집된 데이터 포인트의 비율	의 각에서 초당 수집된 timestamp-quality-value(TQV) 데이터 포인트의 최대 수 AWS 리전입니다 AWS 계정.	5000	예
BatchGetAssetPropertyAggregates 에 대한 요청 속도	각의 각에서 수행할 수 있는 초당 최대 BatchGetAssetPropertyAggregates 요청 수 AWS 리전입니다 AWS 계정.	200	예
BatchGetAssetProperty에 대한 요청 속도	각의 각에서 수행할 수 있는 초당 최대 BatchGetAssetProperty 요청 수 AWS 리전입니다 AWS 계정.	500	예

리소스	설명	할당량	조정 가능
BatchGetAssetsPropertyHistory에 대한 요청 속도	를 수행할 수 있는 초당 최대 BatchGetAssetsPropertyHistory 요청 수입니다.	200	예
의 각 자산 속성에 대해 초당 수집된 BatchPutAssetsProperty 항목 수 AWS 리전입니다 AWS 계정.	이 할당량은 SiteWise Edge 게이트웨이, AWS IoT Core 규칙 및 API 호출을 포함한 모든 소스 BatchPutAssetsProperty 의에 대한 각 자산 속성의 항목에 적용됩니다.	10	아니요
각 자산 속성에 대한 GetAssetPropertyAggregates 요청 및 BatchGetAssetPropertyAggregates 항목 쿼리 비율	각의 각에서 초당 각 자산 속성에 대한 최대 총 GetAssetPropertyAggregates 요청 및 BatchGetAssetPropertyAggregates 항목 수 AWS 리전입니다 AWS 계정.	50	아니요

리소스	설명	할당량	조정 가능
각 자산 속성에 대한 GetAssetPropertyValues 요청 및 BatchGetAssetPropertyValues 항목 쿼리 비율	각 자산 속성의 최대 총 GetAssetPropertyValues 요청 및 BatchGetAssetPropertyValues 항목 수는 각 자산 속성의 각 AWS 리전 에서 초당입니다 AWS 계정.	500	아니요
각 자산 속성에 대한 GetAssetPropertyValueHistory 요청 및 BatchGetAssetPropertyValueHistory 항목 쿼리 비율	각 자산 속성의 최대 총 GetAssetPropertyValueHistory 요청 및 BatchGetAssetPropertyValueHistory 항목 수는 각 자산 속성의 각 AWS 리전 에서 초당입니다 AWS 계정.	30	아니요
GetInterpolatedAssetPropertyValues 요청 비율	각의 각에서 수행할 수 있는 초당 최대 GetInterpolatedAssetPropertyValues 요청 수 AWS 리전 입니다 AWS 계정.	500	예

리소스	설명	할당량	조정 가능
각 GetInterpolatedAssessmentPropertyValues 요청의 결과 수	페이지가 매겨진 GetInterpolatedAssessmentPropertyValues 요청에 대해 반환할 최대 결과 수입니다.	10	예

리소스	설명	할당량	조정 가능
GetAssetPropertyValueHistory 및 BatchGetAssetPropertyHistory	AWS 계정 GetAssetPropertyValueHistory 및 BatchGetAssetPropertyHistory 에서 각 AWS 리전 에 대해 초당 검색된 데이터 포인트의 최대 바이트 속도(MB/초) 입니다BatchGetAssetPropertyHistory . 이 할당량에 대해 평가된 응답 페이로드는 각 데이터 포인트의 타임스탬프-품질-값(TQV) 필드를 사용하고 각 API 요청의 바이트 크기를 다음 4KB 증분으로 반올림합니다. 매초 검색되는 Timestamp-quality-value(TQV) 데이터 포인트는 데이터 유형에 따라 다릅니다. - 정수 - 초당 최대 5 백만 TQV - 더블 - 초당 최대 4 백만 TQV - 부울 - 초당 최대 600만 TQV	100	예

리소스	설명	할당량	조정 가능
	문자열 - 각 문자열 값 크기에 따라 달라 집니다.		

SiteWise Edge 게이트웨이의 할당량

리소스	설명	할당량	조정 가능
에 AWS 리전 대한 각 의 SiteWise Edge 게이트웨이 수 AWS 계정	에 AWS 리전 대해 에서 생성할 수 있는 SiteWise Edge 게이트웨이의 최대 수입니다 AWS 계정.	100	예
SiteWise Edge 게이트웨이의 OPC UA 소스 수	SiteWise Edge 게이트웨이에서 구성할 수 있는 최대 OPC UA 소스 수입니다.	100	아니요
SiteWise Edge 게이트웨이의 총 대상 수	SiteWise Edge 게이트웨이에서 구성할 수 있는 최대 대상 수입니다.	100	아니요

에 대한 할당량 AWS IoT SiteWise Monitor

리소스	설명	할당량	조정 가능
에 AWS 리전 대한 각 의 포털 수 AWS 계정	에 AWS 리전 대해에 서 생성할 수 있는 최대 SiteWise Monitor 포털 수입니다 AWS 계정.	100	예
포털의 프로젝트 수	SiteWise Monitor 포털 내에서 생성할 수 있는	100	예

리소스	설명	할당량	조정 가능
	최대 프로젝트 수입니다.		
프로젝트의 대시보드 수	SiteWise Monitor의 프로젝트 내에서 생성할 수 있는 최대 대시보드 수입니다.	100	예
프로젝트의 루트 자산 수	SiteWise Monitor에서 프로젝트에 추가할 수 있는 최상위 자산의 최대 수입니다.	1	아니요
대시보드의 시각화 수	SiteWise Monitor의 대시보드에 추가할 수 있는 최대 시각적 요소 (예: 차트, 그래프 또는 테이블) 수입니다.	10	예
각 대시보드 시각화의 지표 수	SiteWise Monitor의 대시보드에서 단일 시각화에 표시할 수 있는 최대 지표 또는 데이터 포인트 수입니다.	5	예
각 대시보드 시각화의 임계값 수	SiteWise Monitor의 대시보드에서 각 시각화에 대해 설정할 수 있는 최대 임계값 수준 수입니다.	12	아니요

메타데이터 AWS IoT SiteWise 대량 가져오기 및 내보내기 할당량

리소스	설명	할당량	조정 가능
대기열의 메타데이터 전송 작업 수	대기열의 최대 PENDING 메타데이터 전송 작업 수입니다.	10	예
메타데이터 전송 작업 가져오기 파일의 크기	최대 가져오기 파일 크기(MB)입니다.	100	예
작업의 AWS IoT SiteWise 가져오기 리소스 수	단일 작업의 최대 AWS IoT SiteWise 가져오기 리소스 수입니다. 리소스에는 자산과 자산 모델이 포함됩니다.	5000	예
작업의 AWS IoT SiteWise 내보내기 리소스 수	단일 작업의 최대 AWS IoT SiteWise 내보내기 리소스 수입니다. 리소스에는 자산과 자산 모델이 포함됩니다.	5000	예

데이터 AWS IoT SiteWise 대량 가져오기 할당량

리소스	설명	할당량	조정 가능
실행 중인 대량 가져오기 작업 수	동시에 실행할 수 있는 최대 대량 가져오기 작업 수입니다.	100	아니요
CSV 파일의 크기	대량 가져오기 작업의 최대 CSV 파일 크기 (GB)입니다.	10	아니요
압축되지 않은 parquet 파일의 크기	대량 가져오기 작업에서 압축되지 않은	256MB	아니요

리소스	설명	할당량	조정 가능
	Parquet 파일의 최대 파일 크기(MB)입니다.		
버퍼링된 수집을 위한 CSV 파일의 크기	대량 가져오기 작업에서 버퍼링된 수집을 사용할 때의 최대 CSV 파일 크기(MB)입니다.	256MB	아니요
압축되지 않은 parquet 행 그룹의 크기	압축되지 않은 Parquet 행 그룹의 최대 크기입니다.	64MB	아니요
parquet 파일의 고유 측정 수	parquet 파일의 최대 고유 측정 수입니다.	10000	아니요
버퍼링된 수집에 대한 과거 타임스탬프와 오늘 사이의 일수	버퍼링된 수집을 사용할 때 과거의 타임스탬프와 오늘 날짜 사이의 최대 일수입니다.	30	예
각 CreateBulkImportJobs 의 각 AWS 리전 에 대한 요청 속도 AWS 계정		10	예
각의 각 AWS 리전 에 ListBulkImportJobs 대한 요청 속도 AWS 계정		50	예
각의 각 AWS 리전 에 DescribeBulkImportJobs 대한 요청 속도 AWS 계정		50	예

AWS IoT SiteWise 어시스턴트 API 제한 할당량

AWS IoT SiteWise Assistant API 제한 한도 할당량

리소스	설명	할당량	조정 가능
InvokeAssistant 작업 요청 속도	에서 AWS IoT SiteWise InvokeAssistant API에 만들 수 있는 분당 최대 트랜잭션 수(TPM)입니다 AWS 계정. TPM 제한은 지원되는 모든 리전에 적용되며 일부 리전에서는 조정할 수 있습니다.	10	아니요

이상 감지 할당량

이상 탐지 할당량은 AWS IoT SiteWise 및 Amazon Lookout for Equipment 간에 공유됩니다. 자세한 내용은 [Lookout for Equipment 사용 할당량](#)을 참조하세요.

AWS IoT SiteWise 사용 설명서의 문서 기록

다음 표에서는이 릴리스에 대한 설명서를 설명합니다 AWS IoT SiteWise.

- API 버전: 2019-12-02

변경 사항	설명	날짜
사용자 이름과 암호를 사용하여 외부 애플리케이션을 EMQX 브로커에 연결	새 MQTT 지원 V3 게이트웨이를 생성할 때 기본적으로 배포되는 AWS IoT Greengrass EMQX 브로커에 사용자 이름과 암호를 사용하여 외부 애플리케이션을 연결하는 지원이 추가되었습니다.	2025년 5월 1일
SiteWise Edge에서 MQTT 지원 V3 게이트웨이 지원	새로운 기능 추가 및 더 이상 사용되지 않는 콘텐츠 제거 • MQTT 지원 V3 게이트웨이에 대한 지원이 추가되었습니다. Amazon S3를 사용하여 실시간 데이터 수집 AWS IoT SiteWise 또는 버퍼링된 데이터 수집을 포함하여 MQTT 주제를 구독하기 위해 경로 필터를 사용하여 대상 구성을 개선했습니다. • IoT SiteWise OPC UA 수집기의 버전 3.0.0과 IoT SiteWise 게시자 구성 요소의 버전 4.0.0이 릴리스되었습니다 AWS IoT Greengrass V2. • 자체 호스팅된 SiteWise Edge 게이트웨이의 이전 버	2025년 2월 26일

전을 클래식 스트림인 V2 게이트웨이로 변경했습니다.

- SiteWise Edge 설명서 AWS IoT Greengrass V1 에 대한 참조는 더 이상와 함께 사용할 수 없으므로 제거되었습니다 AWS IoT SiteWise.

[AWS IoT SiteWise Assistant 지원](#)

생성형 AI 기반 AWS IoT SiteWise 어시스턴트인 어시스턴트에 대한 지원이 추가되었습니다.

2024년 11월 18일

[SiteWise Edge APIs에 대해 구성 가능한 세션 제한 시간 추가](#)

및 SiteWise Edge APIs의 비활성 기간을 관리하기 위해 AWS OpsHub 구성 가능한 세션 제한 시간 설정이 추가되었습니다.

2024년 10월 31일

[SiteWise Edge APIs에 대해 구성 가능한 프록시 설정 추가](#)

SiteWise Edge 게이트웨이에 대한 HTTPS 프록시 지원을 활성화하는 트러스트 스토어 정보 관리가 추가되었습니다.

2024년 10월 31일

[SiteWise Edge APIs에 대해에서 CORS 활성화](#)

보안 도메인 간 웹 애플리케이션 액세스를 활성화하기 위해 SiteWise Edge APIs에 대한 CORS 지원이 추가되었습니다.

2024년 9월 30일

[CloudRail 및 Litmus Edge 파트너 데이터 소스 지원](#)

파트너 데이터 소스로서의 CloudRail 및 Litmus Edge 모두에 대한 지원이 추가되었습니다.

2024년 9월 5일

[Siemens Industrial Edge에서 SiteWise Edge를 실행하는 기능 정식 출시](#)

AWS IoT SiteWise 는 이제 Siemens Industrial Edge 디바이스에서 SiteWise Edge를 실행하는 일반 가용성을 지원합니다.

2024년 7월 24일

[OPC UA 데이터 소스의 타임스탬프 구성에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 OPC UA 데이터 소스에 대한 타임스탬프 구성을 지원합니다.

2024년 7월 24일

[OPC UA 데이터 소스의 데이터 유형 변환에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 지원되지 않는 OPC UA 데이터 형식에 대한 데이터 형식 변환을 지원합니다.

2024년 7월 24일

[Siemens Industrial Edge에서 SiteWise Edge의 미리 보기 실행에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 Siemens Industrial Edge 디바이스에서 SiteWise Edge의 미리 보기 실행을 지원합니다.

2023년 11월 26일

[웜 티어 스토리지에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 고객이 산업 데이터를 안전하게 저장하고 액세스할 수 있도록 하는 완전 관리형 스토리지 계층인 웜 스토리지를 지원합니다.

2023년 11월 15일

[사용자 정의 고유 식별자에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 자산, 자산 모델, 속성 및 계층 구조에 대한 사용자 정의 고유 식별자 사용을 지원합니다.

2023년 11월 15일

[산업 자산의 다변량 이상 감지에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 과거 및 실시간 장비 데이터를 Amazon Lookout for Equipment와 통합하여 산업 자산의 다변량 이상 탐지를 지원합니다.

2023년 11월 15일

[AWS IoT SiteWise에서 시계열 데이터의 비용 효율적이고 확장 가능한 수집에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 분석 사용 사례에 필요한 시계열 데이터의 비용 효율적이고 확장 가능한 수집을 지원합니다.

2023년 11월 15일

[일괄 가져오기, 내보내기 및 업데이트에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 산업 장비 메타데이터의 대량 가져오기, 내보내기 및 업데이트를 지원합니다.

2023년 11월 15일

[자산 모델 구성 요소에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 자산 모델 구성 요소를 지원하여 산업 고객이 재사용 가능한 구성 요소를 생성할 수 있도록 지원합니다.

2023년 11월 15일

[IoT 대시보드 애플리케이션에 대한 지원 추가](#)

AWS IoT SiteWise 는 이제 운영 데이터를 시각화하고 상호 작용할 수 있는 오픈 소스 대시보드 애플리케이션을 지원합니다.

2023년 11월 15일

[에 대한 서비스 연결 역할 업데이트 AWS IoT SiteWise](#)

AWS IoT SiteWise 에는 새 서비스 연결 역할이 있으며 AWS IoT TwinMaker 데이터베이스에 대해 메타데이터 검색 쿼리를 실행할 수 있습니다.

2023년 11월 6일

[AWS IoT SiteWise 데이터 스트림 리소스에 대한 태그 지정 업데이트](#)

데이터 스트림 리소스 태깅에 대한 지원이 추가되었습니다.

2022년 8월 18일

SiteWise Edge 게이트웨이 업데이트	이제 엣지에서 클라우드로 전송되는 데이터와 클라우드로 전송되는 순서를 제어하도록 게시자를 구성할 수 있습니다.	2022년 1월 12일
AWS IoT SiteWise 데모 업데이트	이제 데모를 사용하여 SiteWise 모니터 포털을 생성할 수 있습니다.	2022년 1월 10일
업데이트된 스토리지 관리	이제 보존 기간을 정의하여 히터에 데이터가 보관되는 기간을 제어할 수 있습니다.	2021년 11월 29일
추가된 데이터 스트림 관리에 대한 지원	이제 자산 모델 및 자산을 생성하기 AWS IoT SiteWise 전에 데이터를 수집할 수 있습니다.	2021년 11월 24일
업데이트된 자산 모델 계층 구조	하위 자산 모델을 여러 상위 자산 모델과 연결할 수 있습니다.	2021년 10월 28일
리전 출시	in AWS GovCloud(미국 서부) AWS IoT SiteWise 에서 시작되었습니다.	2021년 9월 29일
업데이트된 함수	추가된 기능 - 지표에서는 집계 함수와 임시 함수에 중첩 표현식을 사용할 수 있습니다. - 변환에서는 pretrigger() 함수를 사용하여 현재 변환 계산을 트리거한 속성 업데이트 이전에 변수 값을 검색할 수 있습니다.	2021년 8월 10일

사용자 지정 지표 시간 간격	지표의 사용자 지정 시간 간격 및 오프셋에 대한 지원이 추가되었습니다.	2021년 8월 3일
엣지 AWS IoT SiteWise 에서 사용	이제 엣지 프로세싱 기능이 일반적으로 이용 가능합니다.	2021년 7월 29일
Amazon S3로 데이터 내보내기	AWS IoT SiteWise 이제에서 Amazon S3로 데이터를 내보낼 수 있습니다.	2021년 7월 27일
VPC 엔드포인트(AWS PrivateLink)	이제 컨트롤 플레인 API 작업을 위한 인터페이스 VPC 엔드포인트를 정식으로 사용할 수 있습니다.	2021년 7월 15일
변형	이제 변환에서 여러 자산 속성 변수를 입력할 수 있습니다.	2021년 7월 8일
업데이트된 timestamp() 함수	이제 변환에서 변수를 함수에 대한 timestamp() 인수로 제공할 수 있습니다.	2021년 6월 16일
경보의 일반 사용 가능	이제 경보 기능이 일반적으로 이용 가능합니다.	2021년 5월 27일
Modbus-TCP 프로토콜 어댑터 버전 2 릴리스	Modbus-TCP 프로토콜 어댑터 커넥터 버전 2를 사용할 수 있습니다. 이 릴리스에서는 ASCII, UTF8 및 ISO8859 인코딩 소스 문자열에 대한 지원이 추가되었습니다.	2021년 5월 24일

[업데이트된 Service Quotas](#)[GetInterpolatedAssetPropertyValues](#)

2021년 4월 29일

[yValues](#) API에 [GetInterpolatedAssetPropertyValues](#) 요청 비율, [GetInterpolatedAssetPropertyValues](#) 요청당 결과 수, [GetInterpolatedAssetPropertyValues](#) 의 과거 시작 날짜와 오늘 간의 일수 등의 할당량을 추가했습니다.

[업데이트된 공식 표현식](#)

다음 연산자 및 함수가 추가되었습니다.

2021년 4월 22일

- [연산자](#) `<`, `>`, `<=`, `>=`, `==`, `!=`, `!`, `and`, `or`, `not`이 추가되었습니다.
- [비교 함수](#) `neq(x, y)`가 추가되었습니다.
- [문자열 함수](#) `join()`, `format()`, `f' '`가 추가되었습니다.

[VPC 엔드포인트\(AWS PrivateLink\)](#)

인터페이스 VPC 엔드포인트를 생성하여 Virtual Private Cloud(VPC)와 AWS IoT SiteWise 컨트롤 플레APIs 간에 프라이빗 연결을 설정하는 방법에 대한 정보가 추가되었습니다.

2021년 3월 16일

[IAM 페더레이션](#)

SiteWise Monitor 포털 관리자 및 사용자는 이제 IAM 보안 인증을 사용하여 할당된 포털에 로그인할 수 있습니다.

2021년 3월 16일

리전 출시	중국(베이징) AWS IoT SiteWise 에서 출시되었습니다.	2021년 2월 3일
IoT SiteWise 커넥터 버전 10 릴리스	IoT SiteWise 커넥터 버전 10이 릴리스되었습니다. 이 릴리스에서는 소스 연결이 끊겼다가 다시 설정되었을 때 처리를 개선하도록 StreamManager 를 구성하였습니다. 또한 이 버전은 SourceTimestamp 를 사용할 수 없을 때 ServerTimestamp 와 함께 OPC UA 값을 허용합니다.	2021년 1월 22일
날짜 및 시간 함수	AWS IoT SiteWise 는 이제 날짜 및 시간 함수를 지원합니다.	2021년 1월 21일
함수 구문	이제 함수에 UFCS(Uniform Function Call Syntax)를 사용할 수 AWS IoT SiteWise 있습니다.	2021년 1월 11일
Grafana와 통합	Grafana 대시보드에서 AWS IoT SiteWise 데이터를 시각화하는 방법에 대한 정보가 추가되었습니다.	2020년 12월 15일

[AWS IoT SiteWise 기능 릴리스](#)

이제 경보로 데이터를 모니터링하고 엣지에서 산업 데이터를 처리하며 Modbus TCP 및 이더넷/IP 소스를 SiteWise Edge 게이트웨이에 사용하고 수신 데이터를 데드밴드로 필터링하는 등의 작업을 수행할 수 있습니다.

2020년 12월 15일

- AWS IoT SiteWise에서 경보를 정의, 구성 및 응답하는 데 사용할 수 있는 [경보를 통한 데이터 모니터링](#) 섹션이 추가되었습니다.
- [엣지 디바이스](#)의 산업 데이터 처리를 구성하는 데 사용할 수 있는 엣지 프로세싱 섹션이 추가되었습니다.
- SiteWise Edge 게이트웨이 소스 문서에 [Modbus TCP 및 이더넷/IP](#) 섹션을 추가했습니다.
- 수신되는 산업 데이터를 보내는 위치를 사용자 지정하는 데 사용할 수 있는 [소스 대상](#) 섹션을 추가했습니다.
- 산업용 로컬 서버에서 SiteWise Edge 게이트웨이로 전송되는 데이터의 빈도와 유형을 제어하는 데 사용할 수 있는 [OPC UA 필터링](#) 섹션이 추가되었습니다.

[AWS IoT SiteWise 는 이제 고객 관리형 CMKs 지원합니다.](#)

AWS IoT SiteWise 는 이제 고객 관리형 CMKs를 사용한 암호화를 지원합니다.

2020년 11월 24일

IoT SiteWise 커넥터 버전 8 릴리스	IoT SiteWise 커넥터 버전 8이 출시되었습니다. 이 릴리스에서는 커넥터에 간헐적인 네트워크 연결이 발생할 때의 안정성이 개선되었습니다.	2020년 11월 19일
공식 표현식에 문자열 및 조건문 사용	변환 및 지표용 공식 표현식에서 문자열 및 조건부 함수를 사용하는 방법에 대한 정보가 추가되었습니다.	2020년 11월 16일
AWS IoT Greengrass 스트림 관리자를 사용하여 데이터 수집	AWS IoT Greengrass 엣지 디바이스를 사용하여 로컬 데이터 소스에서 대용량 IoT 데이터를 수집하는 방법에 대한 정보가 추가되었습니다.	2020년 9월 16일
VPC 엔드포인트(AWS PrivateLink)	인터페이스 VPC 엔드포인트를 생성하여 Virtual Private Cloud(VPC)와 AWS IoT SiteWise 데이터 APIs 간에 프라이빗 연결을 설정하는 방법에 대한 정보가 추가되었습니다.	2020년 9월 4일
IoT SiteWise 커넥터 버전 7 릴리스	IoT SiteWise 커넥터 버전 7이 릴리스되었습니다. 이 릴리스에서는 SiteWise Edge 게이트웨이 지표 관련 문제가 해결되었습니다.	2020년 8월 14일

[AWS IoT SiteWise 콘솔에서 IAM Identity Center 사용자 생성](#)

AWS IoT SiteWise 콘솔에서 IAM Identity Center 사용자를 생성하는 방법에 대한 정보가 추가되었습니다. 이제 새 포털 또는 기존 포털에 사용자를 할당할 때 IAM Identity Center 사용자를 생성할 수 있습니다. 이 기능을 사용하도록 [풍력 발전 단지 데이터 시각화 및 공유](#) 자습서를 업데이트했습니다. 이 변경으로 자습서의 단계 수가 줄어듭니다.

2020년 8월 4일

[SiteWise Edge 게이트웨이 문제 해결 개선](#)

SiteWise Edge 게이트웨이 문제 해결 방법 및 소스의 [OPC UA 클라이언트 인증서 내보내기](#) 방법에 대한 기타 정보가 추가되었습니다.

2020년 6월 18일

[콘솔 작업 설명서](#)

[산업 자산 모델링](#), [자산 속성 데이터 쿼리](#) 및 [기타 서비스와 상호 작용](#)에 대한 콘솔 작업 설명서가 추가되었습니다. 다음 지침에 따라 AWS IoT SiteWise 콘솔에서 작업을 완료할 수 있습니다.

2020년 6월 11일

[내보낸 데이터 분석 자습서](#)

[내보내기 기능 AWS CloudFormation 템플릿을](#) 사용하여 Amazon Athena를 사용하여 Amazon S3로 내보낸 자산 데이터를 분석하는 방법을 배울 수 있는 자습서를 추가했습니다.

2020년 5월 27일

[공식 표현식 사용 개선](#)

AWS IoT SiteWise 수식 속성의 동작에 대한 자세한 정보를 추가하고 필터링된 데이터 포인트를 계산하는 방법의 예를 추가했습니다.

2020년 5월 18일

[IoT SiteWise 커넥터 버전 6 릴리스](#)

IoT SiteWise 커넥터 버전 6이 릴리스되었습니다. 이 릴리스에는 새로운 OPC UA 태그의 CloudWatch 지표 및 자동 검색에 대한 지원이 추가되었습니다. 즉, OPC UA 소스에 대한 태그가 변경되면 SiteWise Edge 게이트웨이를 다시 시작할 필요가 없습니다. 이 버전의 커넥터에는 스트림 관리자 및 AWS IoT Greengrass 코어 소프트웨어 v1.10.0 이상이 필요합니다.

2020년 4월 29일

[AWS IoT SiteWise 기능 릴리스](#)

AWS IoT SiteWise 기능 릴리스. 이제 API를 사용하여 SiteWise Edge 게이트웨이를 관리하고, 포털에 로고를 추가하고, SiteWise Edge 게이트웨이 지표를 보는 등 다양한 작업을 수행할 수 있습니다.

2020년 4월 29일

- 새 [데이터 값을 Amazon S3 버킷으로 내보내기](#)는 데 사용할 수 있는 AWS CloudFormation 템플릿과 함께 Amazon S3로 데이터 내보내기 섹션을 추가했습니다.
- SiteWise Edge 게이트웨이 소스 설명서를 개선하고 새로운 SiteWise Edge 게이트웨이 API를 수록한 [데이터 소스 구성](#) 섹션을 추가했습니다.
- SiteWise Edge 게이트웨이가 게시하는 CloudWatch 지표를 설명하는 SiteWise Edge 게이트웨이 지표 섹션이 추가되었습니다.
- Amazon EC2 인스턴스에서 SiteWise Edge 게이트웨이 종속성을 빠르게 구성하는 데 사용할 수 있는 AWS CloudFormation 템플릿과 함께 Amazon EC2에서 SiteWise Edge 게이트웨이 구성 섹션을 추가했습니다.
- SiteWise Monitor 포털의 새로운 권한 기능을 설명하는

[포털 서비스 역할](#) 섹션이 추가되었습니다.

- 포털 서비스 역할 및 포털 로고에 대한 [포털 설명서](#)를 업데이트했습니다.
- [AWS IoT SiteWise 리소스 태그 지정](#) 섹션이 추가되었습니다.
- 새 대시보드 정의 구조에 대한 [대시보드 생성\(CLI\)](#) 섹션을 업데이트했습니다.
- [보안](#) 섹션을 추가했습니다.

[에서 데이터 수집 AWS IoT Events](#)

이벤트가 발생할 AWS IoT Events 때에서 데이터를 수집하는 방법에 대한 정보가 추가되었습니다.

2020년 4월 20일

[SiteWise Monitor에서 풍력 발전소 데이터 시각화 및 공유 자습서](#)

를 사용하여 자산 데이터를 시각화하고 공유하는 방법을 알아보기 AWS IoT SiteWise Monitor 위해 따를 수 있는 자습서가 추가되었습니다.

2020년 3월 12일

[AWS IoT SiteWise 개념](#)

서비스 및 공통 용어에 대해 알아보는 데 사용할 수 있는 AWS IoT SiteWise 개념 용어집이 추가되었습니다.

2020년 3월 5일

[AWS IoT Greengrass 설치 지침 제거](#)

AWS IoT SiteWise 사용 설명서에서 AWS IoT Greengrass 코어 소프트웨어 설치 지침을 제거했습니다. [AWS IoT Greengrass 개발자 안내서](#)는 Amazon EC2 및 Docker와 같은 AWS IoT Greengrass 다른 플랫폼에서 설정하기 위한 디바이스 설정 스크립트와 지침을 제공합니다.

2020년 2월 14일

[AWS IoT Core 규칙을 사용한 데이터 수집 개선](#)

MQTT 메시지에서 데이터를 수집하는 데 사용할 수 있는 AWS IoT SiteWise 규칙 작업의 사용 방법과 문제 해결 <https://docs.aws.amazon.com/iot-sitewise/latest/userguide/iot-rules.html> 방법에 대한 자세한 정보가 추가되었습니다 AWS IoT Core. <https://docs.aws.amazon.com/iot-sitewise/latest/userguide/troubleshooting.html#troubleshoot-rule>

2020년 2월 14일

[IoT SiteWise 커넥터 버전 5 릴리스](#)

IoT SiteWise 커넥터 버전 5가 릴리스되었습니다. 이 릴리스에서는 AWS IoT Greengrass 코어 소프트웨어 v1.9.4와의 호환성 문제를 수정했습니다.

2020년 2월 12일

[IoT SiteWise 커넥터 버전 4 릴리스](#)

IoT SiteWise 커넥터 버전 4가 릴리스되었습니다. 이 릴리스는 OPC UA 서버 재연결 문제를 해결합니다.

2020년 2월 7일

[모델링 산업 자산 재구성](#)

자산 및 모델 업데이트 섹션을 산업 자산 모델링 내의 여러 주제로 재구성했습니다.

2020년 2월 4일

- [자산 및 모델 상태](#)
- [에 대한 데이터 스트림 관리](#)
[AWS IoT SiteWise](#)
- [속성 값 업데이트](#)
- [자산 연결 및 연결 해제](#)
- [자산 및 모델 업데이트](#)
- [에서 자산 및 모델 삭제 AWS IoT SiteWise](#)

[AWS IoT 사물에서 데이터 수집 자습서](#)

신규 또는 기존 AWS IoT 사물 플릿에서 데이터를 수집하도록 AWS IoT SiteWise 규칙 작업을 구성하는 방법을 알아보기 위해 따를 수 있는 자습서를 추가했습니다.

2020년 2월 4일

[에서 데이터 검색 재구성 AWS IoT SiteWise](#)

데이터 검색 섹션을 [자산 속성 값 및 집계 쿼리](#)와 [다른 AWS 서비스와의 상호 작용](#)이라는 두 가지 최상위 섹션으로 재구성했습니다.

2020년 1월 21일

[Amazon DynamoDB에 속성 값 업데이트 게시 자습서](#)

속성 값 알림을 사용하여 DynamoDB에 자산 데이터를 저장하는 방법을 알아보기 위해 따를 수 있는 자습서를 추가했습니다.

2020년 1월 8일

공식 표현식 사용	변환 및 미터법 속성에 사용할 수 있는 상수와 함수를 정리하기 위해 공식 표현식 참조를 추가했습니다. 자산 속성 을 각 속성 유형에 대한 별도의 주제로 재구성했습니다.	2020년 1월 7일
OPC UA 노드 필터 사용	SiteWise Edge 게이트웨이 소스를 추가할 때 OPC UA 노드 필터를 사용하여 SiteWise Edge 게이트웨이 성능을 향상시키는 방법에 대한 정보를 추가했습니다.	2020년 1월 3일
커넥터 업그레이드	새 커넥터 버전이 릴리스될 때 SiteWise Edge 게이트웨이를 업그레이드하는 방법에 대한 정보를 추가했습니다.	2019년 12월 30일
IoT SiteWise 커넥터 버전 3 릴리스	IoT SiteWise 커넥터 버전 3이 릴리스되었습니다. 이 릴리스에서는 IoT: * 권한 요구 사항을 제거했습니다.	2019년 12월 17일
IoT SiteWise 커넥터 버전 2 릴리스	IoT SiteWise 커넥터 버전 2가 릴리스되었습니다. 이 릴리스에는 여러 OPC UA 암호 리소스에 대한 지원이 추가되었습니다.	2019년 12월 10일
대시보드 생성(AWS CLI)	를 AWS IoT SiteWise Monitor 사용하여에서 대시보드를 생성하는 방법에 대한 정보가 추가되었습니다 AWS CLI.	2019년 12월 6일

[AWS IoT SiteWise 버전 2 릴리스](#)

2019년 12월 2일

버전 2에 대한 미리 보기가 릴리스되었습니다 AWS IoT SiteWise. 이제 OPC UA, MQTT 및 HTTP를 통해 데이터를 수집하고 자산 계층 구조에서 데이터를 모델링하고 SiteWise Monitor를 사용하여 데이터를 시각화할 수 있습니다.

- 자산, 자산 모델 및 자산 계층 구조가 변경되어 [자산 모델링](#) 섹션을 다시 작성했습니다.
- AWS IoT Greengrass 커넥터 단계 및 비게이트웨이 [데이터 수집](#) 섹션을 포함하도록 데이터 수집 섹션을 업데이트했습니다.
- SiteWise Monitor 웹 애플리케이션을 사용하는 방법을 보여주는 [AWS IoT SiteWise Monitor](#) 섹션 및 [별도의 애플리케이션 안내서](#)가 추가되었습니다.
- [에서 데이터 쿼리 AWS IoT SiteWise](#) 및 [다른 AWS 서비스와 상호 작용](#) 섹션을 추가했습니다.
- 업데이트된 데모 환경에 맞게 [시작하기](#) 섹션을 다시 작성했습니다.

[AWS IoT SiteWise 버전 1 릴리스](#)

2019년 2월 25일

버전 1의 초기 미리 보기가 릴리스되었습니다 AWS IoT SiteWise.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.