ユーザーガイド

AWS IoT SiteWise

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

AWS IoT SiteWise: ユーザーガイド

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

Amazon の商標およびトレードドレスはAmazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

とは AWS IoT SiteWise 1
の AWS IoT SiteWise 仕組み 2
産業用データを取り込む 2
アセットをモデル化して、収集したデータをコンテキスト化 3
クエリ、アラーム、および予測を使用して分析 4
オペレーションを視覚化する 4
データの保存 5
他の サービスとの統合 5
のユースケース AWS IoT SiteWise 5
製造 5
食品飲料 6
エネルギーとユーティリティ 6
AWS SDKs の使用 6
概念 7
はじめに 14
要件 14
AWS アカウントを設定する 15
にサインアップする AWS アカウント 15
管理アクセスを持つユーザーを作成する 15
クイックスタートデモを使用する 18
AWS IoT SiteWise デモを作成する 18
AWS IoT SiteWise デモを削除する 20
チュートリアル 22
OEE を計算 22
前提条件 22
OEE の計算方法 23
データを取り込む 25
前提条件 26
ステップ 1: AWS IoT ポリシーを作成する 27
ステップ 2: AWS IoT モノを作成する 31
ステップ 3: デバイスアセットモデルを作成する 34
ステップ 4: デバイスフリートアセットモデルを作成する 36
ステップ 5: デバイスアセットを作成および設定する 37
ステップ 6: デバイスフリートアセットを作成および設定する 38

ステップ 7: AWS IoT Core でルールを作成してデバイスアセットにデータを送信する 39
ステップ 8: デバイスクライアントスクリプトを実行する 43
ステップ 9: チュートリアル後にリソースをクリーンアップする 50
データを SiteWise Edge と統合する 52
前提条件 53
ステップ 1: AWS IoT ポリシーを作成する 54
ステップ 2: AWS IoT モノを作成して設定する 55
ステップ 3: SiteWise Edge MQTT 対応 V3 ゲートウェイを設定する 56
ステップ 4: SiteWise Edge ゲートウェイソフトウェアをインストールする 58
ステップ 5: 外部アプリケーションに接続するように EMQX ブローカーを設定する 58
ステップ 6: Mosquitto でデータを発行する 61
ステップ 7: 送信先を指定する 63
ステップ 8: パスフィルターを指定する 66
ステップ 9: AWS IoT リソースを設定する 69
ステップ 10: データを視覚化する 69
ステップ 11: チュートリアルの後にリソースをクリーンアップする 71
追加リソース 74
SiteWise Monitor でデータを視覚化して共有する 74
前提条件 75
ステップ 1: SiteWise Monitor でポータルを作成する 76
ステップ 2: ポータルにサインインする 80
ステップ 3: 風力発電所プロジェクトを作成する 82
ステップ 4: ダッシュボードを作成して、風力発電所データを視覚化する 86
ステップ 5: ポータルを調べる 92
ステップ 6: チュートリアル後にリソースをクリーンアップする 93
Amazon DynamoDB に発行する 96
前提条件 96
ステップ 1: プロパティ値の更新を発行 AWS IoT SiteWise するように を設定する 97
ステップ 2: AWS IoT Core でルールを作成する 99
ステップ 3: DynamoDB テーブルを作成する 101
ステップ 4: DynamoDB ルールアクションを設定する 103
ステップ 5: DynamoDB のデータを調べる 104
ステップ 6: チュートリアル後にリソースをクリーンアップする 105
にデータを取り込む AWS IoT SiteWise 109
データストリームの管理 110
アクセス許可と設定を構成する 111

データストリームをアセットプロパティに関連付ける ... 112
アセットプロパティからデータストリームの関連付けを解除する ... 113
データストリームを削除する ... 114
アセットプロパティエイリアスを更新する ... 115
一般的なシナリオ ... 117
AWS IoT SiteWise APIs を使用してデータを取り込む ... 119
BatchPutAssetPropertyValue API ... 119
CreateBulkImportJob API ... 122
AWS IoT Core ルールを使用する ... 130
必要なアクセス権を付与 ... 131
ルールアクションを設定 ... 132
基本的な取り込みでコストを削減 ... 141
AWS IoT Events アクションを使用する ... 142
AWS IoT Greengrass ストリームマネージャーを使用する ... 143
SiteWise Edge ゲートウェイを使用する ... 144
ゲートウェイの主な概念 ... 144
SiteWise Edge を実装するメリット ... 145
ゲートウェイをセルフホストする ... 146
要件 ... 147
ゲートウェイを作成します。 ... 152
ゲートウェイソフトウェアのインストール ... 155
MQTT 対応、V3 ゲートウェイ ... 158
クラシックストリーム、V2 ゲートウェイ ... 225
データソースを追加する ... 240
SiteWise Edge のコンポーネント ... 283
アセットをフィルタリングする ... 284
プロキシのサポートと信頼ストア ... 286
API を使用する ... 292
Siemens Industrial Edge でゲートウェイをホストする ... 310
セキュリティ ... 311
Siemens Secure Storage と AWS IoT SiteWise Edge アプリケーション ... 311
Siemens Industrial Edge デバイスの送信先 ... 312
プレビューアプリケーションからの移行 ... 313
トラブルシューティング ... 314
AWS IoT SiteWise エッジアプリケーション変更ログ ... 314
要件 ... 316

ゲートウェイを作成します。 ... 317
Siemens Databus user を作成する ... 318
アプリケーションにアクセスする ... 319
アプリケーションをインストール ... 319
インストールされたアプリケーション設定を更新する ... 322
送信先とパスフィルター ... 322
送信先を理解する ... 323
パスフィルターを理解する ... 326
リアルタイム送信先を追加する ... 330
Amazon S3 を使用してバッファされた送信先を追加する ... 338
パスフィルターを追加する ... 345
送信先を管理する ... 349
ゲートウェイを管理 ... 355
AWS IoT SiteWise コンソールを使用して SiteWise Edge ゲートウェイを管理 ... 356
AWS OpsHub for を使用して SiteWise Edge ゲートウェイを管理する AWS IoT SiteWise .. 356
ローカルオペレーティングシステムの認証情報を使用して、SiteWise Edge ゲートウェイにアクセスする ... 358
SiteWise Edge ゲートウェイ証明書を管理 ... 361
SiteWise Edge ゲートウェイコンポーネントパックのバージョンを変更する ... 362
SiteWise Edge ゲートウェイを一覧表示する ... 362
SiteWise Edge ゲートウェイの説明 ... 363
SiteWise Edge ゲートウェイを作成する ... 364
SiteWise Edge ゲートウェイを更新する ... 365
ゲートウェイ機能設定を更新する ... 366
ゲートウェイリソースにタグを付ける ... 367
ゲートウェイのタグを一覧表示する ... 368
ゲートウェイからタグを削除する ... 369
コンポーネントのバージョン AWS IoT SiteWise を更新する ... 370
SiteWise Edge ゲートウェイを削除する ... 370
ゲートウェイのバックアップと復元 ... 372
メトリクスデータの日次バックアップ ... 372
SiteWise Edge ゲートウェイを復元する ... 373
AWS IoT SiteWise データの復元 ... 374
バックアップと復元が正常に完了したことを検証する ... 375
レガシーゲートウェイ (AWS IoT Greengrass Version 1) ... 375
産業用アセットをモデリングする ... 377

アセットおよびモデルの状態 378
アセットのステータスを確認する 379
アセットまたはコンポーネントモデルのステータスを確認する 381
アセットモデルのバージョン 383
アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得する (コンソール) 384
アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得する (AWS CLI) 385
カスタム複合モデル (コンポーネント) 386
インラインカスタム複合モデル 386
コンポーネントモデルベースのカスタム複合モデル 388
パスを使用してカスタム複合モデルのプロパティを参照する 390
オブジェクト ID を設定する 391
オブジェクト UUID を操作する 392
外部 ID を使用する 392
モデルを作成する 394
でアセットモデルを作成する AWS IoT SiteWise 395
コンポーネントモデルを作成する 410
データのプロパティを定義する 414
カスタム複合モデルを作成する (コンポーネント) 495
アセットを作成する 499
アセットを作成する (コンソール) 499
アセットを作成する (AWS CLI) 500
新しいアセットを設定する 502
アセットを検索する 502
前提条件 502
の詳細検索 AWS IoT SiteWise コンソール 503
属性値を更新する 506
アセットの関連付けと関連付け解除 509
アセットの関連付けと関連付け解除 (コンソール) 509
アセットの関連付けと関連付け解除 (AWS CLI) 510
アセットとモデルを更新する 512
でアセットを更新する AWS IoT SiteWise 512
アセットモデルとコンポーネントモデルを更新する 514
カスタム複合モデルを更新する (コンポーネント) 520
アセットモデル書き込みの楽観的ロック 523

でアセットとモデルを削除する AWS IoT SiteWise 527
アセットを削除する 527
アセットモデルを削除する 530
アセットとモデルを使用した一括オペレーション 531
主要な概念と用語 532
サポートされている機能 533
一括操作の前提条件 534
一括インポートジョブを実行する 537
一括エクスポートジョブを実行する 539
ジョブの進捗状況追跡とエラー処理 542
メタデータのインポート例 548
メタデータのエクスポート例 563
メタデータ転送ジョブスキーマ 566
アラームによるデータのモニタリング 585
アラーム型 585
アラームの状態 587
アラーム状態のプロパティ 588
アセットモデルのアラームを定義する 591
アラーム通知の要件 594
AWS IoT Events アラームを定義する 595
外部アラームを定義 629
アセットにアラームを設定する 631
しきい値を設定する (コンソール) 632
しきい値を設定する (AWS CLI) 632
通知設定を行う 635
アラームに対応する 637
アラームに応答する (コンソール) 638
アラームに応答する (API) 641
外部アラーム状態を取り込む 642
外部アラーム状態ストリームをマッピング 642
アラーム状態データを取り込む 644
AWS IoT SiteWise アシスタント 646
AWS IoT SiteWise Assistant を設定する 646
データセットを作成する 648
データセットを編集する 653
データセットを削除する 655

AWS IoT SiteWise アシスタントの質問 656
でデータをモニタリングする AWS IoT SiteWise Monitor 657
SiteWise Monitor のロール 658
SAML フェデレーション 659
SiteWise Monitor の概念 660
(AWS IoT SiteWise Monitor Classic) の使用を開始する 662
ポータルを作成する 663
ポータルを設定 664
管理者を招待 668
ポータルユーザーの追加 671
ダッシュボードを作成する (CLI) 675
ポータルのアラームを有効にする 681
エッジでポータルを有効にする 684
ポータルを管理 684
の使用を開始する AWS IoT SiteWise Monitor (AI 対応) 694
ポータルを作成する 696
ポータルを設定 696
ポータルを管理 699
ポータルを削除する 703
を使用してダッシュボードを作成する AWS CLI 704
ポータルログイン 709
プロジェクトを作成する 709
プロジェクトを更新します。 710
プロジェクトを削除します。 711
ダッシュボードを作成する 711
ダッシュボードの更新 713
ダッシュボードの削除 713
ダッシュボードを設定する 714
からのデータのクエリ AWS IoT SiteWise 735
現在のアセット値をクエリ 736
アセットプロパティの現在の値をクエリ (コンソール) 736
アセットプロパティの現在の値をクエリ (AWS CLI) 737
履歴アセットプロパティ値をクエリ 738
アセットプロパティ集計をクエリ 740
アセットプロパティの集計 (API) 740
アセットプロパティの集計 (AWS CLI) 741

AWS IoT SiteWise クエリ言語 743
クエリ言語リファレンス 743
クエリの最適化 767
メタデータフィルター 767
Raw データフィルター 768
JOIN 最適化 769
大きなクエリ 771
ODBC 771
接続文字列構文 772
接続文字列の例 777
トラブルシューティング 778
他のサービスを操作 779
MQTT トピックのアセットプロパティについて 780
通知を使用 780
アセットプロパティ通知を有効化 (コンソール) 781
アセットプロパティ通知を有効化 (AWS CLI) 781
クエリ通知 783
データを Amazon S3 にエクスポートする 786
Grafana の統合 786
との統合 AWS IoT TwinMaker 788
統合の有効化 789
AWS IoT SiteWise と の統合 AWS IoT TwinMaker 790
機器の異常を検出する 791
予測定義を追加する (コンソール) 792
予測をトレーニング (コンソール) 795
予測の推論を開始または停止する (コンソール) 796
予測定義を追加 (CLI) 797
予測をトレーニングして、推論を開始する (CLI) 800
予測をトレーニングする (CLI) 802
予測の推論を開始または停止する (CLI) 804
ネイティブ異常検出 807
ネイティブ異常検出機能 808
前提条件 808
計算モデル APIs AWS CLI のセットアップ 809
プロパティ要件 809
ラベリングの前提条件 809

モデル評価の前提条件 810
アセットのセンサーで異常検出を有効にする 811
計算モデルを作成する (AWS CLI) 811
ExecuteAction API ペイロードの準備 812
をトレーニングする AWS CLI 813
推論の開始と停止 (AWS CLI) 815
データバインディングの検索 818
アセット間でセンサーの異常検出を有効にする 820
計算モデルを作成する (AWS CLI) 820
ExecuteAction API ペイロードの準備 821
をトレーニングする AWS CLI 822
推論の開始と停止 (AWS CLI) 824
高度なトレーニング設定 827
サンプルレート設定 827
データのラベル付け 828
モデルの評価 830
高度な推論設定 832
高頻度推論 (5 分 ~ 1 時間) 832
低頻度推論 (2 時間 – 1 日) 833
柔軟なスケジューリング 834
モデルバージョンのアクティベーション 835
モデルバージョンの確認 837
推論結果を確認する 838
推論結果を取得する 838
推論結果を理解する 839
異常な動作に対してカスタムアクションをトリガーする (AWS Management Console) 840
ベストプラクティス 841
最小日付範囲を理解する 841
高頻度データとトレーニングと推論間の整合性のサンプリング 841
ラベル付けに関する推奨事項 841
データストレージを管理する 843
ストレージ設定を行う 844
データ保持の影響 844
ウォーム階層の設定 (コンソール) 845
ウォーム階層を構成する (AWS CLI) 846
コールド階層の設定 (コンソール) 849

コールド階層の設定 (AWS CLI) ... 852
ストレージ設定のトラブルシューティング ... 858
エラー:バケットは存在しません ... 858
エラー: Amazon S3 パスへのアクセスが拒否されました ... 858
エラー:ロール ARN を引き受けることができません ... 859
エラー: クロスリージョン Amazon S3 バケットにアクセスできませんでした ... 859
Cold 階層に保存されたデータのファイルパスとスキーマ ... 859
機器データ (測定) ... 860
メトリクス、変換、集計 ... 864
アセットメタデータ ... 868
アセット階層メタデータ ... 872
ストレージデータインデックスファイル ... 874
コードの例 ... 876
基本 ... 880
こんにち AWS IoT SiteWiseは ... 881
基本を学ぶ ... 884
アクション ... 948
セキュリティ ... 1025
データ保護 ... 1026
インターネットトラフィックのプライバシー ... 1027
AWS IoT SiteWise アシスタントビジネスサービスの改善 ... 1027
データ暗号化 ... 1027
保管中の暗号化 ... 1028
転送中の暗号化 ... 1031
キー管理 ... 1032
Identity and Access Management ... 1034
対象者 ... 1035
ID で認証する ... 1035
が IAM と AWS IoT SiteWise 連携する方法 ... 1038
マネージドポリシー ... 1059
サービスにリンクされた役割 ... 1066
アラームのアクセス許可を設定する ... 1087
でのサービス間の混乱した代理の防止 AWS IoT SiteWise ... 1094
アイデンティティとアクセスに関するトラブルシューティング ... 1096
コンプライアンス検証 ... 1098
耐障害性 ... 1099

インフラストラクチャセキュリティ 1099
設定と脆弱性の分析 1100
VPC エンドポイント 1101
サポートされている API オペレーション 1101
インターフェイス VPC エンドポイントを作成する 1104
インターフェイス VPC エンドポイント AWS IoT SiteWise を介したアクセス 1104
VPCエンドポイントポリシーを作成する 1106
セキュリティに関するベストプラクティス 1107
OPC UA サーバーで認証情報を使用する 1107
OPC UA サーバーに暗号化通信モードを使用 1108
コンポーネントを最新の状態に保つ。 1108
SiteWise Edge ゲートウェイのファイルシステムの暗号化 1108
エッジ設定へのセキュアなアクセス。 1109
Siemens Industrial Edge Management のデータの保護 1109
SiteWise Monitor ユーザーへの最小限のアクセス許可の付与 1109
機密情報を公開しない 1109
AWS IoT Greengrass セキュリティのベストプラクティスに従う 1110
関連情報 1110
ログとモニタリング 1111
サービスログをモニタリングする 1111
ログインの管理 AWS IoT SiteWise 1113
例: AWS IoT SiteWise ログファイルエントリ 1115
SiteWise Edge ゲートウェイログをモニタリングする 1115
Amazon CloudWatch Logs を使用 1115
サービスログを使用する 1117
Amazon CloudWatch メトリクスでモニタリングする 1119
AWS IoT Greengrass Version 2 ゲートウェイメトリクス 1119
を使用した API コールのログ記録 AWS CloudTrail 1129
AWS IoT SiteWise CloudTrail の情報 1129
AWS IoT SiteWise CloudTrail のデータイベント 1130
AWS IoT SiteWise CloudTrail の管理イベント 1133
例: AWS IoT SiteWise ログファイルエントリ 1133
リソースのタグ付け 1135
でタグを使用する AWS IoT SiteWise 1135
でタグ付けする AWS Management Console 1135
AWS IoT SiteWise API でタグ付けする 1135

IAM ポリシーでタグを使用する 1137
トラブルシューティング 1139
ゲートウェイのトラブルシューティング 1139
SiteWise Edge ゲートウェイログを設定してアクセスする 1139
SiteWise Edge ゲートウェイの問題のトラブルシューティング 1140
での AWS IoT SiteWise Edge アプリケーションのトラブルシューティング Siemens Industrial Edge 1146
Edge でのオープンソース統合のトラブルシューティング 1147
AWS IoT Greengrass 問題のトラブルシューティング 1150
ポータルのトラブルシューティング 1150
ユーザー、管理者が AWS IoT SiteWise ポータルにアクセスできない 1150
AWS IoT SiteWise ルールアクションのトラブルシューティング 1151
AWS IoT Core ログの設定 1152
再発行エラーアクションを設定 1153
ルール問題をトラブルシューティング 1155
ルールをトラブルシューティング (AWS IoT SiteWise) 1158
ルールをトラブルシューティング (DynamoDB) 1159
一括インポートとエクスポートのトラブルシューティング 1163
エンドポイントとクォータ 1164
エンドポイント 1164
クォータ 1164
AWS IoT SiteWise Assistant API スロットリングの制限 1179
異常検出のクォータ 1180
ドキュメント履歴 1182
..... mcciv

とは AWS IoT SiteWise

AWS IoT SiteWise は、産業機器から大規模にデータを収集、保存、整理、モニタリングできるマネージド型サービスで、データ駆動型の意思決定に役立ちます。 AWS IoT SiteWise を使用して、施設間のオペレーションをモニタリングし、一般的な産業パフォーマンスメトリクスを迅速に計算し、産業機器データを分析するアプリケーションを作成して、コストのかかる機器の問題を防ぎ、本番環境のギャップを軽減できます。

を使用すると AWS IoT SiteWise Monitor、運用ユーザーはウェブアプリケーションを作成して、産業データをリアルタイムで表示および分析できます。平均故障間隔や総合設備効率 (OEE) などのメトリクスを設定およびモニタリングすることで、産業オペレーションに関するインサイトを得ることができます。

AWS IoT SiteWise Edge は、ローカルデバイス上のデータの収集、保存、処理 AWS IoT SiteWise を可能にする のコンポーネントです。これは、インターネットへのアクセスが制限されている場合や、データを非公開にする必要がある場合に役立ちます。

次の図は、 の基本的なアーキテクチャを示しています AWS IoT SiteWise。

トピック

- の AWS IoT SiteWise 仕組み
- のユースケース AWS IoT SiteWise
- AWS SDK でのこのサービスの使用
- AWS IoT SiteWise の概念

の AWS IoT SiteWise 仕組み

AWS IoT SiteWise は、産業用デバイス、プロセス、施設の表現を作成するために使用できるリソースモデリングフレームワークを提供します。機器とプロセスの表現は、 のアセットモデルと呼ばれます AWS IoT SiteWise。アセットモデルでは、使用する生データと、それを有用なメトリクスに処理する方法を定義します。AWS IoT SiteWise コンソールで、産業オペレーション用のアセットとモデルを構築し、視覚化します。エッジまたは AWS クラウドでデータを収集して処理するようにアセットモデルを設定することもできます。

トピック

- 産業用データを取り込む
- アセットをモデル化して、収集したデータをコンテキスト化
- クエリ、アラーム、および予測を使用して分析
- オペレーションを視覚化する
- データの保存
- 他の サービスとの統合

産業用データを取り込む

産業データを取り込み AWS IoT SiteWise 、 の使用を開始します。データの取り込みは、次のいずれかの方法で行われます。

- オンサイトサーバーからの直接取り込み: OPC UA などのプロトコルを利用して、オンサイトデバイスからデータを直接読み取ります。互換性のある SiteWise Edge ゲートウェイソフトウェアを AWS IoT Greengrass V2、一般的な産業用ゲートウェイや仮想サーバーなどの幅広いプラットフォームにデプロイします。最大 100 の OPC UA サーバーを 1 つの AWS IoT SiteWise ゲートウェイに接続できます。詳細については、「AWS IoT SiteWise Edge セルフホストゲートウェイの要件」を参照してください。

  Modbus TCP やイーサネット/IP (EIP) などのプロトコルは、 のコンテキストDomaticaで とのパートナーシップを通じてサポートされることに注意してください AWS IoT Greengrass V2。
- パックによるエッジデータ処理: パックを追加することによって SiteWise Edge ゲートウェイを強化し、包括的なエッジ機能を有効にします。SiteWise Edge を で使用できるように AWS IoT Greengrass V2すると、データ処理がオンサイトで直接実行されてから AWS IoT Greengrass 、ストリームを使用して AWS クラウドに安全に送信されます。詳細については、「SiteWise Edge で OPC UA ソースを設定する」を参照してください。

- 一括オペレーションによる Amazon S3 経由のアダプティブ取り込み: 多数のアセットまたはアセットモデルを使用する場合、一括オペレーションを使用して Amazon S3 バケットからリソースを一括インポートおよびエクスポートします。詳細については、「アセットとモデルを使用した一括オペレーション」を参照してください。
- AWS IoT コアルールを使用した MQTT メッセージ： AWS IoT Core に接続されたデバイスで MQTT メッセージを送信する場合は、 AWS IoT Core ルールエンジンを使用してそれらのメッセージを転送します AWS IoT SiteWise。 AWS IoT Core に接続されたデバイスで MQTT メッセージを送信する場合は、 AWS IoT Core ルールエンジンを使用してそれらのメッセージを にルーティングします AWS IoT SiteWise。詳細については、「AWS IoT Core ルール AWS IoT SiteWise を使用して にデータを取り込む」を参照してください。
- イベントによってトリガーされるデータ取り込み： AWS IoT Events アクションを使用して、イベントが発生した AWS IoT SiteWise ときに にデータを送信する AWS IoT Events ように で IoT SiteWise アクションを設定します。詳細については、「AWS IoT SiteWise から にデータを取り込む AWS IoT Events」を参照してください。
- AWS IoT SiteWise API: Edge またはクラウドのアプリケーションは、データを に直接送信できます AWS IoT SiteWise。詳細については、「AWS IoT SiteWise APIs を使用してデータを取り込む」を参照してください。

アセットをモデル化して、収集したデータをコンテキスト化

データを取り込んだ後、そのデータを使用して、物理オペレーションのモデルを構築することで、アセット、プロセス、および施設の仮想表現を作成できます。デバイスまたはプロセスを表すアセットは、データストリームを AWS クラウドに送信します。アセットは、論理デバイスグループ化を意味することもできます。階層は、アセットを関連付けて複雑なオペレーションをミラーリングすることで形成されます。これらの階層により、アセットは関連付けられた子アセットからデータにアクセスできます。アセットは、アセットモデルから作成されます。アセットモデルはアセットのフォーマットを標準化する宣言的な構造です。アセットのコンポーネントを再利用して、モデルの整理と保守を行います。詳細については、「産業用アセットをモデリングする」を参照してください。

を使用すると AWS IoT SiteWise、受信データをコンテキストメトリクスと変換に変換するようにアセットを設定できます。

- 機器データを受信するときに作業を変換します。
- メトリクスは、ユーザーが定義した間隔で計算されます。

メトリクスと変換は、個々のアセットまたは複数のアセットの両方に適用可能です。AWS IoT SiteWise は、機器データ、メトリクス、および変換に関連するさまざまな時間枠にわたって、平均、合計、カウントなど、一般的に使用される統計集計を自動的に計算します。

アセットは を使用して同期できます AWS IoT TwinMaker。詳細については、「AWS IoT SiteWise と の統合 AWS IoT TwinMaker」を参照してください。

クエリ、アラーム、および予測を使用して分析

クエリを実行し、アラームを設定 AWS IoT SiteWise して、 で収集された日付を分析します。Amazon Lookout を使用して、メトリクス内の異常を自動的に検出し、その根本原因を特定することもできます。

- 特定のアラームを設定して、機器またはプロセスが最適なパフォーマンスから逸脱したときにチームに警告し、問題を迅速に特定して解決します。詳細については、「でアラームを使用してデータをモニタリングする AWS IoT SiteWise」を参照してください。
- AWS IoT SiteWise API オペレーションを使用して、特定の時間間隔におけるアセットプロパティの現在の値、履歴値、集計をクエリします。詳細については、「からのデータのクエリ AWS IoT SiteWise」を参照してください。
- Amazon Lookout for Equipment による異常検出を使用して、機器や動作条件の変化を特定して視覚化します。異常検出を使用すると、オペレーションの予防保全対策を決定できます。この統合により、顧客は AWS IoT SiteWise と Amazon Lookout for Equipment の間でデータを同期できます。詳細については、「Lookout for Equipment で異常を検出する」を参照してください。

オペレーションを視覚化する

SiteWise Monitor をセットアップして、運用従業員用のウェブアプリケーションを作成します。ウェブアプリケーションは、従業員がオペレーションを視覚化するのに役立ちます。IAM Identity Center または IAM を使用して、従業員のさまざまなレベルのアクセスを処理します。産業オペレーション全体の特定のサブセットを表示するように、従業員ごとに一意のログインとアクセス許可を設定します。 は、SiteWise Monitor の使用方法を学ぶために、これらの従業員向けのアプリケーションガイド AWS IoT SiteWise を提供します。

オペレーションの視覚化の詳細については、「でデータをモニタリングする AWS IoT SiteWise Monitor」を参照してください。

データの保存

時系列ストレージを産業用データレイクと統合できます。 AWS IoT SiteWise には、産業用データ用の 3 つのストレージ階層があります。

- リアルタイムアプリケーション用に最適化されたホットストレージ階層。
- 分析ワークロード用に最適化されたウォームストレージ階層。
- レイテンシー耐性の高い運用データアプリケーション用に Amazon S3 を使用する、カスタマー管理のコールドストレージ階層。

AWS IoT SiteWise は、最新のデータをホットストレージ階層に保持することで、ストレージコストを管理するのに役立ちます。次に、履歴データをウォームまたはコールド階層ストレージに移動するデータ保持ポリシーを定義します。詳細については、「でデータストレージを管理する AWS IoT SiteWise」を参照してください。

アセットメタデータをインポートおよびエクスポートすることもできます。詳細については、「アセットメタデータ」を参照してください。

他の サービスとの統合

AWS IoT SiteWise は複数の AWS サービスと統合して、 AWS クラウドで完全な AWS IoT ソリューションを開発します。詳細については、「他の AWS サービスとやり取りする」を参照してください。

のユースケース AWS IoT SiteWise

AWS IoT SiteWise は、さまざまな業界で多くの産業用データ収集および分析アプリケーションに使用されます。

すべてのソースから一貫してデータを収集し、問題を迅速に解決できるようにします。 AWS IoT SiteWise はリモートモニタリングを提供し、データをオンサイトで直接収集したり、多くの施設にまたがる複数のソースから収集したりできます。 は、産業用 IoT データソリューションに必要な柔軟性 AWS IoT SiteWise を提供します。

製造

AWS IoT SiteWise は、機器からデータを収集して活用するプロセスを簡素化し、非効率性を特定して最小化し、産業オペレーションを強化できます。 は、製造ラインや機器からデータを収集する AWS IoT SiteWise のに役立ちます。を使用すると AWS IoT SiteWise、データを AWS クラウドに転

送し、特定の機器やプロセスのパフォーマンスメトリクスを構築できます。生成されたメトリクスを使用して、オペレーションの全体的な効率性を理解し、イノベーションや改善の機会を特定できます。また、製造工程を表示し、設備や工程の欠陥、生産ギャップ、製品不良を特定することができます。

食品飲料

食品飲料業界の施設は、穀物製粉、精肉と加工、電子レンジ調理可能な料理のアセンブリ、調理、冷凍など、多岐にわたる食品加工を手掛けます。食品加工工場では、多くの場合、複数の場所に分散しており、工場および機器オペレーターは中央の場所でプロセスと機器をモニタリングしています。例えば、冷蔵ユニットは、原材料の処理と有効期限を評価します。設備間の廃棄物の生成をモニタリングし、運用効率を確保します。を使用すると AWS IoT SiteWise、複数の場所からのセンサーデータストリームを生産ラインと施設別にグループ化できるため、プロセスエンジニアは施設全体の理解を深め、改善を行うことができます。

エネルギーとユーティリティ

を使用すると AWS IoT SiteWise、機器の問題を簡単かつ効率的に解決できます。アセットのパフォーマンスをリモートでリアルタイムでモニタリングできます。どこからでも機器の履歴データにアクセスして、潜在的な問題を特定し、正確なリソースをディスパッチし、問題を迅速に防止および修正します。

AWS SDK でのこのサービスの使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、デベロッパーが好みの言語でアプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが提供されています。

SDK ドキュメント	コード例
AWS SDK for C++	AWS SDK for C++ コード例
AWS CLI	AWS CLI コード例
AWS SDK for Go	AWS SDK for Go コード例
AWS SDK for Java	AWS SDK for Java コード例
AWS SDK for JavaScript	AWS SDK for JavaScript コード例

SDK ドキュメント	コード例
AWS SDK for Kotlin	AWS SDK for Kotlin コード例
AWS SDK for .NET	AWS SDK for .NET コード例
AWS SDK for PHP	AWS SDK for PHP コード例
AWS Tools for PowerShell	AWS Tools for PowerShell コード例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) コード例
AWS SDK for Ruby	AWS SDK for Ruby コード例
AWS SDK for Rust	AWS SDK for Rust コード例
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP コード例
AWS SDK for Swift	AWS SDK for Swift コード例

可用性の例

必要なものが見つからなかった場合。このページの下側にある [Provide feedback (フィードバックを送信)] リンクから、コードの例をリクエストしてください。

AWS IoT SiteWise の概念

以下は、 の主要な概念です AWS IoT SiteWise。

集計

集計は、すべての時系列データに対して AWS IoT SiteWise が自動的に計算する基本的なメトリクスまたは測定値です。詳細については、「でアセットプロパティの集計をクエリする AWS IoT SiteWise」を参照してください。

アセット

産業機器 AWS IoT SiteWise から にデータを入力または取り込むと、デバイス、機器、プロセスはそれぞれアセットとして表示されます。各アセットには関連データがあります。例えば、機

器には、シリアル番号、ロケーション、製造元とモデル、およびインストール日があります。また、可用性、性能、品質、温度、圧力などの時系列値を持つ場合もあります。アセットを階層にグループ化し、アセットが子アセットに保存されているデータにアクセスできるようにします。詳細については、「産業用アセットをモデリングする」を参照してください。

アセット階層

アセット階層をセットアップして、産業オペレーションの論理的表現を作成します。これを行うには、アセットモデルに階層を定義し、そのモデルから作成されたアセットを指定された階層に関連付けます。親アセットのメトリクスは、子アセットのプロパティからのデータと組み合わせることができ、オペレーション全体や特定の部分に関するインサイトを提供するメトリクスを計算できます。詳細については、「アセットモデルの階層を定義する」を参照してください。

アセットモデル

すべてのアセットはアセットモデルを使用して作成されます。アセットモデルは、アセットのフォーマットを定義して標準化する構造です。同じタイプの複数のアセットに一貫した情報が適用されるため、デバイスのグループを表すアセット内のデータを処理できます。各アセットモデルでは、属性、時系列入力 (測定) 、時系列変換 (変換) 、時系列集約 (メトリクス) 、アセット階層を定義できます。詳細については、「産業用アセットをモデリングする」を参照してください。

アセットモデルをエッジ用に設定することで、アセットモデルのプロパティが処理される場所を制御できます。この機能を利用して、ローカルデバイスのアセットデータを処理し、モニタリングすることができます。

アセットプロパティ

アセットプロパティは、各アセット内の構造であり、産業用データを含みます。各プロパティはデータ型を持ち、単位を持つことができます。プロパティには、属性、測定、変換、または メトリクスを指定できます。詳細については、「データのプロパティを定義する」を参照してください。

エッジで計算するようにアセットプロパティを設定します。エッジでのデータ処理については、SiteWise Edge で OPC UA ソースを設定する を参照してください。

属性

属性は、デバイスメーカーやデバイスの場所など、通常一定に保たれるアセットのプロパティです。属性は事前設定値を持つことがあります。アセットモデルから作成された各アセットには、そのモデルで定義された属性のデフォルト値が含まれます。詳細については、「静的データ (属性) を定義する」を参照してください。

計算モデル

`ComputationModel` は、データに適用できる特定のタイプのコンピューティングの抽象化です。ユーザーが特定の計算エンジンの入力、出力、設定を記述できる一連の計算の設計図を定義します。 `ComputationModel`は ARN を持つ新しいリソースであり、ステートフルでバージョニングされています。詳細については、「計算モデルを作成する (AWS CLI)」を参照してください。

ダッシュボード

各プロジェクトには、ダッシュボードのセットが含まれています。ダッシュボードは、一連のアセットの値に対する一連の可視化を提供します。プロジェクトの所有者は、ダッシュボードとそれに含まれる可視化を作成します。プロジェクト所有者がダッシュボードのセットを共有する準備ができたら、所有者はプロジェクトに閲覧者を招待し、プロジェクト内のすべてのダッシュボードにアクセスできるようにします。ダッシュボードごとに異なる閲覧者のセットが必要な場合は、プロジェクト間でダッシュボードを分割する必要があります。ビューワーがダッシュボードを見るとき、特定のデータが表示されるように時間範囲をカスタマイズできます。

データセット

データセットは、時系列データ、non-time-seriesデータ、シフトスケジュール、メンテナンスレコード、従業員データベースなどの機器以外のデータを表すデータのコレクションです。外部データをサポートし、 AWS IoT SiteWise 分析機能を使用します。これには、データセットソース、データセットスキーマ、データセットパラメータが含まれます。 AWS IoT SiteWise Assistant は、Amazon Kendra インデックスを消費するデータセットを使用します。

データストリーム

アセットモデルとアセットを作成する前 AWS IoT SiteWise でも、産業データを に入力または取り込みます。 はデータストリーム AWS IoT SiteWise を自動的に生成して、機器から raw データストリームを収集します。

データストリームのエイリアス

データストリームのエイリアスは、データストリームを簡単に識別するのに役立ちます。例えば、エイリアス `server1-windfarm/3/turbine/7/temperature` は、風力発電所 #3 のタービン #7 から来る温度値を示します。用語 `server1` は、データソース名であり、OPC UA サーバーを識別するのに役立ち、`server1-` は、この OPC UA サーバーから報告されたすべてのデータストリームにアタッチされるプレフィックスです。

データストリームの関連付け

アセットモデルとアセットを作成したら、データストリームをアセットで定義されたアセットプロパティに関連付けてデータを構造化します。 AWS IoT SiteWise はアセットモデルとアセットを使用して、データストリームからの受信データを処理できます。また、アセットプロパティからのデータストリームの関連付けを解除することもできます。詳細については、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

送信先

SiteWise Edge の送信先は、テレメトリまたは処理されたデータを送信するエンドポイントを表します。SiteWise Edge は、 AWS IoT SiteWise ホット階層、バッファインジェスト、または Amazon S3 バケットを送信先としてサポートします。パスフィルターを使用して、特定の MQTT トピックをサブスクライブするように送信先を設定できます。詳細については、「AWS IoT SiteWise Edge の送信先を理解する」を参照してください。

式

各変換およびメトリクスプロパティには、そのプロパティがデータを変換または集約する方法を定義する式が伴います。これらの式には、 AWS IoT SiteWiseによって提供されるプロパティ入力、演算子、および関数が含まれます。詳細については、「数式を使用する」を参照してください。

測定

測定値は、デバイスまたは機器からの生のセンサー時系列データストリームを表すアセットのプロパティです。詳細については、「機器からのデータストリームを定義する (測定値) 」を参照してください。

メトリクス

メトリクスは、アセットのプロパティであり、集計された時系列データを表します。各メトリクスには、数式 (式) が伴い、データポイントを集計する方法と、その集計を計算する時間間隔の概要が示されています。メトリクスは、指定された時間間隔ごとに 1 つのデータポイントを生成します。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。

MQTT

MQTT (Message Queuing Telemetry Transport) は、センサーとデバイス用の軽量メッセージングプロトコルです。

パック

SiteWise Edge ゲートウェイはパックを使用して、データの収集、処理、およびルーティングの方法を決定します。SiteWise Edge ゲートウェイで利用可能なパックの詳細については、「the section called “パックを使用する”」を参照してください。

データ収集パック

データ収集パックを使用して、SiteWise Edge ゲートウェイが産業用データを収集し、任意の AWS 宛先にルーティングできるようにします。

データ処理パック

データ処理パックを使用して、最大 30 日間、エッジでデータを処理、保存、取得します。SiteWise Edge APIs。

OPC UA

OPC UA (Open Platform Communications Unified Architecture) は、産業用オートメーション用の通信プロトコルです。

パスフィルター

ゲートウェイ内のパスフィルターを使用して MQTT トピックをサブスクライブし、サポートされている送信先に発行します AWS IoT SiteWise 。MQTT ベースのソース、データ処理パイプライン、および送信先はすべて、セルフホスト MQTT 対応 V3 ゲートウェイで MQTT トピックを使用してデータを交換します。トピックフィルターを定義して、取り込みまたは異なる送信先にルーティングするデータを指定できます。

Portal

AWS IoT SiteWise Monitor ポータルは、 AWS IoT SiteWise データを視覚化して共有するために使用できるウェブアプリケーションです。ポータルには 1 人以上の管理者があり、0 個以上のプロジェクトが含まれています。

ポータル管理者

各 SiteWise Monitor ポータルには、1 人以上のポータル管理者があります。ポータル管理者は、ポータルを使用して、アセットとダッシュボードのコレクションを含むプロジェクトを作成します。その後で、ポータル管理者はアセットと所有者を各プロジェクトに割り当てます。プロジェクトへのアクセスを制御することにより、ポータル管理者は、プロジェクトの所有者とビューワーが参照できるアセットを指定します。

プロジェクト

各 SiteWise Monitor ポータルには、一連のプロジェクトが含まれています。各プロジェクトには、 AWS IoT SiteWise アセットと関連付けられたのサブセットがあります。プロジェクト所有者は、1 つ以上のダッシュボードを作成して、それらのアセットに関連付けられたデータの整合性ある表示方法を提供します。プロジェクトの所有者は、プロジェクトに閲覧者を招待して、プロジェクト内のアセットとダッシュボードを表示できるようにすることができます。プロジェクトは、SiteWise Monitor 内の共有の基本単位です。プロジェクト所有者は、 AWS 管理者からポータルへのアクセスを許可されたユーザーを招待できます。ユーザーは、そのポータル内のプロジェクトをそのユーザーと共有する前に、ポータルへのアクセス権を持っている必要があります。

プロジェクトの所有者

各 SiteWise Monitor プロジェクトには所有者がいます。プロジェクトの所有者は、一貫した方法で運用データを表すために、ダッシュボードの形式で可視化を作成します。ダッシュボードを共有する準備ができたら、プロジェクト所有者はビューワーをプロジェクトに招待できます。プロジェクトの所有者は、他の所有者もプロジェクトに割り当てることができます。プロジェクト所有者は、アラームのしきい値や通知設定を設定することができます。

プロジェクトビューワー

各 SiteWise Monitor プロジェクトにはビューワーがいます。プロジェクト閲覧者は、ポータルに接続して、プロジェクト所有者が作成したダッシュボードを表示できます。各ダッシュボードでは、プロジェクトビューワーが時間範囲を調整することで、運用データをより深く理解することができます。プロジェクト閲覧者は、アクセス権のあるプロジェクトのダッシュボードのみを表示できます。プロジェクトビューワーは、アラームを確認したり、スヌーズさせたりすることができます。

プロパティエイリアス

OPC UA サーバーデータストリームパス (例えば、/company/windfarm/3/turbine/7/temperature) などのアセットプロパティにエイリアスを作成するオプションがあり、アセットデータの取り込みまたは取得中にアセットプロパティの識別が容易になります。SiteWise Edge ゲートウェイを使用してサーバーからデータを取り込む場合、プロパティのエイリアスは raw データストリームのパスと一致する必要があります。詳細については、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

プロパティ通知

アセットプロパティのプロパティ通知を有効にすると、 はプロパティが新しい値を受信する AWS IoT Core たびに MQTT メッセージを に AWS IoT SiteWise 発行します。メッセージペイ

ロードには、そのプロパティ値の更新に関する詳細が含まれます。プロパティ値通知を使用して、 の産業データを他の AWS IoT SiteWise AWS サービスに接続するソリューションを作成します。詳細については、「他の AWS サービスとやり取りする」を参照してください。

SiteWise Edge ゲートウェイ

SiteWise Edge ゲートウェイは、データを収集、処理、および指示するために顧客の構内にインストールされます。SiteWise Edge ゲートウェイは、さまざまなプロトコルを介して産業用データソースに接続し、データを収集して処理し、 AWS クラウドに送信します。SiteWise Edge ゲートウェイは、パートナーデータソースに接続することもできます。詳細については、「AWS IoT SiteWise Edge ゲートウェイを使用する」を参照してください。

変換

変換は、アセットのプロパティであり、変換された時系列データを表します。すべての変換には数式 (式) が伴い、データポイントをある形式から別の形式に変換する方法を指定します。変換されたデータポイントは、入力データポイントと 1 対 1 の関係を保持します。詳細については、「データを変換する (変換) 」を参照してください。

視覚化

各ダッシュボードでは、プロジェクトに関連するアセットのプロパティとアラームをどのように表示するかをプロジェクト所有者が決定します。可用性は折れ線グラフで、その他の数値は棒グラフや重要業績評価指標 (KPI、 Key Performance Indicator) で表示されるかもしれません。アラームは、状態グリッドと状態タイムラインでの表示が最適です。プロジェクトの所有者は、それぞれの可視化をカスタマイズして、そのアセットのデータを最もよく理解できるようにしています。

の使用を開始する AWS IoT SiteWise

を使用すると AWS IoT SiteWise、データを収集、整理、分析、視覚化できます。

AWS IoT SiteWise には、実際のデータソースを設定せずにサービスを調べるために使用できるデモが用意されています。詳細については、「AWS IoT SiteWise デモを使用する」を参照してください。

以下のチュートリアルを完了して、 の特定の機能を調べることができます AWS IoT SiteWise。

- AWS IoT モノ AWS IoT SiteWise から にデータを取り込む
- SiteWise Monitor で風力発電所データを視覚化して共有する
- プロパティ値の更新を Amazon DynamoDB に発行する

詳細については、以下のトピックを参照してください AWS IoT SiteWise。

- にデータを取り込む AWS IoT SiteWise
- 産業用アセットをモデリングする
- AWS IoT SiteWise Edge でエッジ機能を設定する
- でデータをモニタリングする AWS IoT SiteWise Monitor
- からのデータのクエリ AWS IoT SiteWise
- 他の AWS サービスとやり取りする

トピック

- 要件
- AWS アカウントを設定する

要件

使用を開始するには、 AWS アカウントが必要です AWS IoT SiteWise。アカウントをお持ちでない場合は、以下のセクションでアカウントのセットアップ方法の詳細をご覧ください。

が AWS IoT SiteWise 利用可能なリージョンを使用します。詳細については、「AWS IoT SiteWise エンドポイントとクォータ」を参照してください。のリージョンセレクターを使用して AWS Management Console 、これらのリージョンのいずれかに切り替えることができます。

AWS アカウントを設定する

トピック

- にサインアップする AWS アカウント
- 管理アクセスを持つユーザーを作成する

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 を保護し AWS IAM Identity Center、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドのルートユーザーとしてサインインするを参照してください。

2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、IAM ユーザーガイドの AWS アカウント 「ルートユーザー (コンソール) の仮想 MFA デバイスを有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。

2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 AWS IAM Identity Center ユーザーガイド」の「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン 「 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。

2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの結合」を参照してください。

AWS IoT SiteWise デモを使用する

demo AWS IoT SiteWise を使用して簡単に を調べることができます AWS IoT SiteWise 。 は、デプロイしてアセットモデル、アセット、SiteWise Monitor ポータルを作成し、最大 1 週間のサンプルデータを生成できる AWS CloudFormation テンプレートとしてデモ AWS IoT SiteWise を提供します。

> **Important**
> デモを作成すると、このデモで作成および消費するリソースは有料です。

トピック

- AWS IoT SiteWise デモを作成する
- AWS IoT SiteWise デモを削除する

AWS IoT SiteWise デモを作成する

AWS IoT SiteWise コンソールから AWS IoT SiteWise デモを作成できます。

> **Note**
> このデモでは、デモに必要な Lambda 関数、1 つの CloudWatch Events ルール、および AWS Identity and Access Management (IAM) ロールが作成されます。これらのリソースは AWS アカウントに表示される場合があります。デモを終了するまで、これらのリソースを維持することをお勧めします。これらのリソースを削除すると、デモが正常に動作しなくなることがあります。

AWS IoT SiteWise コンソールでデモを作成するには

1. [AWS IoT SiteWise console] (コンソール) にアクセスして、ページの右上隅にある [SiteWise デモ] を見つけます。
2. (オプション) [SiteWise デモ] の下で、[デモアセットを保持する日数] フィールドを変更して、デモを削除する前に保持する日数を指定します。

3. (オプション) サンプルデータをモニタリングする SiteWise Monitor ポータルを作成するには、次の手順を実行します。

> **Note**
> このデモで作成および消費する SiteWise Monitor リソースは有料です。詳細については、「AWS IoT SiteWise 料金」の「SiteWise Monitor」を参照してください。

a. [リソースをモニタリング]を選択します。

b. 「許可」を選択します。

c. フェデレーティッド IAM ユーザーにポータルへのアクセス権を付与する既存の IAM ロールを選択します。

> **Important**
> IAM ロールには、以下のアクセス許可が必要です。
>
> JSON

```
{
    "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 Monitor の操作方法の詳細については、AWS IoT SiteWise Monitor 「 アプリケーションガイド」の「What is AWS IoT SiteWise Monitor？」を参照してください。

4. [デモの作成] を選択します。

   デモの作成には約 3 分かかります。デモの作成に失敗した場合は、アカウントに十分な権限がない可能性があります。管理者権限を持つアカウントに切り替えるか、次の手順に従ってデモを削除し、もう一度試してください。

   a. [デモの削除] を選択します。

      デモの削除には約 15 分かかります。

   b. デモの削除に失敗した場合は、[AWS CloudFormation console] (コンソール) を開いて、IoTSiteWiseDemoAssets IoT SiteWise デモアセット という名前のスタックを選択し、右上隅の [削除] を選択します。

   c. デモを再度削除できない場合は、 AWS CloudFormation コンソールの手順に従って削除に失敗したリソースをスキップし、もう一度試してください。

5. デモが正常に作成されたら、[AWS IoT SiteWise console] (コンソール) でデモアセットとデータを調べることができます。

AWS IoT SiteWise デモを削除する

AWS IoT SiteWise デモは 1 週間後にそれ自体、または AWS CloudFormation コンソールからデモスタックを作成した場合は選択した日数だけ削除されます。デモリソースの使用が完了したら、決められた日数が経過する前にデモを削除することができます。デモの作成に失敗した場合も、デモを削除することができます。デモを手動で削除するには、次の手順に従います。

AWS IoT SiteWise デモを削除するには

1. AWS CloudFormation コンソールに移動します。
2. [スタック] のリストから [IoTSiteWiseDemoAssets] を選択します。
3. [Delete (削除)] を選択します。

   スタックを消去すると、デモ用に作成されたすべてのリソースが消去されます。

4. 確認ダイアログで、[スタックの削除] を選択します。

   スタックの消去には約 15 分かかります。デモの消去に失敗した場合は、右上隅の [消去] を再度選択します。デモを再度削除できない場合は、 AWS CloudFormation コンソールの手順に従って削除に失敗したリソースをスキップし、もう一度試してください。

AWS IoT SiteWise チュートリアル

AWS IoT SiteWise チュートリアルページへようこそ。増え続けるチュートリアルのコレクションにより、 AWS IoT SiteWiseの複雑さをナビゲートするために必要な知識とスキルを身に付けることができます。これらのチュートリアルは、ニーズに合わせてさまざまな基本的なトピックを提供します。チュートリアルを詳しく調べながら、 のさまざまな側面に関する貴重なインサイトを発見してください AWS IoT SiteWise。

各チュートリアルでは、特定の機器の例を使用します。これらのチュートリアルはテスト環境を対象としており、架空の会社名、モデル、アセット、プロパティなどを使用しています。目的は、一般的なガイダンスを提供することです。お客様の組織に固有のニーズを満たすかどうかを十分に確認して適応することなく、本番環境で直接使用するためのものではありません。

トピック

- で OEE を計算する AWS IoT SiteWise
- AWS IoT モノ AWS IoT SiteWise から にデータを取り込む
- MQTT 対応 V3 ゲートウェイを使用してデータを SiteWise Edge に統合する
- SiteWise Monitor で風力発電所データを視覚化して共有する
- プロパティ値の更新を Amazon DynamoDB に発行する

で OEE を計算する AWS IoT SiteWise

このチュートリアルでは、製造工程の総合設備効率 (OEE) をコンピューティングする方法の例を示します。その結果、実際の OEE の計算または式は、ここに示すものとは異なる場合があります。一般に、OEE は `Availability * Quality * Performance` として定義されます。OEEの算出について詳しく知りたい方は、[Wikipedia] (ウィキペディア) の[Overall equipment effectiveness] (機器全体の効果) をご覧ください。

前提条件

このチュートリアルを完了するには、次の 3 つのデータストリームを持つデバイスのデータ取り込みを設定する必要があります。

- `Equipment_State` - アイドル、障害、計画された停止、通常の稼働など、マシンの状態を表す数値コード。

- Good_Count - 最後のデータポイント以降に成功した操作の数が各データポイントに含まれているデータストリーム。
- Bad_Count - 最後のデータポイント以降に失敗した操作の数が各データポイントに含まれているデータストリーム。

データ取り込みを設定するには、「にデータを取り込む AWS IoT SiteWise」を参照してください。使用可能な産業オペレーションがない場合は、 AWS IoT SiteWise API を使用してサンプルデータを生成およびアップロードするスクリプトを作成できます。

OEE の計算方法

このチュートリアルでは、Equipment_State、Good_Count、Bad_Count の 3 つのデータ入力ストリームから OEE を計算するアセットモデルを作成します。この例では、砂糖、ポテトチップス、またはペンキなどの包装に使用される汎用包装機器を考えてみます。AWS IoT SiteWise コンソールで、次の測定値、変換、メトリクスを含む AWS IoT SiteWise アセットモデルを作成します。次に、パッケージングマシンを表すアセットを作成し、 が OEE AWS IoT SiteWise を計算する方法を確認できます。

包装機器からの生データストリームを表すために、次の測定を定義します。

測定

- Equipment_State - 数値コードで包装機器の現在の状態を提供するデータストリーム (または測定)。
  - 1024 - マシンはアイドル状態です。
  - 1020 - エラーや遅延などの障害。
  - 1000 - 計画された停止。
  - 1111 - 通常のオペレーション。
- Good_Count - 最後のデータポイント以降に成功した操作の数が各データポイントに含まれているデータストリーム。
- Bad_Count - 最後のデータポイント以降に失敗した操作の数が各データポイントに含まれているデータストリーム。

Equipment_State 測定値データストリームとそれに含まれるコードを使用して、次の変換 (または派生の測定値) を定義できます 変換は、生の測定値と 1 対 1 の関係を持ちます。

変換

- `Idle = eq(Equipment_State, 1024)` - マシンのアイドル状態を含む変換されたデータストリーム。
- `Fault = eq(Equipment_State, 1020)` - マシンの故障状態を含む変換されたデータストリーム。
- `Stop = eq(Equipment_State, 1000)` - マシンの計画された停止状態を含む変換されたデータストリーム。
- `Running = eq(Equipment_State, 1111)` - マシンの通常の動作状態を含む変換されたデータストリーム。

生の測定値と変換された測定値を使用して、指定した時間間隔でマシンデータを集計する次のメトリクスを定義します。このセクションでメトリクスを定義するときは、各メトリクスに同じ時間間隔を選択します。

メトリクス

- `Successes = sum(Good_Count)` - 指定した時間間隔に正常に包装されたパッケージの数。
- `Failures = sum(Bad_Count)` 指定した時間間隔に正常に包装されなかったパッケージの数。
- `Idle_Time = statetime(Idle)` - 指定した時間間隔あたりのマシンの合計アイドル時間 (秒単位)。
- `Fault_Time = statetime(Fault)` - 指定した時間間隔あたりのマシンの合計故障時間 (秒単位)。
- `Stop_Time = statetime(Stop)` - 指定した時間間隔あたりのマシンの計画された停止時間の合計 (秒単位)。
- `Run_Time = statetime(Running)` - 指定した時間間隔あたりのマシンの問題なく実行されている合計時間 (秒単位)。
- `Down_Time = Idle_Time + Fault_Time + Stop_Time` 指定された時間間隔でのマシンの合計ダウンタイム (秒単位)。 以外のマシン状態の合計としてコンピューティングされます。`Run_Time`
- `Availability = Run_Time / (Run_Time + Down_Time)` - 指定された時間間隔のマシンの稼働時間またはマシンが動作できるようスケジュールされた時間の割合。
- `Quality = Successes / (Successes + Failures)` - 指定した時間間隔に正常に包装されたパッケージのマシンの割合。

- `Performance = ((Successes + Failures) / Run_Time) / Ideal_Run_Rate` - 指定された時間間隔でのマシンのパフォーマンスを、プロセスの理想的な実行レート (秒単位) からのパーセンテージで示します。

  たとえば、`Ideal_Run_Rate` が 1 分あたり 60 個のパッケージ (1 秒あたり 1 個のパッケージ) であるとします。`Ideal_Run_Rate` が 1 分単位または 1 時間単位の場合、 `Run_Time`は秒単位であるため適切な単位変換係数で割る必要があります。

- `OEE = Availability * Quality * Performance` - 指定された時間間隔における機械の全体的な設備効率。この式は、OEE を 1 に対する分数として計算します。

> **Note**
>
> OEE が変換として定義された場合、出力値は入力値ごとに計算されます。変換評価では、式内のすべての寄与プロパティで使用可能な最新の値が考慮されるため、予期しない値が生成される可能性があります。同じタイムスタンプを持つプロパティ更新の場合、出力値は他の受信プロパティからの更新によって上書きされる可能性があります。例えば、可用性、品質、およびパフォーマンスが計算されるとき、OEE は他の 2 つのプロパティについて、使用可能な最新のデータポイントで計算されます。これらの寄与値はタイムスタンプを共有し、OEE の出力値が不正確になります。変換の計算では、順序は保証されません。

AWS IoT モノ AWS IoT SiteWise から にデータを取り込む

このチュートリアルでは、デバイスシャドウを使用して AWS IoT 、モノのフリート AWS IoT SiteWise から にデータを取り込む方法について説明します。デバイスシャドウは、 AWS IoT デバイスの現在の状態情報を保存する JSON オブジェクトです。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Device shadow service] (デバイスシャドウサービス) を参照してください。

このチュートリアルを完了したら、モノ AWS IoT SiteWise に基づいて AWS IoT で オペレーションを設定できます。 AWS IoT モノを使用すると、オペレーションを の他の便利な機能と統合できます AWS IoT。たとえば、以下のタスクを実行するように AWS IoT 機能を設定できます。

- データを AWS IoT Events、Amazon DynamoDB、およびその他の AWS サービスにストリーミングするための追加ルールを設定します。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Rules] (ルール) を参照してください。

- AWS IoT フリートインデックス作成サービスを使用して、デバイスデータのインデックス作成、検索、集計を行います。詳細については、「AWS IoT デベロッパーガイド」の「フリートインデックス作成サービス」を参照してください。
- でデバイスを監査して保護します AWS IoT Device Defender。詳細については、「AWS IoT デベロッパーガイド」の 「AWS IoT Device Defender」を参照してください。

このチュートリアルでは、 AWS IoT モノのデバイスシャドウからアセットにデータを取り込む方法について説明します AWS IoT SiteWise。そのためには、1 つ以上の AWS IoT モノを作成し、CPU とメモリの使用状況データで各モノのデバイスシャドウを更新するスクリプトを実行します。このチュートリアルの CPU およびメモリ使用率データを使用して、現実的なセンサーデータを模倣します。次に、モノのデバイスシャドウが更新され AWS IoT SiteWise るたびに でこのデータをアセットに送信する AWS IoT SiteWise アクションを含むルールを作成します。詳細については、「AWS IoT Core ルール AWS IoT SiteWise を使用して にデータを取り込む」を参照してください。

トピック

- 前提条件
- ステップ 1: AWS IoT ポリシーを作成する
- ステップ 2: AWS IoT のモノを作成および設定する
- ステップ 3: デバイスアセットモデルを作成する
- ステップ 4: デバイスフリートアセットモデルを作成する
- ステップ 5: デバイスアセットを作成および設定する
- ステップ 6: デバイスフリートアセットを作成および設定する
- ステップ 7: AWS IoT Core でルールを作成してデバイスアセットにデータを送信する
- ステップ 8: デバイスクライアントスクリプトを実行する
- ステップ 9: チュートリアル後にリソースをクリーンアップする

前提条件

このチュートリアルを完了するには、以下が必要です。

- AWS アカウント。アカウントをお持ちでない場合は、「AWS アカウントを設定する」を参照してください。

- にアクセスUnixするために Windows、、Linux、または macOSを実行する開発用コンピュータ AWS Management Console。詳細については、「AWS Management Consoleの開始方法」を参照してください。
- 管理者権限を持つ AWS Identity and Access Management (IAM) ユーザー。
- Python 開発用コンピュータにインストールされている 3、または AWS IoT モノとして登録するデバイスにインストールされている 3。

ステップ 1: AWS IoT ポリシーを作成する

この手順では、このチュートリアルで使用されるリソースへのアクセスを AWS IoT モノに許可する AWS IoT ポリシーを作成します。

Console

コンソールを使用して AWS IoT ポリシーを作成するには、 AWS IoT Core 次の手順に従います。

AWS IoT ポリシーを作成するには

1. AWS Management Consoleにサインインします。
2. AWS IoT SiteWise がサポートされているAWS リージョンを確認します。必要に応じて、これらのサポートされているいずれかのリージョンに切り替えます。
3. AWS IoT コンソールに移動します。[デバイスに接続] ボタンが表示されたら、それを選択します。
4. 左のナビゲーションペインで、[安全性] を選択し、[ポリシー] を選択します。
5. [作成] を選択します。
6. AWS IoT ポリシーの名前を入力します (例: **`SiteWiseTutorialDevicePolicy`**) 。
7. [ポリシードキュメント] で [JSON] を選択して JSON 形式で次のポリシーを入力します。お客様のリージョンとアカウント ID で *`region`* と *`account-id`* を置き換えます (**`us-east-1`**、**`123456789012`** など)。

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

8. [作成] を選択します。

AWS CLI

> **Important**
>
> このポリシーでは、ワイルドカードを使用して CLI AWS IoT SiteWise のサイズ制限内に留まります。明示的なトピックパスを使用したより制限の厳しいアクセス許可については、代わりに AWS IoT SiteWise コンソールを使用してポリシーを作成します。詳細については、 ???タブで提供されている IoT ポリシーの例を参照してください。

次の AWS CLI コマンドを使用して IoT IoT ポリシーを作成します。

```
aws iot create-policy \
  --policy-name "SiteWiseTutorialDevicePolicy" \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": "iot:Connect",
        "Resource": "arn:aws:iot:region:account-id:client/SiteWiseTutorialDevice*"
      },
      {
        "Effect": "Allow",
        "Action": ["iot:Publish", "iot:Receive"],
        "Resource": [
```

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

ポリシーが正常に作成されたことを確認するには、次のコマンドを使用します。

```
aws iot get-policy --policy-name "SiteWiseTutorialDevicePolicy"
```

このポリシーにより、 AWS IoT デバイスは MQTT メッセージを使用して接続を確立し、デバイスシャドウと通信できるようになります。MQTT メッセージの詳細については、「MQTT とは」を参照してください。デバイスシャドウを操作するために、 AWS IoT モノは で始まるトピックで MQTT メッセージを発行および受信します`$aws/things/thing-name/shadow/`。このポリシーには、`${iot:Connection.Thing.ThingName}` と呼ばれるモノのポリシー変数が組み込まれています。この変数は、各トピックで接続されたモノの名前に置き換わります。`iot:Connect` ステートメントはデバイスが接続を確立できる制限を設定し、モノのポリシー変数は、`SiteWiseTutorialDevice` で始まる名前のみに置き換えることができます。

詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Thing policy variables] (シングポリシー変数) を参照してください。

> **Note**
>
> このポリシーは、名前が `SiteWiseTutorialDevice` で始まるモノに適用されます。モノに別の名前を使用するには、それに応じてポリシーを更新する必要があります。

ステップ 2: AWS IoT のモノを作成および設定する

この手順では、 AWS IoT モノを作成して設定します。開発用コンピュータを AWS IoT モノとして指定できます。進行するにつれて、ここで学習している原則は実際のプロジェクトに適用できることに注意してください。や FreeRTOS など、 AWS IoT SDK を実行できる任意のデバイスで AWS IoT モノを柔軟に作成 AWS IoT Greengrass およびセットアップできます。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[AWS IoT SDKs] (SDK) を参照してください。

Console

AWS IoT モノを作成して設定するには

1. コマンドラインを開き、次のコマンドを実行して、このチュートリアル用のディレクトリを作成します。

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

2. 次のコマンドを実行して、モノの証明書用のディレクトリを作成します。

   ```
   mkdir device1
   ```

   追加のモノを作成する場合は、それに応じてディレクトリ名の番号を増やして、どの証明書がどのモノに属しているかをわかるようにします。

3. AWS IoT コンソールに移動します。
4. 左側のナビゲーションペインで、[管理] セクションの [すべてのデバイス] を選択します。[モノ] を選択します。
5. [まだモノがありません] ダイアログボックスが表示された場合は、[モノの作成] を選択します。それ以外の場合は、[モノの作成] を選択します。
6. [モノを作成する] ページで、[単一のモノを作成する] を選択し、[次へ] を選択します。

7. [モノのプロパティを指定する] ページで、 AWS IoT のモノの名前 (例: **SiteWiseTutorialDevice1**) を入力し、[次へ] を選択します。追加のモノを作成する場合は、それに応じてモノの名前の番号を増やします。

> **Important**
>
> モノの名前は、ステップ 1: ポリシーの作成で作成した AWS IoT ポリシーで使用される名前と一致する必要があります。それ以外の場合、デバイスは に接続できません AWS IoT。

8. デバイス証明書の設定 - オプションページで、新しい証明書の自動生成 (推奨) を選択し、次へを選択します。証明書により AWS IoT 、 はデバイスを安全に識別できます。
9. 証明書にポリシーをアタッチする - オプションページで、ステップ 1: ポリシーの作成で作成した AWS IoT ポリシーを選択し、モノの作成を選択します。
10. [証明書とキーのダウンロード] ダイアログボックスで、次の操作を行います。
    a. [ダウンロード] リンクを選択して、モノの証明書、パブリックキー、プライベートキーをダウンロードします。モノの証明書用に作成したディレクトリ (例: **iot-sitewise-rule-tutorial/device1**) に 3 つのファイルをすべて保存します。

    > **Important**
    >
    > モノの証明書とキーをダウンロードできるのは今回だけです。この証明書とキーは、デバイスが AWS IoTに正常に接続するために必要です。

    b. [ダウンロード] のリンクを選択し、ルート CA 証明書をダウンロードします。ルート CA 証明書を **iot-sitewise-rule-tutorial** に保存します。Amazon Root CA 1 をダウンロードすることをお勧めします。
11. [完了] をクリックします。

AWS CLI

を使用して AWS IoT モノを作成および設定するには、次の手順に従います AWS CLI。

1. コマンドラインを開き、次のコマンドを実行して、このチュートリアル用のディレクトリを作成します。

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

2. チュートリアルディレクトリに移動します。

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

3. 次のコマンドを実行して、モノの証明書のディレクトリを作成します。

```
mkdir device1
```

追加のモノを作成する場合は、それに応じてディレクトリ名の番号を増やして、どの証明書がどのモノに属しているかをわかるようにします。

4. AWS IoT モノを作成します。

```
aws iot create-thing --thing-name "SiteWiseTutorialDevice1"
```

> **Important**
> モノの名前は、ステップ 1 で作成したポリシーで使用される名前パターンと一致する必要があります。それ以外の場合、デバイスは に接続できません AWS IoT。

5. 証明書を作成し、ファイルを保存します。出力の証明書 ARN を書き留めます。次のステップで必要になります。

```
aws iot create-keys-and-certificate \
    --set-as-active \
    --certificate-pem-outfile "device1/device.pem.crt" \
    --public-key-outfile "device1/public.pem.key" \
    --private-key-outfile "device1/private.pem.key"
```

6. ステップ 1 で作成したポリシーを証明書にアタッチします。

```
aws iot attach-policy \
    --policy-name "SiteWiseTutorialDevicePolicy" \
    --target "certificate-arn"
```

7. モノに証明書をアタッチします。

```
aws iot attach-thing-principal \
```

```
    --thing-name "SiteWiseTutorialDevice1" \
    --principal "certificate-arn"
```

8. Amazon ルート CA 証明書をダウンロードします。

```
curl https://www.amazontrust.com/repository/AmazonRootCA1.pem >
 AmazonRootCA1.pem
```

この証明書は、デバイスが に正常に接続するために必要です AWS IoT。

> **Important**
>
> 証明書とキーを安全に保存します。これらの認証情報は、作成後に再度ダウンロードすることはできません。

これで、コンピュータに AWS IoT モノが登録されました。次のいずれかのステップを実行します。

- ステップ 3: 追加のモノを作成せずにデバイスアセットモデルを作成するに進みます。 AWS IoT このチュートリアルは、1 つのモノだけで完了できます。
- 別のコンピュータまたはデバイスでこのセクションの手順を繰り返して、さらに多くの AWS IoT のモノを作成します。このチュートリアルでは、複数のデバイスから一意の CPU およびメモリ使用率データを取り込むことができるよう、このオプションに従うことをお勧めします。
- 同じデバイス (コンピュータ) でこのセクションのステップを繰り返して、さらに多くの AWS IoT のモノを作成します。各 AWS IoT モノはコンピュータから同様の CPU とメモリの使用状況データを受け取るため、このアプローチを使用して、複数のデバイスから一意でないデータを取り込む方法を示します。

ステップ 3: デバイスアセットモデルを作成する

この手順では、CPU とメモリの使用状況データをストリーミングするデバイスを表すアセットモデル AWS IoT SiteWise を に作成します。デバイスのグループを表すアセット内のデータを処理するために、アセットモデルは、同じ型の複数のアセットに対して一貫した情報を適用します。詳細については、「産業用アセットをモデリングする」を参照してください。

デバイスを表すアセットモデルを作成するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左のナビゲーションペインで [モデル] を選択します。
3. アセットモデルの作成 を選択します。
4. [モデルの詳細] で、モデルの名前を入力します。例えば、**SiteWise Tutorial Device Model**。
5. [測定定義] で、次の操作を実行します。

   a. [名前] に「**CPU Usage**」と入力します。

   b. [単位] に「**%**」と入力します。

   c. [データ型] は [倍精度浮動小数点型] のままにします。

   測定プロパティは、デバイスの raw データストリームを表します。詳細については、「機器からのデータストリームを定義する (測定値) 」を参照してください。
6. 2 つ目の測定プロパティを追加する場合は、[新しい測定の追加] を選択します。
7. [測定定義] の 2 行目で、次の操作を実行します。

   a. [名前] に「**Memory Usage**」と入力します。

   b. [単位] に「**%**」と入力します。

   c. [データ型] は [倍精度浮動小数点型] のままにします。
8. [メトリクス定義] で、次の操作を実行します。

   a. [名前] に「**Average CPU Usage**」と入力します。

   b. [計算式] に「**avg(CPU Usage)**」と入力します。表示されたら、自動入力リストから CPU Usage を選択します。

   c. [時間間隔] に「**5 minutes**」と入力します。

   メトリクスプロパティは、ある間隔ですべての入力データポイントを処理し、間隔ごとに 1 つのデータポイントを出力する集計計算を定義します。このメトリクスプロパティは、各デバイスの平均 CPU 使用率を 5 分ごとに計算します。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。
9. [新しいメトリクスの追加] を選択して、2 つ目のメトリクスプロパティを追加します。
10. [メトリクス定義] の 2 行目で、次の操作を実行します。

a. [名前] に「**Average Memory Usage**」と入力します。

b. [計算式] に「**avg(Memory Usage)**」と入力します。表示されたら、自動入力リストから Memory Usage を選択します。

c. [Time interval (時間間隔)] に「**5 minutes**」と入力します。

このメトリクスプロパティは、各デバイスの平均メモリ使用率を 5 分ごとに計算します。

11. (オプション) デバイスごとのコンピューティングに関心のあるその他のメトリクスを追加します。興味深い関数には、min、max などがあります。詳細については、「数式を使用する」を参照してください。「ステップ 4: デバイスフリートアセットモデルを作成する」では、デバイスのフリート全体のデータを使用してメトリクスを計算できる親アセットを作成します。

12. [モデルの作成] を選択します。

ステップ 4: デバイスフリートアセットモデルを作成する

この手順では、 でアセットモデルを作成し AWS IoT SiteWise 、デバイスのコレクションをシンボル化します。このアセットモデル内で、多数のデバイスアセットを 1 つの包括的なフリートアセットにリンクできる構造を確立します。その後、フリートアセットモデルのメトリクスの概要を示し、接続されたすべてのデバイスアセットのデータを統合します。このアプローチにより、フリート全体の総合的なパフォーマンスに関する包括的なインサイトが得られます。

デバイスフリートを表すアセットモデルを作成するには

1. AWS IoT SiteWise コンソールに移動します。

2. 左のナビゲーションペインで [モデル] を選択します。

3. アセットモデルの作成 を選択します。

4. [モデルの詳細] で、モデルの名前を入力します。例えば、**SiteWise Tutorial Device Fleet Model**。

5. [階層定義] で、次の操作を実行します。

   a. [階層名] に「**Device**」と入力します。

   b. [階層モデル] で、デバイスアセットモデル (**SiteWise Tutorial Device Model**) を選択します。

階層は、親 (フリート) アセットモデルと子 (デバイス) アセットモデルの間の関係を定義します。親アセットは、子アセットのプロパティデータにアクセスできます。後でアセットを作成する場合は、親アセットモデルの階層定義に従って、子アセットを親アセットに関連付ける必要があります。詳細については、「アセットモデルの階層を定義する」を参照してください。

6. [Metric definitions (メトリクス定義)] で、次の操作を実行します。

   a. [名前] に「**Average CPU Usage**」と入力します。

   b. [計算式] に「**avg(Device | Average CPU Usage)**」と入力します。自動入力リストが表示されたら、[Device] を選択して階層を選択し、[Average CPU Usage] を選択して前に作成したデバイスアセットからメトリクスを選択します。

   c. [Time interval (時間間隔)] に「**5 minutes**」と入力します。

   このメトリクスプロパティは、**Device** 階層を通じてフリートアセットに関連付けられているすべてのデバイスアセットの平均 CPU 使用率を計算します。

7. [新しいメトリクスの追加] を選択して、2 つ目のメトリクスプロパティを追加します。

8. [メトリクス定義] の 2 行目で、次の操作を実行します。

   a. [名前] に「**Average Memory Usage**」と入力します。

   b. [計算式] に「**avg(Device | Average Memory Usage)**」と入力します。自動入力リストが表示されたら、[Device] を選択して階層を選択し、[Average Memory Usage] を選択して前に作成したデバイスアセットからメトリクスを選択します。

   c. [Time interval (時間間隔)] に「**5 minutes**」と入力します。

   このメトリクスプロパティは、**Device** 階層を通じてフリートアセットに関連付けられているすべてのデバイスアセットの平均メモリ使用率を計算します。

9. (オプション) デバイスのフリート全体のコンピューティングに関心のあるその他のメトリクスを追加します。

10. [モデルの作成] を選択します。

ステップ 5: デバイスアセットを作成および設定する

この手順では、デバイスアセットモデルに基づくデバイスアセットを生成します。次に、測定プロパティごとにプロパティエイリアスを定義します。プロパティエイリアスは、アセットプロパティを識

別する一意の文字列です。後で、アセット ID とプロパティ ID の代わりにエイリアスを使用して、データアップロードのプロパティを識別できます。詳細については、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

デバイスアセットを作成し、プロパティエイリアスを定義するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションペインで [アセット] を選択します。
3. [アセットの作成] を選択します。
4. [モデル情報] で、デバイスアセットモデル **SiteWise Tutorial Device Model** を選択します。
5. [アセット情報] で、アセットの名前を入力します。例えば、**SiteWise Tutorial Device 1**。
6. [アセットの作成] を選択します。
7. 新しいデバイスアセットの場合は、 [編集] を選択します。
8. 測定値の下：
   a. CPU Usage で、プロパティエイリアスとして **/tutorial/device/SiteWiseTutorialDevice1/cpu** と入力します。 AWS IoT モノの名前をプロパティエイリアスに含めると、単一の AWS IoT ルールを使用してすべてのデバイスからデータを取り込むことができます。
   b. Memory Usage で、プロパティエイリアスとして **/tutorial/device/SiteWiseTutorialDevice1/memory** と入力します。
9. [保存] を選択します。

以前に複数の AWS IoT モノを作成した場合は、デバイスごとにステップ 3 ~ 10 を繰り返し、それに応じてアセット名とプロパティエイリアスの数値を増やします。たとえば、2 つ目のデバイスアセットの名前は **SiteWise Tutorial Device 2** となり、そのプロパティエイリアスは **/tutorial/device/SiteWiseTutorialDevice2/cpu** および **/tutorial/device/SiteWiseTutorialDevice2/memory** となります。

ステップ 6: デバイスフリートアセットを作成および設定する

この手順では、デバイスフリートアセットモデルからデバイスフリートアセットを作成します。次に、個々のデバイスアセットをフリートアセットにリンクします。この関連付けにより、フリートア

セットのメトリクスプロパティは、複数のデバイスからのデータをコンパイルおよび分析できます。このデータにより、フリート全体の集合的なパフォーマンスをまとめて確認できます。

デバイスフリートアセットを作成してデバイスアセットを関連付けるには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションペインで [アセット] を選択します。
3. [アセットの作成] を選択します。
4. [モデル情報]で、デバイスフリートアセットモデル **SiteWise Tutorial Device Fleet Model** を選択します。
5. [アセット情報] で、アセットの名前を入力します。例えば、**SiteWise Tutorial Device Fleet 1**。
6. [アセットの作成] を選択します。
7. 新しいデバイスフリートアセットの場合は、 [編集] を選択します。
8. [このアセットに関連付けられているアセット] で、[関連付けられたアセットの追加] を選択し、以下を行います。

   a. [階層] で、[Device] を選択します。この階層は、デバイスとフリートアセットの間の階層関係を識別します。この階層は、このチュートリアルで前にデバイスフリートアセットモデルで定義しました。

   b. [アセット] で、デバイスアセット (SiteWise Tutorial Device 1) を選択します。
9. (オプション) 以前に複数のデバイスアセットを作成した場合は、作成したデバイスアセットごとにステップ 8 ~ 10 を繰り返します。
10. [保存] を選択します。

    これで、デバイスアセットが階層として編成されているのがわかります。

ステップ 7: AWS IoT Core でルールを作成してデバイスアセットにデータを送信する

この手順では、 AWS IoT Coreでルールを確立します。このルールは、デバイスシャドウからの通知メッセージを解釈し、 のデバイスアセットにデータを送信するように設計されています AWS IoT SiteWise。デバイスのシャドウが更新されるたびに、 AWS IoT は MQTT メッセージを送信します。MQTT メッセージに基づいてデバイスのシャドウが変更されたときにアクションを実行する

ルールを作成できます。この場合、この目的は、更新メッセージを処理し、プロパティ値を抽出して、 AWS IoT SiteWiseのデバイスアセットに送信することです。

AWS IoT SiteWise アクションを使用してルールを作成するには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで [メッセージルーティング] を選択し、[ルール] を選択します。
3. [ルールを作成] を選択します。
4. ルールの名前と説明を入力し、[次へ] を選択します。
5. 次の SQL ステートメントを入力し、[次へ] を選択します。

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

このルールクエリステートメントは、デバイスシャドウサービスがシャドウの更新を `$aws/things/thingName/shadow/update/accepted` にパブリッシュするため、機能します。デバイスシャドウの詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の [Device shadow service] (デバイスシャドウサービス) を参照してください。

`WHERE` 句では、このルールクエリステートメントは `topic(3)` 関数を使用して、トピックの 3 番目のセグメントからモノの名前を取得します。次に、このステートメントは、名前がチュートリアルのデバイス名と一致しないデバイスを除外します。 AWS IoT SQL の詳細については、「AWS IoT デベロッパーガイド」のAWS IoT 「SQL リファレンス」を参照してください。

6. [ルールのアクション] で、[AWS IoT SiteWiseのアセットプロパティにメッセージデータを送信する] を選択し、次の操作を行います。

   a. [プロパティエイリアスによる] を選択します。

   b. [Pプロパティエイリアス] に「**`/tutorial/device/${topic(3)}/cpu`**」と入力します。

      `${...}` 構文は置換テンプレートです。 は中括弧内のコンテンツ AWS IoT を評価します。この置換テンプレートは、トピックからモノの名前を取得し、モノごとに固有のエイリアスを作成します。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Substitution templates] (置換テンプレート) を参照してください。

> **Note**
>
> 置換テンプレート内の式は SELECT ステートメントとは別に評価されるため、AS 句を使用して作成されたエイリアスは置換テンプレートを使用して参照することはできません。サポートされている関数と演算子に加えて、元のペイロードに存在する情報のみを参照できます。

c. [エントリ ID - オプション] に **`${concat(topic(3), "-cpu-", floor(state.reported.timestamp))}`** と入力します。

   エントリ ID は、各値エントリの試行を一意に識別します。エントリがエラーを返す場合は、エラーの出力でエントリ ID を検索して、問題をトラブルシューティングすることができます。このエントリ ID の置換テンプレートは、モノの名前とデバイスの報告されたタイムスタンプを組み合わせたものです。たとえば、結果のエントリ ID は `SiteWiseTutorialDevice1-cpu-1579808494` のようになります。

d. [時間 (秒)] に「**`${floor(state.reported.timestamp)}`**」と入力します。

   この置換テンプレートは、デバイスの報告されたタイムスタンプから秒単位で時間を計算します。このチュートリアルでは、デバイスは Unix エポック時間でタイムスタンプ (秒) を浮動小数点数として報告します。

e. [オフセット (ナノ秒) - オプション] に **`${floor((state.reported.timestamp % 1) * 1E9)}`** と入力します。

   この置換テンプレートは、デバイスの報告されたタイムスタンプの小数部分を変換して、秒単位の時間からのナノ秒オフセットを計算します。

> **Note**
>
> AWS IoT SiteWise では、データが Unix エポック時間で現在のタイムスタンプを持っている必要があります。デバイスが正確に時間を報告しない場合、[timestamp()] (タイムスタンプ) を使って AWS IoT ルールエンジンから現在の時間を取得することができます。この関数は、時刻をミリ秒単位で報告するため、ルールアクションの時間パラメータを次の値に更新する必要があります。
>
> - [Time in seconds (時間 (秒))] に「**`${floor(timestamp() / 1E3)}`**」と入力します。

- [オフセット (ナノ秒)] に「**${(timestamp() % 1E3) * 1E6}**と入力します。

f. [データ型] で、[Double (倍精度浮動小数点型)] を選択します。

このデータ型は、アセットモデルで定義したアセットプロパティのデータ型と一致する必要があります。

g. [値] には「**${state.reported.cpu}**」と入力します。置換テンプレートでは、. 演算子を使用して JSON 構造内から値を取得します。

h. [エントリの追加] を選択して、メモリ使用率プロパティに新しいエントリを追加し、そのプロパティに対して次の手順を再度実行します。

i. [プロパティエイリアスによる] を選択します。

ii. [Pプロパティエイリアス] に「**/tutorial/device/${topic(3)}/memory**」と入力します。

iii. [エントリ ID - オプション] に **${concat(topic(3), "-memory-", floor(state.reported.timestamp))}** と入力します。

iv. [時間 (秒)] に「**${floor(state.reported.timestamp)}**」と入力します。

v. [オフセット (ナノ秒) - オプション] に **${floor((state.reported.timestamp % 1) * 1E9)}** と入力します。

vi. [データ型] で、[倍精度浮動小数点型] を選択します。

vii. [値] には「**${state.reported.memory}**」と入力します。

i. [ロール] で、[ロールの作成] を選択して、このルールアクションに対する IAM ロールを作成します。このロールにより、 AWS IoT はデバイスフリートアセットとそのアセット階層のプロパティにデータをプッシュできます。

j. ロール名を入力し、[作成] を選択します。

7. (オプション) ルールのトラブルシューティングに使用できるエラーアクションを設定します。詳細については、「ルールをトラブルシューティング (AWS IoT SiteWise)」を参照してください。

8. [次へ] を選択します。

9. 設定を確認し、[作成] を選択してルールを作成します。

ステップ 8: デバイスクライアントスクリプトを実行する

このチュートリアルでは、データをレポートするために実際のデバイスは使用していません。代わりに、スクリプトを実行して CPU とメモリの使用量で AWS IoT モノのデバイスシャドウを更新し、実際のセンサーデータを模倣します。スクリプトを実行するには、まず、必要な Python パッケージをインストールする必要があります。この手順では、必要な Python パッケージをインストールしてから、デバイスクライアントスクリプトを実行します。

デバイスクライアントスクリプトを設定および実行するには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインの下部にある [設定] を選択します。
3. デバイスクライアントスクリプトで使用するためにカスタムエンドポイントを保存します。このエンドポイントを使用して、モノのシャドウとやり取りします。このエンドポイントは、現在のリージョンのアカウントに固有です。

   カスタムエンドポイントは、次の例のようになります。

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

4. コマンドラインを開き、次のコマンドを実行して、先ほど作成したチュートリアル用のディレクトリに移動します。

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

5. 以下のコマンドを実行して、 AWS IoT Device SDK for Pythonをインストールします。

   ```
   pip3 install AWSIoTPythonSDK
   ```

   詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[AWS IoT Device SDK for Python]を参照してください。
6. 次のコマンドを実行して、psutil をインストールします。psutil は、クロスプラットフォームプロセスおよびシステムユーティリティライブラリです。

   ```
   pip3 install psutil
   ```

   詳細については、[Python Package Index] (パッケージインデックス) の[psutil]を参照してください。

7. `iot-sitewise-rule-tutorial` ディレクトリ `thing_performance.py` にというファイルを作成し、次の Python コードをファイルにコピーします。

```
import AWSIoTPythonSDK.MQTTLib as AWSIoTPyMQTT

import json
import psutil
import argparse
import logging
import time


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

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


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

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

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

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

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

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

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

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


# Configures debug logging for the AWS IoT Device SDK for Python.
def configureLogging():
    logger = logging.getLogger("AWSIoTPythonSDK.core")
    logger.setLevel(logging.DEBUG)
    streamHandler = logging.StreamHandler()
    formatter = logging.Formatter(
        "%(asctime)s - %(name)s - %(levelname)s - %(message)s"
    )
    streamHandler.setFormatter(formatter)
    logger.addHandler(streamHandler)


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

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

8. 以下のパラメータを使用して、コマンドラインから **thing_performance.py** を実行します。

   - `-n`、 `--thingName` - モノの名前 (**SiteWiseTutorialDevice1** など)。
   - `-e`、 `--endpoint` – この手順の前半で保存したカスタム AWS IoT エンドポイント。
   - `-r`、 `--rootCA` — AWS IoT ルート CA 証明書へのパス。
   - `-c`、 `--cert` — AWS IoT モノの証明書へのパス。
   - `-k`、 `--key` — AWS IoT モノの証明書のプライベートキーへのパス。
   - `-d`、 `--requestDelay` -(オプション) 各デバイスシャドウ更新間の待機時間 (秒)。デフォルトは 1 秒です。
   - `-v`、 `--enableLogging` -(オプション) このパラメータが存在する場合、スクリプトによって AWS IoT Device SDK for Pythonからのデバッグメッセージが出力されます。

   コマンドは、次の例のようになります。

```
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
```

   追加の AWS IoT モノのためにスクリプトを実行している場合は、それに応じてモノの名前と証明書ディレクトリを更新します。

9. デバイスでプログラムを開いたり閉じたりして、CPU およびメモリ使用率がどのように変化するかを確認します。スクリプトにより、各 CPU およびメモリ使用率の読み取りが出力されます。スクリプトがデータをデバイスシャドウサービスに正常にアップロードする場合、スクリプトの出力は次の例のようになります。

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

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

10. スクリプトがデバイスシャドウを更新していることを確認するには、次の手順を実行します。

    a. AWS IoT コンソールに移動します。

    b. 左側のナビゲーションペインで、[すべてのデバイス] を選択し、[モノ] を選択します。

    c. モノの SiteWiseTutorialDevice1 を選択します。

    d. [デバイスシャドウ] タブを選択し、[クラシックシャドウ] を選択して、シャドウの状態が次の例のようになっていることを確認します。

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

    モノのシャドウの状態が空であるか、前の例のようになります。スクリプトが実行され、正常に接続されていることを確認します AWS IoT。への接続時にスクリプトが引き続きタイムアウトする場合は AWS IoT、このチュートリアルに従ってモノのポリシーが設定されていることを確認します。

11. ルールアクションが AWS IoT SiteWiseにデータを送信していることを確認するには、次の手順を実行します。

    a. AWS IoT SiteWise コンソールに移動します。

    b. 左側のナビゲーションペインで [アセット] を選択します。

    c. デバイスフリートアセット (SiteWise Tutorial Device Fleet 1) の横にある矢印を選択してアセット階層を展開し、デバイスアセット (SiteWise Tutorial Device1) を選択します。

    d. [測定] を選択します。

    e. 最新値 セルに、CPU Usage および Memory Usage プロパティの値があることを確認します。

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 プロパティと Memory Usage プロパティ に最新の値がない場合は、ページを更新します。数分経過しても値が表示されない場合は、「ルールをトラブルシューティング (AWS IoT SiteWise)」を参照してください。

これで、このチュートリアルは終了です。データのライブ可視化を調べる場合は、 AWS IoT SiteWise Monitorでポータルを設定できます。詳細については、「でデータをモニタリングする AWS IoT SiteWise Monitor」を参照してください。それ以外の場合は、コマンドプロンプトで Ctrl+C キーを押して、デバイスクライアントスクリプトを停止できます。Python プログラムが、料金がかかる量のメッセージを送信することはほとんどありませんが、終了したらプログラムを停止することをお勧めします。

ステップ 9: チュートリアル後にリソースをクリーンアップする

> **Note**
>
> このチュートリアルで作成したリソースは、データを SiteWise Edge に統合チュートリアルに必要です。このステップを完了する予定がある場合は、このステップのリソースをクリーンアップしないでください。

AWS IoT モノからのデータの取り込みに関するチュートリアルを完了したら、追加料金が発生しないようにリソースをクリーンアップします。

で階層アセットを削除するには AWS IoT SiteWise

1. [AWS IoT SiteWise console] (コンソール) に移動します。
2. 左側のナビゲーションペインで [Assets (アセット)] を選択します。
3. でアセットを削除するときは AWS IoT SiteWise、まずアセットの関連付けを解除する必要があります。

   デバイスフリートアセットからデバイスアセットの関連付けを解除するには、次の手順を実行します。

   a. デバイスフリートアセット () を選択しますSiteWise Tutorial Device Fleet 1。
   b. [編集] を選択します。
   c. [このアセットに関連付けられているアセット] で、このデバイスフリートアセットに関連付けられている各デバイスアセットに対して [関連付け解除] を選択します。
   d. [保存] を選択します。

   これで、デバイスアセットが階層として編成されなくなったことがわかります。

4. デバイスアセット (SiteWise Tutorial Device 1) を選択します。
5. [削除] をクリックします。
6. 確認フィールドに「**Delete**」と入力し、[削除] を選択します。
7. デバイスアセットとデバイスフリートアセット (SiteWise Tutorial Device Fleet 1) ごとに、ステップ 4 ~ 6 を繰り返します。

で階層アセットモデルを削除するには AWS IoT SiteWise

1. AWS IoT SiteWise コンソールに移動します。
2. デバイスおよびデバイスフリートアセットを削除していない場合は、削除します。詳細については、前述の手順を参照してください。モデルから作成されたアセットがある場合は、そのモデルを削除できません。
3. 左のナビゲーションペインで [モデル] を選択します。
4. デバイスフリートアセット (SiteWise Tutorial Device Fleet Model 1) を選択します。

   階層アセットモデルを削除する場合は、最初に親アセットモデルを削除する必要があります。

5. [削除] を選択します。
6. 確認フィールドに「**Delete**」と入力し、[削除] を選択します。
7. デバイスのアセットモデル (SiteWise Tutorial Device Model) について、ステップ 4 ~ 6 を繰り返します。

でルールを無効化または削除するには AWS IoT Core

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで [メッセージルーティング] を選択し、[ルール] を選択します。
3. ルールを選択し、[削除] を選択します。

4. 確認ダイアログボックスでルール名を入力し、[削除] を選択します。

MQTT 対応 V3 ゲートウェイを使用してデータを SiteWise Edge に統合する

このチュートリアルでは、MQTT メッセージングプロトコルを使用するサードパーティーのデバイスとセンサーを AWS IoT SiteWise MQTT 対応 V3 ゲートウェイと統合する方法について説明します。MQTT 対応デバイスからデータを収集してモニタリングするように AWS IoT SiteWise エッジゲートウェイを設定する方法について説明します。 AWS IoT SiteWise を使用すると、産業機器データを収集、処理、モニタリングできます。SiteWise Edge 機能を使用して、産業用 IoT オペレーションを最適化し、未加工データを実用的なインサイトに変換します。

このチュートリアルでは、風力発電所のデモンストレーションのデータを使用して、主要な概念を説明します。プロセスに慣れたら、独自のデータを使用してチュートリアルを繰り返すことができます。

このチュートリアルを完了したら、次の項目を実行できます。

- 産業用デバイスからデータを受信するように MQTT 対応の V3 ゲートウェイを設定および設定する
- エッジの機器からの受信 MQTT メッセージを処理および検証する
- サードパーティーの視覚化プラットフォーム AWS IoT SiteWise を使用して でデバイスデータを表示する
- エッジゲートウェイから に処理されたデータを送信 AWS クラウド して、一元化されたストレージとさらなる分析を可能にする

さらに、他の AWS IoT サービスに接続して次のタスクを実行することで、エッジゲートウェイ機能を活用できます。

- Amazon Amazon S3 Timestream、 などのサービスにデータをルーティングする AWS IoT ルールを設定しますAWS Lambda。
- AWS IoT Device Defender を使用して、ゲートウェイ設定をリモートで管理および更新します。
- AWS IoT セキュリティ機能を使用して、安全なデバイス認証と認可を実装します。詳細については、「 AWS IoT デベロッパーガイド」のAWS IoT 「 セキュリティ」を参照してください。
- 機器データに基づいて自動アラートと通知を作成します。詳細については、「 AWS IoT デベロッパーガイド」の「 のルール AWS IoT」を参照してください。

> **Note**
>
> このチュートリアルでは、サードパーティーのサービス、ツール、ドキュメントを参照します。 AWS は、サードパーティーの製品やサービスのベンダーまたはサプライヤーではなく、外部プロバイダーからの情報の正確性を保証することはできません。デプロイする前に、すべてのサードパーティーツールを評価して検証します。

トピック

- 前提条件
- ステップ 1: AWS IoT ポリシーを作成する
- ステップ 2: AWS IoT モノを作成して設定する
- ステップ 3: SiteWise Edge MQTT 対応 V3 ゲートウェイを設定する
- ステップ 4: SiteWise Edge ゲートウェイソフトウェアをインストールする
- ステップ 5: 外部アプリケーションに接続するように EMQX ブローカーを設定する
- ステップ 6: Mosquitto でデータを発行する
- ステップ 7: 送信先を指定する
- ステップ 8: パスフィルターを指定する
- ステップ 9: AWS IoT リソースを設定する
- ステップ 10: データを視覚化する
- ステップ 11: チュートリアルの後にリソースをクリーンアップする
- 追加リソース

前提条件

このチュートリアルを完了するには、以下が必要です。

- AWS アカウント。アカウントをお持ちでない場合は、「AWS アカウントを設定する」を参照してください。
- 管理者権限を持つ AWS Identity and Access Management (IAM) ユーザー。詳細については、「の ID とアクセスの管理 AWS IoT SiteWise」を参照してください。
- デバイスにPythonインストールされている の最新バージョン。

> **Important**
>
> このチュートリアルでは、Ingest data チュートリアルで作成されたリソースを使用する必要があります。このチュートリアルに進む前に完了する必要があります。

ステップ 1: AWS IoT ポリシーを作成する

このチュートリアルでは、データ取り込みチュートリアルで作成した AWS IoT ポリシーを使用します。このポリシーは、デバイスのセキュリティルールを設定し、外部デバイスとセンサーのデジタル表現を作成します AWS IoT。このポリシーは、サードパーティーデバイスが MQTT (Message Queuing Telemetry Transport) AWS IoT Core を使用して にデータを送信することを許可します。MQTT メッセージの詳細については、「MQTT とは」を参照してください。

Console

AWS IoT ポリシーが完了していることを確認します。詳細な手順については、「データを取り込む」チュートリアルの「ステップ 1」を参照してください。 ???

アクティブな AWS IoT ポリシーがあることを確認するには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで、証券、ポリシーを選択します。
3. 作成したポリシーを選択します。例えば、**SiteWiseTutorialDevicePolicy**。
4. ポリシーのステータスがアクティブとしてリストされていることを確認します。

AWS CLI

AWS IoT ポリシーが完了していることを確認します。詳細な手順については、「データを取り込む」チュートリアルの「ステップ 1」を参照してください。 ???

AWS CLI コマンドリファレンスの次の AWS CLI get-policy コマンドを使用して、アクティブな AWS IoT ポリシーがあることを確認します。

```
aws iot get-policy --policy-name "SiteWiseTutorialDevicePolicy"
```

このポリシーにより、 AWS IoT デバイスは MQTT メッセージを使用して接続を確立し、デバイスシャドウと通信できるようになります。デバイスシャドウを操作するために、 AWS IoT モノ

は で始まるトピックで MQTT メッセージを発行および受信します`$aws/things/thing-name/shadow/`。このポリシーには、`${iot:Connection.Thing.ThingName}` と呼ばれるモノのポリシー変数が組み込まれています。この変数は、各トピックで接続されたモノの名前に置き換わります。`iot:Connect` ステートメントはデバイスが接続を確立できる制限を設定し、モノのポリシー変数は、`SiteWiseTutorialDevice` で始まる名前のみに置き換えることができます。

詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Thing policy variables] (シングポリシー変数) を参照してください。

> **Note**
>
> このポリシーは、名前が `SiteWiseTutorialDevice` で始まるモノに適用されます。モノに別の名前を使用するには、それに応じてポリシーを更新する必要があります。

ステップ 2: AWS IoT モノを作成して設定する

このステップでは、エッジデバイスを AWS IoT モノとして登録し、 AWS IoT SiteWise Edge との安全な通信に必要なモノの証明書とキーを生成します。このプロセスは、デバイスが MQTT 対応 V3 ゲートウェイを介してサードパーティーデータを送信するための基盤を確立します。

Console

AWS IoT モノの作成ステップと設定ステップが完了していることを確認します。詳細な手順については、「データを取り込む」チュートリアルの「ステップ 2」を参照してください。

アクティブな AWS IoT モノがあることを確認するには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで、すべてのデバイス、モノを選択します。
3. 作成したモノを選択します。例えば、**`SiteWiseTutorialDevice1`**。
4. 証明書で、ステータスがアクティブとしてリストされていることを確認します。

AWS CLI

AWS IoT モノの作成ステップと設定ステップが完了していることを確認します。詳細な手順については、「データを取り込む」チュートリアルの「ステップ 2」を参照してください。 ???

次の AWS CLI コマンドを使用して、アクティブな AWS IoT ポリシーがあることを確認します。

```
aws iot describe-thing --thing-name "SiteWiseTutorialDevice1"
```

これらのステップを完了すると、デバイスを AWS IoT SiteWise Edge に安全に接続できます。MQTT 認証用に生成した証明書とキーを保存するローカルディレクトリを作成しました。デバイスは AWS IoT コンソールに AWS IoT モノとして登録され、デバイスはデータを SiteWise Edge と統合する準備が整います。産業機器やその他のデバイスを AWS IoT プラットフォームに接続し、SiteWise Edge へのデータの取り込みを開始できます。

ステップ 3: SiteWise Edge MQTT 対応 V3 ゲートウェイを設定する

このステップでは、 AWS IoT SiteWise Edge MQTT 対応 V3 ゲートウェイを作成し、EMQX ブローカーからデータを受信するように設定します。ゲートウェイは、デバイスと の間のブリッジとして機能します AWS IoT。これにより、 に送信する前に、エッジでローカルにデータを処理できます AWS クラウド。この設定により、帯域幅が減少し、クラウド処理の遅延が軽減されます。

Console

AWS IoT SiteWise MQTT 対応 V3 ゲートウェイを作成するには

1. AWS Management Console にサインインして、AWS IoT SiteWise コンソール を開きます。
2. 左側のナビゲーションペインで、エッジゲートウェイを選択し、ゲートウェイの作成を選択します。
3. デプロイターゲットで、セルフホストゲートウェイを選択します。
4. セルフホストゲートウェイオプションで、MQTT 対応、V3 ゲートウェイ - 推奨を選択します。
5. ゲートウェイ設定で：
   a. ゲートウェイ名に、ゲートウェイの名前を入力します。例えば、**`SiteWise Tutorial Device Gateway`**。
   b. Greengrass デバイス OS で、デバイスに適したオプションを選択します。
6. 詳細設定：
   a. デフォルト設定を選択します。
   b. Greengrass コアデバイスの名前を入力するか、 によって生成された名前を使用します AWS IoT SiteWise。
7. [Create gateway (ゲートウェイの作成)] を選択します。

8. 確認ダイアログで、生成とダウンロードを選択して SiteWise Edge ゲートウェイのインストーラを生成します。詳細については、「セルフホスト SiteWise Edge ゲートウェイを作成する」を参照してください。

> **Warning**
> インストーラファイルを安全な場所に保存します。このファイルは再生成できず、後のステップでゲートウェイのセットアップを完了するために必要です。

AWS CLI

AWS CLI を使用してセルフホスト型ゲートウェイを作成します。ゲートウェイの名前を指定し、プラットフォームとゲートウェイのバージョンを指定する必要があります。詳細については、AWS IoT SiteWise API リファレンスのCreateGateway」を参照してください。

この例を使用するには、ユーザー入力プレースホルダーを独自の情報に置き換えます。

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

- `gateway-name` – ゲートウェイの一意の名前。例: `SiteWise Tutorial Device Gateway`。
- `gateway-platform` – と入力します`greengrassV2`。詳細については、 AWS IoT SiteWise API リファレンスのCreateGateway」を参照してください。
  - `coreDeviceThingName` – AWS IoT Greengrass V2 コアデバイスの AWS IoT モノの名前。例えば、`SiteWiseTutorialDevice1`。
  - `coreDeviceOperatingSystem` – のコアデバイスのオペレーティングシステム AWS IoT Greengrass V2。Gateway-version 3 にはオペレーティングシステムの指定が必要です。オプションには`LINUX_AARCH64`、、`LINUX_AMD64`、および が含まれます`WINDOWS_AMD64`。
- `gateway-version` – ゲートウェイのバージョン。
  - ゲートウェイバージョン3に を使用して、MQTT 対応の V3 ゲートウェイを作成します。
- `cli-input-json` – リクエストパラメータを含む JSON ファイル。

次の AWS CLI コマンドを使用して、ゲートウェイが正常に作成されたことを確認します。

```
aws iotsitewise describe-gateway --gateway-id your-gateway-id
```

ステップ 4: SiteWise Edge ゲートウェイソフトウェアをインストールする

ゲートウェイソフトウェアをインストールするには、前のステップでダウンロードしたインストーラパッケージを使用します。インストールプロセスでは、必要なコンポーネントを設定し、Greengrass コアサービスを開始し、デバイスを に登録します AWS IoT Greengrass。インストールが完了したら、ゲートウェイが Edge ゲートウェイのAWS IoT SiteWise コンソールに表示され、Greengrass サービスがデバイスで正しく実行されていることを確認します。

詳細な手順については、「ローカルデバイスに AWS IoT SiteWise Edge ゲートウェイソフトウェアをインストールする」を参照してください。

ステップ 5: 外部アプリケーションに接続するように EMQX ブローカーを設定する

> **Note**
>
> 続行する前に、SiteWise Edge MQTT 対応 V3 ゲートウェイをデプロイしておく必要があります。ゲートウェイは、EMQX ブローカーの設定に必要なインフラストラクチャとセキュリティ設定を提供します。ブローカー設定は、アクティブなゲートウェイのデプロイなしで失敗します。

IoT デバイスと外部アプリケーション間の安全な通信を可能にするように EMQX ブローカーを設定します。EMQX ブローカーは、IoT デバイス、ゲートウェイ、アプリケーション間でデータをルーティングする中央メッセージングハブとして機能します。EMQX ブローカーは、ゲートウェイとエッジの接続されたアプリケーションで信頼性の高いメッセージ配信を実現します。詳細については、「外部アプリケーションを EMQX ブローカーに接続する」を参照してください。

EMQX ブローカーを設定するには

1. EMQX ブローカーをセットアップします。詳細な設定手順については、「認証用の EMQX デプロイ設定を更新する」のステップ 1 ~ 14 に従ってください。
2. 風力発電所のモニタリング用に MQTT トピックを設定します。MQTT 要件の詳細については、「」を参照してくださいMQTT トピックの要件。

   a. CPU 使用率: `SiteWiseTutorialDevice/cpu`
   b. メモリ使用量: `SiteWiseTutorialDevice/memory`
   c. タイムスタンプ: `SiteWiseTutorialDevice/timestamp`
3. 設定を確認し、デプロイを完了します。
   a. 確認 を選択して設定を保存します。
   b. レビューステップに到達するまで次へを選択します。
   c. [Review] ページで、[デプロイ] を選択します。
   d. デプロイが正常に完了するのを待ってから続行します。
4. EMQX ブローカーに送信するペイロード形式を使用してメッセージを準備します。ペイロードの構造化の詳細については、「認証用の EMQX デプロイ設定の更新」を参照してください。
5. 次のセキュリティ対策を実装します。
   a. Transport Layer Security (TLS) 暗号化 (ポート 8833) を使用して、転送中のデータを保護します。詳細については、「AWS IoT SiteWise Edge の EMQX ブローカーへの安全な接続のために TLS を設定する」を参照してください。
   b. ユーザー名とパスワード認証を設定して、デバイス ID を検証します。このセキュリティ対策は、データを保護するのに役立ち、承認されたデバイスのみがシステムに接続できるようにします。詳細については、「ユーザー名とパスワード認証を有効にする」を参照してください。

EMQX では、ユーザー名、IP アドレス、クライアント ID などの識別子に基づいて認可ルールを作成できます。これは、データへのアクセスを制御するのに役立ちます。詳細については、「EMQX で AWS IoT SiteWise Edge の認可ルールを設定する」を参照してください。

デプロイが成功すると、EMQX ブローカーが設定され、外部アプリケーションに安全に接続する準備が整います。

> **Note**
>
> ペイロード形式は、 AWS IoT SiteWise Edge がデータを適切に処理して取り込むための特定の構造に従う必要があります。必要な構造の詳細については、「」を参照してくださいJSON ペイロード構造。

例: CPU、メモリ、タイムスタンプ JSON ペイロードを追加する

CPU JSON ペイロード

```
{
  "propertyAlias": "SiteWiseTutorialDevice/cpu",
  "propertyValues": [
    {
      "quality": "GOOD",
      "timestamp": {
        "offsetInNanos": 0,
        "timeInSeconds": 1753206441
      },
      "value": {
        "integerValue": 45.2
      }
    }
  ]
}
```

メモリ JSON ペイロード

```
{
  "propertyAlias": "SiteWiseTutorialDevice/memory",
  "propertyValues": [
    {
      "quality": "GOOD",
      "timestamp": {
        "offsetInNanos": 0,
        "timeInSeconds": 1753206441
      },
      "value": {
        "integerValue": 67.8
      }
    }
  ]
}
```

タイムスタンプ JSON ペイロード

```
{
  "propertyAlias": "SiteWiseTutorialDevice/timestamp",
  "propertyValues": [
```

```
    {
      "quality": "GOOD",
      "timestamp": {
        "offsetInNanos": 0,
        "timeInSeconds": 1753206441
      },
      "value": {
        "integerValue": 23.5
      }
    }
  ]
}
```

> **Note**
>
> 各 JSON ペイロードは、個別のメッセージとして個別に発行する必要があります。複数のプロパティ値を 1 つのメッセージに結合しないでください。各 CPU、メモリ、タイムスタンプペイロードを独自の MQTT パブリケーションとして送信します。

ペイロードは、IoT デバイスが EMQX ブローカーを介して SiteWise Edge にデバイスデータを送信するために使用する必要のある JSON 構造を定義します。この形式により、 AWS IoT SiteWise はデバイスを識別し、センサーの読み取りを処理できます。これらの設定とペイロード構造を実装すると、風力発電所モニタリングシステムがデータを収集して処理する準備が整います。

ステップ 6: Mosquitto でデータを発行する

MQTT 対応 V3 ゲートウェイを作成したら、テストデータを SiteWise Edge に送信するように Eclipse Mosquitto を設定します。Mosquitto は、デバイス間の軽量メッセージングに MQTT プロトコルを使用するオープンソースの MQTT メッセージブローカーです。Mosquitto クライアントを使用すると、風力発電所センサーからのデータをシミュレートして、MQTT トピックにメッセージを発行できます。Mosquitto を使用して、サードパーティーのサービスや追加の機器を必要とすることなく、デバイスデータをシミュレートします。詳細については、公式の Eclipse Mosquitto ウェブサイトのドキュメントを参照してください。このチュートリアルでは、Ingest data チュートリアルのローカルデータと架空のデータがデモンストレーション目的で使用されています。

Mosquitto CLI クライアントを使用して SiteWise Edge EMQX ブローカーをテストする

1. ローカルデバイスに Mosquitto をインストールします。詳細な手順については、公式の Eclipse Mosquitto ウェブサイトの「Download Mosquitto」を参照してください。

2. 産業データを転送するために外部アプリケーションを接続する方法の詳細については、「」を参照してください外部アプリケーションを EMQX ブローカーに接続する。

> **Important**
>
> ここで設定する MQTT 接続設定が、Mosquitto publish コマンドで使用されている設定と一致していることを確認します。ホストは、SiteWise Edge ゲートウェイの IP アドレスまたはホスト名である必要があります。ポートは通常 1883 (SSL/TLS を使用している場合は 8883) です。

Mosquitto を使用してテストデータを公開します。コマンドラインを開き、次のコマンドを実行します。

例: CPU プロパティ

```
mosquitto_pub -h localhost -p 1883 -t "SiteWiseTutorialDevice/cpu" -m '{
  "propertyAlias": "SiteWiseTutorialDevice/cpu",
  "propertyValues": [
    {
      "quality": "GOOD",
      "timestamp": {
        "timeInSeconds": 1753206441,
        "offsetInNanos": 0
      },
      "value": {
        "integerValue": 45.2
      }
    }
  ]
}'
```

例: メモリプロパティ

```
mosquitto_pub -h localhost -p 1883 -t "SiteWiseTutorialDevice/memory" -m '{
  "propertyAlias": "SiteWiseTutorialDevice/memory",
  "propertyValues": [
    {
      "quality": "GOOD",
      "timestamp": {
        "timeInSeconds": 1753206441,
```

```
        "offsetInNanos": 0
      },
      "value": {
        "integerValue": 72.1
      }
    }
  ]
}'
```

例: Timestamp プロパティ

```
mosquitto_pub -h localhost -p 1883 -t "SiteWiseTutorialDevice/timestamp" -m '{
  "propertyAlias": "SiteWiseTutorialDevice/timestamp",
  "propertyValues": [
    {
      "quality": "GOOD",
      "timestamp": {
        "timeInSeconds": 1753206441,
        "offsetInNanos": 0
      },
      "value": {
        "integerValue": 1683000000
      }
    }
  ]
}'
```

> **Note**
>
> EMQX ブローカーアドレス`localhost`としての の使用は、デモンストレーションのみを目的としています。本番環境または外部デバイスから接続する場合は、特定のデプロイ設定に適切な EMQX ブローカーアドレスを使用する必要があります。接続手順の詳細については、「」を参照してください[Edge の EMQX ブローカーにアプリケーションを接続する AWS IoT SiteWise](#)。

ステップ 7: 送信先を指定する

このステップでは、送信先を指定して、ソースデータの転送先を決定します。Amazon S3 バッファリング AWS IoT SiteWise を送信先として使用します。このオプションは、IoT データを保存および処理するためのスケーラブルな方法を提供します。

Console

送信先を追加するには

1. AWS IoT SiteWise コンソールに移動し、 Edge ゲートウェイを選択します。
2. SiteWise チュートリアルデバイスゲートウェイで、送信先の追加を選択します。
3. 送信先の詳細で、AWS IoT SiteWise Amazon S3 を使用してバッファリングされた を選択します。送信先タイプの詳細については、AWS IoT SiteWise 「ゲートウェイの送信先」を参照してください。
4. 送信先名に、送信先の名前を入力します。例: `SiteWise Tutorial S3 Destination`。
5. S3 アップロード設定で、S3 バケットの場所を入力します。例えば、`s3://sitewise-tutorial-mqtt-data-[your-account-id]`。Amazon S3 の詳細については、「Amazon Simple Storage Service ユーザーガイド」のAmazon S3バケットの作成、設定、および操作」を参照してください。
6. データのアップロード頻度に、1 分から 30 日までの値を入力します。例えば、`1 minute`。
7. データストレージ設定の下：
   - 「ストレージにデータをコピーする」の選択を解除します。この設定は本番環境では推奨されますが、このチュートリアルでは必要ありません。このオプションの選択を解除すると、S3 からデータを削除するオプションが自動的に選択解除されます。
8. 送信先の追加を選択します。

> **Note**
>
> このチュートリアルでは、テストに 1 分間隔を使用します。チュートリアルを完了したら、本稼働ニーズに合わせてこの間隔を調整したり、追加料金が発生しないように削除したりできます。

AWS CLI

例: Amazon S3 を使用してバッファされた新しい AWS IoT SiteWise 送信先を作成する

AWS CLI コマンドリファレンスの update-gateway-capability-configuration を使用して、パブリッシャーを設定します。`capabilityNamespace` パラメータを `iotsitewise:publisher:3` に設定します。

```
{
    "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": "1m",
            "localSizeLimitGB": 10
          },
          "siteWiseImportPolicy": {
            "enableSiteWiseStorageImport": true,
            "enableDeleteAfterImport": true,
            "bulkImportJobRoleArn": "arn:aws:iam::123456789012:role/your-role-name"
          }
        },
        "filters": [
          {
            "type": "PATH",
            "config": {
              "paths": [
                "#"
              ]
            }
          }
        ]
      }
    ]
  }
```

送信先の詳細については、「」を参照してください[Amazon S3 を使用して AWS IoT SiteWise バッファされた送信先を追加する](#)。

ステップ 8: パスフィルターを指定する

このステップでは、パスフィルターを設定して、風力発電所のデバイスデータに対してモニタリングする MQTT トピックを指定します。

パスフィルターは、2 つの特殊文字をサポートする MQTT トピックのワイルドカード仕様に従います。

- + – この記号は、単一レベルのワイルドカードを表し、単一レベルの任意の文字列に一致します。
- # – この記号は、トピック階層内の任意のレベルと一致する複数レベルのワイルドカードを表します。

> **Note**
>
> その他のパスフィルターの詳細については、「」を参照してください[パスフィルター名の特殊文字](#)。

Console

パスフィルターを設定するには

パスフィルターの下：

1. AWS IoT SiteWise コンソールに移動し、 Edge ゲートウェイを選択します。
2. SiteWise チュートリアルデバイスゲートウェイで、送信先の追加を選択します。
3. パスフィルターの追加 を選択して、次のパスフィルターを手動で入力します。
   - `SiteWiseTutorialDevice/#`
   - `windfarm/+/turbine/+/performance/#`
   - `cpu/+/idle-time`
   - `cpu/+/interruption-count/+`
   - `+/memory/consumption`
   - `timestamp/+/measurement`
   - `device/+/status/+`
   - `system/+/performance-log`
4. 送信先の追加を選択します。

パスフィルターのベストプラクティスの詳細については、「」を参照してください[パスフィルターのベストプラクティス](#)。

AWS CLI

パスフィルターを設定するには、次の AWS CLI コマンドを使用します。

例 1: ワイルドカードを使用したデバイスデータ

```
{
 "destinations": [
   {
     "name": "All Device Data Destination"
   }
 ],
 "filters": [
   {
     "type": "PATH",
     "config": {
       "paths": [
         "SiteWiseTutorialDevice/#",
         "windfarm/+/turbine/+/performance/#"
       ]
     }
   }
 ]
}
```

このパスフィルター設定では、マルチレベルワイルドカード (#) を使用して、SiteWiseTutorialDevice のすべてのデータと、風力発電所のタービンのすべてのパフォーマンスデータをキャプチャします。

例 2: CPU とメモリのパフォーマンス

```
{
 "destinations": [
   {
     "name": "Performance Metrics Destination"
   }
 ],
 "filters": [
   {
     "type": "PATH",
```

```
      "config": {
        "paths": [
          "cpu/+/idle-time",
          "+/memory/consumption",
          "cpu/+/interruption-count/+"
        ]
      }
    }
  ]
}
```

この例では、デバイス全体のさまざまな CPU メトリクス (アイドル時間と中断数) とメモリ消費データをキャプチャします。

例 3: デバイス診断

```
{
 "destinations": [
   {
     "name": "Device Diagnostics Destination"
   }
 ],
 "filters": [
   {
     "type": "PATH",
     "config": {
       "paths": [
        "device/+/status/+",
        "system/+/performance-log"
       ]
     }
   }
 ]
}
```

この設定では、+ワイルドカードを使用して、複数のデバイスからの診断データ、特にシステムパフォーマンスログとデバイスステータスの更新をキャプチャします。

これらの 3 つのパスフィルターは、Mosquitto でテストデータを発行するために使用する MQTT トピックと一致します。フィルターにより、SiteWise Edge ゲートウェイが関連する MQTT メッセージをキャプチャして処理できるようになります。パスフィルターを追加する方法の詳細については、「」を参照してくださいAWS IoT SiteWise Edge 送信先にパスフィルターを追加する。

ステップ 9: AWS IoT リソースを設定する

このステップでは、シミュレートされたサードパーティーデバイスを表すために必要な AWS IoT SiteWise アセットモデルとアセットを作成し、エッジゲートウェイを介したデータ取り込みを有効にします。

このステップを開始する前に、Ingest data チュートリアルのステップ 3～8 を完了しておく必要があります。これらのステップは、MQTT 対応 V3 ゲートウェイを介してサードパーティーデータを統合するための基本コンポーネントを確立します。また、センサーデータがどのように流れるかを定義するルールを設定し AWS IoT SiteWise、産業用風力発電所データをシミュレートするデバイスクライアントスクリプトを実行します。

AWS IoT リソース設定を検証するには

1. 次の AWS CLI コマンドを使用して、SiteWise チュートリアルデバイスモデルと SiteWise チュートリアルデバイスフリートモデルが作成され、正しく設定されていることを確認します。

   ```
   aws iotsitewise describe-asset-model --asset-model-id your-device-model-id
   ```

   次の AWS CLI コマンドを使用して、アセットモデルの ID を取得します。

   ```
   aws iotsitewise list-asset-models
   ```

2. 次の AWS CLI コマンドを使用して、SiteWise チュートリアルデバイス 1 アセットと SiteWise チュートリアルデバイスフリート 1 アセットが作成され、正しく設定されていることを確認します。

   ```
   aws iotsitewise describe-asset --asset-id your-asset-id
   ```

   次の AWS CLI コマンドを使用して、アセットの ID を取得します。

   ```
   aws iotsitewise list-assets
   ```

ステップ 10: データを視覚化する

オープンソースバージョンの Grafana を設定して、風力発電所のデバイスデータを視覚化します。Grafana は、リアルタイムの運用データを表示する視覚化プラットフォームです。これらのダッシュボードは、運用効率を追跡し、インフラストラクチャ全体のメンテナンスニーズを特定するの

に役立ちます。統合の詳細については、「」を参照してください[Grafana AWS IoT SiteWise との統合](#)。

Grafana をセットアップするには

1. Grafana の最新バージョンをダウンロードしてインストールする手順については、公式 Grafana ウェブサイトの「Grafana のインストール」を参照してください。
2. オペレーティングシステム固有の詳細な設定手順については、公式 Grafana ウェブサイトの「Configure Grafana」を参照してください。
3. AWS IoT SiteWise データソースを設定します。これにより、Grafana サーバーで AWS IoT SiteWise プラグインを設定できます。プラグインの使用方法の詳細については、「Amazon Managed Grafana ユーザーガイド」のAWS IoT SiteWise「データソースに接続する」を参照してください。

> **Important**
>
> AWS IoT SiteWise データソースとの互換性のために、Grafana の最新バージョンがあることを確認します。

これらのステップを完了したら、Grafana ダッシュボードを構築およびカスタマイズして、風力発電所の運用メトリクスを表示できます。これにより、エッジでの風力発電所のパフォーマンスをリアルタイムで追跡および分析できます。

> **Note**
>
> このチュートリアルでは Grafana のオープンソースバージョンを使用していますが、 は本番環境用の Amazon Managed Grafana AWS も提供しています。Amazon Managed Grafana は、独自の Grafana サーバーをセットアップ、設定、保守する必要がなくなるフルマネージドサービスです。ソリューションをスケールする準備ができたら、Amazon Managed Grafana へのアップグレードを検討してください。Amazon Managed Grafana の詳細と、オープンソースツールを使用してデータを処理および視覚化する方法については、以下を参照してください。
>
> - Amazon Managed Grafana ユーザーガイドの「Amazon Managed Grafana とは」
> - SiteWise Edge とオープンソースツールを使用してデータを処理および視覚化する

これでチュートリアルは完了です。この手順では、MQTT 対応 V3 ゲートウェイを使用してサードパーティーのデバイスデータを統合するように AWS IoT SiteWise Edge を設定しました。この設定により、エッジで産業機器データを収集、処理、視覚化できるため、レイテンシーと運用コストを削減できます。風力発電所のデモを使用することで、MQTT 対応 V3 ゲートウェイを介して CPU やメモリの使用状況データなどの運用メトリクスを収集して処理しました。

IoT ソリューションを強化するには、 を活用して異常検出などの高度な機能を調べるかLookout for Equipment で異常を検出する、Amazon QuickSight ユーザーガイドの Amazon QuickSight などの他の AWS サービスと統合して高度な分析を行うことを検討してください。

ステップ 11: チュートリアルの後にリソースをクリーンアップする

AWS IoT SiteWise Edge へのデータの統合に関するこのチュートリアルを完了したら、追加料金が発生しないようにリソースをクリーンアップします。

で階層アセットを削除するには AWS IoT SiteWise

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションペインで [アセット] を選択します。
3. でアセットを削除するときは AWS IoT SiteWise、まずアセットの関連付けを解除する必要があります。

   デバイスフリートアセットからデバイスアセットの関連付けを解除するには、次の手順を実行します。

   a. デバイスフリートアセット () を選択しますSiteWise Tutorial Device Fleet 1。
   b. [編集] を選択します。
   c. [このアセットに関連付けられているアセット] で、このデバイスフリートアセットに関連付けられている各デバイスアセットに対して [関連付け解除] を選択します。
   d. [保存] を選択します。

   > **Note**
   > デバイスアセットは現在階層として整理されなくなりました。

4. デバイスアセット (SiteWise Tutorial Device 1) を選択します。
5. [削除] を選択します。
6. 確認ダイアログで、「」と入力し**Delete**、「削除」を選択します。

7. デバイスアセットとデバイスフリートアセット (SiteWise Tutorial Device Fleet 1) ごとに、ステップ 4 ~ 6 を繰り返します。

で階層アセットモデルを削除するには AWS IoT SiteWise

1. AWS IoT SiteWise コンソールに移動します。
2. デバイスおよびデバイスフリートアセットを削除します。
3. 左のナビゲーションペインで [モデル] を選択します。
4. デバイスフリートアセット (SiteWise Tutorial Device Fleet Model 1) を選択します。モデルから作成されたアセットがある場合は、そのモデルを削除できません。

   階層アセットモデルを削除する場合は、最初に親アセットモデルを削除する必要があります。
5. [削除] を選択します。
6. 確認ダイアログで、「」と入力し**Delete**、「削除」を選択します。
7. デバイスのアセットモデル (SiteWise Tutorial Device Model) について、ステップ 4 ~ 6 を繰り返します。

でルールを無効化または削除するには AWS IoT Core

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで、メッセージルーティングを選択し、ルールを選択します。
3. ルールを選択し、[削除] を選択します。
4. 確認ダイアログボックスでルール名を入力し、[削除] を選択します。

Amazon S3 バケットを削除するには

1. Amazon S3 コンソールに移動します。
2. 左側のナビゲーションペインで、汎用バケットを選択します。
3. バケットリストで、作成したバケットの横にあるオプションボタンを選択し、ページの上部にある空の を選択します。
4. 確認ダイアログで、削除を確認し、空を選択します。
5. バケットが空になったら、削除を選択してバケットを削除します。
6. 確認ダイアログで、削除を確認するバケットの名前を入力します。

7. [バケットを削除] を選択します。

SiteWise Edge ゲートウェイを削除するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションペインで、エッジゲートウェイを選択します。
3. Gateways で、このチュートリアル用に作成したゲートウェイを選択します。例えば、`SiteWise Tutorial Device Gateway`。
4. [削除] を選択します。
5. ゲートウェイを削除することを確認するには、確認ダイアログ**Delete**に「」と入力し、表示されるウィンドウで「削除」を選択します。

IoT モノを削除するには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで、管理を選択し、モノを選択します。
3. このチュートリアル用に作成した IoT モノを選択します。例えば、`SiteWiseTutorialDevice1`。
4. [削除] を選択します。
5. 確認ダイアログで、モノの名前を入力し、削除を選択します。

AWS IoT Greengrass Core をアンインストールするには

ローカルデバイスから AWS IoT Greengrass Core ソフトウェアをアンインストールします。詳細な手順については、「 AWS IoT Greengrass デベロッパーガイド、バージョン 2」の AWS IoT Greengrass 「 Core ソフトウェアをアンインストールする」を参照してください。

> **Important**
>
> Greengrass をアンインストールすると、すべてのローカル設定とデータが削除されます。続行する前に、重要な情報をバックアップしていることを確認してください。

(オプション) サードパーティーリソースを削除するには

このチュートリアルを完了したら、作成した外部リソースをシャットダウンすることを検討してください。これにより、サードパーティープロバイダーからの料金の発生を防ぐことができます。

追加リソース

詳細については、次のリソースを参照してください。

- 他の AWS サービスとやり取りする
- AWS IoT SiteWise Edge ゲートウェイを使用する
- SiteWise Edge ゲートウェイのトラブルシューティング
- のセキュリティのベストプラクティス AWS IoT SiteWise
- AWS IoT の料金
- にデータを取り込む AWS IoT SiteWise
- でタグを使用する AWS IoT SiteWise

SiteWise Monitor で風力発電所データを視覚化して共有する

このチュートリアルでは、 AWS IoT SiteWise Monitor を使用して、ポータルと呼ばれるマネージドウェブアプリケーションを通じて産業データを視覚化および共有する方法について説明します。各ポータルにはプロジェクトが含まれており、各プロジェクト内でどのデータにアクセスできるかを柔軟に選択できます。その後、各ポータルにアクセスできる組織内のユーザーを指定します。ユーザーは AWS IAM Identity Center アカウントを使用してポータルにサインインするので、既存の ID ストアまたは AWSによって管理されるストアを使用できます。

ユーザーおよび十分なアクセス許可を持つユーザーは、各プロジェクトでダッシュボードを作成し、産業データを有意義な方法で視覚化することができます。その後、ユーザーはこれらのダッシュボードを表示して、データに関するインサイトをすばやく取得し、オペレーションを監視できます。各プロジェクトに対する管理権限または読み取り専用のアクセス許可は、社内の各ユーザーに対して設定できます。詳細については、「でデータをモニタリングする AWS IoT SiteWise Monitor」を参照してください。

チュートリアル全体でデモを強化し AWS IoT SiteWise 、風力発電所のサンプルデータセットを提供します。SiteWise Monitor でポータルを設定し、風力発電所のデータを視覚化するためのプロジェクトとダッシュボードを作成します。このチュートリアルでは、追加のユーザーの作成と、プロジェクトとその関連するダッシュボードを所有または表示するアクセス許可の割り当てについても説明します。

> **Note**
>
> SiteWise Monitor を利用する場合、ポータルにサインインしたユーザーごとに課金されます (1ヶ月あたり)。このチュートリアルでは、3 人のユーザーを作成しますが、サインインする必要があるのは 1 人のユーザーだけです。このチュートリアルを完了すると、1 人のユーザーに対して料金が発生します。詳細については、「AWS IoT SiteWise 料金」を参照してください。

トピック

- 前提条件
- ステップ 1: SiteWise Monitor でポータルを作成する
- ステップ 2: ポータルにサインインする
- ステップ 3: 風力発電所プロジェクトを作成する
- ステップ 4: ダッシュボードを作成して、風力発電所データを視覚化する
- ステップ 5: ポータルを調べる
- ステップ 6: チュートリアル後にリソースをクリーンアップする

前提条件

このチュートリアルを完了するには、以下が必要です。

- AWS アカウント。アカウントをお持ちでない場合は、「AWS アカウントを設定する」を参照してください。
- Unix 、Windows、macOS、Linuxまたは を実行して にアクセスする開発用コンピュータ AWS Management Console。詳細については、「AWS Management Consoleとは?」を参照してください。
- 管理者権限を持つ AWS Identity and Access Management (IAM) ユーザー。
- 実行中の AWS IoT SiteWise 風力発電所のデモ。デモを設定すると、 でモデルとアセットを定義 AWS IoT SiteWise し、風力発電所を表すためにデータをストリーミングします。詳細については、「AWS IoT SiteWise デモを使用する」を参照してください。
- アカウントで IAM Identity Center を有効にした場合は、 AWS Organizations 管理アカウントにサインインします。詳しくは、[AWS Organizations terminology and concepts] (用語と概念) をご覧ください。IAM Identity Center を有効にしなかった場合は、このチュートリアルで有効にして、自分のアカウントを管理アカウントに設定します。

AWS Organizations 管理アカウントにサインインできない場合は、組織に IAM Identity Center ユーザーがある限り、チュートリアルを部分的に完了できます。この場合、ポータルやダッシュボードは作成できますが、プロジェクトに割り当てる IAM Identity Center ユーザーを新規作成することはできません。

ステップ 1: SiteWise Monitor でポータルを作成する

この手順では、 AWS IoT SiteWise Monitorでポータルを作成します。各ポータルは、組織の管理者とユーザーが AWS IAM Identity Center アカウントを使用してサインインできるマネージドウェブアプリケーションです。IAM Identity Center では、会社の既存の ID ストアを使用することも、 AWSによって管理される ID ストアを作成することもできます。会社の従業員は、個別の AWS アカウントを作成せずにサインインできます。

ポータルを作成するには

1. AWS IoT SiteWise コンソールにサインインします。
2. AWS IoT SiteWise がサポートされているAWS IoT SiteWise エンドポイントとクォータを確認し、必要に応じてリージョンを切り替えます。 AWS IoT SiteWise デモは同じリージョンで実行する必要があります。
3. 左のナビゲーションペインで、[ポータル] を選択します。
4. [ポータルの作成] を選択します。
5. すでに IAM Identity Center を有効にしている場合は、ステップ 6 に進みます。それ以外の場合は、以下のステップを実行して IAM Identity Center を有効にします。

   a. Enable AWS IAM Identity Center (SSO) ページで、E メールアドレス、名、姓を入力して、ポータル管理者になるための IAM Identity Center ユーザーを作成します。新しい IAM Identity Center ユーザーのパスワードを設定するための E メールを受信できるように、アクセスできる E メールアドレスを使用します。

      ポータルでは、ポータル管理者がプロジェクトを作成し、ユーザーをプロジェクトに割り当てます。後でさらにユーザーを作成できます。

b. [ユーザーの作成] を選択します。

6. [ポータルの設定] ページで、次のステップを実行します。

   a. **`WindFarmPortal`** などのポータルの名前を入力します。

   b. (オプション) ポータルの説明を入力します。複数のポータルがある場合は、わかりやすい説明を使用すると、各ポータルに含まれる内容を追跡できます。

   c. (オプション) ポータルに表示するイメージをアップロードします。

   d. ポータルに問題があるときにポータルユーザーが連絡できる E メールアドレスを入力し、それを解決するために会社の AWS 管理者からのサポートが必要です。

   e. [ポータルの作成] を選択します。

7. [管理者を招待する] ページでは、IAM Identity Center ユーザーを管理者としてポータルに割り当てることができます。ポータル管理者は、ポータル内のアクセス許可とプロジェクトを管理します。このページで、次のことを行ってください。

   a. ポータル管理者となるユーザーを選択します。のチュートリアルの前半で IAM Identity Center を有効にした場合、作成したユーザーを選択します。

b. (オプション) [選択したユーザーに招待を送信] を選択します。メールソフトを開くと、メッセージの本文に招待状が表示されます。E メールは、ポータル管理者に送信する前にカスタマイズできます。後でポータル管理者に E メールを送信することもできます。SiteWise Monitor を初めて試す方で、ポータル管理者になる方は、ご自身でメールする必要はありません。

c. [Next (次へ)] を選択します。

8. [ユーザーの割り当て] ページでは、ポータルに IAM Identity Center ユーザーを割り当てることができます。ポータル管理者は、これらのユーザーを後でプロジェクトの所有者やビューワーとして割り当てることができます。プロジェクト所有者は、プロジェクト内にダッシュボードを作成することができます。プロジェクトのビューワーは、割り当てられたプロジェクトに対して読み取り専用のアクセス権を持ちます。このページでは、ポータルに追加する IAM Identity Center ユーザーを作成することができます。

> **Note**
>
> AWS Organizations 管理アカウントにサインインしていない場合、IAM Identity Center ユーザーを作成することはできません。[ユーザーの割り当て] を選択して、ポータルユーザーなしでポータルを作成し、このステップをスキップします。

このページで、次のことを行ってください。

a. 次のステップを 2 回行い、2 人の IAM Identity Center ユーザーを作成します。

i. [ユーザーの作成] を選択すると、新規ユーザーの詳細を入力するダイアログボックスが表示されます。

ii. 登録するユーザーの E メールアドレスと名、姓を入力します。IAM Identity Center からユーザーに、パスワードを設定するための E メールが送信されます。これらのユーザーとしてポータルにサインインする場合、アクセス可能なメールアドレスを選択します。各メールアドレスは異なるメールアドレスでなければなりません。ユーザーは、メールアドレスをユーザー名としてポータルにサインインします。

iii. [ユーザーの作成] を選択します。

b. 前のステップで作成した 2 人の IAM Identity Center ユーザーを選択します。

c. [ユーザーの割り当て] を選択して、これらのユーザーをポータルに追加します。

ポータルページが開き、新しいポータルが一覧表示されます。

ステップ 2: ポータルにサインインする

この手順では、ポータルに追加した AWS IAM Identity Center ユーザーを使用して新しいポータルにサインインします。

ポータルにサインインするには

1. [ポータル] ページで、新しいポータルの リンク を選択し、新しいタブでポータルを開きます。

2. チュートリアルの前半で最初の IAM Identity Center ユーザーを作成した場合は、次のステップに従ってユーザーのパスワードを作成します。

   a. メールで件名 (Invitation to join AWS IAM Identity Center) を確認します。

   b. 招待メールを開き、[Accept invitation] を選択します。

   c. 新しいウィンドウで、IAM Identity Center ユーザーのパスワードを設定します。

   先に作成した 2 人目、3 人目の IAM Identity Center ユーザーとして後でポータルにサインインしたい場合は、次のステップでパスワードを設定することもできます。

   > **Note**
   >
   > メールが届かない場合は、IAM Identity Center コンソールでユーザーのパスワードを生成できます。詳細については、「AWS IAM Identity Center ユーザーガイド」の「エンドユーザーの IAM Identiy Center ユーザーパスワードのリセット」を参照してください。

3. IAM Identity Center の Usernameと Passwordを入力します。このチュートリアルの前半で IAM Identity Center ユーザーを作成した場合、Usernameは作成したポータル管理者ユーザーの E メールアドレスです。

   ポータル管理者を含むすべてのポータルユーザーは、自分の IAM Identity Center ユーザー認証情報を使用してサインインする必要があります。通常、この認証情報は、 AWS Management Consoleへのサインインに使用する認証情報とは異なります。

4. Sign in を選択してください。

   ポータルが開きます。

ステップ 3: 風力発電所プロジェクトを作成する

この手順では、ポータルでプロジェクトを作成します。プロジェクトは、一連のアクセス許可、アセット、およびダッシュボードを定義するリソースであり、そのプロジェクト内のアセットデータを視覚化するように設定できます。プロジェクトでは、オペレーションのどのサブセットにアクセスできるか、およびそれらのサブセットのデータを視覚化する方法を定義します。ポータルユーザーは、各プロジェクトの所有者やビューワーとして割り当てることができます。プロジェクト所有者は、データを視覚化するダッシュボードを作成し、他のユーザーとプロジェクトを共有することができます。プロジェクトビューワーはダッシュボードを表示できますが、編集はできません。SiteWise Monitor のロールの詳細については、SiteWise Monitor のロール を参照してください。

風力発電施設プロジェクトを作成するには

1. ポータルの左側のナビゲーションペインで、[アセット] タブを選択します。[アセット] ページでは、ポータルで利用可能なすべてのアセットを探索し、プロジェクトにアセットを追加することができます。

2. アセットブラウザで、[Demo Wind Farm Asset] を選択します。アセットを選択すると、そのアセットのライブデータと履歴データを調べることができます。Shift を押して、複数のアセットを選択し、そのデータを並べて比較することもできます。
3. 左上の [プロジェクトにアセットを追加] を選択します。プロジェクトには、ポータルユーザーがデータを探索するために表示できるダッシュボードが含まれています。各プロジェクトは、のアセットのサブセットにアクセスできます AWS IoT SiteWise。プロジェクトにアセットを追加すると、そのプロジェクトへのアクセス権を持つすべてのユーザーは、そのアセットとその子のデータにもアクセスできます。

4. [プロジェクトにアセットを追加する] ダイアログボックスで、[新しいプロジェクトを作成する] を選択し、[次へ] を選択します。

5. [プロジェクトの新規作成] ダイアログボックスで、[プロジェクト名] と[プロジェクトの説明] を入力し、[アセットをプロジェクトに追加する] を選択します。

新しいプロジェクトのページが開きます。

6. プロジェクトのページでは、ポータルユーザーをこのプロジェクトの所有者やビューワーとして追加することができます。

> **Note**
>
> AWS Organizations 管理アカウントにサインインしていない場合は、このプロジェクトに割り当てるポータルユーザーがいない可能性があるため、このステップをスキップできます。

このページで、次のことを行ってください。

a. [プロジェクトの所有者] で[所有者の追加] または[ユーザーの編集] を選択します。

b. プロジェクト所有者として追加するユーザー (Mary Major など) を選択し、[>>] アイコンを選択します。

c. [Save] を選択します。

IAM Identity Center ユーザーの Mary Majorは、このポータルにサインインしてこのプロジェクトのダッシュボードを編集したり、このポータルの他のユーザーとこのプロジェクトを共有したりできます。

d. [プロジェクトのビューワー] で[ビューワーの追加] または[ユーザーの編集] を選択します。

e. プロジェクトビューワーとして追加するユーザー (Mateo Jackson など) を選択し、>> アイコンを選択します。

f. [Save] を選択します。

IAM Identity Center ユーザーの Mateo Jacksonは、このポータルにサインインして、風力発電施設のプロジェクトのダッシュボードを表示できますが、編集はできません。

ステップ 4: ダッシュボードを作成して、風力発電所データを視覚化する

この手順では、ダッシュボードを作成して、デモ風力発電施設のデータを視覚化します。ダッシュボードには、プロジェクトのアセットデータのカスタマイズ可能な視覚化が含まれています。各視覚化は、折れ線グラフ、棒グラフ、KPI 表示など、異なるタイプを持つことができます。データに最も適した視覚化のタイプを選択できます。プロジェクト所有者はダッシュボードを編集できますが、プロジェクトビューワーはダッシュボードを表示してインサイトを得ることしかできません。

視覚化を含むダッシュボードを作成するには

1. 新規プロジェクトのページで、[ダッシュボードの作成] を選択してダッシュボードを作成し、その編集ページを開きます。

   ダッシュボードの編集ページで、アセット階層からダッシュボードにアセットプロパティをドラッグして、視覚化を作成できます。次に、各視覚化のタイトル、凡例タイトル、タイプ、サイズ、ダッシュボード内の場所を編集できます。

2. ダッシュボードに名前を入力します。

3. Total Average Power から Demo Wind Farm Asset をダッシュボードにドラッグして視覚化する。

4. Demo Turbine Asset 1を選択してそのアセットのプロパティを表示し、Wind Speedをダッシュボードにドラッグして風速の視覚化を作成します。

5. Demo Turbine Asset 2、3、4 (この順) の各風速の新しい視覚化にWind Speedを追加します。

   Wind Speed 視覚化は、次のスクリーンショットのようになります。

6. 風力タービンの Torque (KiloNewton Meter) プロパティに対してステップ 4 と 5 を繰り返して、風力タービントルクの視覚化を作成します。
7. Torque (KiloNewton Meter) の視覚化の視覚化タイプアイコンを選択し、棒グラフのアイコンを選択します。

8. 風力タービンの Wind Direction プロパティに対してステップ 4 と 5 を繰り返して、風向の視覚化を作成します。
9. Wind Direction の視覚化の視覚化型アイコンを選択し、次に、KPI グラフアイコン (30%) を選択します。

10. (オプション) 必要に応じて、各視覚化のタイトル、凡例タイトル、型、サイズ、場所を変更します。
11. 右上の [ダッシュボードの保存] を選択してダッシュボードに保存します。

    ダッシュボードは、次のスクリーンショットのようになります。

12. (オプション) 風力タービンアセットごとに追加のダッシュボードを作成します。

    ベストプラクティスとして、プロジェクトビューワーが個々のアセットに関する問題を調査できるように、アセットごとにダッシュボードを作成することをお勧めします。各視覚化には最大 5 つのアセットしか追加できないため、多くのシナリオで階層アセットに対して複数のダッシュボードを作成する必要があります。

    デモ風力タービンのダッシュボードは、次のスクリーンショットのようになります。

13. (オプション) タイムラインを変更するか、視覚化のデータポイントを選択して、ダッシュボードのデータを調べます。詳細は、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[Viewing dashboards] (ダッシュボードを表示する) を参照してください。

ステップ 5: ポータルを調べる

この手順では、 AWS IoT SiteWise ポータル管理者よりも少ないアクセス許可を持つユーザーとして、ポータルを探索することができます。

ポータルを探索し、チュートリアルを終了するには

- (オプション) プロジェクトに他のユーザーを所有者またはビューワーとして追加した場合、これらのユーザーとしてポータルにサインインすることができます。これにより、ポータル管理者よりも少ないアクセス許可を持つユーザーとして、ポータルを探索することができます。

  > **Important**
  > ポータルにサインインしたユーザーごとに課金されます。詳細については、「AWS IoT SiteWise 料金」を参照してください。

  他のユーザーとしてポータルを探索するには、次のようにします。

  a. ポータルの左下にある [ログアウト] を選択して、ウェブアプリケーションを終了します。
  b. IAM Identity Center アプリケーションポータルの右上にある [サインアウト] を選択して、IAM Identity Center ユーザーからサインアウトします。
  c. プロジェクト所有者またはプロジェクトビューワーとして割り当てた IAM Identity Center ユーザーとしてポータルにサインインします。詳細については、「ステップ 2: ポータルにサインインする」を参照してください。

これで、このチュートリアルは終了です。SiteWise Monitor でデモの風力発電施設を探索し終えたら、次のステップに従ってリソースをクリーンアップします。

ステップ 6: チュートリアル後にリソースをクリーンアップする

チュートリアルを完了したら、リソースをクリーンアップできます。ユーザーがポータルにサインインしない場合、 AWS IoT SiteWise に対して料金は発生しませんが、ポータルと AWS IAM アイデンティティセンターディレクトリ ユーザーを削除することはできます。デモ風力発電施設アセットは、デモの作成時に選択した期間の終了時に削除されます。デモを手動で削除することもできます。詳細については、「AWS IoT SiteWise デモを削除する」を参照してください。

ポータルと IAM Identity Center ユーザーを削除するには、次の手順に従います。

ポータルを削除するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左のナビゲーションペインで、[ポータル] を選択します。

3. ポータル、[WindFarmPortal] を選択し、[削除] を選択します。

   ポータルまたはプロジェクトを削除しても、削除されたプロジェクトに関連付けられているアセットは影響を受けません。

4. [ポータルの削除] ダイアログボックスで、[管理者とユーザーの削除] を選択します。

5. **`delete`** と入力して削除を確認し、[削除] を選択します。

IAM Identity Center ユーザーを削除するには

1. IAM Identity Center コンソールに移動します。
2. 左のナビゲーションペインで、[ユーザー] を選択します。
3. 削除する各ユーザーのチェックボックスをオンにし、[ユーザーの削除] を選択します。

4. [ユーザーの削除] ダイアログボックスで、**DELETE** と入力し、[ユーザーの削除] を選択します。

プロパティ値の更新を Amazon DynamoDB に発行する

このチュートリアルでは、Amazon DynamoDB を使用してデータを保存する便利な方法を紹介します。これにより、 AWS IoT SiteWise API を繰り返しクエリすることなく、履歴アセットデータに簡単にアクセスできます。このチュートリアルを完了すると、風力発電所全体の風速や風向のライブマップなど、アセットデータを消費するカスタムソフトウェアを作成できます。カスタムソフトウェアソリューションを実装せずにデータを監視および可視化する場合は、「でデータをモニタリングする AWS IoT SiteWise Monitor」を参照してください。

このチュートリアルでは、風力発電所のサンプルデータセットを提供する AWS IoT SiteWise デモに基づいて構築します。風力発電施設のデモでは、プロパティ値の更新が AWS IoT Core ルールを介して、作成したDynamoDB テーブルにデータを送信するように設定されています。プロパティ値の更新を有効にすると、 AWS IoT SiteWise は MQTT メッセージ AWS IoT Core で にデータを送信します。次に、それらのメッセージの内容に応じて、DynamoDB アクションなどのアクションを実行する AWS IoT コアルールを定義します。詳細については、「他の AWS サービスとやり取りする」を参照してください。

トピック

- 前提条件
- ステップ 1: プロパティ値の更新を発行 AWS IoT SiteWise するように を設定する
- ステップ 2: AWS IoT Core でルールを作成する
- ステップ 3: DynamoDB テーブルを作成する
- ステップ 4: DynamoDB ルールアクションを設定する
- ステップ 5: DynamoDB のデータを調べる
- ステップ 6: チュートリアル後にリソースをクリーンアップする

前提条件

このチュートリアルを完了するには、以下が必要です。

- AWS アカウント。アカウントをお持ちでない場合は、「AWS アカウントを設定する」を参照してください。
- Windows、macOS、Linux、または Unix を実行して にアクセスする開発コンピュータ AWS Management Console。詳細については、「AWS Management Consoleとは」を参照してください。

- 管理者権限を持つ IAM ユーザー。
- 実行中の AWS IoT SiteWise 風力発電所のデモ。デモを設定すると、 でモデルとアセットを定義 AWS IoT SiteWise し、風力発電所を表すためにデータをストリーミングします。詳細については、「AWS IoT SiteWise デモを使用する」を参照してください。

ステップ 1: プロパティ値の更新を発行 AWS IoT SiteWise するように を設定する

この手順では、デモタービンアセットの [Wind Speed] プロパティでプロパティ値の通知を有効にします。プロパティ値通知を有効にすると、 は MQTT メッセージの各値の更新を AWS IoT Core に AWS IoT SiteWise 発行します。

アセットプロパティでプロパティ値の更新に関する通知を有効にするには

1. AWS IoT SiteWise コンソールにサインインします。
2. AWS IoT SiteWise がサポートされているAWS IoT SiteWise エンドポイントとクォータを確認し、必要に応じて AWS リージョンを切り替えます。 AWS IoT SiteWise デモを実行しているリージョンに切り替えます。
3. 左側のナビゲーションペインで [アセット] を選択します。

4. Demo Wind Farm Asset の横にある矢印を選択して、風力発電施設のアセットの階層を展開します。

5. Demo Turbine を選択し、[編集] を選択します。

6. Wind Speedプロパティの [通知ステータス] を [已启用] に更新します。

7. ページの下部にある [Save asset (アセットを保存)] を選択します。
8. 各 Demo Turbine Asset に対し、手順 5 ~ 7 を繰り返します。
9. デモタービン (Demo Turbine Asset 1など) を選択します。
10. [測定] を選択します。
11. [Wind Speed] プロパティの横にあるコピーアイコンを選択して、通知トピックをクリップボードにコピーします。保存した通知トピックは、このチュートリアルの後半で使用します。通知トピックは、1 つの Turbine から記録するだけで大丈夫です。

通知トピックは、次の例のようになります。

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

ステップ 2: AWS IoT Core でルールを作成する

この手順では、プロパティ値の通知メッセージを解析し、Amazon DynamoDB テーブルにデータを挿入するルールを AWS IoT Core で作成します。 AWS IoT Core ルールは MQTT メッセージを解析し、各メッセージの内容とトピックに基づいてアクションを実行します。このチュートリアルの一部として作成した DynamoDB テーブルにデータを挿入する DynamoDB アクションでルールを作成します。

DynamoDB アクションを使用したルールの作成

1. AWS IoT コンソールに移動します。[今すぐ始める] ボタンが表示された場合はそれをクリックします。
2. 左側のナビゲーションペインで [Act (アクト)] を選択し、[ルール] を選択します。

3. [ルールはまだ作成されていません] ダイアログボックスが表示された場合は、[ルールの作成] を選択します。それ以外の場合は、[作成] を選択します。
4. ルールの名前と説明を入力します。

5. このチュートリアルの前の手順で保存した通知トピックを見つけます。

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

トピックのアセット ID (assets/ の後の ID) を + に置き換えます。これにより、すべてのデモ風力タービンアセットの風速プロパティが選択されます。+ トピックフィルターは、トピック内の 1 つのレベルのすべてのノードを受け入れます。トピックは次の例のようになります。

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

6. 次のルールクエリステートメントを入力します。FROM セクションのトピックを通知トピックに置き換えます。

```
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. [1 つ以上のアクションを設定する] で、[アクションの追加] を選択します。

8. [アクションを選択] ページで、[DynamoDB テーブル (DynamoDBv2) の複数列にメッセージを分割する] を選択します。

9. ページの下部にある [アクションの設定] を選択します。
10. [配置操作] ページで [創建新資源] を選択します。

    新しいタブで DynamoDB コンソールが開きます。次の手順を実行している間は、[ルールアクション] タブを開いたままにしておきます。

ステップ 3: DynamoDB テーブルを作成する

この手順では、ルールアクションから風速データを受信する Amazon DynamoDB テーブルを作成します。

DynamoDB テーブルを作成するには

1. DynamoDB コンソールのダッシュボードで、[テーブルの作成] を選択します。
2. テーブル名を入力します。

3. プライマリキーのページで、以下の操作を行います。
   a. パーティションキーとして **timestamp** を入力します。
   b. [数値] の種類を選択します。
   c. [ソートキーの追加] のチェックボックスをオンにします。
   d. ソートキーとして **asset** と入力し、デフォルトのソートキーの種類は文字列のままにします。
4. [Create] (作成) を選択します。

   テーブルが作成されているという通知が消えたら、テーブルの準備ができたことを示します。

5. アクションの設定のページのタブに戻ります。次のステップを実行している間は、 DynamoDB のタブを開いたままにしておきます。

ステップ 4: DynamoDB ルールアクションを設定する

この手順では、プロパティ値の更新からデータを新しい DynamoDB テーブルに挿入するように Amazon DynamoDB ルールアクションを設定します。

DynamoDB ルールアクションを構成するには

1. [アクションの設定] のページで、[テーブル名] リストを更新し、新しい DynamoDB テーブルを選択します。

2. ロールの作成 を選択して、ルールアクションを実行するための AWS IoT Core アクセスを許可する IAM ロールを作成します。
3. ロール名を入力し、[ロールの作成] を選択します。

4. [添加操作] を選択します。
5. ページの下部にある [ルールの作成] を選択して、ルールの作成を終了します。

   デモのアセットデータが DynamoDB テーブルに表示され始めます。

ステップ 5: DynamoDB のデータを調べる

この手順では、新しい Amazon DynamoDB テーブルのデモのアセットの風速データを調べます。

DynamoDB でアセットデータを検索する

1. DynamoDB テーブルを開いた状態でタブに戻ります。
2. 前に作成したテーブルで、[項目] のタブを選択して、テーブル内のデータを表示します。テーブルに行が表示されない場合は、ページを更新します。数分経過しても行が表示されない場合は、ルールをトラブルシューティング (DynamoDB) を参照してください。

3. テーブルの行で、編集アイコンを選択してデータを展開します。

4. windspeed 構造の横にある矢印を選択して、風速データポイントのリストを展開します。各リストには、風力発電所のデモ AWS IoT SiteWise によって に送信された風速データポイントのバッチが反映されます。独自のルールアクションを設定する場合は、別のデータ形式を使用することもできます。詳細については、「でアセットプロパティ通知をクエリする AWS IoT SiteWise」を参照してください。

チュートリアルを完了したので、追加料金が発生しないように、ルールを無効化または削除し、DynamoDB テーブルを削除します。リソースをクリーンアップするには、「ステップ 6: チュートリアル後にリソースをクリーンアップする」を参照してください。

ステップ 6: チュートリアル後にリソースをクリーンアップする

チュートリアルを完了したら、追加料金が発生しないようにリソースをクリーンアップします。デモの風力発電所アセットは、デモの作成時に選択した期間の終了時に削除されます。エンドポイントを

手動で削除することもできます。詳細については、「AWS IoT SiteWise デモを削除する」を参照してください。

プロパティ値の更新通知を無効にし (デモを削除しなかった場合)、 AWS IoT ルールを無効にまたは削除し、DynamoDB テーブルを削除します。

アセットプロパティでプロパティ値の更新に関する通知を無効にするには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションペインで [アセット] を選択します。

3. Demo Wind Farm Asset の横にある矢印を選択して、風力発電施設のアセットの階層を展開します。

4. Demo Turbine を選択し、[編集] を選択します。

5. [Wind Speed] プロパティの [通知ステータス] を [已禁用] に更新します。

6. ページの下部にある [アセットを保存] を選択します。
7. 各 Demo Turbine Asset に対し、手順 4 ~ 6 を繰り返します。

でルールを無効化または削除するには AWS IoT Core

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで [Act (アクト)] を選択し、[ルール] を選択します。
3. ルールのメニューを選択し、[無効化] または [削除] を選択します。

DynamoDB テーブルを削除するには

1. [DynamoDB console] (DynamoDB コンソール) に移動します。
2. 左のナビゲーションペインで、[テーブル] を選択します。
3. 先ほど作成したテーブル、WindSpeedData を選択します。
4. [テーブルの削除] を選択します。

5. [テーブルの削除] ダイアログで、[削除] を選択します。

にデータを取り込む AWS IoT SiteWise

AWS IoT SiteWise は、産業データを効率的に収集し、対応するアセットと関連付けるように設計されており、産業オペレーションのさまざまな側面を表します。このドキュメントでは、 へのデータ取り込みの実用的な側面に焦点を当て AWS IoT SiteWise、さまざまな産業ユースケースに合わせた複数の方法を提供しています。仮想産業オペレーションを構築する手順については、産業用アセットをモデリングする を参照してください。

産業データは、次のいずれかのオプション AWS IoT SiteWise を使用して に送信できます。

- AWS IoT SiteWise Edge – SiteWise Edge ゲートウェイを AWS IoT SiteWise とデータサーバー間の仲介として使用します。 AWS IoT SiteWise は、SiteWise Edge ゲートウェイをセットアップ AWS IoT Greengrass するために実行できる任意のプラットフォームにデプロイできる AWS IoT Greengrass コンポーネントを提供します。このオプションは、OPC UA サーバープロトコルとのリンクをサポートしています。
- AWS IoT SiteWise API – AWS IoT SiteWise API を使用して、他のソースからデータをアップロードします。ストリーミング BatchPutAssetPropertyValue API を数秒以内の取り込みに使用するか、バッチ指向の CreateBulkImportJob API を使用して、大規模なバッチでのコスト効率の高い取り込みを容易にします。
- AWS IoT コアルール – AWS IoT Core ルールを使用して、 AWS IoT モノまたは別の AWS サービスによって発行された MQTT メッセージからデータをアップロードします。
- AWS IoT Events actions – の特定のイベントによってトリガーされるAWS IoT Events アクションを使用します AWS IoT Events。この方法は、データのアップロードがイベント発生に関連付けられているシナリオに適しています。
- AWS IoT Greengrass ストリームマネージャー – AWS IoT Greengrass ストリームマネージャーを使用して、エッジデバイスを使用してローカルデータソースからデータをアップロードします。このオプションは、データがオンプレミスまたはエッジロケーションから発生する状況に適しています。

これらの方法は、さまざまなソースからのデータを管理するための幅広いソリューションを提供します。各オプションの詳細を調べて、 AWS IoT SiteWise が提供するデータ取り込み機能を包括的に理解します。

のデータストリームを管理する AWS IoT SiteWise

データストリームは、履歴時系列データを含むリソースです。各データストリームは一意のエイリアスによって識別されるため、各データのオリジンを簡単に追跡できます。データストリームは、最初の時系列データが受信 AWS IoT SiteWise されると、 に自動的に作成されます。最初の時系列データがエイリアスで識別された場合、 は、そのエイリアスにアセットプロパティが既に割り当てられていない場合に、そのエイリアスで新しいデータストリーム AWS IoT SiteWise を作成します。または、最初の時系列データがアセット ID とプロパティ ID で識別された場合、 は新しいデータストリーム AWS IoT SiteWise を作成し、そのデータストリームをアセットプロパティに関連付けます。

アセットプロパティにエイリアスを割り当てるには、2 つの方法があります。使用する方法は、データが AWS IoT SiteWise 最初に に送信されるか、アセットが最初に作成されるかによって異なります。

- データを AWS IoT SiteWise 最初に に送信すると、割り当てられたエイリアスを持つデータストリームが自動的に作成されます。後でアセットが作成されたら、AssociateTimeSeriesToAssetProperty API を使用して、データストリームとそのエイリアスをアセットプロパティに関連付けます。
- アセットが最初に作成された場合は、UpdateAssetProperty API を使用してアセットプロパティにエイリアスを割り当てます。データが後で に送信されると AWS IoT SiteWise、データストリームが自動的に作成され、アセットプロパティに関連付けられます。

現在、データストリームを関連付けることができるのは、測定値のみです。測定値とは、タイムスタンプ付きの温度値やタイムスタンプ付きの回転数 (RPM) 値など、デバイスの raw センサーデータストリームを表す一種のアセットプロパティです。

これらの測定値がメトリクスまたは変換を定義すると、受信データは特定の計算をトリガーします。アセットプロパティは一度に 1 つのデータストリームにのみリンクできることに注意してください。

AWS IoT SiteWise は `TimeSeries` Amazon リソースネーム (ARN) リソースに を使用してストレージ料金を決定します。詳細については、「AWS IoT SiteWise 料金」を参照してください。

以下のセクションでは、 AWS IoT SiteWise コンソールまたは API を使用してデータストリームを管理する方法を示します。

トピック

- アクセス許可と設定を構成する

- データストリームをアセットプロパティに関連付ける
- アセットプロパティからデータストリームの関連付けを解除する
- データストリームを削除する
- アセットプロパティエイリアスを更新する
- 一般的なシナリオ

アクセス許可と設定を構成する

データストリームは、最初の時系列データが受信 AWS IoT SiteWise されると、 に自動的に作成されます。取り込まれたデータがアセットプロパティに関連付けられていない場合、 は、アセットプロパティに関連付けるように設定できる、関連付けが解除された新しいデータストリーム AWS IoT SiteWise を作成します。IAM ポリシーを使用して、データを送信するゲートウェイのアクセスコントロールを設定し AWS IoT SiteWise、取り込むデータのタイプを指定します。

次の IAM ポリシーは、アセットプロパティに関連付けられたデータストリームへのデータ取り込みを許可しながら、ゲートウェイからの関連付け解除されたデータ取り込みを無効にします。

Example ゲートウェイからの関連付け解除されたデータ取り込みを無効にする IAM ユーザーポリシー

JSON

```
{
    "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 ゲートウェイからのすべてのデータ取り込みを無効にする IAM ユーザーポリシー

JSON

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

データストリームをアセットプロパティに関連付ける

AWS IoT SiteWise コンソール または を使用してデータストリームを管理します AWS CLI。

Console

AWS IoT SiteWise コンソールを使用してデータストリームを管理します。

データストリームを管理するには (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Data streams] (データストリーム) を選択します。
3. データストリームエイリアスでフィルタリングするか、フィルタードロップダウンメニューで関連付け解除されたデータストリームを選択して、データストリームを選択します。
4. 更新するデータストリームを選択します。複数のデータストリームを選択できます。右上のデータストリームの管理をクリックします。
5. データストリームの関連付けを更新するから関連付けるデータストリームを選択し、測定の選択ボタンをクリックします。
6. 測定の選択セクションで、対応するアセット測定プロパティを見つけます。測定値を選択し、選択をクリックします。
7. ステップ 3 で選択した他のデータストリームに対してステップ 4 と 5 を実行します。アセットプロパティをすべてのデータストリームに割り当てます。
8. 更新を選択して変更をコミットします。更新を確認する確認バナーが表示されます。

AWS CLI

データストリーム (エイリアスで識別) をアセットプロパティ (IDs で識別) に関連付けるには、次のコマンドを実行します。

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

アセットプロパティからデータストリームの関連付けを解除する

Console

AWS IoT SiteWise コンソールを使用して、アセットプロパティからデータストリームの関連付けを解除します。

アセットプロパティからデータストリームの関連付けを解除するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Data streams] (データストリーム) を選択します。
3. データストリームエイリアスでフィルタリングするか、フィルタードロップダウンメニューで関連付けられたデータストリームを選択して、データストリームを選択します。
4. 関連付けを解除するデータストリームを選択します。データストリームエイリアス列にはエイリアスが含まれている必要があります。アセット名とアセットプロパティ名の列には、データストリームが関連付けられているアセットプロパティの値が含まれている必要があります。複数のデータストリームを選択できます。
5. 右上のデータストリームの管理をクリックします。
6. データストリームの関連付けの更新セクションで、測定名列の X をクリックします。`submitted` ステータス列にステータスが表示されます。
7. 更新を選択して変更をコミットします。データストリームとアセットプロパティの関連付けが解除され、エイリアスを使用してデータストリームを識別できるようになりました。

AWS CLI

(`ID`とそのエイリアスによって識別される) アセットプロパティからデータストリームの関連付けを解除するには、次のコマンドを実行します。

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

データストリームとアセットプロパティの関連付けが解除され、エイリアスを使用してデータストリームを識別できるようになりました。エイリアスは、データストリームに関連付けられているため、アセットプロパティに関連付けられなくなりました。

データストリームを削除する

プロパティがアセットモデルから削除されると、 はアセットモデルによって管理されるすべてのアセットからプロパティとそのデータストリーム AWS IoT SiteWise を削除します。また、アセットが削除されると、アセットのすべてのプロパティとそのデータストリームも削除されます。データスト

リームデータを保持する必要がある場合は、削除する前にアセットプロパティとの関連付けを解除する必要があります。

> **Warning**
> プロパティがアセットから削除されると、関連付けられたデータストリームも削除されます。データストリームを保持するには、まずアセットプロパティから関連付けを解除してから、アセットモデルからプロパティを削除するか、アセットを削除します。

Console

AWS IoT SiteWise コンソールを使用して、アセットプロパティからデータストリームの関連付けを解除します。

データストリームを削除するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Data streams] (データストリーム) を選択します。
3. データストリームエイリアスでフィルタリングしてデータストリームを選択します。
4. 削除するデータストリームを選択します。複数のデータストリームを選択できます。
5. データストリームを削除するには、削除ボタンを選択します。

AWS CLI

DeleteTimeSeries API を使用して、エイリアスで特定のデータストリームを削除します。

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

アセットプロパティエイリアスを更新する

エイリアスは AWS リージョン内で一意である必要があります。これには、アセットプロパティとデータストリームの両方のエイリアスが含まれます。別のプロパティまたはデータストリームがそのエイリアスを使用している場合は、アセットプロパティにエイリアスを割り当てないでください。

Console

AWS IoT SiteWise コンソールを使用して、アセットプロパティエイリアスを更新します。

アセットプロパティエイリアスを更新するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [アセット] を選択します。
3. テーブルからアセットを選択します。
4. [Edit] ボタンをクリックします。
5. プロパティテーブルでプロパティタイプを選択します。
6. プロパティを検索し、プロパティエイリアステキストフィールドに新しいエイリアスを入力します。
7. 保存ボタンをクリックして変更を保存します。

AWS CLI

アセットプロパティのエイリアスを更新するには、次のコマンドを実行します。

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

> **Note**
> プロパティ通知が現在有効になっている場合は、引き続き有効になるように再度指定する必要があります。

一般的なシナリオ

データストリームを移動する

データストリームの関連付けを別のアセットプロパティに変更するには、まず現在のアセットプロパティからデータストリームの関連付けを解除します。アセットプロパティからデータストリームの関連付けを解除する場合、そのアセットプロパティにエイリアスを割り当てる必要があります。

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

次に、データストリームを新しいアセットプロパティに再割り当てします。

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

アセットプロパティにエイリアスを割り当てるときのエラー

`UpdateAssetProperty` API を使用して プロパティにエイリアスを割り当てると、次のエラーメッセージが表示されることがあります。

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

このエラーメッセージは、現在別のプロパティまたはデータストリームで使用されているため、エイリアスが プロパティに割り当てられていないことを示します。

これは、データがエイリアス AWS IoT SiteWise で に取り込まれている場合に発生します。別のデータストリームまたはアセットプロパティで使用されていないエイリアスでデータが送信されると、そのエイリアスで新しいデータストリームが作成されます。以下の 2 つのオプションで問題を解決します。

- `AssociateTimeSeriesToAssetProperty` API を使用して、データストリームとそのエイリアスをアセットプロパティに関連付けます。

- データ取り込みを一時的に停止し、データストリームを削除します。UpdateAssetProperty API を使用してエイリアスをアセットプロパティに割り当て、データインジェストを再度オンにします。

データストリームをアセットプロパティに関連付けるときのエラー

データストリームをアセットプロパティに関連付けると、次のエラーメッセージが表示されます。

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

このエラーメッセージは、アセットプロパティが既にデータを含むデータストリームに関連付けられていることを示します。他のデータストリームをそのアセットプロパティに関連付ける前に、そのデータストリームの関連付けを解除または削除する必要があります。

> **Note**
>
> アセットプロパティからデータストリームの関連付けを解除すると、プロパティに割り当てられたエイリアスがデータストリームに割り当てられます。そのエイリアスを プロパティに割り当てたままにするには、データストリームの関連付けを解除する前に、そのプロパティに新しいエイリアスを割り当てます。

アセットプロパティに保存されているデータを保持するには、次の手順を実行します。

- 新しいデータストリームが作成されないように、アセットプロパティにデータが取り込まれていないことを確認します。
- UpdateAssetProperty API を使用して、現在割り当てられているデータストリームに与えられる新しいエイリアスを設定します。
- DisassociateTimeSeriesFromAssetProperty API を使用して、アセットプロパティから現在のデータストリームの関連付けを解除します。
- AssociateTimeSeriesToAssetProperty API を使用して、目的のデータストリームをアセットプロパティに関連付けます。

アセットプロパティに保存されているデータを削除する必要がある場合は、次の操作を行います。

- 新しいデータストリームが作成されないように、アセットプロパティにデータが取り込まれていないことを確認します。
- `DeleteTimeSeries` API を使用して、現在割り当てられているデータストリームを削除します。
- `AssociateTimeSeriesToAssetProperty` API を使用して、目的のデータストリームをアセットプロパティに関連付けます。

AWS IoT SiteWise APIs を使用してデータを取り込む

AWS IoT SiteWise APIs を使用して、タイムスタンプ付きの産業データをアセットの属性と測定プロパティに送信します。APIsを受け入れます。 timestamp-quality-value

BatchPutAssetPropertyValue API

BatchPutAssetPropertyValue オペレーションを使用して、データをアップロードしてください。このオペレーションを使用すると、一度に複数のデータエントリをアップロードできるため、複数のデバイスからデータを収集して、1 つのリクエストですべてを送信できます。

> **Important**
>
> BatchPutAssetPropertyValue オペレーションには以下のクォータが適用されます。
>
> - 1 リクエストあたりのエントリ数: 最大 10。
> - 1 エントリあたりのプロパティ値 (TQV データポイント) 数: 最大 10。
> - AWS IoT SiteWise は、7 日以上前または 10 分以上先のタイムスタンプを持つデータを拒否します。
>
> これらのクォータの詳細については、[API Reference AWS IoT SiteWise] (APIリファレンス) のBatchPutAssetPropertyValue を参照してください。

アセットプロパティを特定するには、次のいずれかを指定します。

- データが送信されるアセットプロパティの `assetId` と `propertyId`。
- データストリームのエイリアスである `propertyAlias` (例えば、`/company/windfarm/3/turbine/7/temperature`)。このオプションを使用するには、最初にアセットプロパティのエイリアスを設定する必要があります。プロパティのエイリアスを設定するには、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

次の例は、JSON ファイルに格納されているペイロードから風力タービンの温度と 1 分あたりの回転数 (RPM) の読み取り値を送信する方法を示しています。

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

のペイロードの例`batch-put-payload.json`には、次のコンテンツがあります。

```
{
  "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 を として指定するとtrue、失敗につながらないすべての値を取り込むことができます。デフォルトの動作は false です。値が無効の場合、エントリ全体が取り込みに失敗します。

ペイロードの各エントリには、一意の文字列として定義できる entryId が含まれています。リクエストのエントリが失敗した場合、各エラーには、対応するリクエストの entryId が含まれるため、再試行するリクエストを確認できます。

propertyValues のリストの各構造は、value、timestamp、およびオプションで quality を含むタイムスタンプ品質値 (TQV) 構造です。

- value - 設定中のプロパティの型に応じて、次のいずれかのフィールドを含む構造。
  - booleanValue
  - doubleValue
  - integerValue
  - stringValue
  - nullValue
- nullValue – 値が Null で、品質が BADまたは のプロパティ値のタイプを示す次のフィールドを持つ構造UNCERTAIN。
  - valueType – {"B"、"D"、"S"、"I"} の列挙型
- timestamp エポック時間からの現在の UNIX エポック時刻 (秒単位) を含む構造。timeInSeconds時間的に正確なデータがある場合は、 timestamp構造で offsetInNanosキーを設定することもできます。 は、過去 7 日以上前のタイムスタンプを持つ

データポイント、または 10 分以上前のタイムスタンプを持つデータポイント AWS IoT SiteWise を拒否します。

- `quality` -(オプション) 次の品質の文字列のいずれか。
  - `GOOD` - (デフォルト) データはいずれの問題による影響も受けません。
  - `BAD` - データはセンサーの障害などの問題による影響を受けます。
  - `UNCERTAIN` - データはセンサーの不正確さなどの問題による影響を受けます。

  が計算でデータ品質 AWS IoT SiteWise を処理する方法の詳細については、「数式におけるデータ品質」を参照してください。

CreateBulkImportJob API

`CreateBulkImportJob` API を使用して、Amazon S3 から大量のデータをインポートします。Amazon S3 のデータは、CSV 形式で保存されている必要があります。データファイルには、次の列を含めることができます。

> **Note**
>
> 1970 年 1 月 1 日 00:00:00 UTC より前のデータはサポートされていません。
> アセットプロパティを特定するには、次のいずれかを指定します。
>
> - データの送信先のアセットプロパティの `ASSET_ID` および `PROPERTY_ID`。
> - データストリームのエイリアスである `ALIAS` (例えば、`/company/windfarm/3/turbine/7/temperature`)。このオプションを使用するには、最初にアセットプロパティのエイリアスを設定する必要があります。プロパティのエイリアスを設定する方法については、「the section called “データストリームの管理”」を参照してください。

- `ALIAS` – OPC UA サーバーのデータストリームパスなど、プロパティを識別するエイリアス (例: `/company/windfarm/3/turbine/7/temperature`)。詳細については、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。
- `ASSET_ID` – アセットの ID。
- `PROPERTY_ID` – アセットプロパティの ID。
- `DATA_TYPE` – プロパティのデータ型。次のいずれかを指定できます。
  - `STRING` - 最大 1024 バイトの文字列。

  - INTEGER - [-2,147,483,648, 2,147,483,647] の範囲を持つ 32 ビットの符合付き整数。
  - DOUBLE - [-10 ^ 100、10 ^ 100] の範囲および IEEE 754 倍精度の浮動小数点数。
  - BOOLEAN – true または false
- TIMESTAMP_SECONDS – データポイントのタイムスタンプ (Unix エポック時間形式)。
- TIMESTAMP_NANO_OFFSET – TIMESTAMP_SECONDS から変換されたナノ秒オフセット。
- QUALITY – (オプション) アセットプロパティ値の品質。値には次のいずれかを指定できます。
  - GOOD - (デフォルト) データはいずれの問題による影響も受けません。
  - BAD - データはセンサーの障害などの問題による影響を受けます。
  - UNCERTAIN - データはセンサーの不正確さなどの問題による影響を受けます。

  が計算でデータ品質 AWS IoT SiteWise を処理する方法の詳細については、「数式におけるデータ品質」を参照してください。
- VALUE – アセットプロパティの値。

Example .csv 形式のデータファイル

```
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 には、一括インポートジョブを作成し、既存のジョブに関する情報を取得するための次の API オペレーションが用意されています。

- CreateBulkImportJob – 新しい一括インポートジョブを作成します。
- DescribeBulkImportJob – 一括インポートジョブに関する情報を取得します。

- ListBulkImportJob – すべての一括インポートジョブの概要をまとめたページ番号付きリストを取得します。

AWS IoT SiteWise 一括インポートジョブを作成する (AWS CLI)

CreateBulkImportJob API オペレーションを使用して、Amazon S3 から AWS IoT SiteWiseにデータを転送します。CreateBulkImportJob API を使用すると、大量の履歴データの取り込みと、分析データストリームのバッファ取り込みを小さなバッチで行うことができます。データの取り込みのためのコスト効率の高いプリミティブを提供します。次の例では AWS CLIを使用しています。

> **Important**
>
> 一括インポートジョブを作成する前に、 AWS IoT SiteWise ウォーム階層または AWS IoT SiteWise コールド階層を有効にする必要があります。詳細については、「でストレージ設定を構成する AWS IoT SiteWise」を参照してください。
> CreateBulkImportJob API は、adaptive-ingestion-flag パラメータを設定するオプション AWS IoT SiteWise を使用して、履歴データの への取り込みをサポートします。
>
> - `false` に設定すると、API は計算や通知をトリガーせずに履歴データを取り込みます。
> - `true` に設定すると、API は新しいデータを取り込み、メトリクスを計算し、データを変換して、7 日以内に進行中の分析と通知を最適化します。

以下のコマンドを実行してください。*`file-name`* の部分は、一括インポートジョブの設定を含むファイルの名前に置き換えます。

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

Example 一括インポートジョブの設定

以下は、構成設定の例です。

- *`adaptive-ingestion-flag`* を `true` または `false` に置き換えます。
  - `false` に設定すると、一括インポートジョブは履歴データを AWS IoT SiteWiseに取り込みます。
  - `true` に設定すると、一括インポートジョブは以下を実行します。
    - 新しいデータを取り込みます AWS IoT SiteWise。

  - メトリクスを計算して変換し、7 日以内のタイムスタンプを持つデータの通知をサポートします。

- *delete-files-after-import-flag* を に置き換えtrueて、 AWS IoT SiteWise ウォーム階層ストレージへの取り込み後に Amazon S3 データバケットからデータを削除します。
- amzn-s3-demo-bucket-*for-errors* を、この一括インポートジョブに関連するエラーの送信先である Amazon S3 バケットの名前に置き換えます。
- amzn-s3-demo-bucket-*for-errors-prefix* を、この一括インポートジョブに関連するエラーの送信先である Amazon S3 バケットのプレフィックスに置き換えます。

  Amazon S3 では、このプレフィックスをフォルダ名として使用して、バケット内のデータを整理します。Amazon S3 バケット内の各オブジェクトには、バケット内の一意の識別子であるキーが含まれます。バケット内のすべてのオブジェクトは、厳密に 1 個のキーを持ちます。このプレフィックスは、スラッシュ (/) で終わる必要があります。詳細は、『Amazon Simple Storage Service ユーザーガイド』の「プレフィックスを使用してオブジェクトを整理する」を参照してください。

- amzn-s3-demo-bucket-*data* を、データのインポート元である Amazon S3 バケットの名前に置き換えます。
- *data-bucket-key* の部分は、データを含む Amazon S3 オブジェクトのキーに置き換えます。すべてのオブジェクトには、一意の識別子であるキーがあります。すべてのオブジェクトは、厳密に 1 個のキーを持ちます。
- *data-bucket-version-id* の部分は、データを含む Amazon S3 オブジェクトの特定のバージョンを識別するバージョン ID に置き換えます。このパラメータはオプションです。
- *column-name* の部分は、.csv ファイルに指定されている列名に置き換えます。
- *job-name* の部分は、一括インポートジョブを識別する一意の名前に置き換えます。
- *job-role-arn* の部分は、 AWS IoT SiteWise が Amazon S3 データの読み取ることを許可する IAM ロールに置き換えます。

> **Note**
>
> 次の例に示すようなアクセス許可をロールが持っていることを確認してください。amzn-s3-demo-bucket-*data* を、データを含む Amazon S3 バケットの名前に置き換えます。また、*amzn-s3-demo-bucket-for-errors* を、この一括インポートジョブに関連するエラーの送信先である Amazon S3 バケットの名前に置き換えます。

JSON

```
{
    "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 レスポンス

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

AWS IoT SiteWise 一括インポートジョブを記述する (AWS CLI)

DescribeBulkImportJob API オペレーションを使用して、 AWS IoT SiteWiseの特定の一括インポートジョブに関する情報を取得します。このオペレーションは、ジョブのステータス、作成時間、ジョブが失敗した場合のエラー情報などの詳細を返します。このオペレーションを使用して、ジョブの進行状況をモニタリングし、問題をトラブルシューティングできます。`DescribeBulkImportJob` を使用するには、`CreateBulkImportJob` オペレーションのジョブ ID が必要です。API は次のような情報を返します。

- Amazon S3 バケットの場所とキーを含む、インポートされるファイルのリスト
- エラーレポートの場所 (該当する場合)
- ファイル形式や CSV 列名など、ジョブ設定の詳細
- ジョブの作成と最終更新のタイムスタンプ
- 現在のジョブのステータス (例えば、ジョブは進行中、完了、失敗など)
- インポートジョブに使用される IAM ロール ARN

完了したジョブについては、結果を確認して、データ統合が成功したことを確認します。ジョブが失敗した場合は、エラーの詳細を調べて、問題を診断し、解決します。

job-ID の部分は、取得する一括インポートジョブの ID に置き換えます。

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

Example レスポンス

```
{
   "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 一括インポートジョブを一覧表示する (AWS CLI)

ListBulkImportJobs API オペレーションを使用して、 AWS IoT SiteWiseの一括インポートジョブの概要のリストを取得します。このオペレーションは、データインポートプロセスをモニタリングおよび管理するための効率的な手段となります。ジョブごとに以下の主要情報を返します。

- ジョブ ID。各一括インポートジョブの一意識別子
- ジョブ名。ジョブの作成時にジョブに割り当てた名前。
- 現在のステータス。ジョブの現在の状態 (COMPLETED、RUNNING、FAILED など)

ListBulkImportJobs は、すべての一括インポートジョブの概要を包括的に把握するために特に役立ちます。これにより、複数のデータインポートを追跡し、注意が必要なジョブを特定し、整理されたワークフローを維持するのに役立ちます。オペレーションはページ分割をサポートしているため、多数のジョブの概要を効率的に取得できます。このオペレーションによって返されるジョブ ID を DescribeBulkImportJob オペレーションで使用して、特定のジョブに関する詳細情報を取得できます。この 2 ステップのプロセスでは、まず、すべてのジョブの概要を把握し、次に目的のジョブの詳細をドリルダウンできます。`ListBulkImportJobs` を使用する場合、フィルターを適用して結果を絞り込むことができます。例えば、ジョブのステータスに基づいてジョブをフィルタリングして、完了したジョブのみ、または実行中のジョブのみを取得できます。この機能は、現在のタスクに最も関連性の高い情報に焦点を当てるのに役立ちます。このオペレーションは、より多くの結果がある場合には、`nextToken` も返します。このトークンを後続の呼び出しで使用して、次の一連のジョブ概要を取得できるため、大量のジョブがある場合でも、すべての一括インポートジョブを繰り返し実行できます。次の例は、 `ListBulkImportJobs`で AWS CLI を使用して、完了したジョブのリストを取得する方法を示しています。

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

Example 完了したジョブフィルターのレスポンス

```
{
```

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

このコマンドは、`ListBulkImportJobs` を使用して、失敗で完了したジョブのリストを取得する方法を示しています。最大は 50 件の結果に設定されており、ページ分割された結果には次のトークンを使用しています。

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

AWS IoT Core ルール AWS IoT SiteWise を使用して にデータを取り込む

のルールを使用して、 AWS IoT モノやその他の AWS サービス AWS IoT SiteWise から にデータを送信します AWS IoT Core。ルールは MQTT メッセージを変換し、 AWS サービスとやり取りするアクションを実行します。 AWS IoT SiteWise ルールアクションは、 AWS IoT SiteWise API から BatchPutAssetPropertyValue オペレーションにメッセージデータを転送します。詳しくは、[AWS IoT Developer Guide] (デベロッパーガイド) の[Rules] (ルール) [AWS IoT SiteWise action] (アクション) をご覧ください。

デバイスシャドウを介してデータを取り込むルールをセットアップするために必要なステップを説明するチュートリアルに従うには、「AWS IoT モノ AWS IoT SiteWise から にデータを取り込む」を参照してください。

から他の AWS サービス AWS IoT SiteWise にデータを送信することもできます。詳細については、「他の AWS サービスとやり取りする」を参照してください。

トピック

- 必要なアクセス権を付与 AWS IoT する
- AWS IoT SiteWise ルールアクションを設定する
- の基本的な取り込みでコストを削減する AWS IoT SiteWise

必要なアクセス権を付与 AWS IoT する

IAM ロールを使用して、各ルールがアクセスできる AWS リソースを制御します。ルールを作成する前に、必要な AWS リソースに対してアクションを実行することをルールに許可するポリシーを持つ IAM ロールを作成する必要があります。 は、ルールの実行時にこのロールを AWS IoT 引き受けます。

AWS IoT コンソールでルールアクションを作成する場合は、ルートアセットを選択して、選択したアセット階層にアクセスできるロールを作成できます。ルールのロールを手動で定義する方法の詳細については、「 AWS IoT デベロッパーガイドAWS IoT 」の「必要なアクセス許可の付与」および「ロールのアクセス許可を渡す」を参照してください。

AWS IoT SiteWise ルールアクションでは、ルールがデータを送信するアセットプロパティ`iotsitewise:BatchPutAssetPropertyValue`へのアクセスを許可するロールを定義する必要があります。セキュリティを向上させるために、 `Condition`プロパティで AWS IoT SiteWise アセット階層パスを指定できます。

次の信頼ポリシーの例では、特定のアセットとその子へのアクセスを許可します。

JSON

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

```
      }
    }
  ]
}
```

ポリシーから Condition を削除すると、すべてのアセットへのアクセスを許可できます。次の信頼ポリシーの例では、現在のリージョン内のすべてのアセットへのアクセスを許可します。

JSON

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

AWS IoT SiteWise ルールアクションを設定する

AWS IoT SiteWise ルールアクションは、ルールを開始した MQTT メッセージからアセットプロパティにデータを送信します AWS IoT SiteWise。複数のデータエントリを異なるアセットプロパティに同時にアップロードできるため、デバイスのすべてのセンサーの更新を 1 つのメッセージで送信できます。また、各データ入力に一度に複数のデータポイントをアップロードすることもできます。

> **Note**
>
> ルールアクション AWS IoT SiteWise を使用して にデータを送信する場合、データは BatchPutAssetPropertyValueオペレーションのすべての要件を満たしている必要があります。たとえば、現在の UNIX エポック時間から 7 日以上前ののタイムスタンプをデータで使用することはできません。詳細については、「AWS IoT SiteWise API を使用したデータの取り込み」を参照してください。

ルールアクションの各データエントリについて、アセットプロパティを識別し、そのアセットプロパティの各データポイントのタイムスタンプ、品質、および値を指定します。ルールアクションには、すべてのパラメータに対する文字列を指定します。

エントリ内のアセットのプロパティを識別するには、以下のいずれかを指定します。

- データを送信するアセットプロパティの [アセット ID] (`assetId`) と [プロパティ ID] (`propertyId`)。アセット ID とプロパティ ID は、 AWS IoT SiteWise コンソールを使用して確認できます。アセット ID がわかっている場合は、 を使用して DescribeAsset AWS CLI を呼び出し、プロパティ ID を見つけることができます。
- [Property alias (プロパティエイリアス)] (`propertyAlias`)。これは、データストリームのエイリアス (`/company/windfarm/3/turbine/7/temperature` など) です。このオプションを使用するには、最初にアセットプロパティのエイリアスを設定する必要があります。プロパティのエイリアスを設定する方法については、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

各エントリーのタイムスタンプには、お使いの機器から報告されたタイムスタンプ、または AWS IoT Coreから提供されたタイムスタンプを使用できます。タイムスタンプは 2 つのパラメータがあります。

- [Time in seconds] (時間 (秒)) (`timeInSeconds`) - センサーまたは機器がデータを報告した UNIX エポック時間 (秒)。
- [Offset in nanos] (オフセット (ナノ秒)) (`offsetInNanos`) -(オプション) 時間 (秒) からのオフセット (ナノ秒)。

> **Important**
>
> タイムスタンプが文字列であったり、小数部があったり、秒単位でない場合、 AWS IoT SiteWise はリクエストを拒否します。タイムスタンプを秒とナノ秒のオフセットに変換する必要があります。 AWS IoT ルールエンジンの機能を使用して、タイムスタンプを変換します。詳細については次を参照してください:
>
> - 正確な時間を報告しないデバイスのタイムスタンプを取得する。
> - 文字列形式のタイムスタンプを変換する

アクションのいくつかのパラメータに置換テンプレートを使用して、コンピューティングを実行したり、関数を呼び出したり、メッセージペイロードから値を引き出したりすることができます。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Substitution templates] (置換テンプレート) を参照してください。

> **Note**
>
> 置換テンプレート内の式は SELECT ステートメントとは別に評価されるため、AS 句を使用して作成されたエイリアスは置換テンプレートを使用して参照することはできません。サポートされている関数と演算子に加えて、元のペイロードに存在する情報のみを参照できます。

トピック

- 正確な時間を報告しないデバイスのタイムスタンプを取得する。
- 文字列形式のタイムスタンプを変換する
- ナノ秒精度のタイムスタンプ文字列を変換する。
- ルール設定の例。
- ルールアクションのトラブルシューティング

正確な時間を報告しないデバイスのタイムスタンプを取得する。

センサーまたは機器が正確な時間データを報告しない場合は、timestamp() を使用して AWS IoT ルールエンジンから現在の Unix エポック時間を取得します。この関数は、ミリ秒単位で時間を出力するので、時間を秒単位、オフセットをナノ秒単位に変換する必要があります。そのためには、次の変換を使用します。

- [Time in seconds (時間 (秒))] (`timeInSeconds`) には、**`${floor(timestamp() / 1E3)}`** を使用して、時間をミリ秒から秒に変換します。
- [Offset in nanos (オフセット (ナノ秒))] (`offsetInNanos`) には、**`${(timestamp() % 1E3) * 1E6}`** を使用して、タイムスタンプオフセット (ナノ秒) をコンピューティングします。

文字列形式のタイムスタンプを変換する

センサーや機器が時刻データを文字列形式で報告する場合 (例えば、`2020-03-03T14:57:14.699Z`)、time_to_epoch(String, String) を使用します。この関数は、タ

イムスタンプとフォーマットパターンをパラメータとして入力し、時刻をミリ秒単位で出力します。次に、時間を秒単位で、オフセットをナノ秒単位で変換する必要があります。そのためには、次の変換を使用します。

- [Time in seconds] (秒単位の時間) (`timeInSeconds`) の場合、**`${floor(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z'") / 1E3)}`** でタイムスタンプ文字列をミリ秒に変換し、さらに秒に変換します。
- [Offset in nanos](オフセット (ナノ秒)) (`offsetInNanos`) には、**`${(time_to_epoch("2020-03-03T14:57:14.699Z", "yyyy-MM-dd'T'HH:mm:ss'Z'") % 1E3) * 1E6}`** を使用して、タイムスタンプ文字列オフセット (ナノ秒) をコンピューティングします。

> **Note**
>
> `time_to_epoch` 関数は、最大ミリ秒精度のタイムスタンプ文字列をサポートします。マイクロ秒またはナノ秒の精度で文字列を変換するには、タイムスタンプを数値に変換するためにルールが呼び出す AWS Lambda 関数を設定します。詳細については、「ナノ秒精度のタイムスタンプ文字列を変換する。」を参照してください。

ナノ秒精度のタイムスタンプ文字列を変換する。

デバイスがナノ秒精度のタイムスタンプ情報を文字列形式 (`2020-03-03T14:57:14.699728491Z` など) で送信する場合、次のステップでルールアクションを設定します。文字列のタイムスタンプを秒単位の時間 (`timeInSeconds`) とナノ単位のオフセット () に変換する AWS Lambda 関数を作成できます`offsetInNanos`。次に、ルールのアクションパラメータで aws_lambda(functionArn, inputJson) を使用して、その Lambda 関数を呼び出し、その出力をルールで使用します。

> **Note**
>
> このセクションでは、次のリソースの作成方法を熟知していることを前提とした高度な手順について説明します。
>
> - Lambda 関数。詳細については、「AWS Lambda デベロッパーガイド」の「初めての Lambda 関数の作成」を参照してください。

- AWS IoT ルールアクションを持つ AWS IoT SiteWise ルール。詳細については、「AWS IoT Core ルール AWS IoT SiteWise を使用して にデータを取り込む」を参照してください。

タイムスタンプ文字列を解析する AWS IoT SiteWise ルールアクションを作成するには

1. 次のプロパティを持つ Lambda 関数を作成します。

   - [Function name] (関数名) - わかりやすい関数名を使用します (例: **`ConvertNanosecondTimestampFromString`**)。
   - ランタイム - Python 3.11 (`python3.11`) など、Python 3 ランタイムを使用します。
   - [Permissions] (アクセス許可) — 基本的な Lambda アクセスアクセス許可を持つロール (AWS LambdaBasicExecutionRole) を作成します。
   - レイヤー — numpy を使用する Lambda 関数用に AWS SDKPandas-Python311 レイヤーを追加します。
   - [Function code] (関数コード) - 次の関数コードを使用します。この関数コードでは、`timestamp` という文字列引数を消費し、そのタイムスタンプの `timeInSeconds` と `offsetInNanos` 値を出力します。

```
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
    }
```

この Lambda 関数は、NumPy からdatetime64を使用してISO 8601形式のタイムスタンプ文字列を入力します。

> **Note**
>
> タイムスタンプ文字列が ISO 8601 形式でない場合、タイムスタンプ形式を定義した pandas で解決策を実装することが可能です。詳しくはpandas.to_datetimeを参照してください。

2. ルールの AWS IoT SiteWise アクションを設定するときは、秒単位の時間 (`timeInSeconds`) とナノ単位のオフセット () に次の置換テンプレートを使用しますoffsetInNanos。これらの置換テンプレートは、メッセージペイロードに `timestamp` のタイムスタンプ文字列が含まれていることを前提としています。この `aws_lambda` 関数は 2 番目のパラメータに JSON 構造消費するため、必要に応じて以下の置換テンプレートを変更できます。

   - [Time in seconds (時間 (秒))] (`timeInSeconds`) には、次の置換テンプレートを使用します。

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

   - Offset in nanos (オフセット (ナノ秒))] (`offsetInNanos`) には、次の置換テンプレートを使用します。

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

   パラメータごとに、*`region`* と *`account-id`* をリージョンと AWS アカウント ID に置き換えます。Lambda 関数に別の名前を使用した場合は、それも変更してください。

3. アクセス AWS IoT 許可を使用して 関数を呼び出すアクセス`lambda:InvokeFunction`許可を付与します。詳細については、「aws_lambda (functionArn, inputJson) 」を参照してください。

4. ルールをテストし (AWS IoT MQTT テストクライアントを使用するなど) 、 が送信するデータ AWS IoT SiteWise を受信することを確認します。

ルールが期待どおりに動作しない場合は、「AWS IoT SiteWise ルールアクションのトラブルシューティング」を参照してください。

> **Note**
>
> このソリューションは、タイムスタンプ文字列ごとに Lambda 関数を 2 回呼び出します。ルールが各ペイロードで同じタイムスタンプを持つ複数のデータポイントを処理する場合、別のルールを作成して Lambda 関数呼び出し回数を減らすことができます。
> そのためには、Lambda を呼び出し、タイムスタンプ文字列を `timeInSeconds` と `offsetInNanos` に変換して元のペイロードをパブリッシュする再パブリッシュアクションを含むルールを作成します。次に、変換されたペイロードを消費するルールアクションを使用して AWS IoT SiteWise ルールを作成します。この方法では、ルールが Lambda を呼び出す回数を減らしますが、 AWS IoT ルールアクションの実行回数を増やします。このソリューションをユースケースに適用する場合は、各サービスの価格を考慮してください。

ルール設定の例。

このセクションでは、 AWS IoT SiteWise アクションでルールを作成するためのルール設定の例を示します。

Example プロパティエイリアスをメッセージトピックとして使用するルールアクションの例

次の例では、 トピック (topic() 経由) をプロパティエイリアスとして使用してアセットプロパティを識別する AWS IoT SiteWise アクションを含むルールを作成します。この例を使用して、すべての風力発電施設のすべての風力タービンに倍精度型のデータを取り込むための 1 つのルールを定義します。この例では、すべてのタービンアセットのプロパティにプロパティエイリアスを定義する必要があります。整数型データを取り込むには、同様のルールをもう一つ定義する必要があります。

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

`sitewise-rule-payload.json` のペイロードの例には、次のコンテンツが含まれています。

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

このルールアクションでは、次のメッセージを風力タービンプロパティエイリアス (`/company/windfarm/3/turbine/7/temperature` など) に、データを取り込むためのトピックとして送信します。

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

Example timestamp () を使用して時刻を決定するルールアクションの例

次の例では、IDs でアセットプロパティを識別し、timestamp() を使用して現在の時刻を決定する AWS IoT SiteWise アクションを含むルールを作成します。

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

sitewise-rule-payload.json のペイロードの例には、次のコンテンツが含まれています。

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

このルールアクションでは、次のメッセージを my/asset/property/topic に送信してデータを取り込みます。

```
{
  "type": "double",
```

```
  "value": "38.3"
}
```

ルールアクションのトラブルシューティング

で AWS IoT SiteWise ルールアクションをトラブルシューティングするには AWS IoT Core、CloudWatch Logs を設定するか、ルールの再発行エラーアクションを設定します。詳細については、「AWS IoT SiteWise ルールアクションのトラブルシューティング」を参照してください。

の基本的な取り込みでコストを削減する AWS IoT SiteWise

AWS IoT Core には、AWS IoT メッセージングコストを発生させ AWS IoT Core ずに を介してデータを送信するために使用できる Basic Ingest と呼ばれる機能が用意されています。基本的な取り込みは、取り込みパスからパブリッシュ/サブスクライブのメッセージブローカーを除去して、大量のデータインジェストワークロードのデータフローを最適化します。メッセージをルーティングするルールがわかっている場合は、基本的な取り込みを使用できます。

基本的な取り込みを使用するには、特別なトピック、`$aws/rules/rule-name` を使用して、特定のルールに直接メッセージを送信します。たとえば、`SiteWiseWindFarmRule` という名前のルールにメッセージを送信するには、トピック `$aws/rules/SiteWiseWindFarmRule` にメッセージを送信します。

ルールアクションで、トピック (10 進数) を含む置換テンプレートが使用されている場合、`$aws/rules/rule-name/original-topic` などの基本的な取り込みの特別なトピックの最後に元のトピックを渡すことができます。たとえば、前のセクションの風力発電施設プロパティエイリアスの例で基本的な取り込みを使用するには、次のトピックにメッセージを送信できます。

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

> **Note**
>
> 上記の例には、ルールアクションに表示されるトピックから基本的な取り込みプレフィックス (//) AWS IoT が削除されるため、2 番目のスラッシュ (`$aws/rules/rule-name/`) が含まれています。この例では、ルールはトピック `/company/windfarm/3/turbine/7/temperature` を受信します。

詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Reducing messaging costs with basic ingest] (ベーシックインジェストでメッセージングコストを削減) を参照してください。

AWS IoT SiteWise から にデータを取り込む AWS IoT Events

を使用すると AWS IoT Events、 AWS クラウドで IoT フリートの複雑なイベントモニタリングアプリケーションを構築できます。の IoT SiteWise アクションを使用して AWS IoT Events 、イベントが発生した AWS IoT SiteWise ときに のアセットプロパティにデータを送信します。

> **Note**
>
> サポート終了通知: 2026 年 5 月 20 日に、 AWS は のサポートを終了します AWS IoT Events。2026 年 5 月 20 日以降、 AWS IoT Events コンソールまたは AWS IoT Events リソースにアクセスできなくなります。詳細については、AWS IoT Events 「サポート終了」を参照してください。

AWS IoT Events は、 AWS クラウド内の IoT デバイスとシステムのイベントモニタリングアプリケーションの開発を効率化するように設計されています。を使用すると AWS IoT Events、次のことができます。

- IoT フリート全体の変更、異常、または特定の条件を検出して対応します。
- 運用効率を向上させ、IoT エコシステムのプロアクティブ管理を可能にします。

AWS IoT SiteWise アクション AWS IoT SiteWise を通じて と統合することで、 はその機能 AWS IoT Events を拡張し、特定のイベント AWS IoT SiteWise に応じて でアセットプロパティを自動的に更新できます。このインタラクションにより、データの取り込みと管理を簡素化できます。実用的なインサイトを活用することもできます。

詳細については、[AWS IoT Events Developer Guide] (デベロッパーガイド) の次のトピックをご覧ください。

- とは AWS IoT Events
- AWS IoT Events のアクション
- IoT SiteWise アクション

で AWS IoT Greengrass ストリームマネージャーを使用する AWS IoT SiteWise

AWS IoT Greengrass ストリームマネージャーは、ローカルソースから AWS クラウドへのデータストリームの転送を容易にする統合機能です。これは、データフローを管理する中間レイヤーとして機能し、エッジで動作しているデバイスが、送信前にデータを収集して保存し AWS IoT SiteWise、さらなる分析と処理を行うことができます。

AWS IoT SiteWise コンソールでローカルソースを設定することにより、データ送信先を追加します。カスタム AWS IoT Greengrass ソリューションでストリームマネージャーを使用して、 にデータを取り込むこともできます AWS IoT SiteWise。

> **Note**
>
> OPC UA ソースからデータを取り込むには、 で実行される AWS IoT SiteWise Edge ゲートウェイを設定します AWS IoT Greengrass。詳細については、「AWS IoT SiteWise Edge ゲートウェイを使用する」を参照してください。

ローカルソースデータの保存先を設定する方法については、「AWS IoT SiteWise Edge の送信先を理解する」を参照してください。

カスタム AWS IoT Greengrass ソリューションでストリームマネージャーを使用してデータを取り込む方法の詳細については、「AWS IoT Greengrass Version 2 デベロッパーガイド」の次のトピックを参照してください。

- とは AWS IoT Greengrass
- [Manage data streams on the AWS IoT Greengrass core] (コアでのデータストリームの管理)。
- AWS IoT SiteWise アセットプロパティへのデータのエクスポート

AWS IoT SiteWise Edge ゲートウェイを使用する

AWS IoT SiteWise Edge は、クラウド機能を産業用エッジ環境に拡張し、ローカルデータ処理、分析、意思決定を可能にします。SiteWise Edge は、 AWS IoT SiteWise およびその他の AWS サービスと統合して、包括的な産業用 IoT ソリューションを提供します。ゲートウェイは、産業機器と の間の仲介として機能します AWS IoT SiteWise。

SiteWise Edge ゲートウェイは、次の 2 つの異なるデプロイターゲットで実行されます。

- AWS IoT Greengrass V2
- Siemens Industrial Edge

SiteWise Edge ゲートウェイを使用して、エッジでデータを収集し、クラウドに発行できます。で実行されているゲートウェイの場合 AWS IoT Greengrass、アセットモデルとアセットを使用してエッジでデータを処理することもできます。

の AWS IoT SiteWise Edge アプリケーションSiemens Industrial Edgeは、産業機器と の統合をサポート AWS IoT SiteWise しているため、洗練されたデータを AWS クラウドに送信する前に、raw マシンデータを集約して処理し、ローカルで分析を実行できます。

SiteWise Edge ゲートウェイの主な概念

SiteWise Edge には、産業環境でのエッジコンピューティングに便利な機能がいくつかあります。

ローカルデータの収集と処理

OPC-UA や MQTT などのプロトコルを使用して、産業アセットからのデータ収集をサポートします。ゲートウェイは AWS IoT Greengrass Core デバイスまたは で実行されますSiemens Industrial Edge。

オフラインオペレーション

インターネットの停止時にデータの収集と処理を継続し、接続が復元されたときにクラウドと同期します。

AWS IoT Greengrass コンポーネントを使用したエッジコンピューティング

IoT SiteWise パブリッシャーを使用して、ローカル変換と計算のためにデータをクラウドと AWS IoT SiteWise プロセッサに転送します。パブリッシャーとプロセッサはどちらも AWS IoT

Greengrass V2 コンポーネントです。 AWS IoT Greengrass コンポーネントの詳細については、AWS「 が提供するコンポーネント」を参照してください。

クラウド機能を拡張 AWS IoT SiteWise するための との統合

AWS IoT SiteWise クラウド機能と連携して、アセットモデル、分析、ダッシュボードをエッジに拡張します。

データ処理パックが有効になっているゲートウェイの場合、 AWS OpsHub で SiteWise Edge ゲートウェイ AWS IoT SiteWise を一元管理できます。 はリモート管理およびモニタリング機能 AWS OpsHub を提供します。詳細については、「AWS OpsHub for を使用して SiteWise Edge ゲートウェイを管理する AWS IoT SiteWise」を参照してください。

パートナーデータソースの統合

パートナーデータソースをゲートウェイに接続し、SiteWise Edge ゲートウェイと AWS クラウドのパートナーからデータを受信します。詳細については、「SiteWise Edge ゲートウェイのパートナーデータソース」を参照してください。

エッジでのローカルビジュアライゼーション

エッジでリアルタイムのインサイトを得るためのカスタムダッシュボードを提供します。

ローカルデバイスで SiteWise Monitor ポータルを使用して、施設内のデータをローカルでモニタリングします。詳細については、「エッジで AWS IoT SiteWise ポータルを有効にする」を参照してください。

SiteWise Edge を実装するメリット

SiteWise Edge には、産業オペレーションと意思決定プロセスを大幅に改善できる多くの利点があります。

- クラウド処理の遅延のないリアルタイムの運用上のインサイト
- 切断された環境での運用継続性
- エッジ前処理による帯域幅とストレージコストの削減
- ローカルでデータ駆動型の意思決定を行う機能による信頼性の向上

を使用して AWS IoT SiteWise Edge ゲートウェイをセルフホストする AWS IoT Greengrass V2

産業機器からデータをクラウドに送信する前にローカルで収集、処理、視覚化するように AWS IoT SiteWise Edge を設定します。を使用したセルフホスト AWS IoT Greengrass Version 2。

AWS IoT SiteWise Edge ゲートウェイは、産業機器と の間の仲介として機能します AWS IoT SiteWise。SiteWise Edge ゲートウェイは AWS IoT Greengrass Version 2、オンプレミスでのデータ収集と処理をサポートします。データ処理パックを有効にして AWS OpsHub インストールしたローカルデバイスの SiteWise Monitor ポータルを使用して、施設内のデータをローカルでモニタリングします。

セルフホスト型ゲートウェイには 2 つのタイプがあります。

MQTT 対応、V3 ゲートウェイ

MQTT 対応の V3 ゲートウェイアーキテクチャは、データ取り込み機能を改善します。MQTT プロトコルを使用して効率的なデータ通信を行い、設定可能なデータ送信先を提供します。これには、Amazon S3 を使用したバッファされたデータ取り込みのオプションと、リアルタイムのデータ取り込みのオプションが含まれます。パスフィルターを実装して特定の MQTT トピックをサブスクライブし、ターゲットを絞ったデータ収集を有効にできます。MQTT 対応 V3 ゲートウェイは Data Processing Pack 機能をサポートしていないことに注意してください。詳細については、「MQTT 対応、 AWS IoT SiteWise Edge 用 V3 ゲートウェイ」を参照してください。

クラシックストリーム、V2 ゲートウェイ

Classic ストリームである V2 ゲートウェイは、従来の AWS IoT SiteWise Edge ゲートウェイアーキテクチャを表します。既存の SiteWise Edge デプロイと、確立されたワークフローに慣れたユーザーに適しています。Classic ストリーム、V2 ゲートウェイはデータ処理パックをサポートしていますが、データ処理パックによって生成されたデータは Amazon S3 を介して取り込むことができないことに注意してください。既存のデプロイとの互換性を維持する必要がある場合、またはデータ処理パックの機能が必要な場合は、 Classic ストリーム、V2 ゲートウェイを使用します。詳細については、「AWS IoT SiteWise Edge のクラシックストリーム、V2 ゲートウェイ」を参照してください。

トピック

- AWS IoT SiteWise Edge セルフホストゲートウェイの要件
- セルフホスト SiteWise Edge ゲートウェイを作成する

- ローカルデバイスに AWS IoT SiteWise Edge ゲートウェイソフトウェアをインストールする
- MQTT 対応、 AWS IoT SiteWise Edge 用 V3 ゲートウェイ
- AWS IoT SiteWise Edge のクラシックストリーム、V2 ゲートウェイ
- AWS IoT SiteWise Edge ゲートウェイにデータソースを追加する
- AWS IoT GreengrassAWS IoT SiteWise Edge の コンポーネント
- SiteWise Edge ゲートウェイでアセットをフィルタリングする
- AWS IoT SiteWise Edge のプロキシサポートを設定し、信頼ストアを管理する
- エッジで AWS IoT SiteWise APIs を使用する

AWS IoT SiteWise Edge セルフホストゲートウェイの要件

AWS IoT SiteWise エッジゲートウェイは、オンプレミスでのデータ収集、処理、公開をサポートする AWS IoT Greengrass コンポーネントのセット AWS IoT Greengrass V2 として で実行されます。で実行される SiteWise Edge ゲートウェイを設定するには AWS IoT Greengrass V2、 にゲートウェイを作成し、SiteWise Edge ゲートウェイソフトウェア AWS クラウド を実行してローカルデバイスをセットアップします。を使用して SiteWise Edge ゲートウェイ AWS Management Console を作成すると、インストールスクリプトが提供されます。ターゲットゲートウェイデバイスでこのスクリプトを実行して、必要なソフトウェアと依存関係を設定します。

ローカルデバイスの要件

ローカルデバイスが SiteWise Edge ゲートウェイソフトウェアをインストールして実行するには、次の要件を満たしている必要があります。

- AWS IoT Greengrass V2 Core ソフトウェアバージョン v2.3.0 以降をサポートします。詳細については、[AWS IoT Greengrass Version 2 Developer Guide] (デベロッパーガイド) の[Requirements] (必要条件) を参照してください。
- 対応プラットフォームは次のいずれかです。
  - OS: Ubuntu 20.04 以降

    アーキテクチャ: x86_64 (AMD64) または ARMv8 (Aarch64)
  - OS: Red Hat Enterprise Linux (RHEL) 8

    アーキテクチャ: x86_64 (AMD64) または ARMv8 (Aarch64)
  - OS: Amazon Linux 2

アーキテクチャ: x86_64 (AMD64) または ARMv8 (Aarch64)

- OS: Debian 11

  アーキテクチャ: x86_64 (AMD64) または ARMv8 (Aarch64)

- OS: Windows Server 2019 以降

  Architecture: x86_64 (AMD64)

> **Note**
> ARM プラットフォームは、データ収集パックのみを使用した SiteWise Edge ゲートウェイをサポートします。データ処理パックはサポートされていません。

- RAM 4 GB 以上。
- SiteWise Edge ゲートウェイソフトウェア用に最低 10 GB のディスクスペースがあること。
- 適切なポートにアクセスできるようにローカルデバイスを設定します。必要なアウトバウンドサービスエンドポイントの完全なリストについては、AWS IoT SiteWise 「 Edge ゲートウェイに必要なサービスエンドポイント」を参照してください。
- Java ランタイム環境 (JRE) バージョン 11 以降。デバイスの PATH環境変数で Java が利用可能である必要があります。Java を使用してカスタムコンポーネントを開発するには、Java Development Kit (JDK) をインストールする必要があります。Amazon Corretto または OpenJDK を使用することをお勧めします。

ローカルデバイスの許可リストを作成する Amazon S3 バケット

次の Amazon S3 バケットへのファイアウォールアクセスを提供するようにローカルデバイスを設定します。デバイスのそれぞれのリージョンに基づいてアクセスを設定します。

リージョン	エンドポイント
アジアパシフィック (東京)	https://iot-sitewise-gateway-ap-northeast-1-785558802005.s3.ap-northeast-1.amazonaws.com
アジアパシフィック (ソウル)	https://iot-sitewise-gateway-ap-northeast-2-310055672453.s3.ap-northeast-2.amazonaws.com

リージョン	エンドポイント
アジアパシフィック (ムンバイ)	https://iot-sitewise-gateway-ap-south-1-677656657204.s3.ap-south-1.amazonaws.com
アジアパシフィック (シンガポール)	https://iot-sitewise-gateway-ap-southeast-1-475191558554.s3.ap-southeast-1.amazonaws.com
アジアパシフィック (シドニー)	https://iot-sitewise-gateway-ap-southeast-2-396319432685.s3.ap-southeast-2.amazonaws.com
カナダ (中部)	https://iot-sitewise-gateway-ca-central-1-842060018567.s3.ca-central-1.amazonaws.com
中国 (北京)	https://iot-sitewise-gateway-cn-north-1-237124890262.s3.cn-north-1.amazonaws.com.cn
欧州 (フランクフルト)	https://iot-sitewise-gateway-eu-central-1-748875242063.s3.eu-central-1.amazonaws.com
欧州 (アイルランド)	https://iot-sitewise-gateway-eu-west-1-383414315062.s3.eu-west-1.amazonaws.com
米国東部 (バージニア北部)	https://iot-sitewise-gateway-us-east-1-223558168232.s3.us-east-1.amazonaws.com および https://iot-sitewise-gateway-us-east-1-223558168232.s3.amazonaws.com/
米国東部 (オハイオ)	https://iot-sitewise-gateway-us-east-2-005072661813.s3.us-east-2.amazonaws.com
AWS GovCloud (米国西部)	https://iot-sitewise-gateway-us-gov-west-1-599984565679.s3.us-gov-west-1.amazonaws.com/
米国西部 (オレゴン)	https://iot-sitewise-gateway-us-west-2-502577205460.s3.us-west-2.amazonaws.com

データ処理パックの要件

- エッジで データ処理パックを使用する予定の場合 AWS IoT SiteWise、ローカルデバイスも次の要件を満たしている必要があります。
  - x86 64 bit クアッドコアプロセッサ搭載。
  - 16 GB 以上の RAM があること。
  - を使用している場合、RAM に 32 GB 以上が必要ですMicrosoft Windows。
  - 256 GB 以上の空きディスク容量があること。
  - ローカルデバイスは、ポート 443 のネットワークインバウンドトラフィックを許可する必要があります。
  - 次のポートは、 で使用するために予約されています AWS IoT SiteWise。80、443、3001、4569、4572、8000、8081、8082、8084、8085、8445、8086、9000、95 および 50010。トラフィック用の予約ポートを使用すると、接続が切断されることがあります。

    > **Note**
    >
    > AWS IoT Greengrass V2 ストリームマネージャーコンポーネントには独自の要件があります。詳細については、「AWS IoT Greengrass Version 2 デベロッパーガイド」の「設定」を参照してください。

- 最小ディスク容量と計算能力の要件は、実装とユースケースに固有のさまざまな要因によって異なります。
  - 断続的なインターネット接続のためにデータをキャッシュするために必要なディスク容量は、次の要因によって異なります。
    - アップロードされたデータストリームの数
    - 1 秒あたりのデータストリームごとのデータポイント
    - 各データポイントのサイズ
    - 通信速度
    - 予想されるネットワークのダウンタイム
  - データのポーリングとアップロードに必要なコンピューティング性能は、次の要因によって異なります。
    - アップロードされたデータストリームの数
    - 1 秒あたりのデータストリームごとのデータポイント

SiteWise Edge ゲートウェイを使用するためのアクセス許可を設定する

SiteWise Edge ゲートウェイを使用するには、次のアクセス許可が必要です。

> **Note**
>
> AWS IoT SiteWise コンソールを使用して SiteWise Edge ゲートウェイを作成すると、これらのアクセス許可が追加されます。

- SiteWise Edge ゲートウェイの IAM ロールでは、 AWS IoT Greengrass V2 デバイスで SiteWise Edge ゲートウェイを使用してアセットモデルデータとアセットデータを処理できる必要があります。

  このロールを担うことで、次のサービスを利用することができます: `credentials.iot.amazonaws.com`。

  アクセス許可の詳細

  ロールには、以下のアクセス許可が必要です。

  - `iotsitewise` - プリンシパルがエッジでアセットモデルデータとアセットデータを取得できるようにする。
  - `iot` – AWS IoT Greengrass V2 デバイスが とやり取りできるようにします AWS IoT。
  - `logs` – AWS IoT Greengrass V2 デバイスが Amazon CloudWatch Logs にログを送信できるようにします。
  - `s3` – AWS IoT Greengrass V2 デバイスが Amazon S3 からカスタムコンポーネントアーティファクトをダウンロードできるようにします。

  JSON

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

セルフホスト SiteWise Edge ゲートウェイを作成する

AWS IoT SiteWise コンソールまたは AWS CLI を使用して、セルフホスト SiteWise Edge ゲートウェイを作成します。この手順では、ユーザー独自のハードウェアにインストールするセルフホスト SiteWise Edge ゲートウェイを作成する方法について説明します。Siemens Industrial Edge で実行される SiteWise Edge ゲートウェイの作成については、「Siemens Industrial Edge で SiteWise Edge ゲートウェイをホストする」を参照してください。

SiteWise Edge ゲートウェイを作成する

Console

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. [Create gateway (ゲートウェイの作成)] を選択します。
4. 「デプロイターゲットの選択」で、セルフホストゲートウェイを選択します。

5. MQTT 対応、V3 ゲートウェイ、または Classic ストリーム、V2 ゲートウェイのいずれかを選択します。各オプションの詳細については、「」を参照してください[を使用して AWS IoT SiteWise Edge ゲートウェイをセルフホストする AWS IoT Greengrass V2](#)。MQTT 対応の V3 ゲートウェイは、将来に備えた機能として推奨されます。
6. ゲートウェイ設定セクションで、SiteWise Edge ゲートウェイの名前を入力するか、 によって生成された名前を使用します AWS IoT SiteWise。
7. [Greengrass デバイス OS] で、この SiteWise Edge ゲートウェイをインストールするデバイスのオペレーティングシステムを選択します。

> **Note**
>
> データ処理パックは x86 プラットフォームでのみ使用できます。Classic ストリーム、V2 ゲートウェイでのみ使用できます。

8. (オプション) エッジでデータを処理および整理するには、[エッジ機能] で [データ処理パック] を選択します。

> **Note**
>
> 企業ディレクトリ内のユーザーグループにこの SiteWise Edge ゲートウェイへのアクセスを許可するには、「[SiteWise Edge でエッジ機能を設定する](#)」を参照してください。

9. (オプション) [高度な設定] で、以下を実行します。
   - [Greengrass core device] (Greengrass コアデバイス) の場合、次のいずれかを選択します。
     - デフォルト設定 – AWS は自動的にデフォルト設定を使用して Greengrass コアデバイスを作成します AWS IoT Greengrass V2。
       1. Greengrass コアデバイスの名前を入力するか、 によって生成された名前を使用します AWS IoT SiteWise。
     - [高度な設定] - 既存の Greengrass コアデバイスを使用するか、手動で作成する場合は、このオプションを選択します。
       1. Greengrass コアデバイスを選択するか、Greengrass コアデバイスの作成を選択して AWS IoT Greengrass V2 コンソールで作成します。詳細については、AWS IoT

Greengrass Version 2 デベロッパーガイドの「AWS IoT Greengrass V2 コアデバイスの設定」を参照してください。

10. [Create gateway (ゲートウェイの作成)] を選択します。
11. [SiteWise Edge ゲートウェイインストーラーを生成] ダイアログボックスで、[生成してダウンロード] を選択します。 AWS IoT SiteWise は、ローカルデバイスの設定に使用できるインストーラーを自動的に生成します。

> **Important**
>
> このファイルを再生成することはできません。インストーラーファイルは、後で使用るするため、必ず安全な場所に保存してください。

AWS CLI

を使用してセルフホスト型ゲートウェイを作成するには AWS CLI、ゲートウェイの名前を指定し、プラットフォームとゲートウェイバージョンを指定します。ゲートウェイの作成時に指定できるオプションは他にも多数あります。詳細については、「 の AWS CLI コマンドリファレンス」の「create-gateway」を参照してください。 AWS IoT SiteWise

この例を使用するには、ユーザー入力プレースホルダーを独自の情報に置き換えます。

```
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` – ゲートウェイの一意の名前。
- `gateway-platform` – ゲートウェイプラットフォームの設定を指定します。セルフホストゲートウェイの場合は、「」と入力します`greengrassV2`。詳細については、AWS CLI「 コマンドリファレンス」の「create-gateway」セクションの「オプション」を参照してください AWS IoT SiteWise。
  - `coreDeviceThingName` – AWS IoT Greengrass V2 コアデバイスの AWS IoT モノの名前。
  - `coreDeviceOperatingSystem` – のコアデバイスのオペレーティングシステム AWS IoT Greengrass V2。オペレーティングシステムの指定は に必須であ

りgateway-version 3、 には適用されませんgateway-version 2。オプションにはLINUX_AARCH64、、LINUX_AMD64、 などがあります。 WINDOWS_AMD64

- gateway-version – ゲートウェイのバージョン。
  - MQTT 対応 V3 ゲートウェイを作成するには、ゲートウェイバージョン3に を使用します。
  - Classic ストリーム V2 ゲートウェイを作成するには、ゲートウェイバージョン2に を使用します。
- cli-input-json – リクエストパラメータを含む JSON ファイル。

SiteWise Edge ゲートウェイ を作成しました[ローカルデバイスに AWS IoT SiteWise Edge ゲートウェイソフトウェアをインストールする](#)。

ローカルデバイスに AWS IoT SiteWise Edge ゲートウェイソフトウェアをインストールする

AWS IoT SiteWise Edge ゲートウェイを作成したら、SiteWise Edge ゲートウェイソフトウェアをローカルデバイスにインストールします。SiteWise Edge ゲートウェイソフトウェアは、Linux またはMicrosoft Windowsサーバーオペレーティングシステムがインストールされているローカルデバイスにインストールできます。

> **Important**
>
> ローカルデバイスがインターネットに接続されていることを確認します。

Linux

次の手順では、SSH を使用してローカルデバイスに接続します。または、USB フラッシュドライブやその他のツールを使用して、インストーラーファイルをローカルデバイスに転送することができます。SSH を使用しない場合は、以下の「ステップ 2: SiteWise Edge ゲートウェイソフトウェアのインストール」に進みます。

SSH の前提条件

SSH を使用してデバイスに接続する前に、以下の前提条件を満たしてください。

- Linux および MacOS - OpenSSH をダウンロードしてインストールします。詳細については、https://www.openssh.com を参照してください。

ステップ 1: インストーラーを SiteWise Edge ゲートウェイデバイスにコピーする

SSH クライアントを使用してローカルデバイスに接続する方法を説明します。

1. デバイスに接続するには、コンピュータのターミナルウィンドウで次のコマンドを実行し、*#####*と *IP* を昇格された権限と IP アドレスを持つユーザー名に置き換えます。

```
ssh username@IP
```

2. AWS IoT SiteWise が生成したインストーラファイルを SiteWise Edge ゲートウェイデバイスに転送するには、次のコマンドを実行します。

> **Note**
>
> - *path-to-saved-installer* は、インストーラーファイルを保存したコンピュータ上のパスとインストーラーファイルの名前に置き換えてください。
> - *IP-address* を、ローカルデバイスの IP アドレスに置き換えます。
> - *directory-to-receive-installer* を、インストーラーファイルの受信に使用するローカルデバイスのパスに置き換えます。

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

ステップ 2: SiteWise Edge ゲートウェイソフトウェアをインストールする

以下の手順では、SiteWise Edge ゲートウェイデバイスのターミナルウィンドウでコマンドを実行します。

1. インストーラーファイルに実行アクセス許可を与える。

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

2. インストーラーを実行します。

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

Windows Server

前提条件

SiteWise Edge ゲートウェイソフトウェアをインストールするには、以下の前提条件が必要です。

- Microsoft Windows Server 2019 以降がインストールされている
- 管理者権限
- PowerShell バージョン 5.1 以降がインストールされている
- SiteWise Edge ゲートウェイインストーラーがプロビジョニング先の Windows Server にダウンロードされている

ステップ 1： 管理者として PowerShell を実行する

1. SiteWise Edge ゲートウェイをインストールする Windows サーバーに、管理者としてログインします。
2. Windows の検索バーに PowerShell と入力します。
3. 検索結果で、Windows PowerShell アプリのコンテンツ (右クリック) メニューを開きます。[管理者として実行] を選択します。

ステップ 2: SiteWise Edge ゲートウェイソフトウェアをインストールする

SiteWise Edge ゲートウェイデバイスのターミナルウィンドウで以下のコマンドを実行します。

1. SiteWise Edge ゲートウェイインストーラーのブロックを解除します。

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

2. インストーラーを実行します。

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

> **Note**
> システム上でスクリプト実行が無効になっている場合は、スクリプト実行ポリシーを `RemoteSigned`に変更します。

```
Set-ExecutionPolicy RemoteSigned
```

次のステップは、必要なセルフホストゲートウェイのタイプによって異なります。MQTT 対応、AWS IoT SiteWise Edge 用 V3 ゲートウェイ または に進みますAWS IoT SiteWise Edge のクラシックストリーム、V2 ゲートウェイ。

MQTT 対応、 AWS IoT SiteWise Edge 用 V3 ゲートウェイ

AWS IoT SiteWise は MQTT 対応の V3 ゲートウェイを使用でき、SiteWise Edge ゲートウェイアーキテクチャの大幅な進歩を表します。このゲートウェイタイプは、データ通信に MQTT (Message Queuing Telemetry Transport) プロトコルを活用し、産業用 IoT デプロイの柔軟性と効率を向上させます。

MQTT 対応 V3 ゲートウェイは、データ転送に MQTT を使用し、デバイスとクラウド間でメッセージを効率的に転送する軽量のパブリッシュ/サブスクライブネットワークプロトコルを可能にします。へのリアルタイムのデータ取り込み AWS IoT SiteWise や Amazon S3 を使用したバッファされたデータ取り込みなど、さまざまなデータ送信先を設定できます。正確なデータ収集を有効にするには、パスフィルターを実装して特定の MQTT トピックをサブスクライブできます。

MQTT 対応 V3 ゲートウェイには、フィルターを「#」(すべてのトピック) に設定して事前設定されたリアルタイム送信先が付属しています。この送信先は、必要に応じてカスタマイズまたは削除できます。データ管理を効率化するために、各ゲートウェイには 1 つのリアルタイム送信先しか存在できません。

MQTT 対応アーキテクチャは、 Classic ストリームである V2 ゲートウェイとは大きく異なります。V2 はストリームベースのアプローチを使用していますが、V3 は MQTT を採用しており、より設定可能なデータ送信先とフィルタリングオプションを提供します。ただし、V3 は V2 で利用可能なデータ処理パックをサポートしていないことに注意してください。

MQTT 対応の V3 ゲートウェイには、いくつかの利点があります。

- MQTT の軽量性によりスケーラビリティが向上し、多数のデバイスや高頻度のデータ転送をより適切に処理できるようになりました。
- パスフィルターによるデータ管理を強化し、データ収集をきめ細かく管理し、不要なデータ転送と処理を削減します。
- 柔軟なデータ処理により、特定のニーズに基づいてリアルタイム処理とバッファストレージ間の設定が可能になります。

- 最新の IoT 通信標準と整合し、将来の機能強化と統合のためのステージを設定します。

特に柔軟なデータ取り込みオプションとデータ収集の正確な制御が必要な場合は、新しいデプロイに MQTT 対応の V3 ゲートウェイを採用することを検討してください。

> **Note**
>
> データ処理パックを必要とする既存のデプロイまたはシナリオでは、Classic ストリーム、V2 ゲートウェイは引き続き実行可能なオプションです。

両方のゲートウェイタイプを提供することで、高度な MQTT 機能を優先するか、既存のシステムとの互換性を優先するかにかかわらず、特定の産業 IoT ニーズに最適なソリューション AWS IoT SiteWise を確実に選択できます。

送信先とパスフィルター

MQTT 対応ゲートウェイの宛先とパスフィルターの詳細については、以下のトピックを参照してください。

- AWS IoT SiteWise Edge の送信先を理解する
- AWS IoT SiteWise Edge リアルタイム送信先を追加する
- Amazon S3 を使用して AWS IoT SiteWise バッファされた送信先を追加する
- AWS IoT SiteWise Edge 送信先のパスフィルターを理解する
- AWS IoT SiteWise Edge 送信先にパスフィルターを追加する
- AWS IoT SiteWise Edge の送信先を管理する

外部アプリケーションを EMQX ブローカーに接続する

このガイドでは、デプロイされた MQTT 対応 V3 AWS IoT SiteWise ゲートウェイの EMQX ブローカーを介して Edge ゲートウェイに外部アプリケーションを接続する方法について説明します。外部アプリケーションには、エッジで産業データとやり取りする必要があるカスタムモニタリングツール、サードパーティーの視覚化ソフトウェア、レガシーシステムなどがあります。

EMQX デプロイ設定、安全な接続のための TLS セットアップ、特定のトピックへのアクセスを制御するための認可ルールなど、Linux と Microsoft Windows環境の両方の設定手順について説明します。

> **Note**
>
> EMQX は AWS IoT SiteWise Edge のベンダーまたはサプライヤーではありません。

> **Important**
>
> ゲートウェイへの接続を保護するには、 AWS IoT Greengrass クライアントデバイス認証機能を使用して証明書ベースの認証を使用することを強くお勧めします。この方法は、相互 TLS (mTLS) 認証を通じて堅牢なセキュリティを提供します。詳細については、 AWS IoT Greengrass Version 2 デベロッパーガイドの「クライアントデバイスをコアデバイスに接続する」を参照してください。

証明書ベースの認証を使用できない場合は、このガイドに従ってユーザー名とパスワードを使用して認証を設定します。

前提条件

- デプロイされオンラインになっている SiteWise Edge MQTT 対応 V3 ゲートウェイ
- ゲートウェイホストへのアクセス
- AWS IoT SiteWise および AWS IoT Greengrass コンソールへのアクセス

トピック

- AWS IoT SiteWise Edge の EMQX ブローカーのメッセージペイロード形式
- EMQX ブローカーを設定する
- Edge の EMQX ブローカーにアプリケーションを接続する AWS IoT SiteWise
- EMQX で AWS IoT SiteWise Edge の認可ルールを設定する

AWS IoT SiteWise Edge の EMQX ブローカーのメッセージペイロード形式

IoT SiteWise パブリッシャーコンポーネントが外部アプリケーションからデータを消費して AWS IoT SiteWise クラウドに公開するには、ブローカーに送信されるペイロードが特定の要件を満たしている必要があります。

AWS IoT SiteWise Edge との MQTT 通信を成功させるには、ペイロード形式を理解することが重要です。接続のセットアッププロセスは後のセクションで説明しますが、実装の計画に役立つペイロード要件を最初に説明します。

MQTT トピックの要件

使用されるレベルや文字の数など、MQTT トピック構造に制限はありません。ただし、トピックはペイロードの `propertyAlias`フィールドと一致することをお勧めします。

Example プロパティエイリアスの例

MQTT トピックが の場合は`site1/line1/compressor1/temperature`、 `propertyAlias` が一致していることを確認します。

```
{
  "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 ペイロード構造

MQTT メッセージペイロードは JSON で記述され、 AWS IoT SiteWise API リファレンスで定義されている`PutAssetPropertyValueEntry`メッセージ形式に従います。

```
{
   "assetId": "string",
   "propertyAlias": "string",
   "propertyId": "string",
```

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

> **Note**
>
> メッセージが有効と見なされるには、次のいずれかの条件のみが該当します。
>
> - `propertyAlias` が設定されている、または
> - `assetId` と の両方`propertyId`が設定されている
>
> `PutAssetPropertyValueEntry` には、このコンテキストでは必須ではない `entryId`フィールドがあります。

EMQX ブローカーを設定する

このセクションでは、ユーザー名とパスワードを追加する方法について説明します。また、追加されたユーザー名とパスワードを使用して外部ソースから TLS 接続を確立する方法についても説明します。Linux または を使用して EMQX ブローカーを設定できますMicrosoft Windows。

> **Note**
>
> ブローカーを設定するには、MQTT 対応 V3 ゲートウェイでデフォルトの EMQX 設定でセットアップされたコアデバイスが必要です。

> **Important**
>
> この手順を完了したら、認可ルールを設定することを強くお勧めします。詳細については、「EMQX で AWS IoT SiteWise Edge の認可ルールを設定する」を参照してください。追加ユーザーの認可ルールは、セキュリティを強化します。

認証用の EMQX デプロイ設定を更新する

認証用に EMQX デプロイ設定を更新するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションで、 Edge セクションの Edge ゲートウェイを選択します。
3. 設定するゲートウェイを選択します。
4. Edge ゲートウェイ設定セクションで、Greengrass コアデバイス値をコピーします。後で使用するために保存します。
5. AWS IoT コンソールを開きます。
6. 左側のナビゲーションの「管理」セクションで、Greengrass デバイスを選択し、「デプロイ」を選択します。
7. 前に保存したコアデバイス値を見つけ、そのリンクを選択してデプロイを開きます。
8. Actions ドロップダウンボタンを選択し、Revise を選択します。
9. 表示されるメッセージを読み、デプロイの修正を選択します。ターゲットの指定ページが表示されます。
10. コンポーネントの設定ステップに到達するまで次へを選択します。
11. ラジオボタン [`aws.greengrass.clientdevices.mqtt.EMQX`] を選択します。
12. コンポーネントの設定ボタンを選択します。コンポーネントの設定ページが表示されます。
13. 設定の更新で、コンポーネントバージョン: 2.*.* のデフォルト設定にリセットを選択します。
14. OS に基づいてマージする設定セクションに次の設定を入力します。

    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` フィールドは Linux ゲートウェイ専用です。

15. [確認] を選択してください。
16. レビューステップに到達するまで次へを選択します。
17. [デプロイ] を選択します。
18. デプロイが成功したら、次のステップに進みます。

ユーザー名とパスワード認証を有効にする

このセクションでは、EMQX ダッシュボード GUI を使用してユーザー名とパスワードを追加する方法について説明します。

> **Note**
> 提供されている EMQX 関連の手順は、参考用です。EMQX のドキュメントと機能は時間の経過とともに変更される可能性があり、そのドキュメントは維持されないため、最新の情報については EMQX の公式ドキュメントを参照することをお勧めします。

EMQX Dashboard

EMQX ダッシュボードを使用してユーザー名とパスワード認証を有効にするには

1. ゲートウェイホスト内にいることを確認します。
2. ブラウザウィンドウを開き、 にアクセスしますhttp://localhost:18083/。
3. のデフォルトのユーザー名**admin**と のデフォルトのパスワードを入力します**public**。詳細については、EMQX ドキュメントの「EMQX ダッシュボード」を参照してください。
4. ログイン後、パスワードを変更するように求められます。EMQX ダッシュボードに進むには、パスワードを更新します。
5. 左側のナビゲーションで、シールドアイコンを選択し、認証を選択します。
6. 組み込みデータベースの行で、ユーザーボタンを選択します。

7. ユーザーを追加するには、プラス記号アイコンボタンを選択します。追加画面が表示されます。
8. 外部アプリケーションのユーザーのユーザー名とパスワードを入力します。
9. [保存] を選択します。選択したユーザー名が認証ページのテーブルに表示されます。

> **Note**
> 既存の承認ルールまたはデフォルトの承認ルールが新しいユーザーに適用されます。これらを確認して、外部アプリケーションのニーズに合わせて調整することをお勧めします。

EMQX Management with Linux

で AWS IoT SiteWise EMQX CLI ツールを使用します`/greengrass/v2/bin/swe-emqx-cli`。

Linux を使用して EMQX Management でユーザー名とパスワード認証を有効にするには

1. 次のコマンドを実行して、管理者パスワードを変更します。

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

2. プロンプトが表示されたら、次の操作を行います。
   1. 現在の管理者ユーザー (デフォルトは `admin`) とパスワード (デフォルトは) を入力します`public`。
   2. 新しいパスワードを入力して確認します。

   成功すると、次のメッセージが表示されます。

```
admin password changed successfully
```

3. 次のコマンドを実行して、外部アプリケーションのユーザーを追加します。

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

4. プロンプトが表示されたら、次の操作を行います。
   1. 新しいユーザーのユーザー名を入力します。

2. 新しいユーザーのパスワードを入力して確認します。

成功すると、次のメッセージが表示されます。

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

5. 次のコマンドを実行して、ユーザー設定を確認します。

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

出力には、設定されたすべてのユーザーが表示されます。

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

Total users: 1
```

EMQX Management with Windows

次のいずれかの場所で AWS IoT SiteWise EMQX CLI ツールを使用します。

- PowerShell: `C:\greengrass\v2\bin\swe-emqx-cli.ps1`
- コマンドプロンプト: `C:\greengrass\v2\bin\swe-emqx-cli.bat`

Windows を使用して EMQX 管理を通じてユーザー名とパスワード認証を有効にするには

1. 次のコマンドを実行して、管理者パスワードを変更します。

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

2. プロンプトが表示されたら、次の操作を行います。

   1. 現在の管理者ユーザー (デフォルトは `admin`) とパスワード (デフォルトは) を入力しますpublic。
   2. 新しいパスワードを入力して確認します。

   成功すると、次のメッセージが表示されます。

```
admin password changed successfully
```

3. 次のコマンドを実行して、外部アプリケーションのユーザーを追加します。

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

4. プロンプトが表示されたら、次の操作を行います。

   1. 新しいユーザーのユーザー名を入力します。
   2. 新しいユーザーのパスワードを入力して確認します。

   成功すると、次のメッセージが表示されます。

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

5. 次のコマンドを実行して、ユーザー設定を確認します。

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

   出力には、設定されたすべてのユーザーが表示されます。

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

Total users: 1
```

Edge の EMQX ブローカーにアプリケーションを接続する AWS IoT SiteWise

EMQX ブローカーは、ポート 8883 の Transport Layer Security (TLS) を使用してすべての通信を暗号化し、転送中にデータを確実に保護します。このセクションでは、アプリケーションと EMQX ブローカー間の接続を確立する手順について説明します。これらのステップに従うことで、産業データの整合性と機密性を維持できます。接続プロセスには、主に 2 つのアプローチがあります。コンポーネントによる自動 IP 検出を使用するか、TLS 証明書のサブジェクト代替名 (SANs) として DNS 名と IP アドレスを手動で設定します。各メソッドには、ネットワークのセットアップとセキュリティ要件に応じて独自の利点があります。このドキュメントでは、両方のオプションについて説明します。

トピック

- AWS IoT SiteWise Edge の EMQX ブローカーへの安全な接続のために TLS を設定する
- AWS IoT SiteWise Edge で EMQX ブローカー接続をテストする
- 独自の CA を使用する
- 外部ファイアウォール接続用のポート 8883 を開く

AWS IoT SiteWise Edge の EMQX ブローカーへの安全な接続のために TLS を設定する

デフォルトでは、 はコアデバイス認証局 (CA) によって署名された EMQX ブローカーの TLS サーバー証明書 AWS IoT Greengrass を生成します。詳細については、「MQTT ブローカーを使用した AWS IoT Greengrass Core デバイスへのクライアントデバイスの接続」を参照してください。

TLS 証明書を取得する

CA 証明書を取得するには、ゲートウェイホストで次のコマンドを実行します。

Linux

ゲートウェイホストのシェルセッションで次のコマンドを実行します。

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

このコマンドは、証明書の場所を表示し、証明書の内容を出力します。

または、次のコマンドを使用して証明書をファイルに保存することもできます。

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

Windows

ゲートウェイホストの PowerShell セッションで次のコマンドを実行します。

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

このコマンドは、証明書の場所を表示し、証明書の内容を出力します。

または、次のコマンドを使用して証明書をファイルに保存することもできます。

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

CLI は、システム上の正確なパスに関係なく、証明書を自動的に見つけます。

ca.pem ファイルの内容を、ブローカーに接続している外部アプリケーションにコピーします。`BrokerCoreDeviceCA.pem` と名前を付けて保存します。

TLS サーバー証明書にカスタム DNS 名/IP アドレスを追加する

によって生成された証明書のサブジェクト代替名 (SAN) AWS IoT Greengrass は ですlocalhost。ゲートウェイホストの外部から TLS 接続を確立すると、ブローカーのホスト名がサーバー証明書localhostの のホスト名と一致しないため、TLS 検証ステップは失敗します。

ホスト名の不一致問題に対処するために、 はコアデバイスエンドポイントを管理する 2 つの方法 AWS IoT Greengrass を提供します。このセクションでは、両方のオプションについて説明します。詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」の「 コアデバイスエンドポイントの管理」を参照してください。

- コアデバイスの IP アドレスを使用して EMQX ブローカーに接続するには、自動 IP 検出セクションを使用します。
- IP アドレスの代わりに DNS 名を使用して EMQX ブローカーに接続するには、手動管理セクションを使用します。

Automated IP discovery

このオプションを使用すると、コアデバイスは IP アドレスを自動的に検出し、サブジェクト代替名 (SAN) としてブローカー証明書に追加できます。

1. `aws.greengrass.clientdevices.IPDetector` コンポーネントをコアデバイスのデプロイに追加します。
2. 変更をデバイスにデプロイする
3. デプロイが完了するまで待ちます。

   デプロイが完了したら、ブローカーの IP アドレスを使用して安全な TLS 接続を確立できます。

   IP アドレスは、ブローカー証明書に SAN として自動的に追加されます。

Manual DNS and IP Configuration

DNS 名と IP アドレスをサブジェクト代替名 (SANs) として TLS 証明書に手動で追加できます。この方法は、ゲートウェイホストの DNS 名を設定している場合に便利です。

> **Important**
>
> IPDetector コンポーネントを使用している場合は、続行する前にデプロイから削除してください。IPDetector コンポーネントは、手動エンドポイント設定を上書きします。

エンドポイントを手動で設定するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションで、「エッジ」セクションの「エッジゲートウェイ」を選択します。
3. 設定するゲートウェイを選択します。
4. Edge ゲートウェイ設定セクションで、Greengrass コアデバイス URL を選択します。コアデバイスのページが表示されます。
5. クライアントデバイスタブを選択します。
6. [Manage endpoints] (エンドポイント管理) を選択します。
7. エンドポイントの管理ダイアログボックスに、DNS 名と SANs として追加する IP アドレスを入力します (複数可) 。ポート 8883 を使用します。
8. [更新] を選択します。

ブローカーの TLS サーバー証明書は自動的に更新され、新しいエンドポイントが含まれます。

Linux を使用して TLS サーバー証明書の更新を確認するには

1. ゲートウェイホストでシェルセッションを開始します。

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

2. コマンドは、次のような出力を返します。

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

3. エンドポイントが SANs のリストに表示されていることを確認します。

Windows を使用して TLS サーバー証明書の更新を確認するには

1. ゲートウェイホストでシェルセッションを開始します。

```
(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. コマンドは、次のような出力を返します。

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

3. 追加したエンドポイントが SANs のリストに含まれていることを確認します。

AWS IoT SiteWise Edge で EMQX ブローカー接続をテストする

TLS 証明書と認証情報を使用して EMQX ブローカーを設定したら、セットアップが正しく機能することを確認することが重要です。接続をテストすることで、セキュリティ設定が適切に実装され、クライアントがブローカーへの暗号化された接続を正常に確立できるようになります。このセクションでは、TLS 暗号化と認証をサポートする広く使用されている MQTT クライアントツールである Mosquitto コマンドラインインターフェイス (CLI) クライアントを使用してブローカー接続をテストする方法について説明します。

Mosquitto CLI クライアントを使用して EMQX ブローカー接続をテストする

このステップでは、mosquitto CLI クライアントを使用してセットアップをテストし、前に作成したユーザー名とパスワードを使用してブローカーに正常に接続できることを確認します。ステップ 3: TLS のセットアップの`BrokerCoreDeviceCA.pem`次のステップを取得するには。

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

> **Note**
>
> 接続先のホスト名/IP アドレスが、クライアントに渡す CA 証明書に記載されているサブジェクト代替名 (SAN) と一致しない場合、SSL:verify エラーが発生することがあります。正しい SAN で証明書を取得する方法については、「ステップ 3: TLS のセットアップ」の「カスタム DNS 名/IP アドレスを TLS サーバー証明書に追加する」を参照してください。

この時点で、すべてのユーザーはブローカー上のすべてのトピックを公開およびサブスクライブできます。EMQX で AWS IoT SiteWise Edge の認可ルールを設定する に進みます。

独自の CA を使用する

AWS IoT Greengrass では、独自の認証機関 (CA) を使用するように独自のクライアントデバイス認証コンポーネントを設定する方法について説明します。クライアントデバイス認証コンポーネント (`aws.greengrass.clientdevices.Auth`) がクライアントデバイスを認証し、クライアントデバイスのアクションを承認します。詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」の「独自の認証機関の使用」を参照してください。

独自の CA を使用するには、デプロイに `aws.greengrass.clientdevices.Auth`コンポーネントを追加して、カスタム設定を指定できます。

外部ファイアウォール接続用のポート 8883 を開く

Linux

Linux ホストファイアウォールルールで、ポート 8883 のインバウンドルールを追加して、ゲートウェイホストの外部からの着信接続を許可します。ファイアウォールがある場合は、ポート 8883 で受信 TLS 接続が許可されていることを確認します。

Windows

Microsoft Windows ホストファイアウォールルールで、ポート 8883 のインバウンドルールを追加して、ゲートウェイホストの外部からの着信接続を許可します。ルールがポート 8883 を指定するポートタイプの許可ルールであることを確認します。これは、外部アプリケーションからブローカーへの接続を許可するようにネットワーク設定に従って設定できます。

EMQX で AWS IoT SiteWise Edge の認可ルールを設定する

EMQX は、ユーザー名、IP アドレス、クライアント ID などの識別子に基づく認可ルールの追加をサポートしています。これは、さまざまなオペレーションやトピックに接続する外部アプリケーションの数を制限する場合に便利です。

トピック

- Linux で組み込みデータベースを使用して認可を設定する
- Windows で組み込みデータベースを使用して認可を設定する
- EMQX デプロイ設定を認可用に更新する
- ユーザー用の EMQX ダッシュボードを使用して認可ルールを追加する

Linux で組み込みデータベースを使用して認可を設定する

認可ルールを設定する場合、デプロイ設定に応じて 2 つの設定選択肢があります。

- Docker – なしで標準Dockerインストールを実行している場合はLitmus Edge、Docker ブリッジゲートウェイ設定を使用します。これは通常、 AWS IoT SiteWise コンポーネントのみをデプロイしている場合に当てはまります。
- Litmus Edge – ゲートウェイに Litmus Edgeをインストールしている場合は、Litmus Edgeネットワークサブネット設定を使用します。

> **Note**
>
> 最初にDockerブリッジゲートウェイを設定し、後で をインストールする場合はLitmus Edge、Litmus Edgeネットワークサブネットオプションを使用して認可ルールを再設定して、すべてのコンポーネント間の適切な通信を確保します。

基本的な認可ルールを追加するには

1. EMQX ブローカーがデプロイされ、実行されていることを確認します。
2. ゲートウェイホストでシェルセッションを開始します。

   Docker without Litmus Edge

   を使用しない標準Dockerインストールの場合はLitmus Edge、以下を実行します。

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

Litmus Edge network subnet

を使用している場合はLitmus Edge、Litmus Edge ネットワークサブネット IP を決定します。

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

出力のサブネット値を書き留めて、次のコマンドを実行します。を前のステップのサブネット値`litmus_subnet_ip`に置き換えます。

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

このツールは、提供された IP アドレスからブローカーへの接続を許可する認可ルールを自動的に作成して適用します。これにより、すべてのトピックへのアクセスが許可されます。これには、IoT SiteWise OPC UA コレクターと IoT SiteWise パブリッシャーが含まれます。

3. EMQX デプロイ設定を認可用に更新する に進みます。

Windows で組み込みデータベースを使用して認可を設定する

このセクションでは、Windows デプロイ用の組み込みデータベースを使用した認可ルールの設定について説明します。

基本的な認可ルールを追加するには

1. EMQX ブローカーがデプロイされ、実行されていることを確認します。
2. AWS IoT SiteWise EMQX CLI ツールを実行します。

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

このツールは、localhost (127.0.0.1) からブローカーへの接続を許可する ACL ルールを自動的に作成して適用します。これにより、すべてのトピックへのアクセスが許可されます。これには、IoT SiteWise OPC UA コレクターと IoT SiteWise パブリッシャーが含まれます。

3. EMQX デプロイ設定を認可用に更新する に進みます。

EMQX デプロイ設定を認可用に更新する

EMQX デプロイ設定を認可用に更新するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションで、「エッジ」セクションの「エッジゲートウェイ」を選択します。
3. 設定するゲートウェイを選択します。
4. Edge ゲートウェイ設定セクションで、Greengrass コアデバイス値をコピーします。後で使用するために保存します。
5. AWS IoT コンソールを開きます。
6. 左側のナビゲーションの「管理」セクションで、Greengrass デバイスを選択し、「デプロイ」を選択します。
7. 前に保存したコアデバイス値を見つけ、そのリンクを選択してデプロイを開きます。
8. Actions ドロップダウンボタンを選択し、Revise を選択します。
9. 表示されるメッセージを読み、デプロイの修正を選択します。ターゲットの指定ページが表示されます。
10. コンポーネントの設定ステップに到達するまで次へを選択します。
11. ラジオボタン [`aws.greengrass.clientdevices.mqtt.EMQX`] を選択します。
12. コンポーネントの設定ボタンを選択します。コンポーネントの設定ページが表示されます。
13. 設定の更新で、コンポーネントバージョン: 2.*.* のデフォルト設定にリセットを選択します。
14. OS に基づいてマージするには、次のコンテンツを 設定 セクションに貼り付けます。

    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
-p 127.0.0.1:18083:18083 -v emqx-data:/opt/emqx/data -e
EMQX_NODE__NAME=emqx@local",
    "requiresPrivilege": "true"
```

```
}
```

Windows

```
{
    "emqxConfig": {
        "authorization": {
            "no_match": "deny",
            "sources": [
                {
                    "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. [確認] を選択してください。
16. レビューステップに到達するまで次へを選択します。
17. [デプロイ] を選択します。

> **Note**
>
> この時点では、ACL ファイルを編集して認可ルールを更新することはできません。または、デプロイが成功ユーザー用の EMQX ダッシュボードを使用して認可ルールを追加するした ら、 に進むこともできます。

ユーザー用の EMQX ダッシュボードを使用して認可ルールを追加する

EMQX ダッシュボードまたは EMQX AWS IoT SiteWise CLI ツールを使用して、認可ルールを追加または更新できます。 AWS IoT SiteWise EMQX CLI ツールは、EMQX の組み込みデータベースを使用して認可を管理します。

> **Note**
>
> 認可ルールの追加は、MQTT トピックパターンとアクセスコントロールの理解を必要とする高度な設定ステップです。EMQX の組み込みデータベースを使用して認可ルールを作成する

方法の詳細については、EMQX ドキュメントの「組み込みデータベースを使用する」を参照してください。

> **Note**
>
> 提供されている EMQX 関連の手順は、参考用です。EMQX のドキュメントと機能は時間の経過とともに変更される可能性があり、そのドキュメントは保持されないため、最新の情報については EMQX の公式ドキュメントを参照することをお勧めします。

EMQX dashboard

この手順では、EMQX ダッシュボードに認可ルールを追加する方法を示します。

EMQX ダッシュボードには、ゲートウェイホスト内からのみアクセスできます。ゲートウェイホストの外部から接続しようとすると、ダッシュボードにアクセスできません。

EMQX ダッシュボードを使用して認可ルールを追加するには

1. ゲートウェイホスト内にいることを確認します。
2. ブラウザウィンドウを開き、 にアクセスしますhttp://localhost:18083/。
3. EMQX ダッシュボードにログインします。この手順では、デフォルトのログイン認証情報が任意のものに変更されていることを前提としています。初期設定の詳細については、「」を参照してくださいユーザー名とパスワード認証を有効にする。
4. シールドアイコンを選択し、ドロップダウンメニューから認可を選択します。
5. 組み込みデータベース行のアクセス許可ボタンを選択します。
6. 組み込みデータベース認可セクションで、ビジネスニーズに合わせてユーザー認可ルールを追加または更新します。ルールの作成に関する詳細なガイダンスについては、EMQX ドキュメントの「組み込みデータベースの使用」セクションを参照してください。

AWS IoT SiteWise CLI tool using Linux

Linux の EMQX AWS IoT SiteWise CLI ツールを使用して認可ルールを管理するには：

- 次の形式を使用して、ユーザーの認可ルールを追加します。

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

Example ユーザーの承認ルールを追加する

この例では、 という名前のユーザーのルールを追加する方法を示しますsystem1。

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

Example : ユーザーの承認ルールを表示する

ユーザーの承認ルールを表示するにはsystem1、次のコマンドを実行します。

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

Example : 既存の承認ルールをすべて表示する

現在使用しているすべての認可ルールを表示するには、次のコマンドを実行します。

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

Example : ユーザーのすべての認可ルールを削除する

特定のユーザーに適用されるすべての認可ルールを削除するには、次のコマンドを実行します。

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

削除を確認するプロンプトが表示されます。

AWS IoT SiteWise CLI tool using Windows

で EMQX AWS IoT SiteWise CLI ツールを使用して認可ルールを管理するにはWindows PowerShell：

- 次の形式を使用して、ユーザーの認可ルールを追加します。

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

Example : ユーザーの承認ルールを追加する

この例では、 という名前のユーザーのルールを追加する方法を示しますsystem1。

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

Example : ユーザーの承認ルールを表示する

ユーザーの承認ルールを表示するにはsystem1、次のコマンドを実行します。

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

Example : 既存の承認ルールをすべて表示する

現在使用しているすべての認可ルールを表示するには、次のコマンドを実行します。

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

Example : ユーザーのすべての認可ルールを削除する

特定のユーザーに適用されるすべての認可ルールを削除するには、次のコマンドを実行します。

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

削除を確認するプロンプトが表示されます。

SiteWise Edge とオープンソースツールを使用してデータを処理および視覚化する

ローカル処理と視覚化用のオープンソースツールを使用して AWS IoT SiteWise Edge MQTT 対応ゲートウェイを設定し、産業用データ管理機能を強化します。

SiteWise Edge を使用すると、外部のオープンソースツールを使用してローカルデータ処理パイプラインを作成できます。Node-RED® を使用して InfluxDBで時系列データを保存し、Grafana® ダッシュボードを使用してオペレーションをモニタリングします。

Node-RED はデータフローを処理して変換し、InfluxDB は時系列データストレージを提供します。Grafana は、リアルタイムの運用データを表示します。SiteWise Edge でこれらのツールを使用して、ローカル環境と の間でデータを同期し AWS クラウド、即時のローカルインサイトと長期的なクラウドベースの分析機能の両方を提供します。

> **Note**
>
> Node-RED®、InfluxDB®、Grafana® は SiteWise Edge のベンダーまたはサプライヤーではありません。

> **Note**
>
> このガイドでは、Amazon Managed Grafana サービスではなくSiteWise Edge 用のオープンソースバージョンの Grafana を使用しています。 https://docs.aws.amazon.com/grafana/latest/userguide/what-is-Amazon-Managed-Service-Grafana.html

デプロイオプション

このソリューションは、2 つのアプローチのいずれかを使用してデプロイできます。Microsoft Windows 手動セットアップでは、コンポーネントの設定とインフラストラクチャとの統合を制御します。ではLinux、Docker を使用して事前設定されたコンポーネントをコンテナにデプロイできます。

運用要件を満たす方法を選択します。

- オープンソース統合を手動でセットアップする (Windows) – カスタム設定または既存のインフラストラクチャの場合
- Docker とのオープンソース統合を設定する (Linux) – 事前設定されたコンポーネントによる迅速なデプロイ用

風力発電所の例の概要

このガイドでは、風力発電所の例を使用して、風力発電所のタービンの風速をモニタリングする方法を示します。この実用的なシナリオは、ローカルとクラウドベースの可視性の両方が運用効率にとって価値がある一般的な産業モニタリングのニーズを示しています。

この統合により、次のことが可能になります。

- AWS IoT SiteWise Edge ゲートウェイを使用して産業機器からデータを収集する
- Node-RED、InfluxDBを使用してデータをローカルで処理する
- InfluxDB を使用してローカルにデータを保存する
- Grafana ダッシュボードを使用してデータをリアルタイムでモニタリングする

このガイドでは、風力発電所の例を使用します。Node-RED を使用して、風速データを生成するタービンをシミュレートします。Node-RED はデータペイロードを変換し、データを SiteWise Edge MQTT ブローカーに発行し、ブローカーからデータを受信するようにサブスクライブして、その

データを InfluxDB にローカルに保存します。このアプローチにより、すべての運用データがローカルですぐにアクセスでき、クラウドでさらに分析できるようになります。このパターンを実装することで、 で高度な分析を実行する能力を維持しながら、ネットワークの中断に対する回復力を得ることができます AWS クラウド。Grafana はローカルモニタリングのために InfluxDB に接続し、クラウド依存関係のないメトリクスをリアルタイムで可視化します。SiteWise Edge MQTT 対応ゲートウェイは、同じ MQTT ブローカーに接続して にデータを送信し AWS IoT SiteWise、エッジオペレーションとクラウドベースのサービス間のブリッジを作成します。

独自のデータと設定を使用して、製造機器、ユーティリティインフラストラクチャ、またはその他の産業資産をモニタリングしているかどうかにかかわらず、特定の産業要件に合わせた同様のワークフローを作成できます。

オープンソース統合の要件

SiteWise Edge とオープンソースの統合を実装する前に、環境が必要な要件を満たしていることを確認してください。

- ハードウェア要件 - ゲートウェイハードウェアは SiteWise Edge ゲートウェイの要件を満たしている必要があります。詳細については、MQTT 対応、V3 ゲートウェイ、および AWS IoT SiteWise Edge セルフホストゲートウェイの要件の「」を参照してくださいAWS IoT SiteWise Edge アプリケーションの要件。

  > **Important**
  > 追加のオープンソースコンポーネントをデプロイする場合は、ハードウェアが InfluxDB、Node-RED、Grafana の要件を満たしていることを確認してください。

- ネットワーク設定は、コンポーネント間のローカル通信と SiteWise Edge のクラウド接続の両方をサポートしている必要があります。
- すべてのサービスは同じホストで実行する必要があります。

セキュリティに関する考慮事項

特に非ローカルネットワークからインターフェイスにアクセスする場合は、コンポーネント間のすべての通信を暗号化することをお勧めします。各コンポーネントに適切なアクセスコントロールを実装し、 AWS IoT SiteWise Edge ゲートウェイ設定と AWS アカウントセキュリティの AWS ベストプラクティスに従います。

開発環境

このガイドでは、ゲートウェイホストでローカルで実行およびアクセスされる Node-RED、InfluxDB、Grafana について説明します。外部アクセスを必要とする本番環境のデプロイでは、TLS 暗号化、認証、認可などのセキュリティ対策を実装します。各アプリケーションのセキュリティのベストプラクティスに従ってください。

サードパーティー製ソフトウェア

このソリューションでは、InfluxDB AWS、Node-RED、Grafana、`node-red-contrib-influxdb`プラグインなど、 が保守していないサードパーティーソフトウェアを使用します。デプロイする前に、これらのコンポーネントが組織のセキュリティ要件、コンプライアンス標準、ガバナンスポリシーに準拠していることを確認してください。

> **Important**
>
> このガイドでは、 が所有または保守していないサードパーティーソフトウェアを参照し、使用します AWS。実装する前に、すべてのコンポーネントがセキュリティ、コンプライアンス、ガバナンスの要件を満たしていることを確認してください。すべてのソフトウェアを最新のセキュリティパッチで更新し、エッジデプロイを保護するためのベストプラクティスに従ってください。
> InfluxDB、Node-RED、Grafana は SiteWise Edge のベンダーまたはサプライヤーではありません。

その他の考慮事項

SiteWise Edge とのオープンソース統合を実装するときは、以下の追加要因を考慮してください。

- すべてのサービス、ツール、コンポーネントの最新バージョンを使用します。
- クラウド送信の前にローカルでデータをフィルタリングして集約し、 AWS IoT SiteWise データ取り込みコストを削減します。InfluxDB で適切なデータ保持期間を設定し、ゲートウェイハードウェアのサイズを適切に設定します。詳細については、AWS IoT SiteWise 料金表を参照してください。
- すべてのデータに定期的なバックアップ手順を実装します。
- ゲートウェイのリソース使用状況をモニタリングし、コンポーネントごとに適切なリソース制限を設定します。InfluxDB にデータ保持ポリシーを実装して、ディスク使用量を管理します。

オープンソース統合を手動でセットアップする (Windows)

このガイドを使用して、Grafana® および Node-RED® に接続する風速データの時系列バケットを手動で作成します。

デプロイ設定を制御するMicrosoft Windowsために、 に Node-RED、InfluxDB®、Grafana を手動でインストールして設定します。InfluxDB を使用して、デバイスから時系列データを保存および管理できます。

手動セットアップの前提条件

開始する前に、以下の要件を満たす必要があります。

> **Note**
>
> 同じホストですべてのサービス (SiteWise Edge、InfluxDB、Node-RED、Grafana) を実行します。

- MQTT 対応 V3 ゲートウェイをインストールします。詳細については、「MQTT 対応、 AWS IoT SiteWise Edge 用 V3 ゲートウェイ」を参照してください。
- これらのサービスをローカルでインストールして実行します。
  - InfluxDB OSS v2。インストール手順については、「InfluxDB のインストール」を参照してください。
  - Node-RED。インストール手順については、「ローカルで Node-RED をインストールする」を参照してください。
  - Grafana。インストール手順については、「Grafana のインストール」を参照してください。

InfluxDB でローカルストレージをセットアップする

InfluxDB® を使用すると、デバイスからの時系列データをローカルに保存できます。ローカルストレージ機能の目的は、ネットワークの中断時に運用上の可視性を維持し、タイムクリティカルなアプリケーションのレイテンシーを減らすことです。データをクラウドに選択的に転送するオプションを維持しながら、エッジで分析と視覚化を実行できます。

このセクションでは、タービン風速データ用の時系列バケットを作成し、Grafana® および Node-RED® 接続用の API トークンを生成します。InfluxDB バケットは、従来のシステムのデータベースと同様に、時系列データの専用ストレージコンテナとして機能します。API トークンを使用すると、データへの安全なプログラムによるアクセスが可能になります。

InfluxDB をセットアップするには

1. 前提条件のステップを完了し、すべてのツールが同じホストで実行されていることを確認したら、ウェブブラウザを開き、http://127.0.0.1:8086 にアクセスします。
2. (オプション) セキュリティを強化するために TLS 暗号化を有効にします。詳細については、『』の「TLS 暗号化を有効にする」を参照してくださいInfluxData Documentation。
3. Node-RED からのデータを保存する時系列 InfluxDB バケットを作成します。バケットは風力発電所データの専用コンテナとして機能し、このデータセットに固有の保持ポリシーを整理および管理できます。詳細については、『』の「バケットの管理」を参照してくださいInfluxData Documentation。
4. (オプション) エッジロケーションのデータ保持期間を設定します。適切な保持期間を設定すると、ローカルオペレーションに不要になった古いデータが自動的に削除されるため、ストレージリソースを効率的に管理できます。

   データ保持の詳細については、の「InfluxDB のデータ保持」を参照してくださいInfluxData Documentation。
5. バケットの API トークンを生成します。このトークンにより、InfluxDB と Node-RED や Grafana などの他のコンポーネント間の安全な通信が可能になります。これにより、認可されたサービスのみがデータストアとの間で読み取りまたは書き込みを行うことができます。詳細については、『』の「トークンの作成」を参照してくださいInfluxData Documentation。

これらのステップを完了すると、時系列データを InfluxDB インスタンスに保存し、エッジ環境でのローカルデータの永続化と分析の基盤を提供できます。

AWS IoT SiteWise データ統合用に Node-RED フローを設定する

Node-RED® を使用すると、2 つのフローを実装して、デバイスと 間のデータを管理できます AWS IoT SiteWise。これらのフローは連携して、ローカルデータフローとクラウドデータフローの両方に対応する包括的なデータ管理ソリューションを作成します。

- データ発行フロー – クラウドに発行します。データ発行フローはデータを に送信します AWS IoT SiteWise。このフローは、センサーデータを生成し、 AWS IoT SiteWise フォーマットに変換し、SiteWise Edge MQTT ブローカーに発行することで、タービンデバイスをシミュレートします。これにより、ストレージ、分析、他の AWS サービスとの統合に AWS IoT SiteWiseのクラウド機能を活用できます。

  詳細については、「データ発行フローを設定する」を参照してください。

- データ保持フロー – エッジにデータを保存します。データ保持フローはSiteWise Edge MQTT ブローカーにサブスクライブしてデータを受信し、InfluxDB形式に変換して、モニタリングのためにローカルに保存します。このローカルストレージは、運用データへの即時アクセスを提供し、タイムクリティカルなアプリケーションのレイテンシーを減らし、ネットワークの中断時の継続性を確保します。

  詳細については、「データ保持フローを設定する」を参照してください。

これらの 2 つのフローは連携して、即時アクセスのためにデータが に送信 AWS IoT SiteWise され、ローカルに保存されます。

Node-RED コンソールにアクセスするには、http://127.0.0.1:1880 にアクセスします。TLS を有効にする方法については、「TLS 暗号化を有効にする」を参照してください。

データ発行フローを設定する

データ発行フローでは、3 つのノードを使用して、産業データをクラウドに送信するパイプラインを作成します。このフローは、クラウドベースの分析、長期ストレージ、他の AWS サービスとの統合を可能にするために不可欠です。まず、シミュレートされたデバイスデータが SiteWise Edge MQTT ブローカーに送信されます。ゲートウェイはブローカーからデータを取得します。これにより、 AWS IoT SiteWise クラウドへの転送が可能になり、強力な分析と視覚化機能を活用できます。

- データ入力 - 産業機器またはシミュレーターからデバイスデータを受信します
- のデータトランスレーター AWS IoT SiteWise - SiteWise Edge ゲートウェイとの互換性を確保するためにデータを AWS IoT SiteWise 形式に変換します
- MQTT パブリッシャー - SiteWise Edge MQTT ブローカーにデータを発行し、ローカルコンシューマーとクラウドコンシューマーの両方が利用できるようにします。

データ入力ノードを設定する

この例では、データ入力ノードは風速データを生成するシミュレートされた風力タービンデバイスを使用します。このノードは、シミュレートされたソース (この例のように) からか、本番環境の実際の産業機器からかにかかわらず、産業用データのエントリポイントとして機能します。

データペイロードにはカスタム JSON 形式を使用して、ローカル処理ツールと AWS IoT SiteWise クラウドサービスの両方で効率的に機能する標準化された構造を提供します。この形式には、タイムスタンプや品質指標などの必須メタデータと実際の測定値が含まれているため、パイプライン全体で包括的なデータ管理と品質追跡が可能になります。挿入ノードをインポートして、タイムスタンプ、品質インジケータ、値を含むこの標準化された JSON 形式でシミュレートされたデータを受信します。

Node-RED 挿入ノードの詳細については、 Node-REDドキュメントの「挿入」セクションを参照してください。

タービンシミュレーターは、この標準化された JSON 形式で毎秒風速データを生成します。

Example : タービンデータペイロード

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

この形式にはいくつかの利点があります。

- name フィールドは特定のプロパティまたは測定値を識別し、同じデバイスから複数のデータポイントを追跡できます。
- ナノ秒timestamp単位の により、正確な履歴分析のための正確な時間追跡が保証されます。
- このqualityインジケータは、信頼性に基づいてデータのフィルタリングと管理に役立ちます。
- 柔軟なvalueフィールドは、さまざまなセンサー出力に対応するためにさまざまなデータ型をサポートします。

Example : タービンシミュレーターの注入ノード

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

データ変換用にノードを設定する

SiteWise Edge ゲートウェイには、 AWS IoT SiteWise クラウドとの互換性を確保するために、特定の形式のデータが必要です。トランスレーターノードは、入力データを必要な AWS IoT SiteWise ペイロード形式に変換する重要なコンポーネントです。この翻訳ステップにより、産業データをクラウド環境で適切に処理、保存、分析 AWS IoT SiteWise できます。

この段階でデータ形式を標準化することで、エッジデバイスとクラウドサービスの統合を有効にし、アセットモデリング、分析、視覚化機能を使用できます。次の構造を使用します。

Example : SiteWise Edge データ解析のペイロード構造

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

> **Note**
>
> を MQTT トピック階層 (例:) `propertyAlias`に一致させます`/Renton/WindFarm/Turbine/WindSpeed`。これにより、データが の正しいアセットプロパティに適切に関連付けられます AWS IoT SiteWise。詳細については、「」の「データストリームエイリアス」の概念を参照してください[AWS IoT SiteWise の概念](#)。

1. AWS IoT SiteWise ペイロード変換用のサンプル関数ノードをインポートします。この関数は、標準化された入力形式から AWS IoT SiteWise互換性のある形式への変換を処理し、適切なタイムスタンプ形式、品質インジケータ、および値の入力を確保します。

```
[
    {
        "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}\nlet typeName = typeNameConverter[typeof input.value]
(input.value)\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 コードが風速データを正しく変換していることを確認します。この関数は、いくつかの重要なタスクを実行します。
   - 入力からプロパティ名を抽出し、propertyAliasとして設定します
   - タイムスタンプをミリ秒から必要な秒およびナノ秒形式に変換します
   - データ信頼性追跡の品質インジケータを保持します
   - 値タイプを自動的に検出し、要件に従って AWS IoT SiteWise フォーマットする
3. ノードをフローに接続し、データ入力ノードと MQTT パブリッシャーの間でリンクします。

ビジネスニーズに固有の関数の作成に関するガイダンスについては、「ドキュメント」の「関数の記述」を参照してください。 Node-RED

MQTT パブリッシャーを設定する

翻訳後、データは SiteWise Edge MQTT ブローカーに発行する準備が整います。

SiteWise Edge MQTT ブローカーにデータを送信するには、以下の設定で MQTT パブリッシャーを設定します。

MQTT Out ノードをインポートするには

1. を使用して MQTT Out 設定ノードをインポートします`"type": "mqtt out"`。MQTT Out ノードを使用すると、ブローカーの設定を共有できます。
2. MQTT ブローカー接続とメッセージルーティングに関連する情報のキーと値のペアを入力します。

サンプル`mqtt out`ノードをインポートします。

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": ""
    }
]
```

サンプル MQTT Out ノードは、次の情報を使用して MQTT 接続を作成します。

- サーバー: `127.0.0.1`
- ポート: `1883`
- プロトコル: `MQTT V5`

次に、MQTT Out ノードは、次の情報を使用してメッセージルーティングを設定します。

- トピック: `/Renton/WindFarm/Turbine/WindSpeed`
- QoS: 1

ノードをデプロイして検証する

3 つのデータ発行フローノードを設定したら、次のステップに従ってフローをデプロイし、データがに正しく送信されていることを確認します。 AWS IoT SiteWise

接続をデプロイして検証するには

1. データ発行フローに示すように、3 つのノードを接続します。

2. デプロイ を選択して、すべてのノード接続の変更を適用します。
3. AWS IoT SiteWise コンソールに移動し、データストリームを選択します。
4. ドロップダウンメニューでエイリアスプレフィックスが選択されていることを確認します。次に、`/Renton/WindFarm/Turbine/WindSpeed`エイリアスを検索します。

検索に正しいエイリアスが表示された場合は、フローをデプロイし、データ転送を検証しました。

データ保持フローを設定する

データ保持フローを使用して、エッジでの運用の可視性を維持できます。これは、ネットワークの中断時や、データへの即時アクセスが必要な場合に役立ちます。このフローは、MQTT ブローカーにサブスクライブしてデバイスデータを受信し、InfluxDB形式に変換してローカルに保存します。このフローを実装することで、オペレーターがクラウド依存関係なしでアクセスできる回復力のあるローカルデータストアを作成し、エッジでリアルタイムのモニタリングと意思決定を可能にします。

このフローは、データが適切にキャプチャおよび保存されるように連携する 3 つの主要コンポーネントで構成されます。

- MQTT サブスクリプションクライアント - ブローカーからデータを受信し、関連するすべての産業データをキャプチャします
- InfluxDB トランスレーター - AWS IoT SiteWise ペイロードを InfluxDB 形式に変換し、効率的な時系列ストレージ用にデータを準備します
- InfluxDB ライター - ローカルストレージを処理し、ローカルアプリケーションのデータの永続性と可用性を確保します

MQTT サブスクリプションクライアントを設定する

- 以下の例をインポート AWS IoT SiteWise して、 の MQTT EMQX ブローカーからデータを受信するように Node-RED で MQTT サブスクリプションクライアントを設定します。

  Example : ノードの 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": ""
    }
]
```

このサブスクリプションにより、ブローカーに発行されたすべての関連データがローカルストレージ用にキャプチャされ、産業オペレーションの完全な記録が提供されます。ノードは、次のサブスクリプション設定で、 MQTT パブリッシャーを設定するセクションと同じ MQTT 接続パラメータを使用します。

- トピック – `/Renton/WindFarm/Turbine/WindSpeed`
- QoS – 1

詳細については、 Node-REDドキュメントの「MQTT ブローカーに接続する」を参照してください。

InfluxDB トランスレーターを設定する

InfluxDB は、インデックス作成用のタグと値のフィールドを使用してデータを整理します。この組織は、時系列データのクエリパフォーマンスとストレージ効率を最適化します。 AWS IoT SiteWise ペイロードを InfluxDB 形式に変換する JavaScript コードを含むサンプル関数ノードをインポートします。トランスレーターはプロパティを 2 つのグループに分割します。

- タグ – 効率的なインデックス作成のための品質と名前のプロパティ
- フィールド – タイムスタンプ (エポックからのミリ秒単位) と値

Example : InfluxDB ペイロードに変換する関数ノード

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

その他の設定オプションについては、Node-RED GitHub リポジトリの node-red-contrib-influxdb を参照してください。 GitHub

InfluxDB ライターのセットアップ

InfluxDB ライターノードは、データ保持フローの最終コンポーネントであり、産業用データをローカル InfluxDB データベースに保存します。このローカルストレージは、ネットワークの中断時に運用上の可視性を維持し、タイムクリティカルなアプリケーションのデータへの即時アクセスを提供する上で重要です。

1. パレットの管理オプションを使用して node-red-contrib-influxdb パッケージをインストールします。このパッケージは、Node-RED を InfluxDB に接続するために必要なノードを提供します。
2. InfluxDB 出力ノードをフローに追加します。このノードは、InfluxDB データベースへのデータの実際の書き込みを処理します。
3. InfluxDB インスタンスへの安全な接続を確立するようにサーバープロパティを設定します。
    a. バージョンを 2.0 に設定する - これは、以前のバージョンとは異なる API を使用する InfluxDB v2.x に接続することを指定します。
    b. URL を に設定する `http://127.0.0.1:8086` - これはローカル InfluxDB インスタンスを指します

c. InfluxDB 認証トークンを入力します。この安全なトークンは、データベースへの接続を承認します。InfluxDB でローカルストレージをセットアップする 手順中にトークンを生成しました。

4. ストレージの場所パラメータを指定して、データの保存場所と保存方法を定義します。

   a. InfluxDB 組織名を入力 – 組織は、バケットとダッシュボードが属するユーザーのグループのワークスペースです。詳細については、『』の「組織の管理」を参照してくださいInfluxData Documentation。

   b. InfluxDB バケットを指定する (例: `WindFarmData`) – バケットは従来のシステムのデータベースと同等であり、時系列データのコンテナとして機能します。

   c. InfluxDB 測定の設定 (例: `TurbineData`) – 測定値はリレーショナルデータベースのテーブルに似ており、関連するデータポイントを整理します。

> **Note**
>
> InfluxDB インスタンスの左側のサイドバーで組織名を見つけます。組織、バケット、測定の概念は、InfluxDB のデータ組織モデルの基本です。詳細については、InfluxDB ドキュメントを参照してください。

保持フローをデプロイして検証する

データ保持フローのすべてのコンポーネントを設定したら、システムが正しく動作していることをデプロイして確認する必要があります。この検証により、産業データがローカルに適切に保存され、すぐにアクセスして分析できるようになります。

1. データ保持フロー図に示すように、3 つのノードを接続します。これにより、データサブスクリプションからローカルストレージへの完全なパイプラインが作成されます。

2. デプロイを選択して変更を適用し、フローをアクティブ化します。これにより、データ収集とストレージプロセスが開始されます。
3. InfluxDB Data Explorer を使用して、データのクエリと視覚化を行います。このツールを使用すると、データが適切に保存されていることを検証し、時系列データの初期ビジュアライゼーションを作成できます。

   Data Explorer では、風速測定値が経時的に記録され、データ生成からローカルストレージまでのパイプライン全体が正しく機能していることを確認できます。

   詳細については、の「Data Explorer でのクエリ」を参照してくださいInfluxData Documentation。

データ公開フローとデータ保持フローの両方がデプロイされたことで、即時のアクセスと回復性のためにローカルコピーを維持しながら、データを AWS IoT SiteWise クラウドに送信する完全なシステムが利用可能になりました。このデュアルパスアプローチにより、エッジでの運用の可視性を維持しながら、クラウドベースの分析とストレージの利点を得ることができます。

SiteWise Edge の Grafana をセットアップする

Grafana® では、産業用データのローカルリアルタイムモニタリングダッシュボードを作成できます。InfluxDB® に保存されているデータを視覚化することで、オペレーターに機器のパフォーマンス、プロセス効率、潜在的な問題に関する迅速なインサイトを提供できます。このエッジでの可視性は、時間的制約のあるオペレーションや、ネットワーク中断中の継続性を維持する上で重要です。

データソースの設定

Grafana を InfluxDB データベースに接続すると、産業用データ用の強力な視覚化レイヤーが作成されます。この接続により、オペレーターがクラウド依存関係なしで情報に基づいた意思決定を行うために使用できるリアルタイムのモニタリングダッシュボードが可能になります。

1. ブラウザで http://127.0.0.1:3000 に移動して、Grafana インスタンスにローカルにアクセスします。TLS を有効にする必要がある場合は、 Grafana Labsドキュメントの「安全なウェブトラフィック用に Grafana HTTPS を設定する」を参照してください。
2. Node-RED InfluxDB がデータを書き込む InfluxDB 時系列バケットを指す InfluxDB データソースを追加します。例えば、`WindFarmData`。この接続により、保存されたデータと視覚化プラットフォーム間のリンクが確立されます。
3. 詳細な手順については、 Grafana Labsドキュメントの「InfluxDB データソースの設定」を参照してください。

SiteWise Edge データ用の Grafana ダッシュボードを作成する

ダッシュボードの作成は、ローカルモニタリングソリューションを構築するための最後のステップです。ダッシュボードは産業データの視覚的表現を提供するため、傾向、異常、潜在的な問題を一目で特定しやすくなります。

- ガイドに従ってダッシュボードを作成します。詳細については、 Grafana Labsドキュメントの「最初のダッシュボードを構築する」を参照してください。このテンプレートは、バケットの名前が で、測定値が であることを前提`WindFarmData`としています`TurbineData`。

  提供されたダッシュボードテンプレートの例をインポートしてクイックスタートガイドを使用して、前のセクションで Node-RED が生成したデータの時系列プロットを含むダッシュボードをすばやく作成することもできます。このテンプレートは、特定のモニタリングニーズに合わせてカスタマイズできる出発点を提供します。

```
{
  "__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 とのオープンソース統合を設定する (Linux)

効率的なデプロイプロセスのために、Docker を使用して Linux 環境で Node-RED®、InfluxDB®、Grafana® を設定できます。この方法では、事前設定されたコンテナを使用するため、コンポーネントの迅速なデプロイと簡単な管理が可能になります。

Docker セットアップの前提条件

開始する前に、 に次のものがあることを確認してください。

- MQTT 対応の V3 ゲートウェイ。詳細については、「MQTT 対応、 AWS IoT SiteWise Edge 用 V3 ゲートウェイ」を参照してください。

- Docker Compose プラグイン。インストール手順については、Docker Manuals ドキュメントの Docker Compose「プラグインのインストール」を参照してください。

サービスをデプロイする

このデプロイは、同じホストで SiteWise Edge、InfluxDB、Node-RED、Grafana を実行します。

環境をセットアップする

1. ルートアクセスを取得します。

```
sudo -i
```

2. .env ファイルを作成するか、以下の環境変数をエクスポートします。

```
export INFLUXDB_PASSWORD=your-secure-influxdb-password
export INFLUXDB_TOKEN=your-secure-influxdb-token
export GRAFANA_PASSWORD=your-secure-grafana-password
```

Docker ネットワークを設定する

- という名前でブリッジネットワークを作成しますSiteWiseEdgeNodeRedDemoNetwork。

```
docker network create --driver=bridge SiteWiseEdgeNodeRedDemoNetwork
```

Docker Compose ファイルを準備する

次の YAML ファイルの内容を SiteWise Edge ゲートウェイデバイスにコピーします。

を展開して Docker Compose YAML ファイルの例を表示する

```
services:
  influxdb:
    image: influxdb:latest
    container_name: influxdb
    ports:
      - "127.0.0.1:8086:8086"
    volumes:
      - influxdb-storage:/.influxdbv2
    environment:
```

```
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=admin
      - DOCKER_INFLUXDB_INIT_PASSWORD=${INFLUXDB_PASSWORD}
      - DOCKER_INFLUXDB_INIT_ORG=iot-sitewise-edge
      - DOCKER_INFLUXDB_INIT_BUCKET=WindFarmData
      - DOCKER_INFLUXDB_INIT_RETENTION=0
      - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=${INFLUXDB_TOKEN}
    networks:
      - SiteWiseEdgeNodeRedDemoNetwork
    restart: unless-stopped

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - "127.0.0.1:3000:3000"
    volumes:
      - grafana-storage:/var/lib/grafana
      - ./grafana/provisioning:/etc/grafana/provisioning
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PASSWORD}
      - GF_INSTALL_PLUGINS=grafana-clock-panel,grafana-simple-json-datasource
      - GF_PATHS_PROVISIONING=/etc/grafana/provisioning
      - GF_PATHS_CONFIG=/etc/grafana/grafana.ini
      - GF_LOG_LEVEL=info
    configs:
      - source: grafana_datasource
        target: /etc/grafana/provisioning/datasources/influxdb.yaml
      - source: grafana_preload_dashboard_config
        target: /etc/grafana/provisioning/dashboards/dashboard.yml
      - source: grafana_preload_dashboard
        target: /etc/grafana/provisioning/dashboards/demo_dashboard.json
    depends_on:
      - influxdb
    networks:
      - SiteWiseEdgeNodeRedDemoNetwork
    restart: unless-stopped

  nodered:
    build:
      context: .
      dockerfile_inline: |
        FROM nodered/node-red:latest
```

```
        RUN npm install node-red-contrib-influxdb
    container_name: nodered
    ports:
      - "127.0.0.1:1880:1880"
    volumes:
      - node_red_data:/data
    environment:
      - NODE_RED_ENABLE_SAFE_MODE=false
      - NODE_RED_ENABLE_PALETTE_EDIT=true
      - NODE_RED_AUTO_INSTALL_MODULES=true
    configs:
      - source: nodered_flows
        target: /data/flows.json
      - source: nodered_settings
        target: /data/settings.js
      - source: nodered_flows_cred
        target: /data/flows_cred.json
    depends_on:
      - influxdb
    networks:
      - SiteWiseEdgeNodeRedDemoNetwork
    restart: unless-stopped

volumes:
  influxdb-storage:
  grafana-storage:
  node_red_data:

networks:
  SiteWiseEdgeNodeRedDemoNetwork:
    external: true

configs:
  grafana_datasource:
    content: |
      apiVersion: 1
      datasources:
        - name: windfarm-demo
          type: influxdb
          access: proxy
          url: http://influxdb:8086
          jsonData:
            version: Flux
            organization: iot-sitewise-edge
```

```
            defaultBucket: WindFarmData
            tlsSkipVerify: true
          secureJsonData:
            token: ${INFLUXDB_TOKEN}
          editable: false

  grafana_preload_dashboard_config:
    content: |
      apiVersion: 1
      providers:
        - name: "Dashboard provider"
          orgId: 1
          type: file
          options:
            path: /etc/grafana/provisioning/dashboards

  grafana_preload_dashboard:
    content: |
      {
        "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": 1,
        "links": [],
        "panels": [
          {
            "datasource": {
              "type": "influxdb",
```

```
            "uid": "PEB0DCBF338B3CEB2"
          },
          "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",
            "targets": [
              {
                "datasource": {
                  "type": "influxdb",
                  "uid": "PEB0DCBF338B3CEB2"
                },
                "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": "Wind Speed",
            "type": "timeseries"
          }
        ],
        "preload": false,
        "schemaVersion": 41,
        "tags": [],
        "templating": {
          "list": []
        },
        "time": {
          "from": "now-6h",
          "to": "now"
        },
        "timepicker": {},
        "timezone": "browser",
        "title": "Demo Dashboard",
        "uid": "eejtureqjo9a8c",
        "version": 2
      }

  nodered_flows:
    content: |
      [
        {
          "id": "95fce448fdd43b47",
          "type": "tab",
          "label": "Demo Flow",
          "disabled": false,
          "info": ""
        },
        {
          "id": "5f63740b66af3386",
          "type": "mqtt out",
          "z": "95fce448fdd43b47",
          "name": "Publish to MQTT broker",
          "topic": "/Renton/WindFarm/Turbine/WindSpeed",
          "qos": "1",
          "retain": "",
```

```
          "respTopic": "",
          "contentType": "",
          "userProps": "",
          "correl": "",
          "expiry": "",
          "broker": "5744207557fa19be",
          "x": 830,
          "y": 200,
          "wires": []
        },
        {
          "id": "8f2eb590d596679b",
          "type": "function",
          "z": "95fce448fdd43b47",
          "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}\nlet typeName = typeNameConverter[typeof
 input.value](input.value)\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": [
            [
              "5f63740b66af3386"
            ]
          ]
        },
        {
          "id": "4b78cbdea5e3258c",
          "type": "inject",
          "z": "95fce448fdd43b47",
```

```
        "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": [
          [
            "8f2eb590d596679b"
          ]
        ]
      },
      {
        "id": "b658bf337ea2e316",
        "type": "influxdb out",
        "z": "95fce448fdd43b47",
        "influxdb": "2f1c38495035d2e4",
        "name": "Store data in InfluxDB",
        "measurement": "TurbineData",
        "precision": "",
```

```
          "retentionPolicy": "",
          "database": "",
          "retentionPolicyV18Flux": "",
          "org": "iot-sitewise-edge",
          "bucket": "WindFarmData",
          "x": 840,
          "y": 340,
          "wires": []
        },
        {
          "id": "9432d39af35b202f",
          "type": "function",
          "z": "95fce448fdd43b47",
          "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": [
            [
              "b658bf337ea2e316"
            ]
          ]
        },
        {
          "id": "b689403d2c80816b",
          "type": "mqtt in",
          "z": "95fce448fdd43b47",
          "name": "Subscribe to MQTT broker",
          "topic": "/Renton/WindFarm/Turbine/WindSpeed",
          "qos": "1",
          "datatype": "auto-detect",
```

```
            "broker": "5744207557fa19be",
            "nl": false,
            "rap": true,
            "rh": 0,
            "inputs": 0,
            "x": 290,
            "y": 340,
            "wires": [
              [
                "9432d39af35b202f"
              ]
            ]
          },
          {
            "id": "4f59bed8e829fc35",
            "type": "comment",
            "z": "95fce448fdd43b47",
            "name": "Data Publish Flow",
            "info": "dfgh",
            "x": 270,
            "y": 160,
            "wires": []
          },
          {
            "id": "b218c7fc58c8b6e7",
            "type": "comment",
            "z": "95fce448fdd43b47",
            "name": "Data Retention flow",
            "info": "",
            "x": 270,
            "y": 300,
            "wires": []
          },
          {
            "id": "5744207557fa19be",
            "type": "mqtt-broker",
            "name": "emqx",
            "broker": "emqx",
            "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": ""
        },
        {
          "id": "2f1c38495035d2e4",
          "type": "influxdb",
          "hostname": "influxdb",
          "port": 8086,
          "protocol": "http",
          "database": "",
          "name": "InfluxDB",
          "usetls": false,
          "tls": "",
          "influxdbVersion": "2.0",
          "url": "http://influxdb:8086",
          "timeout": "",
          "rejectUnauthorized": false
        }
      ]

  nodered_flows_cred:
    content: |
      {
        "2f1c38495035d2e4": {
          "token": "${INFLUXDB_TOKEN}"
        }
      }

  nodered_settings:
    content: |
      module.exports = {
```

```
        flowFile: 'flows.json',
        credentialSecret: false,
        adminAuth: null,
        editorTheme: {
          projects: {
            enabled: false
          }
        }
      }
```

SiteWise Edge デプロイを更新する

1. [AWS IoT console] (コンソール) に移動します。
2. 管理セクションの左側のナビゲーションメニューで Greengrass デバイスを選択し、次に Core デバイスを選択します。
3. SiteWise Edge Gateway に接続されているコアデバイスを選択します。
4. デプロイタブを選択し、デプロイ ID 値を選択します。
5. Actions を選択し、Revise を選択します。
6. ポップアップメッセージを読み、デプロイの修正を選択します。
7. ステップ 2 - コンポーネントを選択するで、次のコンポーネントを選択し、次へを選択します。
   - `aws.greengrass.clientdevices.mqtt.EMQX`
   - `aws.iot.SiteWiseEdgePublisher`
8. ステップ 3 - コンポーネントを設定するで、`aws.greengrass.clientdevices.mqtt.EMQX`コンポーネント値を選択し、次のネットワーク設定を追加します。

```
{
    "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. [次へ] を選択します。
10. ステップ 4 - 詳細設定を構成するで、次へを選択します。
11. [デプロイ] をクリックします。

サービスを起動する

1. Docker Compose ファイルを使用してサービスを開始します。`compose.yaml` ファイルを含むディレクトリで次のコマンドを実行します。

```
docker compose up -d
```

2. サービスにアクセスするための SSH トンネルを作成します。

```
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
```

このデプロイでは、 に次のサービスが作成されます`SiteWiseEdgeNodeRedDemoNetwork network`。

InfluxDB v2 (ポート 8086)

事前設定された組織 (iot-sitewise-edge)、WindFarmData InfluxDB バケット、管理者認証情報が含まれます。

Node-RED (ポート 1880)

InfluxDB ノードと AWS IoT SiteWise 統合用に事前設定されたフローが含まれます

Grafana (ポート 3000)

管理者ユーザー、InfluxDB データソース、モニタリングダッシュボードが含まれます

サービスへのアクセス

デプロイ後、次の URLsと認証情報を使用して サービスにアクセスします。

> **Note**
> ホストまたはゲートウェイマシンから各サービスにアクセスできます。

サービスアクセスの詳細

サービス	[URL]	認証情報
Node-RED	http://127.0.0.1:1880	認証情報は必要ありません
InfluxDB	http://127.0.0.1:8086	ユーザー名: admin パスワード: $INFLUXDB_PASSWORD
Grafana	http://127.0.0.1:3000	ユーザー名: admin パスワード: $GRAFANA_PASSWORD

デプロイメントを確認する

デプロイが成功したことを確認するには、次のチェックを実行します。

1. Node-RED の場合は、2 つのプリロードされたフローが存在することを確認します。
   - データ発行フロー
   - データ保持フロー
2. では AWS IoT SiteWise、 AWS IoT SiteWise コンソールで、エイリアス を使用してデータストリームが存在することを確認します`/Renton/WindFarm/Turbine/WindSpeed`。
3. InfluxDB の場合、Data Explorer を使用して、`WindFarmData`バケット内の`TurbineData`測定値のデータストレージを検証します。
4. Grafana の場合は、ダッシュボードを表示して、Node-RED から生成された時系列データの表示を確認します。

オープンソース統合のデータを処理する

データは、さまざまなツールを使用してさまざまな段階で処理 (変換や集約など) でき、それぞれが異なるモニタリング要件に対応します。

Node-RED ノードでデータを処理

Node-RED® 組み込み処理ノードを使用して、データをリアルタイムで変換します。Node-RED コンソールを使用してこれらのノードを設定し、データパイプラインを作成します。

データ変換ノード

これらのノードを使用して、 の変換と同様に AWS IoT SiteWise、個々のデータポイントを変換します。

- change node - データに対して単純な値の変更を実行します。
- 関数ノード - 複雑なデータ処理のためのカスタム JavaScript 変換を有効にします。

メトリクス計算ノード

これらのノードを使用して、 のメトリクスと同様に AWS IoT SiteWise、複数のデータポイントを 1 つの出力に結合します。

- batch node - バッチ処理用に複数のメッセージをグループ化します。
- join node - 複数のデータストリームを 1 つの出力に結合します。
- アグリゲータノード - 複数のデータポイントから集計メトリクスを計算します。

その他のノードオプションについては、「Node-RED Library」を参照してください。

InfluxDB タスクを作成する

Node-RED は迅速なセットアップによる基本的なデータ処理に優れていますが、フローベースのプログラミングでは複雑なメトリクス計算が困難になる可能性があります。InfluxDB® タスクは、高度な処理ニーズに対応するために、スケジュールされた Flux スクリプトを介して代替手段を提供します。

InfluxDB タスクを以下に使用します。

- 大規模なデータセット間の統計集計
- 複数のプロパティに対する数学的オペレーション

- 複数のソースから派生した測定値

タスク機能

- スケジュールされた実行 - cron 式に基づいてタスクを実行する
- バッチ処理 - 時系列データのオペレーションを最適化する
- エラー復旧 - 失敗したオペレーションを自動的に再試行する
- モニタリング - 詳細ログを通じて実行を追跡する

InfluxDB UI、API、または CLI を使用してタスクを管理します。詳細については、「InfluxDB タスクでデータを処理する」を参照してください。

Grafana 変換を使用する

InfluxDB のソースデータを変更せずに、Grafana® でデータの視覚化を変換します。Grafana 変換は視覚化レイヤーにのみ適用されます。

- Visual Builder - コードを記述せずに変換を作成する
- ライブプレビュー - 変換結果をリアルタイムで表示する
- マルチソース - 複数のデータベースソースからのデータを処理
- ストレージ効率 - 中間結果を保存せずに視覚化時にデータを変換する

詳細については、「データの変換」を参照してください。

オープンソース統合のトラブルシューティング

SiteWise Edge ゲートウェイのオープンソース統合に関連するトラブルシューティングトピックの詳細については、「」を参照してくださいEdge でのオープンソース統合のトラブルシューティング。

AWS IoT SiteWise Edge のクラシックストリーム、V2 ゲートウェイ

Classic ストリーム、 AWS IoT SiteWise Edge 用 V2 ゲートウェイの機能と制限について説明します。

Classic ストリームである V2 ゲートウェイは、MQTT 対応の V3 ゲートウェイが導入される前に、以前の AWS IoT SiteWise デプロイで使い慣れた従来の機能を維持します。これらの SiteWise Edge ゲートウェイは Classic ストリーム、V2 ゲートウェイと見なされます。下位互換性を維持し、デー

タ処理パックで機能します。Classic ストリームである V2 ゲートウェイは、既存のセットアップで信頼性の高いパフォーマンスを提供しますが、新しいゲートウェイオプションと比較して制限があります。具体的には、このゲートウェイタイプは、MQTT 対応 V3 ゲートウェイの送信先で使用できる高度な機能と完全には互換性がありません。MQTT メッセージングプロトコルを使用するには、新しい MQTT 対応 V3 ゲートウェイを作成できます。詳細については、「MQTT 対応、 AWS IoT SiteWise Edge 用 V3 ゲートウェイ」を参照してください。

トピック

- パックを使用して SiteWise Edge でデータを収集および処理する
- AWS IoT SiteWise パブリッシャーコンポーネントを設定する
- 送信先と AWS IoT Greengrass ストリームマネージャー
- AWS IoT SiteWise Edge でエッジ機能を設定する
- AWS IoT SiteWise モデルとアセットのエッジデータ処理を設定する

パックを使用して SiteWise Edge でデータを収集および処理する

AWS IoT SiteWise エッジゲートウェイは、さまざまなパックを使用してデータを収集および処理する方法を決定します。

現在、次のパックが利用できます。

- データ収集パック – このパックを使用して産業データを収集し、 AWS クラウドの送信先にルーティングします。デフォルトでは、このパックは SiteWise Edge ゲートウェイに対して自動的に有効になっています。
- データ処理パック - SiteWise Edge ゲートウェイとエッジで構成されたアセットモデルおよびアセットとの通信を可能にするには、このパックを使用します。エッジ設定を使用すると、オンサイトでコンピューティング、処理するアセットデータを制御できます。その後、 AWS IoT SiteWise または他の AWS サービスにデータを送信できます。データ処理パックの詳細については、the section called “エッジデータ処理を設定する” を参照してください。

パックのアップグレード

> **Important**
>
> データ処理パックのバージョンを 2.0.x より前 (および 2.1.x を含む) からバージョン 2.1.x にアップグレードすると、ローカルに保存された測定値のデータが失われます。

SiteWise Edge ゲートウェイは、お客様のデータを収集し処理する方法を決定するために、さまざまなパックを使用します。 AWS IoT SiteWise コンソールを使用してパックをアップグレードできます。

パックをアップグレードする (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションで、「エッジ」セクションの「エッジゲートウェイ」を選択します。
3. [ゲートウェイ] リストで、アップグレードするパックがある SiteWise Edge ゲートウェイを選択します。
4. [ゲートウェイ設定] セクションで、[ソフトウェアのアップデートが利用可能] を選択します。
5. ソフトウェアバージョンの編集ページで、更新を選択します。

> **Note**
>
> アップグレードできるのは有効になっているパックだけです。この SiteWise Edge ゲートウェイで有効になっているパックのリストを確認するには、概要を選択し、エッジ機能セクションを参照してください。

6. [ソフトウェアバージョンの編集] ページの [ゲートウェイコンポーネントのアップデート] セクションで、以下を実行します。
   - [OPC UA コレクター] を更新するには、バージョンを選択し、[デプロイ] を選択します。
   - [パブリッシャー] を更新するには、バージョンを選択し、[デプロイ] を選択します。
   - [データ処理パック] を更新するには、バージョンを選択し、 [デプロイ] を選択します。
7. 新しいバージョンのデプロイが完了したら、[完了] を選択します。

パックのアップグレードで問題が発生した場合は、「SiteWise Edge ゲートウェイにパックをデプロイできない」を参照してください。

AWS IoT SiteWise パブリッシャーコンポーネントを設定する

AWS IoT SiteWise Edge ゲートウェイを作成してソフトウェアをインストールしたら、SiteWise Edge ゲートウェイがデータを AWS クラウドにエクスポートできるようにパブリッシャーコンポーネントを設定できます。パブリッシャーコンポーネントを使用して、追加機能を有効にしたり、デフォルト設定を設定したりできます。詳細については、「AWS IoT Greengrass Version 2 開発者ガイド」の「AWS IoT SiteWise パブリッシャー」を参照してください。

> **Note**
>
> パブリッシャーの設定は、使用しているゲートウェイのタイプによって異なります。Classic ストリーム、V2 ゲートウェイの場合は、 `iotsitewise:publisher:2`名前空間を使用します。MQTT 対応 V3 ゲートウェイの場合は、 `iotsitewise:publisher:3`名前空間を使用します。

Console

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. パブリッシャーを設定する SiteWise Edge ゲートウェイを選択します。
4. [パブリッシャーの設定] セクションで、[編集] を選択します。
5. [発行の順序] では、次のいずれかを選択します。
   - [最も古いデータを最初に発行する] – デフォルトでは、SiteWise Edge ゲートウェイは最も古いデータを最初にクラウドに発行します。
   - [最も新しいデータを最初に発行する] – SiteWise Edge ゲートウェイは最も新しいデータを最初にクラウドに発行します。
6. (オプション) SiteWise Edge ゲートウェイがデータを圧縮しないようにする場合は、[データのアップロード時に圧縮をアクティブ化] の選択を解除します。
7. (オプション) 古いデータを発行しない場合は、[期限切れデータを除外する] を選択し、以下を実行します。
   - [カットオフ期間] には、値を入力して、単位を選択します。カットオフ期間は 5 分から 7 日の間でなければなりません。例えば、カットオフ期間が 3 日間の場合、3 日間より前のデータはクラウドに発行されません。
8. (オプション) ローカルデバイスでのデータの処理方法に関するカスタム設定を設定するには、[ローカルストレージ設定] を選択し、以下を実行します。
   a. [保持期間] には、数値を入力して単位を選択します。保持期間は 1 分から 30 日の間で、ローテーション期間と同じかそれ以上でなければなりません。例えば、保持期間が 14 日間の場合、SiteWise Edge ゲートウェイは、データを 14 日間保存した後、指定されたカットオフ期間より前のデータをエッジから削除します。

b. [ローテーション期間] には、数値を入力して単位を選択します。ローテーション期間は 1 分より大きく、保持期間と同じかそれ以下でなければなりません。例えば、ローテーション期間が 2 日間の場合、SiteWise Edge ゲートウェイは、カットオフ期間より前のデータをバッチ処理して 1 つのファイルとして保存します。を介したセルフホストゲートウェイの場合 AWS IoT Greengrass V2、SiteWise Edge ゲートウェイはデータのバッチを 2 日に 1 回、次のローカルディレクトリに転送します。 `/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`

c. [ストレージ容量] には、1 以上の値を入力します。ストレージ容量が 2 GB の場合、SiteWise Edge ゲートウェイは、ローカルのデー保存量が 2 GB を超えると、データの削除を開始します。

9. [保存] を選択します。

AWS CLI

UpdateGatewayCapabilityConfiguration API を使用してパブリッシャーを設定します。

`capabilityNamespace` パラメータを `iotsitewise:publisher:2` に設定します。

Example : Classic Stream、V2 ゲートウェイのパブリッシャー設定

パブリッシャーの名前空間: `iotsitewise:publisher:2`

```
{
    "SiteWisePublisherConfiguration": {
        "publishingOrder": "TIME_ORDER",
        "enableCompression": true,
        "dropPolicy": {
            "cutoffAge": "7d",
            "exportPolicy": {
                "retentionPeriod": "7d",
                "rotationPeriod": "6h",
                "exportSizeLimitGB": 10
            }
        }
    },
    "SiteWiseS3PublisherConfiguration": {
        "accessRoleArn": "arn:aws:iam:123456789012:role/roleName",
        "streamToS3ConfigMapping": [
            {
                "streamName": "S3_OPC-UA_Data_Collector",
```

```
                "targetBucketArn": "arn:aws:s3:::amzn-s3-demo-bucket/dataCollector",
                "publishPolicy": {
                    "publishFrequency": "10m",
                    "localSizeLimitGB": 10
                },
                "siteWiseImportPolicy": {
                    "enableSiteWiseStorageImport": true,
                    "enableDeleteAfterImport": true
                }
            }
        ]
    }
}
```

パブリッシャーには以下の設定パラメータがあり、カスタマイズすることができます。

`SiteWisePublisherConfiguration`

`publishingOrder`

データがクラウドに公開される順序。このパラメーターは、次のいずれかの値をとることができます。

- `TIME_ORDER` (古い順にデータを公開) – デフォルトでは、データは古い順にクラウドに公開されます。
- `RECENT_DATA` (新しい順にデータを最初) – データは新しい順にクラウドに公開されます。

`enableCompression`

公開前にデータを圧縮`true`するには、これを に設定します。データ圧縮により、帯域幅の使用量を減らすことができます。

`dropPolicy`

(オプション) クラウドに公開するデータを制御するポリシー。

`cutoffAge`

日、時間、分で指定された、公開されるデータの最大期間。例えば、`7d`、`1d7h16m` です。指定したよりも古いデータは送信されません AWS IoT SiteWise。

カットオフ期間より前のデータはクラウドに公開されません。カットオフ期間は 5 分から 7 日の間でなければなりません。

dカットオフ期間を指定すると mと h、を使用できます。m は分数、hは時数、dは日数を表します。

exportPolicy

(オプション) エッジのデータストレージを管理するポリシー。このポリシーは、カットオフ期間より前のデータに適用されます。

retentionPeriod

SiteWise Edge ゲートウェイは、指定された保持期間に保存された後、カットオフ期間より前のデータをローカルストレージから削除します。保持期間は 1 分から 30 日の間で、ローテーション期間と同じかそれ以上でなければなりません。

d保持期間を指定すると、mと h、を使用できます。m は分数、hは時数、dは日数を表します。

rotationPeriod

カットオフ期間より前のデータをバッチ処理して 1 つのファイルとして保存する時間間隔。SiteWise Edge ゲートウェイは、各ローテーション期間の終了時に 1 バッチ分のデータを以下のローカルディレクトリ (/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports) に転送します。ローテーション期間は 1 分より大きく、保持期間と同じかそれ以下でなければなりません。

ローテーション期間を指定すると、mと h、dを使用できます。m は分数、hは時数、dは日数を表します。

exportSizeLimitGB

ローカルに保存されるデータの最大許容サイズ (GB 単位)。このクォータを超えた場合、SiteWise Edge ゲートウェイは、ローカルに保存されているデータのサイズがクォータ以下になるまで、最も古いデータから削除を開始します。このパラメータの値は、1 以上である必要があります。

SiteWiseS3PublisherConfiguration

accessRoleArn

発行先の Amazon S3 バケットを管理するアクセス AWS IoT SiteWise 許可を付与するアクセスロール。

streamToS3ConfigMapping

ストリームを Amazon S3 設定にマッピングする設定の配列。

`streamName`

Amazon S3 設定から読み取られ、発行されるストリーム。

`targetBucketArn`

発行先のバケット ARN。

`publishPolicy`

`publishFrequency`

SiteWise Edge ゲートウェイが Amazon S3 バケットに発行する頻度。

`localSizeLimitGB`

ローカルディスクに書き込まれるファイルの最大サイズ。このしきい値を超えると、パブリッシャーはバッファされたすべてのデータを宛先に発行します。

`siteWiseImportPolicy`

`enableSiteWiseStorageImport`

Amazon S3 バケットから AWS IoT SiteWise ストレージにデータをインポートするには、これを `true` に設定します。

`enableDeleteAfterImport`

AWS IoT SiteWise ストレージへの取り込み後に Amazon S3 バケット内のファイルを削除するには、これを `true` に設定します。

送信先と AWS IoT Greengrass ストリームマネージャー

AWS IoT Greengrass ストリームマネージャーを使用すると、 AWS クラウド のチャネル AWS IoT Analytics、Amazon Kinesis Data Streams のストリーム、 のアセットプロパティ AWS IoT SiteWise、または Amazon Simple Storage Service (Amazon S3) のオブジェクトの宛先にデータを送信できます。詳細については、[AWS IoT Greengrass Version 2 Developer Guide] (デベロッパーガイド) の[Manage data streams on the AWS IoT Greengrass Core] (Core のデータストリームを管理) を参照してください。

Example : データストリームメッセージ構造

次の例は、ストリームマネージャーによって送信される必須のデータ AWS IoT Greengrass ストリームメッセージ構造を示しています。

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

> **Note**
>
> データストリームメッセージには、その構造に (`assetId` と `propertyId`) または `propertyAlias` が含まれる必要があります。

`assetId`

(オプション) 更新するアセットの ID。

`propertyAlias`

(オプション) OPC UA サーバーのデータストリームパスなど、プロパティを識別するエイリアス。例:

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

詳細については、「 AWS IoT SiteWise ユーザーガイド」の「データストリームの管理」を参照してください。

`propertyId`

(オプション) このエントリのアセットプロパティの ID。

`propertyValues`

(必須) アップロードするプロパティ値のリスト。最大 10 個の `propertyValues` 配列要素を指定できます。

`quality`

(オプション) アセットプロパティ値の品質。

`timestamp`

(必須) アセットプロパティ値のタイムスタンプ。

`offsetInNanos`

(オプション) `timeInSeconds` からのナノ秒オフセット。

`timeInSeconds`

(必須) Unix エポック形式のタイムスタンプの日付 (秒単位)。分数ナノ秒のデータは、`offsetInNanos` で提供されます。

`value`

(必須) アセットプロパティの値。

> **Note**
>
> `value` フィールドには以下の値の 1 つだけが含まれます。

`booleanValue`

(オプション) ブール型 (`true` または `false`) のアセットプロパティデータ。

`doubleValue`

(オプション) 倍精度型 (浮動小数点数) のアセットプロパティデータ。

`integerValue`

(オプション) 整数型 (整数) のアセットプロパティデータ。

`stringValue`

(オプション) 文字列型 (文字のシーケンス) のアセットプロパティデータ。

AWS IoT SiteWise Edge でエッジ機能を設定する

AWS IoT SiteWise Edge を使用してデータを収集し、一時的に保存することで、デバイスデータをローカルで整理して処理できます。エッジ処理を有効にすると、集約されたデータのみを に送信して AWS クラウド 、帯域幅の使用量とクラウドストレージコストを最適化できます。 AWS IoT SiteWise コンポーネントを で使用すると AWS IoT Greengrass、 に送信する前にエッジでデータを収集して処理したり AWS クラウド、SiteWise Edge APIs を使用してオンプレミスで管理したりできます。

データ収集は、 で実行されるデータパックと AWS IoT SiteWise コンポーネントを通じて行われます AWS IoT Greengrass。

> **Note**
>
> - AWS IoT SiteWise は、エッジデータを SiteWise Edge ゲートウェイに最大 30 日間保持します。データの保持期間は、お使いのデバイスの使用可能なディスク空間に依存します。
> - SiteWise Edge ゲートウェイが から 30 AWS クラウド 日間切断された場合、データ処理パックは自動的に無効になります。

トピック

- SiteWise Edge でエッジ機能を設定する

SiteWise Edge でエッジ機能を設定する

AWS IoT SiteWise には、SiteWise Edge ゲートウェイがデータを収集および処理する方法を決定するために使用できる以下のパックが用意されています。SiteWise Edge ゲートウェイのエッジ機能を有効にするには、パックを選択します。

- データ収集パックを使用すると、SiteWise Edge ゲートウェイは複数の OPC UA サーバーからデータを収集し、エッジから AWS クラウドにデータをエクスポートできます。SiteWise Edge ゲートウェイにデータソースを追加すると、アクティブになります。

- データ処理パックを使用すると、SiteWise Edge ゲートウェイはエッジで機器データを処理できます。例えば、アセットモデルを使用して、メトリクスや変換をコンピューティングすることができます。アセットモデルとアセットについては、産業用アセットをモデリングする をご参照ください。

> **Note**
>
> - データ処理パックは x86 プラットフォームでのみ使用できます。

エッジ機能を設定するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. エッジ機能を有効にする SiteWise Edge ゲートウェイを選択します。
4. [エッジ機能] セクションで、[編集] を選択します。
5. [エッジ機能] セクションで、[データ処理パックを有効にする (追加料金が発生します)] を選択します。
6. (オプション) [エッジ LDAP 接続] セクションでは、企業ディレクトリ内のユーザーグループにこの SiteWise Edge ゲートウェイへのアクセス権を付与できます。ユーザーグループは、LDAP (Lightweight Directory Access Protocol) の認証情報を使用して SiteWise Edge ゲートウェイにアクセスできます。その後、 AWS IoT SiteWise アプリケーション、 AWS IoT SiteWise API オペレーション、またはその他のツール AWS OpsHub に を使用して SiteWise Edge ゲートウェイを管理できます。詳細については、「SiteWise Edge ゲートウェイを管理」を参照してください。

   > **Note**
   >
   > Linux または Microsoft Windows認証情報を使用して SiteWise Edge ゲートウェイにアクセスすることもできます。詳細については、「Linux オペレーティングシステムの認証情報を使用して、SiteWise Edge ゲートウェイにアクセスする」を参照してください。

   a. [アクティブ化済み] を選択します。
   b. [Provider name] (プロバイダー名) には、LDAP プロバイダーの名前を入力します。

c. [ホスト名または IP アドレス] には、LDAP サーバーのホスト名または IP アドレスを入力します。

d. [Port] (ポート) には、ポート番号を入力します。

e. [Base distinguished name (DN)] (ベース識エイリアス (DN)) には、ベースの識エイリアス (DN) を入力します。

   次の属性型がサポートされています。commonName (CN) 、LocalityName (L) 、stateOrProvinceName (ST) 、organizationName (O) 、organizationalUnitName (OU) 、countryName (C) 、streetAddress (ストリート) 、domainComponent (DC) 、および userid (UID)。

f. [Admin group DN] (Admin グループの DN) には、DN を入力します。

g. [User group DN] (ユーザーグループ DN) には、DN を入力します。

7. [保存] を選択します。

SiteWise Edge ゲートウェイでエッジ機能をアクティブ化したので、エッジ用にアセットモデルを設定する必要があります。アセットモデルのエッジ設定は、アセットプロパティがコンピューティングされる場所を指定します。すべてのプロパティをエッジでコンピューティングすることもできますし、アセットモデルのプロパティを個別に設定することも可能です。アセットモデルのプロパティには、メトリクス、変換、および測定値が含まれます。

アセットプロパティの詳細については、the section called “データのプロパティを定義する” を参照してください。

アセットモデルを作成したら、次はエッジに合わせた設定を行います。エッジ用のアセットモデルの構成については、the section called “アセットモデルを作成する (コンソール) ” を参照してください。

> **Note**
>
> アセットモデルやダッシュボードは、 AWS クラウドと SiteWise Edge ゲートウェイの間で 10 分ごとに自動的に同期されます。また、ローカルの SiteWise Edge ゲートウェイアプリケーションから手動で同期することもできます。

AWS IoT SiteWise モデルとアセットのエッジデータ処理を設定する

AWS IoT SiteWise Edge を使用して、機器データをローカルで収集、保存、整理、モニタリングできます。SiteWise Edge を使用して、産業用データをモデル化し、SiteWise Monitor を使用して、運用スタッフがデータをローカルで視覚化するためのダッシュボードを作成できます。データをローカルで処理して AWS クラウドに送信するか、 AWS IoT SiteWise API を使用してオンプレミスで処理できます。

AWS IoT SiteWise Edge を使用すると、raw データをローカルで処理し、集約されたデータのみを AWS クラウドに送信して、帯域幅の使用量とクラウドストレージコストを最適化できます。

> **Note**
>
> - AWS IoT SiteWise は、エッジデータを SiteWise Edge ゲートウェイに最大 30 日間保持します。データの保持期間は、お使いのデバイスの使用可能なディスク空間に依存します。
> - SiteWise Edge ゲートウェイが AWS クラウドから 30 日間切断された場合、 は自動的に無効SiteWise Edge で OPC UA ソースを設定するになります。

SiteWise Edge でデータ処理するためのアセットモデルを設定する

エッジで SiteWise Edge ゲートウェイデータを処理する前に、エッジ用にアセットモデルを設定する必要があります。アセットモデルのエッジ設定は、アセットプロパティがコンピューティングされる場所を指定します。エッジですべてのプロパティを計算して結果を AWS クラウドに送信するか、各アセットプロパティを個別に計算する場所をカスタマイズするかを選択できます。詳細については、「AWS IoT SiteWise モデルとアセットのエッジデータ処理を設定する」を参照してください。

アセットのプロパティには、メトリクス、変換、および測定値が含まれます。

- メトリクスはアセットの一定期間のデータを集計したものです。既存のメトリクスデータを使用して新しいメトリクスを計算できます。 AWS IoT SiteWise は、常にメトリクスを AWS クラウドに送信して長期保存します。 は、デフォルトで AWS クラウド上のメトリクスを AWS IoT SiteWise 計算します。エッジでメトリクスを計算するようにアセットモデルを設定できます。 は処理結果を AWS クラウド AWS IoT SiteWise に送信します。
- 変換は、アセットのプロパティのデータポイントをあるフォームから別のフォームにマッピングする数式です。変換は入力データとしてメトリクスを使用することができ、その入力と同じ場所でコンピューティングされ、保存されなければなりません。エッジでコンピューティングするようにメ

トリクス入力を設定した場合、 AWS IoT SiteWise はエッジでその関連する変換もコンピューティングします。

- 測定値は、デフォルトでデバイスが収集して AWS クラウドに送信する raw データとしてフォーマットされます。このデータをローカルデバイスに保存するように、アセットモデルを設定することができます。

アセットプロパティの詳細については、the section called “データのプロパティを定義する” を参照してください。

アセットモデルを作成したら、次はエッジに合わせた設定を行います。エッジ用のアセットモデルの構成については、the section called “アセットモデルを作成する (コンソール) ” を参照してください。

> **Note**
>
> アセットモデルやダッシュボードは、 AWS クラウドと SiteWise Edge ゲートウェイの間で 10 分ごとに自動的に同期されます。SiteWise Edge ゲートウェイを管理 から手動で同期することもできます。

REST APIs と AWS Command Line Interface （ AWS CLI) AWS IoT SiteWise を使用して、SiteWise Edge ゲートウェイにエッジのデータをクエリできます。エッジのデータを SiteWise Edge ゲートウェイにクエリする前に、次の前提条件を満たしている必要があります。

- REST APIに認証情報を設定する必要があります。認証情報の設定については、the section called “ゲートウェイを管理” を参照してください。
- SDK のエンドポイントは、SiteWise Edge ゲートウェイの IP アドレスを指している必要があります。詳細は、お使いのSDK のドキュメントでご確認ください。例えば、[AWS SDK for Java 2.x Developer Guide] (デベロッパーガイド) の [Specifying Custom Endpoints] (カスタムエンドポイントの指定) を参照してください。
- SiteWise Edge ゲートウェイ証明書が登録されている必要があります。SiteWise Edge ゲートウェイ証明書の登録については、SDK のドキュメントに記載されています。例えば、[AWS SDK for Java 2.x Developer Guide] (デベロッパーガイド) の [Registering Certificate Bundles in Node.js] (Node.jsで証明書バンドルを登録する) を参照してください。

を使用したデータのクエリの詳細については AWS IoT SiteWise、「」を参照してくださいからのデータのクエリ AWS IoT SiteWise。

AWS IoT SiteWise Edge ゲートウェイにデータソースを追加する

AWS IoT SiteWise Edge ゲートウェイを設定したら、データソースを追加および設定して、ローカル産業機器から にデータを取り込むことができます AWS IoT SiteWise。SiteWise Edge は、OPC UA などのさまざまなプロトコルや、パートナーデータソースを介して利用可能な他の多くのプロトコルをサポートしています。これらのソースにより、ゲートウェイはローカルサーバーに接続し、産業用データを取得できます。データソースを設定することで、さまざまなデータソースからデータを取り込んで、そのデータストリームをアセットプロパティに関連付けることができます。これにより、包括的な産業アセットモデリングとデータマッピングが可能になります AWS IoT SiteWise。

トピック

- AWS IoT SiteWise Edge ゲートウェイの OPC UA データソース
- SiteWise Edge ゲートウェイのパートナーデータソース

AWS IoT SiteWise Edge ゲートウェイの OPC UA データソース

AWS IoT SiteWise Edge ゲートウェイを設定したら、SiteWise Edge ゲートウェイがローカル産業機器から にデータを取り込むことができるようにデータソースを設定できます AWS IoT SiteWise。各ソースは、SiteWise Edge ゲートウェイが接続して産業用データストリームを取得する OPC UA サーバーなど、ローカルサーバーを表します。SiteWise Edge ゲートウェイのセットアップに関する詳細については、「セルフホスト SiteWise Edge ゲートウェイを作成する」を参照してください。

ゲートウェイタイプ、MQTT 対応、V3 ゲートウェイと Classic ストリーム、V2 ゲートウェイは、OPC UA データの処理方法に影響します。Classic ストリーム、V2 ゲートウェイ、OPC UA データソースは、ゲートウェイ IoT SiteWise パブリッシャー設定に直接追加されます。各データソースはゲートウェイと結合され、データルーティングはソースごとに個別に設定されます。対照的に、MQTT 対応 V3 ゲートウェイを使用すると、OPC UA データソースは MQTT トピックに変換され、一元的な送信先を通じて管理されます。各タイプの詳細については、MQTT 対応、 AWS IoT SiteWise Edge 用 V3 ゲートウェイ「」および「」を参照してくださいAWS IoT SiteWise Edge のクラシックストリーム、V2 ゲートウェイ。

> **Note**
>
> AWS IoT SiteWise は、ソースを追加または編集するたびに SiteWise Edge ゲートウェイを再起動します。SiteWise Edge ゲートウェイは、ソース設定の更新中にデータを取り込みません。SiteWise Edge ゲートウェイを再起動する時間は、SiteWise Edge ゲートウェイのソースのタグの数によって異なります。再起動時間は、数秒 (タグが少数しかない SiteWise

Edge ゲートウェイの場合) から数分 (タグが多数ある SiteWise Edge ゲートウェイの場合) までさまざまです。

ソースを作成したら、データストリームをアセットプロパティに関連付けることができます。アセットの作成および使用方法の詳細については、「産業用アセットをモデリングする」を参照してください。

CloudWatch メトリクスを表示して、データソースが AWS IoT SiteWiseに接続されていることを確認できます。詳細については、「AWS IoT Greengrass Version 2 ゲートウェイメトリクス」を参照してください。

現在、 は次のデータソースプロトコル AWS IoT SiteWise をサポートしています。

- OPC UA - 産業オートメーション用のマシンツーマシン (M2M) 通信プロトコル。

追加の産業用プロトコルのサポート

SiteWise Edge は、データソースパートナーとの統合を通じて、幅広い産業用プロトコルをサポートしています。これらのパートナーシップにより、200 を超える異なるプロトコルとの接続が可能になり、さまざまな産業システムやデバイスに対応できます。

利用可能なデータソースパートナーのリストについては、「」を参照してくださいSiteWise Edge ゲートウェイパートナーのデータソースオプション。

SiteWise Edge で OPC UA ソースを設定する

AWS IoT SiteWise コンソールまたは SiteWise Edge ゲートウェイ機能を使用して、ローカル OPC UA サーバーを表す OPC UA ソースを定義して SiteWise Edge ゲートウェイに追加できます。

トピック

- OPC UA ソースを設定する (コンソール)
- OPC UA ソースを設定する (AWS CLI)

OPC UA ソースを設定する (コンソール)

コンソールを使用して、次の手順で OPC UA ソースを設定できます。

> **Note**
>
> 警告: TQVs重複すると、二重充電が発生する可能性があります。

AWS IoT SiteWise コンソールを使用して OPC UA ソースを設定するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションで、「エッジ」セクションの「エッジゲートウェイ」を選択します。
3. SiteWise Edge ゲートウェイを選択して、OPC UA ソースを追加します。
4. [Add data source] (データソースを追加する) を選択する。
5. ソースの名前を入力します。
6. データソースサーバーの [Local endpoint (ローカルエンドポイント)] を入力します。エンドポイントには、IP アドレスまたはホスト名を指定することができます。また、ローカルエンドポイントにポート番号を追加することもできます。例えば、ローカルエンドポイントは **`opc.tcp://203.0.113.0:49320`** のようになります。
7. (オプション) [Node ID for selection] (選択のためのノード ID) について、ノードフィルターを追加して、 AWS クラウドに取り込まれるデータストリームを制限します。デフォルトでは、SiteWise Edge ゲートウェイはサーバーのルートノードを使用して、すべてのデータストリームを取り込みます。ノードフィルターを使用して、 AWS IoT SiteWiseでモデル化するデータへのパスのみを含めることで、SiteWise Edge ゲートウェイの起動時間と CPU 使用率を減らすことができます。デフォルトでは、SiteWise Edge ゲートウェイは `/Server/` で始まるパスを除くすべての OPC UA パスをアップロードします。OPC UA ノードフィルターを定義するには、ノードパスと `*` および `**` のワイルドカード文字を使用できます。詳細については、「SiteWise Edge で OPC UA ノードフィルターを使用する」を参照してください。
8. 送信先は、MQTT 対応、V3 ゲートウェイ、Classic ストリーム、V2 ゲートウェイによって異なります。
   - クラシックストリーム、V2 ゲートウェイの送信先は、ソースと 1:1 の関係にあります。各ソースは、特定の送信先にデータを送信します。
   - MQTT 対応の V3 ゲートウェイの送信先は、ハブアンドスポークモデルにより、異なるゲートウェイ間で複数のデータソースの設定と管理を一元化できるため、個別に設定されます。V3 ゲートウェイで送信先を設定するには、「」を参照してくださいAWS IoT SiteWise Edge の送信先を理解する。

Classic steams, V2 gateway destinations

- AWS IoT SiteWise リアルタイム – これを選択して、データをストレージに直接送信します AWS IoT SiteWise 。エッジでデータをリアルタイムで取り込み、モニタリングします。
- AWS IoT SiteWise Amazon S3 を使用してバッファリング – Parquet 形式でデータを Amazon S3 に送信し、ストレージに AWS IoT SiteWise インポートします。データをバッチで取り込み、履歴データを費用対効果の高い方法で保存するには、このオプションを選択します。任意の Amazon S3 バケットの場所と、Amazon S3 にデータをアップロードする頻度を設定できます。 AWS IoT SiteWiseへの取り込み後にデータを処理する方法を選択することもできます。 AWS IoT SiteWise と Amazon S3 の両方でデータを使用できるようにするか、インポート後に Amazon S3 からデータを自動的に削除するかを選択できます AWS IoT SiteWise。
  - Amazon S3 バケットはステージングおよびバッファリングメカニズムであり、Parquet 形式のファイルをサポートしています。
  - チェックボックスをオンにすると、データをストレージに AWS IoT SiteWise インポートすると、データは最初に Amazon S3 にアップロードされ、次に AWS IoT SiteWise ストレージにアップロードされます。
    - [Amazon S3 からデータを削除する] チェックボックスをオンにすると、データは SiteWise ストレージにインポートされた後、Amazon S3 から削除されます。
    - [Amazon S3 からデータを削除する] チェックボックスをオンにした場合、データは Amazon S3 と SiteWise ストレージの両方に保存されます。
  - [AWS IoT SiteWise ストレージにデータをインポートする] チェックボックスをオフにした場合、データは Amazon S3 にのみ保存されます。SiteWise ストレージにはインポートされません。

  AWS IoT SiteWise が提供するさまざまなストレージオプションの詳細については、「データストレージを管理する」を参照してください。料金の詳細については、「AWS IoT SiteWise の料金」を参照してください。

- AWS IoT Greengrass ストリームマネージャー – AWS IoT Greengrass ストリームマネージャーを使用して、 のチャネル AWS IoT Analytics、Amazon Kinesis Data Streams のストリーム、 のアセットプロパティ AWS IoT SiteWise、Amazon Simple Storage Service (Amazon S3) のオブジェクトなどの AWS クラウド宛先にデータを送信します。詳細につ

いては、「 AWS IoT Greengrass Version 2 デベロッパーガイド」のAWS IoT Greengrass「 Core でデータストリームを管理する」を参照してください。

AWS IoT Greengrass ストリームの名前を入力します。

MQTT-enabled, V3 gateway destinations

1. 関連する送信先の追加については、MQTT 対応、 AWS IoT SiteWise Edge 用 V3 ゲートウェイ「」を参照してください。
2. ソースの送信先を追加したら、この手順に戻ります。

9. [高度な設定] ペインで、以下のことができます。

   a. ソースサーバーと SiteWise Edge ゲートウェイ間の接続と転送中のデータについて [メッセージのセキュリティモード] を選択します。このフィールドは、OPC UA セキュリティポリシーとメッセージセキュリティモードの組み合わせです。OPC UA サーバーに指定したセキュリティポリシーとメッセージセキュリティモードと同じものを選択します。

   b. ソースが認証を必要とする場合、[Authentication configuration] (認証設定) リストから AWS Secrets Manager シークレットを選択します。SiteWise Edge ゲートウェイは、このデータソースに接続するときに、このシークレットの認証情報を使用します。シークレットをデータソース認証に使用するには、SiteWise Edge ゲートウェイの AWS IoT Greengrass コンポーネントにシークレットをアタッチする必要があります。詳細については、「the section called “データソース認証を設定する”」を参照してください。

      > **Tip**
      > データサーバーに、[Allow anonymous login (匿名ログインを許可)] というオプションがある場合があります。このオプションが [はい] の場合、ソースは認証を必要としません。

   c. (オプション) [データストリームのプレフィックスをアクティブ化 - オプション] を選択して、データストリームプレフィックスをアクティブ化できます。

      - (オプション) [データストリームのプレフィックス] を入力します。SiteWise Edge ゲートウェイは、このプレフィックスをこのソースからのすべてのデータストリームに追加します。データストリームのプレフィックスを使用して、異なるソースから同じ名前を持つデータストリームを区別します。各データストリームは、アカウント内で一意の名前を持つ必要があります。

d. (オプション) [データ型変換] オプションを選択して、サポートされていない OPC UA データ型を文字列に変換してから AWS IoT SiteWiseに取り込みます。単純なデータ型を持つ配列値を JSON 文字列に変換し、DateTime データ型を ISO 8601 文字列に変換します。詳細については、「サポートされていないデータ型の変換」を参照してください。

e. (オプション) [プロパティグループ] として、[新しいグループを追加] を選択します。

   i. プロパティグループの「名前」を入力します。

   ii. [Properties] (プロパティ) の場合。

      1. (オプション) [ノードパス] については、OPC UA ノードフィルタを追加して、 AWS IoT SiteWiseにアップロードされる OPC UA パスを制限します。形式は、[選択できるノード ID] と同様です。

   iii. [Group settings] (グループ設定) については、次のようにします。

      1. データ品質設定で、 AWS IoT SiteWise コレクターが取り込むデータ品質のタイプを選択します。

      2. [スキャンモードの設定] については、[スキャンモード] を使用して標準のサブスクリプションプロパティを設定します。[サブスクライブ] または [ポーリング] を選択できます。スキャンモードの詳細については、 the section called “データインジェスト範囲をフィルタリングする” を参照してください。

         Subscribe

            すべてのデータポイントを送信するには

            i. [サブスクライブ] を選択し、以下を設定します。

               A. データ変更トリガー - データ変更アラートを開始する条件。

               B. サブスクリプションキューサイズ - 監視対象アイテムの通知がキューに入れられる特定のメトリクス用の、OPC-UA サーバー上のキューの深さ。

               C. サブスクリプションパブリッシュ間隔 - サブスクリプションの作成時に指定されたパブリッシュサイクルの間隔 (ミリ秒単位)。

               D. スナップショット間隔 - オプション – AWS IoT SiteWise Edge が安定したデータストリームを取り込むようにするためのスナップショット頻度タイムアウト設定。

E. スキャンレート – SiteWise Edge ゲートウェイが registers を読み取るレート。 はSiteWise Edge ゲートウェイの最小許容スキャンレート AWS IoT SiteWise を自動的に計算します。

F. [タイムスタンプ] – OPC UA データポイントに含めるタイムスタンプ。サーバーのタイムスタンプまたはデバイスのタイムスタンプを使用できます。

> **Note**
>
> IoT SiteWise OPC UA コレクターコンポーネントのバージョン 2.5.0 以降を使用します。以前のバージョンでタイムスタンプ機能を使用すると、設定の更新は失敗します。詳細については、「コンポーネントのバージョン AWS IoT SiteWise を更新する」を参照してください。

ii. [デッドバンド設定] で、[デッドバンドタイプ] を設定します。デッドバンドタイプは AWS IoT SiteWise、ソースが送信するデータと破棄するデータを制御します。デッドバンド設定の詳細は、the section called “データインジェスト範囲をフィルタリングする” を参照してください。

- [なし] – 関連付けられたサーバーは、このプロパティグループのすべてのデータポイントを送信します。
- [パーセンテージ] — 関連付けられたサーバーは、データの範囲の指定されたパーセンテージから外れたデータのみを送信します。この範囲は、ノードごとに定義されたエンジニアリングユニットの最小値と最大値に基づいてサーバーによって計算されます。サーバーがデッドバンドの割合をサポートしていない場合、または定義されたエンジニアリングユニットがない場合、ゲートウェイは以下に記載されている最小値と最大値を使用して範囲を計算します。
- [絶対値] – 関連付けられたサーバーは、特定の範囲外のデータのみを送信します。

A. [デッドバンド値] をデータ範囲のパーセンテージとしてデッドバンドに設定します。

B. (オプション) [最小範囲 - オプション] と [最大範囲 - オプション] を使用して、デッドバンド範囲の最小値と最大値を指定します。

Poll

特定の間隔でデータポイントを送信するには

- [ポーリング] を選択し、以下を設定します。

  A. スキャンレート – SiteWise Edge ゲートウェイが registers を読み取るレート。 はSiteWise Edge ゲートウェイの最小許容スキャンレート AWS IoT SiteWise を自動的に計算します。

  B. [タイムスタンプ] – OPC UA データポイントに含めるタイムスタンプ。サーバーのタイムスタンプまたはデバイスのタイムスタンプを使用できます。

> **Note**
>
> IoT SiteWise OPC UA コレクターコンポーネントのバージョン 2.5.0 以降を使用します。以前のバージョンでタイムスタンプ機能を使用すると、設定の更新は失敗します。詳細については、「コンポーネントのバージョン AWS IoT SiteWise を更新する」を参照してください。

> **Note**
>
> [デッドバンド設定] は、[スキャンモード設定] で [サブスクライブ] を選択した場合に適用されます。

10. [保存] を選択します。

OPC UA ソースを設定する (AWS CLI)

AWS CLIを使用して、SiteWise Edge ゲートウェイの OPC UA データソースを定義できます。これを行うには、OPC UA 機能設定 JSON ファイルを作成し、update-gateway-capability-configuration コマンドを使用して SiteWise Edge ゲートウェイ設定を更新します。すべての OPC UA ソースを 1 つの機能構成で定義する必要があります。

MQTT-enabled, V3 gateway

この機能には、以下の名前空間があります。

- `iotsitewise:opcuacollector:3`

```
{
  "sources": [
    {
      "name": "string",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny" | "X509",
          "certificateBody": "string",
          "certificateChain": "string",
        },
        "endpointUri": "string",
        "securityPolicy": "NONE" | "BASIC128_RSA15" | "BASIC256" | "BASIC256_SHA256"
 | "AES128_SHA256_RSAOAEP" | "AES256_SHA256_RSAPSS",
        "messageSecurityMode": "NONE" | "SIGN" | "SIGN_AND_ENCRYPT",
        "identityProvider": {
          "type": "Anonymous" | "Username",
          "usernameSecretArn": "string"
        },
        "nodeFilterRules": [
          {
            "action": "INCLUDE",
            "definition": {
              "type": "OpcUaRootPath",
              "rootPath": "string"
            }
          }
        ]
      },
      "measurementDataStreamPrefix": "string",
      "typeConversions": {
        "array": "JsonArray",
        "datetime": "ISO8601String"
        },
      "destination": {
        {
          "type":"MQTT"
        }
```

```
      },
      "propertyGroups": [
        {
          "name": "string",
          "nodeFilterRuleDefinitions": [
            {
              "type": "OpcUaRootPath",
              "rootPath": "string"
            }
          ],
          "deadband": {
            "type": "PERCENT" | "ABSOLUTE",
            "value": double,
            "eguMin": double,
            "eguMax": double,
            "timeoutMilliseconds": integer
          },
          "scanMode": {
            "type": "EXCEPTION" | "POLL",
            "rate": integer,
            "timestampToReturn": "SOURCE_TIME" | "SERVER_TIME"
          },
          "dataQuality": {
            "allowGoodQuality": true | false,
            "allowBadQuality": true | false,
            "allowUncertainQuality": true | false
          },
          "subscription": {
            "dataChangeTrigger": "STATUS" | "STATUS_VALUE" |
 "STATUS_VALUE_TIMESTAMP",
            "queueSize": integer,
            "publishingIntervalMilliseconds": integer,
            "snapshotFrequencyMilliseconds": integer
          }
        }
      ]
    }
  ]
}
```

Classic streams, V2 gateway

この機能には、以下の名前空間があります。

- iotsitewise:opcuacollector:2

リクエスト構文

```
{
  "sources": [
    {
      "name": "string",
      "endpoint": {
        "certificateTrust": {
          "type": "TrustAny" | "X509",
          "certificateBody": "string",
          "certificateChain": "string",
        },
        "endpointUri": "string",
        "securityPolicy": "NONE" | "BASIC128_RSA15" | "BASIC256" | "BASIC256_SHA256"
 | "AES128_SHA256_RSAOAEP" | "AES256_SHA256_RSAPSS",
        "messageSecurityMode": "NONE" | "SIGN" | "SIGN_AND_ENCRYPT",
        "identityProvider": {
          "type": "Anonymous" | "Username",
          "usernameSecretArn": "string"
        },
        "nodeFilterRules": [
          {
            "action": "INCLUDE",
            "definition": {
              "type": "OpcUaRootPath",
              "rootPath": "string"
            }
          }
        ]
      },
      "measurementDataStreamPrefix": "string",
      "typeConversions": {
        "array": "JsonArray",
        "datetime": "ISO8601String"
        },
      "destination": {
        "type": "StreamManager",
        "streamName": "string",
        "streamBufferSize": integer,
      },
      "propertyGroups": [
```

```
        {
          "name": "string",
          "nodeFilterRuleDefinitions": [
            {
              "type": "OpcUaRootPath",
              "rootPath": "string"
            }
          ],
          "deadband": {
            "type": "PERCENT" | "ABSOLUTE",
            "value": double,
            "eguMin": double,
            "eguMax": double,
            "timeoutMilliseconds": integer
          },
          "scanMode": {
            "type": "EXCEPTION" | "POLL",
            "rate": integer,
            "timestampToReturn": "SOURCE_TIME" | "SERVER_TIME"
          },
          "dataQuality": {
            "allowGoodQuality": true | false,
            "allowBadQuality": true | false,
            "allowUncertainQuality": true | false
          },
          "subscription": {
            "dataChangeTrigger": "STATUS" | "STATUS_VALUE" |
 "STATUS_VALUE_TIMESTAMP",
            "queueSize": integer,
            "publishingIntervalMilliseconds": integer,
            "snapshotFrequencyMilliseconds": integer
          }
        }
      ]
    }
  ]
}
```

リクエスト本文

sources

OPC UA ソース定義構造のリスト。それぞれに以下の情報が含まれています。

`name`

ソースの一意のわかりやすい名前。

`endpoint`

次の情報を含むエンドポイント構造。

`certificateTrust`

次の情報を含む証明書信頼ポリシー構造。

`type`

ソースに対する証明書信頼モード。次のいずれかを選択します。

- `TrustAny` - SiteWise Edge ゲートウェイは、OPC UA ソースに接続するとき、すべての証明書を信頼します。
- `X509` - SiteWise Edge ゲートウェイは、OPC UA ソースに接続するとき、X.509 証明書を信頼します。このオプションを選択する場合は、`certificateTrust` で `certificateBody` を定義する必要があります。`certificateTrust` で `certificateChain` を定義することもできます。

`certificateBody`

(オプション) X.509 証明書の本文。

このフィールドは、`certificateTrust` で `type` に `X509` を選択する場合に必須です。

`certificateChain`

(オプション) X.509 証明書の信頼チェーン。

このフィールドは、`certificateTrust` で `type` に `X509` を選択した場合のみ使用されます。

`endpointUri`

OPC UA ソースのローカルエンドポイント。たとえば、ローカルエンドポイントは `opc.tcp://203.0.113.0:49320` のようになります。

`securityPolicy`

OPC UA ソースから読み込まれるメッセージを保護するために使用するセキュリティポリシー。次のいずれかを選択します。

- NONE - SiteWise Edge ゲートウェイは、OPC UA ソースからのメッセージをセキュリティ保護しません。別のセキュリティポリシーを選択することをお勧めします。このオプションを選択する場合は、messageSecurityMode に NONE を選択する必要もあります。
- BASIC256_SHA256 - Basic256Sha256のセキュリティポリシーです。
- AES128_SHA256_RSAOAEP - Aes128_Sha256_RsaOaepのセキュリティポリシーです。
- AES256_SHA256_RSAPSS - Aes256_Sha256_RsaPssのセキュリティポリシーです。
- BASIC128_RSA15 - (非推奨) Basic128Rsa15 セキュリティポリシーは、安全とはみなされなくなったため、非推奨とされます。別のセキュリティポリシーを選択することをお勧めします。詳細については、「プロファイル SecurityPolicy – Basic128Rsa15」を参照してください。
- BASIC256 - (非推奨) Basic256 セキュリティポリシーは、安全とはみなされなくなったため、非推奨とされます。別のセキュリティポリシーを選択することをお勧めします。詳細については、SecurityPolicy – Basic256」を参照してください。

> **Important**
>
> NONE 以外のセキュリティポリシーを選択した場合、messageSecurityMode は SIGN または SIGN_AND_ENCRYPT を選択する必要があります。また、SiteWise Edge ゲートウェイを信頼するようにソースサーバーを設定する必要があります。詳細については、「AWS IoT SiteWise Edge ゲートウェイを信頼するように OPC UA サーバーを設定する」を参照してください。

messageSecurityMode

OPC UA ソースへの接続を保護するために使用するメッセージセキュリティモード。次のいずれかを選択します。

- NONE - SiteWise Edge ゲートウェイは OPC UA ソースへの接続をセキュリティ保護しません。他のメッセージセキュリティモードを選択することをお勧めします。このオプションを選択する場合は、securityPolicy に NONE を選択する必要もあります。
- SIGN - SiteWise Edge ゲートウェイと OPC UA ソース間で転送されるデータは署名されますが、暗号化されません。
- SIGN_AND_ENCRYPT - ゲートウェイと OPC UA ソース間で転送されるデータは署名され、暗号化されます。

> **Important**
>
> メッセージセキュリティモードを NONE 以外を選択した場合、securityPolicy は NONE以外を選択する必要があります。また、SiteWise Edge ゲートウェイを信頼するようにソースサーバーを設定する必要があります。詳細については、「AWS IoT SiteWise Edge ゲートウェイを信頼するように OPC UA サーバーを設定する」を参照してください。

`identityProvider`

次の情報を含む ID プロバイダ構造。

`type`

ソースで必要な認証資格情報のタイプ。次のいずれかを選択します。

- `Anonymous` - ソースは接続に認証を必要としません。
- `Username` - ソースに接続するには、ユーザー名とパスワードが必要です。このオプションを選択する場合は、`identityProvider` で `usernameSecretArn` を定義する必要があります。

`usernameSecretArn`

(オプション) AWS Secrets Manager シークレットの ARN。SiteWise Edge ゲートウェイは、このソースに接続するとき、このシークレットの認証情報を使用します。ソース認証に使用するには、シークレットを SiteWise Edge ゲートウェイの IoT SiteWise コネクタにアタッチする必要があります。詳細については、「SiteWise Edge のデータソース認証を設定する」を参照してください。

このフィールドは、`identityProvider` で `type` に `Username` を選択する場合に必須です。

`nodeFilterRules`

AWS クラウドに送信する OPC UA データストリームパスを定義するノードフィルタールール構造のリスト。ノードフィルターを使用して、 AWS IoT SiteWiseでモデル化するデータへのパスのみを含めることで、SiteWise Edge ゲートウェイの起動時間と CPU 使用率を減らすことができます。デフォルトでは、SiteWise Edge ゲートウェイは `/Server/` で始まるパスを除くすべての OPC UA パスをアップロードします。OPC UA ノードフィルターを定義するには、ノードパスと `*` および `**` のワイルドカード文字を使用できます。

詳細については、「SiteWise Edge で OPC UA ノードフィルターを使用する」を参照してください。

リスト内の各構造には、次の情報が含まれている必要があります。

`action`

このノードフィルタルールのアクション。以下のオプションを選択できます。

- `INCLUDE` - SiteWise Edge ゲートウェイには、このルールに一致するデータストリームのみが含まれます。

`definition`

次の情報を含むノードフィルタルール構造。

`type`

このルールのノードフィルタパスのタイプ。以下のオプションを選択できます。

- `OpcUaRootPath` - SiteWise Edge ゲートウェイは、OPC UA パス階層のルートに対して、このノードフィルターパスを評価します。

`rootPath`

OPC UA パス階層のルートに対して評価するノードフィルターパス。このパスは `/` から始まる必要があります。

`measurementDataStreamPrefix`

ソースからのすべてのデータストリームの前に付加する文字列。SiteWise Edge ゲートウェイは、このプレフィックスをこのソースからのすべてのデータストリームに追加します。データストリームのプレフィックスを使用して、異なるソースから同じ名前を持つデータストリームを区別します。各データストリームは、アカウント内で一意の名前を持つ必要があります。

`typeConversions`

サポートされていない OPC UA データ型で使用できる変換のタイプ。各データ型は文字列に変換されます。詳細については、「サポートされていないデータ型の変換」を参照してください。

`array`

文字列に変換される単純な配列データ型。以下のオプションを選択できます。

- `JsonArray` – 単純な配列データ型を文字列に変換することを選択することを示します。

`datetime`

文字列に変換される DateTime データ型。以下のオプションを選択できます。

- `ISO8601String` – ISO 8601 データ型を文字列に変換することを選択することを示します。

`destination`

OPC UA タグの送信先の設定。クラシックストリーム、v2 および MQTT 対応、V3 ゲートウェイでは、送信先の設定が異なります。

`type`

デスティネーションの型。

`streamName` — Classic ストリーム、V2 ゲートウェイのみ

ストリームの名前。ストリーム名は一意である必要があります。

`streamBufferSize` — Classic ストリーム、V2 ゲートウェイのみ

ストリームのバッファサイズ。これは、OPC UA ソースからのデータフローを管理するために重要です。

`propertyGroups`

(オプション) プロトコルが要求する `deadband` と `scanMode` を定義するプロパティグループのリストです。

`name`

プロパティグループの名前です。これは一意な識別子である必要があります。

`deadband`

deadband 値は、データがクラウドに送信される前に発生する必要があるデータポイントの値の最小変更を定義します。これには、以下の情報が含まれています。

`type`

対応するデッドバンドの種類。以下のオプションを選択できます。

- `ABSOLUTE` – データポイントがクラウドに送信されるのに十分な重要性があると考えるために必要な最小絶対変更を指定する固定値。
- `PERCENT` – 最後に送信されたデータポイントの値のパーセンテージとして必要な最小変更を指定する動的値。このタイプのデッドバンドは、データ値が時間の経過とともに大きく変化する場合に便利です。

value

デッドバンドの値。type が ABSOLUTE のとき、この値は単位なしの 2 倍とななります。type が PERCENT の場合、1 と 100 の間の 2 倍の値となります。

eguMin

(オプション) PERCENT　デッドバンドを使用する場合の工学単位の最小値。OPC UA サーバーに工学単位が設定されていない場合に設定します。

eguMax

(オプション) PERCENT デッドバンドを使用する場合の工学単位の最大値。OPC UA サーバーに工学単位が設定されていない場合に設定します。

timeoutMilliseconds

タイムアウトまでの時間をミリ秒単位で指定する。最小値は 100 です。

scanMode

次の情報を含む scanMode 構造体。

type

scanMode の対応型。有効な値は、POLL および EXCEPTION です。

rate

スキャンモードのサンプリング間隔。

timestampToReturn

タイムスタンプのソース。以下のオプションを選択できます。

- SOURCE_TIME – デバイスからのタイムスタンプを使用します。
- SERVER_TIME – サーバーからのタイムスタンプを使用します。

> **Note**
>
> IoT SiteWise OPC UA コレクターコンポーネントのバージョン 2.5.0 以降TimestampToReturnで を使用します。以前のバージョンでこの機能を使用すると、設定の更新は失敗します。詳細については、「コンポーネントのバージョン AWS IoT SiteWise を更新する」を参照してください。

`nodeFilterRuleDefinitions`

(オプション) プロパティグループに含めるノードパスのリスト。プロパティグループの重複があってはいけません。このフィールドに値を指定しなかった場合、グループにはルート下のすべてのパスが含まれ、追加のプロパティグループを作成することはできません。`nodeFilterRuleDefinitions` 構造体には、以下の情報が含まれています。

`type`

`OpcUaRootPath`は、サポートされている唯一のタイプです。これは、`rootPath` の値が OPC UA ブラウジングスペースのルートからの相対パスであることを指定します。

`rootPath`

プロパティグループに含めるパス (ルートからの相対パス) を指定したカンマ区切りのリスト。

Classic ストリーム、V2 ゲートウェイの追加機能設定の例 (AWS CLI)

次の例は、JSON ファイルに保存されているペイロードから OPC UA SiteWise Edge ゲートウェイの機能設定を定義します。

```
aws iotsitewise update-gateway-capability-configuration \
--capability-namespace "iotsitewise:opcuacollector:2" \
--capability-configuration file://opc-ua-configuration.json
```

Example : OPC UA ソースの設定

次の `opc-ua-configuration.json` ファイルは、基本的で安全でない OPC UA ソース設定を定義しています。

```
{
    "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 : プロパティグループが定義された OPC UA ソース設定

次の `opc-ua-configuration.json` ファイルは、プロパティグループが定義されている基本的な安全でない OPC UA ソース構成を定義しています。

```
{
    "sources": [
        {
            "name": "source1",
            "endpoint": {
                "certificateTrust": {
                    "type": "TrustAny"
                },
                "endpointUri": "opc.tcp://10.0.0.9:49320",
                "securityPolicy": "NONE",
                "messageSecurityMode": "NONE",
                "identityProvider": {
                    "type": "Anonymous"
                },
                "nodeFilterRules": [
                    {
                        "action": "INCLUDE",
                        "definition": {
                            "type": "OpcUaRootPath",
                            "rootPath": "/Utilities/Tank"
                        }
                    }
                ]
            },
            "measurementDataStreamPrefix": "propertyGroups",
            "propertyGroups": [
                 {
                     "name": "Deadband_Abs_5",
                     "nodeFilterRuleDefinitions": [
```

```
                    {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank/Temperature/TT-001"
                    },
                    {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank/Temperature/TT-002"
                    }
                ],
                "deadband": {
                    "type":"ABSOLUTE",
                    "value": 5.0,
                    "timeoutMilliseconds": 120000
                }
            },
            {
                "name": "Polling_10s",
                "nodeFilterRuleDefinitions": [
                    {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank/Pressure/PT-001"
                    }
                ],
                "scanMode": {
                    "type": "POLL",
                    "rate": 10000
                }
            },
            {
                "name": "Percent_Deadband_Timeout_90s",
                "nodeFilterRuleDefinitions": [
                    {
                        "type": "OpcUaRootPath",
                        "rootPath": "/Utilities/Tank/Flow/FT-*"
                    }
                ],
                "deadband": {
                    "type":"PERCENT",
                    "value": 5.0,
                    "eguMin": -100,
                    "eguMax": 100,
                    "timeoutMilliseconds": 90000
                }
            }
```

```
            ]
        }
    ]
}
```

Example : OPC UA プロパティ付きソース構成

次の `opc-ua-configuration.json` の JSON 例は、次のプロパティを持つ OPC UA ソース設定を定義しています。

- すべての証明書を信頼します。
- メッセージの保護に `BASIC256` セキュリティポリシーを使用します。
- `SIGN_AND_ENCRYPT` モードを使用して接続をセキュリティ保護します。
- Secrets Manager シークレットに保存されている認証資格情報を使用します。
- パスが `/WindFarm/2/WindTurbine/` で始まるものを除き、データストリームを除外します。
- この「Wind Farm #2」と別のエリアの「Wind Farm #2」を区別するために、すべてのデータストリームパスの先頭に `/Washington` を追加します。

```
{
    "sources": [
        {
            "name": "Wind Farm #2",
            "endpoint": {
                "certificateTrust": {
                    "type": "TrustAny"
                },
                "endpointUri": "opc.tcp://203.0.113.1:49320",
                "securityPolicy": "BASIC256",
                "messageSecurityMode": "SIGN_AND_ENCRYPT",
                "identityProvider": {
                    "type": "Username",
                    "usernameSecretArn":
 "arn:aws:secretsmanager:region:123456789012:secret:greengrass-windfarm2-auth-1ABCDE"
                },
                "nodeFilterRules": [
                  {
                      "action": "INCLUDE",
                      "definition": {
                          "type": "OpcUaRootPath",
```

```
                            "rootPath": "/WindFarm/2/WindTurbine/"
                        }
                    }
                ]
            },
            "measurementDataStreamPrefix": "/Washington"
        }
    ]
}
```

Example : 証明書の信頼による OPC UA ソース設定

次の `opc-ua-configuration.json` の JSON 例は、次のプロパティを持つ OPC UA ソース設定を定義しています。

- 指定された X.509 証明書を信頼します。
- メッセージの保護に `BASIC256` セキュリティポリシーを使用します。
- `SIGN_AND_ENCRYPT` モードを使用して接続をセキュリティ保護します。

```
{
    "sources": [
        {
            "name": "Wind Farm #3",
            "endpoint": {
                "certificateTrust": {
                    "type": "X509",
                    "certificateBody": "-----BEGIN CERTIFICATE-----
          MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
```

```
 3rrszlaEXAMPLE=
          -----END CERTIFICATE-----",
                    "certificateChain": "-----BEGIN CERTIFICATE-----
          MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
          -----END CERTIFICATE-----"
                },
                "endpointUri": "opc.tcp://203.0.113.2:49320",
                "securityPolicy": "BASIC256",
                "messageSecurityMode": "SIGN_AND_ENCRYPT",
                "identityProvider": {
                    "type": "Anonymous"
                },
                "nodeFilterRules": []
              },
            "measurementDataStreamPrefix": ""

        }
    ]
}
```

AWS IoT SiteWise Edge ゲートウェイを信頼するように OPC UA サーバーを設定する

OPC UA ソースを設定するときに None `messageSecurityMode`以外の を選択した場合は、ソースサーバーが AWS IoT SiteWise Edge ゲートウェイを信頼できるようにする必要があります。SiteWise Edge ゲートウェイは、ソースサーバーが必要とする証明書を生成します。プロセスはソースサーバーによって異なります。詳細については、サーバーのドキュメントを参照してください。

以下の手順では、基本的なステップの概要を説明します。

OPC UA サーバーが SiteWise Edge ゲートウェイを信頼できるようにするには

1. OPC UA サーバーを設定するためのインターフェイスを開きます。
2. OPC UA サーバー管理者のユーザー名とパスワードを入力します。
3. インターフェイスで [Trusted Clients (信頼済みクライアント)] を見つけて、[AWS IoT SiteWise Gateway Client (ゲートウェイクライアント)] を選択します。
4. [Trust (信頼)] を選択します。

OPC UA クライアント証明書のエクスポート

一部の OPC UA サーバーでは、SiteWise Edge ゲートウェイを信頼するために OPC UA クライアント証明書ファイルへのアクセス権が必要です。これが OPC UA サーバーに適用される場合は、次の手順を使用して、SiteWise Edge ゲートウェイから OPC UA クライアント証明書をエクスポートできます。その後、OPC UA サーバーに証明書をインポートできます。

ソースの OPC UA クライアント証明書ファイルをエクスポートするには

1. 次のコマンドを実行して、証明書ファイルを含むディレクトリに変更します。*sitewise-work* を *aws.iot.SiteWiseEdgeCollectorOpcua* Greengrass 作業フォルダのローカルストレージパスに置き換え、*source-name* をデータソースの名前に置き換えます。

   デフォルトでは、Greengrass 作業フォルダは Linux では */greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua*、 では *C:/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua* ですMicrosoft Windows。

   ```
   cd /sitewise-work/source-name/opcua-certificate-store
   ```

2. このソースの SiteWise Edge ゲートウェイの OPC UA クライアント証明書は `aws-iot-opcua-client.pfx` ファイルにあります。

   次のコマンドを実行して、証明書を `aws-iot-opcua-client-certificate.pem` という名前の `.pem` ファイルにエクスポートします。

   ```
   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. 証明書ファイル `aws-iot-opcua-client-certificate.pem` を SiteWise Edge ゲートウェイから OPC UA サーバーに転送します。

これを行うには、`scp` プログラムなどの一般的なソフトウェアを使用して、SSH プロトコルを使用したファイルを転送することができます。詳しくは[Wikipedia] (ウィキペディア) の[Secure copy] (セキュアコピー) をご覧ください。

> **Note**
>
> SiteWise Edge ゲートウェイが Amazon Elastic Compute Cloud (Amazon EC2) で実行されていて、初めて接続する場合は、接続するための前提条件を設定する必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「SSH を使用した Linux インスタンスへの接続」を参照してください。

4. SiteWise Edge ゲートウェイを信頼するには、証明書ファイル `aws-iot-opcua-client-certificate.pem` を OPC UA サーバーにインポートします。使用するソースサーバーによりステップは異なります。サーバーのドキュメントを参照してください。

OPC UA でデータインジェスト範囲をフィルタリングする

OPC UA ソースでのデータインジェストは、スキャンモードとデッドバンド範囲を使用することで制御できます。これらの機能により、取り込むデータの種類と、サーバーと SiteWise Edge ゲートウェイがこの情報をいつ、どのように交換するかを制御できます。

品質に基づいてデータを収集またはフィルタリングする

データ品質設定を構成して、OPC UA ソースから収集されるデータを制御できます。データソースには、送信時にメタデータとして品質評価が含まれます。次のいずれか、またはすべてのオプションを選択できます。

- `Good`
- `Bad`
- `Uncertain`

NaN または null 値を処理する

SiteWise Edge は、NaN 値と null 値の収集と処理をサポートしています。

- NaN (数値ではない): 未定義または表現不可能な数値結果を表します。
- Null: 欠落しているデータを示します。

IoT SiteWise OPC UA コレクターは、BAD または UNCERTAIN 品質の NaN 値と Null 値をキャプチャします。これらの特殊な値はローカルストリームに書き込まれるため、より包括的なデータ収集が可能になります。

スキャンモードでデータ収集頻度を制御する。

OPC UA スキャンモードを設定して、OPC UA ソースからデータを収集する方法を制御できます。サブスクリプションモードとポーリングモードを選択することができます。

- サブスクリプションモード - OPC UA ソースは、スキャンレートで定義された頻度で SiteWise Edge ゲートウェイに送信するためにデータを収集します。サーバーは値が変化したときのみデータを送信するので、これが SiteWise Edge ゲートウェイがデータを受信する最大頻度です。
- ポーリングモード - SiteWise Edge ゲートウェイは、スキャンレートで定義された設定頻度で OPC UA ソースをポーリングします。サーバーは値の変化に関係なくデータを送信するため、SiteWise Edge ゲートウェイは常にこの間隔でデータを受信します。

> **Note**
>
> ポーリングモードオプションは、このソースのデッドバンド設定をオーバーライドします。

OPC UA のデータインジェストをデッドバンド範囲でフィルタリングする

OPC UA ソースプロパティグループにデッドバンドを適用して、特定のデータを AWS クラウドに送信せず、フィルタリングして廃棄することができます。デッドバンドは、OPC UA ソースから受信するデータ値の変動を予測するウィンドウを指定します。値がこのウィンドウ内にある場合、OPC UA サーバーはそれを AWS クラウドに送信しません。デッドバンドフィルタリングを使用して、処理して AWS クラウドに送信するデータの量を減らすことができます。SiteWise Edge ゲートウェイの OPC UA ソースを設定する方法については、「AWS IoT SiteWise Edge ゲートウェイの OPC UA データソース」を参照してください。

> **Note**
>
> サーバーは、デッドバンドで指定されたウィンドウ内にあるすべてのデータを削除します。この破棄されたデータは復元できません。

デッドバンドの種類

OPC UA サーバーのプロパティグループについて、2 種類のデッドバンドを指定できます。 AWS クラウドに送信するデータ量と、廃棄するデータ量を選択できます。

- パーセンテージ - 測定値の予想変動率のパーセンテージを使用してウィンドウを指定します。サーバーはこのパーセンテージから正確なウィンドウを計算し、 を超えるデータを AWS クラウドに送信します。たとえば、華氏 -100 度から華氏 +100 度の範囲のセンサーで 2% のデッドバンド値を指定すると、値が華氏 4 度以上変化したときにデータを AWS クラウドに送信するようにサーバーに指示します。

> **Note**
>
> ソースサーバーが工学的単位を定義していない場合、このウィンドウに最小および最大のデッドバンド値をオプションで指定できます。工学単位の範囲が指定されていない場合、OPC UA サーバーはデフォルトで、測定データ型の全範囲を使用とします。

- 絶対 - 正確な単位を使用してウィンドウを指定します。たとえば、センサーでデッドバンド値 2 を指定すると、値が少なくとも 2 ユニット変化したときにデータを AWS クラウドに送信するようにサーバーに指示します。通常の運用で定期的に変動が予想されるようなダイナミックな環境では、絶対デッドバンドを使用することができます。

デッドバンドタイムアウト

オプションで、デッドバンドのタイムアウト設定を行うことができます。このタイムアウト後、OPC UA サーバーは想定されるデッドバンド変動の範囲内であっても現在の測定値を送信します。タイムアウト設定を使用すると、値が定義されたデッドバンドウィンドウを超えない場合でも、AWS IoT SiteWise が常に安定したデータストリームを取り込むようにできます。

SiteWise Edge で OPC UA ノードフィルターを使用する

SiteWise Edge ゲートウェイの OPC UA データソースを定義する場合、ノードフィルターを定義できます。ノードフィルターを使用すると、SiteWise Edge ゲートウェイがクラウドに送信するデータストリームパスを制限できます。ノードフィルターを使用して、 AWS IoT SiteWiseでモデル化するデータへのパスのみを含めることで、SiteWise Edge ゲートウェイの起動時間と CPU 使用率を減らすことができます。デフォルトでは、SiteWise Edge ゲートウェイは `/Server/` で始まるパスを除くすべての OPC UA パスをアップロードします。ノードフィルターで `*` および `**` ワイルドカード文字を使用すると、1 つのフィルターに複数のデータストリームパスを含めることがで

きます。SiteWise Edge ゲートウェイの OPC UA ソースを設定する方法については、「AWS IoT SiteWise Edge ゲートウェイの OPC UA データソース」を参照してください。

> **Note**
>
> AWS IoT SiteWise は、ソースを追加または編集するたびに SiteWise Edge ゲートウェイを再起動します。SiteWise Edge ゲートウェイは、ソース設定の更新中にデータを取り込みません。SiteWise Edge ゲートウェイを再起動する時間は、SiteWise Edge ゲートウェイのソースのタグの数によって異なります。再起動時間は、数秒 (タグが少数しかない SiteWise Edge ゲートウェイの場合) から数分 (タグが多数ある SiteWise Edge ゲートウェイの場合) までさまざまです。

次の表に、OPC UA データソースのフィルターに使用できるワイルドカードを示します。

OPC UA ノードフィルターワイルドカード

ワイルドカード	説明
*	データストリームパスの単一レベルと一致します。
**	データストリームパスの複数のレベルに一致します。

> **Note**
>
> 広範なフィルターを使用してソースを設定し、後でより制限の厳しいフィルターを使用するようにソースを変更すると、 は新しいフィルターと一致しないデータの保存を AWS IoT SiteWise 停止します。

Example : ノードフィルターを使用するシナリオ

次の架空のデータストリームについて考えてみます。

- `/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

以前のデータストリームを使用して、ノードフィルターを定義し、OPC UA ソースから含めるデータを制限できます。

- この例のすべてのノードを選択するには、/ または /**/を使用します。** ワイルドカード文字を使用して、複数のディレクトリまたはフォルダを含めることができます。
- すべての PLC データストリームを選択するには、/*/*/*/PLC* または /**/PLC* を使用します。
- この例のすべてのカウンタを選択するには、/**/Counter* または /*/*/*/Counter*を使用します。
- Line 2 からすべてのカウンターを選択するには、/**/Line 2/Counter* を使用します。

サポートされていないデータ型の変換

必要に応じて、単純な配列と DateTime データ型 AWS IoT SiteWise に対して でデータ型変換を有効にします。 AWS IoT SiteWise はすべての OPC UA データ型をサポートしていません。サポートされていないデータを AWS IoT Greengrass データストリームに送信すると、そのデータは失われます。ただし、サポートされていないネイティブデータ型を文字列に変換することで、データを破棄 AWS IoT SiteWise せずに に取り込むことができます。 は変換されたデータを AWS IoT SiteWise シリアル化するため、後で独自の関数を使用して文字列を元のデータ型にダウンストリームで変換できます。

データソースのデータ型変換設定はいつでも更新でき、各データソースには独自の設定があります。

にデータソースを追加する場合 AWS IoT SiteWise コンソール、詳細設定のデータ型変換の下に 2 つのチェックボックスがあります。文字列に変換するデータ型を指定できます。

さらに、IoT SiteWise OPC UA コレクターはエッジで NaN 値または null 値を受け入れることができます。

- 単純なデータ型を持つ配列値を JSON 文字列に変換する
- DateTime 値を ISO 8601 文字列に変換する

前提条件

- IoT SiteWise OPC UA コレクターのバージョン 2.5.0 以降を使用します。

制限

これらは、 AWS IoT SiteWiseでの OPC UA データ型から文字列への変換の制限です。

- 複雑なデータ型変換はサポートされていません。
- 変換後の文字列制限は 1024 バイトです。文字列が 1024 バイトを超える場合、文字列は によって拒否されます AWS IoT SiteWise。

SiteWise Edge のデータソース認証を設定する

OPC UA サーバーで接続に認証情報が必要な場合は、 AWS Secrets Manager を使用してシークレットを作成し、SiteWise Edge ゲートウェイにデプロイできます。 は、デバイス上のシークレットを AWS Secrets Manager 暗号化して、ユーザー名とパスワードを使用するまで安全に保ちます。 AWS IoT Greengrass シークレットマネージャーコンポーネントの詳細については、 AWS IoT Greengrass Version 2 デベロッパーガイドの「シークレットマネージャー」を参照してください。

Secrets Manager シークレットへのアクセスの管理については、以下を参照してください。

- シー AWS Secrets Manager クレットへのアクセス許可を持つユーザー。
- アカウント内でリクエストが許可または拒否されているかどうかを判断する。

ステップ 1: ソース認証シークレットを作成する

を使用して AWS Secrets Manager 、データソースの認証シークレットを作成できます。シークレットで、データソースの認証詳細を含む **`username`** および **`password`** のキーと値のペアを定義します。

シークレットを作成する (コンソール)

1. AWS Secrets Manager コンソールに移動します。
2. [新しいシークレットを保存] を選択します。
3. [シークレットのタイプ] で、[その他のシークレットのタイプ] を選択します。
4. [Key/value pairs] (キー/値ペア) で、次のように実行します。

   1. 最初の入力ボックスに **username** を入力し、2 番目の入力ボックスにユーザー名を入力します。
   2. [Add row] (行の追加) を選択します。
   3. 最初の入力ボックスに **password** を入力し、2 番目の入力ボックスにパスワードを入力します。
5. [暗号化キー] では、aws/secretsmanager を選択して、[次へ] を選択します。
6. [新しいシークレットの保存] ページで、シークレット名を入力します。
7. (オプション) このシークレットの識別に役立つ説明を入力し、[次へ] を選択します。
8. (オプション) [新規シークレットを保存] ページで [自動ローテーション] をオンにします。詳細については、『AWS Secrets Manager ユーザーガイド』の「シークレットのローテーション」を参照してください。
9. ローテーションスケジュールを指定します。
10. このシークレットをローテーションできる Lambda 関数を選択し、[次へ] を選択します。
11. シークレットの設定を確認し、[保存] を選択します。

SiteWise Edge ゲートウェイが とやり取りすることを許可するには AWS Secrets Manager、SiteWise Edge ゲートウェイの IAM ロールで `secretsmanager:GetSecretValue`アクションを許可する必要があります。Greengrass コアデバイスを使用して IAM ポリシーを検索できます。IAM ポリシーの更新の詳細については、AWS Identity and Access Management 「 ユーザーガイド」の「IAM ポリシーの編集」を参照してください。

Example ポリシー

`secret-arn` の部分をステップ 2 で作成したシークレットの Amazon リソースネーム (ARN) に置き換えます。シークレットの ARN を取得する方法の詳細については、 AWS Secrets Manager ユーザーガイドの「 でシークレットを検索する AWS Secrets Manager」を参照してください。

JSON

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

```
            "secretsmanager:GetSecretValue"
        ],
        "Effect":"Allow",
        "Resource":[
            "secret-arn"
        ]
    }
]
}
```

ステップ 2: シークレットを SiteWise Edge ゲートウェイデバイスにデプロイする

AWS IoT SiteWise コンソールを使用して、SiteWise Edge ゲートウェイにシークレットをデプロイできます。

シークレットを更新する (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Gateways] を選択します。
3. [ゲートウェイ] リストからターゲットの SiteWise Edge ゲートウェイを選択します。
4. ゲートウェイ設定セクションで、Greengrass コアデバイスリンクを選択して、SiteWise Edge ゲートウェイに関連付けられた AWS IoT Greengrass コアを開きます。
5. ナビゲーションペインで、[デプロイ] を選択します。
6. ターゲットデプロイを選択し、[修正] を選択します。
7. [Specify target] (ターゲットの指定) ページで [Next] (次へ) を選択します。
8. [コンポーネントの選択] ページの [公開コンポーネント] セクションで、「選択したコンポーネントのみを表示」をオフにします。
9. aws.Greengrass.SecretManager コンポーネントを検索して選択し、[次へ] を選択します。
10. 選択されたコンポーネントのリストから aws.Greengrass.SecretManager コンポーネントを選択し、[コンポーネントの設定] を選択します。
11. [マージする設定] フィールドで、次の JSON オブジェクトを追加します。

> **Note**
>
> *secret-arn* の部分を、前のステップで作成したシークレットの ARN に置き換えます。シークレットの ARN を取得する方法の詳細については、 AWS Secrets Manager

ユーザーガイドの「 でシークレットを検索する AWS Secrets Manager」を参照してください。

```
{
"cloudSecrets":[
  {
     "arn":"secret-arn"
  }
]
}
```

12. [確認] を選択します。
13. [次へ] を選択します。
14. [詳細設定の設定] ページで、[次へ] を選択します。
15. デプロイの設定を確認し、[デプロイ] を選択します。

ステップ 3: 認証設定を追加する

AWS IoT SiteWise コンソールを使用して、SiteWise Edge ゲートウェイに認証設定を追加できます。

認証設定を追加する (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. [ゲートウェイ] リストからターゲットの SiteWise Edge ゲートウェイを選択します。
3. データソースのリストから、ターゲットデータソースを選択し、[編集] を選択します。
4. [データソースの追加] ページで [詳細設定] を選択します。
5. [認証設定] で、前のステップでデプロイしたシークレットを選択します。
6. [保存] を選択します。

SiteWise Edge ゲートウェイのパートナーデータソース

AWS IoT SiteWise Edge ゲートウェイを使用する場合、パートナーデータソースを SiteWise Edge ゲートウェイに接続し、SiteWise Edge ゲートウェイと AWS クラウドのパートナーからデータを受信できます。これらのパートナーデータソースは、 AWS とパートナーとのパートナーシップによって開発された AWS IoT Greengrass コンポーネントです。パートナーデータソースを追加すると、

AWS IoT SiteWise は、このコンポーネントを作成し、SiteWise Edge ゲートウェイにデプロイします。

> **Note**
> 各ゲートウェイのパートナーごとに 1 つのデータソースを追加できます。

パートナーデータソースを追加するには、次の手順を実行します。

1. SiteWise Edge にパートナーデータソースを追加
2. パートナーの Web ポータルに移動し (該当する場合)、SiteWise Edge ゲートウェイに接続するようにパートナーデータソースを設定します。

トピック

- セキュリティ
- SiteWise Edge ゲートウェイDockerで をセットアップする
- SiteWise Edge にパートナーデータソースを追加
- SiteWise Edge ゲートウェイパートナーのデータソースオプション

セキュリティ

お客様、パートナー間の責任共有モデルの一環として AWS、セキュリティのさまざまな側面を担当するユーザーを以下に示します。

顧客の責任

- パートナーの審査。
- パートナーに与えるネットワークアクセス権の設定。
- SiteWise Edge ゲートウェイマシンリソース (CPU、メモリ、ファイルシステム) の合理的な使用状況のモニタリング。

AWS 責任

- パートナーが必要とするものを除き、カスタマー AWS クラウドリソースからパートナーを分離します。この場合は、 AWS IoT SiteWise インジェスト。
- パートナーソリューションを SiteWise Edge ゲートウェイのマシンリソース (CPU、メモリ、ファイルシステム) の妥当な使用量に制限。

パートナーの責任

- 安全なデフォルトの使用。
- パッチやその他の適切な更新による長期にわたるソリューションの安全維持。
- 顧客データの機密保持。

SiteWise Edge ゲートウェイDockerで をセットアップする

AWS IoT SiteWise には、さまざまなプラットフォームや環境で SiteWise Edge アプリケーションを実行できるDockerイメージが用意されています。このDockerイメージは、産業機器からデータを収集、処理、 AWS クラウドに送信するために必要なすべてのコンポーネントと依存関係をカプセル化します。Docker イメージを使用することにより、サーバー、エッジデバイス、クラウドベースのコンテナサービスなどの Docker 互換ホストに SiteWise Edge アプリケーションをデプロイして実行できます。

パートナーデータソースを追加するには、Docker Engine 1.9.1 以降がローカルデバイスにインストールされている必要があります。

> **Note**
> バージョン 20.10 は、SiteWise Edge ゲートウェイソフトウェアとの互換性が確認されている最新バージョンです。

Docker がインストールされていることを確認する

Docker がインストールされていることを確認するには、SiteWise Edge ゲートウェイに接続されているターミナルから次のコマンドを実行します。

```
docker info
```

コマンドが`docker is not recognized`結果を返すか、古いバージョンの がインストールされているDocker場合は、続行する前に Docker エンジンをインストールします。

Docker をセットアップ

Docker コンテナコンポーネントを実行するシステムユーザーには、ルートまたは管理者のアクセス許可が必要です。または、ルートまたはアドミストレーター以外のユーザーとして実行Dockerするように を設定する必要があります。

Linux デバイスでは、 `ggc_user` ユーザーを`docker`グループに追加して、 なしでDockerコマンドを呼び出す必要がありますsudo。

`ggc_user`、またはDockerコンテナコンポーネントの実行に使用する非ルートユーザーを `docker`グループに追加するには、次のコマンドを実行します。

```
sudo usermod -aG docker ggc_user
```

詳細については、Docker「 エンジンの Linux のインストール後の手順」を参照してください。

SiteWise Edge にパートナーデータソースを追加

パートナーデータソースを SiteWise Edge ゲートウェイに接続するには、データソースとして追加します。データソースとして追加すると、 AWS IoT SiteWise はプライベート AWS IoT Greengrass コンポーネントを SiteWise Edge ゲートウェイにデプロイします。

前提条件

パートナーデータソースを追加するには、次の手順を実行します。

- EasyEdge と CloudRail について、パートナーとアカウントを作成し、アカウントをバインドします。
- SiteWise Edge ゲートウェイDockerで をセットアップする

パートナーデータソースがある SiteWise Edge ゲートウェイを作成する

新しい SiteWise Edge ゲートウェイを作成する場合は、セルフホスト SiteWise Edge ゲートウェイを作成する のステップを実行します。SiteWise Edge ゲートウェイを作成した後、既存の SiteWise Edge ゲートウェイにパートナーデータソースを追加する のステップに従ってパートナーデータソースを追加します。

既存の SiteWise Edge ゲートウェイにパートナーデータソースを追加する

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションで、「エッジ」セクションの「エッジゲートウェイ」を選択します。
3. パートナーデータソースを接続する SiteWise Edge ゲートウェイを選択します。
4. [データソース] で、[データソースを追加] を選択します。

5. [データソースを追加] 画面で [ソースタイプ] を選択して、SiteWise Edge ゲートウェイを接続するパートナーを選択します。各データソースには、独自の設定オプションがあります。データソースには、 AWS ソースとパートナーソースの 2 つのカテゴリがあります。

   パートナーデータソースを使用すると、ゲートウェイごとに 1 つのソースを選択できます。データソースパートナー統合オプションのリストについては、「SiteWise Edge ゲートウェイパートナーのデータソースオプション」を参照してください。最大 100 個の OPC UA データソース (AWS ソース) を追加できます。OPC UA データソースの使用を開始するには、「AWS IoT SiteWise Edge ゲートウェイの OPC UA データソース」を参照してください。

6. ソースの名前を入力します。

7. 以下のデータソースのタブを選択し、設定手順に従ってください。

   CloudRail

   CloudRail 設定の多くは、SiteWise Edge ゲートウェイのデータソースを保存した後に CloudRail ポータルで行われます。ただし、接続の承認が必要です。

   > **Note**
   > CloudRail 接続は Linux でのみ使用できます。

   1. CloudRail アカウントを作成して、 AWS IoT SiteWiseへの接続を開始します。
   2. Docker がゲートウェイにインストールされていることを確認します。詳細については、「SiteWise Edge ゲートウェイDockerで をセットアップする」を参照してください。
   3. 「アクセスとデプロイを承認」契約書を読み、[承認] を選択します。このチェックボックスをオンにすると、 AWS パートナーにデータソースへのアクセス権が付与され、 がパートナーのコンポーネント AWS にデプロイできるようになります。

   > **Note**
   > 測定プレフィックス – オプションは CloudRail ポータル内で設定されます。

> **Note**
>
> パートナーソフトウェアは、 AWS パートナーによって開発、保守、サポートされます。 AWS は、インターフェイス、設定、またはソフトウェアについて責任を負いません。

詳細については、「CloudRail」を参照してください。

EasyEdge

EasyEdge 設定の多くは、SiteWise Edge ゲートウェイのデータソースを保存した後に EasyEdge ポータルで行われます。ただし、接続の承認が必要です。

> **Note**
>
> EasyEdge 接続は Linux でのみ使用できます。

1. EasyEdge アカウントを作成して、 AWS IoT SiteWiseへの接続を開始します。
2. Docker がゲートウェイにインストールされていることを確認します。詳細については、「SiteWise Edge ゲートウェイDockerで をセットアップする」を参照してください。
3. 「アクセスとデプロイを承認」契約書を読み、[承認] を選択します。このチェックボックスをオンにすると、 AWS パートナーにデータソースへのアクセス権が付与され、 がパートナーのコンポーネント AWS にデプロイできるようになります。

> **Note**
>
> 測定プレフィックス – オプションは EasyEdge ポータル内で設定されます。

> **Note**
>
> パートナーソフトウェアは、 AWS パートナーによって開発、保守、サポートされます。 AWS は、インターフェイス、設定、またはソフトウェアについて責任を負いません。

詳細については、「EasyEdge」を参照してください。

Litmus Edge

Litmus 設定は 2 つの方法でアクティブ化できます。Litmus Edge Manager ポータルからの情報を使用して、 AWS IoT SiteWise から直接、Litmus Edge をアクティブ化します。または、 AWS IoT SiteWise から Litmus Edge Manager について、Litmus Edge を手動でアクティブ化することもできます。

> **Note**
>
> Litmus Edge 接続は Linux でのみ使用できます。

でLitmus Edgeアクティベーションコードを使用して をアクティブ化するには AWS IoT SiteWise

AWS IoT SiteWise コンソールで Litmus Edge アクティベーションコードを含む Litmus Edge データソースを追加するときには、この手順を使用します。

1. [コードを使用して今すぐアクティブ化] を選択します。追加設定オプションが表示されます。
2. Litmus Edge Manager を入力して、Litmus Edge を SiteWise Edge ゲートウェイに接続します。詳細については、Litmus Edge Manager ドキュメントの「ステップ 3a: データおよびデバイス管理エンドポイントを設定」を参照してください。
3. Litmus Edge でアクティブ化するLitmus Edge Managerアクティベーションコードを指定します。 AWS IoT SiteWise

4. 必要に応じて、 AWS IoT SiteWise に Litmus Edge Manager CA 証明書を渡します。証明書は、無許可の Litmus Edge Manager によるアクティブ化から Litmus Edge を防ぎます。
5. Docker がゲートウェイにインストールされていることを確認します。詳細については、「SiteWise Edge ゲートウェイDockerで をセットアップする」を参照してください。

> **Note**
>
> AWS IoT SiteWise はパートナーアプリケーションを Docker コンテナとしてデプロイします。アプリケーションは `NET_ADMIN` 機能でデプロイされるため、Litmus Edge Docker コンテナは Litmus Edge Manager を通じて管理できます。Litmus Edge には、デバイス上で実行するためにこの特権アクセスが必要です。Litmus Edge Docker の要件の詳細については、Litmus Edge ドキュメントの「クイックスタートガイド」の「Docker のインストール」を参照してください。

6. 「アクセスとデプロイを承認」契約書を読み、[承認] を選択します。このチェックボックスをオンにすると、 AWS パートナーにデータソースへのアクセス権が付与され、 がパートナーのコンポーネント AWS にデプロイできるようになります。

Litmus Edge を使用して手動でアクティブ化するには

1. [後で Litmus Edge でアクティブ化] を選択します。
2. Docker がゲートウェイにインストールされていることを確認します。詳細については、「SiteWise Edge ゲートウェイDockerで をセットアップする」を参照してください。

> **Note**
>
> AWS IoT SiteWise はパートナーアプリケーションを Docker コンテナとしてデプロイします。アプリケーションは `NET_ADMIN` 機能でデプロイされるため、Litmus Edge Docker コンテナは Litmus Edge Manager を通じて管理できます。Litmus Edge には、デバイス上で実行するためにこの特権アクセスが必要です。Litmus Edge Docker の要件の詳細については、Litmus Edge ドキュメント

の「クイックスタートガイド」の「Docker のインストール」を参照してください。

3. 「アクセスとデプロイを承認」契約書を読み、[承認] を選択します。このチェックボックスをオンにすると、 AWS パートナーにデータソースへのアクセス権が付与され、 がパートナーのコンポーネント AWS にデプロイできるようになります。
4. デプロイが完了したら、Litmus Edge QuickStart Guide ドキュメントの「Litmus Edge Web UI にアクセス」の指示に従います。

> **Note**
> パートナーソフトウェアは、 AWS パートナーによって開発、保守、サポートされます。 AWS は、インターフェイス、設定、またはソフトウェアについて責任を負いません。

詳細については、「Litmus Edge」を参照してください。

8. [保存] を選択します。

SiteWise Edge ゲートウェイパートナーのデータソースオプション

AWS IoT SiteWise を使用すると、産業機器、センサー、その他のサードパーティーシステムなど、さまざまなパートナーデータソースに接続してデータを取り込むことができます。パートナーデータソースを接続するには、データの送信先となるデータソースの設定、必要なアクセス許可と認証の設定 AWS IoT SiteWise、アセットモデルへのデータのマッピングなど、いくつかのステップに従う必要があります。このプロセスにより、パートナーデータが AWS IoT SiteWise 環境にシームレスに統合され、他のデータソースとともにモニタリングおよび分析できます。

このセクションでは、SiteWise Edge ゲートウェイでのサードパーティーデータソース統合に使用できるパートナーを一覧表示します。以下の情報を使用してパートナーデータソースを設定します。

> **Note**
> 各ゲートウェイのパートナーごとに 1 つのデータソースを追加できます。

CloudRail

ポータル:

https://devices.cloudrail.com/

要件

CloudRail 要件の詳細については、CloudRail ウェブサイトの「よくある質問」を参照してください。

CloudRail ドキュメント;

エッジコンピューティング: SiteWise Edge

EasyEdge

ポータル:

https://studio.easyedge.io/

要件

EasyEdge 要件 – ファイアウォールの設定に必要なエンドポイントやポートなど、EasyEdge 要件に関する情報。注: このドキュメントにアクセスするには EasyEdge アカウントが必要です。

EasyEdge ドキュメント;

EasyEdge の AWS

Litmus Edge

Litmus Edge Manager へのアクセス:

Litmus Edge にアクセスするには、Litmus Edge Manager アカウントをセットアップします。

要件

Litmus Edge 要件 – Litmus Edge をデプロイするための推奨設定とシステム要件。

Litmus ドキュメント;

- との統合 AWS IoT SiteWise
- Litmus Edge ドキュメント

AWS IoT GreengrassAWS IoT SiteWise Edge の コンポーネント

SiteWise Edge は AWS IoT Greengrass 、コンポーネントを使用して、エッジで産業データを収集、処理、送信します。これらのコンポーネントは連携して、ローカルデータ処理と AWS IoT SiteWise クラウドサービスとのシームレスな統合を可能にします。

IoT SiteWise パブリッシャー

IoT SiteWise パブリッシャーコンポーネント (`aws.iot.SiteWiseEdgePublisher`) は、以下を担当します。

- 収集したデータを AWS IoT SiteWise クラウドサービスに安全に送信する
- 接続の問題発生時のデータバッファリングと再試行の管理

SiteWise Edge のパブリッシャーの設定の詳細については、「」を参照してくださいAWS IoT SiteWise パブリッシャーコンポーネントを設定する。また、パブリッシャーコンポーネントの詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」のIoT SiteWise パブリッシャー」を参照してください。

IoT SiteWise プロセッサ

IoT SiteWise プロセッサコンポーネント (`aws.iot.SiteWiseEdgeProcessor`) は次のタスクを実行します。

- エッジでのデータ変換と計算の実行
- アセットプロパティ定義と計算のローカルでの実装
- 送信前にデータを集約またはフィルタリングしてデータ量を減らす

プロセッサコンポーネントの詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」のIoT SiteWise プロセッサ」を参照してください。

IoT SiteWise OPC UA コレクター

IoT SiteWise OPC UA コレクター (`aws.iot.SiteWiseEdgeCollectorOpcua`) コンポーネントは、次のように設計されています。

- 産業環境の OPC UA サーバーに接続する
- OPC UA データソースから効率的にデータを収集する
- OPC UA データを と互換性のある形式に変換する AWS IoT SiteWise

OPC UA コレクターコンポーネントの詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」のIoT SiteWise OPC UA コレクター」を参照してください。

IoT SiteWise OPC UA データソースシミュレーター

IoT SiteWise OPC UA データソースシミュレーターコンポーネント (`aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator`) には、次の機能があります。

- サンプルデータを生成するローカル OPC UA サーバーを起動します
- AWS IoT SiteWise ゲートウェイ上の OPC UA AWS IoT SiteWise コレクターコンポーネントによって読み取ることができるデータソースをシミュレートします
- 生成されたサンプルデータを使用して AWS IoT SiteWise 特徴量の探索を有効にします

このコンポーネントは、テストと開発の目的で特に役立ちます。これにより、物理的な機器を使用せずに産業データソースをシミュレートできます。

データソースシミュレーションコンポーネントの詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」のIoT SiteWise OPC UA データソースシミュレーター」を参照してください。

これらの AWS IoT Greengrass コンポーネントは SiteWise Edge 機能を有効にするために機能します。IoT SiteWise パブリッシャーは、データを確実にクラウドに送信し、IoT SiteWise プロセッサはローカル計算とデータ最適化を処理し、IoT SiteWise OPC UA コレクターは一般的な産業用プロトコルとの統合を容易にします。

> **Note**
> これらのコンポーネントを使用するには、エッジデバイスに AWS IoT Greengrass V2 以降がインストールされている必要があります。SiteWise Edge のパフォーマンスを最適化するには、各コンポーネントを適切に設定することが重要です。

SiteWise Edge ゲートウェイでアセットをフィルタリングする

エッジフィルタリングを使用すると、データ処理で使用するために、アセットのサブセットのみを特定のゲートウェイに送信することで、アセットをより効率的に管理できます。アセットがツリー構造、つまり親子構造の場合は、SiteWise Edge ゲートウェイの IAM ロールにアタッチした IAM ポリシーをセットアップすることで、ツリーのルートか親とその子のみを特定の SiteWise Edge ゲートウェイに送信できます。

> **Note**
>
> 既存のアセットをツリー構造に配置する場合は、構造を作成した後、構造に追加した既存のアセットそれぞれに移動し、編集 を選択し、保存 を選択して、 が新しい構造 AWS IoT SiteWise を認識することを確認します。

エッジフィルタリングをセットアップする

SiteWise Edge ゲートウェイにエッジフィルタリングをセットアップするには、ゲートウェイの IAM ロールに次の IAM ポリシーを追加します。*<root-asset-id>* を SiteWise Edge ゲートウェイに送信するルートアセットの ID に置き換えます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iotsitewise:DescribeAsset",
                "iotsitewise:ListAssociatedAssets"
            ],
            "Resource": "arn:aws:iotsitewise:*:*:asset/*",
            "Condition": {
                "StringNotLike": {
                    "iotsitewise:assetHierarchyPath": "/<root-asset-id>*"
                }
            }
        }
    ]
}
```

SiteWise Edge ゲートウェイに現在削除したいアセットがある場合は、SiteWise Edge ゲートウェイにログインし、次のコマンドを実行して、キャッシュを削除 AWS IoT SiteWise して SiteWise Edge ゲートウェイを強制的に と同期させます。

```
sudo rm /greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/sync-app/
sync_resource_bundles/edge.json
```

AWS IoT SiteWise Edge のプロキシサポートを設定し、信頼ストアを管理する

AWS IoT SiteWise Edge で、信頼ストアを設定および管理して、エッジデバイスのプロキシサポートを設定します。まず、プロキシ設定をセットアップしてから、信頼ストアを設定します。信頼ストアは、ゲートウェイのインストール中またはゲートウェイの確立後に手動で設定できます。

- プロキシ – さまざまなネットワーク環境でエッジデバイスと AWS サービス間の接続を容易にします。
- トラストストア – 信頼された証明書を管理して、安全な接続を確保します。適切な設定は、ネットワークセキュリティポリシーへの準拠、制限されたネットワーク環境での通信の有効化、エッジデバイスとクラウドサービス間のデータ転送の最適化に役立ちます。

SiteWise Edge は、さまざまなコンポーネントタイプに複数の信頼ストアを利用して、エッジデバイスからクラウドへの安全で効率的なデータフローを実現します。信頼ストアとプロキシは、既存のゲートウェイで設定することも、新しいゲートウェイの作成時にインストールプロセス中に設定することもできます。

トラストストアとプロキシの設定の要件

信頼ストアを設定するか、プロキシ設定で SiteWise Edge をインストールする前に、前提条件を満たしていることを確認してください。コンポーネントの使用状況と機能の要件に応じて、さまざまな実装要件があります。

プロキシのサポート要件

- プロキシサーバーの URL。URL には、ユーザー情報、ホストのポート番号が含まれている必要があります。例えば、`scheme://[userinfo@]host[:port]`。
  - `scheme` – HTTP または HTTPS である必要があります
  - （オプション）`userinfo` – ユーザー名とパスワードの情報
  - `host` – プロキシサーバーのホスト名または IP アドレス
  - `port` – ポート番号
- プロキシをバイパスするアドレスのリスト。

- （オプション) 自己署名証明書で HTTPS プロキシを使用している場合のプロキシ CA 証明書ファイル。

トラストストアの要件

- HTTPS プロキシを使用したデータ処理パックの完全な機能については、3 つの信頼ストアをすべて更新する必要があります。
- IoT SiteWise OPC UA コレクターと IoT SiteWise パブリッシャーのみを使用する場合は、証明書 AWS IoT Greengrass Core および Java トラストストアを最新バージョンに更新します。

トラストストアとプロキシサーバーのエッジ設定のベストプラクティス

継続的なメンテナンスとエッジ環境で最高レベルのセキュリティを維持するには：

- ネットワークセキュリティ要件に合わせて、プロキシ設定を定期的に確認および更新します。
- ゲートウェイの接続とデータフローをモニタリングして、適切なプロキシ通信を確保する
- 組織の証明書管理ポリシーに従って信頼ストアを維持および更新する
- エッジ環境での安全な通信のために推奨されるベストプラクティスを実装し、それに従うことができます。次に例を示します。
- プロキシとトラストストアの設定を文書化して運用を可視化する
- 組織のセキュリティプラクティスに従って認証情報を管理する

これらのプラクティスは、SiteWise Edge ゲートウェイの安全で信頼性の高いオペレーションを維持しながら、より広範なセキュリティポリシーと整合性を保つのに役立ちます。

AWS IoT SiteWise Edge ゲートウェイのインストール中にプロキシ設定を構成する

ゲートウェイのインストール中にプロキシサーバーと連携するように AWS IoT SiteWise Edge を設定できます。インストールスクリプトは、HTTP プロキシと HTTPS プロキシの両方をサポートし、安全なプロキシ接続のために信頼ストアを自動的に設定できます。

プロキシ設定を使用してインストールスクリプトを実行すると、いくつかの重要なタスクが実行されます。

- プロキシ URL 形式とパラメータを検証して、正しく指定されていることを確認します。
- 設定されたプロキシを介して必要な依存関係をダウンロードしてインストールします。

- プロキシ CA 証明書が指定されている場合は、 AWS IoT Greengrass ルート CA 証明書に追加され、Java KeyStore にインポートされます。
- すべてのアウトバウンド接続でプロキシを使用するように AWS IoT Greengrass (SiteWise Edge が使用する) を設定します。
- 適切なプロキシとトラストストアの設定で SiteWise Edge のインストールを完了します。

ゲートウェイソフトウェアのインストール時にプロキシ設定を構成するには

1. SiteWise Edge ゲートウェイを作成します。詳細については、「セルフホスト SiteWise Edge ゲートウェイを作成する」および「ローカルデバイスに AWS IoT SiteWise Edge ゲートウェイソフトウェアをインストールする」を参照してください。
2. 環境に適したプロキシ設定でインストールスクリプトを実行します。プレースホルダーを特定のプロキシ情報に置き換える

   以下の各項目を置き換えます。

   - `-p`、 `--proxy-url` — プロキシサーバーの URL。URL は `http`または のいずれかである必要がありますhttps。
   - `-n`, `--no-proxy` – プロキシをバイパスするアドレスのカンマ区切りリスト。
   - （オプション）`-c`、 `--proxy-ca-cert` — プロキシ CA 証明書ファイルへのパス。
   - （オプション）`-j`、 `--javastorepass` — Java KeyStore パスワード。デフォルトのパスワードは `changeit` です。

   Linux

   Linux システムの場合は、次のコマンド構造を使用します。

   ```
   sudo ./install.sh -p proxy-url -n no-proxy-addresses [-c proxy-ca-cert-path] [-
   j javastorepass]
   ```

   Windows

   PowerShell を使用するMicrosoft Windowsシステムでは、次のコマンド構造を使用します。

   ```
   .\install.ps1 -ProxyUrl proxy-url -NoProxyAddresses no-proxy-addresses [-
   ProxyCaCertPath proxy-ca-cert-path] [-JavaStorePass javastorepass]
   ```

プロキシが有効なインストール中のトラブルシューティング

SiteWise Edge ゲートウェイに関連するトラストストアの問題の解決の詳細については、「」を参照してください[プロキシが有効なインストールの問題](#)。

AWS IoT SiteWise Edge で HTTPS プロキシをサポートする信頼ストアを手動で設定する

HTTPS プロキシを介して接続するように AWS IoT SiteWise Edge コンポーネントを設定するときは、プロキシサーバーの証明書を適切な信頼ストアに追加します。SiteWise Edge は、複数の信頼ストアを使用して通信を保護します。3 つのトラストストアがあり、それらの使用はゲートウェイ実装の SiteWise Edge コンポーネントタイプによって異なります。

信頼ストアは、プロキシ設定が提供されると、インストールプロセス中に自動的に更新されます。

- [AWS IoT Greengrass Core コンポーネントの信頼ストアを設定する](#) – AWS IoT Greengrass ルート CA 証明書は、 AWS サービスの信頼性を検証するために信頼ストアに含まれています。

  この信頼ストアは、 AWS IoT Greengrass コンポーネントがプロキシを介して サービスと安全に通信 AWS し、それらのサービスの信頼性を検証するのに役立ちます。

- [Java ベースのコンポーネントの信頼ストアを設定する](#) – Java KeyStore (JKS) は、SSL/TLS 接続用の Java ベースのコンポーネントで使用される主要なトラストストアです。

  Java アプリケーションは JKS に依存して安全な接続を確立します。たとえば、Java ベースの IoT SiteWise パブリッシャーまたは IoT SiteWise OPC UA コレクターを使用している場合は、このトラストストアを設定する必要があります。これにより、クラウドにデータを送信したり、OPC UA サーバーからデータを収集したりするときに、これらのコンポーネントが HTTPS プロキシを介して安全に通信できるようになります。

- [システムレベルのコンポーネント信頼ストア設定](#) – HTTPS プロキシを使用する場合は、安全な接続を有効にするために、証明書を適切な信頼ストアに追加する必要があります。

  HTTPS プロキシを使用する場合は、安全な接続を可能にするために、証明書を適切な信頼ストアに追加する必要があります。これは、多くの場合 Rust や Go などの言語で記述されるシステムレベルのコンポーネントが、Java の JKS ではなくシステムの信頼ストアに依存するために必要です。たとえば、プロキシを介して通信する必要があるシステムユーティリティ (ソフトウェアの更新や時刻の同期など) を使用している場合は、システムレベルの信頼ストアを設定する必要があります。これにより、これらのコンポーネントとユーティリティはプロキシを介して安全な接続を確立できます。

AWS IoT Greengrass Core コンポーネントの信頼ストアを設定する

Amazon のルート CA を使用する AWS IoT Greengrass Core 関数の場合：

1. で証明書ファイルを見つける `/greengrass/v2/AmazonRootCA1.pem`
2. このファイルに HTTPS プロキシルート証明書 (自己署名) を追加します。

```
-----BEGIN CERTIFICATE-----
MIIEFTCCAv2gAwIQWgIVAMHSAzWG/5YVRYtRQOxXUTEpHuEmApzGCSqGSIb3DQEK
\nCwUAhuL9MQswCQwJVUzEPMAVUzEYMBYGA1UECgwP1hem9uLmNvbSBJbmMuMRww
... content of proxy CA certificate ...
+vHIRlt0e5JAm5\noTIZGoFbK82A0/nO7f/t5PSIDAim9V3Gc3pSXxCCAQoFYnui
GaPUlGk1gCE84a0X\n7Rp/lND/PuMZ/s8YjlkY2NmYmNjMCAXDTE5MTEyN2cM216
gJMIADggEPADf2/m45hzEXAMPLE=
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIDQTCCAimgF6AwIBAgITBmyfz/5mjAo54vB4ikPmljZKyjANJmApzyMZFo6qBg
ADA5MQswCQYDVQQGEwJVUzEPMA0tMVT8QtPHRh8jrdkGA1UEChMGDV3QQDExBBKW
... content of root CA certificate ...
o/ufQJQWUCyziar1hem9uMRkwFwYVPSHCb2XV4cdFyQzR1KldZwgJcIQ6XUDgHaa
5MsI+yMRQ+hDaXJiobldXgjUka642M4UwtBV8oK2xJNDd2ZhwLnoQdeXeGADKkpy
rqXRfKoQnoZsG4q5WTP46EXAMPLE
-----END CERTIFICATE-----
```

確立されたゲートウェイで HTTPS プロキシを設定する

ポート 8883 の代わりにポート 443 に接続することで、確立されたゲートウェイにプロキシサポートを追加できます。プロキシサーバーの使用の詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」の「Connect on port 443 or through a network proxy」を参照してください。新しいゲートウェイを作成する場合は、ゲートウェイのインストール中にプロキシ設定を設定できます。詳細については、「AWS IoT SiteWise Edge ゲートウェイのインストール中にプロキシ設定を構成する」を参照してください。

SiteWise Edge AWS IoT Greengrass で で HTTPS プロキシを使用すると、ソフトウェアは指定された URL に基づいてプロキシ接続に HTTP と HTTPS を自動的に選択します。

> **Important**
> HTTPS プロキシ経由で接続を試みる前に、必要なすべての信頼ストアを更新します。

Java ベースのコンポーネントの信頼ストアを設定する

データ処理パック内の IoT SiteWise パブリッシャー、IoT SiteWise OPC UA コレクター、Java サービスの場合、デフォルトの Java トラストストアの場所は です。 `$JAVA_HOME/jre/lib/security/cacerts`

証明書を追加するには

1. など、プロキシサーバーの証明書を保存するファイルを作成します`proxy.crt`。

   > **Note**
   > プロキシサーバーの証明書を使用して、事前に ファイルを作成します。

2. 次のコマンドを使用して、Java の信頼ストアに ファイルを追加します。

   ```
   sudo keytool -import -alias proxyCert -keystore /usr/lib/jvm/java-11-openjdk-amd64/lib/security/cacerts -file proxy.crt
   ```

3. プロンプトが表示されたら、デフォルトのパスワードを使用します。 `changeit`

システムレベルのコンポーネント信頼ストア設定

Rust、Go、およびシステムトラストストアを使用する他の言語で記述されたコンポーネントの場合：

Linux

Linux システム: 証明書を に追加する `/etc/ssl/certs/ca-certificates.crt`

Windows

Microsoft Windows システム: トラストストアを設定するには、Microsoft Ignite ドキュメントの Certificate Store の手順に従います。

Windows には、ユーザースコープとコンピュータスコープの個別のストアを含む複数の証明書ストアがあり、それぞれに複数のサブストアがあります。ほとんどの SiteWise Edge セットアップでは、`COMPUTER | Trusted Root Certification Authorities`ストアに証明書を追加することをお勧めします。ただし、特定の設定とセキュリティ要件によっては、別のストアを使用する必要がある場合があります。

トラストストアの問題のトラブルシューティング

SiteWise Edge ゲートウェイに関連するトラストストアの問題の解決の詳細については、「」を参照してください[トラストストアの問題](#)。

エッジで AWS IoT SiteWise APIs を使用する

AWS IoT SiteWise は、APIs のサブセットとエッジ固有の APIsを提供し、エッジにデプロイされたアセットモデルおよび関連するアセットとシームレスにやり取りできるようにします。これらのアセットモデルは、エッジで実行するように設定する必要があります。詳細については、「[SiteWise Edge でデータ処理するためのアセットモデルを設定する](#)」でこのセットアッププロセスの詳細手順を参照してください。

これらの API を設定したら、アセットモデルと個々のアセットに関する包括的なデータを取得できます。アセットモデル、アセット、ダッシュボード、ポータル、およびプロジェクト情報を取得すると、デプロイされたポータルとダッシュボードをモニタリングし、エッジレベルで収集されたアセットデータにアクセスするのに役立ちます。これにより、ウェブ API コールを必要と AWS IoT SiteWise せずに、 とやり取りするためのネットワーク内の中央ホストが提供されます。

トピック

- [使用可能なすべての AWS IoT SiteWise Edge デバイス APIs](#)
- [エッジデバイスで使用する AWS IoT SiteWise エッジ専用 APIs](#)
- [AWS IoT SiteWise Edge APIs で CORS を有効にする](#)
- [AWS IoT SiteWise Edge のセッションタイムアウトを設定する](#)
- [チュートリアル: AWS IoT SiteWise Edge ゲートウェイのアセットモデルを一覧表示する](#)

使用可能なすべての AWS IoT SiteWise Edge デバイス APIs

AWS IoT SiteWise には、エッジデバイスで使用するさまざまな APIs が用意されているため、デバイスでローカルにタスクを完了できます。利用可能なエッジ API には、アセットモデルの取得、アセットプロパティの作成と更新、クラウドへのデータストリームの送信などがあります。これらの

API を活用することで、断続的または限定的なネットワーク接続環境で動作するソリューションを構築できます。

使用可能な AWS IoT SiteWise APIs

エッジデバイスでは AWS IoT SiteWise APIs を使用できます。

- ListAssetModels
- DescribeAssetModel
- ListAssets
- DescribeAsset
- DescribeAssetProperty
- ListAssociatedAssets
- GetAssetPropertyAggregates
- GetAssetPropertyValue
- GetAssetPropertyValueHistory
- ListDashboards
- ListPortals
- ListProjectAssets
- ListProjects
- DescribeDashboard
- DescribePortal
- DescribeProject

使用可能なエッジ専用の API

次の API はエッジ上のデバイスでローカルに使用されます。

- Authenticate – この API を使用して、API コールに使用する SigV4 の一時認証情報を取得できます。

エッジデバイスで使用する AWS IoT SiteWise エッジ専用 APIs

エッジで使用できる AWS IoT SiteWise APIs に加えて、エッジ固有の API もあります。以下では、そうしたエッジ特有の API を説明します。

Authenticate

SiteWise Edge ゲートウェイから認証情報を取得します。LDAP または Linux ユーザープールを使用してローカルユーザーを追加するか、システムに接続する必要があります。ユーザーの追加の詳細については、「LDAP」または「Linux ユーザープール」を参照してください。

リクエストの構文

```
POST /authenticate HTTP/1.1
Content-type: application/json
{
  "username": "string",
  "password": "string",
  "authMechanism": "string"
}
```

URI リクエストのパラメータ

リクエストでは URI パラメータを使用しません。

リクエスト本文

リクエストは以下の JSON 形式のデータを受け入れます。

username

リクエスト呼び出しの検証に使用するユーザー名。

タイプ: 文字列

必須: はい

password

認証情報をリクエストしているユーザーのパスワード。

タイプ: 文字列

必須: はい

authMechanism

ホストでこのユーザーを検証するための認証方法。

タイプ: 文字列

有効な値: `ldap`、`linux`、`winnt`

必須: はい

レスポンスの構文

```
HTTP/1.1 200
Content-type: application/json
{
  "accessKeyId": "string",
  "secretAccessKey": "string",
  "sessionToken": "string",
  "region": "edge"
}
```

レスポンス要素

アクションが成功すると、サービスは HTTP 200 レスポンスを返します。

以下のデータが JSON 形式で返されます。

accessKeyId

仮のセキュリティ認証情報を識別するアクセスキー ID。

長さ制限: 最小長は 16 です。最大長は 128 です。

パターン: `[\w]*`

secretAccessKey

リクエストへの署名に使用できるシークレットアクセスキー。

タイプ: 文字列

sessionToken

仮の認証情報を使用するために、ユーザーがサービス API に渡す必要があるトークン。

タイプ: 文字列

region

API コールのターゲットにするリージョン。

型: 定数 - edge

エラー

IllegalArgumentException

提供された本文ドキュメントの形式に誤りがあるため、リクエストが拒否されました。エラーメッセージは、それぞれのエラーの説明です。

HTTP ステータスコード: 400

AccessDeniedException

ユーザーには、現在の ID プロバイダーに基づく有効な認証情報がありません。このエラーメッセージは認証メカニズムの説明です。

HTTP ステータスコード: 403

TooManyRequestsException

リクエストが認証試行回数の上限に達しました。エラーメッセージには、新しい認証試行が行われるまで待機する時間が含まれています。

HTTP ステータスコード: 429

AWS IoT SiteWise Edge APIs で CORS を有効にする

AWS IoT SiteWise Edge APIs、ウェブアプリケーションは異なるドメイン間で APIsと直接通信できます。これにより、中間サーバーや回避策なしで、シームレスな統合、リアルタイムのデータ交換、クロスドメインデータアクセスが可能になります。CORS 設定は、許可されたオリジンを指定するように設定でき、クロスオリジンアクセスを確実に制御できます。

> **Note**
>
> CORS はバージョン 3.3.1 以降で使用できます。この機能は `aws.iot.SiteWiseEdgeProcessor`コンポーネントのバージョン 3.3.1 以降で使用できます。詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」のAWS IoT SiteWise 「プロセッサ」を参照してください。

SiteWise Edge APIs

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. CORS を有効にする SiteWise Edge ゲートウェイを選択します。 AWS IoT Greengrass V2 デプロイタイプで CORS を有効にできます。
4. ゲートウェイ設定セクションで、関連する Greengrass コアデバイスを選択します。
5. 「デプロイ」タブの「Greengrass デバイス」で、適切なデプロイリンクを選択します。
6. 「アクション」で「デプロイの修正」を選択し、「デプロイの修正」を選択します。

> **Important**
>
> 修正された CORS 対応設定を作成すると、デバイスの現在の設定が置き換えられます。

7. ステップ 1 でターゲットを指定し、デプロイを識別するためのオプションの名前を指定します。
8. ステップ 2、コンポーネントの選択 - オプションで、現在のすべての選択をそのままにして、次へ を選択できます。
9. ステップ 3 で、コンポーネントを設定する - オプションで、aws.iot.SiteWiseEdgeProcessor を選択し、コンポーネントを設定する を選択します。
10. 「設定の更新」セクションの「マージする設定」の下に、次の JSON を入力します。

```
{
    "AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN": "*"
}
```

> **Note**
>
> の値*として を使用すると、すべてのオリジンAWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGINが許可されます。本番環境では、セキュリティを向上させるために正確なオリジン URLsを指定することをお勧めします。

11. [確認] を選択してください。
12. 次へ を選択して、Step5、確認 に到達するまで残りのステップに進みます。
13. 設定の変更を確認し、デプロイを選択して SiteWise Edge ゲートウェイに変更を適用します。

> **Note**
>
> または、グローバル環境変数を AWS IoT SiteWise ゲートウェイ`AWS_SITEWISE_EDGE_ACCESS_CONTROL_ALLOW_ORIGIN*`の に設定することで CORS を有効にすることもできます。

> **Note**
>
> 認証されたプロキシの場合、 は、個別の `username`および `url`フィールドとしてではなく、プロキシ設定の `password`フィールドに含める`userinfo`必要があります。

デプロイが完了すると、SiteWise Edge API で CORS が有効になり、指定されたオリジンが API に対してクロスオリジンリクエストを行うことができます。

AWS IoT SiteWise Edge のセッションタイムアウトを設定する

SiteWise Edge では、SiteWise Edge API のセッションタイムアウトを設定できます。この機能は、指定された期間後に非アクティブなセッションを自動的に終了することでセキュリティを強化します。このセクションでは、 を使用してセッションタイムアウトを設定するプロセスについて説明します AWS IoT SiteWise コンソール。

> **Note**
>
> セッションタイムアウト設定は、 `aws.iot.SiteWiseEdgeProcessor`コンポーネントのバージョン 3.4.0 以降で使用できます。詳細については、「 AWS IoT Greengrass Version 2 デベロッパーガイド」のAWS IoT SiteWise 「プロセッサ」を参照してください。

SiteWise Edge ゲートウェイのセッションタイムアウトを設定するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. セッションタイムアウトを設定する SiteWise Edge ゲートウェイを選択します。

> **Note**
> AWS IoT Greengrass V2 デプロイタイプでセッションタイムアウトを設定できます。

4. ゲートウェイ設定セクションで、関連する Greengrass コアデバイスを選択します。
5. 「デプロイ」タブの「Greengrass デバイス」で、適切なデプロイリンクを選択します。
6. Actions で Revise を選択します。警告を読み、デプロイの修正を選択します。

> **Important**
> 変更されたセッションタイムアウト設定を作成すると、デバイスの現在の設定が置き換えられます。

7. ステップ 1 でターゲットを指定し、オプションの Name を指定して改訂されたデプロイを識別し、Next を選択します。
8. ステップ 2 でコンポーネントを選択する - オプションで、現在の選択をすべてそのままにして、次へ を選択できます。
9. ステップ 3 で、コンポーネントを設定する - オプションで、aws.iot.SiteWiseEdgeProcessor を選択し、コンポーネントを設定する を選択します。
10. 「設定の更新」セクションの「マージする設定」の下に、次の JSON を入力します。

```
{
    "AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES": "240"
}
```

11. の値を分`AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES`単位で設定します。セッションタイムアウト値は 1 分 ~ 10080 分 (7 日間) です。デフォルト値は 240 分 (4 時間) です。
12. [確認] を選択してください。
13. 次へ を選択して、ステップ 5、確認 に到達するまで残りのステップに進みます。
14. 設定の変更を確認し、デプロイを選択して SiteWise Edge ゲートウェイに変更を適用します。

> **Note**
>
> または、グローバル環境変数 AWS_SITEWISE_EDGE_SESSION_TIMEOUT_MINUTES を SiteWise Edge ゲートウェイの目的の値 (分単位) に設定することで、セッションタイムアウトを設定できます。

デプロイが完了すると、新しいセッションタイムアウト設定が SiteWise Edge API に適用されます。

チュートリアル: AWS IoT SiteWise Edge ゲートウェイのアセットモデルを一覧表示する

利用可能な AWS IoT SiteWise APIs のサブセットとエッジ固有の APIs を使用して、エッジ上のアセットモデルとそのアセットを操作できます。このチュートリアルでは、 AWS IoT SiteWise Edge ゲートウェイへの一時的な認証情報を取得し、SiteWise Edge ゲートウェイでアセットモデルのリストを取得する方法について説明します。

前提条件

このチュートリアルの手順では、さまざまなツールを利用できます。そうしたツールを使用するには、対応する前提条件がインストールされていることを確認してください。

このチュートリアルを完了するには、以下が必要です。

- A がデプロイされて、AWS IoT SiteWise Edge セルフホストゲートウェイの要件 が稼働中
- ポート 443 経由で同じネットワーク内の SiteWise Edge ゲートウェイにアクセスします。
- OpenSSL のインストール
- （ AWS OpsHub for AWS IoT SiteWise) for AWS OpsHubAWS IoT SiteWise アプリケーション
- (curl) curl のインストール
- (Python) urllib3 のインストール
- (パイソン) Python3 のインストール
- (Python) Boto3 のインストール
- (Python) BotoCore のインストール

ステップ 1: SiteWise Edge ゲートウェイサービスの署名入り証明書を取得する

SiteWise Edge ゲートウェイで使用できる API への TLS 接続を確立するには、信頼済みの証明書が必要です。この証明書は、OpenSSL または AWS OpsHub for を使用して生成できます AWS IoT SiteWise。

OpenSSL

> **Note**
> このコマンドを実行するには、OpenSSL がインストールされている必要があります。

ターミナルを開き、次のコマンドを実行して、SiteWise Edge ゲートウェイから署名入り証明書を取得します。`<sitewise_gateway_ip>` を SiteWise Edge ゲートウェイの IP に置き換えます。

```
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

は AWS OpsHub に使用できます AWS IoT SiteWise。詳細については、「SiteWise Edge ゲートウェイを管理」を参照してください。

このチュートリアルでは、ダウンロードした SiteWise Edge ゲートウェイ証明書への絶対パスが使用されています。次のコマンドを実行して、証明書の完全なパスをエクスポートします。`<absolute_path_to_certificate>` の部分はその証明書へのパスに置き換えます。

```
export PATH_TO_CERTIFICATE='<absolute_path_to_certificate>'
```

ステップ 2: SiteWise Edge ゲートウェイのホスト名を取得する

> **Note**
> このコマンドを実行するには、OpenSSL がインストールされている必要があります。

このチュートリアルを完了するには、SiteWise Edge ゲートウェイのホスト名が必要です。SiteWise Edge ゲートウェイのホスト名を取得するには、以下を実行します。<sitewise_gateway_ip> を SiteWise Edge ゲートウェイの IP に置き換えます。

```
openssl s_client -connect <sitewise_gateway_ip>:443 </dev/null 2>/dev/null | grep -Po
 'CN = \K.*'| head -1
```

後で使用するため、次のコマンドを実行してホスト名をエクスポートします。<your_edge_gateway_hostname> を SiteWise Edge ゲートウェイのホスト名に置き換えます。

```
export GATEWAY_HOSTNAME='<your_edge_gateway_hostname>'
```

ステップ 3: SiteWise Edge ゲートウェイの仮の認証情報を取得する

署名付き証明書と SiteWise Edge ゲートウェイのホスト名がわかったので、ゲートウェイで API を実行できるように仮の認証情報を取得する必要があります。これらの認証情報は、 AWS OpsHub のを通じて取得 AWS IoT SiteWise することも、API を使用して SiteWise Edge ゲートウェイから直接取得することもできます。 APIs

> **Important**
>
> 認証情報は 4 時間ごとに有効期限が切れるため、SiteWise Edge ゲートウェイで API を使用する直前に認証情報を取得する必要があります。認証情報を 4 時間以上キャッシュしないでください。

for を使用して一時的な認証情報を取得する AWS OpsHubAWS IoT SiteWise

> **Note**
>
> AWS OpsHubAWS IoT SiteWise アプリケーション用の をインストールする必要があります。

for AWS OpsHub AWS IoT SiteWise アプリケーションを使用して一時的な認証情報を取得するには、次の手順を実行します。

1. アプリケーションにログインします。

2. [設定] を選択します。
3. [認証] で [認証情報のコピー] を選択します。
4. 環境に合ったオプションを展開し、[コピー] を選択します。
5. 後で使用するため認証情報を保存しておきます。

SiteWise Edge ゲートウェイの API を使用して仮の認証情報を取得する

SiteWise Edge ゲートウェイの API を使用して仮の認証情報を取得するには、Python スクリプトまたは curl を使用できます。まず、SiteWise Edge ゲートウェイのユーザー名とパスワードが必要です。SiteWise Edge ゲートウェイは SigV4 の認証と認可を使用します。ユーザーの追加の詳細については、「LDAP」または「Linux ユーザープール」を参照してください。これらの認証情報は、API を使用するために必要な SiteWise Edge ゲートウェイのローカル認証情報を取得するために、次のステップで使用されます AWS IoT SiteWise APIs 。

Python

> **Note**
> urllib3 と Python3 がインストールされている必要があります。

Python を使用して認証情報を取得する

1. get_credentials.py というファイルを作成し、そのファイルに次のコードをコピーします。

```
'''
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. ターミナルから get_credentials.py を実行します。<gateway_username> および <gateway_password> の部分は作成した認証情報に置き換えます。

```
python3 get_credentials.py -e https://$GATEWAY_HOSTNAME -c $PATH_TO_CERTIFICATE
 -u '<gateway_username>' -p '<gateway_password>' -m 'linux'
```

curl

> **Note**
>
> curl がインストールされている必要があります。

curl を使用して認証情報を取得する

1. ターミナルから次のコマンドを実行します。<gateway_username> および <gateway_password> の部分は作成した認証情報に置き換えます。

```
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"
}'
```

レスポンスは次のようになります。

```
{
    "username": "sweuser",
    "accessKeyId": "<accessKeyId>",
    "secretAccessKey": "<secretAccessKey>",
    "sessionToken": "<sessionToken>",
    "sessionExpiryTime": "2022-11-17T04:51:40.927095Z",
    "authMechanism": "linux",
    "role": "edge-user"
}
```

2. ターミナルから次のコマンドを実行します。

```
export AWS_ACCESS_KEY_ID=<accessKeyId>
export AWS_SECRET_ACCESS_KEY=<secretAccessKey>
export AWS_SESSION_TOKEN=<sessionToken>
export AWS_REGION=edge
```

ステップ 4: SiteWise Edge ゲートウェイ上のアセットモデルのリストを取得する

署名入り証明書、SiteWise Edge ゲートウェイのホスト名、および SiteWise Edge ゲートウェイの仮の認証情報が揃ったので、`ListAssetModels` API を使用して SiteWise Edge ゲートウェイ上のアセットモデルのリストを取得できます。

Python

> **Note**
>
> Python3、Boto3、および BotoCore がインストールされている必要があります。

Python を使用してアセットモデルのリストを取得する

1. list_asset_model.py というファイルを作成し、そのファイルに次のコードをコピーします。

```
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. ターミナルから list_asset_model.py を実行します。

```
python3 list_asset_model.py
```

curl

> **Note**
>
> curl がインストールされている必要があります。

curl を使用してアセットモデルのリストを取得する

ターミナルから、次のコマンドを実行します。

```
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"
```

レスポンスは次のようになります。

```
{
    "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 Edge で SiteWise Edge ゲートウェイをホストする

AWS IoT SiteWise Edge アプリケーションを使用して、Siemens Industrial Edge でゲートウェイをホストします。 AWS IoT Greengrass V2と同様に、Siemens Industrial Edge の SiteWise Edge を使用して、製造プロセスを最適化したり、運用ワークフローを改善したりできます。

Siemens Industrial Edge デバイスから AWS アカウントにデータを取り込むには、デバイスで SiteWise Edge ゲートウェイを実行します。これを行うには、SiteWise AWS IoT SiteWise Edge サポートチームから Edge アプリケーションへのアクセスをリクエストします。次に、[Siemens 産業用 Edge デバイス - 新規] をデプロイターゲットにして、SiteWise Edge ゲートウェイリソースを作成します。次に、設定ファイルをダウンロードし、Siemens Industrial Edge Management ポータル経由でアプリケーションにアップロードします。必要な Siemens リソースのセットアップ方法など、Siemens Industrial Edge でアプリケーションを実行する方法の詳細については、Siemens ドキュメントの「産業用 Edge とは」を参照してください。

> **Note**
>
> Siemens は SiteWise Edge のベンダーまたはサプライヤーではありません。Siemens Industrial Edge Marketplace は独立したマーケットプレイスです。

トピック

- セキュリティ
- Siemens Secure Storage と AWS IoT SiteWise Edge アプリケーション
- Siemens Industrial Edge デバイスの送信先
- プレビューアプリケーションからの移行
- トラブルシューティング
- AWS IoT SiteWise エッジアプリケーション変更ログ
- AWS IoT SiteWise Edge アプリケーションの要件
- Siemens Industrial Edge 用のゲートウェイを作成する
- アプリケーションの Siemens Databus user を作成する
- AWS IoT SiteWise Edge アプリケーションにアクセスする
- Siemens デバイスにアプリケーションをインストールする
- AWS IoT SiteWise Edge アプリケーション設定を更新する

セキュリティ

お客様、パートナー間の責任共有モデルの一環として AWS、セキュリティのさまざまな側面を担当するユーザーを以下に示します。

顧客の責任

- パートナーの審査。
- パートナーに与えるネットワークアクセス権の設定。
- SiteWise Edge を実行しているデバイスを物理的に保護します。

AWS 責任

- お客様の AWS クラウドリソースからパートナーを分離する。

パートナーの責任

- 安全なデフォルトの使用。
- パッチやその他の適切な更新による長期にわたるソリューションの安全維持。
- 顧客データの機密保持。
- パートナーマーケットプレイスで利用可能な他のアプリケーションを確認します。

Siemens Secure Storage と AWS IoT SiteWise Edge アプリケーション

AWS IoT SiteWise Edge アプリケーションの実行に必要な認証情報とシークレットを保護するために、 Siemens Industrial Edgeは認証情報をデバイスに安全に保存するためのメカニズムを提供します。これらの認証情報を安全に保存するためのサポートがない場合、 AWS IoT SiteWise Edge アプリケーションはデバイスで実行されません。Secure Storage サポートの欠落による実行の失敗は、ログファイルに記録されます。

AWS IoT SiteWise Edge アプリケーションをインストールして実行するには、次の最小 OS バージョンが必要です。デバイスを最新バージョンにアップグレードして、アプリケーションをインストールします。

- 仮想デバイスの場合: IEVD バージョン 1.19 以降
- 物理デバイスの場合: IED-OS バージョン 2.2 以降

の AWS IoT SiteWise Edge アプリケーションSiemens Industrial Edgeは、デバイスをアップグレードするまで実行されません。

Siemens Industrial Edge デバイスの送信先

で AWS IoT SiteWise Edge アプリケーションを使用する場合Siemens Industrial Edge、送信先は、さらなる分析と配布 AWS IoT SiteWise のために に送信する前にデータを準備するのに役立ちます。Amazon S3 を使用してバッファされたデータインジェストのデータ送信先設定を構成することも、リアルタイムのデータインジェストを使用することもできます。どちらも、Siemens Industrial Edgeデバイスデプロイターゲットのパスフィルターを使用して MQTT トピックをサブスクライブできます。

ゲートウェイのSiemens Industrial Edgeデプロイターゲットは、2 つの主要なデータ処理方法をサポートしています。

- AWS IoT SiteWise リアルタイム設定 - データは収集 AWS IoT SiteWise 時に に直接送信されます
- AWS IoT SiteWise Amazon S3 設定を使用してバッファリング - データは Amazon S3 に送信される前に一時的に収集され、バッチで保存されます。

これらのオプションの設定の詳細については、Amazon S3 を使用して AWS IoT SiteWise バッファされた送信先を追加する「」および「」を参照してくださいAWS IoT SiteWise Edge リアルタイム送信先を追加する。

パスフィルターのプレフィックス

Siemens Industrial Edge デプロイターゲットを使用するゲートウェイのパスフィルターは、トピックとデータストリーム名の両方を組み合わせて、データの一意の識別子を作成します。データストリーム名を組み合わせたトピックは、Siemens Industrial Edgeゲートウェイのプレフィックスと呼ばれます。これは、パスフィルターが MQTT トピックのみに基づくセルフホストゲートウェイとは異なります。

Example Siemens データストリームのパスフィルター構造

Siemens データストリームの一般的なパスフィルターには、トピックパスとデータストリーム名の両方が含まれます。

```
ie/d/device1/application1/datastream1
```

コードの説明は以下のとおりです。

- `ie/d/` は Siemens データストリームに必要なプレフィックスです

- `device1/application1` は階層パスを表します
- `datastream1` は特定のデータストリーム名です

> **Note**
>
> Siemens Industrial Edge データストリームを使用する場合は、データストリームに関する完全な情報を受信するために、メタデータ (`ie/m/`) トピックとデータ (`ie/d/`) トピックの両方をパスフィルターに含めるようにしてください。

送信先とパスフィルター

MQTT 対応ゲートウェイの宛先とパスフィルターの詳細については、以下のトピックを参照してください。

- AWS IoT SiteWise Edge の送信先を理解する
- AWS IoT SiteWise Edge リアルタイム送信先を追加する
- Amazon S3 を使用して AWS IoT SiteWise バッファされた送信先を追加する
- AWS IoT SiteWise Edge 送信先のパスフィルターを理解する
- AWS IoT SiteWise Edge 送信先にパスフィルターを追加する
- AWS IoT SiteWise Edge の送信先を管理する

プレビューアプリケーションからの移行

プレビューフェーズで Siemens Industrial Edge 上で SiteWise Edge を実行した場合は、プレビューバージョン (バージョン 1.0.1) から最新バージョンにアップグレードする必要があります。移行するには、以下を実行します。

1. 新しい SiteWise Edge ゲートウェイを作成します。詳細については、「Siemens Industrial Edge 用のゲートウェイを作成する」を参照してください。
2. 新しいゲートウェイごとに新しい Siemens Databus user を作成します。詳細については、「アプリケーションの Siemens Databus user を作成する」を参照してください。
3. IED にバージョン 1.0.1 AWS IoT SiteWise Edge ゲートウェイアプリケーションをアンインストールします。

> **Note**
>
> AWS IoT SiteWise Edge アプリケーションのプレビューバージョンで以前に使用された AWS IoT SiteWise アセットを再設定するときに、データフローの中断に備えます。データ履歴が保持されている間は、新しいゲートウェイを再インストールするときにデータが失われる可能性があります。

4. AWS IoT SiteWise コンソール のプレビュー中に作成した SiteWise Edge ゲートウェイを削除します。
5. 新しいゲートウェイ設定ファイルを使用して AWS IoT SiteWise 、 Edge ゲートウェイアプリケーションを IED にインストールします。詳細については、「Siemens デバイスにアプリケーションをインストールする」を参照してください。

> **Important**
>
> 新しいゲートウェイをインストールすると、SiteWise Edge アプリケーションのプレビューバージョンが上書きされます。バージョン 2.0.0 をインストールした後、バージョン 1.0.1 に戻ることはできません。

新しいゲートウェイと Siemens Databus user を設定すると、データはプロパティに流れます。

SiteWise Edge アプリケーションをバージョンから直接アップグレードすることもできます。ただし、新しいゲートウェイ設定がやはり必要です。

トラブルシューティング

Siemens Industrial Edge デバイス上の SiteWise Edge ゲートウェイのトラブルシューティングについては、「での AWS IoT SiteWise Edge アプリケーションのトラブルシューティング Siemens Industrial Edge」を参照してください。

AWS re:Post にアクセスして、質問に対する回答を見つけることもできます。

AWS IoT SiteWise エッジアプリケーション変更ログ

次の表に、 AWS IoT SiteWise Edge アプリケーションの各バージョンの変更点を示します。

バージョン	変更	

3.0.0	新機能
	
	• にインストールされているゲートウェイの AWS IoT SiteWise コンソールに追加の設定オプションを追加しますSiemens Industrial Edge。
	  • Amazon S3 の送信先を追加および設定します。
	  • AWS IoT SiteWise リアルタイム送信先を削除します。
	  • データルーティングを制御するには、送信先ごとにフィルターを追加します。
	• Siemens IE インストーラの IoT SiteWise パブリッシャーからすべてのデータを送信するグローバルプレフィックス設定のオプションを追加します。プレフィックスはフィルタリング後に適用されます。
	
	バグ修正と機能向上
	
	• AWS IoT SiteWise Edge アプリケーションの最初の試行が失敗した場合に、Siemens Industrial Edge Databus への接続を再試行する機能が追加されました。	

2.0.1	バグ修正と機能向上 • 起動時に AWS 認証情報を取得できなかった場合にアプリが失敗状態になり、終了する問題を修正しました。認証情報の取得が成功するまで再試行するサポートを追加しました。	
2.0.0	• AWS IoT SiteWise Edge アプリケーションが一般利用可能になりました。 • アプリケーションには、Siemens IEVD バージョン 1.19 または Siemens IED-OS バージョン 2.2 が必要です。 • パフォーマンスの向上: メモリと CPU 使用量の削減。 • デバッグの改善点: オプションの設定ファイルをアップロードして、デバッグログを有効にできるようになりました。	

バージョン	変更
	• セキュリティの強化: アプリケーションは SecureStorage API を使用して、認証情報をデバイスに安全に保存します。 • Docker ダイジェスト値: `sha256:4a960f29234a190ebb5224c1fd0f3e99faafccc4cb3d93ca13fef247b6656d18`
1.0.1	初回リリース

AWS IoT SiteWise Edge アプリケーションの要件

で AWS IoT SiteWise Edge を実行するにはSiemens Industrial Edge、以下が必要です。

- Siemens デジタルエクスチェンジプラットフォームアカウント。
- Siemens Industrial Edge Hub (iehub) アカウント。
- Siemens Industrial Edge Management インスタンス。
  - IE アプリ設定サービス。詳細については、Siemens Industrial Edge Management ドキュメントの Installing the IE App Configuration Service manually を参照してください。
- AWS IoT SiteWise Edge アプリケーションのバージョン 2.0.1 以降へのアクセス。詳細については、「AWS IoT SiteWise Edge アプリケーションにアクセスする」を参照してください。
- Siemens Industrial Edge デバイス (IED) または Siemens Industrial Edge 仮想デバイス (IEVD)。
  - ハードウェア要件として、少なくとも 15 GB のディスク容量。
  - 1 GB の RAM と 1 GB の追加スワップメモリ。
  - ポート 443 および 8883 でアウトバウンドトラフィックを許可するデバイス設定。
  - x86-64 ビットプロセッサ。
  - Siemens Industrial Edge Management バージョン 1.13.10 以降。
  - デバイスの Siemens Secure Storage 要件への準拠。
    - 仮想デバイスでは、IEVD バージョン 1.19 以降。
    - 物理デバイスでは、IED-OS バージョン 2.2 以降。
  - Docker Compose の最新バージョン。
  - Docker Engine バージョン 18.091 以降。

- 必要なドメインアクセス。詳細については、「AWS IoT SiteWise エンドポイント」を参照してください。

Siemens Industrial Edge 用のゲートウェイを作成する

適切な Siemens アカウントと IEM インスタンスを取得したら、デプロイタイプの Siemens Edge ゲートウェイ Siemens 産業 Edge デバイスを作成できます。

> **Note**
>
> Siemens Industrial Edge Management でデバイスを実行するためのすべての要件を満たしていることを確認します。詳細については、「AWS IoT SiteWise Edge アプリケーションの要件」を参照してください。

設定ファイルを作成するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. [Create gateway (ゲートウェイの作成)] を選択します。
4. デプロイタイプで、Siemens Industrial Edge デバイスを選択します。
5. SiteWise Edge ゲートウェイの名前を入力するか、 AWS IoT SiteWiseによって生成された名前を使用します。
6. (オプション) [高度な設定] で、以下を実行します。
   - AWS IoT Core モノの名前を入力するか、 によって生成された名前を使用します AWS IoT SiteWise。
7. [Create gateway (ゲートウェイの作成)] を選択します。
8. SiteWise Edge ゲートウェイ設定ファイルの生成ダイアログボックスで、「生成とダウンロード」を選択します。 AWS IoT SiteWise Edge アプリケーションの設定に使用する設定ファイル AWS IoT SiteWise を自動的に生成します。

> **Important**
>
> Edge AWS IoT SiteWise アプリケーションをバックアップおよび復元するには、ゲートウェイ設定ファイルを使用します。SiteWise Edge ゲートウェイ設定ファイルを

AWS Secrets Manager に保存して、ファイルを安全に保存および管理します。Secrets Manager は、機密情報を安全に保存、管理、取得します。

アプリケーションの Siemens Databus user を作成する

AWS IoT SiteWise Edge on Siemens Industrial Edge はSiemens Databusアプリケーションからデータを取り込みます。SiteWise Edge を Siemens Databus に接続するには、 AWS IoT SiteWise に安全に転送するデータへのアクセスを提供する Siemens Databus user が必要です。開始するには、Siemens Databus user を作成し、SiteWise Edge アプリケーションに認証情報を指定します。

Siemens Databus user を作成するには

1. Siemens Industrial Edge Management インスタンスで、[プラットフォームアプリケーション] セクションの [Edge 管理] を選択します。
2. [データ接続] アイコンを選択します。
3. [データバス] を選択します。接続されたデバイスのリストが表示されます。
4. AWS IoT SiteWise Edge アプリケーションに接続するデバイスを選択します。
5. [Launch] (起動する) を選択します。選択したデバイスの Databus Configurator が表示されます。
6. [ユーザー] で Edge デバイスのユーザーを作成します。ユーザーの作成の詳細については、Siemens Industrial Edge Management ドキュメントの「ユーザー」を参照してください。
7. この Siemens Databus がアクセスできるトピックを選択します。これらのトピックでは、 AWS IoT SiteWise Edge がアクセスできる内容を制限します。

> **Important**
>
> Siemens Databus user がアクセスできるすべてのトピックは AWS IoT SiteWiseに発行されます。

> **Note**
>
> Siemens Databus user は、データトピックとメタデータトピックの両方にアクセスする必要があります。`ie/d` で始まるトピックはデータトピックです。`ie/m` で始まるトピッ

クはメタデータトピックです。SiteWise Edge が各トピックのデータとメタデータの両方にアクセスできるように、トピックをペアで共有します。

8. Siemens Databus 設定に適切なアクセス許可を設定します。

Siemens Databus 設定を作成したら、 AWS IoT SiteWise Edge アプリケーションを にインストールできますSiemens Industrial Edge Management。詳細については、「Siemens デバイスにアプリケーションをインストールする」を参照してください。

オプションで、Siemens Industrial Edgeゲートウェイの送信先とパスフィルターを設定することもできます。詳細については、「送信先とパスフィルター」を参照してください。

AWS IoT SiteWise Edge アプリケーションにアクセスする

で AWS IoT SiteWise Edge アプリケーションにアクセスするにはSiemens Industrial Edge、SiteWise Edge サポートチームへのアクセスをリクエストする E メールを送信します。

E メールには、以下の情報を記載してください。

- 貴社の名称と連絡先情報
- 会社名
- Siemens Industrial Edge テナント ID

Siemens デバイスにアプリケーションをインストールする

の SiteWise AWS IoT SiteWise Edge サポートチームに E メールを送信して Edge アプリケーションにアクセスしたらSiemens Industrial Edge、アプリケーションを のインスタンスに割り当てますSiemens Industrial Edge Management。その後、デバイスに AWS IoT SiteWise Edge アプリケーションをインストールできます。

AWS IoT SiteWise Edge アプリケーションをインストールするには

1. 内で提供されたDockerダイジェストが、「」に記載されている最新バージョンSiemens Industrial Edge Managementと一致していることを確認しますAWS IoT SiteWise エッジアプリケーション変更ログ。

   Siemens のDockerダイジェスト値の検索の詳細については、SiemensドキュメントのSiemens Industrial Edge「 デバイス」の「アプリの管理」を参照してください。

   Siemens Industrial Edge Management は、一度に 1 つのバージョンの AWS IoT SiteWise Edge アプリケーションをサポートします。このステップを実行して、Siemens Industrial Edgeデバイスに AWS IoT SiteWise Edge アプリケーションをインストールする前に、アプリケーションの最新バージョンを使用していることを確認します。
2. AWS IoT SiteWise Edge アプリケーションを Siemens Industrial Edge Management に割り当てます。詳細については、Siemensドキュメントの「産業用エッジ管理」セクションの「アプリケーションの管理」を参照してください。
3. [Edge 管理] 内で、[AWS IoT SiteWise Edge] のカタログを参照して選択します。
4. [インストール] を選択します。

   > **Note**
   >
   > Contact Us ボタンが表示された場合は、それを選択し、手順に従って の AWS IoT SiteWise Edge アプリケーションへのアクセスをリクエストしますSiemens Industrial Edge。詳細については、「AWS IoT SiteWise Edge アプリケーションにアクセスする」を参照してください。
5. スキーマ設定オプションで [Databus_Configuration] を選択します。
6. データバス設定の [ユーザー名] と [パスワード] を入力します。Siemens Databus user の作成の詳細については、「アプリケーションの Siemens Databus user を作成する」を参照してください。
7. [Databus_Configuration] の横にある小さな丸い灰色のチェックマークアイコンを選択して、アイコンの色を緑色にします。

   > **Note**
   >
   > 入力設定は、チェックマークアイコンが灰色から緑色に変わった場合にのみ適用されます。それ以外の場合、入力設定は無視されます。

8. [次へ] を選択して、ゲートウェイ設定ファイルをアップロードできる [その他の設定] に移動します。
9. ゲートウェイ設定ファイルをアップロードする場所として SiteWise_Edge_Gateway_Config を選択します。

> **Note**
>
> SiteWise_Edge_Support_Config_Optional ではなく、SiteWise_Edge_Gateway_Config を選択してください。

10. アプリケーションをインストールするデバイスを選択します。
11. [今すぐインストール] を選択します。

オプションで、データを AWS クラウドにエクスポートするようにパブリッシャーコンポーネントを設定できます。詳細については、「AWS IoT SiteWise パブリッシャーコンポーネントの設定」を参照してください。

Siemens Industrial Edge ゲートウェイの送信先を設定するには、「」を参照してください送信先とパスフィルター。

AWS IoT SiteWise Edge アプリケーション設定を更新する

で AWS IoT SiteWise Edge アプリケーション設定を更新するときに考慮すべき点がいくつかありますSiemens Industrial Edge。

> **Note**
>
> AWS IoT SiteWise Edge アプリケーション設定を変更するには、アプリケーションの再起動が必要です。

AWS IoT SiteWise Edge アプリケーションを再起動する理由

- Siemens Databus user AWS IoT SiteWise Edge アプリケーションの新しい 。
- ゲートウェイ設定ファイル (SiteWise_Edge_Gateway_Config ファイル) の変更。
- プロキシ設定の更新 (完全な IEVD 再起動も必要)
- デバッグ問題のためにデバッグログを有効にするには

アプリケーションの再起動

1. Siemens Industrial Edge Management インスタンスで、[プラットフォームアプリケーション] セクションの [Edge 管理] を選択します。
2. [My Installed Apps] を選択します。
3. AWS IoT SiteWise Edge アプリケーションを選択します。
4. [Restart (再起動)] を選択

送信先とパスフィルター

AWS IoT SiteWise Edge の送信先は、産業用データがエッジデバイスからクラウドにどのように流れるかを管理するための柔軟で効率的な方法を提供します。このセクションでは、送信先を設定し、

パスフィルターを使用して特定のデータストリームをルーティングし、ユースケースに適した送信先タイプを選択する方法について説明します。

でホストされている AWS IoT SiteWise Edge アプリケーションと組み合わせて使用するセルフホスト MQTT 対応、V3 ゲートウェイ、およびゲートウェイで、送信先とパスフィルターを使用できます Siemens Industrial Edge。送信先とパスフィルターは、 Classic Streams、V2 ゲートウェイでは機能しません。

トピック

- AWS IoT SiteWise Edge の送信先を理解する
- AWS IoT SiteWise Edge 送信先のパスフィルターを理解する
- AWS IoT SiteWise Edge リアルタイム送信先を追加する
- Amazon S3 を使用して AWS IoT SiteWise バッファされた送信先を追加する
- AWS IoT SiteWise Edge 送信先にパスフィルターを追加する
- AWS IoT SiteWise Edge の送信先を管理する

AWS IoT SiteWise Edge の送信先を理解する

AWS IoT SiteWise Edge の送信先を使用して、ソースデータの送信先を決定します。コスト効率、低レイテンシー、ストレージ要件など、必要な特定の特性に基づいてデータ送信先を選択できます。、当社のパートナー AWS IoT SiteWise、またはカスタムアプリケーションによってキャプチャされたデバイスデータを統合して、エッジでパスフィルター (トピック) を発行およびサブスクライブします。その後、デバイスデータをモデル化、転送、クラウドに保存できます。

> **Note**
>
> セルフホストゲートウェイのすべての送信先機能を最大限に活用するには、最新バージョンの IoT SiteWise パブリッシャーと IoT SiteWise OPC UA コレクターにアップグレードします。ストリームのサポートは、既存のセットアップとの互換性を維持するために、 Classic ストリーム、V2 ゲートウェイで継続されます。詳細については、AWS IoT SiteWise Edge のクラシックストリーム、V2 ゲートウェイを参照してください。

トピック

- SiteWise Edge の送信先がデータ管理を強化する方法
- 送信先タイプ

- ゲートウェイバージョン間で送信先機能を比較する
- 送信先の制限
- SiteWise Edge 送信先のユースケース

SiteWise Edge の送信先がデータ管理を強化する方法

エッジから AWS IoT SiteWise にリアルタイムで、または Amazon S3 を使用してバッチでデータをエクスポートします。

送信先により AWS IoT SiteWise 、環境の柔軟性とスケーラビリティが向上します。送信先は、一元化されたデータ管理モデルを実装し、ソースはデータを中央システムに発行します。送信先は、パスフィルターを使用してデータの送信先を決定します。送信先は複数のパスフィルターをサブスクライブできます。

MQTT 対応ゲートウェイは、セルフホスト型か で実行されているかにかかわらずSiemens Industrial Edge、ローカル通信に MQTT を使用し、フィルターが に設定されたデフォルトのリアルタイム送信先が付属しています#。つまり、デフォルトでは、すべてのトピックのすべてのメッセージが AWS IoT SiteWise リアルタイム送信先に発行されます。詳細については、「AWS IoT SiteWise Edge 送信先のパスフィルターを理解する」を参照してください。各ゲートウェイに 1 つのリアルタイム送信先を追加できます。

送信先タイプ

ゲートウェイの送信先を設定する場合、2 つの主なオプションがあります。 を使用したリアルタイム設定と AWS IoT SiteWise、Amazon S3 を使用したバッファ設定です。各送信先タイプには、独自の設定と考慮事項のセットがあります。

AWS IoT SiteWise リアルタイム設定

これを選択すると、データを AWS IoT SiteWise ホット階層ストレージに直接送信し、リアルタイムでデータの取り込みとモニタリングを容易にします。リアルタイム設定は、特にゲートウェイでクラウドとの接続の問題が発生した場合に、データフローを管理します。接続が失われると、データはゲートウェイに一時的にローカルに保存されます。接続が再確立されると、保存されたデータは自動的にクラウドに送信されます。

データ発行プロセスのさまざまな側面を調整できます。例えば、ローカルに保存されるデータの最大量、再接続時にクラウドにデータを送信する速度、ストレージが容量に達した後にデータを削除するタイミングなどです。

AWS IoT SiteWise ストレージ階層の詳細については、「」を参照してください[でデータストレージを管理する AWS IoT SiteWise](#)。

AWS IoT SiteWise Amazon S3 設定を使用してバッファリング

この送信先タイプを使用すると、ゲートウェイでローカルにデータをバッファし、定期的に Amazon S3 バケットにバッチで送信できます。データは効率的な Parquet 形式で保存され、分析ワークロード用に最適化されています。データが Amazon S3 に保存されたら、 にインポートして、 AWS IoT SiteWise 保存、処理、分析を行うことができます。

データをバッチで取り込み、履歴データを費用対効果の高い方法で保存するには、このオプションを選択します。任意の Amazon S3 バケットの場所と、Amazon S3 にデータをアップロードする頻度を設定できます。 AWS IoT SiteWiseへの取り込み後にデータを処理する方法を選択することもできます。SiteWise と Amazon S3 の両方でデータを使用できるようにするか、Amazon S3 から自動的にデータを削除するかを選択できます。

ゲートウェイバージョン間で送信先機能を比較する

MQTT 対応ゲートウェイの送信先機能は、データフロー管理を合理化します。送信先は、さまざまなエンドポイントへのデータルーティングを一元的に設定することで、データ管理を簡素化します。このアプローチにより、複雑な個別のストリーム設定が不要になり、システム全体の柔軟性と管理が容易になります。

これに対して、Classic ストリーム、V2 ゲートウェイ、SiteWise Edge は、データソースからパブリッシャーに AWS IoT Greengrass ストリーム経由でデータを送信し、データソースごとにデータ送信先を個別に設定します。

AWS IoT SiteWise 送信先機能を使用すると、パブリッシャーのルーティング設定が統合されます。送信先設定を使用すると、送信先とパスフィルターを一元的に管理できます。必要に応じて、送信先の追加、パスフィルターの管理、不要なフィルターや送信先の削除を簡単に行うことができます。

さらに、送信先機能は、産業用 IoT アプリケーションで広く使用されている業界標準のプロトコルである MQTT (Message Queuing Telemetry Transport) を使用します。この MQTT の導入により AWS IoT SiteWise 、さまざまなデバイスやシステムとの統合が容易になります。

送信先の制限

SiteWise Edge ゲートウェイの送信先に関する現在の制限は次のとおりです。

- データ処理パックは MQTT 対応ゲートウェイではサポートされていません。

- データ型のサポートは AWS IoT SiteWise データ型に制限されています。データ型変換を有効にする方法については、「」を参照してください[サポートされていないデータ型の変換](#)。

SiteWise Edge 送信先のユースケース

SiteWise Edge の送信先は、さまざまなアプリケーションで使用されます。主な例をいくつか示します。

産業オートメーション, リアルタイムモニタリングと予測メンテナンス

産業環境では、ファクトリーフロアのセンサーとデバイスが SiteWise Edge にデータを発行できます。送信先は、関連するデータをフィルタリングしてルーティングするように設定できるため、マシンのパフォーマンスをリアルタイムでモニタリングおよび分析できます。パスフィルターを使用して関連する MQTT トピックにサブスクライブし、データを処理してから、処理されたデータを公開できます。これにより、処理されたデータを AWS クラウド分析サービスまたはオンプレミスシステムに選択的にルーティングできます。その後、製造元は予測メンテナンス戦略を実装し、生産プロセスを最適化して、ダウンタイムを短縮できます。

スマートビル, エネルギー効率と稼働率の最適化

ビルディングオートメーションシステムは、データストリームを生成して、HVAC システム、照明、アクセスコントロールなど、ビルディングのさまざまな側面を監視および制御します。SiteWise Edge を使用すると、これらのデータストリームを取り込み、処理し、さまざまな送信先にルーティングできます。施設マネージャーは、関連するデータをフィルタリングして転送するように送信先を設定できます。これにより、データのプライバシーとコンプライアンスを確保しながら、エネルギー効率の測定や占有率の最適化などの高度な機能が可能になります。

これらのユースケースは、SiteWise Edge の Destinations 機能を活用して、さまざまな業界でデータを効率的に取り込み、処理し、ルーティングする方法を示しています。これにより、データのプライバシーとコンプライアンスを確保しながら、リアルタイムモニタリング、予測メンテナンス、エネルギー効率、リモート診断などの高度な機能が可能になります。

AWS IoT SiteWise Edge 送信先のパスフィルターを理解する

トピック

- パスフィルターの要件
- パスフィルターのベストプラクティス
- OPC UA サーバーのパスフィルター

- パスフィルター名の特殊文字

各送信先は、 AWS IoT SiteWise または Amazon S3 にデータをルーティングするように設定されています。パスフィルターを使用すると、送信先の MQTT メッセージを受信するときにフィルタリングする特定のデータを選択できます。パスフィルターは、データストリームの論理名を表し、目的の MQTT トピックのサブスクリプションとして機能します。

MQTT では、データはトピックに編成されます。トピックは、スラッシュ () で区切られた階層文字列です/。たとえば、デバイスは温度データをトピック に発行する場合がありますhome/livingroom/sensor1/temperature。ここで、 home/livingroom/sensor1はセンサーのパスまたは論理名を表し、 temperatureは発行されるデータ型を表します。

パスフィルターを使用して、ワイルドカード (+ および) を使用して特定のトピックまたはさまざまなトピックをサブスクライブできます#。+ ワイルドカードは、トピック階層の 1 つのレベルに一致します。たとえば、 home/+/sensor1/temperatureは home/livingroom/sensor1/temperatureと に一致しますhome/bedroom/sensor1/temperature。# ワイルドカードは、フィルターの最後に使用すると、複数のレベルに一致します。

パスフィルター名内の MQTT 仕様では通常許可されていないさまざまな文字を使用することもできます。これらの文字は、名前内で使用する場合、ワイルドカードとして機能しません。 は、元の命名構造を維持しながら MQTT コンプライアンスを確保するために、エンコードを使用してこれらの文字を AWS IoT SiteWise 変換します。この機能は、他のシステムからの既存の命名規則に対応するために特に便利です。詳細については、「パスフィルター名の特殊文字」を参照してください。

適切なパスフィルターを慎重に選択することで、特定の宛先に送信されるデータを制御できます。パスフィルターを使用して、データフローを IoT システムの要件に合わせて調整します。

パスフィルターの要件

を使用してパスフィルターを入力するときは AWS IoT SiteWise コンソール、次の点に注意してください。

- パスフィルターは新しい行で区切られ、各行は個別のパスフィルターを表します。
- 個々のパスフィルターは 1 ~ 65,535 バイトです。
- パスフィルターを空白にすることはできません。
- Null 値 (U+0000) は使用できません。
- 一度に最大 100 個のパスフィルターまたは 65,535 文字を入力できます。どちらの制限にも達します。

- 全体的な制限は、ゲートウェイ上のすべての送信先の合計で 20,000 パスフィルターです。
- パスフィルター名内で %、#、+、および $文字を使用できますが、 AWS IoT SiteWise は自動的に URI エンコーディングに変換します。

パスフィルターのベストプラクティス

AWS IoT SiteWise 送信先のパスフィルターを作成するときは、データを効果的に管理するために以下の戦略を検討してください。

- デバイス階層をミラーリングするようにフィルターを構造化します。例えば、製造設定 では、 は異なる生産ラインのすべてのマシンからデータを`factory/+/machine/#`キャプチャします。
- デバイスタイプ、場所、または関数に特定のレベルを使用します。例えば、`factory/assembly-line/robot/temperature`。または、スマート農業では、 を使用して`farm/+/crop/+/moisture`、さまざまな分野のさまざまな作物の湿度レベルをモニタリングします。
- ワイルドカードを戦略的に活用する: 1 つのレベルでバリエーション+に を使用し、それ以降のすべてのレベル#をキャプチャします。例えば、 は`building/+/+/energy-consumption`建物内のさまざまなゾーンや床のエネルギー使用量を追跡します。これは、最初の がすべてのフロアを+キャプチャし、2 番目の がすべてのゾーンを+キャプチャすることを前提としています。
- 関連するデータをキャプチャするのに十分なだけ、将来の変更に対応するのに十分な柔軟性を持つフィルターを作成して、特異性と柔軟性のバランスを取ります。たとえば、 `site/+/equipment-type/+/measurement`では、フィルター構造を変更せずに新しいサイトや機器タイプを追加することができます。

フィルターを徹底的にテストして、意図したデータをキャプチャし、IoT システムのアーキテクチャと目標に沿っていることを確認します。

OPC UA サーバーのパスフィルター

OPC UA サーバーの場合、パスフィルターは OPC UA タグ名に対応している必要があります。パスフィルターの最終レベルは、OPC UA タグ名と完全に一致する必要があります。たとえば、OPC UA タグが の場合`Device1.Temperature`、パスフィルターは になります`factory/line1/Device1.Temperature`。などの前述のレベルでワイルドカードを使用して、複数の生産ラインでタグを`factory/+/Device1.Temperature`キャプチャできます。パスフィルター名に特殊文字がある場合は、パスフィルター名の特殊文字「」を参照してください。

パスフィルター名の特殊文字

AWS IoT SiteWise は、OPC UA などの産業用プロトコルで一般的に使用される文字に対応します。通常、標準 MQTT トピック名では許可されません。この機能により、産業システムと MQTT ベースのアーキテクチャをよりスムーズに統合できます。

> **Note**
>
> 特殊文字の処理は統合と移行に役立ちますが、互換性を高めるために、可能な場合は新しい実装の標準の MQTT 命名規則に従うことをお勧めします。

産業ソースからデータを受信すると、 は特殊文字の URI エンコードを使用してトピック名を AWS IoT SiteWise 正規化します。

- `%` は になります `%25` (最初にエスケープ文字としてエンコードされます)
- `#` が `%23` になります
- `+` が `%2B` になります
- `$` は になります `%24` (トピックの開始時にのみ)

このエンコーディングにより、これらの特別な MQTT 文字を含むソースデータを MQTT トピック名として安全に使用できると同時に、元の産業命名規則を保持できます。

Example : パスフィルター名の特殊文字

産業トピック名が AWS IoT SiteWise パスフィルターにどのように表示されるかの例を次に示します。

- `Factory1/Line#2/Sensor+3` が `Factory1/Line%232/Sensor%2B3` になります
- `Plant%A/Unit$1/Temp` が `Plant%25A/Unit%241/Temp` になります
- `Site1/#Section/+Node` が `Site1/%23Section/%2BNode` になります

でサブスクリプションを作成したり、トピック名を表示したりすると AWS IoT SiteWise、エンコードされていない元のバージョンが表示されます。エンコードは自動的に処理され、MQTT コンプライアンスが確保されます。

AWS IoT SiteWise Edge リアルタイム送信先を追加する

リアルタイム送信先タイプを使用すると、デバイスやゲートウェイから IoT データをリアルタイムで AWS IoT SiteWise ストレージに直接ストリーミングできます。このオプションは、バッチ処理やバッファリングを必要とせずに、生成時にデータの即時取り込みと処理を必要とするユースケースに最適です。データが継続的にストリーミングされるため、各ゲートウェイで設定できるリアルタイム送信先は 1 つだけです AWS IoT SiteWise。

> **Note**
>
> TQVs重複すると、二重課金が発生する可能性があります。

リアルタイム送信先を追加するには

AWS IoT SiteWise コンソールまたは AWS CLI を使用して、SiteWise Edge MQTT 対応ゲートウェイにリアルタイムの送信先を追加します。

Console

1. AWS IoT SiteWise コンソールを開きます。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. 送信先を追加するゲートウェイを選択します。
4. 送信先 セクションで、送信先の追加 を選択します。
5. 送信先の追加ページで、送信先の詳細を入力します。
   a. 送信先名フィールドの送信先の名前。
   b. 送信先タイプのAWS IoT SiteWise リアルタイムを選択します。
6. ゲートウェイの発行順序を設定するには、発行順序を古いデータを最初に発行するか、最新のデータを最初に発行します。デフォルトでは、ゲートウェイは最も古いデータを最初に公開します。
7. 最大バッチ待機時間を使用して、パブリッシャーがデータのバッチを送信する前に待機する最大時間を設定します AWS IoT SiteWise。この設定はエイリアスごとに適用されます。データは、次のいずれかになるまでローカルに保存されます。
   - 設定時間が経過した、または
   - エイリアスに対して 10 個の time-quality-value (TQV) エントリが受信されます

どの条件が満たされた場合でも、最初にバッチがクラウドに送信されます。

8. アップロードされたデータを圧縮するには、データのアップロード時に圧縮を有効にするチェックボックスをオンにします。クラウドにアップロードする前にゲートウェイにデータを圧縮させると、帯域幅の使用量が減少します。
9. 期限切れのパブリッシャーデータをフィルタリングするには、期限切れのデータを除外チェックボックスをオンにします。この選択は、アクティブデータと現在のデータのみをに送信します AWS IoT SiteWise。
10. カットオフ期間フィールドに、データセット内でデータが期限切れと見なされる頻度を入力します。データが分単位または日単位でカウントされるかどうかを判断できます。最小カットオフ期間は 5 分です。最大カットオフ期間は 7 日間です。
11. 必要に応じて、ローカルストレージ設定を構成します。

    a. 保持期間の頻度を設定する – ゲートウェイがカットオフ期間より古いデータをローカルに保存する時間。最小保持期間は 1 分です。

       最大保持期間は 30 日間で、ローテーション期間以上です。

    b. ローテーション期間の設定 – 1 つのファイルのカットオフ期間より古いデータを保存するときに指定する時間間隔。ゲートウェイは、各ローテーション期間の終了時に 1 バッチ分のデータをローカルディレクトリ (`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`) に転送します。

       保持期間は 1 分以上で、保持期間と同じである必要があります。

    c. ストレージ容量 (GB) 値を指定して、ローカルに保存されるデータの最大サイズを GB 単位で設定します。データが決定された最大ローカルストレージサイズを超える場合、ゲートウェイは最初に最も古いデータの削除を開始します。ゲートウェイは、ローカルに保存されているデータのサイズがクォータ以下になるまで削除を続けます。

       ストレージ容量は 1 GB 以上である必要があります。

12. パスフィルターを送信先に追加します。詳細については、AWS IoT SiteWise Edge 送信先にパスフィルターを追加する を参照してください。

詳細については、「送信先タイプ」を参照してください。

AWS CLI

Example : AWS IoT SiteWise 新しいリアルタイム送信先を作成する

UpdateGatewayCapabilityConfiguration API を使用してパブリッシャーを設定します。

`capabilityNamespace` パラメータを `iotsitewise:publisher:3` に設定します。

```
{
    "sources": [
        {
            "type": "MQTT"
        }
    ],
    "destinations": [
        {
            "type": "SITEWISE_REALTIME",
            "name": "your-destination-name",
            "config": {
                "publishingOrder": "TIME_ORDER",
                "enableCompression": true,
                "maxBatchWaitTime": "10s"
            },
            "filters": [
                {
                    "type": "PATH",
                    "config": {
                        "paths": [
                            "#"
                        ]
                    }
                }
            ]
        }
    ]
}
```

既存の AWS IoT SiteWise リアルタイム送信先を更新するには、まず `DescribeGatewayCapabilityConfiguration` API を使用して を検索します`destinationId`。

Example : AWS IoT SiteWise リアルタイム送信先を更新する

UpdateGatewayCapabilityConfiguration API を使用してパブリッシャーを設定します。

`capabilityNamespace` パラメータを `iotsitewise:publisher:3` に設定します。

```
{
    "sources": [
        {
            "type": "MQTT"
        }
    ],
    "destinations": [
        {
            "id": "your-existing-destination-id",
            "type": "SITEWISE_REALTIME",
            "name": "your-destination-name",
            "config": {
                "publishingOrder": "TIME_ORDER",
                "enableCompression": true,
                "dropPolicy": {
                    "cutoffAge": "7d",
                    "exportPolicy": {
                        "retentionPeriod": "7d",
                        "rotationPeriod": "6h",
                        "exportSizeLimitGB": 10
                    }
                },
                "maxBatchWaitTime": "10s"
            },
            "filters": [
                {
                    "type": "PATH",
                    "config": {
                        "paths": [
                            "#"
                        ]
                    }
                }
            ]
        }
    ]
}
```

以下の設定オプションは、 `iotsitewise:publisher:3`名前空間を使用するゲートウェイに固有のものです。

`sources`

産業機器から にデータを転送するためのデータソースを定義します AWS IoT SiteWise。MQTT 対応ゲートウェイの場合は、 を使用します`MQTT`。

タイプ: オブジェクトの配列

必須: はい

`destinations`

データの送信先を定義します。送信先は、Amazon S3 を使用してリアルタイムまたはバッファリングされます。少なくとも 1 つの送信先オブジェクトが必要ですが、空の配列を追加できます。ゲートウェイごとに 1 つのリアルタイム送信先を設定できます。詳細については、「AWS IoT SiteWise Edge の送信先を理解する」を参照してください。

タイプ: オブジェクトの配列

必須: はい

`id`

送信先の一意の識別子。既存の送信先 ID を指定するか、空白のままにすることができます。ID を指定しない場合、UUID はデフォルトで生成されます。

タイプ: 文字列

必須: いいえ

`type`

タイプ: 宛先 オプションには、 `SITEWISE_REALTIME`と が含まれます`SITEWISE_BUFFERED`。

- `SITEWISE_REALTIME` – データをリアルタイムでストレージに直接 AWS IoT SiteWise 送信します。
- `SITEWISE_BUFFERED` – Parquet 形式でデータをバッチで Amazon S3 に送信し、 AWS IoT SiteWise ストレージにインポートします。

タイプ: 文字列

必須: はい

name

送信先の一意の名前。

タイプ: 文字列

必須: はい

config

JSON 形式の送信先タイプに固有の設定。設定は、リアルタイム送信先とバッファされた送信先によって異なります。

型: オブジェクト

必須: はい

publishingOrder

データが公開される順序を決定します。データはタイムスタンプに基づいて発行されます。オプションには、 TIME_ORDERと が含まれますRECENT_DATA。

- TIME_ORDER (デフォルト) – 古いデータを最初に公開します。
- RECENT_DATA – 最新のデータを最初に公開します。

タイプ: 文字列

必須: いいえ

enableCompression

に設定するとtrue、送信前のデータ圧縮が有効になります AWS IoT SiteWise。クラウドにアップロードする前にゲートウェイにデータを圧縮させると、帯域幅の使用量が減少します。デフォルト値は true です。

型: ブール値

必須: いいえ

dropPolicy

古いデータを処理する方法を定義します。

タイプ: オブジェクト

必須: いいえ

- `cutoffAge`

  日、時間、分で指定された、公開されるデータの最大期間。例えば、`7d`、`1d7h16m` です。指定したよりも古いデータは送信されません AWS IoT SiteWise。

  カットオフ期間より前のデータはクラウドに公開されません。カットオフ期間は 5 分から 7 日の間でなければなりません。

  `d`カットオフ期間を指定すると `m`と `h`、を使用できます。`m` は分数、`h`は時数、`d`は日数を表します。

  タイプ: 文字列

  必須: はい

- `exportPolicy`

  カットオフ期間を超えるデータを処理する方法を定義します。

  タイプ: オブジェクト

  必須: いいえ

  - `retentionPeriod`

    SiteWise Edge ゲートウェイは、指定された保持期間に保存された後、カットオフ期間より前のデータをローカルストレージから削除します。保持期間は 1 分から 30 日の間で、ローテーション期間と同じかそれ以上でなければなりません。

    `d`保持期間を指定すると、`m`と `h`、を使用できます。`m` は分数、`h`は時数、`d`は日数を表します。

    タイプ: 文字列

    必須: いいえ

  - `rotationPeriod`

    カットオフ期間より前のデータをバッチ処理して 1 つのファイルとして保存する時間間隔。SiteWise Edge ゲートウェイは、各ローテーション期間の終了時に

1 バッチ分のデータを以下のローカルディレクトリ (`/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/exports`) に転送します。ローテーション期間は 1 分より大きく、保持期間と同じかそれ以下でなければなりません。

ローテーション期間を指定すると、`m`と `h`、`d`を使用できます。`m` は分数、`h`は時数、`d`は日数を表します。

タイプ: 文字列

必須: いいえ

- `exportSizeLimitGB`

  ローカルに保存されるデータの最大許容サイズ (GB 単位)。このクォータを超えた場合、SiteWise Edge ゲートウェイは、ローカルに保存されているデータのサイズがクォータ以下になるまで、最も古いデータから削除を開始します。このパラメータの値は、1 以上である必要があります。

  タイプ: 整数

  必須: いいえ

`maxBatchWaitTime`

パブリッシャーがデータのバッチを送信する前に待機する最大時間を設定します AWS IoT SiteWise。この設定はエイリアスごとに適用されます。データは、次のいずれかになるまでローカルに保存されます。

- 設定時間が経過した、または
- エイリアスに対して 10 個の time-quality-value (TQV) エントリが受信されます

`m`カットオフ時間を指定する`d`には、、`h`、および を使用します。`m` は分数、`h`は時数、`d`は日数を表します。

タイプ: 文字列

必須: いいえ

`filters`

データに適用するフィルタ。少なくとも 1 つのフィルターが必要です。

タイプ: 文字列

必須: はい

`type`

フィルターのタイプ。PATH を使用します。

タイプ: 文字列

必須: はい

`config`

JSON 形式のフィルタータイプに固有の設定。少なくとも 1 つのオブジェクトが必要ですが、配列は空にすることができます。

型: オブジェクト

必須: はい

- `paths`

  パスフィルターの配列。詳細については、「AWS IoT SiteWise Edge 送信先のパスフィルターを理解する」を参照してください。デフォルトのパスは # です。

  型: 文字列の配列

  必須: はい

Amazon S3 を使用して AWS IoT SiteWise バッファされた送信先を追加する

バッファされた送信先タイプを使用すると、データをリアルタイムで必要と AWS IoT SiteWise しない場合に、取り込みコストを節約できます。これにより、IoT データを Amazon S3 バケットに一時的に保存してからインポートできます AWS IoT SiteWise。または、データを S3 にアップロードして保存することもできます。インポートする予定があるかどうかは関係ありません AWS IoT SiteWise。これは、デバイスやゲートウェイからデータをバッチ処理およびバッファしてから取り込む場合に便利です AWS IoT SiteWise。このオプションでは、データは設定された頻度で Parquet 形式で指定された S3 バケットにアップロードされます。その後、このデータを AWS IoT SiteWise ストレージにインポートして、さらなる分析と処理を行うことができます。

Amazon S3 を使用してバッファされた送信先を追加するには

AWS IoT SiteWise コンソールまたは AWS CLI を使用して、Amazon S3 を使用してデータをバッファする送信先を SiteWise Edge ゲートウェイに追加します。

Console

AWS Management Console を使用して、Amazon S3 を使用してバッファされた AWS IoT SiteWise 送信先を追加します。

1. AWS IoT SiteWise コンソールを開きます。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. 送信先を追加するゲートウェイを選択します。
4. 送信先 セクションで、送信先の追加 を選択します。
5. 送信先の追加ページで、送信先の詳細を入力します。

   a. 送信先名フィールドの送信先の名前。

   b. AWS IoT SiteWise Amazon S3 for Destination type を使用してバッファされた を選択します。Amazon S3 を使用して AWS IoT SiteWise バッファされた は、データを Parquet 形式でバッチで Amazon Simple Storage Service に送信し、データをストレージに AWS IoT SiteWise インポートします。

6. ゲートウェイデータを保存する場所の Amazon S3 URL を入力します。パスを参照するには、Browse S3 を選択します。バケットが追加されたら、表示を選択してバケットを表示することもできます。
7. 時間枠を入力し、データのアップロード頻度の時間増分を選択して、ゲートウェイが Amazon S3 にデータをアップロードする頻度を指定します。 頻度値は 0 より大きく、30 日以下である必要があります。
8. データストレージ設定で、ゲートウェイデータをインポートした後の対処方法を決定します AWS IoT SiteWise。データストレージについては、次の 2 つの決定事項があります。

   - インポートしたデータを AWS IoT SiteWise ストレージにコピーする場合は、データをストレージにコピーチェックボックスをオンにします。このオプションは、設定された Amazon S3 バケットからインポートされたデータを AWS IoT SiteWise ストレージに複製します。
   - Amazon S3 バケットから AWS IoT SiteWise ストレージにデータをインポートすることを選択した場合は、インポートが完了した後にインポートされたデータを削除するかどうかを指定することもできます。Amazon S3 からデータを削除チェックボックスをオンにして、 AWS IoT SiteWise ストレージにインポートした後に、設定された Amazon S3 バケットからインポートされた日付を削除します。

9. パスフィルターを送信先に追加します。詳細については、AWS IoT SiteWise Edge 送信先にパスフィルターを追加する を参照してください。

AWS CLI

Example : Amazon S3 を使用してバッファされた新しい AWS IoT SiteWise 送信先を作成する

UpdateGatewayCapabilityConfiguration API を使用してパブリッシャーを設定します。

`capabilityNamespace` パラメータを `iotsitewise:publisher:3` に設定します。

```
{
    "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 S3 を使用してバッファされた AWS IoT SiteWise 送信先を更新する

既存の AWS IoT SiteWise リアルタイム送信先を更新するには、まず `DescribeGatewayCapabilityConfiguration` API を使用して を検索します`destinationId`。

パブリッシャーの名前空間: `iotsitewise:publisher:3`

```
{
    "sources": [
      {
        "type": "MQTT"
      }
    ],
    "destinations": [
      {
        "id": "your-existing-destination-id",
        "type": "SITEWISE_BUFFERED",
        "name": "your-s3-destination-name",
        "config": {
          "targetBucketArn": "arn:aws:s3:::amzn-s3-demo-bucket/Optional/SomeFolder",
          "publishPolicy": {
            "publishFrequency": "15m",
            "localSizeLimitGB": 10
          },
          "siteWiseImportPolicy": {
            "enableSiteWiseStorageImport": true,
            "enableDeleteAfterImport": true,
            "bulkImportJobRoleArn": "arn:aws:iam::123456789012:role/your-role-name"
          }
        },
        "filters": [
          {
            "type": "PATH",
            "config": {
              "paths": [
                "#"
              ]
            }
          }
        ]
```

```
      }
    ]
  }
```

以下の設定オプションは、 `iotsitewise:publisher:3`名前空間を使用する MQTT 対応ゲートウェイに固有のものです。

`sources`

産業機器から にデータを転送するためのデータソースを定義します AWS IoT SiteWise。MQTT 対応ゲートウェイの場合は、 を使用します`MQTT`。

タイプ: オブジェクトの配列

必須: はい

`destinations`

データの送信先を定義します。送信先は、Amazon S3 を使用してリアルタイムまたはバッファリングされます。少なくとも 1 つの送信先オブジェクトが必要ですが、空の配列を追加できます。ゲートウェイごとに 1 つのリアルタイム送信先を設定できます。詳細については、「AWS IoT SiteWise Edge の送信先を理解する」を参照してください。

タイプ: オブジェクトの配列

必須: はい

`id`

送信先の一意の識別子。既存の送信先 ID を指定するか、空白のままにすると、送信先に対して新しい ID が自動的に生成されます。

タイプ: 文字列

必須: いいえ

`type`

タイプ: 宛先 オプションには、 `SITEWISE_REALTIME`と が含まれます`SITEWISE_BUFFERED`。`SITEWISE_BUFFERED` を選択してください。

- `SITEWISE_REALTIME` (デフォルト) – データをリアルタイムで AWS IoT SiteWise ストレージに直接送信します。詳細については、「AWS IoT SiteWise Edge リアルタイム送信先を追加する」を参照してください。

- `SITEWISE_BUFFERED` – Parquet 形式でデータをバッチで Amazon S3 に送信し、 AWS IoT SiteWise ストレージにインポートします。

タイプ: 文字列

必須: はい

`name`

送信先の一意の名前。

タイプ: 文字列

必須: はい

`config`

JSON 形式の送信先タイプに固有の設定。設定は、リアルタイム送信先とバッファされた送信先によって異なります。

型: オブジェクト

必須: はい

`targetBucketArn`

発行先のバケット ARN。 AWS IoT SiteWise と Amazon S3 の両方 AWS リージョン に同じ を選択します。プレフィックスを選択した場合、1 ~ 255 文字である必要があります。

> **Note**
>
> AWS IoT SiteWiseゲートウェイを含む は、指定された S3 バケット全体にアクセスできます。バッファされたデータインジェストには専用バケットを使用することをお勧めします。

タイプ: 文字列

必須: はい

`publishPolicy`

発行ポリシーの詳細。

型: オブジェクト

必須: はい

`publishFrequency`

SiteWise Edge ゲートウェイが Amazon S3 バケットに発行する頻度。Amazon S3 へのデータのアップロード頻度は、0 分以上 30 日以下である必要があります。発行頻度の期間を指定するd場合はm、、h、および を使用できます。m は分数、hは時数、dは日数を表します。デフォルト値は 15 分です。

タイプ: 文字列

必須: はい

`localSizeLimitGB`

ローカルディスクに書き込まれるファイルの最大サイズを GB 単位で指定します。このしきい値を超えると、パブリッシャーはバッファされたすべてのデータを宛先に発行します。

タイプ: 整数

必須: はい

`siteWiseImportPolicy`

データをインポートするためのインポートポリシーの詳細 AWS IoT SiteWise。

型: オブジェクト

必須: はい

`enableSiteWiseStorageImport`

Amazon S3 バケットから AWS IoT SiteWise ストレージにデータをインポートtrueするには、これを に設定します。最初は、 でデータのコピーを作成します AWS IoT SiteWise。次に、 を true enableDeleteAfterImportに設定すると、S3 内のデータはコピー後に削除されます AWS IoT SiteWise。料金への影響が適用されます。デフォルト値は true です。

型: ブール値

必須: はい

`enableDeleteAfterImport`

AWS IoT SiteWise ストレージ`true`への取り込み後に Amazon S3 バケット内のファイルを削除するには、これを に設定します。デフォルト値は `true` です。

型: ブール値

必須: はい

`bulkImportJobRoleArn`

データ取り込み中に Amazon S3 からバッファされたデータを読み取るために が AWS IoT SiteWise 引き受ける IAM ロールの ARN。このロールは、エッジデバイスが AWS IoT SiteWise APIs で呼び出して一括インポートプロセスを開始する場合に使用されます。

> **Note**
>
> `enableSiteWiseStorageImport` が に設定されている場合`true`、このパラメータは必須です。

タイプ: 文字列

必須: いいえ

送信先のパスフィルターを追加します。詳細については、「AWS IoT SiteWise Edge 送信先にパスフィルターを追加する」を参照してください。

AWS IoT SiteWise Edge 送信先にパスフィルターを追加する

パスフィルターを送信先に追加します。パスフィルターは MQTT トピック構文を使用します。ここで、 # は任意の数のレベルに一致するワイルドカード文字で、 +は単一のレベルに一致するワイルドカード文字です。複数の送信先をゲートウェイに追加できます。各送信先には、機器テレメトリにサブスクライブされている独自のパスフィルターのセットがあります。

Siemens Industrial Edge ゲートウェイは、互換性のためにプレフィックスを使用します。詳細については、「パスフィルターのプレフィックス」を参照してください。

Console

パスフィルターを追加するには

1. AWS IoT SiteWise コンソールを開きます。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. パスフィルターを追加するゲートウェイを選択します。
4. 「送信先の追加」の「パスフィルターの追加」セクションで、「パスフィルターの追加」を選択します。
5. この送信先がサブスクライブするパスフィルターを入力します。ワイルドカード文字 (# および +) を使用して、複数のパスをサブスクライブできます。
6. パスフィルターを追加 を選択して、パスフィルターをリストに追加します。
7. 必要に応じて、ステップを繰り返してパスフィルターを追加します。
8. 必要なパスフィルターをすべて追加したら、作成 を選択します。

AWS CLI for self-hosted gateways

Example : パスフィルター設定

```
{
  "destinations": [
    {
      ...
    }
  ],
  "filters": [
    {
      "type": "PATH",
      "config": {
        "paths": [
          "home/+/sensor1/temperature",
          "home/livingroom/sensor1/temperature",
          "home/livingroom/sensor1/temperature",
          "building/#"
        ]
      }
    }
  ]
```

```
}
```

AWS CLI for Siemens IEgateways

Example : パスフィルターのプレフィックス設定

各パスフィルターのデータ (ie/d) プレフィックスとメタデータ (ie/m) プレフィックスの両方を使用して、すべてのデータをキャプチャします。

```
{
  "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**
>
> パスフィルターのリストをダウンロードして、送信先間でパスフィルターをコピーします。詳細については、「送信先のすべてのパスフィルターをダウンロードする (コンソール)」を参照してください。

パスフィルターを一括でアップロードする

パスフィルターを一括でアップロードするには、CSV ファイルまたはテキストファイルを使用します。ファイルをアップロードするときに、 は正確な重複 AWS IoT SiteWise を自動的に削除します。たとえば、 `windfarm/site1/`と `windfarm/site1/`は、文字列がまったく同じであるために が AWS IoT SiteWise キャッチする正確な重複です。部分的な重複は削除されず、追加料金が発生します。たとえば、 `windfarm/#`と `windfarm/site1`は重複するトピックです。 `windfarm/site1`は既に に包含されているためです`windfarm/#`。

> **Note**
>
> 追加料金が発生しないように、重複は避けてください。アップロードされたファイルは、.csv 形式または .txt 形式である必要があります。ヘッダーを含めることはできません。1 つの列で構成する必要があります。列で、パスフィルターをリストし、各フィルターを別々の行に表示します。ファイルには他の情報を含めないでください。

ファイルのアップロード要件

これらは追加のパスフィルター要件です。

- 1 つの .csv ファイルまたは .txt ファイルをアップロードできます。その他のフェイル形式はサポートされていません。
- CSV (.csv) ファイルにはヘッダーを含めることはできません。また、1 つの列のみを含める必要があります。
- 各行に 1 つのパスフィルターを設定できます。
- アップロードしたファイルは空にできません。
- をワイルドカード`#`として使用する場合は、トピックフィルターの最後の文字である必要があります。例えば、特定のトピックレベルでスタンドアロンキャラクターとして `topic/#`または を使用します。ただし、 は、 などのトピックレベル名内で通常の文字として使用`#`することもできます`factory/machine#1/topic`。詳細については、「パスフィルター名の特殊文字」を参照してください。
  - + 文字を使用することもできます。たとえば、 `factory/+/temp`を使用して、 `factory/machine2/temp` および をfactory/machine3/temp個別に取得するのではなく、工場のすべての温度を取得します。

AWS IoT SiteWise Edge の送信先を管理する

送信先を追加したら、送信先設定の編集、送信先の削除、パスフィルターの管理など、さまざまなオペレーションを実行して管理できます。

送信先を編集する

テーブルの送信先の横にあるラジオボタンを選択し、編集ボタンを選択して送信先を編集します。

Console

AWS IoT SiteWise コンソールを使用して送信先を編集するには

1. AWS IoT SiteWise コンソールを開きます。
2. 左側のナビゲーションで、 Edge セクションの Edge ゲートウェイを選択します。
3. 適切なゲートウェイを選択します。
4. 送信先セクションで、編集する送信先を選択し、編集を選択します。
5. 送信先を変更し、保存を選択します。

AWS CLI

を使用して送信先を編集するには AWS CLI

- JSON 機能設定情報を変更することで、送信先を編集できます。

```
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**
>
> 送信先typeまたは を更新することはできませんcapability-namespace。たとえば、 のタイプから SITEWISE_REALTIME に切り替えることはできませんSITEWISE_BUFFERED。MQTT 対応ゲートウェイごとに 1 つのリアルタイム送信先を設定できます。

送信先を削除する

送信先が不要になった場合は、SiteWise Edge ゲートウェイから削除できます。

Console

AWS IoT SiteWise コンソールを使用して送信先を削除するには

1. AWS IoT SiteWise コンソールを開きます。
2. 左側のナビゲーションで、 Edge セクションの Edge ゲートウェイを選択します。
3. 適切なゲートウェイを選択します。

4. 送信先セクションで、削除する送信先を選択し、削除を選択します。確認画面が表示されます。
5. 送信先を削除する選択を確認するには、確認ボックスに「削除」と入力します。

AWS CLI

を使用して送信先を削除するには AWS CLI

- ゲートウェイ ID を指定し、削除したい送信先を削除する機能設定を変更して、ゲートウェイ機能設定を削除します。

```
aws iotsitewise update-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
  --capability-configuration '{
    "sources": [
        {
            "type": "MQTT"
        }
    ],
    "destinations": []
}'
```

> **Note**
> 送信先配列は空 ([]) にできますが、送信先オブジェクト自体を機能設定に含める必要があります。

送信先のすべてのパスフィルターをダウンロードする (コンソール)

すべてのパスフィルターを含む CSV ファイルを AWS IoT SiteWise コンソールにダウンロードします。ダウンロードしたパスフィルターのリストを使用して、ゲートウェイの送信先間でパスフィルターリストを簡単に共有できます。

AWS IoT SiteWise コンソールを使用してすべてのパスフィルターの CSV ファイルをダウンロードするには

1. AWS IoT SiteWise コンソールを開きます。

2. 左側のナビゲーションで、 Edge セクションの Edge ゲートウェイを選択します。
3. パスフィルターを含むゲートウェイを選択します。
4. 送信先の追加または送信先の編集を選択します。
5. パスフィルターセクションに移動し、CSV のダウンロードを選択します。

> **Note**
> CSV ファイルには、パスフィルターのリストから選択したパスフィルターに関係なく、特定の宛先内のすべてのパスフィルターが含まれます。

パスフィルターを編集する

個々のパスフィルターを編集して、送信先が受信するデータを絞り込むことができます。

Console

AWS IoT SiteWise コンソールを使用して、各テキストボックス内の個々のパスフィルターを編集できます。

AWS IoT SiteWise コンソールを使用してパスフィルターを編集するには

1. AWS IoT SiteWise コンソールを開きます。
2. 左側のナビゲーションで、 Edge セクションの Edge ゲートウェイを選択します。
3. パスフィルターを含むゲートウェイを選択します。
4. 適切な送信先を選択します。
5. [編集] を選択します。
6. 編集するパスフィルターを含む行のテキストボックスを選択します。
7. パスフィルターのテキストを更新し、編集したパスフィルターのチェックボックスが選択されていることを確認します。
8. [保存] を選択します。

AWS CLI

を使用して送信先のパスフィルターを編集するには AWS CLI、まず現在の設定を取得し、変更してから、 `update-gateway-capability-configuration` コマンドを使用して更新します。

を使用してパスフィルターを編集するには AWS CLI

1. 現在の機能設定を取得します。

```
aws iotsitewise describe-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
  --query "capabilityConfiguration"
```

2. JSON を編集して、必要に応じてパスフィルターを変更します。
3. 変更されたパスフィルターを使用して機能設定を更新します。

```
aws iotsitewise update-gateway-capability-configuration \
  --gateway-id your-gateway-id \
  --capability-namespace "iotsitewise:publisher:3" \
  --capability-configuration json-containing-your-updated-path-filters
```

パスフィルターを削除する

送信先のパスフィルターを削除して、MQTT ソースとデータ処理パイプラインから受信するデータを制御できます。

Console

AWS IoT SiteWise コンソールを使用してパスフィルターを削除するには

1. AWS IoT SiteWise コンソールを開きます。
2. 左側のナビゲーションで、 Edge セクションの Edge ゲートウェイを選択します。
3. パスフィルターを含むゲートウェイを選択します。
4. 適切な送信先を選択します。
5. [編集] を選択します。
6. 送信先の編集画面で、パスフィルターセクションで、削除するパスフィルターを 1 つ以上選択します。
7. [削除] を選択します。削除の確認メッセージが表示されます。パスフィルターの削除を続行する場合は、確認画面で削除を選択します。

AWS CLI

を使用して送信先を削除するには AWS CLI

- 機能設定から削除してパスフィルターを削除します。

```
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**
>
> フィルター配列は空 ([]) にできますが、フィルターオブジェクト自体を機能設定に含める必要があります。

SiteWise Edge ゲートウェイを管理

AWS IoT SiteWise コンソールと API オペレーションを使用して AWS IoT SiteWise Edge ゲートウェイを管理できます。AWS OpsHubAWS IoT SiteWise for Windows アプリケーションを使用して、ローカルデバイスから SiteWise Edge ゲートウェイの一部の側面を管理することもできます。

ローカルデバイスのディスク使用量をモニタリングするには、 AWS OpsHub AWS IoT SiteWise アプリケーションに を使用することを強くお勧めします。`Gateway.AvailableDiskSpace` および `Gateway.UsedPercentageDiskSpace` Amazon CloudWatch メトリクスをモニタリングし、ディスク容量が少なくなったときに通知を受け取るアラームを作成することもできます。Amazon CloudWatch アラームの詳細については、「静的しきい値に基づいて CloudWatch アラームを作成する」を参照してください。

今後のデータを保存するために、端末に十分な空き容量があることを確認してください。ローカルデバイスの容量が足りなくなりそうになると、サービスは、タイムスタンプが最も古いデータを少量ずつ自動的に削除し、次のデータ用の空き容量を確保します。

サービスによってデータが削除されたかどうかをチェックするには、次の操作を行います。

1. AWS IoT SiteWise アプリケーションの にサインイン AWS OpsHub します。
2. [設定] を選択します。
3. [Logs] (ログ) の場合は、時間範囲を指定し、[Download] (ダウンロード) を選択します。
4. ログファイルを解凍します。
5. ログファイルに次のメッセージが含まれている場合、サービスはデータを削除しました。
   「SiteWise Edge ゲートウェイストレージの容量不足を防ぐため、*number* バイト分のデータが削除されました」

AWS IoT SiteWise コンソールを使用して SiteWise Edge ゲートウェイを管理

AWS IoT SiteWise コンソールを使用して、 AWS アカウント内のすべての SiteWise Edge ゲートウェイを設定、更新、モニタリングできます。

AWS IoT SiteWise コンソールの [Edge ゲートウェイ] ページに移動することで、SiteWise Edge ゲートウェイを表示できます。特定のゲートウェイの [Edge ゲートウェイの詳細] ページにアクセスするには、Edge ゲートウェイの名前を選択します。

[Edge ゲートウェイの詳細] ページの [概要] タブから、以下のことができます。

- [データソース] セクションで、データソース設定を更新し、追加のデータソースを設定します。
- [CloudWatch メトリクスを開く] を選択して、CloudWatch メトリクスコンソールでデータソースごとに取り込まれたデータポイントの数を表示します。
- [Edge 機能] セクションで、[編集] をクリックして、SiteWise Edge ゲートウェイにデータパックを追加します。
- [ゲートウェイ設定] セクションで、SiteWise Edge ゲートウェイの接続ステータスを表示します。
- Publisher 設定セクションで、SiteWise Edge AWS IoT SiteWise ゲートウェイの同期ステータスとパブリッシャーコンポーネントの設定を表示します。

[Edge ゲートウェイの詳細] ページの [更新] タブから、Edge ゲートウェイにデプロイされている現在のコンポーネントとパックのバージョンを確認できます。これは、新しいバージョンが利用可能になったときにデプロイする場所でもあります。

AWS OpsHub for を使用して SiteWise Edge ゲートウェイを管理する AWS IoT SiteWise

for AWS OpsHub AWS IoT SiteWise アプリケーションを使用して、セルフホスト SiteWise Edge ゲートウェイを管理およびモニタリングします。このアプリケーションは、次のモニタリング、管理オプションを提供します。

- [Overview] (概要) では、次のことができます。
  - SiteWise Edge ゲートウェイの詳細を表示することで、SiteWise Edge ゲートウェイデバイスのデータを把握し、問題を特定し、SiteWise Edge ゲートウェイのパフォーマンスを向上させることができます。

- エッジのローカルサーバーや機器からのデータをモニタリングする SiteWise Monitor のポータルを表示します。詳しくは、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[What is AWS IoT SiteWise Monitor] (説明) をご覧ください。
- [健康] には、SiteWise Edge ゲートウェイのデータを表示するダッシュボードがあります。プロセスエンジニアなどのドメインエキスパートは、ダッシュボードを使用して、SiteWise Edge ゲートウェイの動作の概要を確認できます。
- [アセット] には、ローカルデバイスにデプロイされたアセットと、アセットプロパティに対して最後に収集または計算された値が表示されます。
- [設定] では、次のことができます。
  - データ処理パックがインストールされている場合は、SiteWise Edge ゲートウェイの設定情報を表示し、リソースを AWS クラウドと同期します。
  - 他のツールを使用して、SiteWise Edge ゲートウェイにアクセスするために使用できる認証ファイルをダウンロードします。
  - SiteWise Edge ゲートウェイのトラブルシューティングに使用できるログをダウンロードします。
  - SiteWise Edge ゲートウェイにデプロイされた AWS IoT SiteWise コンポーネントを表示します。

> **Important**
>
> を AWS OpsHub に使用するには、以下が必要です AWS IoT SiteWise。
>
> - ローカルデバイスと AWS OpsHub AWS IoT SiteWise アプリケーションの は、同じネットワークに接続する必要があります。
> - データ処理パックを有効にする必要があります。

を使用して SiteWise Edge ゲートウェイを管理するには AWS OpsHub

1. AWS OpsHubAWS IoT SiteWise for Windows アプリケーションをダウンロードしてインストールします。
2. アプリケーションを開きます。
3. ゲートウェイにローカル認証情報がセットアップされていない場合は、ローカルオペレーティングシステムの認証情報を使用して、SiteWise Edge ゲートウェイにアクセスする のステップに従ってセットアップします。

4. Linux または LDAP (Lightweight Directory Access Protocol) の認証情報を使用して、SiteWise Edge ゲートウェイにサインインできます。SiteWise Edge ゲートウェイにサインインするには、次のいずれかを実行します。

   Linux

   1. [ホスト名または IP アドレス] には、ローカルデバイスのホスト名または IP アドレスを入力します。
   2. [Authentication] (認証) は、Linuxを選択します。
   3. [User name] (ユーザーネーム) には、お使いの Linux OS のユーザー名を入力してください。
   4. [Password] (パスワード) には、お使いの Linux OS のパスワードを入力します。
   5. [Sign in (サインイン)] を選択します。

   LDAP

   1. [ホスト名または IP アドレス] には、ローカルデバイスのホスト名または IP アドレスを入力します。
   2. [Authentication] (認証) は、LDAPを選択します。
   3. [User name] (ユーザーネーム) には、LDAP のユーザー名を入力します。
   4. [Password] (パスワード) には、LDAP パスワードを入力します。
   5. [Sign in (サインイン)] を選択します。

ローカルオペレーティングシステムの認証情報を使用して、SiteWise Edge ゲートウェイにアクセスする

Lightweight Directory Access Protocol (LDAP) の他に、Linux または Windows の認証情報を使用してセルフホスト SiteWise Edge ゲートウェイにアクセスすることもできます。

> **Important**
>
> Linux の認証情報で SiteWise Edge ゲートウェイにアクセスするには、SiteWise Edge ゲートウェイのデータ処理パックを有効にする必要があります。

Linux オペレーティングシステムの認証情報を使用して、SiteWise Edge ゲートウェイにアクセスする

次のステップは、Ubuntu が搭載された端末を使用することを想定しています。別の Linux ディストリビューションをお使いの場合は、お使いのデバイスの関連ドキュメントを参照してください。

ユーザープールを作成するには。

1. 管理者グループを作成するには、次のコマンドを実行します。

   ```
   sudo groupadd --system SWE_ADMIN_GROUP
   ```

   SWE_ADMIN_GROUP グループのユーザーは、SiteWise Edge ゲートウェイの管理者アクセスを許可できます。

2. ユーザーグループを作成するには、次のコマンドを実行します。

   ```
   sudo groupadd --system SWE_USER_GROUP
   ```

   SWE_USER_GROUP グループのユーザーは、SiteWise Edge ゲートウェイに対して読み取り専用アクセス権を許可できます。

3. 管理者グループにユーザーを追加するには、次のコマンドを実行します。*user-name* と *password* の部分は、追加するユーザーのユーザー名とパスワードに置き換えます。

   ```
   sudo useradd -p $(openssl passwd -1 password) user-name
   ```

4. ユーザーを SWE_ADMIN_GROUP または SWE_USER_GROUP に追加するには、*user-name* を、前のステップで追加したユーザー名に置き換えます。

   ```
   sudo usermod -a -G SWE_ADMIN_GROUP user-name
   ```

ユーザー名とパスワードを使用して、 AWS OpsHub AWS IoT SiteWise アプリケーションの で SiteWise Edge ゲートウェイにサインインできるようになりました。

Windows 認証情報を使用して SiteWise Edge ゲートウェイにアクセスする

次のステップは、Windows が搭載されたデバイスを使用することを想定しています。

> **Important**
>
> セキュリティは、 AWS とお客様の間の責任共有です。12 文字以上、大文字、小文字、数字、記号を組み合わせた強力なパスワードポリシーを作成します。また、Windows ファイアウォールのルールを設定して、ポート 443 では受信トラフィックを許可し、その他のすべてのポートでは受信トラフィックをブロックします。

Windows Server ユーザープールを作成する

1. 管理者として PowerShell を実行します。
   a. SiteWise Edge ゲートウェイをインストールする Windows サーバーに、管理者としてログインします。
   b. Windows の検索バーに PowerShell と入力します。
   c. 検索結果で、Windows PowerShell アプリを右クリックします。[管理者として実行] を選択します。
2. 管理者グループを作成するには、次のコマンドを実行します。

   ```
   net localgroup SWE_ADMIN_GROUP /add
   ```

   SiteWise Edge ゲートウェイに対する管理者アクセス権を許可するには、`SWE_ADMIN_GROUP` グループに属するユーザーである必要があります。
3. ユーザーグループを作成するには、次のコマンドを実行します。

   ```
   net localgroup SWE_USER_GROUP /add
   ```

   SiteWise Edge ゲートウェイに対する読み取り専用アクセス権を許可するには、`SWE_USER_GROUP` グループに属するユーザーである必要があります。
4. ユーザーを追加するには、次のコマンドを実行します。*`user-name`* と *`password`* の部分は、作成するユーザーのユーザー名とパスワードに置き換えます。

   ```
   net user user-name password /add
   ```

5. 管理者グループにユーザーを追加するには、次のコマンドを実行します。*`user-name`* は、追加したいユーザー名に置き換えてください。

```
net localgroup SWE_ADMIN_GROUP user-name /add
```

ユーザー名とパスワードを使用して、 AWS OpsHub AWS IoT SiteWise アプリケーションの で SiteWise Edge ゲートウェイにサインインできるようになりました。

SiteWise Edge ゲートウェイ証明書を管理

SiteWise Monitor や Grafana などのサードパーティーアプリケーションを SiteWise Edge ゲートウェイデバイスで使用できます。これらのアプリケーションでは、サービスとの間に TLS 接続が必要です。SiteWise Edge ゲートウェイは、現在、自己署名付き証明書を使用しています。SiteWise Monitor ポータルなどのアプリケーションをブラウザで開くと、信頼できない証明書であるという警告が表示される場合があります。

以下は、 AWS OpsHub AWS IoT SiteWise アプリケーションの から信頼された証明書をダウンロードする方法を示しています。

1. アプリケーションにサインインします。
2. [設定] を選択します。
3. [Authentication] (認証) で[Download certificate] (証明書のダウンロード) を選択します。

次では、Google Chrome または FireFox を使用することを想定しています。他のブラウザをお使いの場合は、お使いのブラウザの関連ドキュメントを参照してください。前のステップでダウンロードした証明書をブラウザに追加するには、次のいずれかを実行します。

- Google Chrome を使用する場合は、[Google Chrome Enterprise Help documentation] (Google Chrome Enterprise ヘルプドキュメント) にある[Set up certificates] (証明書を設定する) に従ってください。
- Firefox を使用している場合は、[Oracle documentation] (Oracleドキュメント) の[To Load the Certificate into the Mozilla or Firefox Browser] (Mozilla または Firefox のブラウザーに証明書を読み込むには) に従ってください。

SiteWise Edge ゲートウェイコンポーネントパックのバージョンを変更する

AWS IoT SiteWise コンソールを使用して、SiteWise Edge ゲートウェイのコンポーネントパックのバージョンを変更できます。

SiteWise Edge ゲートウェイコンポーネントパックのバージョンを変更するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左のナビゲーションペインで [ゲートウェイ] を選択します。
3. パックのバージョンを変更する SiteWise Edge ゲートウェイを選択します。
4. [ゲートウェイ設定] で、[ソフトウェアバージョンを表示] を選択します。
5. [ソフトウェアのバージョンを編集] ページで、バージョンを更新するパックについて、デプロイするバージョンを選択し、[デプロイ] を選択します。
6. [完了] をクリックします。

SiteWise Edge ゲートウェイを一覧表示する

Console

SiteWise Edge ゲートウェイを一覧表示するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. すべての SiteWise Edge ゲートウェイのリストを表示します。

AWS CLI

を使用してゲートウェイを一覧表示するには AWS CLI、次の手順に従います。

- list-gateways コマンドを使用して、すべてのゲートウェイを表示します。

```
aws iotsitewise list-gateways
```

このコマンドは、ゲートウェイの IDs、名前、その他の情報を含むリストを返します。

ページ分割パラメータを指定することもできます。

```
aws iotsitewise list-gateways --max-results 10 --next-token your-token
```

詳細については、 AWS CLI 「 コマンドリファレンス」の「list-gateways」を参照してください。

SiteWise Edge ゲートウェイの説明

Console

ゲートウェイの詳細を表示するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. 詳細を表示するゲートウェイの名前を選択します。
4. Edge ゲートウェイの詳細ページでゲートウェイの詳細を表示します。

AWS CLI

を使用して特定のゲートウェイに関する詳細情報を取得するには AWS CLI、次の手順に従います。

- ゲートウェイ ID で describe-gateway コマンドを使用します。

  ```
  aws iotsitewise describe-gateway --gateway-id a1b2c3d4-5678-90ab-
  cdef-1a1a1EXAMPLE
  ```

  このコマンドは、ゲートウェイに関する詳細情報を返します。

詳細については、 AWS CLI 「 コマンドリファレンス」の「describe-gateway」を参照してください。

SiteWise Edge ゲートウェイを作成する

Console

SiteWise Edge ゲートウェイを作成するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. [Create gateway (ゲートウェイの作成)] を選択します。
4. ゲートウェイの名前を入力します。
5. ゲートウェイの Greengrass グループを選択します。
6. 必要に応じて、ゲートウェイにタグを追加します。
7. [作成] を選択します。

AWS CLI

を使用して新しい IoT SiteWise ゲートウェイを作成するには AWS CLI、次の手順に従います。

- create-gateway コマンドを使用して新しいゲートウェイを作成します。

```
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"
    }'
```

このコマンドは、新しいゲートウェイの ID と ARN を返します。

```
{
    "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayArn": "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
```

```
}
```

詳細については、 AWS CLI 「 コマンドリファレンス」の「create-gateway」を参照してください。

SiteWise Edge ゲートウェイを更新する

Console

SiteWise Edge ゲートウェイを更新するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. 更新するゲートウェイを選択します。
4. [編集] を選択します。
5. 必要に応じてゲートウェイ名やその他の設定を更新します。
6. [保存] を選択します。

AWS CLI

を使用して既存のゲートウェイを更新するには AWS CLI、次の手順に従います。

- update-gateway コマンドを使用して、ゲートウェイの名前を更新します。

```
aws iotsitewise update-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE \
    --gateway-name "UpdatedGatewayName"
```

  このコマンドが成功した場合、出力は生成されません。

詳細については、 AWS CLI 「 コマンドリファレンス」の「update-gateway」を参照してください。

ゲートウェイ機能設定を更新する

Console

ゲートウェイ機能設定を更新するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. 更新するゲートウェイの名前を選択します。
4. データソースセクションで、編集 を選択します。
5. 必要に応じてデータソース設定を更新します。
6. [保存] を選択します。

AWS CLI

を使用してゲートウェイの機能設定を更新するには AWS CLI、次の手順に従います。

- update-gateway-capability-configuration コマンドを使用して、機能設定を更新します。

```
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": ""
            }
        ]
```

```
    }'
```

このコマンドは、機能の名前空間と同期ステータスを返します。

```
{
    "capabilityNamespace": "iotsitewise:opcuacollector:1",
    "capabilitySyncStatus": "CONFIGURING"
}
```

詳細については、 AWS CLI 「 コマンドリファレンス」のupdate-gateway-capability-configuration」を参照してください。

ゲートウェイリソースにタグを付ける

Console

ゲートウェイリソースにタグを付けるには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. タグを付けるゲートウェイの名前を選択します。
4. [タグ] タブを選択します。
5. [タグを管理] を選択します。
6. 新しいタグを追加を選択し、タグごとにキーと値を入力します。
7. [保存] を選択します。

AWS CLI

を使用してゲートウェイにタグを追加するには AWS CLI、次の手順に従います。

- tag-resource コマンドを使用して、ゲートウェイにタグを追加します。

```
aws iotsitewise tag-resource \
    --resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE" \
    --tags '{
        "Department": "Operations",
```

```
        "Project": "FactoryAutomation"
    }'
```

このコマンドが成功した場合、出力は生成されません。

詳細については、 AWS CLI 「 コマンドリファレンス」の「tag-resource」を参照してください。

ゲートウェイのタグを一覧表示する

Console

ゲートウェイのタグを一覧表示するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. タグを表示するゲートウェイの名前を選択します。
4. [タグ] タブを選択します。
5. ゲートウェイに関連付けられているタグのリストを表示します。

AWS CLI

を使用してゲートウェイに関連付けられたタグを一覧表示するには AWS CLI、次の手順に従います。

- list-tags-for-resource コマンドを使用して、ゲートウェイのタグを一覧表示します。

```
aws iotsitewise list-tags-for-resource \
    --resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
```

このコマンドは、ゲートウェイに関連付けられたタグを返します。

```
{
    "tags": {
        "Environment": "Production",
        "Location": "Factory1",
        "Department": "Operations",
```

```
        "Project": "FactoryAutomation"
    }
}
```

詳細については、 AWS CLI 「 コマンドリファレンス」のlist-tags-for-resource」を参照してください。

ゲートウェイからタグを削除する

Console

ゲートウェイからタグを削除するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. タグを削除するゲートウェイの名前を選択します。
4. [タグ] タブを選択します。
5. [タグを管理] を選択します。
6. 削除する各タグの横にある削除アイコン (X) を選択します。
7. [保存] を選択します。

AWS CLI

を使用してゲートウェイからタグを削除するには AWS CLI、次の手順に従います。

- untag-resource コマンドを使用して、ゲートウェイからタグを削除します。

```
aws iotsitewise untag-resource \
    --resource-arn "arn:aws:iotsitewise:us-east-1:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE" \
    --tag-keys '["Project", "Department"]'
```

  このコマンドが成功した場合、出力は生成されません。

詳細については、 AWS CLI 「 コマンドリファレンス」の「untag-resource」を参照してください。

コンポーネントのバージョン AWS IoT SiteWise を更新する

AWS IoT Greengrass コアデバイスの AWS IoT SiteWise ゲートウェイコンポーネントを更新して、最新の機能、パフォーマンスの向上、セキュリティパッチへのアクセスを確保します。

で AWS IoT SiteWise コンポーネントを更新するには AWS IoT Greengrass

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションペインで、エッジゲートウェイを選択します。
3. 編集するゲートウェイを選択し、編集を選択します。
4. Edge 機能で、ソフトウェアバージョンで、利用可能なソフトウェア更新を選択します。ソフトウェアバージョンの編集ページが表示されます。
5. コンポーネントバージョンを選択します。

> **Note**
>
> 利用可能な最新バージョンを選択することをお勧めします。ゲートウェイコンポーネントup-to-dateに保つことで、産業データの収集と処理に最適な機能を維持できます。

6. [デプロイ] を選択します。これにより、 AWS IoT Greengrass V2 デプロイが開始され、ゲートウェイ上の AWS IoT SiteWise コンポーネントが更新されます。

SiteWise Edge ゲートウェイを削除する

Console

SiteWise Edge ゲートウェイを削除するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. 削除するゲートウェイを選択します。
4. [削除] を選択します。
5. ゲートウェイを削除することを確認するには、表示されるウィンドウで「削除」と入力し、「削除」を選択します。

AWS CLI

を使用してゲートウェイを削除するには AWS CLI、次の手順に従います。

1. ゲートウェイを一覧表示して、削除するゲートウェイのゲートウェイ ID を特定します。

```
aws iotsitewise list-gateways
```

このコマンドは、IDs、名前、その他の情報を含むゲートウェイのリストを返します。

```
{
    "gatewaySummaries": [
        {
            "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
            "gatewayName": "ExampleCorpGateway",
            "gatewayCapabilitySummaries": [
                {
                    "capabilityNamespace": "iotsitewise:opcuacollector:1",
                    "capabilitySyncStatus": "IN_SYNC"
                }
            ],
            "creationDate": 1588369971.457,
            "lastUpdateDate": 1588369971.457
        }
    ]
}
```

2. ID を指定してゲートウェイを削除します。

```
aws iotsitewise delete-gateway --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

このコマンドが成功した場合、出力は生成されません。

> **Note**
>
> ゲートウェイを削除すると、ゲートウェイのファイルの一部がゲートウェイのファイルシステムに残ります。

3. ゲートウェイが削除されたことを確認するには、ゲートウェイを再度一覧表示します。

```
aws iotsitewise list-gateways
```

削除されたゲートウェイはリストに表示されなくなります。

詳細については、 AWS CLI 「 コマンドリファレンス」の「delete-gateway」を参照してください。

SiteWise Edge ゲートウェイをバックアップおよび復元する

このトピックでは、 SiteWise Edge ゲートウェイを復元してメトリックデータをバックアップする方法を説明します。同じマシンで SiteWise Edge ゲートウェイが壊れていて、問題のトラブルシューティングが必要な場合は、SiteWise Edge ゲートウェイの問題のトラブルシューティング」の AWS IoT SiteWise ドキュメントを参照してください。

> **Note**
> このトピックで説明するガイダンスは、 AWS IoT Greengrass V2 バージョン 2.1.0 以降にインストールされている SiteWise Edge ゲートウェイを対象としています。

メトリクスデータの日次バックアップ

データを新しいマシンに転送または復元する場合は、バックアップを作成しておくことが大切です。データをバックアップしておくことで、転送または復元中に運用データが失われるリスクを大幅に軽減できます。

このセクションは、データ処理パックを使用するゲートウェイに適用されます。データ処理パックの詳細については、「SiteWise Edge でデータ処理するためのアセットモデルを設定する」 を参照してください。

influxdb フォルダーのパスは次のとおりです。

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeProcessor/influxdb
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeProcessor\influxdb
```

フォルダー全体とその下にあるすべてのものをバックアップすることをお勧めします。

メトリクスデータは、1.0 SiteWise Edge から外部ハードドライブまたは AWS クラウドに定期的にバックアップすることをお勧めします。

SiteWise Edge ゲートウェイを復元する

SiteWise Edge ゲートウェイの復元を試みる前に、ゲートウェイに接続されているすべてのエッジデバイスが停止または切断されていることを確認してください。

SiteWise Edge ゲートウェイを復元するには、以下の手順を実行します。

1. SiteWise Edge ゲートウェイの作成時にダウンロードしたインストールスクリプトを使用して、SiteWise Edge ゲートウェイを新しいマシンに復元します。SiteWise Edge ゲートウェイをセットアップするには、「ローカルデバイスに SiteWise Edge ゲートウェイソフトウェアをインストールする」の手順をお読みください。

   インストールスクリプトを紛失したり、見つからない場合は、AWS カスタマーサポート にご連絡ください。
2. SiteWise Edge ゲートウェイがインストールされたら、AWS IoT Greengrass コンソールにログインします。
3. コンポーネントを再デプロイするには、[管理] に移動し、[AWS IoT Greengrass デバイス] で [コアデバイス] を選択します。
4. AWS IoT Greengrass コアデバイスのテーブルで、SiteWise Edge ゲートウェイに対応するコアデバイスを選択します。
5. デバイスページが開いたら、[デプロイ] タブを開いてデプロイ ID を選択します。選択した ID で [デプロイ] ページが開きます。
6. [デプロイ] ページが表示されたら、右上の [アクション] ボタンを押して [修正] オプションを選択し、新しいデプロイを開始します。デプロイの設定をします。デプロイをそのまま残しておきたい場合は、[レビュー] と [デプロイ」に進みます。
7. デプロイステータスが `Completed` になるまで待ちます。

> **Note**
>
> SiteWise Edge のすべてのコンポーネントが完全にセットアップされて動作するには数分の時間がかかります。

AWS IoT SiteWise データの復元

以下の手順に従って、新しいマシンにデータを復元できます。

1. `influxdb` フォルダーを新しいマシンにコピーします。
2. ターミナルで次のコマンドを実行することで、SiteWise EdgeProcessor コンポーネントを停止します。

   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. データをバックアップしたパスを見つけて、以下のコマンドを実行します。

   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 コンポーネントを再起動します。

   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
```

バックアップと復元が正常に完了したことを検証する

この手順を使用して、バックアップしたデータと SiteWise Edge ゲートウェイの復元を検証できます。

> **Note**
>
> この手順では、 AWS OpsHub for をインストールする必要があります AWS IoT SiteWise。詳細については、AWS OpsHub for を使用した SiteWise Edge ゲートウェイの管理 AWS IoT SiteWise」を参照してください。

1. Open AWS OpsHub for AWS IoT SiteWise。
2. SiteWise Edge ゲートウェイの [設定] ページで、[コンポーネント] テーブルに一覧表示されている各コンポーネントのステータスを確認します。ステータスの色が緑色で、表示に RUNNING と表示されていることを確認します。
3. ポータルダッシュボードで過去のデータを検証し、過去のデータと新しいデータの両方が正しく設定されていることを確認します。過去のデータと新しいデータの間にはダウンタイムがあります。その場合は、データポイントが収集されていない期間が表示されます。

SiteWise Edge ゲートウェイのバックアップまたは復元で問題が発生した場合は、トラブルシューティングトピック「AWS IoT SiteWise Edge ゲートウェイのトラブルシューティング」を参照してください。

レガシーゲートウェイ (AWS IoT Greengrass Version 1)

> **Note**
>
> で実行されている SiteWise Edge ゲートウェイ AWS IoT Greengrass V1 は、2021 年 7 月 29 日より前にこの機能の使用を開始した場合にのみ使用できます。を使用した AWS IoT SiteWise ゲートウェイの実行の詳細については AWS IoT Greengrass V2、「」を参照して

ください[を使用して AWS IoT SiteWise Edge ゲートウェイをセルフホストする AWS IoT Greengrass V2](#)。

SiteWise Edge ゲートウェイが排他的に実行されるようになり AWS IoT Greengrass V2、産業用 IoT アプリケーションに強化された機能とパフォーマンスが向上しました。この最新バージョンは、モジュラーソフトウェアのデプロイを可能にする最新のコンポーネントベースのフレームワーク上に構築されたアーキテクチャの進化 AWS IoT Greengrass V2 を表しています。これにより、統合インストーラーによるインストールが合理化され、開発者はカスタムコンポーネントのデプロイとローカルテストをより柔軟に行うことができます。コンポーネントベースのモデルでは、より効率的なリソース管理が可能になり、コンポーネントレシピを通じて簡略化された設定アプローチが導入されます。この設計により、コンポーネント間の依存関係の処理が容易になり、継続的なデプロイプラクティスがサポートされ、ローカル開発用の拡張 CLI 機能が提供されます。さらに、 は を使用して設定管理を AWS IoT Greengrass V2 一元化 AWS IoT Core し、ログ記録とモニタリング機能を改善します。これらはすべて、より詳細なセキュリティアクセス許可モデルによって保護されています。

を使用した SiteWise Edge ゲートウェイの開始方法の詳細については AWS IoT Greengrass V2、「」を参照してください[AWS IoT SiteWise Edge セルフホストゲートウェイの要件](#)。これらのリソースは、ゲートウェイの設定、データソースの設定、産業用 IoT インフラストラクチャの管理に関するstep-by-stepの手順を提供します。

> **Note**
>
> IoT IoT サービスの革新と改善が AWS 続く中、最新の機能と機能強化を常に把握しておくことをお勧めします。産業用 IoT ソリューションをさらに最適化できる新機能については、AWS IoT SiteWise および AWS IoT Greengrass ドキュメントを定期的に確認してください。

産業用アセットをモデリングする

AWS IoT SiteWise アセットを使用して、産業オペレーションの仮想表現を作成できます。アセットは、デバイス、機器、または 1 つ以上のデータストリームを AWS クラウドにアップロードするプロセスを表します。たとえば、アセットデバイスの例として、気温、プロペラ回転速度、および電力出力の時系列測定値を AWS IoT SiteWiseのアセットプロパティに送信する風力タービンがあります。

各データストリームは、一意のプロパティエイリアスに対応します。たとえば、エイリアス `/company/windfarm/3/turbine/7/temperature` は、風力発電所 #3 のタービン #7 からの温度データストリームを一意に識別します。温度データを摂氏から華氏に変換するなど、数式を使用して受信測定データを変換するように AWS IoT SiteWise アセットを設定できます。

アセットは、風力発電所全体など、デバイスの論理的なグループを表すこともできます。アセットを他のアセットに関連付けると、複雑な産業オペレーションを表すアセット階層を作成できます。アセットは、関連付けられた子アセット内のデータにアクセスできます。これにより、 AWS IoT SiteWise 式を使用して、風力発電所の正味電力出力などの集計メトリクスを計算できます。

すべてのアセットをアセットモデルから作成する必要があります。アセットモデルはアセットのフォーマットを標準化する宣言的な構造です。アセットモデルでは、同じタイプの複数のアセットに一貫した情報が適用されるため、デバイスのグループを表すアセット内のデータを処理できます。前の図では、3 つのタービンすべてに同じアセットモデルを使用します。これは、すべてのタービンが共通のプロパティセットを共有しているためです。

コンポーネントモデルを作成することもできます。コンポーネントモデルは、アセットモデルやその他のコンポーネントモデルに含めることができる特殊なタイプのアセットモデルです。コンポーネントモデルを使用して、複数のアセットモデル間で共有するセンサー、モーターなどの一般的な再利用可能なサブアセンブリを定義できます。

アセットモデルを定義したら、産業アセットを作成できます。アセットを作成するには、ACTIVE アセットモデルを選択して、そのモデルからアセットを作成します。次に、データストリームのエイリアスや属性など、アセット固有の情報を入力できます。前の図では、1 つのアセットモデルから 3 つのタービンアセットを作成し、各タービンに /company/windfarm/3/turbine/7/temperature のようなデータストリームエイリアスを関連付けます。

既存のアセット、アセットモデル、およびコンポーネントモデルを更新および削除することもできます。アセットモデルを更新すると、そのアセットモデルに基づくすべてのアセットに、基になるモデルに加えた変更が反映されます。コンポーネントモデルを更新する場合、これはコンポーネントモデルを参照するすべてのアセットモデルに基づくすべてのアセットに適用されます。

アセットモデルは、例えば、サブコンポーネントが多数ある複雑な機器をモデル化する場合など、非常に複雑な場合があります。このようなアセットモデルを整理して維持しやすくするために、カスタム複合モデルを使用して、関連するプロパティをグループ化したり、共有コンポーネントを再使用したりできます。詳細については、「カスタム複合モデル (コンポーネント)」を参照してください。

トピック

- アセットおよびモデルの状態
- アセットモデルのバージョン
- カスタム複合モデル (コンポーネント)
- AWS IoT SiteWise オブジェクト IDs を設定する
- のアセットモデルとコンポーネントモデルを作成する AWS IoT SiteWise
- でアセットモデルのアセットを作成する AWS IoT SiteWise
- でアセットを検索する AWS IoT SiteWise コンソール
- 属性値を更新する
- アセットの関連付けと関連付け解除
- アセットとモデルを更新する
- でアセットとモデルを削除する AWS IoT SiteWise
- アセットとモデルを使用した一括オペレーション

アセットおよびモデルの状態

アセット、アセットモデル、またはコンポーネントモデルを作成、更新、または削除すると、変更が反映されるまでに時間がかかります。これらのオペレーションは非同期で AWS IoT SiteWise 解決さ

れ、各リソースのステータスが更新されます。各アセット、アセットモデル、およびコンポーネントモデルには、リソースの状態とエラーメッセージ (該当する場合) を含むステータスフィールドがあります。ステータスは以下のいずれかの値になります。

- ACTIVE – リソースはアクティブです。これは、アセット、アセットモデル、およびコンポーネントモデルを照会して操作できる唯一の状態です。
- CREATING – リソースは作成中です。
- UPDATING – リソースは更新中です。
- DELETING – リソースは削除中です。
- PROPAGATING – (アセットモデルとコンポーネントモデルのみ) 変更は、すべての依存リソースに伝播中です (アセットモデルからアセットへ、またはコンポーネントモデルからアセットモデルへ)。
- FAILED - リソースは、おそらく式の循環参照が原因で、作成または更新オペレーション時に検証に失敗しました。FAILED 状態のリソースは削除できます。

一部の作成、更新、削除オペレーションでは、オペレーションが解決ACTIVEしている間、アセット、アセットモデル、またはコンポーネントモデルが 以外の状態 AWS IoT SiteWise になります。これらのオペレーションのいずれかを実行した後にリソースをクエリまたは操作するには、状態が ACTIVE になるまで待つ必要があります。そうしないと、リクエストは失敗します。

トピック

- アセットのステータスを確認する
- アセットまたはコンポーネントモデルのステータスを確認する

アセットのステータスを確認する

AWS IoT SiteWise コンソールまたは API を使用して、アセットのステータスを確認できます。

トピック

- アセットのステータスを確認する (コンソール)
- アセットのステータスを確認する (AWS CLI)

アセットのステータスを確認する (コンソール)

AWS IoT SiteWise コンソールでアセットのステータスを確認するには、次の手順に従います。

アセットの状態を確認するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. チェックするアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [Asset details] (アセットの詳細) パネルで[Status] (状態) を検索します。

アセットのステータスを確認する (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用して、アセットのステータスを確認できます。

アセットの状態を確認するには、`assetId` パラメータを指定して DescribeAsset オペレーションを使用します。

アセットのステータスを確認するには (AWS CLI)

- 以下のコマンドを実行して、アセットを記述します。*`asset-id`* をアセットの ID または外部 ID に置き換えます。外部 ID はユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

```
aws iotsitewise describe-asset --asset-id asset-id
```

  このオペレーションは、アセットの詳細を含むレスポンスを返します。レスポンスには、次の構造を持つ `assetStatus` オブジェクトが含まれています。

```
{
    ...
    "assetStatus": {
      "state": "String",
      "error": {
         "code": "String",
         "message": "String"
      }
    }
  }
```

アセットの状態は JSON オブジェクトの `assetStatus.state` にあります。

アセットまたはコンポーネントモデルのステータスを確認する

AWS IoT SiteWise コンソールまたは API を使用して、アセットモデルまたはコンポーネントモデルのステータスを確認できます。

トピック

- アセットモデルまたはコンポーネントモデルのステータスを確認する (コンソール)
- アセットモデルまたはコンポーネントモデルのステータスを確認する (AWS CLI)

アセットモデルまたはコンポーネントモデルのステータスを確認する (コンソール)

AWS IoT SiteWise コンソールでアセットモデルまたはコンポーネントモデルのステータスを確認するには、次の手順に従います。

> **Tip**
> アセットモデルとコンポーネントモデルはどちらも、ナビゲーションペインの [モデル] に一覧表示されます。選択されたアセットモデルまたはコンポーネントモデルの [詳細] パネルには、そのタイプが表示されます。

アセットモデルまたはコンポーネントモデルのステータスを確認するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。

2. ナビゲーションペインで、[モデル] を選択します。
3. 確認するモデルを選択します。
4. [Details] (詳細) パネルで[Status] (状態) を探す。

アセットモデルまたはコンポーネントモデルのステータスを確認する (AWS CLI)

を使用して、アセットモデルまたはコンポーネントモデルのステータス AWS CLI を確認できます。

アセットモデルまたはコンポーネントモデルのステータスを確認するには、`assetModelId` パラメータを指定して DescribeAssetModel オペレーションを使用します。

> **Tip**
>
> は、コンポーネントモデルをアセットモデルのタイプとして AWS CLI 定義します。したがって、両方のタイプのモデルに同じ DescribeAssetModel オペレーションを使用します。レスポンスの `assetModelType` フィールドは、それが `ASSET_MODEL` か `COMPONENT_MODEL` かを示します。

アセットモデルまたはコンポーネントモデルのステータスを確認するには (AWS CLI)

- 次のコマンドを実行して、モデルを記述します。*`asset-model-id`* をアセットモデルまたはコンポーネントモデルの ID または外部 ID に置き換えます。外部 ID はユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

このオペレーションは、モデルの詳細を含むレスポンスを返します。レスポンスには、次の構造を持つ assetModelStatus オブジェクトが含まれています。

```
{
    ...
    "assetModelStatus": {
      "state": "String",
      "error": {
         "code": "String",
         "message": "String"
      }
    }
  }
```

モデルの状態は JSON オブジェクトの assetModelStatus.state にあります。

アセットモデルのバージョン

AWS IoT SiteWise は、アセットモデルとコンポーネントモデルの作成および更新オペレーションの非同期処理をサポートします。また、モデルのステータスも更新します。

AWS IoT SiteWise は、作成および更新リクエストにおける有効なモデルの変更を依存リソース (アセットモデルからアセットへ、またはコンポーネントモデルからアセットモデルへ) に伝達します。次に、モデルを ACTIVE 状態にします。

指定されたモデル定義が無効である場合、 はモデルを FAILED状態に AWS IoT SiteWise します。変更は依存リソースに伝播されません。依存リソースとは、モデルが ACTIVE 状態のときに伝播される最後のモデル定義を指します。

上記の情報に基づいて、モデル定義には 2 種類のモデルバージョンがあります。

1. 最新バージョン – 作成または更新リクエストの一部として受け入れられる最新の定義。
2. アクティブバージョン – 最新の定義が正常に処理され、モデルの状態は ACTIVE です。

デフォルトでは、アセットモデルまたはコンポーネントモデルで describe API が呼び出されると、モデルの最新バージョンの詳細が返されます。アセットモデルまたはコンポーネントモデルのアクティブバージョンが必要なシナリオがあります。以下のシナリオの例を参照してください。

- 定義が無効な更新オペレーションでは、アセットモデルは FAILED 状態になります。アセットモデルのアクティブバージョンを取得し、この有効な定義を参照する別の更新リクエストを作成することで、変更を元に戻す必要があります。
- のアプリケーションは、顧客がアセットとそれに対応するアセットモデルを表示できる場所 AWS IoT SiteWise にあります。ユーザーが特定のアセットに対応するアセットモデル定義を参照し、アセットモデルが一時的な UPDATING、PROPAGATING、または FAILED 状態の場合、最新バージョンはアセットにまだ伝播されていないアセットモデル定義を返します。この場合、顧客にアセットモデルのアクティブバージョンを取得する必要があります。

トピック

- アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得する (コンソール)
- アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得する (AWS CLI)

アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得する (コンソール)

AWS IoT SiteWise コンソールでアセットモデルまたはコンポーネントモデルのアクティブバージョンを取得するには、次の手順に従います。

> **Tip**
> アセットモデルとコンポーネントモデルはどちらも、ナビゲーションペインの [モデル] に一覧表示されます。選択されたアセットモデルまたはコンポーネントモデルの [詳細] パネルには、そのタイプが表示されます。

アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. アクティブバージョンを取得するモデルを選択します。
   a. モデルが ACTIVE 状態の場合、そのアクティブバージョンが表示されます。
   b. モデルが一時的な UPDATING、PROPAGATING、または FAILED 状態の場合、[詳細] パネルの [ステータス] にある [アクティブバージョンを表示] を見つけます。

アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得する (AWS CLI)

を使用して、アセットモデルまたはコンポーネントモデルのアクティブバージョン AWS CLI を取得します。

アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得するには、`assetModelVersion` パラメータを指定して DescribeAssetModel オペレーションを使用します。

> **Tip**
>
> は、コンポーネントモデルをアセットモデルのタイプとして AWS CLI 定義します。したがって、両方のタイプのモデルに同じ DescribeAssetModel オペレーションを使用します。レスポンスの `assetModelType` フィールドは、それが `ASSET_MODEL` か `COMPONENT_MODEL` かを示します。

アセットモデルまたはコンポーネントモデルのアクティブバージョンを取得するには (AWS CLI)

- 次のコマンドを実行して、モデルを記述します。*`asset-model-id`* をアセットモデルまたはコンポーネントモデルの ID または外部 ID に置き換えます。外部 ID はユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

  ```
  aws iotsitewise describe-asset-model --asset-model-id asset-model-id --asset-model-
  version ACTIVE
  ```

  このオペレーションは、モデルの詳細を含むレスポンスを返します。レスポンスには、次の構造を持つ `assetModelStatus` オブジェクトが含まれています。

  ```
  {
      ...
      "assetModelName": "string",
      "assetModelProperties": [ ... ],
      ...,
      "assetModelVersion": "string"
  }
  ```

カスタム複合モデル (コンポーネント)

多くの部品がある複雑な機械など、特に複雑な産業用アセットをモデル化する場合、アセットモデルを整理して維持できるようにすることが課題になる可能性があります。

このような場合、カスタム複合モデル、またはコンソールを使用している場合はコンポーネントを既存のアセットモデルとコンポーネントモデルに追加できます。これらは、関連するプロパティをグループ化し、サブコンポーネント定義を再使用して、整理を維持するのに役立ちます。

カスタム複合モデルには 2 つのタイプがあります。

- インラインカスタム複合モデルは、カスタム複合モデルが属するアセットモデルまたはコンポーネントモデルに適用されるグループ化されたプロパティのセットを定義します。これらを使用して、関連するプロパティをグループ化します。これらは、名前、説明、および一連のアセットモデルプロパティで構成されます。再利用できません。
- コンポーネントモデルベースのカスタム複合モデルは、アセットモデルまたはコンポーネントモデルに含めるコンポーネントモデルを参照します。これらを使用して、モデルに標準サブアセンブリを含めます。これらは、参照するコンポーネントモデルの名前、説明、および ID で構成されます。独自のプロパティはありません。参照されるコンポーネントモデルが、作成されたアセットに関連付けられたプロパティを提供します。

以下のセクションでは、カスタム複合モデルを設計で使用する方法について説明します。

トピック

- インラインカスタム複合モデル
- コンポーネントモデルベースのカスタム複合モデル
- パスを使用してカスタム複合モデルのプロパティを参照する

インラインカスタム複合モデル

インラインカスタム複合モデルは、関連するプロパティをグループ化することによってアセットモデルを整理する方法を提供します。

例えば、ロボットアセットをモデル化するとします。ロボットには、サーボモーター、電源、およびバッテリーが含まれています。これらの各構成部品には、モデルに含める独自のプロパティがあります。次のようなプロパティを持つ `robot_model` というアセットモデルを定義できます。

- robot_model
  - servo_status (整数)
  - servo_position (倍精度)
  - powersupply_status (整数)
  - powersupply_temperature (倍精度)
  - battery_status (整数)
  - battery_charge (倍精度)

ただし、サブアセンブリが多数ある場合や、サブアセンブリ自体に多くのプロパティがある場合があります。このような場合、前述の例のように、プロパティがあまりにも多く存在すると、モデルルートで単一のフラットリストを参照および維持するのが面倒になる可能性があります。

このような状況に対処するには、インラインカスタム複合モデルを使用してプロパティをグループ化できます。インラインカスタム複合モデルは、独自のプロパティを定義するカスタム複合モデルです。例えば、次のようにロボットをモデル化できます。

- robot_model
  - servo
    - status (整数)
    - position (倍精度)
  - powersupply
    - status (整数)
    - temperature (倍精度)
  - battery
    - status (整数)
    - charge (倍精度)

前の例では、servo、powersupply、および battery は、robot_model アセットモデル内で定義されたインラインカスタム複合モデルの名前です。これらの複合モデルはそれぞれ、独自のプロパティを定義します。

> **Note**
>
> この場合、各カスタム複合モデルは独自のプロパティを定義し、すべてのプロパティがアセットモデル自体の一部になります (この場合は `robot_model`)。これらのプロパティは、他のアセットモデルやコンポーネントモデルと共有されません。例えば、`servo` というインラインカスタム複合モデルも持つ他のアセットモデルを作成した場合、`robot_model` 内で `servo` を変更しても、他のアセットモデルの `servo` の定義には影響しません。
> このような共有を実装する場合 (例えば、すべてのアセットモデルが共有できるサーボの定義を 1 つだけにするために)、代わりにコンポーネントモデルを作成し、それを参照するコンポーネントモデルベースの複合モデルを作成します。詳細については、次のセクションを参照してください。

インラインカスタム複合モデルを作成する方法の詳細については、「カスタム複合モデルを作成する (コンポーネント)」を参照してください。

コンポーネントモデルベースのカスタム複合モデル

でコンポーネントモデルを作成して、標準の再利用可能なサブアセンブリ AWS IoT SiteWise を定義できます。コンポーネントモデルを作成したら、他のアセットモデルとコンポーネントモデルにリファレンスを追加できます。これを行うには、コンポーネントモデルベースのカスタム複合モデルをコンポーネントを参照するモデルに追加します。コンポーネントへの参照は、多くのモデルから追加することも、同じモデル内で複数回追加することもできます。

このように、モデル間での同じ定義の重複を回避できます。また、コンポーネントモデルに加えた変更は、それを使用するすべてのアセットモデルに反映されるため、モデルのメンテナンスも簡素化されます。

例えば、産業設備に、すべて同じ種類のサーボモーターを使用する多くのタイプの機器があるとします。それらのいくつかは、単一の機器に多くのサーボモーターを持っています。機器タイプごとにアセットモデルを作成しますが、`servo` の定義を毎回複製する必要はありません。1 回だけモデル化し、さまざまなアセットモデルで使用します。後で `servo` の定義に変更を加えた場合、すべてのモデルとアセットで更新されます。

この方法で前の例のロボットをモデル化するには、サーボモーター、電源、およびバッテリーをこのようなコンポーネントモデルとして定義できます。

- `servo_component_model`

  - status (整数)
  - position (倍精度)

- powersupply_component_model
  - status (整数)
  - temperature (倍精度)

- battery__component_model
  - status (整数)
  - charge (倍精度)

その後、これらのコンポーネントを参照する robot_model などのアセットモデルを定義できます。複数のアセットモデルが同じコンポーネントモデルを参照できます。ロボットに複数のサーボモーターがある場合など、1 つのアセットモデルで同じコンポーネントモデルを複数回参照することもできます。

- robot_model
  - servo1 (リファレンス: servo_component_model)
  - servo2 (リファレンス: servo_component_model)
  - servo3 (リファレンス: servo_component_model)
  - powersupply (リファレンス: powersupply_component_model)
  - battery (リファレンス: battery_component_model)

コンポーネントモデルの作成方法については、「コンポーネントモデルを作成する」を参照してください。

他のモデルでコンポーネントモデルを参照する方法については、「カスタム複合モデルを作成する (コンポーネント)」を参照してください。

パスを使用してカスタム複合モデルのプロパティを参照する

アセットモデル、コンポーネントモデル、またはカスタム複合モデルにプロパティを作成する場合、変換やメトリクスなど、その値を使用する他のプロパティからプロパティを参照できます。

AWS IoT SiteWise には、 プロパティを参照するさまざまな方法が用意されています。最も簡単な方法は、多くの場合、プロパティ ID を使用することです。ただし、参照するプロパティがカスタム複合モデルにある場合は、代わりにパスで参照する方が便利です。

パスは、アセットモデルと複合モデル内のネストされた複合モデル間の位置の観点からプロパティを指定するパスセグメントの順序付けられたシーケンスです。

プロパティパスを取得する

プロパティのパスは、AssetModelProperty の `path` フィールドから取得できます。

例えば、 プロパティ `position` を持つカスタム複合モデル `servo` を含むアセットモデル `robot_model` があるとします。`servo` に対して DescribeAssetModelCompositeModel を呼び出した場合、`position` プロパティは次のような `path` フィールドを一覧表示します。

```
"path": [
    {
       "id": "asset model ID",
       "name": "robot_model"
    },
    {
       "id": "composite model ID",
       "name": "servo"
    },
    {
       "id": "property ID",
       "name": "position"
    }
]
```

プロパティパスの使用

プロパティパスは、変換やメトリクスなど、他のプロパティを参照するプロパティを定義するときに使用できます。

プロパティは、変数を使用して別のプロパティを参照します。変数の操作方法の詳細については、「数式で変数を使用する」を参照してください。

プロパティを参照する変数を定義するときには、プロパティの ID またはパスを使用できます。

参照されるプロパティのパスを使用する変数を定義するには、その値の `propertyPath` フィールドを指定します。

例えば、パスを使用してプロパティを参照するメトリクスを持つアセットモデルを定義するには、次のようなペイロードを CreateAssetModel に渡すことができます。

```
{
    ...
    "assetModelProperties": [
        {
            ...
            "type": {
                "metric": {
                    ...
                    "variables": [
                        {
                            "name": "variable name",
                            "value": {
                                "propertyPath": [
                                    path segments
                                ]
                            }
                        }
                    ],
                    ...
                }
            },
            ...
        },
        ...
    ],
    ...
}
```

AWS IoT SiteWise オブジェクト IDs を設定する

AWS IoT SiteWise は、アセット、アセットモデル、プロパティ、階層など、さまざまなタイプの永続オブジェクトを定義します。このようなオブジェクトには、取得、更新、削除に使用できる一意の識別子があります。

AWS IoT SiteWise には、ID 作成用のさまざまなオプションがあります。 は、オブジェクト作成時にデフォルトで 1 つのオプション AWS IoT SiteWise を生成します。ユーザーは、独自の ID をオブジェクトに指定することもできます。

トピック

- オブジェクト UUID を操作する
- 外部 ID を使用する

オブジェクト UUID を操作する

のすべての永続オブジェクト AWS IoT SiteWise には、それを識別するための UUID があります。例えば、アセットモデルにはアセットモデル ID、アセットにはアセット ID などがあります。この ID は、オブジェクトの作成時に割り当てられ、オブジェクトの存続期間中は変更されません。

新しいオブジェクトを作成すると、 はデフォルトで一意の ID AWS IoT SiteWise を生成します。作成時に独自の ID を UUID 形式で指定することもできます。

> **Note**
>
> UUIDs は、作成された AWS リージョン内でグローバルに一意で、同じオブジェクトタイプに対して一意である必要があります。が ID AWS IoT SiteWise を自動的に生成する場合、常に一意です。独自の ID を選択した場合は、一意であることを確認してください。

例えば、CreateAssetModel を呼び出すことによって新しいアセットモデルを作成する場合、リクエストのオプションの `assetModelId` フィールドで独自の UUID を指定できます。

逆に、リクエストから `assetModelId` を省略すると、 AWS IoT SiteWise は新しいアセットモデルの UUID を生成します。

外部 ID を使用する

UUID 以外の形式で独自の ID を定義するには、外部 ID を割り当てることができます。例えば、使用していないシステムで使用している ID を再利用する場合や AWS、人間が読み取れるようにする場合などです。外部 ID の形式はより柔軟です。これらを使用して、UUID を使用するような AWS IoT SiteWise API オペレーションでオブジェクトを参照できます。

UUID と同様、各外部 ID はコンテキスト内で一意である必要があります。例えば、同じ外部 ID を持つ 2 つのアセットモデルを持つことはできません。また、UUID と同様、オブジェクトの存続期間中の外部 ID は 1 つだけであり、変更することはできません。

外部 ID と UUID の違い

外部 ID と UUID は以下の点で異なります。

- すべてのオブジェクトは UUID を持ちますが、外部 ID はオプションです。
- AWS IoT SiteWise は外部 IDsを生成しません。これらはユーザーが自分で指定します。
- オブジェクトにまだない場合は、いつでも外部 ID を割り当てることができます。

外部 ID の形式

有効な外部 ID は以下のプロパティを持ちます。

- 2 ~ 128 文字の長さです。
- 最初と最後の文字は英数字 (A~Z、a~z、0~9) である必要があります。
- 最初と最後の文字以外の文字は英数字、または次のいずれかである必要があります。`_-.:`

例えば、外部 ID は次の正規表現に準拠している必要があります。

```
[a-zA-Z0-9][a-zA-Z0-9_\-.:]*[a-zA-Z0-9]+
```

外部 ID を持つリファレンスオブジェクト

UUID を使用してオブジェクトを参照できる多くの場所では、外部 ID がある場合、外部 ID を代わりに使用できます。このためには、文字列 `externalId:` に外部 ID を追加します。

例えば、UUID (アセットモデル ID) が `a1b2c3d4-5678-90ab-cdef-11111EXAMPLE` のアセットモデルがあるとします。このアセットモデルには、外部 ID `myExternalId` もあります。DescribeAssetModel を呼び出して詳細を取得します。`assetModelId` の値として、次のいずれかを使用できます。

- アセットモデル ID (UUID) 自体の場合: `a1b2c3d4-5678-90ab-cdef-11111EXAMPLE`
- 外部 ID の場合: `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:` プレフィックス自体は外部 ID の一部ではありません。プレフィックスを指定する必要があるのは、UUID または外部 ID を受け入れる API オペレーションに外部 ID を指定するときだけです。例えば、既存のオブジェクトをクエリまたは更新するときにプレフィックスを指定します。
> アセットモデルの作成時など、オブジェクトの外部 ID を定義する場合は、プレフィックスを含めないでください。

この方法で UUIDs の代わりに外部 IDs を の多くの API オペレーションに使用できますが AWS IoT SiteWise、すべてではありません。例えば、GetAssetPropertyValue は UUID を使用する必要があります。これは外部 ID の使用をサポートしていません。

特定の API オペレーションがこの使用をサポートしているかどうかを確認するには、API リファレンスを参照してください。

のアセットモデルとコンポーネントモデルを作成する AWS IoT SiteWise

AWS IoT SiteWise アセットモデルとコンポーネントモデルは、産業データの標準化を促進します。アセットモデルとコンポーネントモデルは、産業用アセットとそのコンポーネントの構造とプロパティを表します。アセットモデルは、風力タービンや製造ラインなどのアセット全体を定義します。コンポーネントモデルは、ブレード、ジェネレーター、センサーなど、アセットを構成する個々のコンポーネントを表します。これらのモデルを作成することで、産業用機器の実際の関係と階層を反映する方法でアセットデータを整理して構造化できるため、モニタリング、分析、保守が容易になります。

アセットモデルまたはコンポーネントモデルには、名前、説明、アセットプロパティ、および (オプションで) プロパティをグループ化し、またはサブアセンブリのコンポーネントモデルを参照するカスタム複合モデルが含まれます。

では AWS IoT SiteWise、産業アセットとそのコンポーネントの構造とプロパティを表すアセットモデルとコンポーネントモデルを作成できます。

- アセットモデルを使用してアセットを作成します。上記の機能に加えて、アセットモデルには、アセット間の関係を定義する階層定義を含めることもできます。
- コンポーネントモデルは、アセットモデルまたは別のコンポーネントモデル内のサブアセンブリを表します。コンポーネントモデルを作成するときには、アセットモデルや他のコンポーネントモデルで参照を追加できます。ただし、コンポーネントモデルから直接アセットを作成することはできません。

アセットモデルまたはコンポーネントモデルを作成したら、カスタム複合モデルを作成して、プロパティをグループ化したり、既存のコンポーネントモデルを参照したりできます。

アセットモデルとコンポーネントモデルの作成方法の詳細については、以下のセクションを参照してください。

トピック

- でアセットモデルを作成する AWS IoT SiteWise
- コンポーネントモデルを作成する
- データのプロパティを定義する
- カスタム複合モデルを作成する (コンポーネント)

でアセットモデルを作成する AWS IoT SiteWise

AWS IoT SiteWise アセットモデルは、産業データの標準化を促進します。アセットモデルには、名前、説明、アセットプロパティ、およびアセット階層定義が含まれます。たとえば、気温、1 分あたりの回転数 (RPM) 、および電力特性を使用して風力タービンモデルを定義できます。次に、正味電力出力特性と風力タービンの階層定義を使用して、風力発電所モデルを定義できます。

> **Note**
>
> - 最下位レベルのノードから業務をモデル化することをお勧めします。たとえば、風力発電所モデルを作成する前に、風力タービンモデルを作成します。アセット階層定義には、既存のアセットモデルへの参照が含まれます。この方法により、モデルの作成時にアセット階層を定義できます。
> - アセットモデルに他のアセットモデルを含めることはできません。別のモデル内でサブアセンブリとして参照できるモデルを定義する必要がある場合は、代わりにコンポーネン

ト-->モデルを作成する必要があります。詳細については、「コンポーネントモデルを作成する」を参照してください。

以下のセクションでは、 AWS IoT SiteWise コンソールまたは API を使用してアセットモデルを作成する方法について説明します。次のセクションでは、モデルの作成に使用できるさまざまなタイプのアセットプロパティとアセット階層についても説明します。

トピック

- アセットモデルを作成する (コンソール)
- アセットモデルを作成する (AWS CLI)
- アセットモデルの例
- アセットモデルの階層を定義する

アセットモデルを作成する (コンソール)

AWS IoT SiteWise コンソールを使用してアセットモデルを作成できます。 AWS IoT SiteWise コンソールには、数式の自動補完など、有効なアセットモデルの定義に役立つさまざまな機能が用意されています。

アセットモデルを作成するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. アセットモデルの作成 を選択します。
4. [モデルの作成] ページで、次の操作を行います。

   a. アセットモデルの [名前] を入力します (**`Wind Turbine`** または **`Wind Turbine Model`** など)。この名前は、このリージョンのアカウントのすべてのモデルで一意である必要があります。

   b. (オプション) モデルの外部 ID を追加します。これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

   c. (オプション) モデルの [測定の定義] を追加します。測定値は、機器からのデータストリームを表します。詳細については、「機器からのデータストリームを定義する (測定値) 」を参照してください。

d. (オプション) モデルの [定義を変換する] を追加します。変換とは、あるフォームから別のフォームにデータをマッピングするための数式です。詳細については、「データを変換する (変換) 」を参照してください。

e. (オプション) モデルの [メトリクスの定義] を追加します。メトリクスとは、時間間隔でのデータを集計する数式です。メトリクスは、関連するアセットからデータを入力できるため、オペレーションまたはオペレーションのサブセットを表す値をコンピューティングすることができます。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。

f. (オプション) モデルの [階層の定義] を追加します。階層とは、アセット間の関係です。詳細については、「アセットモデルの階層を定義する」を参照してください。

g. (オプション) アセットモデルのタグを追加します。詳細については、「AWS IoT SiteWise リソースにタグを付ける」を参照してください。

h. [モデルの作成] を選択します。

アセットモデルを作成すると、 AWS IoT SiteWise コンソールは新しいモデルのページに移動します。このページで、モデルの [ステータス] が表示されます。このステータスは、最初は [作成中] です。このページは自動的に更新されるため、モデルのステータスが更新されるまで待つことができます。

> **Note**
>
> 複雑なモデルの場合は、アセットモデルの作成プロセスに数分かかることがあります。アセットモデルの状態が[ACTIVE] (アクティブ) になると、アセットモデルを使用してアセットを作成できるようになります。詳細については、「アセットおよびモデルの状態」を参照してください。

5. (オプション) アセットモデルを作成したら、アセットモデルをエッジに設定することができます。SiteWise Edge の詳細については、「AWS IoT SiteWise Edge でエッジ機能を設定する」を参照してください。

   a. モデルページで、[Configure for Edge] (エッジの設定) を選択します。

   b. モデル設定ページで、モデルのエッジ設定を選択します。これにより、 がこのアセットモデルに関連付けられたプロパティを計算して保存 AWS IoT SiteWise できる場所が制御されます。エッジ用のモデルの設定については、SiteWise Edge で OPC UA ソースを設定する を参照してください。

c. カスタムエッジ設定で、各アセットモデルプロパティを AWS IoT SiteWise 計算して保存する場所を選択します。

> **Note**
>
> 関連する変換とメトリクスは、同じ場所に設定されている必要があります。エッジ用のモデルの設定については、SiteWise Edge で OPC UA ソースを設定する を参照してください。

d. [保存] を選択します。モデルページで、[Edge configuration] (Edgeの設定) が[Configured] (設定済み) になっていることを確認します。

アセットモデルを作成する (AWS CLI)

AWS Command Line Interface（AWS CLI) を使用してアセットモデルを作成できます。

CreateAssetModel オペレーションを使用して、プロパティと階層を持つアセットモデルを作成します。このオペレーションでは、次の構造を持つペイロードが必要です。

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition
}
```

アセットモデルを作成するには (AWS CLI)

1. `asset-model-payload.json` という名前のファイルを作成して、次の JSON オブジェクトをファイルにコピーします。

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ],
  "assetModelHierarchies": [
```

```
  ],
  "assetModelCompositeModels": [

  ]
}
```

2. 任意の JSON テキストエディタを使用して、以下の `asset-model-payload.json` ファイルを編集します。

   a. アセットモデルの名前 (`assetModelName`) を入力します (**`Wind Turbine`** または **`Wind Turbine Model`** など)。この名前は、この AWS リージョンのアカウントのすべてのアセットモデルとコンポーネントモデルで一意である必要があります。

   b. (オプション) アセットモデルの外部 ID (`assetModelExternalId`) を入力します。これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

   c. (オプション) アセットモデルの説明 (`assetModelDescription`) を入力するか、`assetModelDescription` キーと値のペアを削除します。

   d. (オプション) モデルのアセットプロパティ (`assetModelProperties`) を定義します。詳細については、「データのプロパティを定義する」を参照してください。

   e. (オプション) モデルのアセット階層 (`assetModelHierarchies`) を定義します。詳細については、「アセットモデルの階層を定義する」を参照してください。

   f. (オプション) モデルのアラームを定義します。アラームは、他のプロパティをモニタリングし、機器やプロセスに注意が必要な時期を特定することができます。各アラーム定義は、アラームが使用する一連のプロパティを標準化した複合モデル (`assetModelCompositeModels`) である。詳細については、「でアラームを使用してデータをモニタリングする AWS IoT SiteWise」および「でアセットモデルにアラームを定義する AWS IoT SiteWise」を参照してください。

   g. (オプション) アセットモデルのタグ (`tags`) を追加します。詳細については、「AWS IoT SiteWise リソースにタグを付ける」を参照してください。

3. 次のコマンドを実行して、JSON ファイルの定義からアセットモデルを作成します。

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-payload.json
```

このオペレーションは、アセットを作成するときに参照する `assetModelId` を含むレスポンスを返します。レスポンスには、モデルの状態 (`assetModelStatus.state`) も含まれま

す。これは、最初は CREATING です。アセットモデルのステータスは、変更が反映されるまで CREATING です。

> **Note**
>
> 複雑なモデルの場合は、アセットモデルの作成プロセスに数分かかることがあります。アセットモデルの現在の状態をチェックするには、assetModelId を指定して DescribeAssetModel オペレーションを使用します。アセットモデルのステータスが ACTIVE になったら、アセットモデルを使用してアセットを作成できます。詳細については、「アセットおよびモデルの状態」を参照してください。

4. (オプション) アセットモデルのカスタム複合モデルを作成します。カスタム複合モデルでは、モデル内のプロパティをグループ化したり、コンポーネントモデルを参照してサブアセンブリを含めたりすることができます。詳細については、「カスタム複合モデルを作成する (コンポーネント)」を参照してください。

アセットモデルの例

このセクションでは、 AWS CLI および AWS IoT SiteWise SDKs でアセットモデルを作成するために使用できるアセットモデル定義の例を示します。これらのアセットモデルは、風力タービンと風力発電所を表しています。風力タービンは、センサーの生データを取り込み、電力や平均風速などの値を算出する。風力発電アセットでは、風力発電所内のすべての風車の総電力量などを算出します。

トピック

- 風力タービンアセットモデル
- 風力発電所アセットモデル

風力タービンアセットモデル

次のアセットモデルは、風力発電所のタービンを表しています。風力タービンはセンサーのデータを取り込み、電力や平均風速などの値を算出する。

> **Note**
>
> このサンプルモデルは、 AWS IoT SiteWise デモの風力タービンモデルに似ています。詳細については、「AWS IoT SiteWise デモを使用する」を参照してください。

```
{
  "assetModelType": "ASSET_MODEL",
  "assetModelName": "Wind Turbine Asset Model",
  "assetModelDescription": "Represents a turbine in a wind farm.",
  "assetModelProperties": [
    {
      "name": "Location",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Renton"
        }
      }
    },
    {
      "name": "Make",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Amazon"
        }
      }
    },
    {
      "name": "Model",
      "dataType": "INTEGER",
      "type": {
        "attribute": {
          "defaultValue": "500"
        }
      }
    },
    {
      "name": "Torque (KiloNewton Meter)",
      "dataType": "DOUBLE",
      "unit": "kNm",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "Wind Direction",
      "dataType": "DOUBLE",
```

```
      "unit": "Degrees",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "RotationsPerMinute",
      "dataType": "DOUBLE",
      "unit": "RPM",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "Wind Speed",
      "dataType": "DOUBLE",
      "unit": "m/s",
      "type": {
        "measurement": {}
      }
    },
    {
      "name": "RotationsPerSecond",
      "dataType": "DOUBLE",
      "unit": "RPS",
      "type": {
        "transform": {
          "expression": "rpm / 60",
          "variables": [
            {
              "name": "rpm",
              "value": {
                "propertyId": "RotationsPerMinute"
              }
            }
          ]
        }
      }
    },
    {
      "name": "Overdrive State",
      "dataType": "DOUBLE",
      "type": {
        "transform": {
```

```
          "expression": "gte(torque, 3)",
          "variables": [
            {
              "name": "torque",
              "value": {
                "propertyId": "Torque (KiloNewton Meter)"
              }
            }
          ]
        }
      }
    },
    {
      "name": "Average Power",
      "dataType": "DOUBLE",
      "unit": "Watts",
      "type": {
        "metric": {
          "expression": "avg(torque) * avg(rps) * 2 * 3.14",
          "variables": [
            {
              "name": "torque",
              "value": {
                "propertyId": "Torque (Newton Meter)"
              }
            },
            {
              "name": "rps",
              "value": {
                "propertyId": "RotationsPerSecond"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "5m"
            }
          }
        }
      }
    },
    {
      "name": "Average Wind Speed",
      "dataType": "DOUBLE",
```

```
      "unit": "m/s",
      "type": {
        "metric": {
          "expression": "avg(windspeed)",
          "variables": [
            {
              "name": "windspeed",
              "value": {
                "propertyId": "Wind Speed"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "5m"
            }
          }
        }
      }
    },
    {
      "name": "Torque (Newton Meter)",
      "dataType": "DOUBLE",
      "unit": "Nm",
      "type": {
        "transform": {
          "expression": "knm * 1000",
          "variables": [
            {
              "name": "knm",
              "value": {
                "propertyId": "Torque (KiloNewton Meter)"
              }
            }
          ]
        }
      }
    },
    {
      "name": "Overdrive State Time",
      "dataType": "DOUBLE",
      "unit": "Seconds",
      "type": {
        "metric": {
```

```
          "expression": "statetime(overdrive_state)",
          "variables": [
            {
              "name": "overdrive_state",
              "value": {
                "propertyId": "Overdrive State"
              }
            }
          ],
          "window": {
            "tumbling": {
              "interval": "5m"
            }
          }
        }
      }
    }
  ],
  "assetModelHierarchies": []
}
```

風力発電所アセットモデル

次のアセットモデルは、複数の風力タービンで構成される風力発電所を表しています。このアセットモデルは、風力タービンモデルへの[hierarchy] (階層) を定義しています。これにより、風力発電所内のすべての風力タービンのデータから数値 (平均出力など) を算出することができる。

> **Note**
>
> このサンプルモデルは、 AWS IoT SiteWise デモの風力発電所モデルに似ています。詳細については、「AWS IoT SiteWise デモを使用する」を参照してください。

このアセットモデルは、風力タービンアセットモデル によって異なります。`propertyId` および `childAssetModelId` の値を、既存の風力タービンアセットモデルの値に置き換えます。

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

アセットモデルの階層を定義する

アセットモデル階層を定義することで、産業オペレーションにおけるアセットモデル間の論理的な関連付けを作成することができます。例えば、陸上風力発電所と洋上風力発電所からなる風力発電所を定義することができます。陸上風力発電所には、タービンと陸上の場所が含まれます。洋上風力発電所には、タービンと沖合の場所が含まれます。

子アセットモデルを親アセットモデルに階層的に関連付けると、親アセットモデルのメトリクスは子アセットモデルのメトリクスからデータを入力できるようになります。アセットモデル階層とメトリクスを使用して、オペレーションまたはオペレーションのサブセットに関するインサイトを提供する統計情報をコンピューティングできます。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。

各階層は、親アセットモデルと子アセットモデルの関係を定義します。親アセットモデルでは、同じ子アセットモデルに対して複数の階層を定義することができます。例えば、風力発電所に2種類の風

カタービンがあり、すべての風力タービンが同じアセットモデルで表現されている場合、それぞれの種類に階層を定義することができます。そして、風力発電所のモデルで指標を定義し、風力タービンの種類ごとに独立した統計量と複合的な統計量をコンピューティングすることができます。

親アセットモデルには、複数の子アセットモデルを関連付けることができます。例えば、陸上風力発電所と洋上風力発電所が2つの異なるアセットモデルで表現されている場合、これらのアセットモデルを同じ親風力発電所アセットモデルに関連付けることができます。

また、子アセットモデルは、複数の親アセットモデルと関連付けることができます。例えば、2種類の風力発電所があり、すべての風力タービンが同じアセットモデルで表現されている場合、風力タービンアセットモデルと異なる風力発電所アセットモデルを関連付けることができます。

> **Note**
>
> アセットモデル階層を定義する場合、子アセットモデルは `ACTIVE` であるか、以前の `ACTIVE` バージョンである必要があります。詳細については、「アセットおよびモデルの状態」を参照してください。

階層アセットモデルを定義してアセットを作成したら、アセットを関連付けて親子関係を完了できます。詳細については、「でアセットモデルのアセットを作成する AWS IoT SiteWise」および「アセットの関連付けと関連付け解除」を参照してください。

トピック

- アセットモデル階層を定義する (コンソール)
- アセット階層を定義する (AWS CLI)

アセットモデル階層を定義する (コンソール)

AWS IoT SiteWise コンソールでアセットモデルの階層を定義するときは、次のパラメータを指定します。

- [Hierarchy name] (階層名) - 階層の名前 (**`Wind Turbines`** など)。
- [Hierarchy model] (階層モデル) - 子アセットモデル。
- 階層外部 ID (オプション) – これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

詳細については、「アセットモデルを作成する (コンソール) 」を参照してください。

アセット階層を定義する (AWS CLI)

AWS IoT SiteWise API を使用してアセットモデルの階層を定義する場合は、次のパラメータを指定します。

- name 階層の名前 (など)。**Wind Turbines**
- childAssetModelId - 階層の子アセットモデルの ID または外部 ID。ListAssetModels オペレーションを使用して、既存のアセットモデルの ID を検索できます。

Example 階層定義の例

次の例は、風力発電所の風力タービンとの関係を表すアセットモデル階層を示しています。このオブジェクトは、AssetModelHierarchy の例です。詳細については、「アセットモデルを作成する (AWS CLI)」を参照してください。

```
{
  ...
  "assetModelHierarchies": [
    {
      "name": "Wind Turbines",
      "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
    },
  ]
}
```

コンポーネントモデルを作成する

AWS IoT SiteWise コンポーネントモデルを使用して、アセットモデルや他のコンポーネントモデルから参照できるサブアセンブリを定義します。このようにして、コンポーネントの定義を他の複数のモデル間で、または同じモデル内で複数回再利用できます。

コンポーネントモデルを定義するプロセスは、アセットモデルを定義するのと非常に似ています。アセットモデルと同様に、コンポーネントモデルには名前、説明、アセットプロパティがあります。ただし、コンポーネントモデル自体を使用してアセットを直接作成することはできないため、コンポーネントモデルにアセット階層定義を含めることはできません。また、コンポーネントモデルではアラームを定義できません。

例えば、モーター温度、エンコーダー温度、絶縁抵抗プロパティを持つサーボモーターのコンポーネントを定義できます。次に、CNC マシンなど、サーボモーターを含む機器のアセットモデルを定義できます。

> **Note**
>
> - 最下位レベルのノードから業務をモデル化することをお勧めします。例えば、CNC マシンのアセットモデルを作成する前に、サーボモーターコンポーネントを作成します。アセットモデルには、既存のコンポーネントモデルへの参照が含まれています。
> - コンポーネントモデルから直接アセットを作成することはできません。コンポーネントを使用するアセットを作成するには、アセットのアセットモデルを作成する必要があります。次に、コンポーネントを参照するカスタム複合モデルを作成します。アセットモデルの作成の詳細については、「でアセットモデルを作成する AWS IoT SiteWise」を参照してください。カスタム複合モデルの作成の詳細については、「カスタム複合モデルを作成する (コンポーネント)」を参照してください。

以下のセクションでは、 AWS IoT SiteWise API を使用してコンポーネントモデルを作成する方法について説明します。

トピック

- コンポーネントモデルを作成する (AWS CLI)
- コンポーネントモデルの例

コンポーネントモデルを作成する (AWS CLI)

AWS Command Line Interface（ AWS CLI) を使用してコンポーネントモデルを作成できます。

CreateAssetModel オペレーションを使用して、プロパティを持つコンポーネントモデルを作成します。このオペレーションでは、次の構造を持つペイロードが必要です。

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
}
```

コンポーネントモデルを作成するには (AWS CLI)

1. `component-model-payload.json` という名前のファイルを作成して、次の JSON オブジェクトをファイルにコピーします。

```
{
  "assetModelType": "COMPONENT_MODEL",
  "assetModelName": "",
  "assetModelDescription": "",
  "assetModelProperties": [

  ]
}
```

2. 任意の JSON テキストエディタを使用して、以下の `component-model-payload.json` ファイルを編集します。

   a. コンポーネントモデルの名前 (`assetModelName`) を入力します (**`Servo Motor`** または **`Servo Motor Model`** など)。この名前は、この AWS リージョンのアカウントのすべてのアセットモデルとコンポーネントモデルで一意である必要があります。

   b. (オプション) コンポーネントモデルの外部 ID (`assetModelExternalId`) を入力します。これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

   c. (オプション) アセットモデルの説明 (`assetModelDescription`) を入力するか、`assetModelDescription` キーと値のペアを削除します。

   d. (オプション) コンポーネントモデルのアセットプロパティ (`assetModelProperties`) を定義します。詳細については、「データのプロパティを定義する」を参照してください。

   e. (オプション) アセットモデルのタグ (`tags`) を追加します。詳細については、「AWS IoT SiteWise リソースにタグを付ける」を参照してください。

3. 次のコマンドを実行して、JSON ファイルの定義からコンポーネントモデルを作成します。

```
aws iotsitewise create-asset-model --cli-input-json file://component-model-
payload.json
```

オペレーションは、アセットモデルまたは別のコンポーネントモデルでコンポーネントモデルにリファレンスを追加するときに参照する `assetModelId` を含むレスポンスを返します。レスポンスには、モデルの状態 (`assetModelStatus.state`) も含まれます。これは、最初は `CREATING` です。コンポーネントモデルのステータスは、変更が伝播されるまで `CREATING` です。

> **Note**
>
> 複雑なモデルの場合、コンポーネントモデルの作成プロセスに数分かかることがあります。コンポーネントモデルの現在のステータスを確認するには、`assetModelId` を指定して DescribeAssetModel オペレーションを使用します。コンポーネントモデルのステータスが `ACTIVE` になったら、アセットモデルまたは他のコンポーネントモデルでコンポーネントモデルへの参照を追加できます。詳細については、「アセットおよびモデルの状態」を参照してください。

4. (オプション) コンポーネントモデルのカスタム複合モデルを作成します。カスタム複合モデルでは、モデル内のプロパティをグループ化したり、別のコンポーネントモデルを参照することでサブアセンブリを含めたりすることができます。詳細については、「カスタム複合モデルを作成する (コンポーネント)」を参照してください。

コンポーネントモデルの例

このセクションでは、 AWS CLI および AWS IoT SiteWise SDKs でコンポーネントモデルを作成するために使用できるコンポーネントモデル定義の例を示します。このコンポーネントモデルは、CNC マシンなど、別の機器で使用できるサーボモーターを表します。

トピック

- サーボモーターコンポーネントモデル

サーボモーターコンポーネントモデル

次のコンポーネントモデルは、CNC マシンなどの機器で使用できるサーボモーターを表します。サーボモーターは、温度や電気抵抗など、さまざまな測定値を提供します。これらの測定値は、サーボモーターコンポーネントモデルを参照するアセットモデルから作成されたアセットのプロパティとして使用できます。

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

データのプロパティを定義する

アセットプロパティは、アセットデータを含む各アセット内の構造です。アセットプロパティには、次の型があります。

- [Attributes] (属性) – デバイスの製造元や地理的なリージョンなど、一般的に静的なプロパティを持つアセット。詳細については、「静的データ (属性) を定義する」を参照してください。
- [測定値] - タイムスタンプ付きの回転速度値や摂氏のタイムスタンプ付きの温度値など、アセットの raw デバイスのセンサーデータストリーム。測定は、データストリームエイリアスによって定義されます。詳細については、「機器からのデータストリームを定義する (測定値) 」を参照してください。
- [Transforms] (変換) - タイムスタンプ付き華氏温度値など、アセットの変換された時系列値。変換は、式とその式で使用する変数によって定義されます。詳細については、「データを変換する (変換) 」を参照してください。
- [Metrics] (メトリクス) - 時間当たりの平均温度など、指定した時間間隔で集計されたアセットのデータ。メトリクスは、時間間隔、式、およびその式で使用する変数によって定義されます。メトリック表現は、関連するアセットのメトリクスプロパティを入力できるため、オペレーションまたはオペレーションのサブセットを表すメトリックをコンピューティングすることができます。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。

詳細については、「でアセットモデルを作成する AWS IoT SiteWise」を参照してください。

測定、変換、およびメトリクスを使用して、総合設備効率 (OEE) をコンピューティングする方法の例については、「で OEE を計算する AWS IoT SiteWise」を参照してください。

トピック

- 静的データ (属性) を定義する
- 機器からのデータストリームを定義する (測定値)
- データを変換する (変換)
- プロパティおよびその他のアセットからのデータを集計する (メトリクス)
- 数式を使用する

静的データ (属性) を定義する

アセット属性は、デバイスの製造元や地理的位置など、一般的に静的な情報を表します。アセットモデルから作成する各アセットには、このモデルの属性が含まれています。

トピック

- 属性を定義する (コンソール)
- 属性を定義する (AWS CLI)

属性を定義する (コンソール)

AWS IoT SiteWise コンソールでアセットモデルの属性を定義するときは、次のパラメータを指定します。

- [Name] (名前) - プロパティの名前。
- [Default value] (デフォルト値) –(オプション) この属性のデフォルト値。モデルから作成されたアセットは、属性に対してこの値を持ちます。モデルから作成されたアセットのデフォルト値を上書きする方法の詳細については、「属性値を更新する」を参照してください。
- [Data type] (データ型) - プロパティのデータ型で、次のいずれかになります。
  - [String] (文字列) – 最大 1024 バイトの文字列。
  - [Integer] (整数) - [-2,147,483,648, 2,147,483,647] の範囲を持つ 32 ビットの符合付き整数。
  - [Double] (倍) – [-10^100、10^100] の範囲および IEEE 754 倍精度の浮動小数点数。
  - [Boolean] (ブール値) - `true` または `false`。
- 外部 ID – (オプション) これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

詳細については、「アセットモデルを作成する (コンソール) 」を参照してください。

属性を定義する (AWS CLI)

AWS IoT SiteWise API を使用してアセットモデルの属性を定義する場合は、次のパラメータを指定します。

- `name` - プロパティの名前。
- `defaultValue` -(オプション) この属性のデフォルト値。モデルから作成されたアセットは、属性に対してこの値を持ちます。モデルから作成されたアセットのデフォルト値を上書きする方法の詳細については、「属性値を更新する」を参照してください。
- `dataType` - プロパティのデータ型。次のいずれかです。
  - `STRING` - 最大 1024 バイトの文字列。
  - `INTEGER` - [-2,147,483,648, 2,147,483,647] の範囲を持つ 32 ビットの符合付き整数。
  - `DOUBLE` - [-10 ^ 100、10 ^ 100] の範囲および IEEE 754 倍精度の浮動小数点数。
  - `BOOLEAN` – `true` または `false`
- `externalId` – (オプション) これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

Example 属性定義の例

次の例では、デフォルト値でアセットのモデル番号を表す属性を示します。このオブジェクトは、[Attribute] (属性) を含む AssetModelProperty の例である。このオブジェクトをCreateAssetModel リクエストペイロードの一部として指定して、属性プロパティを作成できます。詳細については、「アセットモデルを作成する (AWS CLI)」を参照してください。

```
{
...
"assetModelProperties": [
{
  "name": "Model number",
  "dataType": "STRING",
  "type": {
    "attribute": {
      "defaultValue": "BLT123"
    }
  }
}
],
```

```
...
}
```

機器からのデータストリームを定義する (測定値)

測定値は、タイムスタンプ付き温度値やタイムスタンプ付きの 1 分あたりの回転数 (RPM) 値など、デバイスの raw センサーデータストリームを表します。

トピック

- 測定値を定義する (コンソール)
- 測定値を定義する (AWS CLI)

測定値を定義する (コンソール)

AWS IoT SiteWise コンソールでアセットモデルの測定値を定義するときは、次のパラメータを指定します。

- [Name] (名前) - プロパティの名前。
- [Unit] (単位) - プロパティの科学単位 (mm、摂氏など)。
- [Data type] (データ型) - プロパティのデータ型で、次のいずれかになります。
  - [String] (文字列) – 最大 1024 バイトの文字列。
  - [Integer] (整数) - [-2,147,483,648, 2,147,483,647] の範囲を持つ 32 ビットの符合付き整数。
  - [Double] (倍) – [-10^100、10^100] の範囲および IEEE 754 倍精度の浮動小数点数。
  - [Boolean] (ブール値) - `true` または `false`。
- 外部 ID – (オプション) これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

詳細については、「アセットモデルを作成する (コンソール) 」を参照してください。

測定値を定義する (AWS CLI)

AWS IoT SiteWise API を使用してアセットモデルの測定値を定義する場合は、次のパラメータを指定します。

- `name` - プロパティの名前。
- `dataType` - プロパティのデータ型。次のいずれかです。
  - `STRING` - 最大 1024 バイトの文字列。

- INTEGER - [-2,147,483,648, 2,147,483,647] の範囲を持つ 32 ビットの符合付き整数。
- DOUBLE - [-10 ^ 100、10 ^ 100] の範囲および IEEE 754 倍精度の浮動小数点数。
- BOOLEAN – true または false
- unit –(オプション) プロパティの科学単位 (mm、摂氏など)。
- externalId – (オプション) これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

Example 測定値の定義の例

次の例は、アセットの温度センサーの読み取り値を表す測定値を示しています。このオブジェクトは、[Measurement] (測定値) を含むAssetModelProperty の例である。このオブジェクトをCreateAssetModel リクエストペイロードの一部として指定して、測定プロパティを作成できます。詳細については、「アセットモデルを作成する (AWS CLI)」を参照してください。

アセットモデルを定義するときは、[Measurement] (測定値) は空の構造になります。これは、後で各アセットが一意のデバイスデータストリームを使用するように設定するためです。アセットの測定プロパティをデバイスのセンサーデータストリームに接続する方法の詳細については、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

```
{
      ...
      "assetModelProperties": [
      {
          "name": "Temperature C",
          "dataType": "DOUBLE",
          "type": {
              "measurement": {}
          },
          "unit": "Celsius"
      }
  ],
      ...
}
```

データを変換する (変換)

変換は、アセットプロパティのデータポイントを別のフォームにマッピングする数式です。変換式は、アセットプロパティ変数、リテラル、演算子、関数で構成されます。変換されたデータポイント

は、入力データポイントと one-to-one の関係を保持します。 は、入力プロパティが新しいデータポイントを受信するたびに、新しい変換されたデータポイントを AWS IoT SiteWise 計算します。

> **Note**
>
> 同じタイムスタンプを持つプロパティ更新の場合、出力値は他の受信プロパティからの更新によって上書きされる可能性があります。

たとえば、アセットに単位が摂氏の名前 `Temperature_C` の温度測定ストリームがある場合、数式 `Temperature_F = 9/5 * Temperature_C + 32` を使用して各データポイントを華氏に変換できます。が`Temperature_C`測定ストリームでデータポイント AWS IoT SiteWise を受信するたびに、対応する`Temperature_F`値は数秒以内に計算され、 `Temperature_F`プロパティとして使用できます。

変換に複数の変数が含まれる場合、先に到着したデータポイントからすぐにコンピューティングが開始されます。例えば、ある部品メーカーが製品の品質をモニタリングするためにトランスを使用する場合を考えてみましょう。部品の種類によって異なる基準を用いて、メーカーは次のような測定値で工程を表現しています。

- `Part_Number` - 部品のタイプを識別する文字列。
- `Good_Count` - 部品が規格に適合している場合、1 だけ増加する整数。
- `Bad_Count` - 部品が規格に適合しない場合、1 だけ増加する整数。

また、メーカーは `Quality_Monitor` に等しい変換 `if(eq(Part_Number, "BLT123") and (Bad_Count / (Good_Count + Bad_Count) > 0.1), "Caution", "Normal")` を作成します。

この変換は、特定の部品型で生産された不良部品の割合をモニタリングします。部品番号がBLT123で、不良部品の割合が10% (0.1) を超える場合、変換は `"Caution"` を返します。そうでない場合は、変換は `"Normal"` を返します。

> **Note**
>
> - `Part_Number` が他の測定値より先に新しいデータポイントを受信した場合、`NQuality_Monitor`変換は新しい `Part_Number` 値と最新の `Good_Count` および `Bad_Count` 値を使用します。エラーを回避するため、次の製造の前に `Good_Count`、`Bad_Count` をリセットしてください。

- すべての変数が新しいデータポイントを受け取った後にのみ式を評価したい場合は、[metrics] (メトリクス) を使用します。

トピック

- 変換を定義する (コンソール)
- 変換を定義する (AWS CLI)

変換を定義する (コンソール)

AWS IoT SiteWise コンソールでアセットモデルの変換を定義するときは、次のパラメータを指定します。

- [Name] (名前) - プロパティの名前。
- [Unit] (単位) - プロパティの科学単位 (mm、摂氏など)。
- [Data type] (データ型) - 変換のデータ型で、[Double] (倍) または [String] (文字列) です。
- 外部 ID – (オプション) これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。
- [Formula] (コンピューティング式) – 変換式。変換式では、集計関数や一時関数は使用できません。オートコンプリート機能を開くには、入力を開始するか、下矢印キーを押します。詳細については、「数式を使用する」を参照してください。

> **Important**
>
> 変換は、整数、倍、ブール値、文字列型のプロパティを入力することができます。ブール値は、0 (false) 、1 (true) に変換されます。
> 変換には、属性でないプロパティを1つ以上、任意の数だけ入力する必要があります。
> AWS IoT SiteWise は、属性でない入力プロパティが新しいデータポイントを受け取るたびに、新しい変換後のデータポイントをコンピューティングする。新しい属性値では、変換の更新は開始されません。アセットプロパティデータの API オペレーションのリクエストレートは、変換計算の結果にも適用されます。
> 数式では、倍値または文字列値のみを出力できます。入れ子になった式は、文字列など他のデータ型を出力することができますが、式全体としては、数値または文字列として評価されなければなりません。[jp function] (jp関数) を使うと、文字列を数値に変換すること

ができます。ブール値は 1 (true) または 0 (false) でなければならない。詳細については、「未定義の値、無限の値、およびオーバーフロー値」を参照してください。

詳細については、「アセットモデルを作成する (コンソール) 」を参照してください。

変換を定義する (AWS CLI)

AWS IoT SiteWise API を使用してアセットモデルの変換を定義する場合は、次のパラメータを指定します。

- `name` - プロパティの名前。
- `unit` –(オプション) プロパティの科学単位 (mm、摂氏など)。
- `dataType` - 変換のデータ型は、`DOUBLE` または `STRING` である必要があります。
- `externalId` – (オプション) これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。
- `expression` - 変換式。変換式では、集計関数や一時関数は使用できません。詳細については、「数式を使用する」を参照してください。
- `variables` - 式で使用するアセットの他のプロパティを定義する変数のリスト。各変数構造には、式で使用する単純な名前と、その変数にリンクされるプロパティを定義する `value` 構造が含まれます。`value` 構造体には、以下の情報が含まれています。
  - `propertyId` - 値を入力するプロパティの ID。ID の代わりにプロパティの名前を使用できます。

> **Important**
>
> 変換は、整数、倍、ブール値、文字列型のプロパティを入力することができます。ブール値は、`0` (false) 、1 (true) に変換されます。
> 変換には、属性でないプロパティを1つ以上、任意の数だけ入力する必要があります。
> AWS IoT SiteWise は、属性でない入力プロパティが新しいデータポイントを受け取るたびに、新しい変換後のデータポイントをコンピューティングする。新しい属性値では、変換の更新は開始されません。アセットプロパティデータの API オペレーションのリクエストレートは、変換計算の結果にも適用されます。
> 数式では、倍値または文字列値のみを出力できます。入れ子になった式は、文字列など他のデータ型を出力することができますが、式全体としては、数値または文字列として評価されなければなりません。[jp function] (jp関数) を使うと、文字列を数値に変換すること

ができます。ブール値は 1 (true) または 0 (false) でなければならない。詳細については、「未定義の値、無限の値、およびオーバーフロー値」を参照してください。

Example [Transform job definition] (変換定義)

次の例は、アセットの温度測定データを摂氏から華氏に変換する変換プロパティを示しています。このオブジェクトは、[Transform] (変換) を含む AssetModelProperty の例である。このオブジェクトをCreateAssetModel リクエストペイロードの一部として指定して、変換プロパティを作成できます。詳細については、「アセットモデルを作成する (AWS CLI)」を参照してください。

```
{
...
"assetModelProperties": [
...
{
  "name": "Temperature F",
  "dataType": "DOUBLE",
  "type": {
    "transform": {
      "expression": "9/5 * temp_c + 32",
      "variables": [
        {
          "name": "temp_c",
          "value": {
            "propertyId": "Temperature C"
          }
        }
      ]
    }
  },
  "unit": "Fahrenheit"
}
],
...
}
```

Example 3つの変数を含む変換定義。

次の例では、BLT123の部品の10%以上が基準を満たしていない場合に警告メッセージ (`"Caution"`) を返す変換プロパティを示しています。それ以外の場合は、情報メッセージ (`"Normal"`) を返す。

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

プロパティおよびその他のアセットからのデータを集計する (メトリクス)

メトリクスは、集計関数を使用してすべての入力データポイントを処理し、指定された時間間隔ごとに 1 つのデータポイントを出力する数式です。たとえば、メトリクスは、温度データストリームから時間当たりの平均温度を計算できます。

メトリクスは、関連付けられたアセットのメトリクスからデータを入力できるため、オペレーションまたはオペレーションのサブセットに関するインサイトを提供する統計情報を計算できます。たとえば、メトリクスは、風力発電所のすべての風力タービンの平均時間温度を計算できます。アセット間の関連付けを定義する方法の詳細については、「アセットモデルの階層を定義する」を参照してください。

メトリクスは、各時間間隔でのデータを集約することなく、他のプロパティからデータを入力することも可能です。コンピューティング式に[attribute] (属性) を指定した場合、 AWS IoT SiteWise はその属性の[latest] (最新) の値をコンピューティング式に使用します。式でメトリクスを指定すると、 は式を計算する時間間隔の最後の値 AWS IoT SiteWise を使用します。つまり、 `OEE = Availability * Quality * Performance`、 `Availability`、 `Quality` はすべて同じアセットモデル上の他のメトリクスであり、`Performance` のようなメトリクスを定義することができるのです。

AWS IoT SiteWise は、すべてのアセットプロパティの基本的な集計メトリクスのセットも自動的に計算します。計算コストを削減するために、基本的な計算にカスタムメトリクスを定義する代わりに、これらの集計を使用できます。詳細については、「でアセットプロパティの集計をクエリする AWS IoT SiteWise」を参照してください。

トピック

- メトリクスを定義する (コンソール)
- メトリクスを定義する (AWS CLI)

メトリクスを定義する (コンソール)

AWS IoT SiteWise コンソールでアセットモデルのメトリクスを定義するときは、次のパラメータを指定します。

- [Name] (名前) - プロパティの名前。
- [Data type] (データ型) - 変換のデータ型で、[Double] (倍) または [String] (文字列) です。
- 外部 ID – (オプション) これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。
- [Formula] (計算式) - メトリクス式。メトリクス式では、[aggregation functions] (集約関数) を使用して、階層内のすべての関連するアセットに対するプロパティからデータを入力することができます。入力を開始するか、下矢印キーを押すと、オートコンプリート機能を実行できます。詳細については、「数式を使用する」を参照してください。

> **Important**
>
> メトリクスは、整数、倍、ブール値、文字列型のプロパティのみ使用可能です。ブール値は、0 (false) 、1 (true) に変換されます。
> メトリクスの式でメトリクス入力変数を定義する場合、それらの入力には出力メトリクスと同じ時間間隔が必要です。
> 数式では、倍値または文字列値のみを出力できます。入れ子になった式は、文字列など他のデータ型を出力することができますが、式全体としては、数値または文字列として評価されなければなりません。[jp function] (jp関数) を使うと、文字列を数値に変換することができます。ブール値は 1 (true) または 0 (false) でなければならない。詳細については、「未定義の値、無限の値、およびオーバーフロー値」を参照してください。

- [Time interval] (時間間隔) - メトリクスの時間間隔。AWS IoT SiteWise は次のタンブリングウィンドウ時間間隔をサポートし、各間隔は前の間隔が終了したときに開始される。
  - [1 minute] (1 分) – 1 分。毎分の終わりに処理されます (午前 00:00:00、午前 00:01:00、午前 00:02:00 など)。
  - [5 minutes] (5 分) – 5 分。正時から 5 分ごとに処理されます (午前 00:00:00、午前 00:05:00、午前 00:10:00 など)。
  - [15 minutes] (15 分) – 15 分。正時から 15 分ごとに処理されます (午前 00:00:00、午前 00:15:00、午前 00:30:00 など)。
  - [1 hour] (1 時間) – 1 時間 (60 分)。UTC の毎時の終わりに処理されます (午前 00:00:00、午前 01:00:00、午前 02:00:00 など)。
  - [1 day] (1 日) – 1 日 (24 時間)。UTC の毎日の終わりに処理されます (月曜日の午前 00:00:00、火曜日の午前 00:00:00 など)。
  - [1 week] (1 週間) – 1 週間 (7 日間)。UTC の毎週日曜日の終わりに処理されます (月曜日の午前 00:00:00 ごと)。
  - [Custom interval] (カスタム時間間隔) - 1 分 ~ 1 週間の間で任意の時間間隔を入力できます。
- [Offset date] (オフセット日) -(オプション) データを集計するためのリファレンス日です。
- [Offset time] (オフセット時間) -(オプション) データを集計するリファレンス時間。オフセット時刻は、00:00:00 ~ 23:59:59 の間でなければなりません。
- [Offset time zone] (オフセットタイムゾーン) -(オプション) オフセットのタイムゾーンです。指定しない場合、デフォルトのオフセットタイムゾーンは協定世界時 (UTC) です。

サポートされているタイムゾーン

- (UTC+00:00) 世界協定時間
- (UTC+01:00) 中央ヨーロッパ時間
- (UTC+02:00) 東ヨーロッパ
- (UTC03+:00) 東アフリカ時間
- (UTC+04:00) 近東時間
- (UTC+05:00) パキスタンラホール時間
- (UTC+05:30) インド標準時
- (UTC+06:00) バングラデシュ標準時
- (UTC+07:00) ベトナム標準時
- (UTC+08:00) 中国 [Taiwan] (台湾) (台北市) 時間
- (UTC+09:00) 日本標準時
- (UTC+09:30) オーストラリア中部時間
- (UTC+10:00) オーストラリア東部時間
- (UTC+11:00) ソロモン標準時
- (UTC+12:00) ニュージーランド標準時
- (UTC-11:00) ミッドウェー諸島時間
- (UTC-10:00) ハワイ標準時
- (UTC-09:00) アラスカ標準時
- (UTC-08:00) 太平洋標準時
- (UTC-07:00) フェニックス標準時
- (UTC-06:00) 中央標準時
- (UTC-05:00) 東部標準時
- (UTC-04:00) プエルトリコ米領バージン諸島時間
- (UTC-03:00) アルゼンチン標準時
- (UTC-02:00) サウスジョージア時間
- (UTC-01:00) 中央アフリカ時間

Example オフセット付きのカスタム時間間隔 (コンソール)

次の例では、2021 年 2 月 20 日午後 6 時 30 分 30 秒 (PST) にオフセットされた 12 時間の時間間隔を定義する方法を示します。

オフセット付きのカスタム時間間隔 を定義するには

1. [Time interval] (時間間隔) で [Custom interval] (カスタム間隔) を選択します。
2. [Time interval] (時間間隔) については、次のいずれかを行ってください。
   - **12** を入力し、[hours] (時間) を選択します。
   - **720** を入力し、[minutes] (分) を選択します。
   - **43200** を入力し、[seconds] (秒) を選択します。

   > **Important**
   > [Time interval] (時間間隔) は単位に関係なく整数でなければならない。

3. [Offset date] (オフセット日付) は、[2021/02/20]を選択します。
4. [Offset time] (オフセット時間) には、**18:30:30** を入力します。
5. [Offset timezone] (オフセットタイムゾーン) は、[(UTC-08:00) Pacific Standard Time] ((UTC-08:00) 太平洋標準時) を選択してください。

2021 年 7 月 1 日午後 6:30:30 (PST) 以前または 6:30:30 にメトリクスを作成した場合、2021 年 7 月 1 日午後 6:30:30 (PST) に最初の集計結果が表示されます。2 番目の集計結果は、2021 年 7 月 2 日午前 6:30:30 (PST) であり、以下同様です。

メトリクスを定義する (AWS CLI)

AWS IoT SiteWise API を使用してアセットモデルのメトリクスを定義する場合は、次のパラメータを指定します。

- `name` - プロパティの名前。
- `dataType` - メトリクスのデータ型 (`DOUBLE` または `STRING`)。
- `externalId` – (オプション) これはユーザー定義 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

- `expression` - メトリクス式。メトリクス式では、[aggregation functions] (集約関数) を使用して、階層内のすべての関連するアセットに対するプロパティからデータを入力することができます。詳細については、「数式を使用する」を参照してください。
- `window` - メメトリクスタンブリングウィンドウの時間間隔とオフセットで、各間隔は前の間隔が終了したときに開始されます。
  - `interval` - タンブリングウィンドウの時間間隔です。時間間隔は 1 分以上 1 週間以内にする必要があります。
  - `offsets` - タンブリングウィンドウのオフセット。

  詳細については、[AWS IoT SiteWise API Reference] (API リファレンス) の TumblingWindow を参照してください。

  Example オフセット付きカスタム時間間隔 (AWS CLI)

  次の例では、2021 年 2 月 20 日午後 6 時 30 分 30 秒 (PST) にオフセットされた 12 時間の時間間隔を定義する方法を示します。

```
{
    "window": {
        "tumbling": {
            "interval": "12h",
            "offset": " 2021-07-23T18:30:30-08"
        }
    }
}
```

  2021 年 7 月 1 日午後 6:30:30 (PST) 以前または 6:30:30 にメトリクスを作成した場合、2021 年 7 月 1 日午後 6:30:30 (PST) に最初の集計結果が表示されます。2 番目の集計結果は、2021 年 7 月 2 日午前 6:30:30 (PST) であり、以下同様です。
- `variables` - 式で使用するアセットまたは子アセットのその他のプロパティを定義する変数のリスト。各変数構造には、式で使用する単純な名前と、その変数にリンクされるプロパティを定義する `value` 構造が含まれます。`value` 構造体には、以下の情報が含まれています。
  - `propertyId` - 値を取得するプロパティの ID。プロパティが (階層からモデルで定義されているのではなく) 現在のモデルで定義されている場合は、ID の代わりにプロパティの名前を使用できます。

- `hierarchyId` -(オプション) プロパティの子アセットをクエリする階層の ID。ID の代わりに階層定義の名前を使用できます。この値を省略すると、 は現在のモデルで プロパティ AWS IoT SiteWise を見つけます。

> **Important**
>
> メトリクスは、整数、倍、ブール値、文字列型のプロパティのみ使用可能です。ブール値は、`0` (false) 、1 (true) に変換されます。
> メトリクスの式でメトリクス入力変数を定義する場合、それらの入力には出力メトリクスと同じ時間間隔が必要です。
> 数式では、倍値または文字列値のみを出力できます。入れ子になった式は、文字列など他のデータ型を出力することができますが、式全体としては、数値または文字列として評価されなければなりません。[jp function] (jp関数) を使うと、文字列を数値に変換することができます。ブール値は 1 (true) または 0 (false) でなければならない。詳細については、「未定義の値、無限の値、およびオーバーフロー値」を参照してください。

- `unit` –(オプション) プロパティの科学単位 (mm、摂氏など)。

Example メトリクス定義の例

次の例は、アセットの温度測定データを集計して、時間当たりの最高温度を華氏で計算するメトリクスプロパティを示しています。このオブジェクトは、[Metric] (メトリクス) を含む AssetModelProperty の例です。このオブジェクトは、CreateAssetModel リクエストペイロードの一部として指定して、メトリクスプロパティを作成できます。詳細については、「アセットモデルを作成する (AWS CLI)」を参照してください。

```
{
      ...
      "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 関連アセットからデータを入力するメトリクス定義例。

次の例は、複数の風力タービンの平均電力データを集約し、風力発電所の総平均電力を算出するメトリックプロパティを示しています。このオブジェクトは、[Metric] (メトリクス) を含む AssetModelProperty の例です。このオブジェクトは、CreateAssetModel リクエストペイロードの一部として指定して、メトリクスプロパティを作成できます。

```
{
      ...
      "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"
      }
    ],
    ...
}
```

数式を使用する

数式を使用すると、raw 産業データを変換および集計する数学関数を定義して、オペレーションに関する洞察を得ることができます。数式は、リテラル、演算子、関数、および変数を組み合わせてデータを処理します。数式を使用したアセットプロパティの定義方法については、データを変換する (変換) および プロパティおよびその他のアセットからのデータを集計する (メトリクス) を参照してください。変換とメトリックは、式のプロパティです。

トピック

- 数式で変数を使用する
- 数式でリテラルを使用する
- 数式で演算子を使用する
- 数式で定数を使用する
- 数式で関数を使用する
- 数式表現チュートリアル

数式で変数を使用する

変数は数式内の AWS IoT SiteWise アセットプロパティを表します。変数を使用して、式中の他のアセットプロパティから値を入力し、定数プロパティ ([attributes]) (属性) 、raw データストリーム ([measurements]) (測定値) 、および他の式プロパティからのデータを処理できるようにします。

変数は、同じアセットモデル、または関連する子アセットモデルのアセットプロパティを表すことができます。子アセットモデルから変数を入力できるのは、メトリクス式のみです。

コンソールと API では、異なる名前で変数を識別します。

- AWS IoT SiteWise コンソール – アセットプロパティ名を式内の変数として使用します。
- AWS IoT SiteWise API (AWS CLI ， AWS SDKs) – ExpressionVariable 構造を使用して変数を定義します。これには変数名とアセットプロパティへの参照が必要です。変数名には小文字、数字、アンダースコアを使用することができます。次に、式の中でアセットプロパティをリファレンスするために、変数名を使用します。

変数名名では、大文字と小文字が区別されます。

詳細については、[Defining metrics] (メトリクスの定義) の [Defining transforms] (変換の定義) を参照してください。

変数を使用してプロパティを参照する

変数の値は、参照するプロパティを定義します。 AWS IoT SiteWise では、さまざまな方法でこれを行うことができます。

- プロパティ ID によって: プロパティの一意の ID (UUID) を指定して識別できます。
- 名前によって: プロパティが同じアセットモデルにある場合、プロパティ ID フィールドに名前を指定できます。
- パスによって: 変数値は、パスによってプロパティを参照できます。詳細については、「パスを使用してカスタム複合モデルのプロパティを参照する」を参照してください。

> **Note**
>
> 変数は AWS IoT SiteWise コンソールではサポートされていません。これらは、 AWS Command Line Interface AWS CLI) や AWS SDKs を含む AWS IoT SiteWise API によって使用されます。

からのレスポンスで受け取る変数 AWS IoT SiteWise には、ID とパスの両方を含む、値に関する完全な情報が含まれます。

ただし、変数を に渡す場合 AWS IoT SiteWise （ たとえば、「作成」または「更新」呼び出しで ）、これらのいずれかを指定するだけで済みます。例えば、パスを指定した場合、ID を指定する必要はありません。

数式でリテラルを使用する

AWS IoT SiteWise では、式と式でのリテラルの使用がサポートされています。リテラルは、特定のデータ型を表す固定値です。では AWS IoT SiteWise、数式で数値と文字列リテラルを定義できます。リテラルは、データ変換、アラーム条件、視覚化計算など、さまざまなコンテキストで使用できます。

- 数字

  数字と科学的記数法を用いて、整数と 2 倍を定義することができる。[E notation] (指数表記) は指数表記で数字を表現することができます。

  例:`1`、 `2.0`、 `.9`、 `-23.1`、 `7.89e3`、 `3.4E-5`

- 文字列

  文字列を定義するには、' (引用符) と" (二重引用符) を使用します。開始と終了の引用符の種類は一致させる必要があります。文字列の宣言に使用した引用符と一致する引用符をエスケープするには、その引用符文字を 2 回使用します。これは AWS IoT SiteWise 文字列内の唯一のエスケープ文字です。

  例: `'active'`、 `"inactive"`、 `'{"temp": 52}'`、 `"{""temp"": ""high""}"`

数式で演算子を使用する

式には、次の一般的な算術演算子を使用することができます。

演算子	説明
+	両オペランドが数値の場合、この算術演算子は左右のオペランドを加算します。 どちらかのオペランドが文字列の場合、この算術演算子は左右のオペランドを文字列として連結する。例えば、式 `1 + 2 + " is three"` は `"3 is three"` に評価される。連結された文字列は最大 1024 文字まで可能である。文字列が 1024 文字を超える場合、 AWS IoT

演算子	説明
	SiteWise はそのコンピューティングのためのデータポイントを出力しない。
-	左のオペランドから右のオペランドを減算します。 この算術演算子は、数値オペランドにのみ使用できます。
/	左のオペランドを右のオペランドで除算します。 この算術演算子は、数値オペランドにのみ使用できます。
*	左右のオペランドを乗算します。 この算術演算子は、数値オペランドにのみ使用できます。
^	左のオペランドを右のオペランドでべき乗します (べき乗)。 この算術演算子は、数値オペランドにのみ使用できます。
%	左のオペランドを右のオペランドで除算した剰余を返します。結果の符号は、左オペランドと同じです。この動作は、モジュロ演算とは異なります。 この算術演算子は、数値オペランドにのみ使用できます。
x < y	1 が x より小さい場合は y を返し、それ以外の場合は 0 を返します。

演算子	説明
`x > y`	1 が x より大きい場合は y を返し、それ以外の場合は 0 を返します。
`x <= y`	1 が x より小さいか、または等しい場合は y を返し、それ以外の場合は 0 を返します。
`x >= y`	1 が xより大きいか、または等しい場合は y を返し、それ以外の場合は 0 を返します。
`x == y`	1 が x に等しい場合は y を返し、それ以外の場合は 0 を返します。
`x != y`	1 が x に等しくない場合は y を返し、それ以外の場合は 0 を返します。
`!x`	1 が x (false) と評価された場合は 0 を返し、それ以外の場合は 0 を返します。 x は、次の場合に (false) と評価される。 • x は数値オペランドであり、0 に評価される。 • x は空白の文字列に評価される。 • x は空白の配列に評価される。 • x が None に評価される。

演算子	説明
`x and y`	`0` が xと評価された場合 (false) 、`0` を返す。それ以外の場合は、y の評価結果を返す。 x または y は、次の場合に (false) と評価される。 • x または y は数値オペランドであり、`0` に評価される。 • x または y は空白の文字列として評価される。 • x または y が空白の配列に評価されます。 • x または y が None に評価される。
`x or y`	1 が x (true) に評価された場合、1 を返す。それ以外の場合は、y の評価結果を返す。 x または y は、次の場合に (false) と評価される。 • x または y は数値オペランドであり、`0` に評価される。 • x または y は空白の文字列として評価される。 • x または y が空白の配列に評価されます。 • x または y が None に評価される。

演算子	説明
`not x`	1 が x (false) と評価された場合は `0` を返し、それ以外の場合は `0` を返します。 x は、次の場合に (false) と評価される。 • x は数値オペランドであり、`0` に評価される。 • x は空白の文字列に評価される。 • x は空白の配列に評価される。 • x が None に評価される。

`[]`
 	
`s[index]`
	文字列 `index` のインデックス `s` にある文字を返す。これは Python のインデックス構文に相当する。
	
	Example 例
	
	• `"Hello!"[1]` は `e` を返します。
	• `"Hello!"[-2]` は `o` を返します。

演算子	説明

`[]`
 	
`s[start:end:step]`
	文字列 s のスライスを返す。これは Python のスライス構文に相当する。この演算子は次の引数を持つ。
	
	• `start` -(オプション) スライスの包括的な開始インデックスを指定します。デフォルトは `0` です。
	• `end` -(オプション) スライスの排他的終了インデックスです。デフォルトは文字列の長さです。
	• `step` -(オプション) スライスの各ステップで増分する番号です。例えば、2 を指定すると1文字おきにスライスを返し、`-1` を指定するとスライスを反転させることができる。デフォルトは 1 です。
	
	`step` 引数を省略すると、そのデフォルト値が使用される。たとえば、`s[1:4:1]` と `s[1:4]` は同じです。
	
	引数は整数または[none] (無し) 定数でなければならない。を指定するとnone、 はその引数のデフォルト値 AWS IoT SiteWise を使用します。
	
	Example 例
	
	• `"Hello!"[1:4]` は `"ell"` を返します。
	• `"Hello!"[:2]` は `"He"` を返します。
	• `"Hello!"[3:]` は `"lo!"` を返します。
	• `"Hello!"[:-4]` は `"He"` を返します。
	• `"Hello!"[::2]` は `"Hlo"` を返します。
	• `"Hello!"[::-1]` は `"!olleH"` を返します。

数式で定数を使用する

では AWS IoT SiteWise、式と式で定数を使用して、固定値または事前定義されたパラメータを表すことができます。定数は、データ変換、アラーム条件、視覚化計算など、さまざまなコンテキストで使用できます。定数を使用すると、式を簡素化して、読みやすく、保守性を高めることができます。

式には、次の一般的な数学定数を使用できます。すべての定数は大文字と小文字を区別しません。

> **Note**
> 定数と同じ名前の変数を定義すると、その変数が定数をオーバーライドします。

定数	説明
`pi`	数字のπ (π) です: `3.141592653589793`
`e`	数字のe: `2.718281828459045`
`true`	数字の 1 に相当します。では AWS IoT SiteWise、ブール値は同等の数値に変換されます。
`false`	数字の 0 に相当します。では AWS IoT SiteWise、ブール値は同等の数値に変換されます。
`none`	無価値。この定数を使用すると、[conditional expression] (条件式) の結果として何も出力しないことができます。

数式で関数を使用する

次の関数を使用して、数式中のデータを操作することができます。

変換とメトリクスは、それぞれ異なる機能をサポートしています。次の表は、各型の数式プロパティに対応する関数の種類を示したものです。

> **Note**
>
> 1 つの数式には、最大 10 個の関数を含めることができます。

関数型	変換	メトリクス
数式で一般的な関数を使用する	はい	はい
数式で比較関数を使用する	はい	はい
数式で条件関数を使用する	はい	はい
数式で文字列関数を使用する	はい	はい
数式で集計関数を使用する	いいえ	はい
数式で時間関数を使用する	はい	はい

関数型	変換	メトリクス
数式で日付および時刻関数を使用する	はい	はい

関数構文

関数を作成するには、次の構文を使用します。

通常の構文

通常の構文では、関数名の後に括弧が付き、引数は 0 個以上となります。

function_name(*argument1, argument2, argument3, ...*)。例えば、通常の構文を持つ関数は、`log(x)` や `contains(s, substring)` のようになります。

統一関数呼び出し構文 (UFCS)

UFCSでは、オブジェクト指向プログラミングにおけるメソッド呼び出しの構文を用いて、関数を呼び出すことができる。UFCSでは、最初の引数の後にドット (.) が付き、次に関数名、残りの引数がある場合は括弧の中に入る。

argument1.*function_name*(*argument2, argument3, ...*)。例えば、UFCSを使った関数は、`x.log()` や `s.contains(substring)` のようになります。

UFCS を使用して後続の関数を連鎖することもできます。 AWS IoT SiteWise は、現在の関数の評価結果を次の関数の最初の引数として使用します。

例えば、`message.jp('$.status').lower().contains('fail')` の代わりに `contains(lower(jp(message, '$.status')),'fail')` を使用することができます。

詳しくは、[D Programming Language] (D プログラミング言語) のウェブサイトをご覧ください。

> **Note**
>
> UFCS はすべての AWS IoT SiteWise 関数に使用できます。

AWS IoT SiteWise 関数では、大文字と小文字は区別されません。例えば、`lower(s)` と `Lower(s)` を入れ替えて使用することができます。

数式で一般的な関数を使用する

[transforms] (変換) と [metrics] (メトリクス) において、一般的な数学的関数をコンピューティングすることができます。

関数	説明
`abs(x)`	x の絶対値を返します。
`acos(x)`	x のアークコサイン (逆余弦) を返します。
`asin(x)`	x のアークサイン (逆正弦) を返します。
`atan(x)`	x のアークタンジェント (逆正接) を返します。
`cbrt(x)`	x の立方根を返します。
`ceil(x)`	x より大きい最も近い整数を返します。
`cos(x)`	x のコサイン (余弦) を返します。
`cosh(x)`	x のハイパーボリックコサイン (双曲線余弦) を返します。
`cot(x)`	x のコタンジェントを返します。
`exp(x)`	e の x 乗を返します。
`expm1(x)`	戻り値`exp(x) - 1`。`exp(x) - 1` が小さい値である場合に、より正確に x をコンピューティングするためにこの関数を使用します。
`floor(x)`	x より小さい最も近い整数を返します。
`log(x)`	$\log_e$ の e (底 x) を返します。
`log10(x)`	$\log_{10}$ の 10 (底 x) を返します。

関数	説明
`log1p(x)`	戻り値`log(1 + x)`。`log(1 + x)` が小さい値である場合に、より正確に x をコンピューティングするためにこの関数を使用します。
`log2(x)`	$\log_2$ の 2 (底 x) を返します。
`pow(x, y)`	x の y 乗を返します。`x ^ y` に相当します。
`signum(x)`	x (負の入力の場合は -1、ゼロの入力の場合は 0、正の入力の場合は +1) 符号を返します。
`sin(x)`	x のサイン (正弦) を返します。
`sinh(x)`	x のハイパーボリックサイン (双曲線正弦) を返します。
`sqrt(x)`	x の平方根を返します。
`tan(x)`	x のタンジェント (正接) を返します。
`tanh(x)`	x のハイパーボリックタンジェント (双曲線正接) を返します。

数式で比較関数を使用する

変換とメトリクスでは、次の比較関数を使用して、2 つの値と出力 1 (true) または 0 (false) を比較できます。 は、辞書順で文字列 AWS IoT SiteWise を比較します。

関数	説明
`gt(x, y)`	1 が x より大きい場合は y を返し、それ以外の場合は 0 (`x > y`) を返します。 この関数は、x と y が数値と文字列のように互換性のない型の場合は、値を返さない。

関数	説明
`gte(x, y)`	1 が x より大きいか、または等しい場合は y を返し、それ以外の場合は `0` (x ≥ y) を返します。 AWS IoT SiteWise は、引数が の相対許容値内にある場合、引数を等しいと見なします`1E-9`。これは Python のisclose 関数と同じような動作をします。 この関数は、x と y が数値と文字列のように互換性のない型の場合は、値を返さない。
`eq(x, y)`	1 が x に等しい場合は y を返し、それ以外の場合は `0` (x == y) を返します。 AWS IoT SiteWise は、引数が の相対許容値内にある場合、引数を等しいと見なします`1E-9`。これは Python のisclose 関数と同じような動作をします。 この関数は、x と y が数値と文字列のように互換性のない型の場合は、値を返さない。
`lt(x, y)`	1 が x より小さい場合は y を返し、それ以外の場合は `0` (x < y) を返します。 この関数は、x と y が数値と文字列のように互換性のない型の場合は、値を返さない。

関数	説明
`lte(x, y)`	1 が x より小さいか、または等しい場合は y を返し、それ以外の場合は 0 (x ≤ y) を返します。 AWS IoT SiteWise は、引数が の相対許容値内にある場合、引数を等しいと見なします1E-9。これは Python のisclose 関数と同じような動作をします。 この関数は、x と y が数値と文字列のように互換性のない型の場合は、値を返さない。
`isnan(x)`	1 が x に等しい場合は NaN を返し、それ以外の場合は 0 を返します。 x が文字列の場合、この関数は値を返さない。

数式で条件関数を使用する

変換とメトリクスでは、次の関数を使用して条件をチェックし、その条件が true と評価されるか false と評価されるかで異なる結果を返すことができます。

関数	説明
`if(condition, result_if_true, result_if_false)`	`condition` を評価し、条件の評価結果が true の場合に `result_if_true` 、`false` の場合には `result_if_false` を返します。 `condition` は数字でなければなりません。この関数は、0 および空の文字列を `false` と見なし、それ以外 (NaN を含む) の場合を `true` と見なします。ブール値は、0 (false) 、1 (true) に変換されます。 この関数から[none] (無し) 定数を返すことで、特定の条件の出力を破棄することができます。

関数	説明
	つまり、条件を満たさないデータポイントをフィルタリングすることができるのです。詳細については、「データポイントをフィルタリングする」を参照してください。 Example 例 • `if(0, x, y)` は変数 y を返す。 • `if(5, x, y)` は変数 x を返す。 • `if(gt(temp, 300), x, y)` は、変数 x が temp よりも大きい場合、変数 300 を返す。 • `if(gt(temp, 300), temp, none)` が temp 以上なら変数 300 を、none が temp より小さいなら (300値なし) を返す。 1つ以上の引数が条件付き関数である入れ子関数には、UFCS を使用することをお勧めします。`if(condition, result_if_true)` で条件を評価し、`elif(condition, result_if_true, result_if_false)` で追加の条件を評価することができます。 例えば、`if(condition1, result1_if_true).elif(condition2, result2_if_true, result2_if_false)` の代わりに `if(condition1, result1_if_true, if(condition2, result2_if_true, result2_if_false))` を使用することができます。 また、追加の中間条件関数を連結することもできます。たとえば、`if(condition1, result1_if_true, if(condit`

関数	説明
	`ion2, result2_if_true, if(condition3, result3_if_true result3_if_false)))` のように複数の if ステートメントをネストするのではなく `if(condition1, result1_if_true).elif(condition2, result2_if_true).elif(condition3, result3_if_true, result3_if_false)` を使用することができます。 **Important** UFCS で `elif(condition, result_if_true, result_if_false)` を使用する必要があります。

数式で文字列関数を使用する

[transforms] (変換) と[metrics] (メトリクス) では、次の関数を使用して文字列を操作することができます。詳細については、「式で文字列を使用する」を参照してください。

> **Important**
> 数式では、倍値または文字列値のみを出力できます。入れ子になった式は、文字列など他のデータ型を出力することができますが、式全体としては、数値または文字列として評価されなければなりません。[jp function] (jp関数) を使うと、文字列を数値に変換することができます。ブール値は 1 (true) または 0 (false) でなければならない。詳細については、「未定義の値、無限の値、およびオーバーフロー値」を参照してください。

関数	説明
`len(s)`	文字列 s の長さをを返します。

関数	説明
`find(s, substring)`	文字列 `substring` が文字列 `s` に含まれるインデックスを返す。
`contains(s, substring)`	文字列 `1` が文字列 `s` を含んでいれば `substring` 、それ以外の場合は `0` を返します。
`upper(s)`	文字列 `s` を大文字で返す。
`lower(s)`	文字列 `s` を小文字に変換して返す。

関数	説明
`jp(s, json_path)`	文字列 s をJsonPath 式 `json_path` で評価し、その結果を返す。 この関数を使うと、次のことができます。 • シリアル化された JSON 構造体から値、配列、またはオブジェクトを取り出す。 • 文字列を数値に変換する。例えば、`jp('111', '$')` という数式は、111 を数値として返す。 JSON 構造体から文字列値を取り出し、数値として返すには、複数の入れ子 `jp` 関数を使用する必要があります。外側の `jp` 関数は JSON 構造体から文字列を抽出し、内側の `jp` 関数はその文字列を数値に変換する。 文字列 `json_path` には、文字列リテラルを含める必要があります。つまり、`json_path` は文字列として評価される式ではありえないということです。 Example 例 • `jp('{"status":"active","value":15}', '$.value')` は 15 を返します。 • `jp('{"measurement":{"reading":25,"confidence":0.95}}', '$.measurement.reading')` は 25 を返します。 • `jp('[2,8,23]', '$[2]')` は 23 を返します。 • `jp('{"values":[3,6,7]}', '$.values[1]')` は 6 を返します。

関数	説明
	• `jp('111', '$')` は 111 を返します。 • `jp(jp('{"measurement":{"reading":25,"confidence":"0.95"}}', '$.measurement.confidence'), '$')` は 0.95 を返します。
`join(s0, s1, s2, s3, ...)`	文字列をデリミタ付きで連結して返す。この関数は、最初の入力文字列をデリミタとして使い、残りの入力文字列を結合する。これは、Javaの join(CharSequence delimiter, CharSequence... elements) 関数と同じような動作をします。 Example 例 • `join("-", "aa", "bb", "cc")` は `aa-bb-cc` を返す
`format(expression: "format")` 、または `format("format", expression)`	指定された書式の文字列を返す。この関数は、`expression` を評価し、指定されたフォーマットで値を返す。これは、Javaのformat (String format, Object... args)関数と同じような動作をします。サポートされるフォーマットの詳細については、Java Platform, Standard Edition 7 API SpecificationのClass Formatterの Conversions を参照してください。 Example 例 • `format(100+1: "d")` は文字列 101 を返す。 • `format("The result is %d", 100+1)` は文字列 `The result is 101` を返す。

関数	説明
`f'expression'`	連結された文字列を返す。この書式付き関数を使うと、簡単な式で文字列の連結や書式設定を行うことができます。これらの関数は、入れ子になった式を含むことができる。{} (中括弧) を使って、式を補間することができます。これは、Python の[formatted string literals] (フォーマットされた文字列リテラル) と同じような動作をします。 Example 例 • `f'abc{1+2: "f"}d'` は `abc3.000000d` を返します。この例の式を評価するには、次のようにします。 1. `format(1+2: "f")` は浮動小数点数 `3.000000` を返します。 2. `join('', "abc", 1+2, 'd')` は文字列 `abc3.000000d` を返す。 また、次のように式を書くこともできます `join('', "abc", format(1+2: "f"), 'd')`。

数式で集計関数を使用する

[metrics] (メトリクス) のみでは、各タイムインターバルの入力値を集計し、単一の出力値を算出する次の関数を使用することができます。集計関数は、関連付けられたアセットからデータを集計できません。

集計関数の引数には、変数、数値リテラル、時間関数 、入れ子式、または集計関数を使用することができます。式 `max(latest(x), latest(y), latest(z))` は集計関数を引数として使用し、x、y および z プロパティの現在の最大値を返します。

集計関数に入れ子になった式を使用することができます。入れ子になった式を使用する場合、次のルールが適用されます。

- 各引数は1つの変数しか持つことができません。

  Example

  例えば、`avg(x*(x-1))` と `sum(x/2 )/avg(y^2 )` がサポートされています。

  例えば、`min(x/y)` はサポートされていません。
- 各引数は多段階に入れ子になった式を持つことができる。

  Example

  例えば、`sum(avg(x^2 )/2)` がサポートされています。
- 引数によって、異なる変数を持つことができます。

  Example

  例えば、`sum(x/2, y*2)` がサポートされています。

> **Note**
>
> - 式に測定値が含まれている場合、 は測定値の現在の時間間隔の最後の値 AWS IoT SiteWise を使用して集計を計算します。
> - 式に属性が含まれている場合、 は属性の最新の値 AWS IoT SiteWise を使用して集計を計算します。

関数	説明
avg(x_0, ..., x_n)	現在の時間間隔における指定された変数値の平均を返します。 この関数は、与えられた変数が現在の時間間隔に少なくとも 1 つのデータポイントを持つ場合にのみデータポイントを出力します。
sum(x_0, ..., x_n)	現在の時間間隔における指定された変数値の合計を返します。

関数	説明
	この関数は、与えられた変数が現在の時間間隔に少なくとも 1 つのデータポイントを持つ場合にのみデータポイントを出力します。
min(x_0, ..., x_n)	現在の時間間隔における指定された変数の最小値を返します。 この関数は、与えられた変数が現在の時間間隔に少なくとも 1 つのデータポイントを持つ場合にのみデータポイントを出力します。
max(x_0, ..., x_n)	現在の時間間隔における指定された変数の最大値を返します。 この関数は、与えられた変数が現在の時間間隔に少なくとも 1 つのデータポイントを持つ場合にのみデータポイントを出力します。
count(x_0, ..., x_n)	現在の時間間隔における指定された変数のデータポイントの合計数を返します。条件を満たすデータポイントの数を数える方法の詳細については、「条件に一致するデータポイントをカウントする」を参照してください。 この関数は、時間間隔ごとにデータポイントを計算します。
stdev(x_0, ..., x_n)	現在の時間間隔に対する指定された変の数値の標準偏差を返します。 この関数は、与えられた変数が現在の時間間隔に少なくとも 1 つのデータポイントを持つ場合にのみデータポイントを出力します。

数式で時間関数を使用する

時間関数を使用して、データポイントのタイムスタンプに基づいた値を返すことができます。

メトリクスで時間関数を使用する。

[metrics] (メトリクス) のみ、データポイントのタイムスタンプに基づいた値を返す次の関数を使用できます。

時間関数の引数は、ローカルのアセットモデルまたは入れ子式のプロパティである必要があります。これは、子アセットモデルのプロパティを一時関数で使用できないことを意味します。

一時関数で入れ子になった式を使用することができます。入れ子になった式を使用する場合、次のルールが適用されます。

- 各引数は1つの変数しか持つことができません。

  例えば、`latest( t*9/5 + 32 )` がサポートされています。
- 引数には集計関数を指定できません。

  例えば、`first( sum(x) )` はサポートされていません。

関数	説明
`first(x)`	現在の時間間隔における最も早いタイムスタンプを持つ、指定された変数値を返します。
`last(x)`	現在の時間間隔における最新のタイムスタンプを持つ、指定された変数値を返します。
`earliest(x)`	現在の時間間隔が開始する前の指定された変数の最新の値を返します。 入力プロパティの履歴に少なくとも 1 つのデータポイントがある場合、この関数は、時間間隔ごとにデータポイントを計算します。詳細については、「time-range-defintion」を参照してください。
`latest(x)`	現在の時間間隔が終了する前の最新のタイムスタンプを持つ、指定された変数の最新の値を返します。

関数	説明
	入力プロパティの履歴に少なくとも 1 つのデータポイントがある場合、この関数は、時間間隔ごとにデータポイントを計算します。詳細については、「time-range-defintion」を参照してください。
`statetime(x)`	指定された時間間隔において、指定された変数が正である時間 (秒単位) を返します。[comparison functions] (比較関数) を使用して、`statetime` 関数が消費する変換プロパティを作成することができます。 たとえば、`Idle` プロパティが `0` または `1` の場合、次の式 `IdleTime = statetime(Idle)` を使用して時間間隔ごとのアイドル時間を計算できます。詳細については、「statetime シナリオの例」を参照してください。 この関数は、入力変数としてメトリクスプロパティをサポートしていません。 入力プロパティの履歴に少なくとも 1 つのデータポイントがある場合、この関数は、時間間隔ごとにデータポイントを計算します。

関数	説明
`TimeWeightedAvg(x, [interpolation])`	ポイント間の時間間隔で重み付けされた入力データの平均を返します。 計算と間隔の詳細については、「時間加重関数のパラメータ」を参照してください。 オプションの引数 `interpolaton` は文字列定数である必要があります。 • `locf` – デフォルト値です。この計算では、データポイント間の間隔に Last Observed Carry Forward (LOCF) 計算アルゴリズムが使用されます。このアプローチでは、データポイントは次の入力データポイントのタイムスタンプまで最後の観測値として使用されます。 良好データポイントの後の値が、次のデータポイントのタイムスタンプまでの値として外挿されます。 • `linear` – この計算では、データポイント間の間隔に線形補間計算アルゴリズムが使用されます。 2 つの良好データポイント間の値が、それらデータポイントの値間の線形補間値として外挿されます。 良好データポイントと不良データポイントの間の値、または最後の良好データポイントの後の値が、良好データポイントとして外挿されます。

関数	説明
`TimeWeightedStDev(x, [algo])`	ポイント間の時間間隔で重み付けされた入力データの標準偏差を返します。 計算と間隔の詳細については、「時間加重関数のパラメータ」を参照してください。 この計算では、データポイント間の間隔に Last Observed Carry Forward (LOCF) 計算アルゴリズムが使用されます。このアプローチでは、データポイントは次の入力データポイントのタイムスタンプまで最後の観測値として使用されます。重みは、データポイント間またはウィンドウ境界間の時間間隔 (秒単位) の形式で求められます。 オプションの引数 `algo` は文字列定数である必要があります。 • f – デフォルト値です。頻度の重み付きの偏りなし加重サンプル分散を返します。TimeWeight の単位は秒です。通常、このアルゴリズムは標準偏差未満であることを想定しており、加重サンプルに対する標準偏差のベッセル補正として知られています。 • p – 偏りあり加重サンプル分散 (母分散とも呼ばれる) を返します。 計算には以下の式が使用されます。 • S_p = 母集団の標準偏差 • S_f = 頻度の標準偏差 • X_i = 入力データ • ω_i = 時間間隔に等しい重み (秒) • μ^* = 入力データの加重平均

関数	説明
	母集団の標準偏差の計算式: $$S_p^2 = \frac{\sum_{i=1}^{N} \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^{N} \omega_i}$$ 頻度の標準偏差の計算式: $$S_f^2 = \frac{\sum_{i=1}^{N} \omega_i (x_i - \mu^*)^2}{\sum_{i=1}^{N} \omega_i - 1}$$

次の図は`first`、 が現在の時間間隔に対して時間関数 、`latest`、、`lastearliest`および AWS IoT SiteWise を計算する方法を示しています。

> **Note**
>
> - `first(x)` の時間範囲 `last(x)` は (現在のウィンドウの開始、現在のウィンドウの終了] です。
> - `latest(x)` の時間範囲は (時間の開始、現在のウィンドウの終了] です。
> - `earliest(x)` の時間範囲は (時間の開始、前のウィンドウの終了] です。

時間加重関数のパラメータ

集計ウィンドウ用に計算される時間加重関数では、次のことが考慮されます。

- ウィンドウ内のデータポイント数
- データポイント間の時間間隔数
- ウィンドウの前の最後のデータポイント
- ウィンドウの後の最初のデータポイント (一部のアルゴリズムのみ)

条件:

- 不良データポイント – 品質が良くないか、値が数値以外のデータポイント。不良データポイントは、ウィンドウの結果計算では考慮されません。
- 不良間隔 – 不良データポイントの後の間隔。最初の既知のデータポイントの前の間隔も不良間隔とみなされます。
- 良好データポイント – 品質が良好でかつ値が数値のデータポイント。

> **Note**
>
> - AWS IoT SiteWise は、変換とメトリクスを計算するときにのみ`GOOD`品質データを消費します。`UNCERTAIN` および `BAD` データポイントは無視します。
> - 最初の既知のデータポイントの前の間隔も不良間隔とみなされます。詳細については「the section called “数式表現チュートリアル”」を参照してください。

最後の既知のデータポイントの後の間隔は無限に続き、以降のすべてのウィンドウに影響します。新しいデータポイントがくると、この関数は間隔を再計算します。

上記のルールに従って、集計ウィンドウの結果が計算され、ウィンドウの境界が制限されます。デフォルトでは、この関数はウィンドウ全体が良好間隔である場合にのみウィンドウの結果を送信します。

ウィンドウの良好間隔がウィンドウの長さよりも短い場合、この関数はウィンドウを送信しません。

ウィンドウの結果に影響するデータポイントに変更があると、データポイントがウィンドウの外側にあっても、ウィンドウは再計算されます。

入力プロパティの履歴に少なくとも 1 つのデータポイントがあり、計算が開始されていた場合は、時間間隔ごとに時間加重集計関数が計算されます。

Example statetime シナリオの例

次のプロパティを持つアセットがある例を考えてみましょう。

- `Idle` - `0` または 1 である測定値をいう。値が 1 の場合、マシンはアイドル状態です。
- `Idle Time` - 1分間隔で、機械がアイドル状態である時間を秒単位でコンピューティングする式 `statetime(Idle)` を使用した指標。

`Idle` プロパティには、以下のデータポイントがあります。

タイムスタンプ	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 は、 の値から `Idle Time`プロパティを 1 分ごとに計算します`Idle`。この計算が完了すると、`Idle Time` プロパティには次のデータポイントが存在します。

タイムスタンプ	2:00:00 PM	2:01:00 PM	2:02:00 PM	2:03:00 PM	2:04:00 PM
Idle Time	該当なし	30	60	45	0

AWS IoT SiteWise は、毎分`Idle Time`の終わりに に対して次の計算を実行します。

- 2:00 PM 時点 (1:59 PM から 2:00 PM の場合)
  - `Idle` には 2:00 PM より前のデータがないため、データポイントは計算されません。
- 2:01 PM 時点 (2:00 PM から 2:01 PM の場合)
  - 2:00:00 PM の時点で、マシンはアクティブです (`Idle` は `0`)。
  - 2:00:30 PM の時点で、マシンはアイドル状態です (`Idle` は `1`)。
  - 2:01:00 PM にインターバルが終了する前に `Idle` は再度変更されないため、`Idle Time` は 30 秒になります。
- 2:02 PM 時点 (2:01 PM から 2:02 PM の場合)
  - 2:01:00 PM の時点で、マシンはアイドル状態です (2:00:30 PM 時点の直前のデータポイントによる)。
  - 2:01:15 PM の時点で、マシンはまだアイドル状態です。
  - 2:02:00 PM にインターバルが終了する前に `Idle` は再度変更されないため、`Idle Time` は 60 秒になります。
- 2:03 PM 時点 (2:02 PM から 2:03 PM の場合)
  - 2:02:00 PM の時点で、マシンはアイドル状態です (2:01:15 PM 時点の直前のデータポイントによる)。

- 2:02:45 PM の時点で、マシンはアクティブです。
- 2:03:00 PM にインターバルが終了する前に `Idle` は再度変更されないため、`Idle Time` は 45 秒になります。

- 2:04 PM 時点 (2:03 PM から 2:04 PM の場合)
  - 2:03:00 PM の時点で、マシンはアクティブです (2:02:45 PM 時点の直前のデータポイントによる)。
  - 2:04:00 PM にインターバルが終了する前に `Idle` は再度変更されないため、`Idle Time` は 0 秒になります。

Example TimeWeightedAvg シナリオと TimeWeightedStDev シナリオの例

次の表は、1 分間のウィンドウメトリクスの入力例と出力例を示しています: `Avg(x)`, `TimeWeightedAvg(x)`, `TimeWeightedAvg(x, "linear")`, `stDev(x)`, `timeWeightedStDev(x)`, `timeWeightedStDev(x, 'p')`。

1 分間の集計ウィンドウの入力例:

> **Note**
>
> これらのデータポイントはすべて、`GOOD` の品質を持ちます。

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

集計結果の出力:

> **Note**
> なし – このウィンドウでは結果は生成されません。

時間	Avg(x)	TimeWeightedAvg(x)	TimeWeightedAvg(X, "linear")	stDev(X)	timeWeightedStDev(x)	timeWeightedStDev(x, 'p')
3:00:00	4	なし	なし	0	なし	なし
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	8.5440037 4531753	7.724054437220943	7.659416862050705
3:04:00	0	10	2.5	0	10.084389681792215	10
3:05:00	なし	0	0	なし	0	0

変換で時間関数を使用する

[変換] のみ、`pretrigger()` 関数を使用して、現在の変換の計算を開始させたプロパティ更新前の変数の GOOD 品質値を取得することができます。

メーカーが AWS IoT SiteWise を使用してマシンのステータスをモニタリングする例を考えてみましょう。メーカーは、次のような測定値と変換を使用してプロセスを表現しています。

- 測定値 `current_state` は 0 または 1 となります。

  - 本機がクリーニング状態の場合、current_state は 1 になる。
  - 機械が製造状態にある場合、current_state は 0 になる。
- cleaning_state_durationに相当する変換、if(pretrigger(current_state) == 1, timestamp(current_state) - timestamp(pretrigger(current_state)), none)。この変換は、マシンがクリーニング状態である時間を秒単位で Unix エポックフォーマットで返します。詳しくは、数式で条件関数を使用する および timestamp () (タイムスタンプ) 関数を参照してください。

クリーニング状態が予想以上に長く続くようであれば、メーカーが調査する場合もあるります。

また、多変量変換で pretrigger( ) 関数を使うこともできます。例えば、x、y という 2 つの測定値があり、z と等しい変換値 x + y + pretrigger(y) があるとします。次の表は、午前 9 時から午前 9 時 15 分までの x、y、z の値である。

> **Note**
>
> - この例では、測定値の値が時系列に到着することを想定しています。例えば、午前 09 時 00 分のx の値は、午前 09 時 05 分の x の値より先に到着する。
> - 午前 9 時 5 分のデータポイントが午前 9 時のデータポイントより先に到着した場合、午前 9 時 5 分には z はコンピューティングされません。
> - 午前 9 時 5 分 の x の値が午前 9 時の x の値より先に到着し、y の値が時系列に到着する場合、z は午前 9 時 5 分の 22 = 20 + 1 + 1 と等しくなる。

	午前 9 時 0 分	午前 9 時 5 分	午前 9 時 10 分	午前 9 時 15 分
x	10	20		30
y	1	2	3	
z = x + y + pretrigger(y)	y は午前 09:00 以前にデータポイントを受信していません。したがって、z は午前 09:00 には	23 = 20 + 2 + 1 pretrigger(y) は 1 に等しい。	25 = 20 + 3 + 2 x は新しいデータポイントを受信しません。pretrigge	36 = 30 + 3 + 3 y は新しいデータポイントを受信しません。したがって、午前

	午前 9 時 0 分	午前 9 時 5 分	午前 9 時 10 分	午前 9 時 15 分
	コンピューティングされません。		`r(y)` は 2 に等しい。	09 時 15 分には `pretrigger(y)` は3に等しい。

数式で日付および時刻関数を使用する

[transforms] (変換) と[metrics] (メトリクス) では、日付と時刻の関数を次のように使用できます。

- データポイントの現在のタイムスタンプを UTC またはローカルタイムゾーンで取得する。
- `year`、`month`、`day_of_month` などの引数でタイムスタンプを構成する。
- `unix_time` 引数で年や月などの期間を抽出する。

関数	説明
`now()`	現在の日付と時刻を Unix エポックフォーマットで秒単位で返します。
`timestamp()`	• 変換では、この関数は入力メッセージのタイムスタンプをUnix工ポックフォーマットで秒単位で返します。 変換のみでは、次のいずれかを行うことができます。 • 関数の引数として変数を与える。`timestamp(` *`variable-name`* `)` 関数は、指定された変数の最新の GOOD 品質値のタイムスタンプを Unix の工ポックフォーマットで秒単位で返す。 例えば、アセットに `Temperature_F` という変換プロパティがあり、 `9/5 * Temperature_C` 式を使用して各温度データポイントを摂氏から華氏に変換する場合、`timestamp(Temperature_F)`

関数	説明
	関数を使用して、GOOD プロパティに対する最新の Temperature_F 品質値のタイムスタンプを取得することができます。 • pretrigger() 関数の引数として使用します。timestamp(pretrigger(*variable-name*)) 関数は、現在の変換の計算を開始させたプロパティ更新前の指定された変数の GOOD 品質値のタイムスタンプを Unix エポック形式で秒単位で返します。詳細については、「変換で時間関数を使用する」を参照してください。 • メトリクスでは、この関数は、現在のウィンドウの終了時に取得したタイムスタンプを Unix エポックフォーマットで秒単位で返します。

関数	説明
`mktime(time_zone, year, month, day_of_month, hour, minute, second)`	入力された時刻を Unix エポックフォーマットで秒単位で返します。 この機能を使用するには、次の条件があります。 • タイムゾーンの引数は、引用符で囲まれた文字列 (`'UTC'`) でなければならない。指定されなかった場合、デフォルトのタイムゾーンは UTC です。 タイムゾーン引数は、最初または最後の引数とすることができる。 • 年、月、日、時、分、秒の引数は順番に指定する必要があります。 • 年、月、日付の引数は必須です。 この機能を使用する場合、次の制限があります。 • `year` - 有効な値は 1970 ~ 2250 です。 • `month` - 有効な値は 1 ~ 12 です。 • `day-of-month` - 有効な値は 1 ~ 31 です。 • `hour` - 有効な値は 0 ~ 23 です。 • `minute` - 有効な値は 0 ~ 59 です。 • `second` - 有効な値は 0 ~ 60 です。浮動小数点数であってもよい。 例: • `mktime(2020, 2, 29)` • `mktime('UTC+3', 2021, 12, 31, 22)`

関数	説明
	• `mktime(2022, 10, 13, 2, 55, 13.68, 'PST')`

関数	説明
`localtime(unix_time, time_zone)`	Unix 時間から、指定されたタイムゾーンの年、月、曜日、日、時、分、秒を返します。 この機能を使用するには、次の条件があります。 • タイムゾーンの引数は、引用符で囲まれた文字列 (`'UTC'`) でなければならない。指定されなかった場合、デフォルトのタイムゾーンは UTC です。 • Unix time 引数は、Unix エポックフォーマットによる秒単位の時間である。有効な範囲は 1 ~ 31556889864403199 です。浮動小数点数であってもよい。 レスポンスの例:`2007-12-03T10:15:30+01:00[Europe/Paris]` `localtime(unix_time, time_zone)` は単体の機能ではありません。`year()`、`mon()`、`mday`、`wday()`、`yday()`、hou 関数は、`localtime(unix_time, time_zone)` を引数とする。 例: • `year(localtime('GMT', 1605898608.8113723))` • `now().localtime().year()` • `timestamp().localtime('PST').year()` • `localtime(1605289736, 'Europe/London').year()`

関数	説明
year(localtime(unix_time, time_zone)	localtime(unix_time, time_zone) から年を返します。
mon(localtime(unix_time, time_zone))	localtime(unix_time, time_zone) から月を返します。
mday(localtime(unix_time, time_zone))	localtime(unix_time, time_zone) から月の特定の日を返します。
wday(localtime(unix_time, time_zone))	localtime(unix_time, time_zone) から週の特定の日を返します。
yday(localtime(unix_time, time_zone))	localtime(unix_time, time_zone) から年の特定の日を返します。
hour(localtime(unix_time, time_zone))	localtime(unix_time, time_zone) から時間を返します。
minute(localtime(unix_time, time_zone))	localtime(unix_time, time_zone) からの分数を返します。
sec(localtime(unix_time, time_zone))	localtime(unix_time, time_zone) から 2 番目を返すします

対応するタイムゾーン形式

タイムゾーンの引数は、次の方法で指定することができます。

- タイムゾーンオフセット - UTC の場合は `'Z'`、オフセット (`'+2'` または `'-5'`) を指定します。
- オフセットID - タイムゾーンの略語とオフセットを組み合わせて使用します。例えば、`'GMT+2'` と `'UTC-01:00'` です。タイムゾーンの略語には3文字のみを使用します。
- リージョンベースの ID - 例えば、`'Etc/GMT+12'` や `'Pacific/Pago_Pago'` など。

サポートされているタイムゾーンの略語

日付と時刻の機能は、次の 3 文字のタイムゾーンの略語をサポートしています。

- 東部標準時 -05:00
- 日本時間 - -10:00
- アメリカ標準時 -07:00
- ACT - オーストラリア/ダーウィン
- AET - オーストラリア/シドニー
- AGT - アメリカ/アルゼンチン/ブエノスアイレス
- ART - アフリカ/カイロ
- AST - アメリカ/アンカレッジ
- BET - アメリカ/サンパウロ
- BST - アジア/ダッカ
- CAT - アフリカ/ハラール
- CET - ヨーロッパ/パリ
- CNT - アメリカ/セントジョンズ
- CST - アメリカ/シカゴ
- CTT - アジア/上海
- EAT - アフリカ/アディスアベバ
- IET - アメリカ/インディアナ/インディアナポリス
- IST - アジア/コルカタ
- JST - アジア/東京
- MIT - 太平洋/アピア
- NET - アジア/エレバン
- NST - 太平洋/オークランド
- PLT - アジア/カラチ
- PRT - アメリカ/プエルトリコ
- PST - アメリカ/ロスアンジェルス
- SST - 太平洋/ガダルカナル
- VST - アジア/ホーチミン

サポートされているリージョンベースの ID

日付および時刻関数は、UTC+00:00 との関係で整理された次のリージョンベースの ID をサポートします。

- その他/GMT+12 (UTC-12:00)
- 太平洋/パゴパゴ (UTC-11:00)
- 太平洋/サモア (UTC-11:00)
- 太平洋/ニウエ (UTC-11) :00)
- アメリカ/サモア (UTC-11:00)
- その他/GMT+11 (UTC-11:00)
- 太平洋/ミッドウェイ (UTC-11:00)
- 太平洋/ホノルル (UTC-10:00)
- 太平洋/ラロトンガ島 (UTC-10:00)
- 太平洋/タヒチ (UTC-10:00)
- 太平洋/ジョンストン (UTC-10:00)
- アメリカ/ハワイ (UTC-10:00)
- SystemV/HST10 (UTC-10:00)
- その他/GMT+10 (UTC-10:00)
- 太平洋/マルケサス諸島 (UTC-09:30)
- その他/GMT+9 (UTC-09:00)
- 太平洋/ガンビア (UTC-09:00)
- アメリカ/アトカ (UTC-09:00)
- SystemV/YST9 (UTC-09:00)
- アメリカ/アダック (UTC-09:00)
- アメリカ/アリューシャン (UTC-09:00)
- その他/GMT+8 (UTC-08:00)
- アメリカ/アラスカ (UTC-08:00)
- アメリカ/ジュノー (UTC-08:00)
- アメリカ/メトラカトラ (UTC-08:00)
- アメリカ/ヤクタト (UTC-08:00)

- 太平洋/ピトケアン (UTC-08:00)
- アメリカ/シトカ (UTC-08:00)
- アメリカ/アンカレッジ (UTC-08:00)
- SystemV/PST8 (UTC-08:00)
- アメリカ/ノーム (UTC-08:00)
- SystemV/YST9YDT (UTC-08:00)
- カナダ/ユーコン (UTC-07:00)
- アメリカ/太平洋-新 (UTC-07:00)
- その他/GMT+7 (UTC-07:00)
- アメリカ/アリゾナ州 (UTC-07:00)
- アメリカ/ドーソンクリーク (UTC-07:00)
- カナダ/太平洋 (UTC-07:00)
- PST8PDT (UTC-07:00)
- SystemV/MST7 (UTC-07:00)
- アメリカ/ドーソン (UTC-07:00)
- アメリカ/ティファナ (UTC-07:00)
- アメリカ/ティファナ (UTC-07:00)
- アメリカ/クレストン (UTC-07:00)
- アメリカ/エルモシロ (UTC-07:00)
- アメリカ/サンタイザベル (UTC-07:00)
- アメリカ/バンクーバー (UTC-07:00)
- アメリカ/エンセナダ (UTC-07:00)
- アメリカ/フェニックス (UTC-07:00)
- アメリカ/ホワイトホース (UTC-07:00)
- アメリカ/フォートネルソン (UTC-07:00)
- SystemV/PST8PDT (UTC-07:00)
- アメリカ/ロサンゼルス (UTC-07:00)
- アメリカ/太平洋 (UTC-07:00)
- アメリカ/エルサルバドル (UTC-06:00)

- アメリカ/グアテマラ (UTC-06:00)
- アメリカ/ベリーズ (UTC-06:00)
- アメリカ/マナグア (UTC-06:00)
- アメリカ/テグシガルパ (UTC-06:00)
- その他/GMT+6 (UTC-06:00)
- 太平洋/イースター (UTC-06:00)
- メキシコ/バハスール (UTC-06:00)
- アメリカ/レジーナ (UTC-06:00)
- アメリカ/デンバー (UTC-06:00)
- 太平洋/ガラパゴス (UTC-06:00)
- アメリカ/イエローナイフ (UTC-06:00)
- アメリカ/スウィフトカレント (UTC-06:00)
- アメリカ/イヌビック (UTC-06:00)
- アメリカ/マサトラン (UTC-06:00)
- アメリカ/ボイズ (UTC-06:00)
- アメリカ/コスタリカ (UTC-06:00)
- MST7MDT (UTC-06:00)
- SystemV/CST6 (UTC-06:00)
- アメリカ/チワワ (UTC-06:00)
- アメリカ/オジナガ (UTC-06:00)
- チリ/イースター島 (UTC-06:00)
- アメリカ/山地 (UTC-06:00)
- アメリカ/エドモントン (UTC-06:00)
- カナダ/山地 (UTC-06:00)
- アメリカ/ケンブリッジベイ (UTC-06:00)
- ナバホ (UTC-06:00)
- SystemV/MST7MDT (UTC-06:00)
- カナダ/サスカチュワン (UTC-06:00)
- アメリカ/シップロック (UTC-06:00)

- アメリカ/パナマ (UTC-05:00)
- アメリカ/シカゴ (UTC-05:00)
- アメリカ/シカゴ (UTC-05:00)
- その他/GMT+5 (UTC-05:00)
- メキシコ/一般 (UTC-05:00)
- アメリカ/ポルトアクレ (UTC-05:00)
- アメリカ/グアヤキル (UTC-05:00)
- アメリカ/ランキンインレット (UTC-05:00)
- アメリカ/中部 (UTC-05:00)
- アメリカ/レイニーリバー (UTC-05:00)
- アメリカ/インディアナ/ノックス (UTC-05:00)
- アメリカ/ノースダコタ/ビューラ (UTC-05:00)
- アメリカ/モンテレイ (UTC-05:00)
- アメリカ/ジャマイカ (UTC-05:00)
- アメリカ/アティコカン (UTC-05:00)
- アメリカ/コーラルハーバー (UTC-05:00)
- アメリカ/ノースダコタ/中部 (UTC-05:00)
- アメリカ/ケイマン (UTC-05:00)
- アメリカ/インディアナ/テルシティ (UTC-05:00)
- アメリカ/メキシコシティ (UTC-05:00)
- アメリカ/マタモロス (UTC-05:00)
- CST6CDT (UTC-05:00)
- アメリカ/ノックスインディアナ州 (UTC-05:00)
- アメリカ/ボゴタ (UTC-05:00)
- アメリカ/メノマニー (UTC-05:00)
- アメリカ/レゾリュート (UTC-05:00)
- SystemV/EST5 (UTC-05:00)
- カナダ/中部 (UTC-05:00)
- ブラジル/アクレ (UTC-05:00)

- アメリカ/カンクン (UTC-05:00)
- アメリカ/リマ (UTC-05:00)
- アメリカ/バイアバンデラス (UTC-05:00)
- アメリカ/インディアナ-スターク (UTC-05:00)
- アメリカ/リオブランコ (UTC-05:00)
- SystemV/CST6CDT (UTC-05:00)
- ジャマイカ (UTC-05:00)
- アメリカ/メリダ (UTC-05:00)
- アメリカ/ノースダコタ/ニューセーラム (UTC-05:00)
- アメリカ/ウィニペグ (UTC-05:00)
- アメリカ/クイアバ (UTC-04:00)
- アメリカ/マリゴ (UTC-04:00)
- アメリカ/インディアナ/ピーターズバーグ (UTC-04:00)
- チリ/コンチネンタル (UTC-04:00)
- アメリカ/グランドターク (UTC-04:00)
- キューバ (UTC-04:00)
- その他/GMT+4 (UTC-04:00)
- アメリカ/マナウス (UTC-04:00)
- アメリカ/フォートウェイン (UTC-04:00)
- アメリカ/セントトーマス (UTC-04:00)
- アメリカ/アンギラ (UTC-04:00)
- アメリカ/ハバナ (UTC-04:00)
- アメリカ/ミシガン (UTC-04:00)
- アメリカ/バルバドス (UTC-04:00)
- アメリカ/ルイビル (UTC-04:00)
- アメリカ/キュラソー (UTC-04:00)
- アメリカ/ガイアナ (UTC-04:00)
- アメリカ/マルティニーク (UTC-04:00)
- アメリカ/プエルトリコ (UTC-04:00)

- アメリカ/ポートオブスペイン (UTC-04:00)
- SystemV/AST4 (UTC-04:00)
- アメリカ/インディアナ/ベベイ (UTC-04:00)
- アメリカ/インディアナ/ビンセンズ (UTC-04:00)
- アメリカ/クラレンダイク (UTC-04:00)
- アメリカ/アンティグア (UTC-04:00)
- アメリカ/インディアナポリス (UTC-04:00)
- アメリカ/イカルイト (UTC-04:00)
- アメリカ/セントビンセント (UTC-04:00)
- アメリカ/ケンタッキー/ルイビル (UTC-04:00)
- アメリカ/ドミニカ (UTC-04:00)
- アメリカ/アスンシオン (UTC-04:00)
- EST5EDT (UTC-04:00)
- アメリカ/ナッソー (UTC-04:00)
- アメリカ/ケンタッキー/モンティセロ (UTC-04:00)
- ブラジル/西部 (UTC-04:00)
- アメリカ/アルバ (UTC-04:00)
- アメリカ/インディアナ/インディアナポリス (UTC-04:00)
- アメリカ/サンティアゴ (UTC-04:00)
- アメリカ/ラパス (UTC-04:00)
- アメリカ/サンダーベイ (UTC-04:00)
- アメリカ/インディアナ/マレンゴ (UTC-04:00)
- アメリカ/ブランサブロン (UTC-04:00)
- アメリカ/サントドミンゴ (UTC-04:00)
- アメリカ/東部 (UTC-04:00)
- カナダ/東部 (UTC-04:00)
- アメリカ/ポルトープランス (UTC-04:00)
- アメリカ/サンバルテルミー (UTC-04:00)
- アメリカ/ニピゴン (UTC-04:00)

- アメリカ/東インディアナ (UTC-04:00)
- アメリカ/セントルシア (UTC-04:00)
- アメリカ/モントセラト (UTC-04:00)
- アメリカ/ローワープリンセズ (UTC-04:00)
- アメリカ/デトロイト (UTC-04:00)
- アメリカ/トルトラ (UTC-04:00)
- アメリカ/ポルトヴェーリョ (UTC-04:00)
- アメリカ/カンポグランデ (UTC-04:00)
- アメリカ/バージン (UTC-04:00)
- アメリカ/パングナータング (UTC-04:00)
- アメリカ/モントリオール (UTC-04:00)
- アメリカ/インディアナ/ウィナマック (UTC-04:00)
- アメリカ/ボアビスタ (UTC-04:00)
- アメリカ/グレナダ (UTC-04:00)
- アメリカ/ニューヨーク (UTC-04:00)
- アメリカ/セントキッツ (UTC-04:00)
- アメリカ/カラカス (UTC-04:00)
- アメリカ/グアドループ (UTC-04:00)
- アメリカ/トロント (UTC-04:00)
- SystemV/EST5EDT (UTC-04:00)
- アメリカ/アルゼンチン/カタマルカ (UTC-03:00)
- カナダ/大西洋 (UTC-03:00)
- アメリカ/アルゼンチン/コルドバ (UTC-03:00)
- アメリカ/アラゲイナ (UTC-03:00)
- アメリカ/アルゼンチン/サルタ (UTC-03:00)
- その他/GMT+3 (UTC-03:00)
- アメリカ/モンテビデオ (UTC-03:00)
- ブラジル/東部 (UTC-03:00)
- アメリカ/アルゼンチン/メンドーサ (UTC-03:00)
- アメリカ/アルゼンチン/リオガジェゴス (UTC-03:00)

- アメリカ/カタマルカ (UTC-03:00)
- アメリカ/コルドバ (UTC-03:00)
- アメリカ/サンパウロ (UTC-03:00)
- アメリカ/アルゼンチン/フフイ (UTC-03:00)
- アメリカ/カイエンヌ (UTC-03:00)
- アメリカ/レシフェ (UTC-03:00)
- アメリカ/ブエノスアイレス (UTC-03:00)
- アメリカ/パラマリボ (UTC-03:00)
- アメリカ/モンクトン (UTC-03:00)
- アメリカ/メンドーサ (UTC-03:00)
- アメリカ/サンタレム (UTC-03:00)
- 大西洋/バミューダ (UTC-03:00)
- アメリカ/マセイオ (UTC-03:00)
- 大西洋/スタンレー (UTC-03:00)
- アメリカ/ハリファックス (UTC-03:00)
- 南極大陸/ロセラ (UTC-03:00)
- アメリカ/アルゼンチン/サンルイス (UTC-03:00)
- アメリカ/アルゼンチン/ウシュアイア (UTC-03:00)
- 南極大陸/パーマー (UTC-03:00)
- アメリカ/プンタアレナス (UTC-03:00)
- アメリカ/グレイスベイ (UTC-03:00)
- アメリカ/フォルタレザ (UTC-03:00)
- アメリカ/チュール (UTC-03:00)
- アメリカ/アルゼンチン/ラリオハ州 (UTC-03:00)
- アメリカ/ベレム (UTC-03:00)
- アメリカ/フフイ (UTC-03:00)
- アメリカ/バイア (UTC-03:00)
- アメリカ/グースベイ (UTC-03:00)
- アメリカ/アルゼンチン/サンファン (UTC-03:00)
- アメリカ/アルゼンチン/コモドリヴァダヴィア (UTC-03:00)

- アメリカ/アルゼンチン/トゥクマン (UTC-03:00)
- アメリカ/ロサリオ (UTC-03:00)
- SystemV/AST4ADT (UTC-03:00)
- アメリカ/アルゼンチン/ブエノスアイレス (UTC-03:00)
- アメリカ/セントジョンズ (UTC-02:30)
- カナダ/ニューファンドランド (UTC-02:30)
- アメリカ/ミクロン (UTC-02:00)
- その他/GMT+2 (UTC-02:00)
- アメリカ/ゴッドホープ (UTC-02:00)
- アメリカ/ノローニャ (UTC-02:00)
- ブラジル/デノローニャ (UTC-02:00)
- 大西洋/サウスジョージア (UTC-02:00)
- その他/GMT+1 (UTC-01:00)
- 大西洋/カーボベルデ (UTC-01:00)
- 太平洋/キリティマティ (UTC+14:00)
- その他/GMT-14 (UTC+ 14:00)
- 太平洋/ファカオフォ (UTC+13:00)
- 太平洋/エンダーベリー (UTC+13:00)
- 太平洋/アピア (UTC+13:00)
- 太平洋/トンガタプ (UTC+13:00)
- その他/GMT-13 (UTC+ 13:00)
- NZ-CHAT (UTC+12:45)
- 太平洋/チャタム (UTC+12:45)
- 太平洋/クェゼリン (UTC+12:00)
- 南極大陸/マクマード (UTC+12:00)
- 太平洋/ウォリス (UTC+12:00)
- 太平洋/フィジー (UTC+12:00)
- 太平洋/フナフチ (UTC+12:00)
- 太平洋/ナウル (UTC+12:00)
- クェゼリン (UTC+12:00)

- NZ (UTC+ 12:00)
- 太平洋/ウェイク (UTC+12:00)
- 南極大陸/南極 (UTC+12:00)
- 太平洋/タラワ (UTC+12:00)
- 太平洋/オークランド (UTC+12:00)
- アジア/カムチャツカ (UTC+12:00)
- その他/GMT-12 (UTC+ 12:00)
- アジア/アナディリ (UTC+ 12:00)
- 太平洋/マジュロ (UTC+12:00)
- 太平洋/ポナペ (UTC+11:00)
- 太平洋/ブーゲンビル (UTC+11:00)
- 南極大陸/マッカリー (UTC+11:00)
- 太平洋/ポンペイ (UTC+11:00)
- 太平洋/エフェート (UTC+11:00)
- 太平洋/ノーフォーク (UTC+11:00)
- アジア/マガダン (UTC+11:00)
- 太平洋/コスラエ (UTC+11:00)
- アジア/サハリン (UTC+11:00)
- 太平洋/ヌメア (UTC+11:00)
- その他/GMT-11 (UTC+ 11:00)
- アジア/スレドネコリムスク (UTC+11:00)
- 太平洋/ガダルカナル (UTC+11:00)
- オーストラリア/ロードハウ (UTC+10:30)
- オーストラリア/LHI (UTC+10:30)
- オーストラリア/ホバート (UTC+10:00)
- 太平洋/ヤップ (UTC+10:00)
- オーストラリア/タスマニア (UTC+10:00)
- 太平洋/ポートモレスビー (UTC+10:00)
- オーストラリア/ACT (UTC+10:00)
- オーストラリア/ビクトリア (UTC+10:00)

- 太平洋/チューク (UTC+10:00)
- オーストラリア/クイーンズランド (UTC+10:00)
- オーストラリア/キャンベラ (UTC+10:00)
- オーストラリア/カリー (UTC+10:00)
- 太平洋/グアム (UTC+ 10:00)
- 太平洋/トラック (UTC+10:00)
- オーストラリア/NSW (UTC+10:00)
- アジア/ウラジオストク (UTC+10:00)
- 太平洋/サイパン (UTC+10:00)
- 南極大陸/デュモンデュルビル (UTC+10:00)
- オーストラリア/シドニー (UTC+10:00)
- オーストラリア/ブリスベン (UTC+10:00)
- その他/GMT-10 (UTC+ 10:00)
- アジア/ウストネーラ (UTC+10:00)
- オーストラリア/メルボルン (UTC+10:00)
- オーストラリア/リンデマン (UTC+10:00)
- オーストラリア/北部 (UTC+09:30)
- オーストラリア/ヤンコウィンナ (UTC+09:30)
- オーストラリア/アデレード (UTC+09:30)
- オーストラリア/ブロークンヒル (UTC+09:30)
- オーストラリア/南部 (UTC+09:30)
- オーストラリア/ダーウィン (UTC+09:30)
- その他/GMT-9 (UTC+ 09:00)
- 太平洋/パラオ (UTC+09:00)
- アジア/チタ (UTC+09:00)
- アジア/ディリ (UTC+09:00)
- アジア/ジャヤプラ (UTC+09:00)
- アジア/ヤクーツク (UTC+09:00)
- アジア/平壌 (UTC+09:00)
- ROK (UTC+09:00)

- アジア/ソウル (UTC+09:00)
- アジア/ハンディガ (UTC+09:00)
- 日本 (UTC+09:00)
- アジア/東京 (UTC+09:00)
- オーストラリア/ユークラ (UTC+08:45)
- アジア/クチン (UTC+08:00)
- アジア/重慶 (UTC+08:00)
- その他/GMT-8 (UTC+ 08:00)
- オーストラリア/パース (UTC+08:00)
- アジア/マカオ (UTC+08:00)
- アジア/マカオ (UTC+08:00)
- アジア/チョイバルサン (UTC+08:00)
- アジア/上海 (UTC+08:00)
- 南極大陸/ケーシー (UTC+08:00)
- アジア/ウランバートル (UTC+08:00)
- アジア/重慶 (UTC+08:00)
- アジア/ウランバートル (UTC+08:00)
- アジア/台北 (UTC+08:00)
- アジア/マニラ (UTC+08:00)
- PRC (UTC+08:00)
- アジア/ウジュンパンダン (UTC+08:00)
- アジア/ハルビン (UTC+08:00)
- シンガポール (UTC+08:00)
- アジア/ブルネイ (UTC+08:00)
- オーストラリア/西部 (UTC+08:00)
- アジア/香港 (UTC+08:00)
- アジア/マカッサル (UTC+08:00)
- 香港 (UTC+08:00)
- アジア/クアラルンプール (UTC+08:00)
- アジア/イルクーツク (UTC+08:00)

- アジア/シンガポール (UTC+08:00)
- アジア/ポンティアナック (UTC+07:00)
- その他/GMT-7 (UTC+ 07:00)
- アジア/プノンペン (UTC+07:00)
- アジア/ノボシビルスク (UTC+07:00)
- 南極大陸/デイビス (UTC+07:00)
- アジア/トムスク (UTC+07:00)
- アジア/ジャカルタ (UTC+07:00)
- アジア/バルナウル (UTC+07:00)
- インド洋/クリスマス (UTC+07:00)
- アジア/ホーチミン (UTC+07:00)
- アジア/ホバート (UTC+07:00)
- アジア/バンコク (UTC+07:00)
- アジア/ビエンチャン (UTC+07:00)
- アジア/ノボクズネツク (UTC+07:00)
- アジア/クラスノヤルスク (UTC+07:00)
- アジア/サイゴン (UTC+07:00)
- アジア/ヤンゴン (UTC+06:30)
- アジア/ラングーン (UTC+06:30)
- インド洋/ココス (UTC+06:30)
- アジア/カシュガル (UTC+06:00)
- その他/GMT-6 (UTC+ 06:00)
- アジア/アルマティ (UTC+06:00)
- アジア/ダッカ (UTC+06:00)
- アジア/オムスク (UTC+06:00)
- アジア/ダッカ (UTC+06:00)
- インド洋/チャゴス (UTC+06:00)
- アジア/キジローダ (UTC+06:00)
- アジア/ビシュケク (UTC+06:00)
- 南極大陸/ボストーク (UTC+06:00)

- アジア/ウルムチ (UTC+06:00)
- アジア/ティンプー (UTC+06:00)
- アジア/ティンプー (UTC+06:00)
- アジア/カトマンズ (UTC+05:45)
- アジア/カトマンズ (UTC+05:45)
- アジア/コルカタ (UTC+05:30)
- アジア/コロンボ (UTC+05:30)
- アジア/カルカッタ (UTC+05:30)
- アジア/アクタウ (UTC+05:00)
- その他/GMT-5 (UTC+ 05:00)
- アジア/サマルカンド (UTC+05:00)
- アジア/カラチ (UTC+05:00)
- アジア/エカテリンブルク (UTC+05:00)
- アジア/ドゥシャンベ (UTC+ 05:00)
- アジア/ドゥシャンベ (UTC+05:00)
- アジア/オラール (UTC+05:00)
- アジア/タシケント (UTC+05:00)
- 南極大陸/モーソン (UTC+05:00)
- アジア/アクトベ (UTC+05:00)
- アジア/アシュハバード (UTC+05:00)
- アジア/アシガバート (UTC+05:00)
- アジア/アティラウ (UTC+05:00)
- インド洋/ケルゲレン (UTC+05:00)
- イラン (UTC+04:30)
- アジア/テヘラン (UTC+04:30)
- アジア/カブール (UTC+04:30)
- アジア/エレバン (UTC+04:00)
- その他/GMT-4 (UTC+ 04:00)
- その他/GMT-4 (UTC+ 04:00)
- アジア/ドバイ (UTC+04:00)

- インド洋/レユニオン (UTC+04:00)
- ヨーロッパ/サラトフ (UTC+04:00)
- ヨーロッパ/サマラ (UTC+04:00)
- インド/マエ (UTC+04:00)
- アジア/バクー (UTC+04:00)
- アジア/マスカット (UTC+04:00)
- ヨーロッパ/ボルゴグラード (UTC+04:00)
- ヨーロッパ/アストラカン (UTC+04:00)
- アジア/トビリシ (UTC+04:00)
- ヨーロッパ/ウリヤノフスク (UTC+04:00)
- アジア/アデン (UTC+03:00)
- アフリカ/ナイロビ (UTC+03:00)
- ヨーロッパ/イスタンブール (UTC+03:00)
- その他/GMT-3 (UTC+ 03:00)
- ヨーロッパ/ザポロージエ (UTC+03:00)
- イスラエル (UTC+03:00)
- インド洋/コモロ (UTC+03:00)
- 南極大陸/シオワ (UTC+03:00)
- アフリカ/モガディシュ (UTC+03:00)
- ヨーロッパ/ブカレスト (UTC+03:00)
- アフリカ/アスメラ (UTC+03:00)
- ヨーロッパ/マリエハムン (UTC+03:00)
- アジア/イスタンブール (UTC+03:00)
- ヨーロッパ/ティラスポリ (UTC+03:00)
- ヨーロッパ/モスクワ (UTC+03:00)
- ヨーロッパ/キシナウ (UTC+03:00)
- ヨーロッパ/ヘルシンキ (UTC+03:00)
- アジア/ベイルート (UTC+03:00)
- アジア/テルアビブ (UTC+03:00)
- アフリカ/ジブチ (UTC+03:00)

- ヨーロッパ/シンフェロポリ (UTC+03:00)
- ヨーロッパ/ソフィア (UTC+03:00)
- アジア/ガザ (UTC+03:00)
- アフリカ/アスマラ (UTC+03:00)
- ヨーロッパ/リガ (UTC+03:00)
- アジア/バグダッド (UTC+03:00)
- アジア/ダマスカス (UTC+03:00)
- アフリカ/ダルエスサラーム (UTC+03:00)
- アフリカ/アディスアベバ (UTC+03:00)
- ヨーロッパ/ウジゴロド (UTC+03:00)
- アジア/エルサレム (UTC+03:00)
- アジア/リヤド (UTC+03:00)
- アジア/クウェート (UTC+03:00)
- ヨーロッパ/キーロフ (UTC+03:00)
- アフリカ/カンパラ (UTC+03:00)
- ヨーロッパ/ミンスク (UTC+03:00)
- アジア/カタール (UTC+03:00)
- ヨーロッパ/キエフ (UTC+03:00)
- アジア/バーレーン (UTC+03:00)
- ヨーロッパ/ビリニュス (UTC+03:00)
- インド洋/アンタナナリボ (UTC+03:00)
- インド洋/マヨット (UTC+03:00)
- ヨーロッパ/タリン (UTC+03:00)
- トルコ (UTC+03:00)
- アフリカ/ジュバ (UTC+03:00)
- アジア/ニコシア (UTC+03:00)
- アジア/ファマグスタ (UTC+03:00)
- W-SU (UTC+03:00)
- EET (UTC+03:00)
- アジア/ヘブロン (UTC+03:00)

- アジア/アンマン (UTC+03:00)
- ヨーロッパ/ニコシア (UTC+03:00)
- ヨーロッパ/アテネ (UTC+03:00)
- アフリカ/カイロ (UTC+02:00)
- アフリカ/ババーネ (UTC+02:00)
- ヨーロッパ/ブリュッセル (UTC+02:00)
- ヨーロッパ/ワルシャワ (UTC+02:00)
- CET (UTC+02:00)
- ヨーロッパ/ルクセンブルグ (UTC+02:00)
- その他/GMT-2 (UTC+ 02:00)
- リビア (UTC+02:00)
- アフリカ/キガリ (UTC+02:00)
- アフリカ/トリポリ (UTC+02:00)
- ヨーロッパ/カリーニングラード (UTC+02:00)
- アフリカ/ウィンドフック (UTC+02:00)
- ヨーロッパ/マルタ (UTC+02:00)
- ヨーロッパ/ビュッセン (UTC+02:00)
-
- ヨーロッパ/スコピエ (UTC+02:00)
- ヨーロッパ/サラエボ (UTC+02:00)
- ヨーロッパ/ローマ (UTC+02:00)
- ヨーロッパ/チューリッヒ (UTC+02:00)
- ヨーロッパ/ジブラルタル (UTC+02:00)
- アフリカ/ルブンバシ (UTC+02:00)
- ヨーロッパ/ファドゥーツ (UTC+02:00)
- ヨーロッパ/リュブリャナ (UTC+02:00)
- ヨーロッパ/ベルリン (UTC+02:00)
- ヨーロッパ/ストックホルム (UTC+02:00)
- ヨーロッパ/ブダペスト (UTC+02:00)
- ヨーロッパ/ザグレブ (UTC+02:00)

- ヨーロッパ/パリ (UTC+02:00)
- アフリカ/セウタ (UTC+02:00)
- ヨーロッパ/プラハ (UTC+02:00)
- 南極大陸/トロール (UTC+02:00)
- アフリカ/ハボローネ (UTC+02:00)
- ヨーロッパ/コペンハーゲン (UTC+02:00)
- ヨーロッパ/ウィーン (UTC+02:00)
- ヨーロッパ/ティラネ (UTC+02:00)
- MET (UTC+02:00)
- ヨーロッパ/アムステルダム (UTC+02:00)
- アフリカ/マプト (UTC+02:00)
- ヨーロッパ/サンマリノ (UTC+02:00)
- ポーランド (UTC+02:00)
- ヨーロッパ/アンドラ (UTC+02:00)
- ヨーロッパ/オスロ (UTC+02:00)
- ヨーロッパ/ポドゴリツァ (UTC+02:00)
- アフリカ/ブジュンブラ (UTC+02:00)
- 大西洋/ヤンマイエン (UTC+02:00)
- アフリカ/マセル (UTC+02:00)
- ヨーロッパ/マドリッド (UTC+02:00)
- アフリカ/ブランタイヤ (UTC+02:00)
- アフリカ/ルサカ (UTC+02:00)
- アフリカ/ハラレ (UTC+02:00)
- アフリカ/ハルツーム (UTC+02:00)
- アフリカ/ヨハネスブルグ (UTC+02:00)
- ヨーロッパ/ベオグラード (UTC+02:00)
- ヨーロッパ/ブラチスラバ (UTC+02:00)
- 北極/ロンゲルビアン (UTC+02:00)
- エジプト (UTC+02:00)

- ヨーロッパ/バチカン (UTC+02:00)
- ヨーロッパ/モナコ (UTC+02:00)
- ヨーロッパ/ロンドン (UTC+01:00)
- その他/GMT-1 (UTC+ 01:00)
- ヨーロッパ/ジャージー (UTC+01:00)
- ヨーロッパ/ガーンジー (UTC+01:00)
- ヨーロッパ/マン島 (UTC+01:00)
- アフリカ/チュニス (UTC+01:00)
- アフリカ/マラボ (UTC+01:00)
- GB-Eire (UTC+01:00)。
- アフリカ/ラゴス (UTC+01:00)
- アフリカ/アルジェ (UTC+01:00)
- GB (UTC+01:00)
- ポルトガル (UTC+01:00)
- アフリカ/サントメ (UTC+01:00)
- アフリカ/ンジャメナ (UTC+01:00)
- 大西洋/フェロー (UTC+01:00)
- アイルランド (UTC+01:00)
- 大西洋/フェロー (UTC+01:00)
- ヨーロッパ/ダブリン (UTC+01:00)
- アフリカ/リブレビル (UTC+01:00)
- アフリカ/エルアイウン (UTC+01:00)
- アフリカ/エルアイウン (UTC+01:00)
- アフリカ/ドゥアラ (UTC+01:00)
- アフリカ/ブラザビル (UTC+01:00)
- アフリカ/ポルトノボ (UTC+01:00)
- 大西洋/マデイラ (UTC+01:00)
- ヨーロッパ/リスボン (UTC+01:00)
- 大西洋/Canary (UTC+01:00)

- アフリカ/カサブランカ (UTC+01:00)
- ヨーロッパ/ベルファスト (UTC+01:00)
- アフリカ/ルアンダ (UTC+01:00)
- アフリカ/キンシャサ (UTC+01:00)
- アフリカ/バンギ (UTC+01:00)
- WET (UTC+01:00)
- アフリカ/ニアメー (UTC+01:00)
- GMT (UTC+00:00)
- その他/GMT-0 (UTC+ 00:00)
- 大西洋/セントヘレナ (UTC+00:00)
- その他/GMT+0 (UTC+ 00:00)
- アフリカ/バンジュール (UTC+00:00)
- その他etc/GMT (UTC+ 00:00)
- アフリカ/フリータウン (UTC+00:00)
- アフリカ/バマコ (UTC+00:00)
- アフリカ/コナクリ (UTC+00:00)
- ユニバーサル (UTC+ 00:00)
- アフリカ/ヌアクショット (UTC+00:00)
- UTC (UTC+00:00)
- その他/ユニバーサル (UTC+00:00)
- 大西洋/アゾレス (UTC+00:00)
- アフリカ/アビジャン (UTC+00:00)
- アフリカ/アクラ (UTC+00:00)
- その他/UCT (UTC+ 00:00)
- GMT0 (UTC+00:00)
- ズールー (UTC+00:00) ズールー (UTC+00:00)
- アフリカ/ワガドゥグ (UTC+00:00)
- 大西洋/レイキャビク (UTC+00:00)
- その他/ズールー (UTC+ 00:00)
- アイスランド (UTC+00:00)

- アフリカ/ロメ (UTC+00:00)
- グリニッジ (UTC+00:00)
- その他/gmt0 (UTC+ 00:00)
- アメリカ/ダンマルクシャーン (UTC+00:00)
- アフリカ/ダカール (UTC+00:00)
- アフリカ/ビサウ (UTC+00:00)
- その他/グリニッジ (UTC+ 00:00)
- アフリカ/ティンブクトゥ (UTC+00:00)
- UCT (UTC+00:00)
- アフリカ/モンロビア (UTC+00:00)
- その他/UTC (UTC+00:00)

数式表現チュートリアル

AWS IoT SiteWiseで数式表現を使うには、次のチュートリアルを参考にしてください。

トピック

- 式で文字列を使用する
- データポイントをフィルタリングする
- 条件に一致するデータポイントをカウントする
- 数式の遅延データ
- 数式のデータ品質
- 未定義の値、無限の値、およびオーバーフロー値

式で文字列を使用する

数式表現の中で文字列を操作することができます。また、属性や測定のプロパティを参照する変数から文字列を入力することができます。

> **Important**
>
> 数式では、倍値または文字列値のみを出力できます。入れ子になった式は、文字列など他のデータ型を出力することができますが、式全体としては、数値または文字列として評価されなければなりません。[jp function] (jp関数) を使うと、文字列を数値に変換することができま

す。ブール値は 1 (true) または 0 (false) でなければならない。詳細については、「未定義の値、無限の値、およびオーバーフロー値」を参照してください。

AWS IoT SiteWise には、文字列の操作に使用できる次の式機能が用意されています。

- [String literals] (文字列リテラル)
- [index operator] (インデックス演算子) (`s[index]`)
- [slice operator] (スライス演算子) (`s[start:end:step]`)
- [Comparison functions]比較関数: 文字列を[lexicographic order] (辞書式順序) に比較する関数です。
- シリアル化された JSON オブジェクトをパースし、文字列を数値に変換できる `jp` 関数を含む文字列関数

データポイントをフィルタリングする

[if function] (if関数) を使用すると、条件を満たさないデータポイントをフィルタリングすることができます。`if` 関数は条件を評価し、`true`、`false` の結果に対して異なる値を返す。`if` 関数の 1 ケースの出力として[none constant] (無し定数) を使用すると、そのケースのデータポイントを破棄することができます。

条件に合致するデータポイントをフィルタリングする場合。

- `if` 関数を使用して、ある条件を満たすかどうかをチェックする条件を定義し、`none` を `result_if_true` または `result_if_false` の値として返す変換を作成します。

Example 例: 水が沸騰していないデータポイントをフィルタリングする。

ある機械の中の水の温度 (摂氏) を測定する測定器 `temp_c` があるとしましょう。次のような変換を定義することで、水が沸騰していないデータポイントをフィルタリングすることができます。

- 変換: `boiling_temps = if(gte(temp_c, 100), temp_c, none)` - 100℃以上であれば温度を返し、そうでなければデータポイントを返さない。

条件に一致するデータポイントをカウントする

[comparison functions] (比較関数) と [sum ()] を使用して、条件が真であるデータポイントの数を カウントできます。

条件に一致するデータポイントをカウントする場合。

1. 比較関数を使用して、別のプロパティのフィルター条件を定義する変換を作成します。
2. その条件が満たされるデータポイントを合計するメトリクスを作成します。

Example 例: 水が沸騰しているデータポイントの数を数える

ある機械の中の水の温度 (摂氏) を測定する測定器 `temp_c` があるとしましょう。次の変換プロパティとメトリクスプロパティを定義して、水が沸騰しているデータポイントの数をカウントできます。

- 変換: `is_boiling = gte(temp_c, 100)` - 温度が100℃以上であれば 1、そうでなければ0を返す。
- メトリクス: `boiling_count = sum(is_boiling)` - 水が沸騰しているデータポイントの数を返します。

数式の遅延データ

AWS IoT SiteWise は、最大 7 日経過したデータの遅延データ取り込みをサポートします。が遅延データ AWS IoT SiteWise を受信すると、過去のウィンドウに遅延データを入力するメトリクスの既存の値を再計算します。これらの再計算では、データ処理料金が発生します。

> **Note**
>
> が遅延データを入力するプロパティ AWS IoT SiteWise を計算する場合、各プロパティの現在の式を使用します。

がメトリクスの過去のウィンドウ AWS IoT SiteWise を再計算すると、そのウィンドウの以前の値が置き換えられます。そのメトリクスの通知を有効にした場合、 はプロパティ値通知 AWS IoT SiteWise も発行します。つまり、以前に通知を受信したのと同じプロパティとタイムスタンプについて、新しいプロパティ値の更新通知を受け取ることができます。アプリケーションまたはデータレイクがプロパティ値通知を使用する場合、データが正確になるように、前の値を新しい値で更新する必要があります。

数式のデータ品質

では AWS IoT SiteWise、各データポイントに品質コードがあり、次のいずれかになります。

- GOOD - データはいずれの問題の影響も受けません。
- BAD - データはセンサーの障害などの問題による影響を受けます。
- UNCERTAIN - データはセンサーの不正確さなどの問題による影響を受けます。

AWS IoT SiteWise は、変換とメトリクスを計算するときにGOOD品質データのみを消費します。は、正常な計算のためにGOOD品質データのみを AWS IoT SiteWise 出力します。計算が失敗した場合、 AWS IoT SiteWise はその計算のデータポイントを出力しません。これは、計算の結果、未定義、無限、またはオーバーフロー値になった場合に発生します。

データのクエリとデータ品質によるフィルタ処理の詳細については、「からのデータのクエリ AWS IoT SiteWise」を参照してください。

未定義の値、無限の値、およびオーバーフロー値

一部の数式 (`x / 0`、、 など`log(0)`) は`sqrt(-1)`、実数システムで未定義、無限、または でサポートされている範囲外の値を計算します AWS IoT SiteWise。アセットプロパティの式が未定義、無限、またはオーバーフロー値を計算した場合、その計算のデータポイントは出力 AWS IoT SiteWise されません。

AWS IoT SiteWise 式の結果として数値以外の値を計算した場合、 はデータポイントを出力しません。つまり、文字列、配列、[none constant] (無し定数) をコンピューティングする数式を定義した場合、 AWS IoT SiteWise はそのコンピューティングのデータポイントを出力しないのです。

Example 例

次の式はそれぞれ、数値として表現 AWS IoT SiteWise できない値になります。これらの式を計算しても、データポイントは出力 AWS IoT SiteWise されません。

- `x / 0` は未定義です。
- `log(0)` は未定義です。
- `sqrt(-1)` は実数系では未定義です。
- `"hello" + " world"` は文字列です。
- `jp('{"values":[3,6,7]}', '$.values')` は配列です。
- `if(gte(temp, 300), temp, none)` が none より小さいとき、`temp` は `300` です。

カスタム複合モデルを作成する (コンポーネント)

カスタム複合モデル、またはコンポーネント (コンソールを使用している場合) を使用すると、アセットモデルとコンポーネントモデルをより高度に整理することができます。これらを使用し、プロパティをグループ化するか、他のモデルを参照することで、モデルを構造化できます。カスタム複合モデルの使用の詳細については、「カスタム複合モデル (コンポーネント)」を参照してください。

既存のアセットモデルまたはコンポーネントモデル内にカスタム複合モデルを作成します。カスタム複合モデルには 2 つのタイプがあります。モデル内の関連プロパティをグループ化するには、インラインカスタム複合モデルを作成できます。アセットモデルまたはコンポーネントモデル内のコンポーネントモデルを参照するには、コンポーネントモデルベースのカスタム複合モデルを作成できます。

以下のセクションでは、 AWS IoT SiteWise API を使用してカスタム複合モデルを作成する方法について説明します。

トピック

- インラインコンポーネントを作成する (コンソール)
- インラインカスタム複合モデルを作成する (AWS CLI)
- コンポーネントモデルベースのコンポーネントを作成する (コンソール)
- コンポーネントモデルベースのカスタム複合モデルを作成する (AWS CLI)

インラインコンポーネントを作成する (コンソール)

AWS IoT SiteWise コンソールを使用して、独自のプロパティを定義するインラインコンポーネントを作成できます。

> **Note**
>
> これはインラインコンポーネントであるため、これらのプロパティは現在のアセットモデルにのみ適用され、他の場所では共有されません。
> 再利用可能なモデルを作成する必要がある場合 (複数のアセットモデル間で共有する場合や、1 つのアセットモデルに複数のインスタンスを含める場合など)、代わりにコンポーネントモデルに基づいてコンポーネントを作成する必要があります。詳細については、次のセクションを参照してください。

コンポーネントを作成するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. コンポーネントを追加するアセットモデルを選択します。
4. [プロパティ] タブで、[コンポーネント] を選択します。
5. [コンポーネントを作成] を選択します。
6. [コンポーネントを作成する] ページで、次の操作を行います。
   a. コンポーネントの [名前] (**ServoMotor** や **ServoMotor Model** など) を入力します。この名前は、このリージョンのアカウントのすべてのモデルで一意である必要があります。
   b. (オプション) モデルの [属性定義] を追加します。属性は、ほとんど変化しない情報を表します。詳細については、「静的データ (属性) を定義する」を参照してください。
   c. (オプション) モデルの [測定の定義] を追加します。測定値は、機器からのデータストリームを表します。詳細については、「機器からのデータストリームを定義する (測定値) 」を参照してください。
   d. (オプション) モデルの [定義を変換する] を追加します。変換とは、あるフォームから別のフォームにデータをマッピングするための数式です。詳細については、「データを変換する (変換) 」を参照してください。
   e. (オプション) モデルの [メトリクスの定義] を追加します。メトリクスとは、時間間隔でのデータを集計する数式です。メトリクスは、関連するアセットからデータを入力できるため、オペレーションまたはオペレーションのサブセットを表す値をコンピューティングすることができます。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。
   f. [コンポーネントを作成] を選択します。

インラインカスタム複合モデルを作成する (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用して、独自のプロパティを定義するインラインカスタム複合モデルを作成できます。

CreateAssetModelCompositeModel オペレーションを使用して、プロパティを持つインラインモデルを作成します。このオペレーションでは、次の構造を持つペイロードが必要です。

> **Note**
>
> これはインライン複合モデルであるため、これらのプロパティは現在のアセットモデルにのみ適用され、他の場所では共有されません。「インライン」と呼ばれるのは、`composedAssetModelId` フィールドの値を提供しないためです。
> 再利用可能なモデルを作成する必要がある場合 (複数のアセットモデル間で共有する場合や、1 つのアセットモデル内に複数のインスタンスを含める場合など)、代わりにコンポーネントモデルベースの複合モデルを作成する必要があります。詳細については、次のセクションを参照してください。

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

コンポーネントモデルベースのコンポーネントを作成する (コンソール)

AWS IoT SiteWise コンソールを使用して、コンポーネントモデルに基づいてコンポーネントを作成できます。

コンポーネントモデルベースのコンポーネントを作成するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. コンポーネントを追加するアセットモデルを選択します。
4. [プロパティ] タブで、[コンポーネント] を選択します。
5. [コンポーネントを作成] を選択します。
6. [コンポーネントを作成する] ページで、次の操作を行います。
   a. コンポーネントが基づくコンポーネントモデルを選択します。
   b. コンポーネントの [名前] (**ServoMotor** や **ServoMotor Model** など) を入力します。この名前は、このリージョンのアカウントのすべてのモデルで一意である必要があります。
   c. [コンポーネントを作成] を選択します。

コンポーネントモデルベースのカスタム複合モデルを作成する (AWS CLI)

を使用して、アセットモデル内にcomponent-model-basedカスタム複合モデル AWS CLI を作成できます。コンポーネントモデルベースのカスタム複合モデルは、既に他の場所で定義したコンポーネントモデルへの参照です。

CreateAssetModelCompositeModel オペレーションを使用して、コンポーネントモデルベースのカスタム複合モデルを作成します。このオペレーションでは、次の構造を持つペイロードが必要です。

> **Note**
>
> この例では、`composedAssetModelId` の値は、既存のコンポーネントモデルのアセットモデル ID または外部 ID です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。コンポーネントモデルの作成方法の例については、「コンポーネントモデルを作成する (AWS CLI)」を参照してください。

```
{
    "assetModelCompositeModelName": "CNCLathe_ServoMotorA",
    "assetModelCompositeModelType": "CUSTOM",
    "composedAssetModelId": component model ID
]
```

これは単なる参照であるため、コンポーネントモデルベースのカスタム複合モデルには、名前以外の独自のプロパティはありません。

同じコンポーネントの複数のインスタンスをアセットモデルに追加する場合 (複数のサーボモーターを持つ CNC マシンなど)、それぞれに独自の名前があり、すべて同じ `composedAssetModelId` を参照する複数のコンポーネントモデルベースのカスタム複合モデルを追加できます。

コンポーネントを別のコンポーネントの中に入れ子にできます。そのためには、この例に示すように、コンポーネントモデルベースの複合モデルをコンポーネントモデルの 1 つに追加できます。

でアセットモデルのアセットを作成する AWS IoT SiteWise

アセットモデルからアセットを作成できます。アセットを作成する前に、アセットモデルが必要です。アセットモデルを作成していない場合は、「でアセットモデルを作成する AWS IoT SiteWise」を参照してください。

> **Note**
>
> アセットは `ACTIVE` モデルからのみ作成できます。モデルの状態が `ACTIVE` でない場合は、そのモデルからアセットを作成できるようになるまで数分待たなければならない場合があります。詳細については、「アセットおよびモデルの状態」を参照してください。

トピック

- アセットを作成する (コンソール)
- アセットを作成する (AWS CLI)
- 新しいアセットを設定する

アセットを作成する (コンソール)

AWS IoT SiteWise コンソールを使用してアセットを作成できます。

アセットを作成するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. [Create asset (アセットの作成)] を選択します。

4. [アセットの作成] ページで、次の操作を実行します。

   a. [モデル] で、アセットを作成するアセットモデルを選択します。

   > **Note**
   > モデルが [アクティブ] でない場合は、アクティブになるまで待つか、[FAILED (失敗)] の場合は問題を解決する必要があります。

   b. アセットの [名前] を入力します。

   c. (オプション) アセットにタグを追加します。詳細については、「AWS IoT SiteWise リソースにタグを付ける」を参照してください。

   d. [Create asset (アセットの作成)] を選択します。

   アセットを作成すると、 AWS IoT SiteWise コンソールは新しいアセットのページに移動します。このページでは、アセットの [ステータス] が表示されます。このステータスは、最初は [作成中] です。このページは自動的に更新されるため、アセットのステータスが更新されるまで待つことができます。

   > **Note**
   > アセットの作成プロセスには、最大で 1 分かかる場合があります。[Status] (状態) が [ACTIVE] (アクティブ) になったら、アセットに対して更新オペレーションを実行できます。詳細については、「アセットおよびモデルの状態」を参照してください。

アセットを作成したら、「新しいアセットを設定する」を参照してください。

アセットを作成する (AWS CLI)

AWS Command Line Interface（ AWS CLI) を使用して、アセットモデルからアセットを作成できます。

アセットを作成するには `assetModelId` が必要です。アセットモデルを作成したが、その `assetModelId` が不明な場合は、ListAssetModels API を使用して、すべてのアセットモデルを表示できます。

アセットモデルからアセットを作成するには、以下のパラメータを指定して CreateAsset API を使用します。

- `assetName` - 新しいアセットの名前。アセットを識別するのに役立つ名前をアセットに付けます。
- `assetModelId` – アセットの ID。これは UUID 形式の実際の ID です。または、ある場合は `externalId:myExternalId` です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

アセットを作成するには (AWS CLI)

- 次のコマンドを実行して、アセットを作成します。*asset-name* をアセットの名前に置き換え、*asset-model-id* をアセットモデルの ID または外部 ID に置き換えます。

```
aws iotsitewise create-asset \
  --asset-name asset-name \
  --asset-model-id asset-model-id
```

このオペレーションは、新しいアセットの詳細およびステータスを含むレスポンスを次の形式で返します。

```
{
  "assetId": "String",
  "assetArn": "String",
  "assetStatus": {
    "state": "String",
    "error": {
      "code": "String",
      "message": "String"
    }
  }
}
```

アセットの `state` は、アセットが作成されるまでは `CREATING` です。

> **Note**
>
> アセットの作成プロセスには、最大で 1 分かかる場合があります。アセット状態をチェックするには、`assetId` パラメータにアセット ID を指定して DescribeAsset オペ

レーションを使用します。アセットの state が ACTIVE になったら、アセットに対して更新オペレーションを実行できます。詳細については、「アセットおよびモデルの状態」を参照してください。

アセットを作成したら、「新しいアセットを設定する」を参照してください。

新しいアセットを設定する

でアセットを作成した後 AWS IoT SiteWise、アセットとそのデータを最大限に活用するために実行できる次のステップがいくつかあります。これらのステップには、アセットからデータを取り込むようにデータストリームを設定する、アセットのパフォーマンスをモニタリングするためのアラームと通知を設定する、アセットのデータを表示する視覚化とダッシュボードを作成する、さらに分析や自動化を行うためにアセットを他の AWS サービスやサードパーティーアプリケーションと統合するなどが含まれます。

以下のオプションアクションを使用してアセットの設定を完了します。

- アセットに測定プロパティがある場合、のデータストリームを管理する AWS IoT SiteWise。
- アセットに一意の属性値がある場合、属性値を更新する。
- アセットが親アセットの場合、アセットの関連付けと関連付け解除。

でアセットを検索する AWS IoT SiteWise コンソール

AWS IoT SiteWise コンソール 検索機能を使用して、メタデータとリアルタイムのプロパティ値フィルターに基づいてアセットを検索します。

前提条件

AWS IoT SiteWise では、産業データをより適切に整理およびモデル AWS IoT TwinMaker 化するために、 と統合するためのアクセス許可が必要です。アクセス許可を付与している場合は AWS IoT SiteWise、ExecuteQuery API を使用します。アクセス許可を付与しておらず AWS IoT SiteWise、開始にあたってサポートが必要な場合は、「」を参照してくださいAWS IoT SiteWise と の統合 AWS IoT TwinMaker。

の詳細検索 AWS IoT SiteWise コンソール

メタデータ検索

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[アセット] の [高度な検索] を選択します。
3. [高度な検索] の [メタデータ検索] オプションを選択します。
4. パラメータを入力します。効率的な検索のために、できるだけ多くのフィールドに入力します。
   a. [アセット名] - 完全なアセット名、または広範囲検索用の部分的な名前を入力します。
   b. [プロパティ名] - 完全なプロパティ名、または広範囲検索用の部分的な名前を入力します。
   c. [演算子] — 演算子を以下から選択します。
      - =
      - <
      - >
      - <=
      - >=
   d. [プロパティ値] — この値は、プロパティの最新値と比較されます。
   e. [プロパティ値のタイプ] — プロパティのデータ型。次から選択します。
      - 倍精度
      - 整数
      - 文字列
      - ブール値
5. [検索] を選択してください。
6. [検索結果] テーブルから、[名前] 列からアセットを選択します。そのアセットの詳細アセットページが表示されます。

部分検索

アセット検索には、すべてのパラメータを指定する必要はありません。メタデータ検索オプションを使用した部分検索の例をいくつか示します。

- 名前でアセットを検索します。
  - [アセット名] フィールドのみに値を入力します。
  - [プロパティ名] および [プロパティ値] フィールドは空です。
- 特定の名前を持つプロパティを含むアセットを検索します。
  - [プロパティ名] フィールドのみに値を入力します。
  - [アセット名] および [プロパティ値] フィールドは空です。
- プロパティの最新の値に基づいてアセットを検索します。
  - [プロパティ名] および [プロパティ値] フィールドに値を入力します。
  - [演算子] と [プロパティ値のタイプ] を選択します。

クエリビルダー検索

1. AWS IoT SiteWise コンソールに移動します。

2. ナビゲーションペインで、[アセット] の [高度な検索] を選択します。
3. [高度な検索] で、[クエリビルダー] オプションを選択します。
4. [クエリビルダー] ペインで、`asset_name`、`asset_id`、および `asset_description` を取得する SQL クエリを書き込みます。
5. [検索] を選択してください。
6. [検索結果] テーブルから、[名前] 列からアセットを選択します。そのアセットの詳細アセットページが表示されます。

> **Note**
>
> - SQL クエリの `SELECT` 句には、[検索結果] テーブルで有効なアセットを確保するために、`asset_name` および `asset_id` フィールドを含める必要があります。
> - クエリビルダーは、結果テーブルに [名前]、[アセット ID]、[説明] のみを表示します。`SELECT` 句にフィールドを追加しても、結果テーブルに列は追加されません。

属性値を更新する

アセットは、属性のデフォルト値を含むアセットモデルの属性を継承します。場合によっては、アセットメーカーのプロパティなど、アセットモデルのデフォルト属性をそのままにしておくこともできます。また、アセットの緯度と経度など、継承された属性を更新することもできます。

Updating an attribute value (console)

AWS IoT SiteWise コンソールを使用して、属性アセットプロパティの値を更新できます。

属性の値を更新するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. 属性を更新するアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [編集] を選択します。
5. 更新する属性を検索し、新しい値を入力します。

6. [保存] を選択します。

Updating an attribute value (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用して属性値を更新できます。

この手順を完了するには、アセットの `assetId` とプロパティの `propertyId` を知っている必要があります。外部 ID を使用することもできます。アセットを作成したが、その `assetId` がわからない場合は、ListAssets API を使用して、特定のモデルのすべてのアセットを一覧表示しま

す。DescribeAsset オペレーションを使用して、プロパティ ID を含むアセットのプロパティを表示します。

BatchPutAssetPropertyValue オペレーションを使用して、アセットに属性値を割り当てます。このオペレーションを使用すると、複数の属性を一度に設定できます。このオペレーションのペイロードにはエントリのリストが含まれ、各エントリにはアセット ID、プロパティ ID、属性値が含まれます。

属性の値を更新するには (AWS CLI)

1. `batch-put-payload.json` という名前のファイルを作成して、次の JSON オブジェクトをファイルにコピーします。このペイロードの例は、風力タービンの緯度と経度を設定する方法を示しています。ID、値、タイムスタンプを更新して、ユースケースのペイロードを変更します。

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

- ペイロードの各エントリには、一意の文字列として定義できる `entryId` が含まれています。リクエストのエントリが失敗した場合、各エラーには、対応するリクエストの `entryId` が含まれるため、再試行するリクエストを確認できます。
- 属性値を設定するには、各属性プロパティの `propertyValues` のリストに 1 つのタイムスタンプ品質値 (TQV) 構造を含めることができます。この構造には、新しい `value` および現在の `timestamp` を含める必要があります。
  - `value` - 設定中のプロパティの型に応じて、次のいずれかのフィールドを含む構造。
    - `booleanValue`
    - `doubleValue`
    - `integerValue`
    - `stringValue`
    - `nullValue`
  - `timestamp` – 現在の Unix エポック時間を秒単位で含む構造`timeInSeconds`。. は、過去 7 日以上、または今後 5 分より新しいタイムスタンプを持つデータポイント AWS IoT SiteWise を拒否します。

BatchPutAssetPropertyValue の ペイロードを準備する方法の詳細については、AWS IoT SiteWise APIs を使用してデータを取り込む を参照してください。

2. 次のコマンドを実行して、属性値を AWS IoT SiteWiseに送信します。

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

アセットの関連付けと関連付け解除

アセットのモデルで子アセットモデルの階層が定義されている場合は、子アセットをアセットに関連付けることができます。親アセットは、関連するアセットのデータにアクセスし、集計できます。階層アセットモデルの詳細については、「アセットモデルの階層を定義する」を参照してください。

トピック

- アセットの関連付けと関連付け解除 (コンソール)
- アセットの関連付けと関連付け解除 (AWS CLI)

アセットの関連付けと関連付け解除 (コンソール)

AWS IoT SiteWise コンソールを使用して、アセットの関連付けと関連付け解除を行うことができます。

アセットを関連付けるには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. 子アセットを関連付ける親アセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [編集] を選択します。
5. [このアセットに関連付けられているアセット] で、[関連付けられたアセットの追加] を選択します。

6. [階層] で、親アセットと子アセットの関係を定義する階層を選択します。
7. [アセット] で、関連付ける子アセットを選択します。
8. [保存] を選択します。

アセットの関連付けを解除するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. 子アセットの関連付けを解除する親アセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [編集] を選択します。
5. [このアセットに関連付けられているアセット] の [関連付けを削除] を選択します。

6. [保存] を選択します。

アセットの関連付けと関連付け解除 (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用して、アセットの関連付けと関連付け解除を行うことができます。

このステップでは、子アセットモデルとの関係を定義する親アセットモデル内の階層 (`hierarchyId`) の ID を知る必要があります。DescribeAsset オペレーションを使用して、レスポンス内の階層 ID を検索します。

階層 ID を検索するには

- 次のコマンドを実行して、親アセットを記述します。*parent-asset-id* を親アセットの ID または外部 ID に置き換えます。

```
aws iotsitewise describe-asset --asset-id parent-asset-id
```

このオペレーションは、アセットの詳細を含むレスポンスを返します。レスポンスには、次の構造を持つ assetHierarchies リストが含まれています。

```
{
  ...
  "assetHierarchies": [
    {
      "id": "String",
      "name": "String"
    }
  ],
  ...
}
```

階層 ID は、アセット階層のリスト内の階層の id の値です。

階層 ID を取得したら、アセットをその階層に関連付けるか、関連付けを解除できます。

子アセットを親アセットに関連付けるには、AssociateAssets オペレーションを使用します。子アセットを親アセットとの関連付けから解除するには、DisassociateAssets オペレーションを使用します。次のパラメータを指定します。これらのパラメータは、両方のオペレーションで同じです。

- assetId – 親アセットの ID または外部 ID。
- hierarchyId - 親アセットの階層 ID または外部 ID。
- childAssetId – 子アセットの ID または外部 ID。

アセットを関連付けるには (AWS CLI)

- 子アセットを親アセットに関連付けるには、次のコマンドを実行します。*parent-asset-id*、*hierarchy-id*、および *child-asset-id* をそれぞれの ID に置き換えます。

```
aws iotsitewise associate-assets \
```

```
  --asset-id parent-asset-id \
  --hierarchy-id hierarchy-id \
  --child-asset-id child-asset-id
```

アセットの関連付けを解除するには (AWS CLI)

- 子アセットを親アセットとの関連付けから解除するには、次のコマンドを実行します。*parent-asset-id*、*hierarchy-id*、および *child-asset-id* をそれぞれの ID に置き換えます。

```
aws iotsitewise disassociate-assets \
  --asset-id parent-asset-id \
  --hierarchy-id hierarchy-id \
  --child-asset-id child-asset-id
```

アセットとモデルを更新する

でアセット、アセットモデル、コンポーネントモデルを更新 AWS IoT SiteWise して、名前と定義を変更できます。これらの更新オペレーションは非同期であり、伝達に時間がかかります AWS IoT SiteWise。追加の変更を行う前に、アセットまたはアセットモデルのステータスを確認します。更新されたアセットまたはモデルを引き続き使用するには、変更が反映されるまで待機する必要があります。

トピック

- でアセットを更新する AWS IoT SiteWise
- アセットモデルとコンポーネントモデルを更新する
- カスタム複合モデルを更新する (コンポーネント)
- アセットモデル書き込みの楽観的ロック

でアセットを更新する AWS IoT SiteWise

AWS IoT SiteWise コンソールまたは API を使用して、アセットの名前を更新できます。

アセットを更新すると、変更が反映されるまで、アセットの状態は `UPDATING` になります。詳細については、「アセットおよびモデルの状態」を参照してください。

トピック

- アセットを更新する (コンソール)
- アセットを更新する (AWS CLI)

アセットを更新する (コンソール)

AWS IoT SiteWise コンソールを使用してアセットの詳細を更新できます。

アセットを更新するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. 更新するアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [編集] を選択します。
5. アセットの [名前] を更新します。
6. (オプション) このページで、アセットのその他の情報を更新します。詳細については次を参照してください:
   - のデータストリームを管理する AWS IoT SiteWise
   - 属性値を更新する
   - 他の AWS サービスとやり取りする
7. [保存] を選択します。

アセットを更新する (AWS CLI)

AWS Command Line Interface（AWS CLI) を使用して、アセットの名前を更新できます。

アセットを更新するには、UpdateAsset オペレーションを使用します。以下のパラメータを指定します。

- `assetId` – アセットの ID。これは UUID 形式の実際の ID です。または、ある場合は `externalId:myExternalId` です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。
- `assetName` - アセットの新しい名前。

アセットの名前を更新するには (AWS CLI)

- アセットの名前を更新するには、次のコマンドを実行します。*asset-id* をアセットの ID または外部 ID に置き換えます。*asset-name* をアセットの新しい名前で更新します。

```
aws iotsitewise update-asset \
  --asset-id asset-id \
  --asset-name asset-name
```

アセットモデルとコンポーネントモデルを更新する

AWS IoT SiteWise コンソールまたは API を使用して、アセットモデルまたはコンポーネントモデルを更新します。

既存のプロパティのタイプまたはデータ型、あるいは既存のメトリックのウィンドウは変更できません。また、モデルのタイプをアセットモデルからコンポーネントモデル、または逆方向に変更することはできません。

> **Important**
>
> - アセットモデルまたはコンポーネントモデルからプロパティを削除すると、 はそのプロパティの以前のデータをすべて AWS IoT SiteWise 削除します。コンポーネントモデルの場合、これはそのコンポーネントモデルを使用するすべてのアセットモデルに影響するため、変更がどの程度の範囲に適用される可能性があるかを理解するように特に注意してください。
> - アセットモデルから階層定義を削除すると、 はその階層内のすべてのアセットの関連付け AWS IoT SiteWise を解除します。

アセットモデルを更新すると、基になるモデルに加えた変更がそのモデルに基づくすべてのアセットに反映されます。変更が反映されるまで、各アセットには `UPDATING` 状態になります。それらのア

セットを操作する前に、それらが ACTIVE 状態に戻るまで待機する必要があります。この間、更新されたアセットモデルのステータスは PROPAGATING になります。

コンポーネントモデルを更新すると、そのコンポーネントモデルを組み込んだすべてのアセットモデルに変更が反映されます。コンポーネントモデルの変更が伝播されるまで、影響を受ける各アセットモデルは UPDATING 状態になり、その後、PROPAGATING になり、前の段落で説明したように、関連するアセットを更新します。それらのアセットを操作する前に、それらが ACTIVE 状態に戻るまで待機する必要があります。この間、更新されたコンポーネントモデルのステータスは PROPAGATING になります。

詳細については、「アセットおよびモデルの状態」を参照してください。

トピック

- アセットまたはコンポーネントモデルの更新 (コンソール)
- アセットまたはコンポーネントモデルを更新する (AWS CLI)

アセットまたはコンポーネントモデルの更新 (コンソール)

AWS IoT SiteWise コンソールを使用して、アセットモデルまたはコンポーネントモデルを更新できます。

アセットモデルまたはコンポーネントモデルを更新するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. 更新するアセットモデルまたはコンポーネントモデルを選択します。
4. [編集] を選択します。
5. [モデルの編集] ページで、次のいずれかの操作を行います。
    - [モデルの詳細] で、モデルの [名前] を変更します。
    - 任意の [属性定義] を変更します。既存の属性の [データ型] は変更できません。詳細については、「静的データ (属性) を定義する」を参照してください。
    - [測定の定義] を変更します。既存の測定値の [データ型] は変更できません。詳細については、「機器からのデータストリームを定義する (測定値) 」を参照してください。
    - 任意の [定義を変換する] を変更します。詳細については、「データを変換する (変換) 」を参照してください。

- [メトリクスの定義] を変更します。既存のメトリクスの [時間間隔] は変更できません。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。
- (アセットモデルのみ) 任意の [階層の定義] を変更します。既存の階層の [階層モデル] を変更することはできません。詳細については、「アセットモデルの階層を定義する」を参照してください。

6. [保存] を選択します。

> **Note**
>
> 最後に [モデルを編集する] ページを開いてから別のユーザーがアセットモデルを正常に更新した場合、コンソールで行われた更新リクエストは拒否されます。コンソールは、[モデルを編集する] ページを [最新の状態に更新] して、更新されたモデルを読み込むようにユーザーを促します。更新を再度行い、保存を再試行する必要があります。詳細については、「アセットモデル書き込みの楽観的ロック」を参照してください。

アセットまたはコンポーネントモデルを更新する (AWS CLI)

AWS Command Line Interface（AWS CLI) を使用して、アセットモデルまたはコンポーネントモデルを更新します。

UpdateAssetModel API を使用して、アセットモデルまたはコンポーネントモデルの名前、説明、およびプロパティを更新します。アセットモデルの場合のみ、階層を更新できます。以下のパラメータを指定します。

- `assetModelId` – アセットの ID。これは UUID 形式の実際の ID です。または、ある場合は `externalId:myExternalId` です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

ペイロードで更新されたアセットモデルを指定します。アセットモデルまたはコンポーネントモデルの予想される形式については、「でアセットモデルを作成する AWS IoT SiteWise」を参照してください。

> **Warning**
>
> UpdateAssetModel API は、ペイロードで指定したモデルによって既存のモデルを上書きします。モデルのプロパティまたは階層を削除しないようにするには、更新されたアセットモデルのペイロードに ID と定義を含める必要があります。モデルの既存の構造をクエリする方法については、「DescribeAssetModel オペレーション」を参照してください。

> **Note**
>
> 次の手順では、タイプ AWS/ALARM の複合モデルのみを更新できます。CUSTOM 複合モデルを更新する場合は、代わりに UpdateAssetModelCompositeModel を使用します。詳細については、「カスタム複合モデルを更新する (コンポーネント)」を参照してください。

アセットモデルまたはコンポーネントモデルを更新するには (AWS CLI)

1. 次のコマンドを実行して、既存のアセットモデル定義を取得します。*asset-model-id* を、更新するアセットモデルまたはコンポーネントモデルの ID または外部 ID に置き換えます。

   ```
   aws iotsitewise describe-asset-model --asset-model-id asset-model-id
   ```

   上記のコマンドは、モデルの最新バージョンに対応するモデル定義を返します。

   アセットモデルが FAILED 状態にあるユースケースでは、アクティブなバージョンに対応する有効なモデル定義を取得して、更新リクエストを構築します。詳細については、「アセットモデルのバージョン」を参照してください。次のコマンドを実行して、アクティブなモデル定義を取得します。

   ```
   aws iotsitewise describe-asset-model --asset-model-id asset-model-id --asset-model-
   version ACTIVE
   ```

   このオペレーションは、モデルの詳細を含むレスポンスを返します。このレスポンスには以下の構造が含まれます。

   ```
   {
       "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"
}
```

詳細については、「DescribeAssetModel オペレーション」を参照してください。

2. update-asset-model.json という名前のファイルを作成し、前のコマンドのレスポンスをファイルにコピーします。
3. update-asset-model.json の JSON オブジェクトから次のキーと値のペアを削除します。

   - assetModelId
   - assetModelArn
   - assetModelCompositeModelSummaries
   - assetModelCreationDate
   - assetModelLastUpdateDate
   - assetModelStatus
   - assetModelType
   - assetModelVersion
   - eTag

   UpdateAssetModel オペレーションでは、次の構造を持つペイロードが必要です。

```
{
```

```
  "assetModelName": "String",
  "assetModelDescription": "String",
  "assetModelProperties": Array of AssetModelProperty,
  "assetModelHierarchies": Array of AssetModelHierarchyDefinition,
  "assetModelCompositeModels": Array of AssetModelCompositeModel
}
```

4. update-asset-model.json で、次のいずれかを行ってください。
   - アセットモデルの名前を変更します (assetModelName)。
   - アセットモデルの説明を変更、追加、または削除します (assetModelDescription)。
   - アセットモデルのプロパティを変更、追加、または削除します (assetModelProperties)。既存のプロパティの dataType、または既存のメトリクスの window を変更することはできません。詳細については、「データのプロパティを定義する」を参照してください。
   - アセットモデルの階層を変更、追加、または削除します (assetModelHierarchies)。既存の階層の childAssetModelId は変更できません。詳細については、「アセットモデルの階層を定義する」を参照してください。
   - アセットモデルのタイプ AWS/ALARM (assetModelCompositeModels) の複合モデルを変更、追加、または削除します。アラームは、他のプロパティをモニタリングし、機器やプロセスに注意が必要な時期を特定することができます。各アラーム定義は、アラームが使用する一連のプロパティを標準化した複合モデルです。詳細については、「でアラームを使用してデータをモニタリングする AWS IoT SiteWise」および「でアセットモデルにアラームを定義する AWS IoT SiteWise」を参照してください。
5. 次のコマンドを実行して、update-asset-model.json に保存されている定義でアセットモデルを更新します。*asset-model-id* をアセットモデルの ID に置き換えます。

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://model-payload.json
```

> **Important**
>
> 複数のユーザーがアセットモデルを同時に更新すると、ユーザーの変更が別のユーザーによって誤って上書きされる可能性があります。これを防ぐには、条件付き更新リクエストを定義する必要があります。「アセットモデル書き込みの楽観的ロック」を参照してください。

カスタム複合モデルを更新する (コンポーネント)

AWS IoT SiteWise API を使用してカスタム複合モデルを更新するか、 AWS IoT SiteWise コンソールを使用してコンポーネントを更新できます。

トピック

- コンポーネントを更新する (コンソール)
- カスタム複合モデルを更新する (AWS CLI)

コンポーネントを更新する (コンソール)

AWS IoT SiteWise コンソールを使用してコンポーネントを更新できます。

コンポーネントを更新するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. コンポーネントがあるアセットモデルを選択します。
4. [プロパティ] タブで、[コンポーネント] を選択します。
5. 更新するコンポーネントを選択します。
6. [編集] を選択します。
7. [コンポーネントを編集] ページで、次のいずれかの操作を行います。
   - [モデルの詳細] で、モデルの [名前] を変更します。
   - 任意の [属性定義] を変更します。既存の属性の [データ型] は変更できません。詳細については、「静的データ (属性) を定義する」を参照してください。
   - [測定の定義] を変更します。既存の測定値の [データ型] は変更できません。詳細については、「機器からのデータストリームを定義する (測定値) 」を参照してください。
   - 任意の [定義を変換する] を変更します。詳細については、「データを変換する (変換) 」を参照してください。
   - [メトリクスの定義] を変更します。既存のメトリクスの [時間間隔] は変更できません。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。
8. [保存] を選択します。

カスタム複合モデルを更新する (AWS CLI)

AWS Command Line Interface（ AWS CLI) を使用して、カスタム複合モデルを更新します。

名前または説明を更新するには、UpdateAssetModelCompositeModel オペレーションを使用します。インラインカスタム複合モデルのみ、プロパティも更新できます。コンポーネントモデルベースのカスタム複合モデルのプロパティは、参照されるコンポーネントモデルが関連するプロパティを提供するため、更新できません。

> **Important**
>
> カスタム複合モデルからプロパティを削除すると、 はそのプロパティの以前のデータをすべて AWS IoT SiteWise 削除します。既存のプロパティのタイプまたはデータ型は変更できません。
>
> 既存の複合モデルプロパティを同じ `name` を持つ新しいプロパティに置き換えるには、以下を実行します。
>
> 1. 既存のプロパティ全体を削除して `UpdateAssetModelCompositeModel` リクエストを送信します。
> 2. 新しいプロパティを含む 2 番目の `UpdateAssetModelCompositeModel` リクエストを送信します。新しいアセットプロパティは、前のアセットプロパティ`name`と同じ を持ち AWS IoT SiteWise 、新しい一意の を生成します`id`。

カスタム複合モデルを更新するには (AWS CLI)

1. 既存の複合モデル定義を取得するには、次のコマンドを実行します。*`composite-model-id`* を更新対象のカスタム複合モデルの ID または外部 ID に置き換え、 *`asset-model-id`* をカスタム複合モデルが関連付けられているアセットモデルに置き換えます。詳細については、「AWS IoT SiteWise ユーザーガイド」を参照してください。

   a. 以下のコマンドを実行します。

   ```
   aws iotsitewise describe-asset-model-composite-model \
   --asset-model-composite-model-id composite-model-id \
   --asset-model-id asset-model-id
   ```

   b. 上記のコマンドは、関連付けられたモデルの最新バージョンに対応する複合モデル定義を返します。アセットモデルが `FAILED` 状態にあるユースケースでは、アクティブなバージョ

ンに対応する有効なモデル定義を取得して、更新リクエストを構築します。詳細については、「アセットモデルのバージョン」を参照してください。

c. 次のコマンドを実行して、アクティブなモデル定義を取得します。

```
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. 詳細については、DescribeAssetModelCompositeModel オペレーションを参照してください。

2. `update-custom-composite-model.json` という名前のファイルを作成し、前のコマンドのレスポンスをファイルにコピーします。

3. `update-custom-composite-model.json` の JSON オブジェクトから、以下のフィールドを除くすべてのキーと値のペアを削除します。

   - `assetModelCompositeModelName`
   - `assetModelCompositeModelDescription` (存在する場合)
   - `assetModelCompositeModelProperties` (存在する場合)

4. `update-custom-composite-model.json` で、次のいずれかを行ってください。

   - `assetModelCompositeModelName` の値を変更します。
   - `assetModelCompositeModelDescription` を追加または削除するか、その値を変更します。
   - インラインカスタム複合モデルのみ: `assetModelCompositeModelProperties` のアセットモデルのプロパティを変更、追加、または削除します。

   このファイルに必要な形式の詳細については、UpdateAssetModelCompositeModel のリクエスト構文を参照してください。

5. 次のコマンドを実行して、`update-custom-composite-model.json` に保存されている定義でカスタム複合モデルを更新します。*composite-model-id* を複合モデルの ID に、*asset-model-id* をその中のアセットモデルの ID に置き換えます。

```
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**
>
> 複数のユーザーがアセットモデルを同時に更新すると、ユーザーの変更が別のユーザーによって誤って上書きされる可能性があります。これを防ぐには、条件付き更新リクエストを定義する必要があります。「アセットモデル書き込みの楽観的ロック」を参照してください。

アセットモデル書き込みの楽観的ロック

アセットモデルを更新する場合、ユーザーは以下を実行します。

1. 現在のアセットモデル定義を読みます。
2. 必要な変更を加えてアセットモデル定義を編集します。
3. アセットモデルを新しい定義で更新します。

2 人のユーザーがモデルを更新するシナリオでは、次のようになる可能性があります。

- ユーザー A はアセットモデル X 定義を読み込みます。
- ユーザー B はアセットモデル X 定義を読み取って変更をコミットし、X の定義を変更します。
- ユーザー A は、ユーザー B の変更を確認したり、組み込んだりせずに、アセットモデル X についてユーザー B によって行われた変更をコミットして上書きします。

オプティミスティックロックは、上記のシナリオのような偶発的な上書きを防ぐために AWS IoT SiteWise で使用されるメカニズムです。楽観的ロックは、更新または削除されるアセットモデルの現在のバージョンが、 AWS IoT SiteWiseの現在のバージョンと同じであることを確認する戦略です。これにより、アセットモデルの書き込みが誤って更新されて上書きされないように保護されます。

楽観的ロックを使用して、アセットモデルの書き込みを実行するには、次の手順に従います。

トピック

- 楽観的ロックを使用した、アセットモデルの書き込みの実行 (コンソール)
- 楽観的ロックを使用したアセットモデルの書き込みの実行 (AWS CLI)

楽観的ロックを使用した、アセットモデルの書き込みの実行 (コンソール)

次の手順では、コンソールでアセットモデルのアクティブバージョンに対して楽観的ロックを使用してアセットモデルの書き込みを実行する方法について説明します。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. 更新するアセットモデルまたはコンポーネントモデルを選択します。
4. [編集] を選択します。
5. [モデルの編集] ページで変更を加えます。
6. [保存] を選択します。

> **Note**
>
> 場合によっては、ユーザーがモデルの編集を開始してから、モデルに加えた編集を保存するまでの間に、1 つ以上のモデル更新が成功することがあります。
> ユーザーが誤って新しい成功した更新を上書きしないように、ユーザーの書き込みは拒否されます。コンソールは [保存] ボタンを無効にし、[モデルの編集] ページを更新するようにユーザーに求めます。ユーザーは、モデルの新しいアクティブバージョンを再度更新する必要があります。ユーザーは、以下の追加ステップを実行する必要があります。

7. [更新] を選択します。
8. ステップ 5 と 6 を再度実行します。

楽観的ロックを使用したアセットモデルの書き込みの実行 (AWS CLI)

次の手順では、 AWS CLIで楽観的ロックを使用してアセットモデルの書き込みを行う方法について説明します。

1. 現在のモデル定義に関連付けられた ETag を取得する

   ETag は、アセットモデルの新しい表現ごとに生成される一意のトークンです。DescribeAssetModel API を呼び出して、レスポンスから現在のアセットモデル定義と、関連付けられた ETag を取得します。

同時更新中、ユーザーは成功した更新 (ACTIVE 状態のモデル) または失敗した更新 (FAILED 状態のモデル) を実行します。ユーザーが成功した更新を誤って上書きしないようにするには、アセットモデルのバージョン からアセットモデルのアクティブバージョンを取得し、ETag 値を取得する必要があります。

次のコマンドを実行してください。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id \
--asset-model-version ACTIVE
```

レスポンスは以下の構造を返します。

```
{
  "assetModelId": "String",
  "assetModelArn": "String",
  "assetModelName": "String",
  ...
  "eTag": "String"
}
```

> **Note**
>
> 更新を上書きしないように、アセットモデルの最新バージョンとその ETag を取得する必要があります。

2. 書き込み条件を指定して、UPDATE および DELETE オペレーションを実行する

以下のアセットモデル API は、楽観的ロックをサポートしています。

- UpdateAssetModel
- DeleteAssetModel
- CreateAssetModelCompositeModel
- UpdateAssetModelCompositeModel
- DeleteAssetModelCompositeModel

> **Note**
>
> 下記のシナリオでは、UpdateAssetModel API をリファレンスとして使用します。条件は、上記のすべてのオペレーションに適用されます。

下記のシナリオでは、同時実行制御要件に応じて異なる書き込み条件について説明します。

- 成功した更新を上書きしないように、次のコマンドを実行します。最後の読み取りアクティブバージョン以降、新しいアクティブバージョンが存在してはいけません。e-tag を、アクティブバージョンの読み取りで使用される API オペレーションで返された ETag に置き換えます。

```
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
```

- モデルの作成が失敗すると、FAILED 状態のため、アクティブバージョンはまだ存在しません。変更がコミットされるまでは、存在する新しいアクティブバージョンを上書きすることは可能です。最後の読み取り時にアクティブバージョンが存在しない場合、新しいアクティブバージョンを上書きしないように、次のコマンドを実行します。

```
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
```

- 次のコマンドを実行して、成功または失敗した更新を上書きしないようにします。このコマンドは、最後の最新バージョンの読み取り以降に最新バージョンが作成されないようにする書き込み条件を定義します。e-tag を、アクティブバージョンの読み取りで使用される API オペレーションで返された ETag に置き換えます。

```
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
```

書き込み条件が FALSE に評価された場合、書き込みリクエストは PreconditionFailedException で失敗します。

でアセットとモデルを削除する AWS IoT SiteWise

アセットとモデルは、完了 AWS IoT SiteWise 時に から削除できます。削除オペレーションは非同期であり、伝達に時間がかかります AWS IoT SiteWise。

トピック

- でアセットを削除する AWS IoT SiteWise
- でアセットモデルを削除する AWS IoT SiteWise

でアセットを削除する AWS IoT SiteWise

AWS IoT SiteWise コンソールまたは API を使用して、環境に不要になったアセットを削除できます。アセットモデルを削除すると、関連するアセットとコンポーネントモデルもすべて削除されます。ただし、アセットまたはモデルの削除は永続的なアクションであり、削除されたリソースに関連付けられたデータも削除されることに注意してください。アセットやモデルを削除する前に、影響を受ける可能性のある依存関係や統合を確認し、重要なデータのバックアップがあることを確認することをお勧めします。

アセットを削除する前に、まずその子アセットの関連付けを解除して、親アセットとの関連付けを解除する必要があります。詳細については、「アセットの関連付けと関連付け解除」を参照してください。 AWS Command Line Interface（AWS CLI) を使用する場合は、ListAssociatedAssets オペレーションを使用してアセットの子を一覧表示できます。

アセットを削除すると、変更が反映されるまで、アセットのステータスは DELETING になります。詳細については、「アセットおよびモデルの状態」を参照してください。アセットを削除した後は、そのアセットに対してクエリを実行することはできません。使用すると、API は HTTP 404 レスポンスを返します。

> **Important**
> AWS IoT SiteWise は、削除されたアセットのすべてのプロパティデータを削除します。

トピック

- アセットを削除する (コンソール)
- アセットを削除する (AWS CLI)

アセットを削除する (コンソール)

AWS IoT SiteWise コンソールを使用してアセットを削除できます。

アセットを削除するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. 削除するアセットを選択します。

   > **Tip**
   > 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. アセットに [関連付けられたアセット] がある場合は、各アセットを削除します。アセットの名前を選択して、そのページに移動して、アセットを削除できます。
5. アセットのページで、[削除] を選択します。
6.]アセットの削除] ダイアログで、次の操作を行います。
   a. 削除を確定するには、**`Delete`** と入力します。
   b. [削除] を選択します。

アセットを削除する (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用してアセットを削除できます。

アセットを削除するには、DeleteAsset オペレーションを使用します。以下のパラメータを指定します。

- `assetId` – アセットの ID。これは UUID 形式の実際の ID です。または、ある場合は `externalId:myExternalId` です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

アセットを削除するには (AWS CLI)

1. 次のコマンドを実行して、アセットの階層を一覧表示します。*asset-id* をアセットの ID または外部 ID に置き換えます。

```
aws iotsitewise describe-asset --asset-id asset-id
```

このオペレーションは、アセットの詳細を含むレスポンスを返します。レスポンスには、次の構造を持つ `assetHierarchies` リストが含まれています。

```
{
  ...
  "assetHierarchies": [
    {
      "id": "String",
      "name": "String"
    }
  ],
  ...
}
```

詳細については、DescribeAsset オペレーションを参照してください。

2. 階層ごとに次のコマンドを実行して、その階層に関連付けられているアセットの子を一覧表示します。*asset-id* をアセットの ID または外部 ID に置き換え、*hierarchy-id* を階層の ID または外部 ID に置き換えます。

```
aws iotsitewise list-associated-assets \
  --asset-id asset-id \
  --hierarchy-id hierarchy-id
```

詳細については、「ListAssociatedAssets オペレーション」を参照してください。

3. 次のコマンドを実行して、関連する各アセットを削除してから、アセットを削除します。*asset-id* をアセットの ID または外部 ID に置き換えます。

```
aws iotsitewise delete-asset --asset-id asset-id
```

でアセットモデルを削除する AWS IoT SiteWise

AWS IoT SiteWise コンソールまたは API を使用して、アセットモデルを削除できます。

アセットモデルを削除する前に、まずアセットモデルから作成されたすべてのアセットを削除する必要があります。

アセットモデルを削除すると、変更が反映されるまで、アセットのステータスは DELETING になります。詳細については、「アセットおよびモデルの状態」を参照してください。アセットモデルを削除した後は、そのアセットモデルをクエリすることはできません。使用すると、API は HTTP 404 レスポンスを返します。

トピック

- アセットモデルを削除する (コンソール)
- アセットモデルを削除する (AWS CLI)

アセットモデルを削除する (コンソール)

AWS IoT SiteWise コンソールを使用してアセットモデルを削除できます。

アセットモデルを削除するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Models (モデル)] を選択します。
3. 削除するアセットモデルを選択します。
4. モデルに [アセット] がある場合は、各アセットを削除します。アセットの名前を選択してそのページに移動し、そこでアセットを削除できます。詳細については、「アセットを削除する (コンソール) 」を参照してください。
5. モデルのページで、[削除] を選択します。
6. [モデルを削除] ダイアログで、次の操作を行います。
   a. 削除を確定するには、**Delete** と入力します。
   b. [削除] を選択します。

アセットモデルを削除する (AWS CLI)

AWS Command Line Interface（AWS CLI) を使用してアセットモデルを削除できます。

アセットモデルを削除するには、「DeleteAssetModel オペレーション」を使用します。以下のパラメータを指定します。

- `assetModelId` – アセットの ID。これは UUID 形式の実際の ID です。または、ある場合は `externalId:myExternalId` です。詳細については、AWS IoT SiteWise ユーザーガイドの 外部 ID を持つリファレンスオブジェクトを参照してください。

アセットモデルを削除するには (AWS CLI)

1. 次のコマンドを実行して、モデルから作成されたすべてのアセットを一覧表示します。*asset-model-id* をアセットモデルの ID または外部 ID に置き換えます。

   ```
   aws iotsitewise list-assets --asset-model-id asset-model-id
   ```

   詳細については、「ListAssets オペレーション」を参照してください。
2. 前のコマンドでモデルからアセットが返された場合は、各アセットを削除します。詳細については、「アセットを削除する (AWS CLI)」を参照してください。
3. 次のコマンドを実行してアセットモデルを削除します。*asset-model-id* をアセットモデルの ID または外部 ID に置き換えます。

   ```
   aws iotsitewise delete-asset-model --asset-model-id asset-model-id
   ```

> **Important**
>
> 前回の読み取りオペレーション以降に同時に更新されたアセットモデルを削除しないようにするには、条件付き削除リクエストを定義する必要があります。「アセットモデル書き込みの楽観的ロック」を参照してください。

アセットとモデルを使用した一括オペレーション

多数のアセットまたはアセットモデルを使用するには、一括オペレーションを使用してリソースを別の場所に一括インポートおよびエクスポートします。例えば、Amazon S3 バケット内のアセッ

トまたはアセットモデルを定義するデータファイルを作成し、一括インポートを使用して AWS IoT SiteWiseで作成または更新できます。または、 に多数のアセットまたはアセットモデルがある場合は AWS IoT SiteWise、Amazon S3 にエクスポートできます。

> **Note**
>
> AWS IoT TwinMaker API で オペレーションを呼び出す AWS IoT SiteWise ことで、 で一括オペレーションを実行します。 AWS IoT TwinMaker ワークスペースを設定 AWS IoT TwinMaker または作成しなくても、これを行うことができます。必要なのは、 AWS IoT SiteWise コンテンツを配置できる Amazon S3 バケットだけです。

トピック

- 主要な概念と用語
- サポートされている機能
- 一括操作の前提条件
- 一括インポートジョブを実行する
- 一括エクスポートジョブを実行する
- ジョブの進捗状況追跡とエラー処理
- メタデータのインポート例
- メタデータのエクスポート例
- AWS IoT SiteWise メタデータ転送ジョブスキーマ

主要な概念と用語

AWS IoT SiteWise 一括インポートおよびエクスポート機能は、次の概念と用語に依存します。

- インポート: アセットまたはアセットモデルを Amazon S3 バケット内のファイルから AWS IoT SiteWiseに移動するアクション。
- エクスポート: アセットまたはアセットモデル AWS IoT SiteWise を から Amazon S3 バケットに移動するアクション。
- ソース: コンテンツを移動する開始場所。

  例えば、Amazon S3 バケットはインポートソースであり、 AWS IoT SiteWise はエクスポートソースです。

- 送信先: コンテンツを移動する移動先の場所。

  例えば、Amazon S3 バケットはエクスポート先であり、 はインポート先 AWS IoT SiteWise です。
- AWS IoT SiteWise スキーマ: このスキーマは、メタデータのインポートとエクスポートに使用されます AWS IoT SiteWise。
- 最上位リソース： アセットやアセットモデルなど、個別に作成または更新できる AWS IoT SiteWise リソース。
- サブリソース： 最上位 AWS IoT SiteWise リソース内のネストされたリソース。例としては、プロパティ、階層、複合モデルなどがあります。
- メタデータ: リソースのインポートまたはエクスポートの成功に必要な主要情報。メタデータの例は、アセットおよびアセットモデルの定義です。
- metadataTransferJob: `CreateMetadataTransferJob` の実行時に作成されたオブジェクト。

サポートされている機能

このトピックでは、一括操作を実行するときにできることについて説明します。一括操作は、以下の機能をサポートしています。

- 最上位リソースの作成: ID を定義しないアセットまたはアセットモデル、または ID が既存のものと一致しないアセットモデルをインポートすると、新しいリソースとして作成されます。
- 最上位リソースの置換: ID が既存のものと一致するアセットまたはアセットモデルをインポートすると、既存のリソースが置き換えられます。
- サブリソースの作成、置換、または削除: インポートによってアセットやアセットモデルなどの最上位のリソースが置き換えられると、新しい定義によって、プロパティ、階層、複合モデルなど、すべてのサブリソースが置き換えられます。

  例えば、一括インポート中にアセットモデルを更新し、更新されたバージョンで元は存在しなかったプロパティが定義されると、新しいプロパティが作成されます。既に存在するプロパティが定義された場合、既存のプロパティが更新されます。更新されたアセットモデルで元々存在するプロパティが省略された場合、そのプロパティは削除されます。
- 最上位以外のリソースの削除: 一括操作ではアセットまたはアセットモデルは削除されません。一括オペレーションは、作成または更新のみを行います。

一括操作の前提条件

このセクションでは、 AWS サービスとローカルマシン間でリソースを交換するための AWS Identity and Access Management (IAM) アクセス許可など、一括オペレーションの前提条件について説明します。一括操作を開始する前に、以下の前提条件を満たす必要があります。

- リソースを保存する Amazon S3 バケットを作成します。Amazon S3 の使用の詳細については、「Amazon S3 とは」を参照してください。

IAM 許可

一括オペレーションを実行するには、Amazon S3 AWS IoT SiteWiseとローカルマシン間の AWS リソースの交換を許可するアクセス許可を持つ AWS Identity and Access Management (IAM) ポリシーを作成する必要があります。 Amazon S3 IAM ポリシーの作成の詳細については、「IAM ポリシーの作成」を参照してください。

一括操作を実行するには、以下のポリシーが必要です。

AWS IoT SiteWise ポリシー

このポリシーは、一括オペレーションに必要な AWS IoT SiteWise API アクションへのアクセスを許可します。

```
{
    "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 ポリシー

このポリシーは、一括オペレーションの操作に使用する AWS IoT TwinMaker API オペレーションへのアクセスを許可します。

```
{
    "Sid": "MetadataTransferJobApiAccess",
    "Effect": "Allow",
    "Action": [
        "iottwinmaker:CreateMetadataTransferJob",
        "iottwinmaker:CancelMetadataTransferJob",
        "iottwinmaker:GetMetadataTransferJob",
        "iottwinmaker:ListMetadataTransferJobs"
    ],
    "Resource": "*"
}
```

Amazon S3 ポリシー

このポリシーは、一括オペレーションのメタデータを転送するための Amazon S3 バケットへのアクセスを提供します。

For a specific Amazon S3 bucket

一括オペレーションメタデータの操作に 1 つの特定のバケットを使用する場合、このポリシーはそのバケットへのアクセスを提供します。

```
{
    "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

多くの異なるバケットを使用して一括操作メタデータを操作する場合、このポリシーは任意のバケットへのアクセスを提供します。

```
{
    "Effect": "Allow",
    "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:GetBucketLocation",
        "s3:ListBucket",
        "s3:AbortMultipartUpload",
        "s3:ListBucketMultipartUploads",
        "s3:ListMultipartUploadParts"
    ],
    "Resource": "*"
}
```

インポートおよびエクスポートオペレーションのトラブルシューティングについては、「一括インポートとエクスポートのトラブルシューティング」を参照してください。

一括インポートジョブを実行する

一括インポートは、メタデータを AWS IoT SiteWise ワークスペースに移動するアクションです。たとえば、一括インポートでは、メタデータをローカルファイルまたは Amazon S3 バケット内のファイルから AWS IoT SiteWise ワークスペースに移動できます。

ステップ 1: インポートするファイルを準備する

AWS IoT SiteWise ネイティブ形式ファイルをダウンロードして、アセットとアセットモデルをインポートします。詳細については、「AWS IoT SiteWise メタデータ転送ジョブスキーマ」を参照してください。

ステップ 2: 準備したファイルを Amazon S3 にアップロードする

このファイルを Amazon S3 にアップロードします。詳しくは、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 へのファイルのアップロード」を参照してください。

メタデータをインポートする (コンソール)

を使用してメタデータ AWS IoT SiteWise コンソール を一括インポートできます。ステップ 1: インポートするファイルを準備する と ステップ 2: 準備したファイルを Amazon S3 にアップロードする に従って、インポートできるファイルを準備します。

Amazon S3 から にデータをインポートする AWS IoT SiteWise コンソール

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインから [一括オペレーション 新規] を選択します。
3. [新規インポート] を選択して、インポートプロセスを開始します。
4. [メタデータをインポート] ページで、以下の操作を行います。
   - [Amazon S3 を参照] を選択して、Amazon S3 バケットとファイルを表示します。
   - 準備されたインポートファイルを含む Amazon S3 バケットに移動します。
   - インポートするファイルを選択します。
   - 選択したファイルを確認し、[インポート] を選択します。
5. の SiteWise メタデータページの一括オペレーションでは、新しく作成されたインポートジョブがジョブの進行状況テーブル AWS IoT SiteWise コンソール に表示されます。

メタデータをインポートする (AWS CLI)

インポートアクションを実行するには、以下の手順を使用します。

Amazon S3 から にデータをインポートする AWS CLI

1. AWS IoT SiteWise メタデータ転送ジョブスキーマ に続いて、インポートするリソースを指定するメタデータファイルを作成します。このファイルを Amazon S3 バケットに保存します。

   インポートするメタデータファイルの例については、「メタデータのインポート例」を参照してください。

2. 次に、リクエスト本文を含む JSON ファイルを作成します。リクエスト本文では、転送ジョブの送信元と送信先を指定します。このファイルは、前のステップのファイルとは別のものです。Amazon S3 バケットを送信元として、`iotsitewise` を送信先として指定してください。

   次の例はリクエスト本文を示しています。

```
{
      "metadataTransferJobId": "your-transfer-job-Id",
      "sources": [{
          "type": "s3",
          "s3Configuration": {
              "location": "arn:aws:s3:::amzn-s3-demo-bucket/
your_import_metadata.json"
          }
      }],
      "destination": {
          "type": "iotsitewise"
      }
  }
```

3. 次の AWS CLI コマンド`CreateMetadataTransferJob`を実行して、 を呼び出します。この例では、前のステップからのリクエスト本文ファイルは `createMetadataTransferJobExport.json` という名前です。

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
  --cli-input-json file://createMetadataTransferJobImport.json
```

   これは、メタデータ転送ジョブを作成し、選択したリソースの転送プロセスを開始します。

一括エクスポートジョブを実行する

一括エクスポートは、 AWS IoT SiteWise ワークスペースから Amazon S3 バケットにメタデータを移動するアクションです。

Amazon S3 への AWS IoT SiteWise コンテンツの一括エクスポートを実行する場合、エクスポートする特定のアセットモデルとアセットを制限するフィルターを指定できます。

フィルターは、JSON リクエストのソースセクション内の `iotSiteWiseConfiguration` セクションで指定する必要があります。

> **Note**
>
> 複数のフィルターをリクエストに含めることができます。一括オペレーションでは、いずれかのフィルターに一致するアセットモデルとアセットがエクスポートされます。
> フィルターを指定しなかった場合、一括オペレーションはすべてのアセットモデルとアセットをエクスポートします。

Example フィルターを使用したリクエスト本文

```
{
      "metadataTransferJobId": "your-transfer-job-id",
      "sources": [
       {
        "type": "iotsitewise",
        "iotSiteWiseConfiguration": {
          "filters": [
           {
              "filterByAssetModel": {
                  "assetModelId": "asset model ID"
              }
            },
            {
              "filterByAssetModel": {
                  "assetModelId": "asset model ID",
                  "includeAssets": true
              }
            },
            {
              "filterByAssetModel": {
```

```
                    "assetModelId": "asset model ID",
                    "includeOffspring": true
                }
              }
            ]
           }
         }
        ],
        "destination": {
           "type": "s3",
           "s3Configuration": {
             "location": "arn:aws:s3:::amzn-s3-demo-bucket"
           }
      }
}
```

メタデータをエクスポート (コンソール)

次の手順で、コンソールのエクスポートアクションについて説明します。

でエクスポートジョブを作成する AWS IoT SiteWise コンソール

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインから [一括オペレーション 新規] を選択します。
3. [新規エクスポート] を選択して、エクスポートプロセスを開始します。
4. [メタデータをエクスポート] ページで、以下の操作を行います。
   - エクスポートジョブの名前を入力します。これは、Amazon S3 バケットでエクスポートされたファイルに使用される名前です。
   - エクスポートするリソースを選択します。これによって、ジョブのフィルターが設定されます。
     - すべてのアセットとアセットモデルをエクスポートします。アセットとアセットモデルにフィルターを使用します。
     - アセットをエクスポートします。アセットをフィルタリングします。
       - エクスポートフィルターに使用するアセットを選択します。
       - (オプション) 子または関連するアセットモデルを追加します。
     - アセットモデルをエクスポートします。アセットモデルをフィルタリングします。
       - エクスポートフィルターに使用するアセットモデルを選択します。

    - (オプション) 子、または関連するアセット、またはその両方を追加します。
  - [次へ] を選択します。
- Amazon S3 バケットに移動します。
  - [Amazon S3 を参照] を選択して、Amazon S3 バケットとファイルを表示します。
  - ファイルを配置する必要がある Amazon S3 バケットに移動します。
  - [次へ] を選択します。
- エクスポートジョブを確認して、[エクスポート] を選択します。

5. の SiteWise メタデータページの一括オペレーションでは、新しく作成されたインポートジョブがジョブの進行状況テーブル AWS IoT SiteWise コンソール に表示されます。

メタデータのエクスポート時にフィルターを使用するさまざまな方法については、「メタデータのエクスポート例」を参照してください。

メタデータをエクスポート (AWS CLI)

次の手順では、 AWS CLI エクスポートアクションについて説明します。

から Amazon S3 AWS IoT SiteWise にデータをエクスポートする

1. リクエスト本文を含んだ JSON ファイルを作成します。リクエスト本文では、転送ジョブの送信元と送信先を指定します。次の例はリクエスト本文の例を示しています。

```
{
    "metadataTransferJobId": "your-transfer-job-Id",
    "sources": [{
        "type": "iotsitewise"
    }],
    "destination": {
        "type": "s3",
        "s3Configuration": {
            "location": "arn:aws:s3:::amzn-s3-demo-bucket"
        }
    }
}
```

   メタデータ転送ジョブの送信先として Amazon S3 バケットを必ず指定してください。

> **Note**
>
> この例では、すべてのアセットモデルとアセットをエクスポートします。エクスポートを特定のアセットモデルまたはアセットに制限するには、リクエスト本文にフィルターを含めることができます。エクスポートフィルターの適用の詳細については、「メタデータのエクスポート例」を参照してください。

2. 次のステップで使用するリクエスト本文ファイルを保存します。この例では、ファイル名は `createMetadataTransferJobExport.json` です。
3. 次の AWS CLI コマンド`CreateMetadataTransferJob`を実行して を呼び出します。

```
aws iottwinmaker create-metadata-transfer-job --region us-east-1 \
         --cli-input-json file://createMetadataTransferJobExport.json
```

入力 JSON ファイル `createMetadataTransferJobExport.json` をユーザー独自の転送ファイル名に置き換えます。

ジョブの進捗状況追跡とエラー処理

一括処理ジョブの処理には時間がかかります。各ジョブは、リクエスト AWS IoT SiteWise の受信順に処理されます。アカウントごとに一度に 1 つずつ処理されます。ジョブが完了すると、キュー内の次の は自動的に処理を開始します。 はジョブを非同期的に AWS IoT SiteWise 解決し、進行するたびに各 のステータスを更新します。各ジョブには、リソースの状態とエラーメッセージ (該当する場合) を含んだステータスフィールドがあります。

ステータスは以下のいずれかの値になります。

- `VALIDATING` – 送信されたファイル形式とその内容を含むジョブを検証します。
- `PENDING` – ジョブはキューにあります。この状態のジョブは AWS IoT SiteWise コンソールからキャンセルできますが、他のすべての状態は最後まで継続されます。
- `RUNNING` – ジョブの処理中。インポートファイルによって定義されているリソースを作成および更新中です。または、選択されたエクスポートジョブフィルターに基づいてリソースをエクスポート中です。キャンセルされた場合、このジョブによってインポートされたリソースは削除されません。詳細については「ジョブの進捗状況と詳細を確認する (コンソール)」を参照してください。
- `CANCELLING` – ジョブはアクティブにキャンセルされています。

- ERROR – 1 つ以上のリソースが処理に失敗しました。詳細については、詳細なジョブレポートを確認してください。詳細については「エラーの詳細を検査する (コンソール)」を参照してください。
- COMPLETED – ジョブはエラーなしで完了しました。
- CANCELLED – ジョブはキャンセルされ、キューに入れられません。RUNNING ジョブをキャンセルした場合、キャンセル時にこのジョブによってインポートされたリソースは削除されません AWS IoT SiteWise。

トピック

- ジョブの進捗状況追跡
- のエラーを検査する AWS IoT SiteWise

ジョブの進捗状況追跡

ジョブの進捗状況と詳細を確認する (コンソール)

メタデータをインポートする (コンソール) または メタデータをエクスポート (コンソール) を参照して、一括ジョブを開始します。

AWS IoT SiteWise コンソールでのジョブの進行状況の概要：

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインから [一括オペレーション 新規] を選択します。
3. AWS IoT SiteWise コンソールのジョブの進行状況テーブルに、一括オペレーションジョブのリストが表示されます。
4. [ジョブタイプ] 列は、エクスポートジョブかインポートジョブかを示します。[インポート日] 列には、ジョブが開始された日付が表示されます。
5. [ステータス] 列には、ジョブのステータスが表示されます。ジョブを選択して、そのジョブの詳細を表示できます。
6. 選択したジョブには、成功時には [成功]、またはジョブが失敗した場合は失敗のリストが表示されます。エラーの説明も、リソースタイプごとに表示されます。

AWS IoT SiteWise コンソールでのジョブの詳細の概要：

AWS IoT SiteWise コンソールのジョブの進行状況テーブルに、一括オペレーションジョブのリストが表示されます。

1. 詳細を表示するジョブを選択します。
2. [インポート] ジョブの場合、`Data source ARN` はインポートファイルの Amazon S3 の場所を表します。
3. [エクスポート] ジョブの場合、`Data destination ARN` はエクスポート後のファイルの Amazon S3 の場所を表します。
4. `Status` と `Status reason` は、現在のジョブに関する追加の詳細を提供します。詳細については、「ジョブの進捗状況追跡とエラー処理」を参照してください。
5. `Queued position` は、プロセスキュー内のジョブの位置を表します。ジョブは一度に 1 つずつ処理されます。キューに入れられた位置が 1 の場合、ジョブが次に処理されることを示します。
6. ジョブの詳細ページには、ジョブの進捗状況のカウントも表示されます。
   - ジョブの進捗状況カウントタイプは次のとおりです。
     i. `Total resources` – 転送プロセス内のアセットの合計数を示します。
     ii. `Succeeded` – プロセス中に正常に転送されたアセットの数を示します。
     iii. `Failed` – プロセス中に失敗したアセットの数を示します。
     iv. `Skipped` – プロセス中にスキップされたアセットの数を示します。
7. ジョブステータスが `PENDING` または `VALIDATING` の場合、すべてのジョブの進捗状況カウントを – として表示します。これは、ジョブの進捗状況カウントが評価されていることを示します。
8. ジョブステータスが `RUNNING` の場合は、`Total resources` カウント、処理のために送信されたジョブが表示されます。詳細カウント (`Succeeded`、`Failed`、および `Skipped`) は、処理されたリソースに適用されます。詳細カウントの合計は、ジョブのステータスが `COMPLETED` または `ERROR` になるまで、`Total resources` カウントよりも小さくなります。
9. ジョブのステータスが `COMPLETED` または `ERROR` の場合、`Total resources` カウントは詳細カウント (`Succeeded`、`Failed`、および `Skipped`) の合計に等しくなります。
10. ジョブのステータスが `ERROR` の場合、[ジョブの失敗] テーブルで特定のエラーと失敗の詳細を確認します。詳細については、「エラーの詳細を検査する (コンソール)」を参照してください。

ジョブの進捗状況と詳細を確認する (AWS CLI)

一括オペレーションを開始した後、以下の API アクションを使用してステータスを確認または更新できます。

- 特定のジョブに関する情報を取得するには、GetMetadataTransferJob API アクションを使用します。

  以下のように **GetMetadataTransferJob** API を使用して情報を取得します。

  1. 転送ジョブを作成して実行します。GetMetadataTransferJob API を呼び出します。

     Example AWS CLI コマンド：

     ```
     aws iottwinmaker get-metadata-transfer-job \
             --metadata-transfer-job-id your_metadata_transfer_job_id \
             --region your_region
     ```

  2. GetMetadataTransferJob API は、以下のパラメータを持つ MetadataTransferJobProgress オブジェクトを返します。

     - succeededCount – プロセスで正常に転送されたアセットの数を示します。
     - failedCount – プロセス中に失敗したアセットの数を示します。
     - skippedCount – プロセス中にスキップされたアセットの数を示します。
     - totalCount – 転送プロセス内のアセットの合計数を示します。

     これらのパラメータは、ジョブの進捗状況ステータスを示します。ステータスが RUNNING の場合、まだ処理されていないリソースの数を追跡するのに役立ちます。

     スキーマ検証エラーが発生した場合、または failedCount が 1 以上の場合、ジョブの進捗状況ステータスは ERROR になります。ジョブの完全なエラーレポートは Amazon S3 バケットに配置されます。詳細については、「のエラーを検査する AWS IoT SiteWise」を参照してください。

- 現在のジョブを一覧表示するには、ListMetadataTransferJobs API アクションを使用します。

  JSON ファイルを使用して、返されたジョブを現在の状態に基づいてフィルタリングします。以下の手順を参照してください。

  1. 使用するフィルターを指定するには、 AWS CLI 入力 JSON ファイルを作成します。 は以下を使用します。

     ```
     {
         "sourceType": "s3",
         "destinationType": "iottwinmaker",
     ```

```
    "filters": [{
        "state": "COMPLETED"
    }]
}
```

有効な state 値のリストについては、「AWS IoT TwinMaker API リファレンスガイド」の「ListMetadataTransferJobsFilter」を参照してください。

2. 次のコマンド AWS CLI 例では、JSON ファイルを引数として使用します。

```
aws iottwinmaker list-metadata-transfer-job --region your_region \
        --cli-input-json file://ListMetadataTransferJobsExample.json
```

- ジョブをキャンセルするには、CancelMetadataTransferJob API アクションを使用します。この API は、既にエクスポートまたはインポートされたリソースに影響を与えることなく、特定のメタデータ転送ジョブをキャンセルします。

```
aws iottwinmaker cancel-metadata-transfer-job \
        --region your_region \
        --metadata-transfer-job-id job-to-cancel-id
```

のエラーを検査する AWS IoT SiteWise

エラーの詳細を検査する (コンソール)

AWS IoT SiteWise コンソールのエラーの詳細：

1. AWS IoT SiteWise コンソールに移動します。
2. 一括オペレーションジョブのリストについては、 のジョブの進行状況表を参照してください。AWS IoT SiteWise コンソール
3. ジョブを選択して、ジョブの詳細を表示します。
4. ジョブのステータスが COMPLETED または ERROR の場合、Total resources カウントは詳細カウント (Succeeded、Failed、および Skipped) の合計に等しくなります。
5. ジョブのステータスが ERROR の場合、[ジョブの失敗] テーブルで特定のエラーと失敗の詳細を確認します。

6. [ジョブの失敗] テーブルには、ジョブレポートの内容が表示されます。`Resource type` フィールドには、次のようなエラーまたは失敗の場所が示されます。

   - 例えば、 `Resource type` フィールドの `Bulk operations template` の検証エラーは、インポートテンプレートとメタデータスキーマのファイル形式が一致しないことを示します。詳細については「AWS IoT SiteWise メタデータ転送ジョブスキーマ」を参照してください。
   - `Resource type` フィールドの失敗した `Asset` は、別のアセットとの競合によりアセットが作成されないことを示します。 AWS IoT SiteWise リソースのエラーと競合については、「一般的なエラー」を参照してください。

エラーの詳細を検査する (AWS CLI)

転送ジョブ中に生成されたエラーを処理および診断するには、`GetMetadataTransferJob` API アクションの使用に関する次の手順を参照してください。

1. 転送ジョブを作成して実行した後、GetMetadataTransferJob を呼び出します。

```
aws iottwinmaker get-metadata-transfer-job \
        --metadata-transfer-job-id your_metadata_transfer_job_id \
        --region us-east-1
```

2. ジョブの状態が `COMPLETED` になったら、ジョブの結果の確認を開始できます。
3. `GetMetadataTransferJob` を呼び出すと、`MetadataTransferJobProgress` というオブジェクトを返します。

   MetadataTransferJobProgress オブジェクトには、以下のパラメータが含まれています。

   - failedCount: 転送プロセス中に失敗したアセットの数を示します。
   - skippedCount: 転送プロセス中にスキップされたアセットの数を示します。
   - succeededCount: 転送プロセス中に成功したアセットの数を示します。
   - totalCount: 転送プロセスに関連するアセットの合計数を示します。
4. さらに、API コールは、署名付き URL を含む要素 `reportUrl` を返します。転送ジョブにさらに調査する必要がある問題がある場合は、この URL にアクセスしてください。

メタデータのインポート例

このセクションでは、メタデータファイルを作成して、単一の一括インポートオペレーションでアセットモデルとアセットをインポートする方法について説明します。

一括インポートの例

1 回の一括インポートオペレーションで、多くのアセットモデルとアセットをインポートできます。次の例は、メタデータファイルを作成して、これを行う方法を示しています。

この例のシナリオでは、作業セルに産業用ロボットを含むさまざまな作業サイトがあります。

この例では、2 つのアセットモデルを定義します。

- `RobotModel1`: このアセットモデルは、作業サイトにある特定のタイプのロボットを表します。ロボットには測定プロパティ `Temperature` があります。
- `WorkCell`: このアセットモデルは、いずれかの作業サイト内のロボットのコレクションを表します。アセットモデルは、階層 `robotHierarchyOEM1` を定義し、作業セルにロボットが含まれる関係を表します。

この例では、いくつかのアセットも定義しています。

- `WorkCell1`: ボストンサイト内の作業セル
- `RobotArm123456`: その作業セル内のロボット
- `RobotArm987654`: その作業セル内の別のロボット

次の JSON メタデータファイルは、これらのアセットモデルとアセットを定義します。このメタデータで一括インポートを実行すると AWS IoT SiteWise、階層関係を含むアセットモデルとアセットが 内に作成されます。

インポート用のメタデータファイル

```
{
    "assetModels": [
        {
            "assetModelExternalId": "Robot.OEM1.3536",
            "assetModelName": "RobotModel1",
            "assetModelProperties": [
                {
                    "dataType": "DOUBLE",
```

```
                    "externalId": "Temperature",
                    "name": "Temperature",
                    "type": {
                        "measurement": {
                            "processingConfig": {
                                "forwardingConfig": {
                                    "state": "ENABLED"
                                }
                            }
                        }
                    },
                    "unit": "fahrenheit"
                }
            ]
        },
        {
            "assetModelExternalId": "ISA95.WorkCell",
            "assetModelName": "WorkCell",
            "assetModelProperties": [],
            "assetModelHierarchies": [
                {
                    "externalId": "workCellHierarchyWithOEM1Robot",
                    "name": "robotHierarchyOEM1",
                    "childAssetModelExternalId": "Robot.OEM1.3536"
                }
            ]
        }
    ],
    "assets": [
        {
            "assetExternalId": "Robot.OEM1.3536.123456",
            "assetName": "RobotArm123456",
            "assetModelExternalId": "Robot.OEM1.3536"
        },
        {
            "assetExternalId": "Robot.OEM1.3536.987654",
            "assetName": "RobotArm987654",
            "assetModelExternalId": "Robot.OEM1.3536"
        },
        {
            "assetExternalId": "BostonSite.Area1.Line1.WorkCell1",
            "assetName": "WorkCell1",
            "assetModelExternalId": "ISA95.WorkCell",
            "assetHierarchies": [
```

```
                {
                    "externalId": "workCellHierarchyWithOEM1Robot",
                    "childAssetExternalId": "Robot.OEM1.3536.123456"
                },
                {
                    "externalId": "workCellHierarchyWithOEM1Robot",
                    "childAssetExternalId": "Robot.OEM1.3536.987654"
                }
            ]
        }
    ]
}
```

モデルとアセットの初期オンボーディングの例

この例のシナリオでは、会社の産業用ロボットを含むさまざまな作業サイトがあります。

この例では、複数のアセットモデルを定義します。

- Sample_Enterprise – このアセットモデルは、サイトが属する会社を表します。アセットモデルは、サイトとエンタープライズの関係を表す階層 Enterprise to Site を定義します。
- Sample_Site – このアセットモデルは、社内の製造サイトを表します。アセットモデルは、階層 Site to Line を定義して、サイトとラインの関係を表します。
- Sample_Welding Line – このアセットモデルは、作業サイト内のアセンブリラインを表します。アセットモデルは、ロボットとラインの関係を表す階層 Line to Robot を定義します。
- Sample_Welding Robot – このアセットモデルは、作業サイトの特定のタイプのロボットを表します。

この例では、アセットモデルに基づいてアセットも定義します。

- Sample_AnyCompany Motor - このアセットは、Sample_Enterprise アセットモデルから作成されます。
- Sample_Chicago - このアセットは、Sample_Site アセットモデルから作成されます。
- Sample_Welding Line 1 - このアセットは、Sample_Welding Line アセットモデルから作成されます。
- Sample_Welding Robot 1 - このアセットは、Sample_Welding Robot アセットモデルから作成されます。

- `Sample_Welding Robot 2` - このアセットは、`Sample_Welding Robot` アセットモデルから作成されます。

次の JSON メタデータファイルは、これらのアセットモデルとアセットを定義します。このメタデータで一括インポートを実行すると AWS IoT SiteWise、階層関係を含むアセットモデルとアセットが 内に作成されます。

インポート用のアセットとモデルをオンボードする JSON ファイル

```
{
    "assetModels": [
        {
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetModelName": "Sample_Welding Robot",
            "assetModelProperties": [
                {
                    "dataType": "STRING",
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "name": "Serial Number",
                    "type": {
                        "attribute": {
                            "defaultValue": "-"
                        }
                    },
                    "unit": "-"
                },
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "name": "CycleCount",
                    "type": {
                        "measurement": {}
                    },
                    "unit": "EA"
                },
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "name": "Joint 1 Current",
                    "type": {
                        "measurement": {}
```

```
                    },
                    "unit": "Amps"
                },
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Robot_Joint_1_Max_Current",
                    "name": "Max Joint 1 Current",
                    "type": {
                        "metric": {
                            "expression": "max(joint1current)",
                            "variables": [
                                {
                                    "name": "joint1current",
                                    "value": {
                                        "propertyExternalId":
 "External_Id_Welding_Robot_Joint_1_Current"
                                    }
                                }
                            ],
                            "window": {
                                "tumbling": {
                                    "interval": "5m"
                                }
                            }
                        }
                    },
                    "unit": "Amps"
                }
            ]
        },
        {
            "assetModelExternalId": "External_Id_Welding_Line",
            "assetModelName": "Sample_Welding Line",
            "assetModelProperties": [
                {
                    "dataType": "DOUBLE",
                    "externalId": "External_Id_Welding_Line_Availability",
                    "name": "Availability",
                    "type": {
                        "measurement": {}
                    },
                    "unit": "%"
                }
            ],
```

```
            "assetModelHierarchies": [
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "name": "Line to Robot",
                    "childAssetModelExternalId": "External_Id_Welding_Robot"
                }
            ]
        },
        {
            "assetModelExternalId": "External_Id_Site",
            "assetModelName": "Sample_Site",
            "assetModelProperties": [
                {
                    "dataType": "STRING",
                    "externalId": "External_Id_Site_Street_Address",
                    "name": "Street Address",
                    "type": {
                        "attribute": {
                            "defaultValue": "-"
                        }
                    },
                    "unit": "-"
                }
            ],
            "assetModelHierarchies": [
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "name": "Site to Line",
                    "childAssetModelExternalId": "External_Id_Welding_Line"
                }
            ]
        },
        {
            "assetModelExternalId": "External_Id_Enterprise",
            "assetModelName": "Sample_Enterprise",
            "assetModelProperties": [
                {
                    "dataType": "STRING",
                    "name": "Company Name",
                    "externalId": "External_Id_Enterprise_Company_Name",
                    "type": {
                        "attribute": {
                            "defaultValue": "-"
                        }
```

```
                    },
                    "unit": "-"
                }
            ],
            "assetModelHierarchies": [
                {
                    "externalId": "External_Id_Enterprise_TO_Site",
                    "name": "Enterprise to Site",
                    "childAssetModelExternalId": "External_Id_Site"
                }
            ]
        }
    ],
    "assets": [
        {
            "assetExternalId": "External_Id_Welding_Robot_1",
            "assetName": "Sample_Welding Robot 1",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "attributeValue": "S1000"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "alias": "AnyCompany/Chicago/Welding Line/S1000/Count"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "alias": "AnyCompany/Chicago/Welding Line/S1000/1/Current"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Robot_2",
            "assetName": "Sample_Welding Robot 2",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "attributeValue": "S2000"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
```

```
                    "alias": "AnyCompany/Chicago/Welding Line/S2000/Count"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "alias": "AnyCompany/Chicago/Welding Line/S2000/1/Current"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Line_1",
            "assetName": "Sample_Welding Line 1",
            "assetModelExternalId": "External_Id_Welding_Line",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Line_Availability",
                    "alias": "AnyCompany/Chicago/Welding Line/Availability"
                }
            ],
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_1"
                },
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_2"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Site_Chicago",
            "assetName": "Sample_Chicago",
            "assetModelExternalId": "External_Id_Site",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "childAssetExternalId": "External_Id_Welding_Line_1"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Enterprise_AnyCompany",
            "assetName": "Sample_AnyEnterprise Motor",
            "assetModelExternalId": "External_Id_Enterprise",
```

```
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Enterprise_TO_Site",
                    "childAssetExternalId": "External_Id_Site_Chicago"
                }
            ]
        }
    ]
}
```

次のスクリーンショットは、前のコード例を実行 AWS IoT SiteWise コンソール した後に に表示されるモデルのスクリーンショットです。

次のスクリーンショットは、前のコード例を実行 AWS IoT SiteWise コンソール した後に に表示されるモデル、アセット、階層を示しています。

追加アセットのオンボーディングの例

この例では、アカウント内の既存のアセットモデルにインポートする追加のアセットを定義します。

- `Sample_Welding Line 2` - このアセットは、`Sample_Welding Line` アセットモデルから作成されます。
- `Sample_Welding Robot 3` - このアセットは、`Sample_Welding Robot` アセットモデルから作成されます。
- `Sample_Welding Robot 4` - このアセットは、`Sample_Welding Robot` アセットモデルから作成されます。

この例の初期アセットを作成するには、「モデルとアセットの初期オンボーディングの例」を参照してください。

次の JSON メタデータファイルは、これらのアセットモデルとアセットを定義します。このメタデータで一括インポートを実行すると AWS IoT SiteWise、階層関係を含むアセットモデルとアセットが 内に作成されます。

追加のアセットをオンボードする JSON ファイル

```
{
    "assets": [
        {
            "assetExternalId": "External_Id_Welding_Robot_3",
            "assetName": "Sample_Welding Robot 3",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "attributeValue": "S3000"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "alias": "AnyCompany/Chicago/Welding Line/S3000/Count"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "alias": "AnyCompany/Chicago/Welding Line/S3000/1/Current"
                }
            ]
```

```
        },
        {
            "assetExternalId": "External_Id_Welding_Robot_4",
            "assetName": "Sample_Welding Robot 4",
            "assetModelExternalId": "External_Id_Welding_Robot",
            "assetProperties": [
                {
                    "externalId": "External_Id_Welding_Robot_Serial_Number",
                    "attributeValue": "S4000"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Cycle_Count",
                    "alias": "AnyCompany/Chicago/Welding Line/S4000/Count"
                },
                {
                    "externalId": "External_Id_Welding_Robot_Joint_1_Current",
                    "alias": "AnyCompany/Chicago/Welding Line/S4000/1/Current"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Line_1",
            "assetName": "Sample_Welding Line 1",
            "assetModelExternalId": "External_Id_Welding_Line",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_1"
                },
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_2"
                },
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_3"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Welding_Line_2",
            "assetName": "Sample_Welding Line 2",
            "assetModelExternalId": "External_Id_Welding_Line",
            "assetHierarchies": [
```

```
                {
                    "externalId": "External_Id_Welding_Line_TO_Robot",
                    "childAssetExternalId": "External_Id_Welding_Robot_4"
                }
            ]
        },
        {
            "assetExternalId": "External_Id_Site_Chicago",
            "assetName": "Sample_Chicago",
            "assetModelExternalId": "External_Id_Site",
            "assetHierarchies": [
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "childAssetExternalId": "External_Id_Welding_Line_1"
                },
                {
                    "externalId": "External_Id_Site_TO_Line",
                    "childAssetExternalId": "External_Id_Welding_Line_2"
                }
            ]
        }
    ]
}
```

次のスクリーンショットは、前のコード例を実行 AWS IoT SiteWise コンソール した後に に表示されるモデル、アセット、階層を示しています。

新しいプロパティのオンボーディングの例

この例では、既存のアセットモデルに新しいプロパティを定義します。追加のアセットとモデルをオンボードするには、「追加アセットのオンボーディングの例」を参照してください。

- `Joint 1 Temperature` – このプロパティは、`Sample_Welding Robot` アセットモデルに追加されます。この新しいプロパティは、`Sample_Welding Robot` アセットモデルから作成された各アセットにも伝播されます。

既存のアセットモデルに新しいプロパティを追加するには、次の JSON メタデータファイルの例を参照してください。JSON に示されているように、既存の `Sample_Welding Robot` アセットモデル定義全体が新しいプロパティとともに提供される必要があります。既存の定義のプロパティリスト全体が指定されていない場合、 は省略されたプロパティ AWS IoT SiteWise を削除します。

新しいプロパティをオンボードする JSON ファイル

この例では、アセットモデルに新しいプロパティ `Joint 1 Temperature` を追加します。

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

データストリームを管理する例

この例は、アセットプロパティに関連付けられたデータストリームを管理する 2 つの方法を示しています。アセットプロパティエイリアスの名前を変更する場合、アセットプロパティのデータストリームに現在保存されている履歴データには 2 つのオプションがあります。

- オプション 1 – 現在のデータストリームを保持し、エイリアスのみ名前を変更して、履歴データに新しいエイリアスでアクセスできるようにします。

  JSON メタデータファイルの例では、ID `External_Id_Welding_Robot_Cycle_Count` を持つアセットプロパティがエイリアスを `AnyCompany/Chicago/Welding Line/S3000/Count-Updated` に変更します。このアセットプロパティの履歴データは、この変更後も同じままです。
- オプション 2 – 新しいエイリアスでアクセスできるアセットプロパティに新しいデータストリームを割り当てます。古いデータストリームとその履歴データには、古いエイリアスでアクセスできますが、アセットプロパティには関連付けられていません。

  JSON メタデータファイルの例では、ID `External_Id_Welding_Robot_Joint_1_Current` を持つアセットプロパティがエイリアスを `AnyCompany/Chicago/Welding Line/S4999/1/Current` に変更します。今回は、追加の値 `retainDataOnAliasChange` が存在し、`False` に設定されます。この設定では、元のデータストリームはアセットプロパティから関連付け解除され、履歴データを含まない新しいデータストリームが作成されます。

元の履歴データを使用して古いデータストリームにアクセスするには、 でデータストリームページに移動し AWS Console Home、古いエイリアス を検索しますAnyCompany/Chicago/Welding Line/S3000/1/Current。

プロパティエイリアスを更新する JSON ファイル

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

メタデータのエクスポート例

Amazon S3 への AWS IoT SiteWise コンテンツの一括エクスポートを実行する場合、エクスポートする特定のアセットモデルとアセットを制限するフィルターを指定できます。

フィルターは、リクエスト本文の `sources` セクション内の `iotSiteWiseConfiguration` セクションで指定します。

> **Note**
>
> 複数のフィルターを含めることができます。一括オペレーションは、いずれかのフィルターに一致するアセットモデルまたはアセットをエクスポートします。
> フィルターを指定しなかった場合、オペレーションはすべてのアセットモデルとアセットをエクスポートします。

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

アセットモデルでフィルタリングする

特定のアセットモデルをフィルタリングできます。そのモデルを使用するすべてのアセット、または階層内のすべてのアセットモデルを含めることもできます。アセットと階層の両方を含めることはできません。

階層の詳細については、「アセットモデルの階層を定義する」を参照してください。

Asset model

このフィルターには、指定されたアセットモデルが含まれます。

```
"filterByAssetModel": {
    "assetModelId": "asset model ID"
}
```

Asset model and its assets

このフィルターには、指定されたアセットモデルとそのアセットモデルを使用するすべてのアセットが含まれます。

```
"filterByAssetModel": {
    "assetModelId": "asset model ID",
    "includeAssets": true
}
```

Asset model and its hierarchy

このフィルターには、指定されたアセットモデルと、階層内の関連するすべてのアセットモデルが含まれます。

```
"filterByAssetModel": {
    "assetModelId": "asset model ID",
    "includeOffspring": true
}
```

アセットでフィルタリングする

特定のアセットをフィルタリングできます。そのアセットモデル、または関連するすべてのアセットを階層に含めることもできます。アセットモデルと階層の両方を含めることはできません。

階層の詳細については、「アセットモデルの階層を定義する」を参照してください。

Asset

このフィルターには、指定されたアセットが含まれます。

```
"filterByAsset": {
    "assetId": "asset ID"
}
```

Asset and its asset model

このフィルターには、指定したアセットと、それが使用するアセットモデルが含まれます。

```
"filterByAsset": {
    "assetId": "asset ID",
    "includeAssetModel": true
}
```

Asset and its hierarchy

このフィルターには、指定されたアセットと、階層内の関連するすべてのアセットが含まれます。

```
"filterByAsset": {
    "assetId": "asset ID",
    "includeOffspring": true
}
```

AWS IoT SiteWise メタデータ転送ジョブスキーマ

独自の一括インポートおよびエクスポートオペレーションを実行するときは、 AWS IoT SiteWise メタデータ転送ジョブスキーマを参照用に使用します。

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "IoTSiteWise",
  "description": "Metadata transfer job resource schema for IoTSiteWise",
  "definitions": {
    "Name": {
      "type": "string",
      "minLength": 1,
      "maxLength": 256,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "Description": {
      "type": "string",
      "minLength": 1,
      "maxLength": 2048,
      "pattern": "[^\\u0000-\\u001F\\u007F]+"
    },
    "ID": {
      "type": "string",
      "minLength": 36,
      "maxLength": 36,
      "pattern": "^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$"
    },
    "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"
      }
    }
  }
}
```

でアラームを使用してデータをモニタリングする AWS IoT SiteWise

データに対してアラームを設定し、機器の動作やプロセスが最適な状態でない場合に、チームにアラート通知を送信することができます。機器やプロセスの最適なパフォーマンスとは、特定のメトリクスの値が上限と下限の範囲内にあることを指します。これらのメトリクスが運用の範囲外の場合、機器のオペレーターに通知して問題を解決する必要があります。アラームを使用して、問題を迅速に特定し、オペレーターに通知して、機器やプロセスのパフォーマンスを最大限に高めることができます。

トピック

- アラーム型
- アラームの状態
- アラーム状態のプロパティ
- でアセットモデルにアラームを定義する AWS IoT SiteWise
- でアセットにアラームを設定する AWS IoT SiteWise
- でアラームに応答する AWS IoT SiteWise
- で外部アラーム状態を取り込む AWS IoT SiteWise

アラーム型

AWS クラウドで検出するアラームと、外部プロセスで検出するアラームを定義できます。 は、次のタイプのアラーム AWS IoT SiteWise をサポートしています。

- AWS IoT Events アラーム

> **Note**
>
> サポート終了通知: 2026 年 5 月 20 日に、 AWS は のサポートを終了します AWS IoT Events。2026 年 5 月 20 日以降、 AWS IoT Events コンソールまたは AWS IoT Events リソースにアクセスできなくなります。詳細については、AWS IoT Events 「サポート終了」を参照してください。

AWS IoT Events アラームは、 で検出するアラームです AWS IoT Events。 はアセットプロパティ値をアラームモデル AWS IoT SiteWise に送信します AWS IoT Events。次に、 はアラーム状態を AWS IoT Events に送信します AWS IoT SiteWise。アラームが検出されるタイミングや、アラームの状態が変化したときの通知先などのオプションを設定することができます。また、アラームの状態が変化したときに発生する [AWS IoT Events actions] (アクション) を定義することができます。

のアラーム AWS IoT Events は、アラームモデルのインスタンスです。アラームモデルは、アラームのしきい値や重要度、アラームの状態が変化したときの処理などを指定します。アラームモデルの各属性を設定するとき、アラームがモニタリングするアセットモデルから属性プロパティを指定します。アセットモデルに基づくすべてのアセットは、 がアラームの特性 AWS IoT Events を評価するときに 属性の値を使用します。詳細については、[AWS IoT Events Developer Guide] (デベロッパーガイド) の[Using alarms] (アラームの使用) を参照してください。

状態が変更されたときに AWS IoT Events アラームに応答できます。例えば、アラームがアクティブになったときに、それを確認したり、スヌーズしたりすることができます。また、アラームの有効化、無効化、リセットも可能です。

SiteWise Monitor ユーザーは、SiteWise Monitor ポータルで AWS IoT Events アラームを視覚化、設定、応答できます。詳細については、[AWS IoT SiteWise Monitor Developer Guide] (デベロッパーガイド) の[Monitoring with alarms] (アラームによるモニタリング) を参照してください。

> **Note**
>
> AWS IoT Events 料金は、これらのアラームを評価し、 AWS IoT SiteWise と の間でデータを転送するために適用されます AWS IoT Events。詳細については、「AWS IoT Events 料金表」を参照してください。

- 外部アラーム

  外部アラームは、外部で評価するアラームです AWS IoT SiteWise。アラームの状態を報告するデータソースがある場合は、外部アラームを使用します。外部アラームは、アラーム状態データを取り込む測定プロパティを含んでいます。

  外部アラームの状態が変わっても、確認やスヌーズはできません。

  SiteWise Monitor のユーザーは、SiteWise Monitor のポータルで外部アラームの状態を見ることができますが、これらのアラームを構成したり応答したりすることはできません。

AWS IoT SiteWise は外部アラームの状態を評価しません。

アラームの状態

産業用アラームには、モニタリングする機器やプロセスの状態に関する情報と、アラーム状態に対するオペレータの対応に関する情報(オプション) が含まれています。

AWS IoT Events アラームを定義するときは、確認フローを有効にするかどうかを指定します。フロー承認はデフォルトで有効になっています。このオプションを有効にすると、オペレーターはアラームを承認し、アラームに関する詳細や対処方法をメモに残すことができます。アクティブなアラームが非アクティブになる前にオペレーターが承認しないと、アラームはラッチされた状態になります。ラッチされた状態は、アラームがアクティブになり、承認されなかったことを示します。したがって、オペレータは装置またはプロセスをチェックし、ラッチされたアラームを承認する必要があります。

アラームには次の状態があります。

- [正常] (`Normal`) – アラームは有効ですが、非アクティブです。工業プロセスや機器が期待通りに動作している。
- [アクティブ] (`Active`) – アラームはアクティブです。産業プロセスまたは装置が動作範囲外にあり、注意が必要である。
- [承認済み] (`Acknowledged`) – オペレータがアラームの状態を承認しました。

  この状態は、承認フローを有効にしたアラームにのみ適用されます。
- [ラッチされた] (`Latched`) – アラームは正常に戻ったが、アクティブであり、オペレーターはそれを承認しなかった。産業プロセスや機器では、アラームを正常に戻すためにオペレーターの注意が必要です。

  この状態は、承認フローを有効にしたアラームにのみ適用されます。
- [スヌーズ] (`SnoozeDisabled`) – オペレータがアラームをスヌーズしたため、アラームは無効になります。アラームをスヌーズさせる時間をオペレータが定義します。この時間が経過すると、アラームは通常状態に戻ります。
- [無効] (`Disabled`) — アラームは無効になり、検出されません。

アラーム状態のプロパティ

AWS IoT SiteWise は、アラーム状態データを文字列にシリアル化された JSON オブジェクトとして保存します。このオブジェクトは、アラームの状態、およびオペレータの応答アクションやアラームが評価するルールなどの追加情報を含みます。

アラーム状態のプロパティは、その名前と構造体型である AWS/ALARM_STATE によって識別されます。詳細については、「でアセットモデルにアラームを定義する AWS IoT SiteWise」を参照してください。

アラーム状態データオブジェクトは、次の情報が含まれています。

stateName

アラームの状態。詳細については、「アラームの状態」を参照してください。

データ型: STRING

customerAction

(オプション) アラームに対するオペレータの応答に関する情報を含むオブジェクト。オペレータは、アラームの有効化、無効化、確認、スヌーズを行うことができます。その際、アラームの状態データには、彼らの応答と、応答時に残すことのできるメモが含まれます。このオブジェクトには、次の情報が含まれます。

actionName

オペレータがアラームに対応するために取るアクションの名前。この値には、次の文字列のいずれかが含まれます。

- ENABLE
- DISABLE
- SNOOZE
- ACKNOWLEDGE
- RESET

データ型: STRING

enable

(オプション) オペレータがアラームを有効にしたときに、customerAction に存在するオブジェクトです。オペレータがアラームを有効にすると、アラーム状態は Normal になります。このオブジェクトには、次の情報が含まれます。

note

(オプション) 顧客がアラームを有効にする際に残すメモ。

データ型: STRING

最大文字数: 128 文字

disable

(オプション) オペレータがアラームを無効にしたときに customerAction に存在するオブジェクト。オペレータがアラームを有効にすると、アラーム状態は Disabled になります。このオブジェクトには、次の情報が含まれます。

note

(オプション) 顧客がアラームを無効にする際に残すメモ。

データ型: STRING

最大文字数: 128 文字

acknowledge

(オプション) オペレータがアラームを承認するときに customerAction に存在するオブジェクト。オペレータがアラームを有効にすると、アラーム状態は Acknowledged になります。このオブジェクトには、次の情報が含まれます。

note

(オプション) 顧客がアラームを確認する際に残すメモ。

データ型: STRING

最大文字数: 128 文字

snooze

(オプション) オペレータがアラームをスヌーズしたときに customerAction に存在するオブジェクト。オペレータがアラームを有効にすると、アラーム状態は SnoozeDisabled になります。このオブジェクトには、次の情報が含まれます。

snoozeDuration

オペレータがアラームをスヌーズする時間 (秒)。この時間経過後、アラームは Normal 状態になります。

データ型: INTEGER

note

(オプション) お客様がアラームをスヌーズした際に残すメモです。

データ型: STRING

最大文字数: 128 文字

ruleEvaluation

(オプション) アラームを評価するルールに関する情報を含むオブジェクトです。このオブジェクトには、次の情報が含まれます。

simpleRule

プロパティ値としきい値を比較演算子で比較する、単純なルールに関する情報を持つオブジェクト。このオブジェクトには、次の情報が含まれます。

inputProperty

このアラームが評価するプロパティの値。

データ型: DOUBLE

operator

このアラームがプロパティをしきい値を比較するために使用する比較演算子。この値には、次の文字列のいずれかが含まれます。

- < - 未満
- <= - 次
- == - 等しい。
- != - 等しくない。
- >= - 以上
- > - 次より大きい

データ型: STRING

threshold

プロパティ値に対して、本アラームが比較するしきい値を設定する。

データ型: DOUBLE

でアセットモデルにアラームを定義する AWS IoT SiteWise

アセットモデルは、産業用データとアラームの標準化を推進します。アセットモデルにアラーム定義を行うことで、アセットモデルに基づくすべてのアセットのアラームを標準化することができます。

アセットモデルにアラームを定義するには、[composite asset model] (複合アセットモデル) を使用します。複合アセットモデルとは、他のアセットモデルに特定のプロパティのセットを標準化したアセットモデルです。複合アセットモデルは、アセットモデルに特定のプロパティが存在することを保証します。アラームは型、ステート、および (オプションの) 出典プロパティを持つため、アラーム複合モデルはこれらのプロパティの存在を強制する。

各複合アセットモデルには、その複合モデルのプロパティを定義する型があります。アラーム複合モデルは、アラーム型、アラーム状態、および (オプションの) アラーム出典のプロパティを定義します。複合モデルを使ってアセットモデルからアセットを作成すると、アセットモデルで指定したプロパティに加えて、複合モデルのプロパティもアセットに含まれます。

複合モデル内の各プロパティは、その複合モデルの型に応じて識別できる名前を持つ必要があります。複合モデルプロパティは、複雑なデータ型を持つプロパティに対応しています。これらのプロパティは、`STRUCT` データ型と、プロパティの複合データ型を指定する `dataTypeSpec` 特性を持つ。複合データ型プロパティは、文字列としてシリアライズされた JSON データを含む。

アラーム複合モデルは次のような特性を持ちます。各プロパティは、この型の複合モデル用に識別するための名前を持つ必要があります。

アラーム型

アラームの型。次のいずれかを指定します。

- `IOT_EVENTS` – AWS IoT Events alarm. AWS IoT SiteWise はデータを に送信 AWS IoT Events して、このアラームの状態を評価します。このアラーム定義の AWS IoT Events アラームモデルを定義するには、アラームソースプロパティを指定する必要があります。

  > **Note**
  >
  > サポート終了通知: 2026 年 5 月 20 日に、 AWS は のサポートを終了します AWS IoT Events。2026 年 5 月 20 日以降、 AWS IoT Events コンソールまたは AWS IoT Events リソースにアクセスできなくなります。詳細については、AWS IoT Events 「サポート終了」を参照してください。

- `EXTERNAL` - 外部アラーム アラームの状態を測定値として取り込みます。

プロパティ名: AWS/ALARM_TYPE

プロパティ型: [attribute] (属性)

データ型: STRING

[Alarm state] (アラームの状態)

アラームの状態を表す時系列データです。これは文字列としてシリアライズされたオブジェクトであり、アラームの状態やその他の情報を含む。詳細については、「アラーム状態のプロパティ」を参照してください。

プロパティ名: AWS/ALARM_STATE

プロパティの種類: [measurement] (測定)

データ型: STRUCT

データ構造型:AWS/ALARM_STATE

[Alarm source] (アラーム出典)

(オプション) アラームの状態を評価するリソースの Amazon リソースネーム (ARN) です。 AWS IoT Events アラームの場合、これはアラームモデルの ARN です。

プロパティ名: AWS/ALARM_SOURCE

プロパティ型: [attribute] (属性)

データ型: STRING

Example アラーム複合モデルの例

次のアセットモデルは、温度をモニタリングするアラームを持つボイラーを表します。 は温度データを AWS IoT SiteWise に送信 AWS IoT Events してアラームを検出します。

```
{
  "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": {}
          }
        }
```

```
        ]
      }
    ]
  }
```

トピック

- でのアラーム通知の要件 AWS IoT SiteWise
- の AWS IoT Events アラームを定義する AWS IoT SiteWise
- で外部アラームを定義する AWS IoT SiteWise

でのアラーム通知の要件 AWS IoT SiteWise

AWS IoT Events は、 AWS アカウントの AWS Lambda 関数を使用してアラーム通知を送信します。アラーム通知を有効にするには、アラームと同じ AWS リージョンにこの Lambda 関数を作成する必要があります。このLambda関数は、テキスト通知の送信に[Amazon Simple Notification Service (Amazon SNS)]、メール通知の送信に[Amazon Simple Email Service (Amazon SES)]を使用しています。 AWS IoT Events アラームを作成するときは、アラームが通知を送信するために使用するプロトコルと設定を設定します。

> **Note**
>
> サポート終了通知: 2026 年 5 月 20 日に、 AWS は のサポートを終了します AWS IoT Events。2026 年 5 月 20 日以降、 AWS IoT Events コンソールまたは AWS IoT Events リソースにアクセスできなくなります。詳細については、AWS IoT Events 「サポート終了」を参照してください。

AWS IoT Events には、 アカウントでこの Lambda 関数を作成するために使用できる AWS CloudFormation スタックテンプレートが用意されています。詳細については、[AWS IoT Events Developer Guide] (デベロッパーガイド) の[Alarm notification Lambda function] (アラーム通知 Lambda 関数) を参照してください。

の AWS IoT Events アラームを定義する AWS IoT SiteWise

> **Note**
>
> サポート終了通知: 2026 年 5 月 20 日に、 AWS は のサポートを終了します AWS IoT Events。2026 年 5 月 20 日以降、 AWS IoT Events コンソールまたは AWS IoT Events リソースにアクセスできなくなります。詳細については、AWS IoT Events 「サポート終了」を参照してください。

AWS IoT Events アラームを作成すると、 はアセットプロパティ値を AWS IoT SiteWise に送信 AWS IoT Events して、アラームの状態を評価します。 AWS IoT Events アラーム定義は、 で定義したアラームモデルによって異なります AWS IoT Events。アセットモデルで AWS IoT Events アラームを定義するには、アラームソースプロパティとして AWS IoT Events アラームモデルを指定するアラーム複合モデルを定義します。

AWS IoT Events アラームは、アラームしきい値やアラーム通知設定などの入力によって異なります。これらの入力は、アセットモデル上の属性として定義します。そして、これらの入力をモデルに基づいて各アセットでカスタマイズすることができます。 AWS IoT SiteWise コンソールでこれらの属性を作成できます。 AWS CLI または API でアラームを定義する場合は、アセットモデルでこれらの属性を手動で定義する必要があります。

また、カスタムアラーム通知アクションなど、アラームが検出されたときに発生するその他のアクションを定義することも可能です。例えば、Amazon SNSのトピックにプッシュ通知を送信するアクションを設定することができます。定義できるアクションの詳細については、「 AWS IoT Events デベロッパーガイド」の「他の AWS のサービスの使用」を参照してください。

アセットモデルを更新または削除すると、 の AWS IoT Events アラームモデルがこのアセットモデルに関連付けられたアセットプロパティをモニタリングしているかどうか AWS IoT SiteWise を確認できます。これにより、 AWS IoT Events アラームが現在使用しているアセットプロパティを削除できなくなります。でこの機能を有効にするには AWS IoT SiteWise、 アクセス`iotevents:ListInputRoutings`許可が必要です。このアクセス許可により AWS IoT SiteWise 、 は でサポートされている ListInputRoutings API オペレーションを呼び出すことができます AWS IoT Events。詳細については、「(オプション) ListInputRoutingsの許可。」を参照してください。

> **Note**
> アラーム通知機能は、中国 (北京) リージョンでは利用できません。

トピック

- AWS IoT Events アラームを定義する (AWS IoT SiteWise コンソール)
- AWS IoT Events アラームを定義する (AWS IoT Events コンソール)
- AWS IoT Events アラームを定義する (AWS CLI)

AWS IoT Events アラームを定義する (AWS IoT SiteWise コンソール)

AWS IoT SiteWise コンソールを使用して、既存のアセットモデルに AWS IoT Events アラームを定義できます。新しいアセットモデルで AWS IoT Events アラームを定義するには、アセットモデルを作成し、以下の手順を実行します。詳細については、「でアセットモデルを作成する AWS IoT SiteWise」を参照してください。

> **Important**
> 各アラームには、そのアラームの比較対象となるしきい値を指定する属性が必要です。アラームを定義する前に、アセットモデルでしきい値属性を定義する必要があります。
> 例えば、風力タービンが定格最大風速 50mph を超えた場合にアラームを発生させたい場合を考えてみましょう。アラームを定義する前に、デフォルト値が 50 の属性 ([最大風速]) を定義する必要があります。

アセットモデルで AWS IoT Events アラームを定義するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. アラームを定義するアセットモデルを選択します。
4. [アラーム] タブを選択します。
5. [アラームの追加] を選択します。
6. [アラーム型のオプション] セクションで、[AWS IoT Events アラーム] を選択します。
7. [アラームの詳細] セクションで、次の手順を実行します。

a. アラームの名前を入力します。

b. (オプション) アラームの説明を入力します。

8. [しきい値定義] セクションでは、アラーム検出のタイミングとアラームの重要度を定義します。以下の操作を実行します。

   a. アラームを検出した[プロパティ] を選択します。このプロパティが新しい値を受信するたびに、 は値を AWS IoT SiteWise に送信 AWS IoT Events してアラームの状態を評価します。

   b. プロパティとしきい値の比較に使用する [演算子] を選択します。次のオプションから選択します。

      - [< 未満]
      - [<= より小さい、または等しい]
      - == 等しい
      - != 等しくない
      - [>= より大きい、または等しい]
      - [> より大きい]

   c. Value には、しきい値として使用する属性プロパティを選択します。 AWS IoT Events は、プロパティの値をこの属性の値と比較します。

   d. アラームの[Severity] (重要度) を入力します。このアラームの重要度を反映させるために、チームが理解できる数字を使用してください。

9. (オプション) [通知設定 - オプション] セクションで、次の操作を行います。

   a. [アクティブ] を選択します。

      > **Note**
      > [非アクティブ] を選択した場合、アラーム通知は送信されません。

   b. [受信者] では、受信者を選択します。

      > **Important**
      > アラーム通知を AWS IAM Identity Center ユーザーに送信できます。この機能を使用するには、IAM Identity Center を有効にする必要があります。IAM アイデンティティセンターは、一度に 1 つの AWS リージョンでのみ有効にできます。このこ

とは、IAM Identity Center を有効にしたリージョンでのみ、アラーム通知を定義できることを意味します。詳細については、[AWS IAM Identity Center User Guide] (ユーザーガイド) の[Getting started] (使用開始) を参照してください。

c. [Protocol] (プロトコル) については、次のオプションから選択してください。

- E メールとテキスト – SMS メッセージと E メールで IAM Identity Center ユーザーに通知します。
- E メール – E メールで IAM Identity Center ユーザーに通知します。
- テキスト – SMS メッセージで IAM Identity Center ユーザーに通知します。

d. [送信者] は、送信者を選択します。

> **Important**
>
> Amazon Simple Email Service (Amazon SES) で送信者メールアドレスの確認が必要です。詳細については、「Amazon Simple Email Service デベロッパーガイド」の「E メールアドレスのアイデンティティの確認」を参照してください。

10. [デフォルトのアセット状態] セクションでは、このアセットモデルから作成されたアラームに対するデフォルトの状態を設定できます。

> **Note**
>
> このアセットモデルから作成するすべてのアセットに対して、後のステップでこのアラームを有効または無効にすることができます。

11. [詳細設定] セクションでは、アクセス許可、追加の通知設定、アラーム状態のアクション、SiteWise Monitor のアラームモデル、および承認フローを設定できます。

> **Note**
>
> AWS IoT Events アラームには、次のサービスロールが必要です。
>
> - がアラーム状態値の送信先として AWS IoT Events 引き受けるロール AWS IoT SiteWise。
> - Lambda にデータを送信するために が AWS IoT Events 引き受けるロール。このロールは、アラームが通知を送信する場合にのみ必要です。

[許可] セクションで、以下の操作を行います。

a. [AWS IoT Events ロール] の場合、既存のロールを使用するか、必要なアクセス許可を持つロールを作成します。このロールには、`iotsitewise:BatchPutAssetPropertyValue` アクセス許可と、 iotevents.amazonaws.com がロールを引き受けることを許可する信頼関係が必要です。

b. [AWS IoT Events Lambda ロール] については、既存のロールを使用するか、必要なアクセス許可を持つロールを作成します。このロールには、`lambda:InvokeFunction`、 `sso-directory:DescribeUser` のアクセス許可と、`iotevents.amazonaws.com` がそのロールを担うことができる信頼関係が必要です。

12. (オプション) [追加の通知設定] セクションで、次を実行します。

a. [受信者属性] には、通知の受信者を指定する値を持つ属性を定義します。IAM Identity Center ユーザーを受信者として選択することができます。

属性を作成するか、アセットモデル上の既存の属性を使用することができます。

- [新規受信者属性を作成] を選択した場合は、受信者属性名とその属性に対する受信者のデフォルト値 - (オプション) を指定します。
- 「既存の属性を使用する」を選択した場合は、[受信者の属性名] で属性を選択します。アラームは、選択した属性のデフォルト値を使用します。

このアセットモデルから作成する各アセットで、デフォルト値を上書きすることができます。

b. [カスタムメッセージ属性] には、デフォルトの状態変化メッセージに加えて、送信するカスタムメッセージを指定する属性を定義します。例えば、このアラームの対処方法をチームが理解するのに役立つメッセージを指定することができます。

属性を作成するか、アセットモデル上の既存の属性を使用するかを選択することができます。

- 「新しいカスタムメッセージ属性を作成する」を選択した場合は、その属性に対するカスタムメッセージ属性名とカスタムメッセージのデフォルト値 - オプションを指定します。
- 「既存の属性を使用する」を選択した場合は、[カスタムメッセージの属性名] でその属性を選択します。アラームは、選択した属性のデフォルト値を使用します。

このアセットモデルから作成する各アセットで、デフォルト値を上書きすることができます。

c. [Lambda 関数の管理] では、次のいずれかを実行します。

- で新しい Lambda 関数 AWS IoT SiteWise を作成するには、AWS マネージドテンプレートから新しい Lambda を作成するを選択します。
- 既存の Lambda 関数を使用するには、[既存の Lambda を使用する] を選択して、関数の名前を選択します。

詳細については、AWS IoT Events デベロッパーガイドの「アラーム通知の管理」を参照してください。

13. (オプション) [状態アクションの設定] セクションでは、次を実行します。

a. [アクションを編集] を選択します。

b. [アラーム状態アクションの追加] でアクションを追加して、[保存] を選択します。

アクションは最大10個まで追加可能です。

AWS IoT Events は、アラームがアクティブなときにアクションを実行できます。組み込みアクションを定義して、タイマーを使用したり、変数を設定したり、他の AWS リソースにデータを送信したりできます。詳細については、[AWS IoT Events Developer Guide] (デベロッパーガイド) の[Supported actions] (対応アクション) を参照してください。

14. (オプション) [SiteWise Monitor でアラームモデルの管理 - オプション] でアクティブまたは非アクティブを選択します。

このオプションを使用すると、SiteWise Monitorss でアラームモデルを更新することができます。デフォルトでは、このオプションは有効になっています。

15. [承認フロー] で、アクティブまたは非アクティブを選択します。フロー承認の詳細については、アラームの状態 を参照してください。

16. [アラームの追加] を選択します。

> **Note**
>
> AWS IoT SiteWise コンソールは、複数の API リクエストを実行して、アラームをアセットモデルに追加します。[アラームの追加] を選択すると、コンソールがダイアログ

ボックスを開き、これらの API リクエストの進捗状況が表示されます。各 API リクエストが成功するまで、または API リクエストが失敗するまで、このページに滞在してください。リクエストに失敗した場合は、ダイアログボックスを閉じて問題を修正し、[アラームの追加] を選択して再試行してください。

AWS IoT Events アラームを定義する (AWS IoT Events コンソール)

AWS IoT Events コンソールを使用して、既存のアセットモデルに AWS IoT Events アラームを定義できます。新しいアセットモデルで AWS IoT Events アラームを定義するには、アセットモデルを作成し、以下の手順を実行します。詳細については、「でアセットモデルを作成する AWS IoT SiteWise」を参照してください。

> **Important**
>
> 各アラームには、そのアラームの比較対象となるしきい値を指定する属性が必要です。アラームを定義する前に、アセットモデルでしきい値属性を定義する必要があります。
> 例えば、風力タービンが定格最大風速 50mph を超えた場合にアラームを発生させたい場合を考えてみましょう。アラームを定義する前に、デフォルト値が `50` の属性 ([最大風速]) を定義する必要があります。

アセットモデルで AWS IoT Events アラームを定義するには

1. AWS IoT Events コンソールに移動します。
2. ナビゲーションペインで、アラームモデルを選択します。
3. [アラームモデルの作成] を選択します。
4. アラームの名前を入力します。
5. (オプション) アラームの説明を入力します。
6. アラームターゲットセクションで、次の手順を実行します。
   a. ターゲットオプションでは、AWS IoT SiteWise アセットプロパティを選択します。
   b. アラームを追加するアセットモデルを選択します。
7. [しきい値定義] セクションでは、アラーム検出のタイミングとアラームの重要度を定義します。以下の操作を実行します。

a. アラームを検出した[プロパティ] を選択します。このプロパティが新しい値を受信するたびに、 は値を AWS IoT SiteWise に送信 AWS IoT Events してアラームの状態を評価します。

b. プロパティとしきい値の比較に使用する [演算子] を選択します。次のオプションから選択します。

- [< 未満]
- [<= より小さい、または等しい]
- == 等しい
- != 等しくない
- [>= より大きい、または等しい]
- [> より大きい]

c. Value には、しきい値として使用する属性プロパティを選択します。 AWS IoT Events は、プロパティの値をこの属性の値と比較します。

d. アラームの[Severity] (重要度) を入力します。このアラームの重要度を反映させるために、チームが理解できる数字を使用してください。

8. (オプション) [通知設定 - オプション] セクションで、次の操作を行います。

a. [プロトコル] については、次のオプションから選択してください。

- E メールとテキスト – SMS メッセージと E メールで IAM Identity Center ユーザーに通知します。
- E メール – E メールで IAM Identity Center ユーザーに通知します。
- テキスト – SMS メッセージで IAM Identity Center ユーザーに通知します。

b. [送信者] は、送信者を選択します。

> **Important**
>
> Amazon Simple Email Service (Amazon SES) で送信者メールアドレスの確認が必要です。詳細については、[Amazon Simple Email Service Developer Guide] (Amazon Simple Email Service デベロッパーガイド) の[Verifying email addresses in Amazon SES] (Amazon SES でのメールアドレスの確認) を参照してください。

c. [受信者属性 - オプション] で属性を選択します。アラームは、選択した属性のデフォルト値を使用します。

d. [カスタムメッセージ属性 - オプション] で属性を選択します。アラームは、選択した属性のデフォルト値を使用します。

9. [インスタンス] セクションで、このアラームのデフォルト状態を指定します。後のステップで、このアセットモデルから作成するすべてのアセットに対して、このアラームを有効または無効にすることができます。

10. [詳細設定] の設定では、アクセス許可、追加の通知設定、アラーム状態のアクション、SiteWise Monitor のアラームモデル、および承認フローを構成できます。

> **Note**
>
> AWS IoT Events アラームには、次のサービスロールが必要です。
>
> - がアラーム状態値の送信を AWS IoT Events 引き受けるロール AWS IoT SiteWise。
> - Lambda にデータを送信するために が AWS IoT Events 引き受けるロール。このロールは、アラームが通知を送信する場合にのみ必要です。

a. [承認フロー] セクションで有効または無効を選択します。フロー承認の詳細については、アラームの状態 を参照してください。

b. [許可] セクションで、以下の操作を行います。

   i. [AWS IoT Events ロール] の場合、既存のロールを使用するか、必要なアクセス許可を持つロールを作成します。このロールには、`iotsitewise:BatchPutAssetPropertyValue` アクセス許可と、iotevents.amazonaws.com がロールを引き受けることを許可する信頼関係が必要です。

   ii. [Lambda ロール] については、既存のロールを使用するか、必要なアクセス許可を持つロールを作成します。このロールには、`lambda:InvokeFunction`、 `sso-directory:DescribeUser` のアクセス許可と、`iotevents.amazonaws.com` がそのロールを担うことができる信頼関係が必要です。

c. (オプション) [Additional notification settings] (追加の通知設定) ペインで、次を実行します。

   - [Lambda 関数の管理] では、次のいずれかを実行します。

     - で新しい Lambda 関数 AWS IoT Events を作成するには、「新しい Lambda 関数を作成する」を選択します。

- 既存の Lambda 関数を使用するには、[既存の Lambda 関数を使用する] を選択して、関数の名前を選択します。

詳細については、AWS IoT Events デベロッパーガイドの「アラーム通知の管理」を参照してください。

d. (オプション) [状態アクションの設定 - オプション] では、次を実行します。

- [アラーム状態アクション] でアクションを追加して、[保存] を選択します。

  アクションは最大10個まで追加可能です。

AWS IoT Events は、アラームがアクティブなときにアクションを実行できます。組み込みアクションを定義して、タイマーを使用したり、変数を設定したり、他の AWS リソースにデータを送信したりできます。詳細については、[AWS IoT Events Developer Guide] (デベロッパーガイド) の[Supported actions] (対応アクション) を参照してください。

11. [作成] を選択します。

> **Note**
>
> AWS IoT Events コンソールは、複数の API リクエストを実行して、アラームをアセットモデルに追加します。[アラームの追加] を選択すると、コンソールがダイアログボックスを開き、これらの API リクエストの進捗状況が表示されます。各 API リクエストが成功するまで、または API リクエストが失敗するまで、このページに滞在してください。リクエストに失敗した場合は、ダイアログボックスを閉じて問題を修正し、[アラームの追加] を選択して再試行してください。

AWS IoT Events アラームを定義する (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用して、アセットプロパティをモニタリングする AWS IoT Events アラームを定義できます。新規または既存のアセットモデルにアラームを定義することができます。アセットモデルでアラームを定義したら、 でアラームを作成し AWS IoT Events 、アセットモデルに接続します。このプロセスでは、次のことを行います。

ステップ

- ステップ 1: アセットモデルにアラームを定義する

- ステップ 2: AWS IoT Events アラームモデルを定義する
- ステップ 3: AWS IoT SiteWise と 間のデータフローを有効にする AWS IoT Events

ステップ 1: アセットモデルにアラームを定義する

新規または既存のアセットモデルに、アラーム定義と関連プロパティを追加します。

アセットモデルにアラームを定義するには (CLI)。

1. `asset-model-payload.json` という名前のファイルを作成します。これらの他のセクションのステップに従って、アセットモデルの詳細をファイルに追加しますが、アセットモデルの作成または更新のリクエストは送信しないでください。このセクションでは、`asset-model-payload.json` ファイルのアセットモデル詳細にアラーム定義を追加します。
   - アセットモデルの作成方法については、アセットモデルを作成する (AWS CLI) を参照してください。
   - 既存のアセットモデルを更新する方法については、アセットまたはコンポーネントモデルを更新する (AWS CLI) を参照してください。

   > **Note**
   > アセットモデルには、アラームでモニタリングするアセットプロパティを含め、少なくとも1つのアセットプロパティが定義されている必要があります。

2. アセットモデルにアラーム複合モデル (`assetModelCompositeModels`) を追加する。 AWS IoT Events アラーム複合モデルは `IOT_EVENTS`タイプを指定し、アラームソースプロパティを指定します。アラームソースプロパティは、アラームモデルを作成した後に追加します AWS IoT Events。

   > **Important**
   > アラーム複合モデルは、後で作成する AWS IoT Events アラームモデルと同じ名前である必要があります。アラームモデル名には、英数字のみ使用できます。アラームモデルに同じ名前を使用できるように、一意な英数字の名前を指定します。

```
{
  ...
```

```
  "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. アセットモデルにアラームしきい値属性を追加する。このしきい値に使用するデフォルト値を指定します。このモデルに基づいて、各アセットでこのデフォルト値を上書きすることができます。

> **Note**
>
> アラームしきい値属性は、INTEGER または DOUBLE でなければならない。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "Temperature Max Threshold",
```

```
      "dataType": "DOUBLE",
      "type": {
        "attribute": {
          "defaultValue": "105.0"
        }
      }
    }
  ]
}
```

4. (オプション) アセットモデルにアラーム通知属性を追加します。これらの属性は、アラームの状態が変更されたときに AWS IoT Events が通知を送信するために使用する IAM Identity Center の受信者およびその他の入力を指定します。このモデルに基づいて、各アセットでこれらのデフォルトをオーバーライドすることができます。

> **Important**
>
> アラーム通知を AWS IAM Identity Center ユーザーに送信できます。この機能を使用するには、IAM Identity Center を有効にする必要があります。IAM アイデンティティセンターは、一度に 1 つの AWS リージョンでのみ有効にできます。このことは、IAM Identity Center を有効にしたリージョンでのみ、アラーム通知を定義できることを意味します。詳細については、[AWS IAM Identity Center User Guide] (ユーザーガイド) の[Getting started] (使用開始) を参照してください。

以下の操作を実行します。

a. IAM Identity Center ID ストアの ID を指定する属性を追加します。IAM Identity Center の ListInstances API オペレーションを使用して、ID ストアを一覧表示することができます。このオペレーションは、IAM Identity Center を有効にしたリージョンでのみ機能します。

```
aws sso-admin list-instances
```

次に、ID ストア ID (例: `d-123EXAMPLE`) を属性のデフォルト値として指定します。

```
{
  ...
  "assetModelProperties": [
    ...
    {
```

```
      "name": "identityStoreId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "d-123EXAMPLE"
        }
      }
    }
  ]
}
```

b. 通知の送信先の IAM Identity Center ユーザーの ID を指定する属性を追加します。デフォルトの通知受信者を定義するには、デフォルト値として IAM Identity Center ユーザー ID を追加します。次のいずれかを行って、IAM Identity Center ユーザー ID を取得します。

   i. IAM Identity Center の ListUsers API を使用すると、ユーザー名がわかっているユーザーの ID を取得することができます。*d-123EXAMPLE* は ID ストアの ID に、*[Name]* (名前) はユーザーのユーザー名に置き換えてください。

```
aws identitystore list-users \
  --identity-store-id d-123EXAMPLE \
  --filters AttributePath=UserName,AttributeValue=Name
```

   ii. IAM Identity Center コンソール を使用してユーザーをブラウズして、ユーザー ID を探することができます。

そして、ユーザー ID (例えば、`123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE`) を属性のデフォルト値として指定するか、デフォルト値なしで属性を定義します。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "userId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "123EXAMPLE-a1b2c3d4-5678-90ab-cdef-33333EXAMPLE"
        }
```

```
      }
    }
  ]
}
```

c. (オプション) SMS (テキスト) メッセージ通知のデフォルトの送信者 ID を指定する属性を追加します。Amazon Simple Notification Service (Amazon SNS) が送信するメッセージの送信者として、送信者 ID が表示されます。詳細については、「 AWS End User Messaging SMS ユーザーガイド」の「 で送信者 ID をリクエスト AWS End User Messaging SMSする」を参照してください。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "senderId",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "MyFactory"
        }
      }
    }
  ]
}
```

d. (オプション) E メール通知で[from] (送信元) アドレスとして使用するデフォルトのE メールアドレスを指定する属性を追加します。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "fromAddress",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "my.factory@example.com"
        }
      }
```

```
    }
  ]
}
```

e. (オプション) E メール通知で使用するデフォルトの件名を指定する属性を追加します。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "emailSubject",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "[ALERT] High boiler temperature"
        }
      }
    }
  ]
}
```

f. (オプション) 通知に含める追加メッセージを指定する属性を追加します。デフォルトでは、通知メッセージはアラームに関する情報を含んでいます。また、ユーザーにより多くの情報を与える追加メッセージを含めることもできます。

```
{
  ...
  "assetModelProperties": [
    ...
    {
      "name": "additionalMessage",
      "dataType": "STRING",
      "type": {
        "attribute": {
          "defaultValue": "Turn off the power before you check the alarm."
        }
      }
    }
  ]
}
```

5. アセットモデルを作成、または既存のアセットモデルを更新します。次のいずれかを行います：

- アセットモデルを作成するには、次のコマンドを実行します。

```
aws iotsitewise create-asset-model --cli-input-json file://asset-model-
payload.json
```

- 既存のアセットモデルを更新するには、次のコマンドを実行します。*asset-model-id* をアセットモデルの ID に置き換えます。

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://asset-model-payload.json
```

コマンドを実行したら、レスポンスに `assetModelId` に注意してください。

例:ボイラーアセットモデル

次のアセットモデルは、温度データを報告するボイラーを表しています。このアセットモデルでは、ボイラーのオーバーヒートを検出するアラームを定義しています。

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

ステップ 2: AWS IoT Events アラームモデルを定義する

でアラームモデルを作成します AWS IoT Events。では AWS IoT Events、式を使用してアラームモデルの値を指定します。式を使用して の値を指定 AWS IoT SiteWise し、アラームへの入力として評価して使用できます。がアセットプロパティ値をアラームモデル AWS IoT SiteWise に送信すると、 AWS IoT Events は式を評価してプロパティの値またはアセットの ID を取得します。アラームモデルでは、次のような表現が可能です。

- アセットプロパティ値

  アセットプロパティの値を取得するには、次の表現を使用します。*assetModelId* をアセットモデルの ID に、*propertyId* をプロパティの ID に置き換えてください。

  ```
  $sitewise.assetModel.`assetModelId`.`propertyId`.propertyValue.value
  ```

- [Asset IDs] (アセット ID)

  アセットのIDを取得するには、次の表現を使用します。*assetModelId* をアセットモデルの ID に、*propertyId* をプロパティの ID に置き換えてください。

  ```
  $sitewise.assetModel.`assetModelId`.`propertyId`.assetId
  ```

> **Note**
>
> アラームモデルを作成するときは、 AWS IoT SiteWise 値に評価される式の代わりにリテラルを定義できます。これにより、アセットモデルに定義する属性の数を減らすことができます。ただし、値をリテラルで定義した場合、アセットモデルに基づいてアセットでその値をカスタマイズすることはできません。また、 AWS IoT SiteWise Monitor ユーザーはアセットに対してのみアラーム設定を設定できるため、アラームをカスタマイズすることもできません。

AWS IoT Events アラームモデルを作成するには (CLI)

1. でアラームモデルを作成するときは AWS IoT Events、アラームが使用する各プロパティの ID を指定する必要があります。これには、以下が含まれます。

   - 複合アセットモデルにおけるアラーム状態プロパティ。

- アラームがモニタリングしているプロパティ。
- しきい値属性
- (オプション) IAM Identity Center ID ストア ID 属性
- (オプション) IAM Identity Center ユーザー ID 属性
- (オプション) SMS 送信者 ID 属性
- (オプション) E メールの[from] (送信元) アドレス属性
- (オプション) E メールの件名属性
- (オプション) 追加のメッセージ属性

次のコマンドを実行し、アセットモデル上のこれらのプロパティの ID を取得します。*asset-model-id* を前のステップのアセットモデルの ID に置き換えてください。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

このオペレーションは、アセットモデルの詳細を含むレスポンスを返します。アラームが使用する各プロパティの ID を記録しておきます。これらの ID は、次のステップで AWS IoT Events アラームモデルを作成するときに使用します。

2. でアラームモデルを作成します AWS IoT Events。以下の操作を実行します。

   a. `alarm-model-payload.json` という名前のファイルを作成します。

   b. 次の JSON オブジェクトをファイルにコピーしてください。

   c. アラームの名前 (`alarmModelName`) 、説明 (`alarmModelDescription`) 、重要度 (`severity`) を入力します。重要度には、お客様の会社の重要度レベルを反映した整数を指定します。

   > **Important**
   > アラームモデルは、先にアセットモデルで定義したアラーム複合モデルと同じ名前である必要があります。
   > アラームモデル名には、英数字のみ使用できます。

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
```

```
  "severity": 3
}
```

d. アラームに比較ルール (alarmRule) を追加します。このルールでは、モニタリングするプロパティ (inputProperty) 、比較するしきい値 (threshold) 、使用する比較演算子 (comparisonOperator) を定義する。

- *assetModelId* をアセットモデルの ID に置き換えてください。
- *alarmPropertyId* を、アラームがモニタリングするプロパティの ID に置き換える。
- *thresholdAttributeId* をしきい値属性プロパティの ID に置き換える。
- *GREATER* は、プロパティ値としきい値の比較に使用する演算子で置き換えてください。次のオプションから選択します。
  - 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. アラームの状態が変化した時に、 AWS IoT SiteWise にアラームの状態を送信するアクション (alarmEventActions) を追加する。

> **Note**
>
> アドバンストな設定として、アラームの状態が変化したときに実行する追加アクションを定義することができます。例えば、 AWS Lambda 関数を呼び出したり、MQTT トピックに発行したりします。詳細については、「デベロッパーガイド」の「他の AWS のサービスの使用」を参照してください。 AWS IoT Events

- *assetModelId* をアセットモデルの ID に置き換えてください。
- *alarmPropertyId* を、アラームがモニタリングするプロパティの ID に置き換える。
- *alarmStatePropertyId* をアラーム複合モデル内のアラーム状態プロパティの ID で置き換える。

```
{
  "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. (オプション) アラーム通知設定を行います。アラーム通知アクションは、アカウント内のLambda関数を使用して、アラーム通知を送信します。詳細については、「でのアラーム通知の要件 AWS IoT SiteWise」を参照してください。アラーム通知の設定では、IAM Identity Center ユーザーに送信する SMS と E メールの通知を設定できます。以下の操作を実行します。

   i. `alarm-model-payload.json` のペイロードにアラーム通知設定 (`alarmNotification`) を追加する。

      - *`alarmNotificationFunctionArn`* を、アラーム通知を処理する Lambda 関数の ARN に置き換える。

```
{
  "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. (オプション) アラームの状態が変化したときに IAM Identity Center ユーザーに送信する SMS 通知 (`smsConfigurations`) を設定し ます。

- *identityStoreIdAttributeId* を、IAM Identity Center ID ストアの ID を含む属性の ID に置き換えます。
- *userIdAttributeId* を、IAM Identity Center ユーザーの ID を含む属性の ID に置き換えます。
- *senderIdAttributeId* をAmazon SNSの送信者IDを含む属性のIDに置き換えるか、ペイロードから `senderId` を削除してください。
- *additionalMessageAttributeId* を追加メッセージを含む属性の ID で置き換えるか、ペイロードから `additionalMessage` を削除する。

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
```

```
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        },
        "smsConfigurations": [
          {
            "recipients": [
              {
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.va
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                }
              }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.
          }
        ]
      }
    ]
  }
}
```

iii. (オプション) アラームの状態が変化したときに IAM Identity Center ユーザーに送信する E メール通知 (`emailConfigurations`) を設定します。

- *`identityStoreIdAttributeId`* を、IAM Identity Center ID ストア ID 属性プロパティの ID に置き換えます。
- *`userIdAttributeId`* を、IAM Identity Center ユーザー ID 属性プロパティの ID に置き換えます。

- *fromAddressAttributeId* を「from」 アドレス属性プロパティの ID で置き換えるか、ペイロードから from を削除する。
- *emailSubjectAttributeId* をE メールの件名属性プロパティの ID で置き換えるか、ペイロードから subject を削除する。
- *additionalMessageAttributeId* を追加メッセージ属性プロパティの ID で置き換えるか、ペイロードから additionalMessage 削除する。

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        },
        "smsConfigurations": [
          {
```

```
            "recipients": [
              {
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.va
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                }
              }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.
          }
        ],
        "emailConfigurations": [
          {
            "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value"
            "recipients": {
              "to": [
                {
                  "ssoIdentity": {
                    "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.va
                    "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                  }
                }
              ]
            },
            "content": {
              "subject":
"$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value
              "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.
            }
          }
        ]
      }
    ]
  }
```

```
}
```

g. (オプション) alarm-model-payload.jsonのペイロードにアラーム機能 (alarmCapabilities) を追加する。このオブジェクトでは、アセットモデルに基づいて、フロー承認が有効かどうか、およびアセットに対するデフォルトの有効状態を指定することができます。フロー承認の詳細については、アラームの状態 を参照してください。

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        },
        "smsConfigurations": [
          {
            "recipients": [
              {
```

```
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                }
              }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
          }
        ],
        "emailConfigurations": [
          {
            "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value",
            "recipients": {
              "to": [
                {
                  "ssoIdentity": {
                    "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                    "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                  }
                }
              ]
            },
            "content": {
              "subject":
"$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value",
              "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
            }
          }
        ]
      }
    ]
  },
  "alarmCapabilities": {
    "initializationConfiguration": {
      "disabledOnInitialization": false
```

```
    },
    "acknowledgeFlow": {
      "enabled": true
    }
  }
}
```

h. がデータの送信先として引き受け AWS IoT Events ることができる IAM サービスロール (`roleArn`) を追加します AWS IoT SiteWise。このロールを担うには、`iotsitewise:BatchPutAssetPropertyValue` の許可と、`iotevents.amazonaws.com` がそのロールを担うことができる信頼関係が必要です。通知を送信するには、このロールには `lambda:InvokeFunction` および `sso-directory:DescribeUser` のアクセス許可も必要です。詳細については、[AWS IoT Events Developer Guide] (デベロッパーガイド) の[Alarm service roles] (アラームサービスロール) を参照してください。

- を、これらのアクションを実行するために が引き受け AWS IoT Events ることができるロールの ARN `roleArn`に置き換えます。

```
{
  "alarmModelName": "BoilerTemperatureHighAlarm",
  "alarmModelDescription": "Detects when the boiler temperature is high.",
  "severity": 3,
  "alarmRule": {
    "simpleRule": {
      "inputProperty":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.propertyValue.value",
      "comparisonOperator": "GREATER",
      "threshold":
 "$sitewise.assetModel.`assetModelId`.`thresholdAttributeId`.propertyValue.value"
    }
  },
  "alarmEventActions": {
    "alarmActions": [
      {
        "iotSiteWise": {
          "assetId":
 "$sitewise.assetModel.`assetModelId`.`alarmPropertyId`.assetId",
          "propertyId": "'alarmStatePropertyId'"
        }
      }
```

```
    ]
  },
  "alarmNotification": {
    "notificationActions": [
      {
        "action": {
          "lambdaAction": {
            "functionArn": "alarmNotificationFunctionArn"
          }
        },
        "smsConfigurations": [
          {
            "recipients": [
              {
                "ssoIdentity": {
                  "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                  "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                }
              }
            ],
            "senderId":
"$sitewise.assetModel.`assetModelId`.`senderIdAttributeId`.propertyValue.value",
            "additionalMessage":
"$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
          }
        ],
        "emailConfigurations": [
          {
            "from":
"$sitewise.assetModel.`assetModelId`.`fromAddressAttributeId`.propertyValue.value",
            "recipients": {
              "to": [
                {
                  "ssoIdentity": {
                    "identityStoreId":
"$sitewise.assetModel.`assetModelId`.`identityStoreIdAttributeId`.propertyValue.value"
                    "userId":
"$sitewise.assetModel.`assetModelId`.`userIdAttributeId`.propertyValue.value"
                  }
                }
              ]
            },
```

```
            "content": {
              "subject":
 "$sitewise.assetModel.`assetModelId`.`emailSubjectAttributeId`.propertyValue.value",
              "additionalMessage":
 "$sitewise.assetModel.`assetModelId`.`additionalMessageAttributeId`.propertyValue.valu
            }
          }
        ]
      }
    ]
  },
  "alarmCapabilities": {
    "initializationConfiguration": {
      "disabledOnInitialization": false
    },
    "acknowledgeFlow": {
      "enabled": false
    }
  },
  "roleArn": "arn:aws:iam::123456789012:role/MyIoTEventsAlarmRole"
}
```

i. 次のコマンドを実行して、 のペイロードから AWS IoT Events アラームモデルを作成します`alarm-model-payload.json`。

```
aws iotevents create-alarm-model --cli-input-json file://alarm-model-
payload.json
```

j. この操作では、アラームモデルの ARN、`alarmModelArn` を含む応答が返されます。この ARN をコピーして、次のステップでアセットモデルのアラーム定義に設定します。

ステップ 3: AWS IoT SiteWise と 間のデータフローを有効にする AWS IoT Events

AWS IoT SiteWise および で必要なリソースを作成したら AWS IoT Events、リソース間のデータフローを有効にしてアラームを有効にできます。このセクションでは、前のステップで作成したアラームモデルを使用するために、アセットモデルのアラーム定義を更新します。

AWS IoT SiteWise と AWS IoT Events (CLI) 間のデータフローを有効にするには

- アセットモデルにアラームの出典としてアラームモデルを設定します。以下の操作を実行します。

a. 次のコマンドを実行して、既存のアセットモデル定義を取得します。*asset-model-id* をアセットモデルの ID に置き換えます。

```
aws iotsitewise describe-asset-model --asset-model-id asset-model-id
```

このオペレーションは、アセットモデルの詳細を含むレスポンスを返します。

b. `update-asset-model-payload.json` という名前のファイルを作成し、前のコマンドのレスポンスをファイルにコピーします。

c. `update-asset-model-payload.json` ファイルから次の key-value ペアを削除します。

- `assetModelId`
- `assetModelArn`
- `assetModelCreationDate`
- `assetModelLastUpdateDate`
- `assetModelStatus`

d. 先に定義したアラーム複合モデルに、アラーム出典プロパティ (`AWS/ALARM_SOURCE`) を追加します。*alarmModelArn* をアラームモデルの ARN に置換し、アラーム出典プロパティの値を設定する。

```
{
  ...
  "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. 次のコマンドを実行し、`update-asset-model-payload.json` ファイルに保存されている定義でアセットモデルを更新します。*`asset-model-id`* をアセットモデルの ID に置き換えます。

```
aws iotsitewise update-asset-model \
  --asset-model-id asset-model-id \
  --cli-input-json file://update-asset-model-payload.json
```

これで、アセットモデルには AWS IoT Eventsで検出するアラームが定義されました。このアセットモデルに基づくすべてのアセットにおいて、対象プロパティをモニタリングするアラームです。各アセットにアラームを設定することで、各アセットのしきい値や IAM Identity Center 受信者などのプロパティをカスタマイズすることができます。詳細については、「でアセットにアラームを設定する AWS IoT SiteWise」を参照してください。

で外部アラームを定義する AWS IoT SiteWise

外部アラームは、 AWS IoT SiteWise以外で検知したアラームの状態を含みます。

外部アラームを定義する (コンソール)

AWS IoT SiteWise コンソールを使用して、既存のアセットモデルに外部アラームを定義できます。新しいアセットモデルに外部アラームを定義するには、アセットモデルを作成し、次のステップを実行します。詳細については、「でアセットモデルを作成する AWS IoT SiteWise」を参照してください。

アセットモデルでアラームを定義するには。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択します。
3. アラームを定義するアセットモデルを選択します。
4. [Alarm definitions] (アラーム定義) タブを選択します。
5. [Add alarm] (アラームの追加) を選択します。
6. [Alarm type options] (アラーム型オプション) で、[External alarm] (外部アラーム) を選択します。
7. アラームの名前を入力します。
8. (オプション) アラームの説明を入力します。
9. [Add alarm] (アラームの追加) を選択します。

外部アラームを定義する (CLI)

を使用して AWS CLI 、新規または既存のアセットモデルに外部アラームを定義できます。

アセットモデルに外部アラームを追加するには、アセットモデルにアラーム複合モデルを追加します。外部アラーム複合モデルは、EXTERNAL 型を指定し、アラーム出典プロパティを指定しない。次の複合アラームの例では、外部温度アラームを定義しています。

```
{
  ...
  "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": {}
          }
        }
      ]
    }
  ]
}
```

新規または既存のアセットモデルに複合モデルを追加する方法の詳細については、次を参照してください。

- アセットモデルを作成する (AWS CLI)
- アセットまたはコンポーネントモデルを更新する (AWS CLI)

外部アラームを定義した後、アセットモデルに基づいてアラームの状態をアセットに取り込むことができます。詳細については、「で外部アラーム状態を取り込む AWS IoT SiteWise」を参照してください。

でアセットにアラームを設定する AWS IoT SiteWise

アセットモデルに AWS IoT Events アラームを定義したら、アセットモデルに基づいて各アセットにアラームを設定できます。アラームのしきい値や通知設定を編集することができます。これらの値はそれぞれアセット上の属性であるため、属性のデフォルト値を更新することでこれらの値を設定することができます。

> **Note**
> これらの値はアラームには設定できますが AWS IoT Events 、外部アラームには設定できません。

トピック

- しきい値を設定する (コンソール)
- しきい値を設定する (AWS CLI)
- で通知設定を構成する AWS IoT SiteWise

しきい値を設定する (コンソール)

AWS IoT SiteWise コンソールを使用して、アラームのしきい値を指定する属性の値を更新できます。

アラームのしきい値を更新するには (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. アラームしきい値を更新したいアセットを選択します。

   > **Tip**
   > 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [編集] を選択します。
5. アラームがしきい値に使用している属性を検索し、新しい値を入力します。
6. [保存] を選択します。

しきい値を設定する (AWS CLI)

AWS Command Line Interface （ AWS CLI) を使用して、アラームのしきい値を指定する属性の値を更新できます。

この手順を完了するには、アセットの assetId とプロパティの propertyId を知っている必要があります。外部 ID を使用することもできます。アセットを作成したが、その assetId がわからない場合は、ListAssets API を使用して、特定のモデルのすべてのアセットを一覧表示します。DescribeAsset オペレーションを使用して、プロパティ ID を含むアセットのプロパティを表示します。

BatchPutAssetPropertyValue オペレーションを使用して、アセットに属性値を割り当てます。このオペレーションを使用すると、複数の属性を一度に設定できます。このオペレーションのペイロードにはエントリのリストが含まれ、各エントリにはアセット ID、プロパティ ID、属性値が含まれます。

属性の値を更新するには (AWS CLI)

1. batch-put-payload.json という名前のファイルを作成して、次の JSON オブジェクトをファイルにコピーします。このペイロードの例は、風力タービンの緯度と経度を設定する方法を示しています。ID、値、タイムスタンプを更新して、ユースケースのペイロードを変更します。

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

- ペイロードの各エントリには、一意の文字列として定義できる entryId が含まれています。リクエストのエントリが失敗した場合、各エラーには、対応するリクエストの entryId が含まれるため、再試行するリクエストを確認できます。
- 属性値を設定するには、各属性プロパティの propertyValues のリストに 1 つのタイムスタンプ品質値 (TQV) 構造を含めることができます。この構造には、新しい value および現在の timestamp を含める必要があります。
  - value - 設定中のプロパティの型に応じて、次のいずれかのフィールドを含む構造。
    - booleanValue
    - doubleValue
    - integerValue
    - stringValue
    - nullValue
  - timestamp – 現在の Unix エポック時間を秒単位で含む構造timeInSeconds。. は、過去 7 日以上、または今後 5 分より新しいタイムスタンプを持つデータポイント AWS IoT SiteWise を拒否します。

BatchPutAssetPropertyValue の ペイロードを準備する方法の詳細については、AWS IoT SiteWise APIs を使用してデータを取り込む を参照してください。

2. 次のコマンドを実行して、属性値を AWS IoT SiteWiseに送信します。

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

で通知設定を構成する AWS IoT SiteWise

アラーム通知設定は、 AWS IoT SiteWise コンソールまたは AWS Command Line Interface () を使用して設定できますAWS CLI。

通知設定を行う (コンソール)

AWS IoT SiteWise コンソールを使用して、アラームの通知設定を指定する属性の値を更新できます。

アラームの通知設定を更新するには (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [アセット] を選択します。
3. アラーム設定を更新するアセットを選択します。
4. [編集] を選択します。
5. 変更したい通知設定にアラームが使用している属性を探し、その新しい値を入力します。
6. [保存] を選択します。

通知設定を行う (CLI)

AWS Command Line Interface （ AWS CLI) を使用して、アラームの通知設定を指定する属性の値を更新できます。

この手順を完了するには、アセットの `assetId` とプロパティの `propertyId` を知っている必要があります。外部 ID を使用することもできます。アセットを作成したが、その `assetId` がわからない場合は、ListAssets API を使用して、特定のモデルのすべてのアセットを一覧表示します。DescribeAsset オペレーションを使用して、プロパティ ID を含むアセットのプロパティを表示します。

BatchPutAssetPropertyValue オペレーションを使用して、アセットに属性値を割り当てます。このオペレーションを使用すると、複数の属性を一度に設定できます。このオペレーションのペイロードにはエントリのリストが含まれ、各エントリにはアセット ID、プロパティ ID、属性値が含まれます。

属性の値を更新するには (AWS CLI)

1. `batch-put-payload.json` という名前のファイルを作成して、次の JSON オブジェクトをファイルにコピーします。このペイロードの例は、風力タービンの緯度と経度を設定する方法を示しています。ID、値、タイムスタンプを更新して、ユースケースのペイロードを変更します。

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

- ペイロードの各エントリには、一意の文字列として定義できる `entryId` が含まれています。リクエストのエントリが失敗した場合、各エラーには、対応するリクエストの `entryId` が含まれるため、再試行するリクエストを確認できます。
- 属性値を設定するには、各属性プロパティの `propertyValues` のリストに 1 つのタイムスタンプ品質値 (TQV) 構造を含めることができます。この構造には、新しい `value` および現在の `timestamp` を含める必要があります。
  - `value` - 設定中のプロパティの型に応じて、次のいずれかのフィールドを含む構造。
    - `booleanValue`
    - `doubleValue`
    - `integerValue`
    - `stringValue`
    - `nullValue`
  - `timestamp` – 現在の Unix エポック時間を秒単位で含む構造`timeInSeconds`。. は、過去 7 日以上、または今後 5 分より新しいタイムスタンプを持つデータポイント AWS IoT SiteWise を拒否します。

BatchPutAssetPropertyValue の ペイロードを準備する方法の詳細については、AWS IoT SiteWise APIs を使用してデータを取り込む を参照してください。

2. 次のコマンドを実行して、属性値を AWS IoT SiteWiseに送信します。

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

でアラームに応答する AWS IoT SiteWise

AWS IoT Events アラームの状態が変わったら、以下を実行してアラームに応答できます。

- アラームを確認することで、問題に対処していることを示すことができます。
- アラームをスヌーズして、一時的に無効にします。
- アラームを無効にすると、再度有効にするまで永久にアラームを無効にすることができます。
- 無効になっているアラームを有効にして、アラーム状態を検出する。
- アラームをリセットすると、アラームの状態や最新の値がクリアされます。

AWS IoT SiteWise コンソールまたは AWS IoT Events API を使用してアラームに応答できます。

> **Note**
> AWS IoT Events アラームには応答できますが、外部アラームには応答できません。

トピック

- アラームに応答する (コンソール)
- アラームに応答する (API)

アラームに応答する (コンソール)

AWS IoT SiteWise コンソールを使用して、アラームの確認、スヌーズ、無効化、または有効化を行うことができます。

トピック

- アラームを承認する (コンソール)。
- アラームをスヌーズする (コンソール)。
- アラームを無効にする (コンソール)。
- アラームを有効にする (コンソール)。
- アラームをリセットする (コンソール)。

アラームを承認する (コンソール)。

問題を処理していることを示すために、アラームを承認することができます。

> **Note**
> アラームを確認できるように、アラームのフロー承認を有効にする必要があります。このオプションは、 AWS IoT SiteWise コンソールからアラームを定義した場合、デフォルトで有効になります。

アラームを確認するには (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。

2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. アラームを確認したいアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [Alarms] (アラーム) を選択します。
5. 確認するアラームを選択し、[Actions] (アクション) を選択して応答アクションメニューを表示します。
6. [Acknowledge] (承認) を選択します。アラームの状態が [Acknowledged] (承認済み) に変化します。

アラームをスヌーズする (コンソール)。

アラームをスヌーズして、一時的にアラームを無効にすることができます。アラームをスヌーズする時間を指定します。

アラームをスヌーズするには (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. アラームをスヌーズさせたいアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [Alarms] (アラーム) を選択します。
5. スヌーズするアラームを選択し、[Actions] (アクション) を選択すると、対応アクションメニューが表示されます。
6. [Snooze] (スヌーズ) を選択します。スヌーズする時間を指定するモデルが開きます。
7. [Snooze length] (スヌーズの長さ) を選択するか、 [Custom snooze length] (スヌーズの長さをカスタマイズする) を入力します。
8. [保存] を選択します。アラームの状態が [Snoozed] (スヌーズ済み) に切り替わります。

アラームを無効にする (コンソール)。

アラームを無効にして、検出しないようにすることができます。アラームを無効にした後、アラームを検出させたい場合は、再度有効にする必要があります。

アラームを無効にするには (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. アラームを無効にするアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [Alarms] (アラーム) を選択します。
5. 無効にするアラームを選択し、[Actions] (アクション) を選択してレスポンスアクションメニューを表示します。
6. [Disable (無効化)] を選択します。アラームの状態が[Disabled] (無効) に変化します。

アラームを有効にする (コンソール)。

アラームを無効にしたり、スヌーズした後、再び検出するように有効化することができます。

アラームを有効にするには (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. アラームを有効にするアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [Alarms] (アラーム) を選択します。
5. 有効にするアラームを選択し、[Actions] (アクション) を選択してレスポンスアクションメニューを表示します。

6. [Enable (有効化)] を選択します。アラームの状態が[Normal] (正常) に変化します。

アラームをリセットする (コンソール)。

アラームをリセットして、その状態や最新の値をクリアすることができます。

アラームをリセットするには (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. アラームをリセットするアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [Alarms] (アラーム) を選択します。
5. 有効にするアラームを選択し、[Actions] (アクション) を選択してレスポンスアクションメニューを表示します。
6. [リセット] を選択します。アラームの状態が[Normal] (正常) に変化します。

アラームに応答する (API)

AWS IoT Events API を使用して、アラームの確認、スヌーズ、無効化、有効化、リセットを行うことができます。詳細は、[AWS IoT Events API Reference] (API リファレンス) の次の操作を参照してください。

- BatchAcknowledgeAlarm
- BatchSnoozeAlarm
- BatchDisableAlarm
- BatchEnableAlarm
- BatchResetAlarm

詳細については、[AWS IoT Events Developer Guide] (デベロッパーガイド) の[Responding to alarms] (アラームへの対応) を参照してください。

で外部アラーム状態を取り込む AWS IoT SiteWise

外部アラームは、外部で評価するアラームです AWS IoT SiteWise。アラームの状態を報告するデータソースがあり、それを AWS IoT SiteWiseに取り込みたい場合に外部アラームを使用することができます。

アラーム状態プロパティは、アラーム状態データ値に特定のフォーマットを要求します。各データ値は、文字列にシリアライズされた JSON オブジェクトである必要があります。そして、シリアライズされた文字列を文字列の値として取り込みます。詳細については、「アラーム状態のプロパティ」を参照してください。

Example アラーム状態データ値例 (シリアル化されていない)

```
{
  "stateName": "Active"
}
```

Example アラーム状態データ値の例 (シリアル化)

```
{\"stateName\":\"Active\"}
```

> **Note**
>
> データソースがこのフォーマットでデータをレポートできない場合、またはデータを取り込む前にこのフォーマットに変換できない場合、アラームプロパティを使用しないことを選択することができます。その代わり、例えば文字列データ型を持つ計測プロパティとしてデータを取り込むことができます。詳細については、「機器からのデータストリームを定義する (測定値) 」および「にデータを取り込む AWS IoT SiteWise」を参照してください。

で外部アラーム状態ストリームをマッピングする AWS IoT SiteWise

データストリームをアラーム状態のプロパティにマッピングするために、プロパティのエイリアスを定義することができます。これにより、データの取り込みや取得の際に、アラーム状態のプロパティを簡単に特定することができます。プロパティのエイリアスについての詳細は、のデータストリームを管理する AWS IoT SiteWise を参照してください。

トピック

- 外部アラーム状態ストリームをマッピング (コンソール)
- 外部アラーム状態ストリームをマッピング (AWS CLI)

外部アラーム状態ストリームをマッピング (コンソール)

データストリームをアラーム状態のプロパティにマッピングするために、プロパティのエイリアスを定義することができます。これにより、データの取り込みや取得の際に、アラーム状態のプロパティを簡単に特定することができます。プロパティのエイリアスについての詳細は、のデータストリームを管理する AWS IoT SiteWise を参照してください。

AWS IoT SiteWise コンソールを使用して、アラーム状態プロパティのエイリアスを設定できます。

アラーム状態プロパティにプロパティエイリアスを設定する方法 (コンソール)。

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. プロパティエイリアスを設定するアセットを選択します。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [編集] を選択します。
5. [アラーム] までスクロールし、 セクションを展開します。
6. [外部アラーム] で、[プロパティエイリアス - オプション] にエイリアスを入力します。
7. [保存] を選択します。

外部アラーム状態ストリームをマッピング (AWS CLI)

データストリームをアラーム状態のプロパティにマッピングするために、プロパティのエイリアスを定義することができます。これにより、データの取り込みや取得の際に、アラーム状態のプロパティを簡単に特定することができます。プロパティのエイリアスについての詳細は、のデータストリームを管理する AWS IoT SiteWise を参照してください。

AWS Command Line Interface （ AWS CLI) を使用して、アラーム状態プロパティのエイリアスを設定できます。

この手順を完了するには、アセットの assetId とプロパティの propertyId を知っている必要があります。外部 ID を使用することもできます。アセットを作成したが、その assetId がわからない場合は、ListAssets API を使用して、特定のモデルのすべてのアセットを一覧表示します。DescribeAsset オペレーションを使用して、プロパティ ID を含むアセットのプロパティを表示します。

> **Note**
>
> DescribeAsset レスポンスには、当該アセットに対する複合アセットモデルのリストが含まれます。各アラームは複合モデルです。propertyId を求めるには、アラームの合成モデルを見つけ、その合成モデルの中で AWS/ALARM_STATE のプロパティを見つけます。

プロパティのエイリアスの設定方法については、アセットプロパティエイリアスを更新する を参照してください。

でアラーム状態データを取り込む AWS IoT SiteWise

アラーム状態プロパティは、アラーム状態をシリアル化された JSON 文字列として受け取る。アラーム状態を外部アラームに取り込むには AWS IoT SiteWise、このシリアル化された文字列をタイムスタンプ付き文字列値として取り込みます。次の例は、アクティブなアラームの状態データの値を示しています。

```
{\"stateName\":\"Active\"}
```

アラーム状態プロパティを識別するには、次のいずれかを指定することができます。

- データデスティネーションのアラームプロパティの assetId、 propertyId です。
- データストリームのエイリアスである propertyAlias (例えば、/company/windfarm/3/turbine/7/temperature/high)。このオプションを使用するには、まず、アラームプロパティのエイリアスを設定する必要があります。アラーム状態のプロパティにプロパティエイリアスを設定する方法については、で外部アラーム状態ストリームをマッピングする AWS IoT SiteWise を参照してください。

次のBatchPutAssetPropertyValue API ペイロードの例は、外部アラームの状態をフォーマットする方法を示しています。風力タービンの1分間あたりの回転数 (RPM) が高すぎる場合に報告される外部アラームです。

Example アラーム状態データの BatchPutAssetPropertyValue ペイロードの例。

```
{
    "entries": [
      {
        "entryId": "unique entry ID",
        "propertyAlias": "/company/windfarm/3/turbine/7/temperature/high",
        "propertyValues": [
          {
            "value": {
              "stringValue": "{\"stateName\":\"Active\"}"
            },
            "timestamp": {
              "timeInSeconds": 1607550262
            }
          }
        ]
      }
    ]
  }
```

`BatchPutAssetPropertyValue` API を使ったデータインジェストの方法については、AWS IoT SiteWise APIs を使用してデータを取り込む を参照してください。

その他のデータインジェスト方法については、にデータを取り込む AWS IoT SiteWise を参照してください。

AWS IoT SiteWise アシスタント

Assistant AWS IoT SiteWise は生成 AI を活用したアシスタントです。これにより、プラントマネージャー、品質エンジニア、メンテナンス技術者などのユーザーは、運用データとエンタープライズデータから直接インサイトを得て、問題を解決し、アクションを実行できます。

AWS IoT SiteWise Assistant は、 AWS IoT データ、アセットモデル、マニュアル、ドキュメントからの情報を、重要なイベントのわかりやすい概要に統合します。また、インタラクティブな詳細な質疑応答セッションが可能になり、診断、根本原因探索、ガイド付きレコメンデーションが容易になります。

トピック

- AWS IoT SiteWise Assistant を設定する
- データセットを作成する
- データセットを編集する
- データセットを削除する
- AWS IoT SiteWise アシスタントの質問

AWS IoT SiteWise Assistant を設定する

AWS IoT SiteWise アシスタント設定

1. AWS IoT SiteWise コンソール にサインインします。

> **Note**
>
> AWS IoT TwinMaker サービスとの統合を有効にするアクセス許可を付与します。これは、 AWS IoT SiteWise Assistant とダッシュボードが AWS IoT SiteWise リソースで SQL クエリを実行するために必要です。「Integrate AWS IoT SiteWise and AWS IoT TwinMaker」を参照してください。

2. 左側のナビゲーションパネルから Assistant を選択します。

データセットを作成する

> **Note**
>
> AWS IoT SiteWise Assistant は、エンタープライズレベルの知識とガイダンスのために、Amazon Kendra インデックスを持つデータセットを使用する必要があります。Amazon Kendra インデックスがない場合は、「インデックスを作成する」を参照してください。データセットを追加すると、アシスタントのレスポンスの品質が向上し、幻覚が最小限に抑えられます。

Console

AWS IoT SiteWise コンソールでデータセットを作成する

1. データセットは、AWS IoT SiteWise アシスタントページのデータセットセクションに表示されます。
2. データセットが存在しない場合は、データセットの作成を選択します。
3. データセットの詳細ページで、ドロップダウンメニューからデータセットに関連付ける Kendra インデックスを選択します。
4. データセット名は、ステップ 3 で選択した Kendra インデックスによって入力されます。必要に応じて名前を編集します。
5. (オプション) データセットの説明は、ステップ 3 で選択した Kendra インデックスによって入力されます。必要に応じて説明を編集します。
6. アクセス許可セクションで、以下から選択します。
   a. [新しいサービスロールの作成と使用] を選択します。デフォルトでは、 はサービスロール AWS IoT SiteWise を自動的に作成します。このロールにより、 AWS IoT SiteWise アシスタントは Kendra インデックスにアクセスできます。
   b. 既存の[サービスロールの使用] を選択し、対象のロールを選択します。
7. [作成] を選択します。

ユーザーが新しいサービスロールの作成と使用を選択した場合に、 によってユーザー AWS IoT SiteWise 用に作成されたサービスロール。

AWS CLI

でデータセットを作成する AWS CLI

1. データセットの作成に使用される IAM ロールを作成します。次のアクセス許可ポリシーを使用します。

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kendra:Retrieve"
            ],
            "Resource": "arn:aws:kendra:*:*:index/*"
        }
    ]
}
```

次の信頼関係を使用します。

JSON

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iotsitewise.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

2. この例で提供されているテンプレートを使用して、create-dataset.json ファイルを作成します。このデータセットに接続する`roleArn`には`datasetId`、、`kendra knowledgeBaseArn`、 を入力します。

```
{
    "datasetId": "<UUID>",
    "datasetName": "DatasetForAssistant",
    "datasetSource": {
       "sourceType": "KENDRA",
       "sourceFormat": "KNOWLEDGE_BASE",
       "sourceDetail": {
          "kendra": {
            "knowledgeBaseArn": "arn:aws:kendra::%s:index/index",
            "roleArn": "arn:aws:iam::%s:role/role"
          }
       }
    }
}
```

3. 次のコマンドを使用してデータセットを作成します。

```
aws iotsitewise create-dataset --cli-input-json file://create-dataset.json --
region us-east-1
```

データセットを編集する

Console

データセットを編集する

1. データセットは、アシスタントページのデータセットセクションに表示されます。編集するデータセットを選択します。編集を選択して編集を開始します。
2. データセットの詳細ページで、ドロップダウンメニューから Kendra インデックスを選択してデータセットに関連付けます。
3. データセット名は、ステップ 2 で選択した Kendra インデックスによって入力されます。必要に応じて名前を編集します。
4. (オプション) データセットの説明は、ステップ 2 で選択した Kendra インデックスによって入力されます。必要に応じて説明を編集します。
5. アクセス許可セクションで、以下から選択します。
   a. [新しいサービスロールの作成と使用] を選択します。デフォルトでは、 はサービスロール AWS IoT SiteWise を自動的に作成します。このロールにより、 AWS IoT SiteWise アシスタントは Kendra インデックスにアクセスできます。
   b. 既存の[サービスロールの使用] を選択し、対象のロールを選択します。
6. 変更の保存を選択して、選択を保存します。

AWS CLI

でデータセットを編集する AWS CLI

1. この例で提供されているテンプレートを使用して、update-dataset.json ファイルを作成します。このデータセットに接続するroleArnにはdatasetId、、kendra knowledgeBaseArn、 を入力します。

```
{
    "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. 次のコマンドを使用してデータセットを更新します。

```
aws iotsitewise update-dataset --cli-input-json file://update-dataset.json --
region us-east-1
```

データセットを削除する

Console

データセットを削除する

1. データセットは、アシスタントページのデータセットセクションに表示されます。データセットを選択します。[削除] を選択します。
2. ポップアップに confirm と入力して、削除を確認します。

3. [削除] を選択します。

AWS CLI

データセットを削除する

- を使用してデータセットを削除しますdatasetId。

```
aws iotsitewise delete-dataset --region us-east-1 --dataset-id <UUID>
```

AWS IoT SiteWise アシスタントの質問

AWS IoT SiteWise Assistant のクエリの詳細についてはAWS IoT SiteWise Assistant に尋ねる質問の例、「」を参照してください。

でデータをモニタリングする AWS IoT SiteWise Monitor

AWS IoT SiteWise を使用して、SiteWise Monitor ウェブポータルを作成することで、プロセス、デバイス、および機器からのデータをモニタリングできます。SiteWise Monitor は、マネージドウェブアプリケーションの形式でポータルを作成するために AWS IoT SiteWise 使用できる の機能です。その後、これらのポータルを使用して、運用データを表示および共有できます。ダッシュボードを使用してプロジェクトを作成し、 AWS IoTに接続されているプロセス、デバイス、機器からのデータを可視化できます。

プロセスエンジニアのようなドメインエキスパートは、ポータルを使用して運用データに関するインサイトを迅速に取得し、デバイスや機器の動作を把握できます。

以下は、風力発電所のデータを表示するダッシュボードの例です。

は時間の経過とともにデータを AWS IoT SiteWise キャプチャするため、SiteWise Monitor を使用して、時間の経過とともに運用データ、または特定の時点で最後に報告された値を表示できます。これにより、通常であれば見つけることが難しいインサイトを発見できます。

SiteWise Monitor のロール

4 つのロールが SiteWise Monitor と相互作用します。

AWS 管理者

AWS 管理者は AWS IoT SiteWise コンソールを使用してポータルを作成します。 AWS 管理者は、ポータル管理者を割り当て、ポータルユーザーを追加することもできます。ポータル管理者は、後でポータルユーザーを所有者または閲覧者としてプロジェクトに割り当てます。管理者は AWS AWS コンソールでのみ動作します。

ポータル管理者

各 SiteWise Monitor ポータルには、1 人以上のポータル管理者があります。ポータル管理者は、ポータルを使用して、アセットとダッシュボードのコレクションを含むプロジェクトを作成します。その後で、ポータル管理者はアセットと所有者を各プロジェクトに割り当てます。プロジェクトへのアクセスを制御することにより、ポータル管理者は、プロジェクトの所有者とビューワーが参照できるアセットを指定します。

プロジェクトの所有者

各 SiteWise Monitor プロジェクトには所有者がいます。プロジェクトの所有者は、一貫した方法で運用データを表すために、ダッシュボードの形式で可視化を作成します。ダッシュボードを共有する準備ができたら、プロジェクト所有者はビューワーをプロジェクトに招待できます。プロジェクトの所有者は、他の所有者もプロジェクトに割り当てることができます。プロジェクト所有者は、アラームのしきい値や通知設定を設定することができます。

プロジェクトビューワー

各 SiteWise Monitor プロジェクトにはビューワーがいます。プロジェクト閲覧者は、ポータルに接続して、プロジェクト所有者が作成したダッシュボードを表示できます。各ダッシュボードでは、プロジェクトビューワーが時間範囲を調整することで、運用データをより深く理解することができます。プロジェクト閲覧者は、アクセス権のあるプロジェクトのダッシュボードのみを表示できます。プロジェクトビューワーは、アラームを確認したり、スヌーズさせたりすることができます。

組織によっては、同じユーザーが複数のロールを実行することがあります。

次の図は、4 つのロールが SiteWise Monitor ポータルでどのように相互作用するかを示しています。

[AWS IAM Identity Center or] (または) IAM を使用して、データにアクセスできるユーザーを管理できます。データユーザーは、IAM Identity Center または IAM 認証情報を使用してデスクトップまたはモバイルブラウザから SiteWise Monitor にサインインすることができます。

SAML フェデレーション

IAM Identity Center および IAM は、SAML (Security Assertion Markup Language) 2.0 によるID フェデレーションをサポートしています。SAML 2.0 は、多くの外部アイデンティティプロバイダ (IdP) がユーザーを認証し、そのアイデンティティおよびセキュリティ情報をサービスプロバイダ (SP) に渡すために使用するオープンスタンダードです。SP は通常、アプリケーションまたはサービスです。SAML フェデレーションは、SiteWise Monitor のポータル管理者とユーザーが、企業のユーザー

名やパスワードなどの外部認証情報を使用して、割り当てられたポータルにサインインできるようにします。

IAM Identity Center および IAM は、SiteWise Monitor ポータルへのアクセスに SAML ベースのフェデレーションを使用するように設定できます。

IAM アイデンティティセンター

ポータル管理者とユーザーは、会社のユーザー名とパスワードを使用して AWS アクセスポータルにサインインできます。そして、割り当てられた SiteWise Monitor ポータルに移動することができます。IAM Identity Center は証明書を使用して、ID プロバイダーと の間の SAML 信頼関係を設定します AWS。詳細については、AWS IAM Identity Center ユーザーガイドの「SCIM プロファイルと SAML 2.0 の実装」を参照してください。

IAM

ポータル管理者とユーザーは、割り当てられた SiteWise Monitor ポータルにアクセスするための一時的なセキュリティ認証情報を要求することができます。IAM で SAML ID プロバイダー ID を作成して、ID プロバイダーと の間の信頼関係を設定します AWS。詳細については、「IAM ユーザーガイド」の「API アクセスに SAML ベースのフェデレーションを使用する AWS」を参照してください。

ポータル管理者とユーザーは、会社のポータルにサインインし、 AWS マネジメントコンソールに移動するオプションを選択できます。そして、割り当てられた SiteWise Monitor ポータルに移動することができます。会社のポータルは、ID プロバイダーと の間の信頼の交換を処理します AWS。詳細については、IAM ユーザーガイドの「SAML 2.0 フェデレーティッドユーザーが AWS マネジメントコンソールにアクセスできるようにする」を参照してください。

> **Note**
>
> ポータルへのユーザーまたは管理者の追加では、IP の制限などのユーザー許可を制限する IAM ポリシーを作成しないようにしてください。アクセス許可が制限されたアタッチされたポリシーは、 AWS IoT SiteWise ポータルに接続できません。

SiteWise Monitor の概念

SiteWise Monitor を使用するには、次の概念に精通している必要があります。

Portal

AWS IoT SiteWise Monitor ポータルは、 AWS IoT SiteWise データを視覚化して共有するために使用できるウェブアプリケーションです。ポータルには 1 人以上の管理者があり、0 個以上のプロジェクトが含まれています。

プロジェクト

各 SiteWise Monitor ポータルには、一連のプロジェクトが含まれています。各プロジェクトには、 AWS IoT SiteWise アセットと関連付けられたのサブセットがあります。プロジェクト所有者は、1 つ以上のダッシュボードを作成して、それらのアセットに関連付けられたデータの整合性ある表示方法を提供します。プロジェクトの所有者は、プロジェクトに閲覧者を招待して、プロジェクト内のアセットとダッシュボードを表示できるようにすることができます。プロジェクトは、SiteWise Monitor 内の共有の基本単位です。プロジェクト所有者は、 AWS 管理者からポータルへのアクセスを許可されたユーザーを招待できます。ユーザーは、そのポータル内のプロジェクトをそのユーザーと共有する前に、ポータルへのアクセス権を持っている必要があります。

アセット

産業機器 AWS IoT SiteWise からデータが に取り込まれると、デバイス、機器、プロセスはそれぞれアセットとして表されます。各アセットには、関連付けられたプロパティとアラームがあります。ポータル管理者は、各プロジェクトにアセットのセットを割り当てます。

プロパティ

プロパティは、アセットに関連する時系列データです。たとえば、機器には、シリアル番号、ロケーション、製造元とモデル、およびインストール日があります。また、可用性、性能、品質、温度、圧力などの時系列値を持つ場合もあります。

アラーム

アラームは、機器の動作範囲外を識別するために、プロパティをモニタリングします。各アラームは、しきい値とモニタリングするプロパティを定義します。プロパティがしきい値を超えると、アラームがアクティブになり、お客様やお客様のチームの誰かが問題に対処する必要があることを示します。プロジェクト所有者は、アラームのしきい値や通知設定をカスタマイズすることができます。プロジェクトビューワーはアラームを確認したり、スヌーズしたりすることができ、アラームの詳細や対処方法をメッセージとして残すことができます。

ダッシュボード

各プロジェクトには、ダッシュボードのセットが含まれています。ダッシュボードは、一連のアセットの値に対する一連の可視化を提供します。プロジェクトの所有者は、ダッシュボードと

それに含まれる可視化を作成します。プロジェクト所有者がダッシュボードのセットを共有する準備ができたら、所有者はプロジェクトに閲覧者を招待し、プロジェクト内のすべてのダッシュボードにアクセスできるようにします。ダッシュボードごとに異なる閲覧者のセットが必要な場合は、プロジェクト間でダッシュボードを分割する必要があります。ビューワーがダッシュボードを見るとき、特定のデータが表示されるように時間範囲をカスタマイズできます。

視覚化

各ダッシュボードでは、プロジェクトに関連するアセットのプロパティとアラームをどのように表示するかをプロジェクト所有者が決定します。可用性は折れ線グラフで、その他の数値は棒グラフや重要業績評価指標 (KPI、 Key Performance Indicator) で表示されるかもしれません。アラームは、状態グリッドと状態タイムラインでの表示が最適です。プロジェクトの所有者は、それぞれの可視化をカスタマイズして、そのアセットのデータを最もよく理解できるようにしています。

(AWS IoT SiteWise Monitor Classic) の使用を開始する

お客様が組織の AWS 管理者である場合は、 AWS IoT SiteWise コンソールからポータルを作成します。組織のメンバーが AWS IoT SiteWise データを表示できるようにポータルを作成するには、次の手順を実行します。

1. ポータルを設定して作成する
2. ポータル管理者の追加と招待メールの送信
3. ポータルユーザーの追加

ポータルを作成すると、ポータル管理者は AWS IoT SiteWise アセットを表示し、ポータルのプロジェクトに割り当てることができます。プロジェクト所有者は、ダッシュボードを作成してアセットのプロパティを可視化できます。これは、プロジェクトビューワーがデバイス、プロセス、および機器のパフォーマンスを理解するのに役立ちます。

> **Note**
>
> ポータルにユーザーまたは管理者を追加するときは、制限された IP など、ユーザーのアクセス許可を制限する AWS Identity and Access Management (IAM) ポリシーを作成しないでください。アクセス許可が制限されたアタッチされたポリシーは、 AWS IoT SiteWise ポータルに接続できません。

チュートリアルでは、風力発電施設データを使用して、特定のシナリオのプロジェクト、ダッシュボード、複数のユーザーを含むポータルをセットアップするために必要な手順を説明します。詳細については、「SiteWise Monitor で風力発電所データを視覚化して共有する」を参照してください。

トピック

- SiteWise Monitor でポータルを作成する
- SiteWise Monitor でポータルを設定する
- SiteWise Monitor で管理者を招待する
- SiteWise Monitor にポータルユーザーを追加する
- AWS IoT SiteWise ダッシュボードの作成 (AWS CLI)
- でポータルのアラームを有効にする AWS IoT SiteWise
- エッジで AWS IoT SiteWise ポータルを有効にする
- SiteWise Monitor ポータルを管理する

SiteWise Monitor でポータルを作成する

AWS IoT SiteWise コンソールで SiteWise Monitor ポータルを作成します。

ポータルを作成するには

1. AWS IoT SiteWise コンソールにサインインします。
2. ナビゲーションペインで、[モニタリング]、[開始方法] の順に選択します。

3. [ポータルの作成] を選択します。

次に、ポータルを設定するために基本情報を指定する必要があります。

SiteWise Monitor でポータルを設定する

ユーザーはポータルを使用してデータを表示します。ポータルの名前、説明、ブランディング、ユーザー認証、サポートお問い合わせメール、アクセス許可などをカスタマイズできます。

AWS IoT SiteWise > Monitor > Portals > Create portal

Step 1
Portal configuration

Step 2- *optional*
Additional features

Step 3
Invite administrators

Step 4
Assign users

Portal configuration

Each web portal provides enterprise users with access to your IoT SiteWise assets. Learn more

Portal details

Portal name
Choose a portal name to identify the web portal to your users. Company name is recommended.

example-factory-1

Name should be 1-128 characters and only contain A-Z a-z 0-9 _ and -.

Description - *optional*
Create a description of your portal

Example Corp Factory #1 in Renton, WA

Description should contain a maximum of 2048 characters.

Portal branding

You can provide your logo image to display your brand in this web portal.

Logo image
Upload a square, high-resolution .png file. The image is displayed on a dark background.

Choose file

The file size must be less than 1 MB.

User authentication

Your users can sign in to this portal with their AWS Single Sign-On (AWS SSO) or AWS Identity and Access Management (IAM) credentials. If you choose AWS SSO, you must enable the service for your AWS account.

You haven't enabled AWS SSO in your account yet. When you create your first portal user, this automatically enables AWS SSO in your AWS account.

Create user

- (●) AWS SSO
  Your users can sign in to the portal with their corporate usernames and passwords.
- ( ) IAM
  Your users can sign in to the portal with their IAM credentials.

Support contact email

You can provide an email address for cases where there's a problem or issue with this portal and your users need to contact support to resolve.

Email

support@example.com

Tags

This resource doesn't have any tags.

Add tag

You can add up to 50 more tags.

Permissions

SiteWise Monitor assumes this role to give permissions to your federated users to access AWS IoT SiteWise resources. Learn

ポータルを設定するには

1. ポータルの名前を入力します。
2. (オプション) ポータルの説明を入力します。複数のポータルがある場合は、わかりやすい説明を使用すると、各ポータルに含まれる内容を追跡できます。
3. (オプション) 画像をアップロードして、ブランドをポータルに表示します。正方形の PNG 画像を選択します。正方形でない画像をアップロードすると、画像は正方形に縮小されます。
4. 以下のオプションのいずれかを選択してください：
   - ポータルのユーザーが会社のユーザー名とパスワードでこのポータルにサインインする場合は、[IAM Identity Center] を選択します。

     アカウントで IAM Identity Center を有効にしていない場合は、次のことを行います。

     a. [ユーザーの作成] を選択します。

     b. [ユーザーの作成] ページで、最初のポータルを作成するために、ユーザーの E メールアドレス、名前、姓を入力し、[ユーザーの作成] を選択します。

> **Note**
>
> - AWS 最初のポータルユーザーを作成すると、 アカウントで IAM Identity Center が自動的に有効になります。
> - IAM Identity Center は、一度に 1 つのリージョンでしか設定できません。SiteWise Monitor は、IAM Identity Center に対して設定されたリージョンに接続します。つまり、IAM Identity Center へのアクセスには 1 つのリー

ジョンを使用しますが、どのリージョンにもポータルを作成することができます。

- ポータルのユーザーが IAM の認証情報を使ってこのポータルにサインインする場合は、[IAM] を選択します。

> **Important**
> ユーザーまたはロールがこのポータルにサインインするには `iotsitewise:DescribePortal` アクセス許可が必要です。

5. ポータルに問題があり、解決するためにサポートが必要な場合にポータルユーザーが連絡できる E メールアドレスを入力します。
6. (オプション) ポータルにタグを追加します。詳細については、「AWS IoT SiteWise リソースにタグを付ける」を参照してください。
7. 以下のオプションのいずれかを選択してください：
   - [新しいサービスロールの作成と使用] を選択します。デフォルトでは、SiteWise Monitor は自動的に各ポータルのサービスロールを作成します。このロールにより、ポータルユーザーは AWS IoT SiteWise リソースにアクセスできます。詳細については、「のサービスロールを使用する AWS IoT SiteWise Monitor」を参照してください。
   - 既存の[サービスロールの使用] を選択し、対象のロールを選択します。
8. [次へ] を選択します。
9. (オプション) ポータルのアラームを有効にします。詳細については、「でポータルのアラームを有効にする AWS IoT SiteWise」を参照してください。
10. Create. AWS IoT SiteWise will create your portal を選択します。

> **Note**
> コンソールを閉じる場合、管理者とユーザーを追加することでセットアッププロセスを完了できます。詳細については、「でのポータル管理者の追加または削除 AWS IoT SiteWise」を参照してください。このポータルを維持しない場合は、リソースを使用しないようにするため、ポータルを削除します。詳細については、「でポータルを削除する AWS IoT SiteWise」を参照してください。

[状態] 欄には、次のいずれかの値を指定することができる。

- CREATING - ポータルの作成リクエスト AWS IoT SiteWise を処理しています。このプロセスは完了までに数分かかることがあります。
- UPDATING - ポータルの更新リクエスト AWS IoT SiteWise を処理しています。このプロセスは完了までに数分かかることがあります。
- PENDING - AWS IoT SiteWise DNS レコードの伝播が終了するのを待っています。このプロセスは完了までに数分かかることがあります。ステータスが「保留中」の場合は、ポータルを削除できます。
- DELETING - ポータルの削除リクエスト AWS IoT SiteWise を処理しています。このプロセスは完了までに数分かかることがあります。
- アクティブ - ポータルがアクティブになると、ポータルユーザーはポータルにアクセスできます。
- FAILED - ポータルの作成、更新、または削除のリクエストを処理 AWS IoT SiteWise できませんでした。が Amazon CloudWatch Logs AWS IoT SiteWise にログを送信できるようにした場合、これらのログを使用して問題のトラブルシューティングを行うことができます。詳細については、CloudWatch Logs AWS IoT SiteWise によるモニタリング」を参照してください。

ポータルを作成すると、メッセージが表示されます。

次に、1 人以上のポータル管理者をポータルに招待する必要があります。これまでのところ、ポータルを作成しましたが、誰もアクセスできません。

SiteWise Monitor で管理者を招待する

新しいポータルの使用を開始するには、ポータル管理者を割り当てる必要があります。ポータル管理者は、プロジェクトを作成し、プロジェクト所有者を選択して、アセットをプロジェクトに割り当てます。ポータル管理者は、すべての AWS IoT SiteWise アセットを表示できます。

ユーザー認証サービスに基づいて、次のいずれかを選択します。

IAM Identity Center

SiteWise Monitor を初めて使用する場合、先に作成したユーザーをポータル管理者として選択することができます。ポータル管理者として別のユーザーを追加する場合は、このページから IAM Identity Center ユーザーを作成することができます。もう一つの方法として、外部 ID プロバイ

ダを IAM Identity Center に接続することもできます。詳細については、AWS IAM Identity Center ユーザーガイドをご参照ください。

管理者を招待するには

1. ポータル管理者になってもらいたいユーザーのチェックボックスをオンにします。これにより、ユーザーが [ポータル管理者] リストに追加されます。

> **Note**
>
> IAM Identity Center をアイデンティティストアとして使用しており、 AWS Organizations 管理アカウントにサインインしている場合は、[ユーザーの作成] を選択することで IAM Identity Center ユーザーを作成することができます。IAM Identity Center から新しいユーザーに、パスワードを設定するための E メールが送信されます。そして、そのユーザーを管理者としてポータルに割り当てることができます。詳細については、「IAM Identity Center での ID の管理」を参照してください。

2. (オプション) [選択したユーザーに招待を送信] を選択します。E メールクライアントが開き、招待状がメッセージ本文に入力されます。

   E メールは、ポータル管理者に送信する前にカスタマイズできます。後でポータル管理者に E メールを送信することもできます。SiteWise Monitor を初めてを使用していて、新しい IAM Identity Center または IAM ユーザーやロールをポータル管理者として追加する場合、自分自身にメールを送信する必要はありません。

3. 管理者として不要なユーザーを追加する場合は、そのユーザーのチェックボックスをオフにします。

4. ポータル管理者の招待が終了したら、[Next (次へ)] を選択します。

IAM

ポータル管理者にするユーザーまたはロールを選択することができます。ポータル管理者として別のユーザーまたはロールを追加する場合は、IAM コンソールでユーザーまたはロールを作成することができます。詳細については、IAM ユーザーガイドの「AWS アカウントでの IAM ユーザーの作成」および「IAM ロールの作成」を参照してください。

管理者を招待するには

1. 以下の操作を実行します。

- [IAM ユーザー] を選択して、IAM ユーザーをポータル管理者として追加します。
- [IAM ロール] を選択して、ポータル管理者として IAM ロールを追加します。

2. ポータルサイトの管理者にしたいユーザーやロールのチェックボックスを選択します。これにより、ユーザーまたはロールが [Portal administrators] (ポータル管理者) リストに追加されます。
3. 管理者として不要なユーザーまたはロールを追加した場合は、そのユーザーまたはロールのチェックボックスを外します。
4. ポータル管理者の招待が終了したら、[次へ] を選択します。

> **Important**
>
> ユーザーまたはロールがこのポータルにサインインするには `iotsitewise:DescribePortal` アクセス許可が必要です。

> **Note**
>
> IAM Identity Center をアイデンティティストアとして使用しており、 AWS Organizations 管理アカウントにサインインしている場合は、[ユーザーの作成] を選択することで IAM Identity Center ユーザーを作成することができます。IAM Identity Center から新しいユーザーに、パスワードを設定するための E メールが送信されます。そして、そのユーザーを管理者としてポータルに割り当てることができます。詳細については、「IAM Identity Center での ID の管理」を参照してください。

ポータル管理者のリストは後で変更できます。詳細については、「でのポータル管理者の追加または削除 AWS IoT SiteWise」を参照してください。

> **Note**
>
> ポータル管理者のみがプロジェクトを作成し、アセットを割り当てることができるため、少なくとも1 人のポータル管理者を指定する必要があります。

最後の手順として、新しいポータルにアクセスできるユーザーを追加します。

SiteWise Monitor にポータルユーザーを追加する

ポータルにアクセスできるユーザーを制御します。各ポータルでは、ポータル管理者は 1 つ以上のプロジェクトを作成し、ポータルユーザーを各プロジェクトの所有者または閲覧者として割り当てます。各プロジェクト所有者は、プロジェクトを所有または閲覧するように追加のポータルユーザーを招待することができます。

ユーザー認証サービスに基づいて、次のいずれかを選択します。

IAM Identity Center

ユーザーを [ユーザー] リストに追加する場合は、次のステップを実行します。

ポータルユーザーを追加するには

1. ポータルに追加するユーザーを [ユーザー] リストから選択します。これにより、ユーザーが [ポータルユーザー] リストに追加されます。SiteWise Monitor を初めて使用する場合、ポータル管理者をポータルユーザーとして追加する必要はありません。

   > **Note**
   >
   > IAM Identity Center をアイデンティティストアとして使用しており、 AWS Organizations 管理アカウントにサインインしている場合は、[ユーザーの作成] を選択することで IAM Identity Center ユーザーを作成することができます。IAM Identity Center から新しいユーザーに、パスワードを設定するための E メールが送信されます。そして、そのユーザーをユーザーとしてポータルに割り当てることができます。詳細については、「IAM Identity Center での ID の管理」を参照してください。

2. ポータルにアクセスさせたくないユーザーを追加する場合は、そのユーザーのチェックボックスをオフにします。
3. ユーザーの選択が終了したら、[ユーザーの割り当て] を選択します。

IAM

[IAM ユーザー] または [IAM ロール] のリストに追加したいユーザーまたはロールが表示されたら、次のステップを実行します。

ポータルユーザーを追加するには

1. 次のオプションを実行します。

   - [IAM ユーザー] () を選択して、IAM ユーザーをポータルユーザーとして追加します。
   - [IAM ロール] を選択して、IAM ロールをポータルユーザーとして追加します。

   SiteWise Monitor を初めて使用する場合、ポータル管理者をポータルユーザーとして追加する必要はありません。

2. ポータルユーザーにするユーザーまたはロールのチェックボックスを選択します。これにより、ユーザーまたはロールが [Portal users] (ポータルユーザー) リストに追加されます。
3. ポータルにアクセスさせたくないユーザーを追加する場合は、そのユーザーのチェックボックスをオフにします。
4. ユーザーの選択が終了したら、[ユーザーの割り当て] を選択します。

> **Important**
>
> ユーザーまたはロールがこのポータルにサインインするには `iotsitewise:DescribePortal` アクセス許可が必要です。

お疲れ様でした。ポータルの作成、ポータル管理者の割り当て、ポータルを利用するユーザーの割り当てが完了しました。ポータル管理者は、プロジェクトを作成し、それらのプロジェクトにアセットを追加できるようになりました。次に、プロジェクトの所有者はダッシュボードを作成して、各プロジェクトのアセットのデータを可視化することができます。

ポータルユーザーのリストは、後で変更することができます。詳細については、「でポータルユーザーを追加または削除する AWS IoT SiteWise」を参照してください。

ポータルに変更を加える必要がある場合は、「SiteWise Monitor ポータルを管理する」を参照してください。

ポータルを始めるには、SiteWise Monitor アプリケーションガイドの「開始方法」を参照してください。

AWS IoT SiteWise ダッシュボードの作成 (AWS CLI)

を使用してダッシュボードで視覚化 (またはウィジェット) を定義する場合は AWS CLI、JSON `dashboardDefinition` ドキュメントで次の情報を指定する必要があります。この定義は、CreateDashboard および UpdateDashboard オペレーションのパラメータです。

`widgets`

ウィジェット定義構造のリスト。それぞれに次の情報が含まれています。

`type`

widget. のタイプ AWS IoT SiteWise には、次のウィジェットタイプが用意されています。

- `sc-line-chart` - 折れ線グラフ 詳しくは、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[Line charts] (折れ線グラフ) をご覧ください。
- `sc-scatter-chart` — 散布図。詳細については、AWS IoT SiteWise Monitor アプリケーションガイドの「散布図」を参照してください。
- `sc-bar-chart` - 棒グラフ 詳細については、AWS IoT SiteWise Monitor アプリケーションガイドの「棒グラフ」を参照してください。
- `sc-status-grid` - アセットプロパティの最新値をグリッドで表示する状態ウィジェットです。詳しくは、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[Status widgets] (状態ウィジェット) をご覧ください。
- `sc-status-timeline` - アセットプロパティの履歴値を時系列で表示する状態ウィジェットです。詳しくは、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[Status widgets] (状態ウィジェット) をご覧ください。
- `sc-kpi` - 重要業績評価指標 (KPI、 key performance indicator) の視覚化。詳しくは、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[KPI widgets] (KPI ウィジェット) をご覧ください。
- `sc-table` — テーブルウィジェット 詳しくは、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[Table widgets] (テーブルウィジェット) をご覧ください。

`title`

ウィジェットのタイトル。

`x`

ウィジェットの水平位置。グリッドの左から開始します。この値は、ダッシュボードのグリッド内のウィジェットの位置を参照します。

`y`

グリッドの上部から始まる、ウィジェットの垂直位置。この値は、ダッシュボードのグリッド内のウィジェットの位置を参照します。

`width`

ダッシュボードのグリッド上のスペース数で表されるウィジェットの幅。

`height`

ダッシュボードのグリッド上のスペースの数で表されるウィジェットの高さ。

`metrics`

それぞれがこのウィジェットのデータストリームを定義するメトリクス構造のリスト。リスト内の各構造には、次の情報が含まれている必要があります。

`label`

このメトリクスに表示するラベル。

`type`

この metric. のデータソースのタイプは、次のメトリクスタイプ AWS IoT SiteWise を提供します。

- `iotsitewise` – ダッシュボードには、 AWS IoT SiteWiseにあるアセットプロパティのデータがフェッチされます。このオプションを選択した場合は、このメトリクスに対して `assetId` および `propertyId` を定義する必要があります。

`assetId`

(オプション) AWS IoT SiteWiseのアセットの ID。

このフィールドは、このメトリクスで `type` に `iotsitewise` を選択した場合、必須です。

`propertyId`

(オプション) AWS IoT SiteWiseのアセットプロパティの ID。

このフィールドは、このメトリクスで `type` に `iotsitewise` を選択した場合、必須です。

`analysis`

(オプション) ウィジェットに表示するトレンドラインなどの分析を定義する構造です。詳しくは、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[Configuring trend lines] (トレンドラインの設定) をご覧ください。ウィジェットのプロパティごとに、各型のトレンドラインを 1 つずつ追加することができます。解析構造には、次の情報が含まれています。

`trends`

(オプション) 各ウィジェットのトレンド分析を定義するトレンド構造のリスト。リスト内の各構造には、次の情報が含まれています。

`type`

トレンドラインの種類。次のオプションを選択します。

- `linear-regression` - 線形回帰の表示。SiteWise Monitorは、最小二乗法で線形回帰を計算します。

`annotations`

(オプション) ウィジェットのしきい値を定義するアノテーション構造。詳しくは、[AWS IoT SiteWise Monitor Application Guide] (アプリケーションガイド) の[Configuring thresholds] (しきい値の設定) をご覧ください。1 つのウィジェットにつき最大 6 つのアノテーションを追加することができます。アノテーション構造には、次の情報が含まれます。

`y`

(オプション) このウィジェットの水平方向のしきい値をそれぞれ定義するアノテーション構造のリスト。リスト内の各構造には、次の情報が含まれています。

`comparisonOperator`

しきい値の比較演算子。次のいずれかを選択します。

- `LT` - `value` より小さいデータポイントを少なくとも 1 つ持つプロパティがハイライトされます。

- GT - value より大きいデータポイントを少なくとも 1 つ持つプロパティがハイライトされます。
- LTE - value 以下のデータポイントを少なくとも 1 つ持つプロパティがハイライトされます。
- GTE - value 以上のデータポイントを少なくとも 1 つ持つプロパティがハイライトされます。
- EQ - value と同じデータポイントを少なくとも 1 つ持つプロパティがハイライトされます。

value

データポイントを comparisonOperator と比較するためのしきい値。

color

(オプション) しきい値の色を表す 6 桁の 16 進数コード。視覚化では、しきい値のルールを満たすデータポイントが少なくとも 1 つあるプロパティについて、この色でプロパティの凡例が表示されます。デフォルトは黒 (#000000) です。

showValue

(オプション) ウィジェットの余白にしきい値の値を表示するかどうかを設定します。デフォルトは true です。

properties

(オプション) ウィジェットのプロパティのフラットディクショナリです。この構造のメンバーはコンテキスト依存です。 AWS IoT SiteWise には、 を使用する以下のウィジェットが用意されていますproperties。

- 折れ線グラフ、散布図、棒グラフには次のプロパティがあります。

  colorDataAcrossThresholds

  (オプション) このウィジェットでしきい値を超えたデータの色を変更するかどうかを設定します。このオプションを有効にすると、しきい値を超えたデータが選択した色で表示されます。デフォルトは true です。

- [Status grids] (状態グリッド) は、次の特性を持ちます。

  labels

  (オプション) 状態グリッドに表示するラベルを定義する構造。ラベル構造には、次の情報が含まれる。

`showValue`

(オプション) このウィジェットの各アセットプロパティの単位と値を表示するかどうかを指定します。デフォルトは `true` です。

Example ダッシュボード定義の例

次の例では、JSON ファイルに保存されているペイロードからダッシュボードを定義します。

```
aws iotsitewise create-dashboard \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeEXAMPLE \
  --dashboard-name "Wind Farm Dashboard" \
  --dashboard-definition file://dashboard-definition.json
```

次の `dashboard-definition.json` の JSON 例では、次の視覚化ウィジェットを使用してダッシュボードを定義します。

- ダッシュボードの左上にある風力発電所の総電力を視覚化する折れ線グラフ。この折れ線グラフには、風力発電所の出力が期待される最小出力よりも小さくなるタイミングを示すしきい値が含まれています。この折れ線グラフには、線形回帰のトレンドラインも含まれています。
- ダッシュボードの右上にある 4 つのタービンの風速を表示する棒グラフ。

> **Note**
>
> 次の例は、ダッシュボードの折れ線グラフと棒グラフの視覚化を表します。このダッシュボードは、風力発電所のダッシュボードの例に似ています。

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

でポータルのアラームを有効にする AWS IoT SiteWise

AWS IoT Events ポータルの でサポートされているアラーム機能を有効にして、ポータル管理者が SiteWise Monitor ポータルで AWS IoT Events アラームモデルを作成、編集、削除できるようにします。プロジェクト所有者は、アラームを設定することができます。プロジェクトビューワーは、アラームの詳細を確認することができます。このセクションでは、 AWS IoT SiteWise コンソールを使用してポータルのアラーム機能を有効にする方法について説明します。

> **Important**
>
> - ポータルに外部アラームを作成することはできません。
> - アラーム通知を送信する場合は、ユーザー認証サービスに IAM Identity Center を選択する必要があります。
> - アラーム通知機能は、中国 (北京) では利用できません AWS リージョン。

ポータルを設定し作成すると、[ステップ 2 の追加機能] でアラームとアラーム通知を有効にすることができます。ユーザー認証サービスに基づいて、次のいずれかを選択します。

IAM Identity Center

ポータルのアラームを有効にするには

1. (オプション) [アラームを有効にする] を選択します。
   - [AWS IoT SiteWise アクセスロール] では、既存のロールを使用するか、必要なアクセス許可を持つロールを作成します。このロールには、`iotevents:BatchPutMessage` のアクセス許可と、`iot.amazonaws.com` と `iotevents.amazonaws.com` がロールを担うことができる信頼関係が必要です。
2. (オプション) [アラーム通知を有効にする] を選択します。
   a. [Sender] (送信者) は、送信者を選択します。

> **Important**
>
> Amazon SES で送信者メールアドレスの確認が必要です。詳細については、[Amazon Simple Email Service Developer Guide] (Amazon Simple Email Service デベロッパーガイド) の[Verifying email addresses in Amazon SES] (Amazon SES でのメールアドレスの確認) を参照してください。

b. [AWS Lambda ロール] の場合、既存のロールを使用するか、必要なアクセス許可を持つロールを作成します。このロールには、`lambda:InvokeFunction`、`sso-directory:DescribeUser` のアクセス許可と、`iotevents.amazonaws.com` と `lambda.amazonaws.com` がロールを担うことができる信頼関係が必要です。

c. [AWS Lambda 関数] では、既存の Lambda 関数を選択するか、アラーム通知を管理する関数を作成します。詳細については、AWS IoT Events デベロッパーガイドの「アラーム通知の管理」を参照してください。

IAM

ポータルのアラームを有効にするには

- (オプション) [アラームを有効にする] を選択します。

- [AWS IoT SiteWise アクセスロール] では、既存のロールを使用するか、必要なアクセス許可を持つロールを作成します。このロールには、`iotevents:BatchPutMessage` の許可と、`iot.amazonaws.com` と `iotevents.amazonaws.com` がロールを担うことができる信頼関係が必要です。

SiteWise Monitor のアラームの詳細については、AWS IoT SiteWise アプリケーションガイドの「アラームによるモニタリング」を参照してください。

エッジで AWS IoT SiteWise ポータルを有効にする

エッジでポータルを有効にすると、アカウントでデータ処理パックが有効になっているすべての SiteWise Edge ゲートウェイでこのポータルが利用できるようになります。

エッジでポータルを有効にするには

1. [エッジ設定] セクションで、[エッジでこのポータルを有効にする] をオンにします。
2. [Create] (作成) を選択します。

SiteWise Monitor ポータルを管理する

ポータルのさまざまな側面を管理および設定することができます。これには、ユーザーまたは管理者の追加と削除、ユーザーのアクセス許可とロールの設定、ポータルの URL と名前のカスタマイズ、サポート連絡先情報の設定、およびポータル管理者への招待メールの送信が含まれます。

1. AWS IoT SiteWise コンソールにサインインします。
2. ナビゲーションペインで、[モニター]、[ポータル] の順に選択します。

3. ポータルを選択し、[詳細の表示] を選択します (または [名前] を選択します)。
4. 次のいずれかの管理タスクを実行できます。

   - でポータルの詳細を変更する AWS IoT SiteWise
   - でのポータル管理者の追加または削除 AWS IoT SiteWise
   - ポータル管理者に招待メールを送信する
   - でポータルユーザーを追加または削除する AWS IoT SiteWise
   - でポータルを削除する AWS IoT SiteWise

ポータルを作成する方法については、「(AWS IoT SiteWise Monitor Classic) の使用を開始する」を参照してください。

トピック

- でポータルの詳細を変更する AWS IoT SiteWise
- でのポータル管理者の追加または削除 AWS IoT SiteWise
- ポータル管理者に招待メールを送信する
- でポータルユーザーを追加または削除する AWS IoT SiteWise
- でポータルを削除する AWS IoT SiteWise

でポータルの詳細を変更する AWS IoT SiteWise

ポータルの名前、説明、サポート E メール、ブランド、アクセス許可を変更できます。

1. ポータルの詳細ページの [ポータルの詳細] セクションで、[編集] を選択します。

2. [名前]、[説明]、[ポータルのブランド]、[サポートお問い合わせ E メール]、または [アクセス許可] を更新します。
3. 完了したら、[保存] を選択します。

でのポータル管理者の追加または削除 AWS IoT SiteWise

いくつかのステップで、ポータルの管理者としてユーザーを追加または削除できます。ユーザー認証サービスに基づいて、次のいずれかを選択します。

IAM Identity Center

ポータル管理者を追加するには

1. ポータルの詳細ページで、[Portal administrators] (ポータル管理者) セクションの [Assign administrators] (管理者の割り当て) を選択します。
2. [管理者の割り当て] () ページで、ポータルに管理者として追加するユーザーのチェックボックスを選択します。

> **Note**
>
> IAM Identity Center をアイデンティティストアとして使用しており、 AWS Organizations 管理アカウントにサインインしている場合は、[ユーザーの作成] を選

択することで IAM Identity Center ユーザーを作成することができます。IAM Identity Center から新しいユーザーに、パスワードを設定するための E メールが送信されます。そして、そのユーザーを管理者としてポータルに割り当てることができます。詳細については、「IAM Identity Center での ID の管理」を参照してください。

3. [Assign administrators] (管理者の割り当て) を選択します。

ポータル管理者を削除するには

- ポータルの詳細ページの [Portal administrators (ポータル管理者)] セクションで、削除する各ユーザーのチェックボックスをオンにし、[Remove from portal (ポータルから削除)] を選択します。

> **Note**
>
> ポータル管理者を最低 1 人選択することをお勧めします。

IAM

ポータル管理者を追加するには

1. ポータルの詳細ページで、[ポータル管理者] () セクションの [管理者の割り当て] を選択します。

2. [管理者の割り当て] ページで、次の操作を行います。
   - IAM ユーザーをポータル管理者として追加する場合は、[IAM ユーザー] を選択します。
   - ポータル管理者として IAM ロールを追加する場合は、[IAM ロール] () を選択します。
3. ポータルサイトの管理者にしたいユーザーやロールのチェックボックスを選択します。これにより、ユーザーまたはロールが [ポータル管理者] リストに追加されます。
4. [管理者の割り当て] を選択します。

> **Important**
>
> ユーザーまたはロールがこのポータルにサインインするには `iotsitewise:DescribePortal` アクセス許可が必要です。

ポータル管理者を削除するには

- ポータルの詳細ページの [ポータル管理者] セクションで、削除する各ユーザーのチェックボックスをオンにし、[ポータルから削除] を選択します。

> **Note**
>
> ポータル管理者なしでポータルを離れることはお勧めしません。

ポータル管理者に招待メールを送信する

ポータルサイトの管理者に招待メールを送信することができます。

1. ポータルの詳細ページの [ポータル管理者] セクションで、ポータル管理者のチェックボックスをオンにします。

2. [招待状を送信] を選択します。E メールクライアントが開き、招待状がメッセージ本文に入力されます。

   E メールは、ポータル管理者に送信する前にカスタマイズできます。

でポータルユーザーを追加または削除する AWS IoT SiteWise

ポータルにアクセスできるユーザーを選択します。ポータルユーザーは、SiteWise Monitor ポータル内のユーザーのリストに表示されます。このリストから、ポータル管理者はプロジェクト所有者を追加でき、プロジェクト所有者はプロジェクトビューワーを追加できます。

> **Note**
>
> ポータルの管理者およびポータルユーザーが、ユーザーの追加または削除を必要とする場合、ポータルのサポートメールを通じて連絡してくる場合があります。

ユーザー認証サービスに基づいて、次のいずれかを選択します。

IAM Identity Center

ポータルユーザーを追加するには

1. ポータルの詳細ページの [Portal users (ポータルユーザー)] セクションで [ユーザーの割り当て] を選択します。
2. [ユーザーの割り当て] ページで、ポータルに追加するユーザーのチェックボックスを選択します。

   > **Note**
   >
   > IAM Identity Center をアイデンティティストアとして使用しており、 AWS Organizations 管理アカウントにサインインしている場合は、[ユーザーの作成] を選択することで IAM Identity Center ユーザーを作成することができます。IAM Identity Center から新しいユーザーに、パスワードを設定するための E メールが送信されま

す。そして、そのユーザーをユーザーとしてポータルに割り当てることができます。詳細については、「IAM Identity Center での ID の管理」を参照してください。

3. [ユーザーの割り当て] を選択します。

ポータルユーザーを削除するには

- ポータルの詳細ページで、[Portal users] (ポータルのユーザー) セクションで、ポータルから削除するユーザーのチェックボックスを選択し、[Remove from portal] (ポータルから削除) を選択します。

IAM

ポータルユーザーを追加するには

1. ポータルの詳細ページの [ポータルユーザー] セクションで [ユーザーの割り当て] を選択します。
2. [ユーザーの割り当て] ページで、次の操作を行います。
   - [IAM ユーザー] を選択して、IAM ユーザーをポータルユーザーとして追加します。
   - [IAMロール] を選択して、IAM ロールをポータルユーザーとして追加します。
3. ポータルユーザーとして追加したいユーザーまたはロールのチェックボックスを選択します。これにより、ユーザーまたはロールが [ポータルユーザー] リストに追加されます。
4. [ユーザーの割り当て] を選択します。

ポータルユーザーを削除するには

- ポータルの詳細ページで、[ポータルのユーザー] セクションで、ポータルから削除するユーザーのチェックボックスを選択し、[ポータルから削除] を選択します。

> **Important**
>
> ユーザーまたはロールがこのポータルにサインインするには `iotsitewise:DescribePortal` アクセス許可が必要です。

でポータルを削除する AWS IoT SiteWise

テスト目的でポータルを作成した場合、またはすでに存在するポータルの複製を作成した場合は、ポータルを削除することがあります。

> **Note**
>
> ポータルを削除するには、まずポータル内のすべてのダッシュボードとプロジェクトを手動で削除する必要があります。詳細については、SiteWise Monitor アプリケーションガイドの「プロジェクトの削除」および「ダッシュボードの削除」を参照してください。

1. ポータルの詳細ページで、[削除] を選択します。

> **Important**
>
> ポータルを削除すると、ポータルに含まれるすべてのプロジェクト、および各プロジェクトのすべてのダッシュボードが失われます。この操作は元に戻すことができません。アセットデータは影響を受けません。

2. [ポータルの削除] ダイアログボックスで、[管理者とユーザーの削除] を選択します。

   ポータルを削除する前に、ポータルから管理者とユーザーを削除する必要があります。ポータルに管理者またはユーザーがいない場合、ボタンは表示されず、次の手順に進むことができます。

3. ポータル全体を削除する場合は、フィールドに **`delete`** と入力して削除を確認します。

4. [削除] を選択します。

の使用を開始する AWS IoT SiteWise Monitor (AI 対応) - プレビュー

組織の AWS 管理者は、 AWS IoT SiteWise コンソールからポータルを作成し、組織のメンバーに AWS IoT SiteWise データを表示させることができます。開始するには、次の手順を実行します。

1. ポータルを設定して作成します。
2. ポータル管理者を追加し、招待メールを送信します。
3. ポータルユーザーを追加します。

ポータルを作成すると、ポータル管理者はプロジェクトを作成し、プロジェクトにユーザーを追加できます。次に、プロジェクトメンバーはダッシュボードを作成して、接続されたデータを視覚化し

AWS IoT SiteWise、接続されたデバイス、プロセス、機器の動作をモニタリングできるようにします。

> **Note**
>
> ポータルにユーザーまたは管理者を追加するときは、制限された IP など、ユーザーのアクセス許可を制限する AWS Identity and Access Management (IAM) ポリシーを作成しないでください。アクセス許可が制限されたアタッチされたポリシーは、 AWS IoT SiteWise ポータルに接続できません。

チームと共有するためのプロジェクトを作成します。その後、プロジェクト所有者はダッシュボードを作成してアセットのプロパティを視覚化し、プロジェクトビューワーがデバイス、プロセス、機器の動作を理解するのに役立ちます。また、チームにオペレーションの一貫したビューを提供します。

ダッシュボードは、プロジェクトデータを視覚化して理解するのに役立ちます。企業やアプリケーションのユーザーが AWS IoT デバイスやデータを追跡するのに役立ちます。ニーズに合わせてデータを最適に表示する視覚化タイプを選択します。視覚化を並べ替えてサイズを変更し、チームに合ったレイアウトを作成します。デバイス、プロセス、機器のアセットとデータを調べ、問題をすばやく特定して運用効率を向上させます。

トピック

- ポータルを作成する
- ポータルを設定
- ポータルを管理
- ポータルを削除する
- を使用してダッシュボードを作成する AWS CLI
- ポータルログイン
- プロジェクトを作成する
- プロジェクトを更新します。
- プロジェクトを削除します。
- ダッシュボードを作成する
- ダッシュボードの更新
- ダッシュボードの削除

- ダッシュボードを設定する

ポータルを作成する

AWS IoT SiteWise コンソールで SiteWise Monitor ポータルを作成します。

ポータルを作成するには

1. AWS IoT SiteWise コンソールにサインインします。
2. ナビゲーションペインで、モニタリング、開始を選択します。
3. ポータルの作成 (AI 対応) を選択します。

次に、ポータルを設定するために基本情報を指定する必要があります。

ポータルを設定

ユーザーはポータルを使用してデータを表示します。ポータルの名前、説明、ブランディング、ユーザー認証、サポートお問い合わせメール、アクセス許可などをカスタマイズできます。

ポータルを設定する手順：

1. ポータルの名前を入力します。
2. (オプション) ポータルの説明を入力します。複数のポータルがある場合は、わかりやすい説明を使用すると、各ポータルに含まれる内容を追跡できます。

3. (オプション) 画像をアップロードして、ブランドをポータルに表示します。正方形の PNG 画像を選択します。正方形でない画像をアップロードすると、画像は正方形に縮小されます。
4. サポートに関する問題については、サポート連絡先の E メールアドレスボックスに E メールアドレスを入力します。
5. ユーザー認証ボックスで、次のオプションを選択します。
   - ポータルのユーザーが会社のユーザー名とパスワードでこのポータルにサインインする場合は、[IAM Identity Center] を選択します。

     アカウントで IAM Identity Center を有効にしていない場合は、次のことを行います。

     a. [ユーザーの作成] を選択します。
     b. [ユーザーの作成] ページで、最初のポータルを作成するために、ユーザーの E メールアドレス、名前、姓を入力し、[ユーザーの作成] を選択します。

     > **Note**
     > IAM 認証情報のサポートは間もなく開始されます。

6. サービスアクセスセクションで、次のいずれかのオプションを選択します。
   - [新しいサービスロールの作成と使用] を選択します。デフォルトでは、SiteWise Monitor は自動的に各ポータルのサービスロールを作成します。このロールにより、ポータルユーザーは AWS IoT SiteWise リソースにアクセスできます。詳細については、「のサービスロールを使用する AWS IoT SiteWise Monitor」を参照してください。
   - 既存の[サービスロールの使用] を選択し、対象のロールを選択します。
7. このポータルの AWS IoT SiteWise Assistant を有効にするには、 を選択します。 AWS IoT SiteWise Assistant は、迅速なデータ分析、リアルタイムのインサイト、ガイド付きレコメンデーションを提供します。

   > **Note**
   > AWS IoT SiteWise Assistant を有効にすると、料金が発生します。エンタープライズレベルのナレッジソリューションとガイダンスを使用するには、Amazon Kendra インデックスに関連付けられたデータセットが必要です。

8. (オプション) ポータルにタグを追加します。詳細については、「AWS IoT SiteWise リソースにタグを付ける」を参照してください。

9. ポータルの作成を選択します。ポータル AWS IoT SiteWise を作成します。

> **Note**
>
> コンソールを閉じる場合、管理者とユーザーを追加することでセットアッププロセスを完了できます。詳細については、「ポータル管理者を追加または削除する」を参照してください。このポータルを維持しない場合は、リソースを使用しないようにするため、ポータルを削除します。詳細については、「ポータルを削除する」を参照してください。

ポータルを作成すると、メッセージが表示されます。

ポータルが作成されると、ポータルセクションに一覧表示されます。ポータルの詳細セクションには、各ポータルの名前、説明、ID、URL、ステータス、最終更新日と作成日、ポータルブランド、サポート E メールが一覧表示されます。

[状態] 欄には、次のいずれかの値を指定することができる。

- CREATING - ポータルの作成リクエスト AWS IoT SiteWise を処理しています。このプロセスは完了までに数分かかることがあります。
- UPDATING - ポータルの更新リクエスト AWS IoT SiteWise を処理しています。このプロセスは完了までに数分かかることがあります。
- 保留中 - AWS IoT SiteWise DNS レコードの伝播が終了するのを待っています。このプロセスは完了までに数分かかることがあります。ステータスが「保留中」の場合は、ポータルを削除できます。

- DELETING - ポータルの削除リクエスト AWS IoT SiteWise を処理しています。このプロセスは完了までに数分かかることがあります。
- アクティブ - ポータルがアクティブになると、ポータルユーザーはポータルにアクセスできます。
- 失敗 - ポータルの作成、更新、または削除のリクエストを処理 AWS IoT SiteWise できませんでした。が Amazon CloudWatch Logs AWS IoT SiteWise にログを送信できるようにした場合、これらのログを使用して問題のトラブルシューティングを行うことができます。詳細については、CloudWatch Logs AWS IoT SiteWise によるモニタリング」を参照してください。

ポータルを管理

ポータルのさまざまな側面を管理および設定することができます。これには、管理者の追加と削除、アクセス許可とロールの設定、名前、説明のカスタマイズ、サポート E メールの設定、ポータル管理者への招待が含まれます。

1. AWS IoT SiteWise コンソールにサインインします。
2. ナビゲーションペインで、[モニター]、[ポータル] の順に選択します。

3. ポータルを選択し、ポータルを開く (またはポータルの名前を選択する) を選択します。
4. 次のいずれかの管理タスクを実行できます。
   - ポータル属性を編集する
   - ポータル管理者を追加または削除する

- ポータル管理者に招待メールを送信する
- でポータルを削除する AWS IoT SiteWise

ポータル属性を編集する

ポータルの名前、説明、ブランド、サポート E メール、サービスアクセスを変更できます。

1. ポータルの詳細ページの [ポータルの詳細] セクションで、[編集] を選択します。

2. 名前、説明、ポータルブランド、サポート連絡先 E メール、AWS IoT SiteWise アシスタント、またはサービスアクセスを更新します。
3. 完了したら、[変更の保存] を選択します。

ポータル管理者を追加または削除する

いくつかのステップで、ポータルの管理者としてユーザーを追加または削除できます。ユーザー認証サービスに基づいて、次のいずれかを選択します。

IAM Identity Center

ポータル管理者を追加するには

1. ポータルの詳細ページの管理者セクションで、管理者の割り当てを選択します。
2. 管理者の割り当てページで、管理者としてポータルに追加するユーザーを選択します。

> **Note**
>
> IAM Identity Center をアイデンティティストアとして使用しており、 AWS Organizations 管理アカウントにサインインしている場合は、[ユーザーの作成] を選択することで IAM Identity Center ユーザーを作成することができます。IAM Identity Center から新しいユーザーに、パスワードを設定するための E メールが送信されます。そして、そのユーザーを管理者としてポータルに割り当てることができます。詳細については、「IAM Identity Center での ID の管理」を参照してください。

3. [Assign administrators] (管理者の割り当て) を選択します。

ポータル管理者を削除するには

- ポータルの詳細ページの [ポータル管理者] セクションで、削除する各ユーザーのチェックボックスをオンにし、[ポータルから削除] を選択します。

> **Note**
>
> Administrators(#) には、ポータルの管理者数が一覧表示されます。複数のポータル管理者を追加して、プロジェクトを管理および操作できます。

ポータル管理者に招待メールを送信する

ポータルサイトの管理者に招待メールを送信することができます。

1. ポータルの詳細ページの管理者セクションで、ポータル管理者のチェックボックスをオンにします。
2. [招待状を送信] を選択します。E メールクライアントが開き、招待状がメッセージ本文に入力されます。

   E メールは、ポータル管理者に送信する前にカスタマイズできます。

ポータルを削除する

テスト目的でポータルを作成した場合、またはすでに存在するポータルの複製を作成した場合は、ポータルを削除することがあります。

> **Note**
>
> ポータルを削除するには、まずポータル内のすべてのダッシュボードとプロジェクトを手動で削除する必要があります。

1. ポータルの詳細ページで、[削除] を選択します。

   > **Important**
   >
   > ポータルを削除すると、ポータルに含まれるすべてのプロジェクト、および各プロジェクトのすべてのダッシュボードが失われます。この操作は元に戻すことができません。アセットデータは影響を受けません。

2. ポータルの削除ダイアログボックスで、管理者とユーザーの削除を選択します。

   ポータルを削除する前に、ポータルから管理者とユーザーを削除する必要があります。ポータルに管理者またはユーザーがいない場合、ボタンは表示されず、次の手順に進むことができます。

3. ポータル全体を削除する場合は、フィールドに **confirm** と入力して削除を確認します。
4. [削除] を選択します。

を使用してダッシュボードを作成する AWS CLI

を使用してダッシュボードで視覚化 (またはウィジェット) を定義する場合は AWS CLI、JSON `dashboardDefinition` ドキュメントで次の情報を指定する必要があります。この定義は、CreateDashboard および UpdateDashboard オペレーションのパラメータです。

`displaySettings`

以下のパラメータを含む表示設定。

- `numRows` – ダッシュボードレイアウトの行数。各行は cellSize 全体です。
- `numColumbs` – ダッシュボードレイアウトの列数。各列は cellSize 全体です。
- `cellSize` – (オプション) レイアウト内のセルのサイズをピクセル単位で指定します。正の数である必要があります。デフォルトは 10 です。
- `significantDigits` – (オプション) ダッシュボードに表示する重要な桁数。デフォルトは 4 です。

`querySettings`

次のパラメータを持つクエリ情報。

- `refreshRate` – (オプション) データがミリ秒単位で更新されるレート。1000、5000、10000、60000、30000 の値を使用できます。

`defaultViewport`

指定しない場合、デフォルトは過去 5 分です。次のパラメータが含まれます。

- `duration` – (オプション) 現在の時刻からデータをクエリする過去の期間を決定します。
- `start` – (オプション) タイプは Date です。データをクエリする開始時間範囲。指定された`end`日付が必要です。
- `end` – (オプション) タイプは Date です。データをクエリする終了時間範囲。指定された`start`日付が必要です。

`widgets`

次の情報を含むウィジェット定義構造のリスト。

`type`

widget. のタイプ AWS IoT SiteWise には、次のウィジェットタイプが用意されています。

- `xy-plot` – 設定に応じて折れ線グラフまたは散布図。
- `bar-chart` - 棒グラフ
- `kpi-chart` – 主要なパフォーマンスインジケータチャート。
- `status-timeline` – 1 つ以上のデータソースからの時系列データを視覚化してナビゲートするステータスウィジェット。
- `text` – テキストウィジェット。
- `table` — テーブルウィジェット

`id`

ウィジェットの一意の識別子。

`x`

ダッシュボードの左から始まるウィジェットの水平位置。この値は、ダッシュボードのグリッド内のウィジェットの位置を参照します。

`y`

ダッシュボードの上部から始まるウィジェットの垂直位置。この値は、ダッシュボードのグリッド内のウィジェットの位置を参照します。

`z`

ウィジェットの相対的な順序。大きな Z 値ウィジェットは、下部の Z 値ウィジェットの前に重複している場合に表示されます。

`width`

ダッシュボード上のセル数で表されるウィジェットの幅。

`height`

ダッシュボード上のセル数で表されるウィジェットの高さ。

`properties`

ウィジェットのプロパティのリスト。ウィジェットのタイプによって異なります。詳細については、「IoT App Kit」を参照してください。

Example ダッシュボード定義の例

次の例では、JSON ファイルに保存されているペイロードからダッシュボードを定義します。

```
aws iotsitewise create-dashboard \
  --project-id a1b2c3d4-5678-90ab-cdef-eeeeeEXAMPLE \
  --dashboard-name "Example Dashboard" \
  --dashboard-definition file://dashboard-definition.json
```

次の `dashboard-definition.json` の JSON 例では、次の視覚化ウィジェットを使用してダッシュボードを定義します。

```
{
    "displaySettings": {
        "numColumns": 200,
        "numRows": 1000,
        "cellSize": 20,
        "significantDigits": 4
    },
    "widgets": [{
        "id": "Ot73JcxUoc6oEXAMPLE",
```

```
        "type": "xy-plot",
        "width": 33,
        "height": 20,
        "x": 0,
        "y": 0,
        "z": 0,
        "properties": {
            "aggregationType": "AVERAGE",
            "queryConfig": {
                "source": "iotsitewise",
                "query": {
                    "assets": [{
                        "assetId": "97c97abf-e883-47bb-a3f4-EXAMPLE",
                        "properties": [{
                            "propertyId": "97cc61f4-57a4-4c5f-a82c-EXAMPLE",
                            "refId": "692ce941-f3d9-4074-a297-EXAMPLE",
                            "aggregationType": "AVERAGE",
                            "color": "#7d2105",
                            "resolution": "1m"
                        }]
                    }],
                    "properties": [],
                    "assetModels": [],
                    "alarms": [],
                    "alarmModels": []
                }
            },
            "line": {
                "connectionStyle": "linear",
                "style": "solid"
            },
            "symbol": {
                "style": "filled-circle"
            },
            "axis": {
                "yVisible": true,
                "xVisible": true
            },
            "legend": {
                "visible": true,
                "position": "right",
                "width": "30%",
                "height": "30%",
                "visibleContent": {
```

```
                    "unit": true,
                    "asset": true,
                    "latestValue": true,
                    "latestAlarmStateValue": true,
                    "maxValue": false,
                    "minValue": false
                }
            }
        }
    }, {
        "id": "fto7rF40Ny1EXAMPLE-G",
        "type": "bar-chart",
        "width": 33,
        "height": 20,
        "x": 0,
        "y": 20,
        "z": 0,
        "properties": {
            "aggregationType": "AVERAGE",
            "queryConfig": {
                "source": "iotsitewise",
                "query": {
                    "assets": [{
                        "assetId": "97c97abf-e883-47bb-a3f4-EXAMPLE",
                        "properties": [{
                            "propertyId": "c84ca8f3-3dea-478a-afec-EXAMPLE",
                            "aggregationType": "AVERAGE",
                            "refId": "2960b958-2034-4d6e-bcc2-EXAMPLE"
                        }]
                    }],
                    "properties": [],
                    "assetModels": [],
                    "alarms": [],
                    "alarmModels": [],
                    "requestSettings": {
                        "aggregation": "AVERAGE"
                    }
                }
            },
            "axis": {
                "showX": true,
                "showY": true
            },
            "styleSettings": {
```

```
                "2960b958-2034-4d6e-bcc2-360f1f02e505": {
                    "color": "#7d2105"
                }
            }
        }
    }],
    "querySettings": {
        "refreshRate": 5000
    }
}
```

ポータルログイン

ユーザーログイン

1. ブラウザで、アプリケーション URL を入力します。
2. ユーザー名とパスワードを入力し、サインインボタンをクリックします。
3. これでアプリケーションにログインしました。

プロジェクトを作成する

プロジェクトを作成する

1. プロジェクトは 2 つの方法で作成されます。
   a. ホームページの「クイックスタート」の「ようこそ」セクションで、「プロジェクトの作成」を選択します。
   b. 左側のナビゲーションペインから、プロジェクトを選択します。右上隅で作成 を選択してプロジェクトを作成します。
2. 「プロジェクトの作成」セクションで、プロジェクト名を入力し、オプションの 説明を指定します。

3. [Create] (作成) を選択します。

プロジェクトを更新します。

プロジェクトを編集する

1. プロジェクトページの右上隅にある更新ボタンを選択して、プロジェクトの詳細を編集します。
2. プロジェクト名を編集してプロジェクト名を変更します。
3. 説明の詳細を編集して、プロジェクトの説明を変更します。
4. 更新を選択して変更を保存します。

プロジェクトを削除します。

プロジェクトの削除

1. プロジェクトを削除できるのは、プロジェクト内のすべてのダッシュボードが削除された後のみです。
2. プロジェクトページの右上にある削除ボタンを選択します。
3. プロジェクトを削除することをもう一度確認します。
4. 削除を選択してプロジェクトを削除します。

ダッシュボードを作成する

ダッシュボードを作成する

1. ダッシュボードは、次の 2 つの方法で作成します。
   a. ホームページの「ダッシュボードの構築」からダッシュボードを作成します。
      i. 既存のプロジェクトでダッシュボードを作成するには、ダッシュボードをホストするプロジェクトを選択するのドロップダウンメニューからプロジェクト名を選択します。
      ii. プロジェクトがない場合は、「プロジェクトの作成」を選択し、「確認」を選択します。

b. ダッシュボードの下のプロジェクトセクションで、プロジェクトからダッシュボードを作成します。

2. 右上隅で作成 を選択します。
3. ダッシュボード名を入力し、オプションのダッシュボードの説明を入力します。
4. [作成] を選択します。

5. 新しく作成したダッシュボードを設定します。

ダッシュボードの更新

ダッシュボードセクションには、プロジェクトのダッシュボードが一覧表示されます。リストからダッシュボードを選択します。

ダッシュボードの更新

1. 更新するダッシュボードを選択します。

2. ダッシュボード名を更新し、必要に応じてダッシュボードの説明を更新します。更新を選択して変更を保存します。

ダッシュボードの削除

ダッシュボードセクションには、プロジェクトのダッシュボードが一覧表示されます。リストからダッシュボードを選択します。

ダッシュボードの削除

1. 削除するダッシュボードを選択します。

2. 削除を選択してダッシュボードを削除します。これは元に戻すことはできません。

ダッシュボードを設定する

ダッシュボードセクションには、プロジェクトのダッシュボードが一覧表示されます。リストからダッシュボードを選択します。編集モードでは、ウィジェットを追加して設定することで、ダッシュボードを設定できます。プレビューボタンを使用すると、変更を視覚化できます。

ダッシュボードを設定する手順：

- さまざまなタイプのデータウィジェットをダッシュボードキャンバスにドラッグアンドドロップして、データを視覚化します。
- 左側の Resource Explorer から目的のウィジェットにデータを追加します。Resource explorer は、Modeled、Unmodeled、および Dynamic アセットセクションで構成されます。アセット名またはプロパティ名で検索します。追加するプロパティを選択し、追加を選択します。
- ウィジェットの設定を変更して、レイアウトとスタイルを微調整します。タイトル、しきい値、その他の設定の詳細などのコンポーネントを設定します。
- データを表示する時間範囲を設定します。

- データを表示する時間範囲を選択します。右上隅から時間範囲と更新レートを選択し、範囲をパーソナライズします。メニューからデータを更新するレートを選択します。
- トラックボールのマウススクロールホイールまたは右クリックを使用して、ウィジェットの時間範囲を選択します。これにより、表示の時間範囲が移動します。
- [Save] を選択します。

トピック

- リソースエクスプローラー
- ウィジェット
- ウィジェットを設定する
- ウィジェットを使用する
- ウィジェットのアラーム
- AWS IoT SiteWise ウィジェットでのアシスタントの使用
- AWS IoT SiteWise Assistant に尋ねる質問の例

リソースエクスプローラー

このセクションでは、モデル化アセット、モデル化されていないアセット、および動的アセットについて説明します。3 つのアセットのいずれかを選択し、ウィジェットに追加して視覚化します。

トピック

- モデル化
- モデル化されていない
- 動的アセット

モデル化

このセクションでは、モデル化されたアセットの選択と視覚化のプロセスについて説明します。

アセットの選択

アセットは次のようにクエリできます。

- アセット名を検索します。ワイルドカード を使用します*。たとえば、 は、テキスト で始まるアセット名`Wind*`を返します`Wind`。この機能を使用するには、 と統合 AWS IoT TwinMakerする必要があります。
- デフォルトでは、すべてのアセットが一覧表示されます。

リストされているアセットから、名前、説明、ID、またはアセットモデル ID でフィルタリングします。アセットを 1 つ選択して、そのプロパティ (データストリーム) とアラームを一覧表示します。

データストリームの選択

データストリームは、データストリームメニューの下に表示されます。https://docs.aws.amazon.com/iot-sitewise/latest/APIReference/「https://www.com のプロパティメタデータ」でリストされているデータストリームをフィルタリングします。選択したウィジェットに応じて、1 つ以上のデータストリームを選択します。

- KPI とゲージは 1 つのデータストリームのみをサポートします。
- 残りのウィジェットは、複数選択で複数のデータストリームをサポートします。

アラームの選択

AWS IoT SiteWise アラームは、アラームデータストリームメニューの下に表示されます。アラームメタデータでリストされているアラームデータストリームをフィルタリングします。 名、入力プロパティ、複合モデル ID は、フィルタリングに使用されるメタデータの一部です。選択したウィジェットに応じて、1 つ以上のデータストリームを選択します。

- KPI とゲージは 1 つのアラームのみをサポートします。
- 残りのウィジェットは、複数選択の複数のアラームをサポートしています。

モデル化されたアセットの視覚化

1. ウィジェットをキャンバスにドラッグします。各ウィジェットパネルのプロパティを選択して、ダッシュボードを作成します。
2. Filter オプションは、アセットをフィルタリングして、視覚化するアセットを選択します。フィルタリングは、テキスト、プロパティ、または値によって行われます。フィルタリングは、ブラウザにロードされたアセット用であり、バックエンドフィルタリング用ではありません。
3. ウィジェットに追加するアセットを検索して一覧表示します。

4. キャンバスのウィジェットにアセットを追加します。
5. リセットを選択して別のアセットを選択するか、選択したアセットを変更します。
6. ダッシュボードを保存します。プレビューモードでは、ドロップダウンメニューから異なるアセットを選択して、データパネルを再構築せずに各アセットのプロパティをモニタリングします。

> **Note**
>
> 右側の設定ホイールには、ページサイズ、スティッキーの最初の列、スティッキーの最後の列、列の設定など、ユーザーが選択できる設定が表示されます。設定をカスタマイズし、確認を選択して変更を適用します。

モデル化されていない

このセクションでは、モデル化されていないデータストリームを検索し、ウィジェットに追加して視覚化する方法について説明します。

モデル化されていないデータストリームの視覚化

1. ウィジェットをキャンバスにドラッグします。各ウィジェットパネルのプロパティを選択して、ダッシュボードを作成します。
2. モデル化されていないデータストリームは、時系列セクションに一覧表示されます。カスタマイズ可能なプロパティがあります。
3. Filter オプションは、視覚化するデータストリームをフィルタリングします。フィルタリングは、ブラウザにロードされたデータストリーム用であり、バックエンドフィルタリング用ではありません。
4. データストリームをキャンバスのウィジェットに追加します。
5. データストリームの選択を解除するには、リセットを選択します。
6. ダッシュボードを保存します。プレビューモードでは、ドロップダウンメニューから異なるアセットを選択して、データパネルを再構築せずに各アセットのプロパティをモニタリングします。

> **Note**
>
> 右側の設定ホイールには、ページサイズ、スティッキーファースト列、スティッキーラスト列、列設定など、ユーザーが選択できる設定が表示されます。設定をカスタマイズし、確認を選択して変更を適用します。

動的アセット

新しい SiteWise Monitor では、選択したアセットモデルのアセットを動的に切り替えることができます。ドロップダウンメニューから選択することで、さまざまなアセットのプロパティを視覚化できます。

動的アセットの視覚化

1. リソースエクスプローラーの動的アセットタブを選択します。
2. ドロップダウンメニューからアセットモデルを選択して、 のアセットを一覧表示します。
3. ドロップダウンメニューからデフォルトのアセットを選択します。
4. アセットモデルの設定 を選択して、アセットモデルを選択します。
5. ダッシュボードを保存します。プレビューモードでは、ドロップダウンメニューから異なるアセットを選択して、データパネルを再構築せずに、各アセットのプロパティをモニタリングします。

ウィジェット

ウィジェットは、アラーム、高性能ライブストリーミング、他の IoT App Kit コンポーネントとのスムーズな同期など、幅広い機能をサポートしています。ダッシュボードでは、次のウィジェットがサポートされています。

- 線 – 線ウィジェットは、時間の経過に伴う傾向と変化を表示する視覚化ウィジェットです。これは、直線の線セグメントで接続された点またはマーカーで表される一連のデータポイントで構成され、折れ線グラフを作成します。アラーム、しきい値、高性能ライブストリーミング、他の IoT App Kit コンポーネントとのスムーズな同期など、幅広い機能をサポートしています。このウィジェットは、複雑なデータを明確かつ簡潔に通信するようにカスタマイズできます。
- 棒グラフ – 棒グラフは、時系列データを表示する強力な視覚化ツールです。アラーム、高性能ライブストリーミング、他の IoT App Kit コンポーネントとのスムーズな同期など、幅広い機能をサポートしています。

- タイムライン – タイムラインウィジェットは、データソースから時系列データを視覚化してナビゲートする方法を提供します。データストリームの値がタイムラインで異なる色であることを表示するのに一意です。アラーム、高性能ライブストリーミング、他の IoT App Kit コンポーネント間でのスムーズな同期など、豊富な機能をサポートしています。数値以外のデータ型を表示するのに最適です。
- KPI – 主要業績評価指標 (KPI) コンポーネントは、アセットプロパティの概要をコンパクトに表現します。アラームとしきい値をサポートしています。この概要は、デバイス、機器、およびプロセスの全体的なパフォーマンスに関する重要なインサイトを提供します。KPI は 1 つのデータストリームまたはアラームのみをサポートし、複数のデータストリームはサポートしません。
- ゲージ – ゲージコンポーネントは、アセットプロパティの概要をコンパクトに表現します。これは、デバイス、機器、またはプロセスの全体的なパフォーマンスに関する重要なインサイトを視覚化するために使用されます。機能的には KPI と同じですが、視覚的には異なります。ゲージには、データストリームの値、しきい値、値の範囲が表示されます。Gauge を使用して、1 つ以上のデータソースからの AWS IoT データを操作できます。
- テーブル – テーブルコンポーネントは、1 つ以上の時系列データソースからの 1 つ以上のデータストリームを表示するためのコンパクトな形式を提供します。プロパティ、最新の値、および単位を含むアセットが表形式で表示されます。 AWS IoT SiteWise アラームをサポートします。
- テキスト – テキストウィジェットは、さまざまな色とフォントでテキストを書き込むのに役立ちます。テキストを URL に関連付けることでリンクを作成できます。このウィジェットでは、プロパティフィールドとしきい値フィールドは有効になっていません。

ウィジェットを設定する

ウィジェットがダッシュボードに追加されると、右側のパネルの設定アイコンを選択してウィジェットを設定できます。

- スタイル – ウィジェットタイトルにタイトルを追加します。ウィジェットの設定は異なります。以下にいくつかの例を示します。
  - 棒ウィジェット：
    - 解像度と集計 – 解像度と集計の値をここで設定します。

  - フォーマットデータ – 表示する小数点以下の桁数に小数点以下の桁数を設定します。
  - 表示スタイル – 表示する値を選択します。
  - 軸 — 軸を表示する場合に選択します。
- 線ウィジェット：
  - 解決と集約 – 解決と集約の値をここで設定します。
  - フォーマットデータ – 表示する小数点以下の桁数に小数点以下の桁数を設定します。
  - Y 軸 – ラベル、最小値、最大値を追加します。
  - ウィジェットスタイル – 線タイプ、線スタイル、線の太さ、データポイントシェイプ値を選択します。
  - 凡例 – 整列と表示を選択します。
- ゲージウィジェット：
  - 解像度と集計 – 解像度と集計の値をここで設定します。
  - フォーマットデータ – 表示する小数点以下の桁数に小数点以下の桁数を設定します。
  - 表示スタイル – 表示する値を選択します。
  - Y 軸 – ラベル、最小値、最大値を追加します。
  - フォント – フォントサイズ、ユニットフォントサイズ、ラベルフォントサイズ値を選択します。

- プロパティ – ウィジェットのすべてのプロパティがこのセクションに表示されます。ウィジェットごとにプロパティが異なります。以下にいくつかの例を示します。
  - 線ウィジェット：
    - ラベル – デフォルトのデータストリーム名を使用するか、新しい名前を付けるかを選択します。
    - スタイル – 行タイプ、行スタイルを表示する小数に設定します。
    - Y 軸 – デフォルトのスタイルで値を選択し、Y 軸コントロールを表示し、最小値と最大値を設定します。
  - テーブルウィジェット：
    - ラベル – デフォルトのデータストリーム名を使用するか、新しい名前を付けるかを選択します。
  - テーブルウィジェット：
    - ラベル – デフォルトのデータストリーム名を使用するか、新しい名前を付けるかを選択します。

- しきい値 – ウィジェットのしきい値を追加します。ウィジェットの設定は異なります。以下にいくつかの例を示します。
  - 棒グラフウィジェット：
    - ウィジェットに追加するしきい値の追加を選択します。
    - Operator を選択し、しきい値の値を指定します。カラーパレットの色でしきい値をカスタマイズします。

  - すべてのデータにしきい値を適用できます。
- 線ウィジェット：
  - ウィジェットに追加するしきい値の追加を選択します。
  - Operator を選択し、しきい値の値を指定します。カラーパレットの色でしきい値をカスタマイズします。
  - ドロップダウンメニューからしきい値を表示する方法を選択します。
- ゲージウィジェット：
  - ウィジェットに追加するしきい値の追加を選択します。
  - Operator を選択し、しきい値の値を指定します。カラーパレットの色でしきい値をカスタマイズします。

ウィジェットを使用する

ダッシュボード内のウィジェットは、個別に使用することも、複数選択することもできます。

ダッシュボードでウィジェットを編集する

1 つのウィジェットを選択して編集します。ダッシュボードで複数のウィジェットを編集するには、Shift + 左クリックし、ダッシュボード内のすべてのウィジェットを選択します。選択すると、ユーザーは新しいデータストリームを追加し、スタイル設定でウィジェットのタイトルを変更できます。ダッシュボード内のすべてのウィジェットのタイトルが変更されます。

キャンバスを右クリックし、以下を実行します。

- Copy – ウィジェットのコピーをキャンバスに追加します。

- 削除 – ウィジェットを削除します。
- Bring to front – 選択したウィジェットをキャンバスの前面に移動します。
- Send to back – 選択したウィジェットをキャンバスの背面に送信します。

ウィジェットのサイズ変更

ウィジェットのサイズを個別に変更するか、ダッシュボードでウィジェットを複数選択してグループ内で変更します。

ウィジェットのサイズを変更するには：

- 1 つのウィジェットのサイズを変更するには、ウィジェットを選択し、角でドラッグしてサイズを変更します。
- 複数のウィジェットのサイズを変更するには、Shift + 左クリックで複数のウィジェットを選択し、角でドラッグしてサイズを変更します。

ダッシュボードでウィジェットを削除する

ダッシュボードでウィジェットを複数選択して、ウィジェットを個別に削除するか、グループ内で削除します。

ウィジェットを削除するには：

- 1 つのウィジェットを削除するには、ウィジェットを選択し、右クリックして削除を選択します。また、選択して右上隅の X をクリックしてウィジェットを削除することもできます。
- 複数のウィジェットを削除するには、Shift + 左クリックで複数のウィジェットを選択し、右クリックして削除を選択します。

ウィジェットのアラーム

アラームは、機器やプロセスが最適な状態でない場合に、お客様とチームに警告を発します。機器やプロセスの最適なパフォーマンスとは、特定のメトリクスの値が上限と下限の範囲内にあることを指します。これらのメトリクスが動作範囲外になった場合、装置のオペレータに通知し、問題を解決する必要があります。アラームを使用すると、問題を迅速に特定し、オペレーターに通知して、機器やプロセスのパフォーマンスを最大限に高めることができます。

アセットに関連付けられたアラームは、Resource Explorer の Modeled タブにあります。

- アセットを検索して選択します。
- データストリームテーブルを下にスクロールしてアラームデータストリームセクションに移動し、展開します。
- アラームテーブルでアラームを選択し、追加を選択します。

トピック

- さまざまなウィジェットのアラーム

さまざまなウィジェットのアラーム

すべてのウィジェットの場合：

- データストリームのプロパティ設定は、ウィジェットに追加するプロパティのタイプによって異なります。データストリームプロパティはプロパティ設定を完全にサポートしていますが、アラームプロパティは現在プロパティ設定設定を許可していません。
- アラームデータストリームを追加すると、関連する入力プロパティデータストリームもグラフに追加されます。アラームデータストリームを削除すると、その入力プロパティも削除されます。
- アラームの入力プロパティデータストリームを個別に制御するには、両方を個別に追加する必要があります。

以下の例では、一部のウィジェットでアラームがどのように使用されるかを示しています。

- 折れ線グラフ
  - アラームとその入力プロパティデータストリームがグラフに追加されます。
  - グラフの凡例にアラーム状態が表示され、アラームの状態が変わったときにデータストリームにカーソルを合わせているアイコンが表示されます。
  - グラフ設定からアラームアイコンをオフにできます。

- KPI とゲージ
  - アラームとその入力プロパティデータストリームが、選択したウィジェットに追加されます。
  - アラームしきい値がウィジェットに追加され、設定に基づいて色が変わります。
  - ウィジェットでアラーム状態を選択し、アラームの詳細を表示し、概要の生成をクリックしてを呼び出し AWS IoT SiteWise てアラームの概要を取得できます。

- [テーブル]

  - アラームとその入力プロパティは、テーブルの行として追加されます。
- 棒グラフ
  - アラームはしきい値としてグラフに追加され、しきい値を超えるデータストリームの色が変更されます。
  - 関連するデータストリームは個別に追加できます。
  - ウィジェットから AWS IoT SiteWise Assistant を操作することはできません。

- ステータスタイムライン
  - アラームはタイムラインにしきい値として追加されます。
  - アラーム状態とその入力プロパティデータをタイムラインに追加する作業が進行中です。
  - ウィジェットから AWS IoT SiteWise Assistant を操作することはできません。

AWS IoT SiteWise ウィジェットでのアシスタントの使用

Assistant AWS IoT SiteWise は生成 AI を活用したアシスタントです。これにより、プラントマネージャー、品質エンジニア、メンテナンス技術者などのユーザーは、運用データとエンタープライズデータから直接インサイトを得て、問題を解決し、アクションを実行できます。 AWS IoT SiteWise

Assistant は、 AWS IoT データ、アセットモデル、マニュアル、およびドキュメントからの情報を、重要なイベントのわかりやすい概要に統合します。また、インタラクティブな詳細な質問と回答セッションが可能になり、診断、根本原因の調査、ガイド付きレコメンデーションが容易になります。

AWS IoT SiteWise Assistant ボタンはダッシュボードの右上隅にあります。これをクリックして Assistant をアクティブ化します。ダッシュボードのプレビューモードでのみ使用できます。

Assistant は、次のシナリオ AWS IoT SiteWise で使用します。

トピック

- ユースケース - アラームの概要
- ユースケース - 状況概要
- ユースケース - 詳細な概要

ユースケース - アラームの概要

ダッシュボードで選択したパネルの現在のアラームを要約します。アラームは、ラインウィジェット、KPI ウィジェット、ゲージウィジェット、テーブルウィジェットでサポートされています。アラームを含むウィジェットを選択し、要約します。

- ウィジェットでアクティブアラームを選択します。
- アラームの重要度とルールの式が表示されます。
- サマリーを生成する を選択して、サマリーを生成します。

ユースケース - 状況概要

要約するウィジェットを最大 3 つ選択します。ウィジェットとプロパティの組み合わせにすることができます。3 つ以上を選択すると、 Assistant はエラーを返します。

AWS IoT SiteWise Assistant を使用して状況概要を生成する

1. AI Assistant をクリックします。3 つのオプションを含むメニューが表示されます。
   a. 選択した項目 – 3 つのみを選択します。3 つ以上を選択することはできません。
   b. すべてをクリア – 選択をクリアします。
   c. Generate summary – 選択した項目に関する概要を生成します。
2. サマリーの生成 を選択して、選択した項目に関するサマリーを生成します。

次の図では、ウィジェットが選択されており、 AWS IoT SiteWise Assistant からの概要が表示されています。

ユースケース - 詳細な概要

これは、ユーザーが詳細な調査を行い、SOPs (標準運用手順) 、マニュアル、ドキュメントにアクセスし、次のステップを検討できるユースケースです。前のセクションの例では、ユーザーがこのプロパティの SOP の詳細を学習することを選択した場合、このプロパティの SOP についてアシスタントに質問します。これにより、SOP に関する詳細な情報がユーザーに表示されます。

次の例では、「アラーム windSpeedAlarm の SOP はありますか？」の回答を表示します。

AWS IoT SiteWise Assistant に尋ねる質問の例

> **Note**
>
> - AWS IoT SiteWise Assistant は、エンタープライズレベルの知識とガイダンスのために Amazon Kendra インデックスを持つデータセットを使用する必要があります。Amazon Kendra インデックスがない場合は、「インデックスを作成して作成」を参照してください。データセットを追加すると、アシスタントのレスポンスの品質が向上します。詳細については、「データセットの作成」を参照してください。
> - 一部の質問では AWS IoT TwinMaker 、統合が必要です。詳細については、「Integrate AWS IoT TwinMaker and AWS IoT SiteWise」を参照してください。

ダッシュボードでアラームの概要を取得した後、同じ会話の一部として Assistant に尋ねるフォローアップの質問もあります。

- 上記の概要からアセットの詳細を表示しますか？
- ルートから上記のアセットへの階層パスは何ですか？
- 言及されたアセットの依存する子孫アセットは何ですか？

- アクティブなアラームを持つ、言及されたアセットの依存アセットは何ですか？
- アクティブなアラームを持つすべてのアセットを検索します。

ダッシュボードでプロパティの概要を取得した後、同じ会話の一部として Assistant に尋ねるフォローアップの質問もあります。

- 過去 24 時間に同じ分析を実行します。
- 上記のプロパティに関連するドキュメントを検索します。
- アセット ID 1da67d28-14f8-4f71-a06a-386f0425a21d/asset名デモタービンアセット 1 の詳細を入力します。

API から AWS IoT SiteWise Assistant を呼び出します。

- アセット ID のアラーム名 windSpeedAlarm のアラーム概要を生成しますd591e153-e5cf-4206-96bb-ce3c119d9d2d。
- アセット ID のアラーム名 windSpeedAlarm について、過去 12 時間 / 2 日 / 1 週間のアラーム概要を生成しますd591e153-e5cf-4206-96bb-ce3c119d9d2d。
- アセット ID ab187fb7-d74b-44d9-bd9b-f2f19a9137ccのプロパティ ID のプロパティ概要を生成する d591e153-e5cf-4206-96bb-ce3c119d9d2d
- ab187fb7-d74b-44d9-bd9b-f2f19a9137cc アセット ID のプロパティ ID について、過去 12 時間 / 2 日 / 1 週間のプロパティ概要を生成しますd591e153-e5cf-4206-96bb-ce3c119d9d2d。
- アセット名 Turbine のアセットを検索します。
- アセット ID 、 5356168c-3390-456f-802c-9f6e047810d4のプロパティ ID の現在のプロパティ値を教えてくださいd591e153-e5cf-4206-96bb-ce3c119d9d2d3cbb084e-1ded-4b08-9f21-1b47b2fb86fd。
- アセット ID d591e153-e5cf-4206-96bb-ce3c119d9d2dとアセット ID の関係は何ですか3cbb084e-1ded-4b08-9f21-1b47b2fb86fd。
- 風力タービンの低 RPM 問題を解決する方法に関するドキュメントをご覧ください。
- プロパティエイリアス WindSpeed のプロパティ概要を生成します。
- ナレッジベースに従って、運用前チェックとは何ですか？

からのデータのクエリ AWS IoT SiteWise

AWS IoT SiteWise API オペレーションを使用して、特定の時間間隔におけるアセットプロパティの現在の値、履歴値、集計をクエリできます。 は、さまざまな統合ニーズを満たすために複数のクエリインターフェイス AWS IoT SiteWise を提供します。

- 直接 API オペレーション - 特定のデータ取得ニーズに対するシンプルでターゲットを絞った API コール
- SQL のようなクエリ言語 - 複雑なデータ分析のための強力で柔軟なクエリ
- ODBC ドライバー - ビジネスインテリジェンスツールおよびアプリケーションとの統合

これらのクエリ機能を使用して、次の操作を行います。

- 運用データに関するリアルタイムのインサイトを取得する
- 過去の傾向とパターンを分析する
- 産業アセット全体のパフォーマンスメトリクスを計算する
- IoT データをエンタープライズシステムやダッシュボードと統合する
- 産業データを活用するカスタムアプリケーションを構築する

たとえば、特定のプロパティ値を持つすべてのアセットを検出したり、データのカスタム表現を構築したり、 AWS IoT SiteWise アセットに保存されている産業データと統合するソフトウェアソリューションを開発したりできます。また、 AWS IoT SiteWise Monitorではリアルタイムのアセットデータを調べることもできます。SiteWise Monitor の設定方法については、「でデータをモニタリングする AWS IoT SiteWise Monitor」を参照してください。

このセクションで説明するオペレーションは、タイムスタンプ、品質、値 (TQV) 構造を含むプロパティ値オブジェクトを返します。

- `timestamp` には、ナノ秒のオフセットがある現在の UNIX エポック時刻 (秒単位) が含まれます。
- `quality` には、データポイントの品質を示す、以下のいずれかの文字列が含まれます。
  - `GOOD` - データはいずれの問題の影響も受けません。
  - `BAD` - データはセンサーの障害などの問題による影響を受けます。
  - `UNCERTAIN` - データはセンサーの不正確さなどの問題による影響を受けます。
- `value` には、プロパティのタイプに応じて、以下のいずれかのフィールドが含まれます。

- `booleanValue`
- `doubleValue`
- `integerValue`
- `stringValue`
- `nullValue`

トピック

- の現在のアセットプロパティ値をクエリする AWS IoT SiteWise
- の履歴アセットプロパティ値をクエリする AWS IoT SiteWise
- でアセットプロパティの集計をクエリする AWS IoT SiteWise
- AWS IoT SiteWise クエリ言語
- クエリの最適化
- ODBC

の現在のアセットプロパティ値をクエリする AWS IoT SiteWise

このチュートリアルでは、アセットプロパティの現在の値を取得する 2 つの方法を示します。 AWS IoT SiteWise コンソールを使用するか、 AWS Command Line Interface () で API を使用できます AWS CLI。

トピック

- アセットプロパティの現在の値をクエリ (コンソール)
- アセットプロパティの現在の値をクエリ (AWS CLI)

アセットプロパティの現在の値をクエリ (コンソール)

AWS IoT SiteWise コンソールを使用して、アセットプロパティの現在の値を表示できます。

アセットプロパティの現在の値を取得するには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [Assets (アセット)] を選択します。
3. クエリするプロパティを持つアセットを選択します。

4. 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。
5. プロパティの種類のタブを選択します。たとえば、測定プロパティの現在の値を表示するには、[測定] を選択します。
6. 表示するプロパティを検索します。現在の値が [最新の値] 列に表示されます。

アセットプロパティの現在の値をクエリ (AWS CLI)

AWS Command Line Interface（AWS CLI) を使用して、アセットプロパティの現在の値をクエリできます。

GetAssetPropertyValue オペレーションを使用して、アセットプロパティの現在の値をクエリします。

アセットプロパティを特定するには、次のいずれかを指定します。

- データが送信されるアセットプロパティの `assetId` と `propertyId`。
- データストリームのエイリアスである `propertyAlias` (例えば、`/company/windfarm/3/turbine/7/temperature`)。このオプションを使用するには、最初にアセットプロパティのエイリアスを設定する必要があります。プロパティのエイリアスを設定するには、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

アセットプロパティの現在の値を取得するには (AWS CLI)

- 次のコマンドを実行して、アセットプロパティの現在の値を取得します。*asset-id* をアセットの ID に置き換え、*property-id* をプロパティの ID に置き換えます。

```
aws iotsitewise get-asset-property-value \
  --asset-id asset-id \
  --property-id property-id
```

オペレーションは、プロパティの現在の TQV を含むレスポンスを次の形式で返します。

```
{
  "propertyValue": {
    "value": {
      "booleanValue": Boolean,
      "doubleValue": Number,
      "integerValue": Number,
```

```
      "stringValue": "String",
      "nullValue": {
          "valueType": "String"
      }
    },
    "timestamp": {
      "timeInSeconds": Number,
      "offsetInNanos": Number
    },
    "quality": "String"
  }
}
```

の履歴アセットプロパティ値をクエリする AWS IoT SiteWise

AWS IoT SiteWise API GetAssetPropertyValueHistory オペレーションを使用して、アセットプロパティの履歴値をクエリできます。

アセットプロパティを特定するには、次のいずれかを指定します。

- データが送信されるアセットプロパティの `assetId` と `propertyId`。
- データストリームのエイリアスである `propertyAlias` (例えば、`/company/windfarm/3/turbine/7/temperature`)。このオプションを使用するには、最初にアセットプロパティのエイリアスを設定する必要があります。プロパティのエイリアスを設定するには、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

以下のパラメータのいずれかを渡して、結果を絞り込みます。

- `startDate` - 履歴データをクエリる範囲の開始点を (Unix エポックタイムで秒単位)。
- `endDate` - 履歴データをクエリする範囲の終端を (Unix エポックタイムで秒単位)。
- `maxResults` - 1 回のリクエストで返す結果の最大数。デフォルトは `20` 結果です。
- `nextToken` - このオペレーションの前の呼び出しから返されたページ割りトークン。
- `timeOrdering` - 返された値に適用する順序。`ASCENDING` または `DESCENDING`。
- `qualities` – 結果をフィルタリングする品質: `GOOD`、`BAD` または `UNCERTAIN`。

アセットプロパティの値の履歴をクエリするには (AWS CLI)

1. 次のコマンドを実行して、アセットプロパティの値の履歴を取得します。このコマンドは、特定の 10 分間隔でプロパティの履歴をクエリします。*asset-id* をアセットの ID に置き換え、*property-id* をプロパティの ID に置き換えます。日付パラメータをクエリする間隔に置き換えます。

```
aws iotsitewise get-asset-property-value-history \
  --asset-id asset-id \
  --property-id property-id \
  --start-date 1575216000 \
  --end-date 1575216600
```

オペレーションは、プロパティの履歴 TQV を含むレスポンスを次の形式で返します。

```
{
  "assetPropertyValueHistory": [
    {
      "value": {
        "booleanValue": Boolean,
        "doubleValue": Number,
        "integerValue": Number,
        "stringValue": "String",
        "nullValue": {
            "valueType": "String"
        }
      },
      "timestamp": {
        "timeInSeconds": Number,
        "offsetInNanos": Number
      },
      "quality": "String"
    }
  ],
  "nextToken": "String"
}
```

2. より多くの値エントリが存在する場合、nextToken フィールドからページ割りトークンを後続の GetAssetPropertyValueHistory オペレーションの呼び出しに渡すことができます。

でアセットプロパティの集計をクエリする AWS IoT SiteWise

AWS IoT SiteWise は、集計されたアセットプロパティ値を自動的に計算します。これは、複数の時間間隔で計算される一連の基本メトリクスです。 は、アセットプロパティの分、時間、日ごとに次の集計を AWS IoT SiteWise 計算します。

- [average] (平均) - 時間間隔におけるプロパティ値の平均 (中間)。
- [count] (カウント) - 時間間隔におけるプロパティのデータポイント数
- [maximum] (最大) - 時間間隔におけるプロパティの最大値。
- [minimum] (最小) - 時間間隔におけるプロパティの最小値。
- [standard deviation] (標準偏差) - 時間間隔におけるプロパティ値の標準偏差。
- [sum] (合計) - 時間間隔におけるプロパティ値の合計。

文字列やブール値などの数値以外のプロパティの場合、 はカウント集計のみを AWS IoT SiteWise 計算します。

アセットデータのカスタムメトリクスを計算することもできます。メトリクスプロパティを使用して、オペレーション固有の集計を定義します。メトリクスプロパティは、 AWS IoT SiteWise API 用に事前計算されていない追加の集計関数と時間間隔を提供します。詳細については、「プロパティおよびその他のアセットからのデータを集計する (メトリクス)」を参照してください。

トピック

- アセットプロパティの集計 (API)
- アセットプロパティの集計 (AWS CLI)

アセットプロパティの集計 (API)

AWS IoT SiteWise API を使用して、アセットプロパティの集計を取得します。

GetAssetPropertyAggregates オペレーションを使用して、アセットプロパティの集計をクエリします。

アセットプロパティを特定するには、次のいずれかを指定します。

- データが送信されるアセットプロパティの `assetId` と `propertyId`。
- データストリームのエイリアスである `propertyAlias` (例えば、`/company/windfarm/3/turbine/7/temperature`)。このオプションを使用するには、最初にアセットプロパティのエイ

リアスを設定する必要があります。プロパティのエイリアスを設定するには、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

次の必須パラメータを渡す必要があります。

- `aggregateTypes` - 取得する集計のリスト。`AVERAGE`、`COUNT`、`MAXIMUM`、`MINIMUM`、`STANDARD_DEVIATION`、`SUM` のいずれかを指定できます。
- `resolution` – メトリクスを取得する時間間隔: `1m` (1 分) `15m`、 (15 分) `1h`、 (1 時間) 、または `1d` (1 日) 。
- `startDate` - 履歴データをクエリる範囲の開始点を (Unix エポックタイムで秒単位)。
- `endDate` - 履歴データをクエリする範囲の終端を (Unix エポックタイムで秒単位)。

次のパラメータのいずれかを渡して、結果を絞り込むこともできます。

- `maxResults` - 1 回のリクエストで返す結果の最大数。デフォルトは `20` 結果です。
- `nextToken` - このオペレーションの前の呼び出しから返されたページ割りトークン。
- `timeOrdering` - 返された値に適用する順序。`ASCENDING` または `DESCENDING`。
- `qualities` – 結果をフィルタリングする品質: `GOOD`、`BAD` または `UNCERTAIN`。

> **Note**
>
> GetAssetPropertyAggregates オペレーションは、このセクションで説明する他のオペレーションとは異なる形式の TQV を返します。`value` 構造には、リクエスト内の各 `aggregateTypes` のフィールドが含まれます。`timestamp` には、集計が発生した時間 (秒単位の Unix エポック時間) が含まれます。

アセットプロパティの集計 (AWS CLI)

アセットプロパティの集計をクエリするには (AWS CLI)

1. アセットプロパティの集計を取得するには、次のコマンドを実行します。このコマンドは、特定の 1 時間間隔で 1 時間の分解能で平均と合計をクエリします。*`asset-id`* をアセットの ID に置

き換え、*property-id* をプロパティの ID に置き換えます。パラメータをクエリする集計と間隔に置き換えます。

```
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
```

オペレーションは、プロパティの履歴 TQV を含むレスポンスを次の形式で返します。レスポンスには、リクエストされた集計のみが含まれます。

```
{
  "aggregatedValues": [
    {
      "timestamp": Number,
      "quality": "String",
      "value": {
        "average": Number,
        "count": Number,
        "maximum": Number,
        "minimum": Number,
        "standardDeviation": Number,
        "sum": Number
      }
    }
  ],
  "nextToken": "String"
}
```

2. より多くの値エントリが存在する場合、nextToken フィールドから後続の GetAssetPropertyAggregates オペレーションの呼び出しにページ割りトークンを渡すことができます。

> **Note**
>
> クエリ範囲にnull値 TQVsAssetPropertyValue API」を参照してください。count を除くすべての統計は、文字列 TQVs の統計と同様にnullレスポンスになります。クエリ範囲にダ

ブルタイプ TQVsDouble.NaNが含まれている場合、カウントを除くすべての計算は になりますDouble.NaN。

AWS IoT SiteWise クエリ言語

AWS IoT SiteWise データ取得 ExecuteQuery API オペレーションを使用すると、宣言的な構造定義に関する情報と、それに関連付けられた時系列データを以下から取得できます。

- モデル
- アセット
- 測定値
- メトリクス
- 変換
- 集計

これは、1 つの API リクエストで、クエリステートメントのような SQL を使用して実行できます。

> **Note**
> この機能は、 AWS GovCloud (米国西部) 、カナダ (中部) 、中国 (北京) 、米国東部 (オハイオ) を除く AWS IoT SiteWise 、 が利用可能なすべての リージョンで使用できます。

トピック

- のクエリ言語リファレンス AWS IoT SiteWise

のクエリ言語リファレンス AWS IoT SiteWise

AWS IoT SiteWise は、データを操作するための豊富なクエリ言語をサポートしています。使用可能なデータ型、演算子、関数、およびコンストラクトについて、以下のトピックで説明します。

クエリ言語で AWS IoT SiteWise クエリを記述クエリの例するには、「」を参照してください。

トピック

- クエリリファレンスビュー
- サポートされているデータ型
- サポートされている句
- 論理演算子
- 比較演算子
- SQL 関数
- クエリの例

クエリリファレンスビュー

このセクションでは、プロセスメタデータやテレメトリデータなど AWS IoT SiteWise、 のビューを理解するのに役立つ情報を提供します。

次の表に、ビューの名前と説明を示します。

データモデル

ビュー名	ビューの説明
アセット	アセットとモデルの派生に関する情報を含んでいます。
asset_property	アセットプロパティの構造に関する情報を含んでいます。
raw_time_series	時系列の履歴データを含んでいます。
latest_value_time_series	時系列の最新値を含んでいます。
precomputed_aggregates	自動的に計算された集計されたアセットプロパティ値を含んでいます。これらは、複数の時間間隔で計算される一連の基本メトリクスです。

次のビューは、各ビューの列名とデータ型を一覧表示します。

View:asset

列名	datatype
asset_id	文字列
asset_name	文字列
asset_description	文字列
asset_model_id	文字列
parent_asset_id	文字列
asset_external_id	文字列
asset_model_external_id	文字列
hierarchy_id	文字列

View:asset_property

列名	datatype
asset_id	文字列
property_id	文字列
property_name	文字列
property_alias	文字列
property_external_id	文字列
asset_composite_model_id	文字列
property_type	文字列

列名	datatype			
property_data_type	文字列			
int_attribute_value	integer			
double_attribute_value	double			
ブール値	boolean			
string_attribute_value	文字列			

View:raw_time_series

列名	datatype
asset_id	文字列
property_id	文字列
property_alias	文字列
event_timestamp	timestamp
quality	文字列
boolean_value	boolean
int_value	integer
double_value	double
string_value	文字列

View:latest_value_time_series

列名	datatype
asset_id	文字列
property_id	文字列
property_alias	文字列
event_timestamp	timestamp
quality	文字列
boolean_value	boolean
int_value	integer
double_value	double
string_value	文字列

View:precomputed_aggregates

列名	datatype
asset_id	文字列
property_id	文字列
property_alias	文字列
event_timestamp	timestamp
quality	文字列
resolution	文字列
sum_value	double
count_value	integer

列名	datatype
average_value	double
maximum_value	double
minimum_value	double
stdev_value	double

サポートされているデータ型

AWS IoT SiteWise クエリ言語は、次のデータ型をサポートしています。

スカラー値

データ型	説明
`STRING`	最大長 1024 バイトの文字列。
`INTEGER`	`-2,147,483,648 to 2,147,483,647` からの範囲の符号付き 32 ビット整数。
`DOUBLE`	の範囲が の浮動小数点数`-10^100 to 10^100`、または`IEEE 754`倍精度`Nan`の浮動小数点数。
`BOOLEAN`	`true` または `false`
`TIMESTAMP`	ISO-8601 準拠のタイムスタンプ： • `'yyyy-MM-dd HH:mm:ss[.SSS] '` • `TIMESTAMP 'yyyy-MM-dd[\s|T]HH:mm:ss[.SSS][+HH:mm|'Z'] '` • `'yyyy-MM-dd'T'HH:mm:ss[.SSS][+HH:mm|'Z'] '` • `'yyyy-MM-dd'T'HH:mm:ss+[hh:mm] '`

> **Note**
>
> Null: 定義されたデータがないtrueことを示すブール値。

Example

TIMESTAMP 値の例：

```
TIMESTAMP '2025-12-21 23:59:59.999Z'
TIMESTAMP '2025-12-21 23:59:59+23:59'
'2025-12-21 23:59:59'
'2025-12-21T23:59:59.123+11:11'
```

> **Note**
>
> 倍精度データは正確ではありません。一部の値は正確に変換されず、精度が制限されているため、すべての実数を表すわけではありません。クエリ内の浮動小数点データは、内部的に表される値と同じではない場合があります。入力された数の精度が高すぎると、値は四捨五入されます。

サポートされている句

SELECT ステートメントは、1 つ以上のビューからデータを取得するために使用されます。 は JOINおよび INNER JOINオペレーション AWS IoT SiteWise をサポートします。

ビューは、明示的なJOIN構文、または FROM句のカンマ区切り表記で結合されます。

Example

一般的なSELECTステートメント：

```
SELECT expression [, ...]
  [ FROM table_name AS alias [, ...] ]
  [ WHERE condition ]
  [ GROUP BY expression [, ...] ]
  [ HAVING condition ]
  [ ORDER BY expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] [, ...] ]
```

```
  [ LIMIT expression ]
```

Example

さまざまな句を含む SELECT ステートメント：

```
SELECT
  a.asset_name,
  a.asset_id,
  p.property_type,
  p.property_data_type,
  p.string_attribute_value,
  p.property_name
FROM asset a, asset_property p
WHERE a.asset_description LIKE '%description%'
AND p.property_type IN ('attribute', 'metric')
OR p.property_id IN (
  SELECT property_id
  FROM raw_time_series
  WHERE event_timestamp BETWEEN TIMESTAMP '2025-01-01 00:00:00' AND TIMESTAMP
 '2025-01-02 00:00:00'
  GROUP BY asset_id, property_id
  HAVING COUNT(*) > 100
 )
GROUP BY p.property_type
HAVING COUNT(*) > 5
ORDER BY a.asset_name ASC
LIMIT 20;
```

Note

暗黙的な JOINは、内部スキーマに基づく JOINキーワードを使用せずに、2 AWS IoT SiteWiseつ以上の異なるテーブルを組み合わせます。これは、メタデータasset_idテーブルと raw データテーブル間の フィールドと property_idフィールドJOINで を実行するのと同じです。このパターンにより、SiteWise は未加工のデータテーブルから取得するときにクエリ内の特定のメタデータフィルターを活用でき、スキャンされる全体的なデータが少なくなります。

Example クエリの：

```
SELECT a.asset_name, p.property_name, r.event_timestamp
FROM asset a, asset_property p, raw_time_series r
```

```
WHERE a.asset_name='my_asset' AND p.property_name='my_property'
```

上記の例では、指定されたメタデータ名に属するアセットプロパティのデータのみをスキャンします。

Example 上記のクエリと同等に最適化されていません。

```
SELECT a.asset_name, p.property_name, r.event_timestamp
FROM asset a
JOIN asset_property p ON a.asset_id=p.asset_id
JOIN raw_time_series r ON p.asset_id=r.asset_id AND p.property_id=r.property_id
WHERE a.asset_name='my_asset' AND p.property_name='my_property'
```

各句とその説明を以下に示します。

句	署名	説明
LIMIT	`LIMIT { count }`	この句は、結果セットを指定された行数に制限します。および ORDER BYOFFSETLIMIT句の有無にかかわらず、 を使用できます。 LIMIT は、[0,2147483647] の負以外の整数でのみ機能します。
ORDER BY	`ORDER BY expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ]`	ORDER BY 句は、クエリの結果セットをソートします。
GROUP BY	`GROUP BY expression [, ...]`	GROUP BY 句は、クエリのグループ化列を識別します。これは集計式と組み合わせて使用されます。

句	署名	説明
HAVING	`HAVING boolean-expression`	HAVING 句は、GROUP BY 句によって作成されたグループ行をフィルタリングします。
SUB SELECT	`SELECT column1, column2 FROM table1 WHERE column3 IN (SELECT column4 FROM table2);`	別のSELECTステートメントに埋め込まれたSELECTステートメント。
JOIN	`SELECT column1, column2 FROM table1 JOIN table2 ON table1.column1 = table2.column1;`	
INNER JOIN	`SELECT columns FROM table1 INNER JOIN table2 ON table1.column = table2.column;`	は、結合条件に一致する両方のテーブルのすべての行INNER JOINを返します。
UNION	`query { UNION [ ALL ] } another_query`	UNION 演算子は 2 つの引数の集合和を計算し、結果セットから重複するレコードを自動的に削除します。

論理演算子

AWS IoT SiteWise は、次の論理演算子をサポートしています。

演算子	署名	説明
AND	a AND b	両方の値が true の場合、TRUE

演算子	署名	説明
OR	a OR b	TRUE 1 つの値が true の場合
NOT	NOT 式	TRUE 式が false の場合、および式が true FALSEの場合
IN	x IN式	TRUE 式の if 値
BETWEEN	BETWEEN a AND b	TRUE 上限と下限の間の値の場合、 には両方の制限が含まれます。
LIKE	LIKE パターン	TRUE 値がパターンの場合 LIKE はワイルドカードをサポートしています。例については、以下を参照してください。 • % は文字列内の 1 つ以上の文字を置き換えます。 • _ は文字列内の 1 文字を置き換えます。 • ESCAPE は、LIKE パターンでエスケープ文字を指定する文字とともに使用されます。

すべての論理演算子の例：

関数	例
AND	`SELECT a.asset_name` `  FROM asset AS a, latest_value_time_series AS t`

関数	例
	`WHERE t.int_value > 30 AND t.event_timestamp > TIMESTAMP '2025-05-15 00:00:01'`
または	`SELECT a.asset_name FROM asset AS a WHERE a.asset_name like 'abc' OR a.asset_name like 'pqr'`
NOT	`SELECT ma.asset_id AS a_id FROM asset AS ma WHERE (ma.asset_id NOT LIKE 'some %patterna%' escape 'a') AND ma.asset_id='abc'`
IN	`SELECT a.asset_name FROM asset AS a WHERE a.asset_name IN ('abc', 'pqr')`
BETWEEN	`SELECT asset_id, int_value, event_timestamp AS i_v FROM raw_time_series WHERE event_timestamp BETWEEN TIMESTAMP '2025-04-15 00:00:01' and TIMESTAMP '2025-05-15 00:00:01'`

関数	例

LIKE	• % パターン：
	`SELECT POWER(rw.int_value, 5) AS raised_value FROM raw_time_series AS rw WHERE rw.asset_id LIKE 'some%pattern%' AND rw.int_value > 30`
	• _ パターン：
	`SELECT asset_id, property_id FROM asset_property WHERE string_attribute_value LIKE 'Floor_'`
	• ESCAPE パターン：
	`SELECT asset_id FROM asset WHERE asset_name LIKE 'MyAsset/_%' ESCAPE '/'`

比較演算子

AWS IoT SiteWise では、次の比較演算子がサポートされています。すべての比較オペレーションは、組み込みデータ型で使用でき、ブール値に評価されます。

論理演算子

演算子	説明
<	Less than (より小さい)
>	Greater than (より大きい)
<=	以下

演算子	説明
>=	以上
=	Equals (等しい)
! =	Not equal

数値以外の比較オペレーションのトゥルーステーブル

Type	タイプ >= x	タイプ <= x	タイプ > x	タイプ < x	タイプ = x	!= x と入力します。
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

一部の述語は演算子のように動作しますが、特別な構文があります。以下を参照してください。

Comparision 述語

演算子	説明
IS NULL	値が であるかどうかをテストしますNULL。
IS NOT NULL	値が でないかどうかをテストしますNULL。

NaN 演算子

NaN、または「数値ではない」は、浮動小数点演算の特殊な値です。以下は、NaN比較とその仕組みのリストです。

- NaN 値は一重引用符で囲む必要があります。たとえば、NaN「」です。
- NaN 値は互いに等しいと見なされます。
- NaN が他の数値より大きい。
- AVG()、、 などの集計関数STDDEV()でSUM()、いずれかの値が の場合NaN、結果は になりますNaN。
- MAX() や などの集計関数ではMIN()、NaN値は計算に含まれます。

NaN 値の比較

比較	結果
`'NaN' ≥ x`	真
`'NaN' ≤ x`	x が NaN に等しい場合は true、それ以外の場合は False
`'NaN' > x`	x が NaN に等しい場合は False、それ以外の場合は True
`'NaN' < x`	False
`'NaN' = x`	x が NaN に等しい場合は true、それ以外の場合は False
`'NaN' != x`	x が NaN に等しい場合は False、それ以外の場合は True

SQL 関数

サポートされている関数グループは次のとおりです。

トピック

- スカラー関数
- 集計関数

スカラー関数

スカラー関数は 1 つ以上の入力値を取得し、1 つの出力値を返します。これらは、データの操作と取得のために SQL (構造化クエリ言語) で広く使用され、データ処理タスクの効率を向上させます。

トピック

- Null データ関数
- 文字列関数

- 数学関数
- 日時関数
- 型変換関数

Null データ関数

NULL データ関数は、値がないことを表す NULL 値を処理または操作します。関数を使用すると、NULLs他の値に置き換えたり、値が NULL であるかどうかを確認したり、特定の方法で NULLs を処理するオペレーションを実行したりできます。

関数	署名	説明
COALESCE	COALESCE (expression1、expression2、...、expressionN)	すべての式が null と評価された場合、COALESCE は null を返します。式は同じタイプである必要があります。

Example COALESCE 関数の

```
SELECT COALESCE (l.double_value, 100) AS non_double_value FROM latest_value_time_series
 AS l LIMIT 1
```

文字列関数

文字列関数は、テキストデータの操作と処理に使用される組み込みツールです。これにより、連結、抽出、フォーマット、文字列内の検索などのタスクが可能になります。これらの関数は、データベース内のテキストベースのデータのクリーニング、変換、分析に不可欠です。

文字列関数

関数	署名	説明
LENGTH	長さ (文字列)	文字列の長さを返します。
CONCAT	CONCAT (文字列、文字列)	文字列内の引数を連結します。
SUBSTR	• SUBSTR (文字列、開始)	次のいずれかを返します。

関数	署名	説明
	• SUBSTR (文字列、開始、長さ) • SUBSTR (文字列、regexp)	• 指定された場所から始まり、オプションで指定された長さを持つ入力文字列の部分文字列を返します。 • 指定された正規表現に一致する入力文字列の最初の部分文字列を返します。 開始パラメータに 1 ベースのインデックス作成を使用します。
UPPER	UPPER (文字列)	入力文字列の文字を大文字に変換します。
LOWER	LOWER (文字列)	入力文字列の文字を小文字に変換します。
TRIM	TRIM (文字列)	文字列の先頭、末尾、または両側からスペース文字を削除します。
LTRIM	LTRIM (文字列)	文字列の先頭からスペース文字を削除します。
RTRIM	RTRIM (文字列)	文字列の末尾からスペース文字を削除します。
STR_REPLACE	STR_REPLACE (文字列、 から へ)	指定された部分文字列のすべての出現を、指定された別の部分文字列に置き換えます。

すべての 関数の例：

関数	例
LENGTH	`SELECT LENGTH(a.asset_id) AS asset_id_length FROM asset AS a`
CONCAT	`SELECT CONCAT(p.property_id, p.property_name) FROM asset_property AS p`
SUBSTR	• `SELECT SUBSTR(a.asset_name, 1, 3) AS substr-val FROM asset AS a` • `SELECT SUBSTR(p.property_name, 3) AS substr_val1 FROM asset_property AS p` • `SELECT SUBSTR(p.property_name, '@[^.]*') AS substr_val2 FROM asset_property AS p`
UPPER	`SELECT UPPER(d.string_value) AS up_string FROM raw_time_series AS d`
LOWER	`SELECT LOWER(d.string_value) AS low_string FROM raw_time_series AS d`
TRIM	`SELECT TRIM(d.string_value) AS tm_string FROM raw_time_series AS d`
LTRIM	`SELECT LTRIM(d.string_value) AS ltrim_string FROM raw_time_series AS d`
RTRIM	`SELECT RTRIM(d.string_value) AS rtrim_string FROM raw_time_series AS d`

関数	例
STR_REPLACE	`SELECT STR_REPLACE(d.string_value, 'abc', 'def') AS replaced_string FROM raw_time_series AS d`

連結演算子

連結演算子 `||`、またはパイプ演算子 は、2 つの文字列を結合します。`CONCAT` 関数の代わりに使用でき、複数の文字列を組み合わせると読みやすくなります。

Example 連結演算子の

```
SELECT a.asset_name || ' - ' || p.property_name
  AS full_name
  FROM asset a, asset_property p
```

数学関数

数学関数は、数値データの計算を実行するために SQL クエリで使用される事前定義された数学演算です。これらは、データベースからデータを抽出して個別に処理することなく、データを操作および変換する方法を提供します。

数学関数

関数	署名	説明
`POWER`	POWER (int|double、int|double)	2 番目の引数の累乗に上げられた最初の引数の値を返します。
`ROUND`	• ROUND (int|double、decimal_places_int) • ROUND (int|double)	最も近い整数に丸めます。
`FLOOR`	FLOOR (int|double)	指定された値以下の最大の整数を返します。

すべての 関数の例：

関数	例
POWER	• `POWER (3, 77)` • `POWER (2.3, 3.9)` • `POWER (1.0, 4.2)`
ROUND	`ROUND (32.12435, 3)`
FLOOR	`FLOOR (21.2)`

日時関数

日付時刻関数は、日付と時刻で機能します。これらの関数を使用すると、日付の特定コンポーネントを抽出し、計算を実行し、日付値を操作できます。

これらの関数で許可される識別子は次のとおりです。

- YEAR
- MONTH
- DAY
- HOUR
- MINUTE
- SECOND

関数	署名	説明
NOW	NOW ()	ミリ秒の精度で現在のタイムスタンプを返します。これは、クエリ内で実行された時点の正確な時間を提供します。

関数	署名	説明
DATE_ADD	DATE_ADD (識別子、間隔_期間、列)	日付/時刻と日数/時間、または日付/時刻と日付/時間間隔の合計を返します。
DATE_SUB	DATE_SUB (識別子、間隔_期間、列)	日付/時刻と日数/時間の差、または日付/時刻と日付/時間間隔の差を返します。
TIMESTAMP_ADD	TIMESTAMP_ADD (識別子、間隔_期間、列)	指定された時間単位の時間間隔を日時式に追加します。
TIMESTAMP_SUB	TIMESTAMP_SUB (識別子、間隔_期間、列)	指定された時間単位の時間間隔を日時式から減算します。
CAST	CAST (式 AS TIMESTAMP FORMAT パターン)	指定された形式パターンを使用して、文字列式をタイムスタンプに変換します。一般的なパターンには、標準の日時形式`'yyyy-MM-dd HH:mm:ss'` が含まれます。例: `SELECT CAST('2023-12-25 14:30:00' AS TIMESTAMP) AS converted_timestamp`

Example リストされた関数を使用した SQL クエリの :

```
SELECT r.asset_id, r.int_value,
  date_add(DAY, 7, r.event_timestamp) AS date_in_future,
  date_sub(YEAR, 2, r.event_timestamp) AS date_in_past,
  timestamp_add(DAY, 2, r.event_timestamp) AS timestamp_in_future,
  timestamp_sub(DAY, 2, r.event_timestamp) AS timestamp_in_past,
  now() AS time_now
FROM raw_time_series AS r
```

型変換関数

型変換関数は、値のデータ型をある値から別の値に変更するために使用されます。データの互換性を確保し、特定の形式のデータを必要とするオペレーションを実行するために不可欠です。

関数	署名	説明
TO_DATE	• TO_DATE (整数) • TO_DATE (式、形式)	• エポックミリ秒の整数を日付値に変換します。 • 文字列表現を日付値に変換します。
TO_TIMESTAMP	• TO_TIMESTAMP (ダブル) • TO_TIMESTAMP (文字列、形式)	• エポック秒整数をタイムスタンプデータ型に変換します。 • 日付と時刻の文字列表現をタイムスタンプデータ型に変換します。
TO_TIME	• TO_TIME (int) • TO_TIME (文字列、形式)	• エポックミリ秒の整数を時間値に変換します。 • 文字列表現を時間値に変換します。
CAST	CAST (<式> AS <データ型>)	エンティティ、または評価される式を 1 つの型から別の型に変換します。 サポートされているデータ型は次のとおりです。 • BOOLEAN • INTEGER • INT • TIMESTAMP • DATE • CHAR

関数	署名	説明
		• CHARACTER • STRING

Example リストされた関数を使用した SQL クエリの :

```
SELECT TO_TIMESTAMP (100) AS timestamp_value,
  TO_DATE(r.event_timestamp) AS date_value,
  TO_TIME(r.event_timestamp) AS time_value
FROM raw_time_series AS r
```

集計関数

集計関数は、複数のデータ行にわたって計算を実行して 1 つの要約結果を生成するデータベースオペレーションです。これらの関数は、データセットを分析して、合計、平均、カウント、その他の統計メジャーなどの計算された値を返します。

関数	署名	説明
AVG	AVG (式)	数値式の平均を返します。
COUNT	COUNT (式)	指定された条件に一致する行数を返します。
MAX	MAX (式)	選択した式の最大値を返します。
MIN	MIN (式)	選択した式の最小値を返します。
SUM	SUM (式)	数値式の合計を返します。
STDDEV	STDDEV (式)	サンプル標準偏差を返します。
GROUP BY	GROUP BY 式	グループ化列によって作成された行を返します。

関数	署名	説明
HAVING	HAVING ブール式	GROUP BY 句でフィルタリングされたグループ行を返します。

すべての関数の例：

関数	例
AVG	`SELECT d.asset_id, d.property_id, AVG(d.int_value) FROM raw_time_series AS d`
COUNT	`SELECT COUNT(d.int_value) FROM raw_time_series AS d`
MAX	`SELECT MAX(d.int_value) FROM raw_time_series AS d`
MIN	`SELECT MIN(d.int_value) FROM raw_time_series AS d`
SUM	`SELECT SUM(d.int_value) FROM raw_time_series AS d`
STDDEV	`SELECT STDDEV(d.int_value) FROM raw_time_series AS d`
• グループ化の条件 • HAVING	`SELECT MAX(d.int_value) AS max_int_value, d.asset_id FROM raw_time_series AS d GROUP BY d.asset_id`

関数	例
	`HAVING MAX(d.int_value) > 5`

クエリの例

メタデータフィルタリング

次の例は、 AWS IoT SiteWise クエリ言語の SELECTステートメントを使用してメタデータをフィルタリングする場合です。

```
SELECT a.asset_name, p.property_name
FROM asset a, asset_property p
WHERE a.asset_name LIKE 'Windmill%'
```

値のフィルタリング

AWS IoT SiteWise クエリ言語の SELECTステートメントを使用した値フィルタリングの例を次に示します。

```
SELECT a.asset_name, r.int_value
FROM asset a, raw_time_series r
WHERE r.int_value > 30
AND r.event_timestamp > TIMESTAMP '2022-01-05 12:15:00'
AND r.event_timestamp < TIMESTAMP '2022-01-05 12:20:00'
```

クエリの最適化

メタデータフィルター

メタデータまたは raw データをクエリするときは、 WHERE句を使用してメタデータフィールドでフィルタリングし、スキャンされるデータ量を減らします。メタデータスキャンを制限するには、次の演算子を使用します。

- 等しい (=)
- 等しくない (!=)
- LIKE

- IN
- AND
- または

属性プロパティの場合、次のフィールドを使用して結果をフィルタリングします。

- `double_attribute_value`
- `int_attribute_value`
- `boolean_attribute_value`
- `string_attribute_value`

これらのフィールドは、属性タイプのアセットプロパティの latest_value_time_series テーブルよりもパフォーマンスが向上します。

> **Note**
>
> 演算子の右側にあるリテラルを使用して、データスキャンを適切に制限します。たとえば、次のクエリは、厳密な文字列リテラルを使用するよりもパフォーマンスが悪くなります。
>
> ```
> SELECT property_id FROM asset_property WHERE property_name = CONCAT('my',
> 'property')
> ```

Example メタデータフィルターの場合：

```
SELECT p.property_name FROM asset_property p
WHERE p.property_type = 'attribute' AND p.string_attribute_value LIKE 'my-property-%'
```

Raw データフィルター

すべての raw データテーブル (raw_time_series、latest_value_time_series、precomputed_aggregates) には、行に関連付けられたタイムスタンプがあります。メタデータフィルターに加えて、 `event_timestamp`フィールドで`WHERE`句フィルターを使用して、スキャンされるデータの量を減らします。次のオペレーションを使用して、raw データスキャンを制限します。

- 等しい (=)
- Greater than (>)
- 未満 (<)
- 以上 (>=)
- 以下 (<=)
- BETWEEN
- AND

precomputed_aggregates テーブルをクエリするときは、常に `WHERE`句で品質フィルターを指定します。これにより、特に `BAD`または データを探している場合に、クエリがスキャンする`UNCERTAIN`データの量が減少します。precomputed_aggregates テーブルをクエリするときは、解像度フィルター (1m、15m、1h、または 1d) を使用することを強くお勧めします。解像度フィルターを指定しない場合、 AWS IoT SiteWise はデフォルトですべての解像度で完全なテーブルスキャンを行います。これは非効率です。

> **Note**
>
> 等しくなく(`!=`)、`OR`演算子は通常、raw データスキャンに意味のあるフィルターを適用しません。raw データ値 (string_value、double_value など) のフィルタも raw データスキャンを制限しません。

JOIN 最適化

AWS IoT SiteWise SQL は、2 つのテーブルをマージする `JOIN`キーワードをサポートします。フィールドで (`ON`キーワードを使用して) アクティブにフィルタリング`JOIN`する のみがサポートされます。フルデカルト結合は禁止されています。

AWS IoT SiteWise は`JOIN`、 `JOIN`キーワードを使用せずに暗黙的な もサポートしています。これらは、異なるメタデータテーブル間、およびメタデータテーブルと raw テーブル間で許可されます。例えば、次のクエリは、

```
SELECT a.asset_name, p.property_name FROM asset a, asset_property p
```

この同等のクエリよりもパフォーマンスが優れています。

```
SELECT a.asset_name, p.property_name FROM asset a
JOIN asset_property p ON a.asset_id = p.asset_id
```

次の暗黙的な結合が許可されます (O は許可され、X は禁止されます) 。

	アセット	asset_property	latest_value_time_series	raw_time_series	precomputed_aggregates	subquery
アセット	X	O	O	O	O	X
asset_property	O	X	O	O	O	X
latest_value_time_series	O	O	X	X	X	X
raw_time_series	O	O	X	X	X	X
precomputed_aggregates	O	O	X	X	X	X
subquery	X	X	X	X	X	X

可能な場合は、暗黙的な JOINを使用します。JOIN キーワードを使用する必要がある場合は、個々の JOINed テーブルにフィルターを適用して、スキャンされるデータを最小限に抑えます。たとえば、このクエリの代わりに、次のようにします。

```
SELECT level1.asset_id, level2.asset_id, level3.asset_id
FROM asset AS level1
JOIN asset AS level2 ON level2.parent_asset_id = level1.asset_id
JOIN asset AS level3 ON level3.parent_asset_id = level2.asset_id
WHERE level1.asset_name LIKE 'level1%'
AND level2.asset_name LIKE 'level2%'
AND level3.asset_name LIKE 'level3%'
```

より効率的なクエリを使用します。

```
SELECT level1.asset_id, level2.asset_id, level3.asset_id
FROM asset AS level1
JOIN (SELECT asset_id, parent_asset_id FROM asset WHERE asset_name LIKE 'level2%') AS
 level2 ON level2.parent_asset_id = level1.asset_id
JOIN (SELECT asset_id, parent_asset_id FROM asset WHERE asset_name LIKE 'level3%') AS
 level3 ON level3.parent_asset_id = level2.asset_id
WHERE level1.asset_name LIKE 'level1%'
```

メタデータフィルターをサブクエリにプッシュすることで、スキャンプロセス中に 内の個々のテーブルがフィルタリングされるようにJOINします。同じ効果を得るために、サブクエリで LIMITキーワードを使用することもできます。

大きなクエリ

デフォルトよりも多くの行を生成するクエリの場合は、ExecuteQuery API のページサイズを最大値の 20,000 に設定します。これにより、全体的なクエリパフォーマンスが向上します。

一部のクエリでスキャンされるデータの量を減らすには、 LIMIT句を使用します。集計関数と特定のテーブル全体の句 (GROUP BY、ORDER BY、JOIN) では、LIMIT句を適用する前にフルスキャンを完了する必要があります。

> **Note**
>
> AWS IoT SiteWise は、 LIMIT句が適用されていても、特に複数のプロパティをスキャンする raw データクエリの場合、最小量のデータをスキャンすることがあります。

ODBC

のオープンソース ODBC ドライバー AWS IoT SiteWise は、開発者 AWS IoT SiteWise 向けに への SQL リレーショナルインターフェイスを提供し、Power BI Desktop や Microsoft Excel などのビジネスインテリジェンス (BI) ツールからの接続を可能にします。ODBC ドライバーは現在 Windows AWS IoT SiteWise で使用でき、Okta および Microsoft Azure Active Directory (AD) との SSO をサポートしています。

詳細については、AWS IoT SiteWise GitHub の「ODBC ドライバードキュメント」を参照してください。

トピック

- ODBC ドライバーの接続文字列の構文とオプション
- ODBC AWS IoT SiteWise ドライバーの接続文字列の例
- ODBC ドライバーとの接続のトラブルシューティング

ODBC ドライバーの接続文字列の構文とオプション

ODBC ドライバーの接続文字列オプションを指定するための構文は次のとおりです。

```
Driver={AWS IoT SiteWise ODBC Driver};(option)=(value);
```

使用できるオプションは以下のとおりです。

ドライバー接続オプション

- **Driver** （必須）– ODBC で使用されているドライバー。

  デフォルトは AWS IoT SiteWiseです。
- **DSN** – 接続の設定に使用するデータソース名 (DSN)。

  デフォルトは `NONE` です。
- **Auth** – 認証モード。これは、次のいずれかである必要があります。
  - `AWS_PROFILE` – デフォルトの認証情報チェーンを使用します。
  - `IAM` – IAM AWS 認証情報を使用します。
  - `AAD` – Azure Active Directory (AD) ID プロバイダーを使用します。
  - `OKTA` – Okta ID プロバイダーを使用します。

  デフォルトは `AWS_PROFILE` です。

エンドポイント設定オプション

- **EndpointOverride** – AWS IoT SiteWise サービスのエンドポイントオーバーライド。これは、リージョンを上書きする高度なオプションです。例:

  ```
  iotsitewise.us-east-1.amazonaws.com
  ```

- **Region** – AWS IoT SiteWise サービスエンドポイントの署名リージョン。

  デフォルトは `us-east-1` です。

認証情報プロバイダーオプション

- **ProfileName** – AWS 設定ファイルのプロファイル名。

  デフォルトは `NONE` です。

AWS IAM 認証オプション

- **UID** または **AccessKeyId** – AWS ユーザーアクセスキー ID。接続文字列で `UID`と の両方`AccessKeyId`が指定されている場合、`UID`値は空でない限り使用されます。

  デフォルトは `NONE` です。
- **PWD** または **SecretKey** – AWS ユーザーシークレットアクセスキー。接続文字列で `PWD`と の両方`SecretKey`が指定されている場合、空でない限り、 `PWD`の値が使用されます。

  デフォルトは `NONE` です。
- **SessionToken** – 多要素認証 (MFA) が有効になっているデータベースにアクセスするために必要な一時的なセッショントークン。 `=` 入力に末尾を含めないでください。

  デフォルトは `NONE` です。

Okta の SAML ベースの認証オプション

- **IdPHost** – 指定された IdP のホスト名。

  デフォルトは `NONE` です。
- **UID** または **IdPUserName** – 指定された IdP アカウントのユーザー名。接続文字列で `UID`と の両方`IdPUserName`が指定されている場合、`UID`値は空でない限り使用されます。

  デフォルトは `NONE` です。
- **PWD** または **IdPPassword** – 指定された IdP アカウントのパスワード。接続文字列で `PWD`と の両方`IdPPassword`が指定されている場合、`PWD`値は空でない限り使用されます。

  デフォルトは `NONE` です。

- **OktaApplicationID** – AWS IoT SiteWise アプリケーションに関連付けられた一意の Okta 提供 ID。アプリケーション ID (AppId) を検索する場所は、アプリケーションメタデータで指定された entityIDフィールドにあります。以下に例を示します。

  ```
  entityID="http://www.okta.com//(IdPAppID)
  ```

  デフォルトは NONE です。
- **RoleARN** – 発信者が引き受けるロールの Amazon リソースネーム (ARN)。

  デフォルトは NONE です。
- **IdPARN** – IdP を記述する IAM の SAML プロバイダーの Amazon リソースネーム (ARN)。

  デフォルトは NONE です。

Azure Active Directory の SAML ベースの認証オプション

- **UID** または **IdPUserName** – 指定された IdP アカウントのユーザー名。

  デフォルトは NONE です。
- **PWD** または **IdPPassword** – 指定された IdP アカウントのパスワード。

  デフォルトは NONE です。
- **AADApplicationID** – Azure AD に登録されたアプリケーションの一意の ID。

  デフォルトは NONE です。
- **AADClientSecret** – トークンの取得を承認するために使用される Azure AD の登録済みアプリケーションに関連付けられたクライアントシークレット。

  デフォルトは NONE です。
- **AADTenant** – Azure AD テナント ID。

  デフォルトは NONE です。
- **RoleARN** – 発信者が引き受けるロールの Amazon リソースネーム (ARN)。

  デフォルトは NONE です。
- **IdPARN** – IdP を記述する IAM の SAML プロバイダーの Amazon リソースネーム (ARN)。

  デフォルトは NONE です。

AWS SDK (アドバンスド) オプション

- **RequestTimeout** – AWS SDK がタイムアウトする前にクエリリクエストを待機するミリ秒単位の時間。正以外の値を指定すると、リクエストのタイムアウトは無効になります。

  デフォルトは 3000 です。

- **ConnectionTimeout** – AWS SDK がタイムアウトする前にオープン接続経由でデータを転送するのを待機するミリ秒単位の時間。値が 0 の場合、接続タイムアウトは無効になります。この値は負にすることはできません。

  デフォルトは 1000 です。

- **MaxRetryCountClient** – SDK の 5xx エラーコードによる再試行可能なエラーの最大再試行回数。値は負にすることはできません。

  デフォルトは 0 です。

- **MaxConnections** – AWS IoT SiteWise サービスへの同時オープン可能な HTTP 接続の最大数。値は正である必要があります。

  デフォルトは 25 です。

ODBC ドライバーのログ記録オプション

- **LogLevel** – ドライバーのログ記録のログレベル。次のいずれかにする必要があります。
  - 0 (オフ) 。
  - 1 (ERROR)。
  - 2 (警告) 。
  - 3 (INFO)。
  - 4 (デバッグ) 。

  デフォルトは 1 (ERROR) です。

  警告： DEBUG ログ記録モードを使用すると、ドライバーが個人情報を記録する可能性があります。

- **LogOutput** – ログファイルを保存するフォルダ。

  デフォルトは次のとおりです。

  - Windows: %USERPROFILE%、または利用できない場合は %HOMEDRIVE%%HOMEPATH%。

- macOS と Linux: `$HOME`、または使用できない場合は、関数`pw_dir`の フィールドから値`getpwuid(getuid())`を返します。

SDK ログ記録オプション

AWS SDK ログレベルは ODBC AWS IoT SiteWise ドライバーログレベルとは別です。一方を設定しても、他方には影響しません。

SDK ログレベルは、環境変数 を使用して設定されます`SW_AWS_LOG_LEVEL`。次の値を指定できます:

- `OFF`
- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`
- `FATAL`

が設定され`SW_AWS_LOG_LEVEL`ていない場合、SDK ログレベルはデフォルト に設定されます`WARN`。

プロキシを介した接続

ODBC ドライバーは、プロキシ AWS IoT SiteWise を介した への接続をサポートします。この機能を使用するには、プロキシ設定に基づいて次の環境変数を設定します。

- **`SW_PROXY_HOST`** – プロキシホスト。
- **`SW_PROXY_PORT`** – プロキシポート番号。
- **`SW_PROXY_SCHEME`** – プロキシスキーム、 `http`または `https`。
- **`SW_PROXY_USER`** – プロキシ認証のユーザー名。
- **`SW_PROXY_PASSWORD`** – プロキシ認証のユーザーパスワード。
- **`SW_PROXY_SSL_CERT_PATH`** – HTTPS プロキシへの接続に使用する SSL 証明書ファイル。
- **`SW_PROXY_SSL_CERT_TYPE`** – プロキシクライアント SSL 証明書のタイプ。
- **`SW_PROXY_SSL_KEY_PATH`** – HTTPS プロキシへの接続に使用するプライベートキーファイル。

- **SW_PROXY_SSL_KEY_TYPE** – HTTPS プロキシへの接続に使用されるプライベートキーファイルのタイプ。
- **SW_PROXY_SSL_KEY_PASSWORD** – HTTPS プロキシへの接続に使用されるプライベートキーファイルへのパスフレーズ。

ODBC AWS IoT SiteWise ドライバーの接続文字列の例

IAM 認証情報を使用して ODBC ドライバーに接続する例

```
Driver={AWS IoT SiteWise ODBC Driver};Auth=IAM;AccessKeyId=(your access key
 ID);SecretKey=(your secret key);SessionToken=(your session token);Region=us-east-1;
```

プロファイルを使用して ODBC ドライバーに接続する例

```
Driver={AWS IoT SiteWise ODBC Driver};ProfileName=(the profile name);region=us-east-1;
```

ドライバーは、 で指定された認証情報を使用するか`~/.aws/credentials`、環境変数 でファイルが指定されている場合は`AWS_SHARED_CREDENTIALS_FILE`、そのファイルの認証情報を使用して接続を試みます。

Okta を使用して ODBC ドライバーに接続する例

```
Driver={AWS IoT SiteWise ODBC Driver};Auth=OKTA;region=us-east-1;idPHost=(your host at
 Okta);idPUsername=(your user name);idPPassword=(your password);OktaApplicationID=(your
 Okta AppId);roleARN=(your role ARN);idPARN=(your Idp ARN);
```

Azure Active Directory (AAD) を使用して ODBC ドライバーに接続する例

```
Driver={AWS IoT SiteWise ODBC Driver};Auth=AAD;region=us-east-1;idPUsername=(your
 user name);idPPassword=(your password);aadApplicationID=(your AAD
 AppId);aadClientSecret=(your AAD client secret);aadTenant=(your AAD
 tenant);roleARN=(your role ARN);idPARN=(your idP ARN);
```

指定されたエンドポイントとログレベル 2 で ODBC ドライバーに接続する例 (警告)

```
Driver={AWS IoT SiteWise ODBC Driver};Auth=IAM;AccessKeyId=(your access
 key ID);SecretKey=(your secret key);EndpointOverride=iotsitewise.us-
east-1.amazonaws.com;Region=us-east-1;LogLevel=2;
```

ODBC ドライバーとの接続のトラブルシューティング

> **Note**
>
> DSN でユーザー名とパスワードが既に指定されている場合は、ODBC ドライバーマネージャーが要求したときに再度指定しないでください。

メッセージ`01S02`を含む のエラーコードは、接続文字列オプションが接続文字列で複数回渡されたときに`Re-writing (connection string option) (have you specified it several times?)`発生します。オプションを複数回指定すると、エラーが発生します。DSN と接続文字列との接続を行う場合、DSN で接続オプションが既に指定されている場合は、接続文字列で再度指定しないでください。

他の AWS サービスとやり取りする

AWS IoT SiteWise は AWS IoT MQTT パブリッシュ/サブスクライブメッセージブローカーにアセットデータを発行できるため、他の AWS サービスのアセットデータを操作できます。 は、 AWS IoT コアルールを使用してアセットデータを他の AWS のサービスにルーティングするために使用できる一意の MQTT トピックを各アセットプロパティに AWS IoT SiteWise 割り当てます。たとえば、以下のタスクを実行するように AWS IoT Core ルールを設定できます。

- 機器の故障を特定し、[AWS IoT Events] にデータを送信して適切な担当者に通知します。
- データを Amazon DynamoDB に送信することにより、外部ソフトウェアソリューションで使用する選択したアセットデータを履歴化します。
- [AWS Lambda] 関数をトリガーして週次レポートを生成します。

DynamoDB にプロパティ値を保存するルールを設定するのに必要なステップを説明するチュートリアルに従います。詳細については、「プロパティ値の更新を Amazon DynamoDB に発行する」を参照してください。

ルールの設定方法の詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の [Rules] (ルール) を参照してください。

他の AWS サービスのデータを に戻すこともできます AWS IoT SiteWise。 AWS IoT SiteWise ルールアクションを通じてデータを取り込むには、「」を参照してくださいAWS IoT Core ルール AWS IoT SiteWise を使用して にデータを取り込む。

トピック

- MQTT トピックのアセットプロパティについて
- でアセットプロパティ通知を有効にする AWS IoT SiteWise
- でアセットプロパティ通知をクエリする AWS IoT SiteWise
- アセットプロパティ通知とともにデータを Amazon S3 にエクスポートする
- Grafana AWS IoT SiteWise との統合
- AWS IoT SiteWise と の統合 AWS IoT TwinMaker
- Lookout for Equipment で異常を検出する

MQTT トピックのアセットプロパティについて

すべてのアセットプロパティには、次の形式で一意の MQTT トピックパスがあります。

```
$aws/sitewise/asset-models/assetModelId/assets/assetId/properties/propertyId
```

> **Note**
>
> AWS IoT SiteWise は、 AWS IoT Core ルールエンジンの # (複数レベル) トピックフィルターワイルドカードをサポートしていません。+ (単数レベル) ワイルドカードを使用できます。たとえば、次のトピックフィルターを使用して、特定のアセットモデルのすべての更新を照合できます。
>
> ```
> $aws/sitewise/asset-models/assetModelId/assets/+/properties/+
> ```
>
> トピックフィルターのワイルドカードの詳細については、[AWS IoT Core Developer Guide] (Core デベロッパーガイド) の [Topics] (トピック) を参照してください。

でアセットプロパティ通知を有効にする AWS IoT SiteWise

プロパティ通知を有効にしてアセットデータの更新を に発行し AWS IoT Core、データに対してクエリを実行できます。アセットプロパティ通知を使用すると、 は Amazon S3 への AWS IoT SiteWise データのエクスポートに使用できる AWS CloudFormation テンプレート AWS IoT SiteWise を提供します。

> **Note**
>
> アセットデータは、値が変更されたかどうかにかかわらず AWS IoT SiteWise、 によって受信される AWS IoT Core たびに に送信されます。

トピック

- アセットプロパティ通知を有効化 (コンソール)
- アセットプロパティ通知を有効化 (AWS CLI)

アセットプロパティ通知を有効化 (コンソール)

デフォルトでは、 AWS IoT SiteWise はプロパティ値の更新を発行しません。 AWS IoT SiteWise コンソールを使用して、アセットプロパティの通知を有効にできます。

アセットプロパティの通知を有効または無効にするには (コンソール)

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [アセット] を選択します。
3. アセットを選択して、プロパティの通知を有効にします。

> **Tip**
> 矢印アイコンを選択して、アセット階層を展開してアセットを検索できます。

4. [編集] を選択します。
5. アセットプロパティの [通知ステータス] で、[有効] を選択します。

また、[無効] を選択して、アセットプロパティの通知を無効にすることもできます。

6. [保存] を選択します。

アセットプロパティ通知を有効化 (AWS CLI)

デフォルトでは、 AWS IoT SiteWise はプロパティ値の更新を発行しません。 AWS Command Line Interface （ AWS CLI) を使用して、アセットプロパティの通知を有効または無効にできます。

この手順を完了するには、アセットの `assetId` とプロパティの `propertyId` を知っている必要があります。外部 ID を使用することもできます。アセットを作成したが、その `assetId` がわからない場合は、ListAssets API を使用して、特定のモデルのすべてのアセットを一覧表示します。DescribeAsset オペレーションを使用して、プロパティ ID を含むアセットのプロパティを表示します。

UpdateAssetProperty オペレーションを使用して、アセットプロパティの通知を有効または無効にします。以下のパラメータを指定します。

- `assetId` - アセットの ID。
- `propertyId` - アセットプロパティの ID。
- `propertyNotificationState` - プロパティ値の通知状態: `ENABLED` または `DISABLED`。
- `propertyAlias` - プロパティのエイリアス。通知状態を更新するときに、プロパティの既存のエイリアスを指定します。このパラメータを省略すると、プロパティの既存のエイリアスは削除されます。

アセットプロパティの通知を有効または無効にするには (CLI)

1. 次のコマンドを実行して、アセットプロパティのエイリアスを取得します。*asset-id* をアセットの ID に置き換え、*property-id* をプロパティの ID に置き換えます。

```
aws iotsitewise describe-asset-property \
  --asset-id asset-id \
  --property-id property-id
```

このオペレーションは、アセットプロパティの詳細を含むレスポンスを次の形式で返します。プロパティエイリアスは JSON オブジェクト `assetProperty.alias` にあります。

```
{
  "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
  "assetName": "Wind Turbine 7",
  "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
  "assetProperty": {
    "id": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "name": "Wind Speed",
    "alias": "/company/windfarm/3/turbine/7/windspeed",
    "notification": {
      "topic": "$aws/sitewise/asset-models/a1b2c3d4-5678-90ab-cdef-11111EXAMPLE/
assets/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/properties/a1b2c3d4-5678-90ab-
cdef-33333EXAMPLE",
      "state": "DISABLED"
    },
    "dataType": "DOUBLE",
    "unit": "m/s",
    "type": {
```

```
      "measurement": {}
    }
  }
}
```

2. アセットプロパティの通知を有効にするには、次のコマンドを実行します。*property-alias* は、前のコマンドのレスポンスのプロパティエイリアスに置き換えるか、`--property-alias` を省略して、エイリアスなしでプロパティを更新します。

```
aws iotsitewise update-asset-property \
  --asset-id asset-id \
  --property-id property-id \
  --property-notification-state ENABLED \
  --property-alias property-alias
```

また、`--property-notification-state DISABLED` を渡して、アセットプロパティの通知を無効にすることもできます。

でアセットプロパティ通知をクエリする AWS IoT SiteWise

アセットプロパティ通知をクエリするには、SQL ステートメントで構成される AWS IoT Core ルールを作成します。

AWS IoT SiteWise は、アセットプロパティデータの更新を次の形式で AWS IoT Core に発行します。

```
{
  "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 リスト内の各構造は、タイムスタンプ品質値 (TQV) 構造です。

- timestamp には、ナノ秒のオフセットがある現在の UNIX エポック時刻 (秒単位) が含まれます。
- quality には、データポイントの品質を示す、以下のいずれかの文字列が含まれます。
  - GOOD - データはいずれの問題の影響も受けません。
  - BAD - データはセンサーの障害などの問題による影響を受けます。
  - UNCERTAIN - データはセンサーの不正確さなどの問題による影響を受けます。
- value には、プロパティのタイプに応じて、以下のいずれかのフィールドが含まれます。
  - booleanValue
  - doubleValue
  - integerValue
  - stringValue
  - nullValue

nullValue – 値が Null で品質が BADまたは のプロパティ値のタイプを示す次のフィールドを持つ構造UNCERTAIN。

- valueType – {"B"、"D"、"S"、"I"} の列挙型

values 配列から値を解析するには、ルールの SQL ステートメントで複雑にネストされたオブジェクトクエリを使用する必要があります。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Nested object queries] (オブジェクトクエリのネスト) を参照するか、アセットプロパティ通知メッセージの解析の具体例については、プロパティ値の更新を Amazon DynamoDB に発行する チュートリアルを参照してください。

Example 値の配列を抽出するクエリの例

次のステートメントは、そのプロパティを持つすべてのアセットの特定の Double 型プロパティについて、更新されたプロパティ値の配列をクエリする方法を示しています。

```
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'
```

前のルールクエリステートメントは、次の形式でデータを出力します。

```
{
  "windspeed": [
    26.32020195042838,
    26.282584572975477,
    26.352566977372508,
    26.283084346171442,
    26.571883739599322,
    26.60684140743005,
    26.628738636715045,
    26.273486932802125,
    26.436379105473964,
    26.600590095377303
  ]
}
```

Example 単一の値を抽出するクエリの例

次のステートメントは、そのプロパティを持つすべてのアセットの特定の Double 型プロパティについて、プロパティ値の配列から最初の値をクエリする方法を示しています。

```
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'
```

前のルールクエリステートメントは、次の形式でデータを出力します。

```
{
  "windspeed": 26.32020195042838
}
```

> **Important**
>
> このルールクエリステートメントは、各バッチの最初の値以外の値の更新を無視します。各バッチには最大 10 個の値を含めることができます。残りの値を含める必要がある場合は、アセットプロパティ値を他のサービスに出力するより複雑なソリューションを設定する必要があります。例えば、配列内の各値を別のトピックに再発行する AWS Lambda アクションでルールを設定し、そのトピックをクエリして各値を目的のルールアクションに発行する別のルールを設定できます。

アセットプロパティ通知とともにデータを Amazon S3 にエクスポートする

受信データを からアカウントの AWS IoT SiteWise Amazon S3 バケットにエクスポートできます。履歴レポートを作成したり、複雑な方法でデータを分析したりできる形式でデータをバックアップできます。

から時系列データをエクスポートするには AWS IoT SiteWise、コールド階層機能を有効にして、データを Amazon S3 バケットに保存します。詳細については、「 でデータストレージを管理する AWS IoT SiteWise」を参照してください。

アセットモデルとアセットメタデータをエクスポートするには AWS IoT SiteWise、一括オペレーション機能を使用してメタデータを Amazon S3 バケットにエクスポートします。詳細については、「アセットとモデルを使用した一括オペレーション」を参照してください。

Grafana AWS IoT SiteWise との統合

Grafana は、ダッシュボード内のデータを視覚化およびモニタリングするために使用されるデータ視覚化プラットフォームです。Grafana バージョン 10.4.0 以降では、 AWS IoT SiteWise プラグイン

を使用して Grafana ダッシュボードで AWS IoT SiteWise アセットデータを視覚化します。ユーザーは、単一の Grafana ダッシュボードを使用して AWS IoT SiteWise、複数の AWS ソース (Amazon Timestream、Amazon CloudWatch など) やその他のデータソースからのデータを視覚化できます。

AWS IoT SiteWise プラグインを使用するには、次の 2 つのオプションがあります。

- [Local Grafana servers] (ローカル Grafana サーバー)

  AWS IoT SiteWise プラグインは、管理する Grafana サーバーに設定できます。プラグインの追加方法と使用方法の詳細については、GitHub サイトの [AWS IoT SiteWise Datasource README] (データソース README) ファイルを参照してください。

- AWS Grafana 用 Managed Service

  Managed Service for AWS Grafana (AMG) で AWS IoT SiteWise プラグインを使用できます。AMG が Grafana サーバーを管理するため、お客様はハードウェアやその他の Grafana インフラストラクチャを構築、パッケージ化、デプロイすることなく、データを視覚化することができます。詳細については、[AWS Managed Service for Grafana] (Managed Service for Grafana ユーザーガイド) の次のトピックを参照してください。

  - [What is Amazon Managed Service for Grafana (AMG) ?] (Amazon Managed Service for Grafana (AMG) とは？)
  - AWS IoT SiteWise データソースの使用

Example Grafana ダッシュボードの例

次の Grafana ダッシュボードは、[demo wind farm] (デモの風力発電施設) を視覚したものです。このデモダッシュボードには、[Grafana Play] のウェブサイトからアクセスできます。

AWS IoT SiteWise と の統合 AWS IoT TwinMaker

と を統合すると、 AWS IoT SiteWise コンソールで AWS IoT SiteWise のデータ取得 `ExecuteQuery` API や高度なアセット検索など AWS IoT SiteWise、 の堅牢な機能 AWS IoT TwinMaker にアクセスできます。サービスを統合してこれらの機能を使用するには、まず、統合を有効にする必要があります。

トピック

- 統合の有効化
- AWS IoT SiteWise と の統合 AWS IoT TwinMaker

統合の有効化

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。 AWS IoT SiteWise サポートされているアクションの詳細については、「サービス認可リファレンス」の「 で定義されるアクション AWS IoT SiteWise」を参照してください。

AWS IoT TwinMaker サービスにリンクされたロールの詳細については、「 AWS IoT TwinMaker ユーザーガイド」の「 のサービスにリンクされたロール AWS IoT TwinMaker」を参照してください。

AWS IoT SiteWise と を統合する前に AWS IoT TwinMaker、 が AWS IoT TwinMaker リンクされたワークスペースと AWS IoT SiteWise 統合できるようにする以下のアクセス許可を付与する必要があります。

- `iotsitewise:EnableSiteWiseIntegration` – リンクされた AWS IoT TwinMaker ワークスペースとの統合 AWS IoT SiteWise を許可します。この統合により AWS IoT TwinMaker 、 は AWS IoT TwinMaker サービスにリンクされたロール AWS IoT SiteWise を介して のすべてのモデリング情報を読み取ることができます。このアクセス許可を有効にするには、以下のポリシーをユーザーの IAM ロールに追加します。

  JSON

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

AWS IoT SiteWise と の統合 AWS IoT TwinMaker

AWS IoT SiteWise と を統合するには AWS IoT TwinMaker、以下が必要です。

- AWS IoT SiteWise アカウントで設定されたサービスにリンクされたロール
- AWS IoT TwinMaker アカウントで設定されたサービスにリンクされたロール
- AWS IoT TwinMaker リージョン`IoTSiteWiseDefaultWorkspace`のアカウントの ID を持つワークスペース。

AWS IoT SiteWise コンソールを使用して統合するには

コンソールに [AWS IoT TwinMakerとの統合] バナーが表示されたら、[許可を付与] を選択します。前提条件はアカウントで作成されます。

を使用して統合するには AWS CLI

AWS IoT TwinMaker を使用して AWS IoT SiteWise と を統合するには AWS CLI、次のコマンドを入力します。

1. `iotsitewise.amazonaws.com` の `AWSServiceName` を指定して `CreateServiceLinkedRole` を呼び出します。

```
aws iam create-service-linked-role --aws-service-name iotsitewise.amazonaws.com
```

2. `iottwinmaker.amazonaws.com` の `AWSServiceName` を指定して `CreateServiceLinkedRole` を呼び出します。

```
aws iam create-service-linked-role --aws-service-name iottwinmaker.amazonaws.com
```

3. `IoTSiteWiseDefaultWorkspace` の ID を指定して `CreateWorkspace` を呼び出します。

```
 aws iottwinmaker create-workspace --workspace-id IoTSiteWiseDefaultWorkspace
```

Lookout for Equipment で異常を検出する

> **Note**
> 異常検出は、Amazon Lookout for Equipment が利用可能なリージョンでのみ使用できます。

Amazon Lookout for Equipment AWS IoT SiteWise と統合すると、産業機器の異常検出と予測メンテナンスを通じて、産業機器に関するインサイトを得ることができます。Lookout for Equipment は、産業用機器を監視するための機械学習 (ML) サービスであり、異常な機器動作を検出し、潜在的な障害を特定します。Lookout for Equipment を使用すると、予知保全プログラムを実装し、最適ではない機器プロセスを特定できます。Lookout for Equipment の詳細については、「Amazon Lookout for Equipment User Guide」の「What is Amazon Lookout for Equipment?」を参照してください。

ML モデルをトレーニングして異常な機器の動作を検出する予測を作成すると、 はアセットプロパティ値を Lookout for Equipment AWS IoT SiteWise に送信し、ML モデルをトレーニングして異常な機器の動作を検出します。アセットモデルで予測定義を定義するには、Lookout for Equipment がデータにアクセスするために必要な IAM ロールと、Lookout for Equipment に送信して処理済みデータを Amazon S3 に送信するプロパティを指定します。詳細については、「でアセットモデルを作成する AWS IoT SiteWise」を参照してください。

AWS IoT SiteWise と Lookout for Equipment を統合するには、以下の大まかなステップを実行します。

- 追跡するプロパティの概要を示す予測定義をアセットモデルに追加します。予測定義は、そのアセットモデルに基づくアセットに予測を作成するために使用される測定値、変換、メトリクスの再利用可能なコレクションです。
- 指定した履歴データに基づいて予測をトレーニングします。
- スケジュール推論。特定の予測を実行する AWS IoT SiteWise 頻度を示します。

推論がスケジュールされると、Lookout for Equipment モデルは機器から受信したデータをモニタリングし、機器の動作の異常を探します。SiteWise Monitor で結果を表示および分析するには、 AWS IoT SiteWise GET API オペレーション、または Lookout for Equipment コンソールを使用します。また、アセットモデルからアラームディテクターを使用してアラームを作成し、異常な機器の動作を警告することもできます。

トピック

- 予測定義を追加する (コンソール)
- 予測をトレーニング (コンソール)
- 予測の推論を開始または停止する (コンソール)
- 予測定義を追加 (CLI)
- 予測をトレーニングして、推論を開始する (CLI)
- 予測をトレーニングする (CLI)
- 予測の推論を開始または停止する (CLI)

予測定義を追加する (コンソール)

によって収集されたデータを AWS IoT SiteWise Lookout for Equipment に送信するには、アセットモデルに AWS IoT SiteWise 予測定義を追加する必要があります。

AWS IoT SiteWise アセットモデルに予測定義を追加するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[モデル] を選択し、予測定義を追加するアセットモデルを選択します。
3. [予測] を選択します。
4. [予測定義を追加] を選択します。
5. 予測定義の詳細を定義します。

   a. 予測定義の一意の [名前] と [説明] を入力します。予測定義を作成した後は名前を変更できないため、慎重に名前を選択します。

   b. AWS IoT SiteWise がアセットデータを Amazon Lookout for Equipment と共有できるようにする [IAM アクセス許可ロール] を作成または選択します。ロールには、以下の IAM ポリシーと信頼ポリシーが必要です。ロールを作成する手順については、「カスタム信頼ポリシーを使用したロールの作成 (コンソール)」を参照してください。

      IAM ポリシー

      JSON

```
{
    "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:us-east-1:Account_ID:inference-
scheduler/IoTSiteWise_*",
                "arn:aws:lookoutequipment:us-east-1:Account_ID:model/
IoTSiteWise_*",
                "arn:aws:lookoutequipment:us-east-1: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::111122223333:role/Role_name"
        }
    ]
}
```

信頼ポリシー

JSON

```
{
    "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:us-
east-1: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:us-
east-1:Account_ID:*"
                }
            }
        }
    ]
}
```

c. [次へ] を選択します。

6. Lookout for Equipment に送信するデータ属性 (測定値、変換、メトリクス) を選択します。

   a. (オプション) 測定値を選択します。
   b. (オプション) 変換を選択します。
   c. (オプション) メトリクスを選択します。
   d. [次へ] を選択します。

7. 選択を確認します。アセットモデルに予測定義を追加するには、概要ページで [予測定義を追加] を選択します。

アクティブな予測がアタッチされている既存の予測定義を [編集] または [削除] することもできます。

予測をトレーニング (コンソール)

アセットモデルに予測定義を追加したら、アセットにある予測をトレーニングできます。

で予測をトレーニングするには AWS IoT SiteWise

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[アセット] を選択し、モニタリングするアセットを選択します。
3. [予測] を選択します。
4. トレーニングする予測を選択します。
5. [アクション] で、[トレーニングを開始] を選択し、以下を実行します。

a. 予測の詳細で、 がアセットデータを Lookout for Equipment と共有 AWS IoT SiteWise できるようにする IAM アクセス許可ロールを選択します。新しいロールを作成する必要がある場合は、[新しいロールを作成] を選択します。

b. [トレーニングデータの設定] では、[トレーニングデータの時間範囲] を入力して、予測のトレーニングに使用するデータを選択します。

c. (オプション) 後処理の後、データのサンプリングレートを選択します。

d. (オプション) [データラベル] では、ラベル付きデータを保持する Amazon S3 バケットとプレフィックスを指定します。データのラベル付けの詳細については、「Amazon Lookout for Equipment User Guide」の「Labeling your data」を参照してください。

e. [次へ] を選択します。

6. (オプション) トレーニングの完了後すぐに予測をアクティブにする場合は、[詳細設定] で、[トレーニング後に予測を自動的にアクティブ化する] を選択して、以下を実行します。

   a. [入力データ] で、[データアップロードの頻度] として、データのアップロード頻度を定義し、[オフセット遅延時間] として、使用するバッファの量を定義します。

   b. [次へ] を選択します。

7. 予測の詳細を確認し、[保存して開始] を選択します。

予測の推論を開始または停止する (コンソール)

> **Note**
>
> Lookout for Equipment の料金は、 AWS IoT SiteWise と Lookout for Equipment の間で転送されたデータを含むスケジュールされた推論に適用されます。詳細については、「Amazon Lookout for Equipment の料金」を参照してください。

予測 `lookoutequipment:CreateDataset` を追加したが、トレーニング後のアクティブ化を選択しなかった場合、アクティブ化して、アセットのモニタリングを開始する必要があります。

予測の推論を開始するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[アセット] を選択し、予測を追加するアセットを選択します。
3. [予測] を選択します。

4. アクティブ化する予測を選択します。
5. [アクション] で、[推論を開始] を選択し、以下を実行します。
   a. [入力データ] で、[データアップロードの頻度] として、データのアップロード頻度を定義し、[オフセット遅延時間] として、使用するバッファの量を定義します。
   b. [保存して開始] を選択します。

予測の推論を停止するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[アセット] を選択し、予測を追加するアセットを選択します。
3. [予測] を選択します。
4. 停止する予測を選択します。
5. [アクション] で、[推論を停止] を選択します。

予測定義を追加 (CLI)

新規または既存のアセットモデルで予測定義を定義するには、 AWS Command Line Interface () を使用できますAWS CLI。アセットモデルで予測定義を定義したら、Lookout for Equipment で異常検出を行うために、 でアセットの予測 AWS IoT SiteWise をトレーニングし、推論をスケジュールします。

前提条件

これらのステップを完了するには、アセットモデルと少なくとも 1 つのアセットを作成する必要があります。詳細については、「アセットモデルを作成する (AWS CLI)」および「アセットを作成する (AWS CLI)」を参照してください。

を初めて使用する場合は AWS IoT SiteWise、 `CreateBulkImportJob` API オペレーションを呼び出してアセットプロパティ値をインポートする必要があります。これは AWS IoT SiteWiseモデルのトレーニングに使用されます。詳細については、「AWS IoT SiteWise 一括インポートジョブを作成する (AWS CLI) 」を参照してください。

予測定義を追加するには

1. `asset-model-payload.json` という名前のファイルを作成します。これらの他のセクションのステップに従って、アセットモデルの詳細をファイルに追加しますが、アセットモデルの作成または更新のリクエストは送信しないでください。

   - アセットモデルの作成方法については、「アセットモデルを作成する (AWS CLI)」を参照してください。
   - 既存のアセットモデルを更新する方法については、「アセットまたはコンポーネントモデルを更新する (AWS CLI)」を参照してください。

2. 次のコードを追加することによって、Lookout for Equipment 複合モデル (`assetModelCompositeModels`) をアセットモデルに追加します。

   - *`Property`* を、含めるプロパティの ID に置き換えます。これらの ID を取得するには、`DescribeAssetModel` を呼び出します。
   - Lookout for Equipment が AWS IoT SiteWise データにアクセスできるようにする IAM ロールの ARN *`RoleARN`*に置き換えます。

```
{
  ...
  "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. アセットモデルを作成、または既存のアセットモデルを更新します。次のいずれかを行います：

   - アセットモデルを作成するには、次のコマンドを実行します。

     ```
     aws iotsitewise create-asset-model --cli-input-json file://asset-model-
     payload.json
     ```

   - 既存のアセットモデルを更新するには、次のコマンドを実行します。*asset-model-id* を、更新するアセットモデルの ID に置き換えます。

     ```
     aws iotsitewise update-asset-model \
       --asset-model-id asset-model-id \
       --cli-input-json file://asset-model-payload.json
     ```

コマンドを実行したら、レスポンスに `assetModelId` に注意してください。

予測をトレーニングして、推論を開始する (CLI)

予測定義が定義されたので、それに基づいてアセットをトレーニングし、推論を開始できます。予測をトレーニングするが、推論を開始しない場合は、予測をトレーニングする (CLI) に進みます。予測をトレーニングして、アセットの推論を開始するには、ターゲットリソースの `assetId` が必要です。

予測をトレーニングし、推論を開始するには

1. 以下のコマンドを実行して、assetModelCompositeModelSummaries で assetModelCompositeModelId を見つけます。*asset-model-id* を アセットまたはコンポーネントモデルを更新する (AWS CLI) で作成したアセットモデルの ID に置き換えます。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 次のコマンドを実行して、TrainingWithInference アクションの actionDefinitionId を検索します。*asset-model-id* を前のステップで使用した ID に置き換え、*asset-model-composite-model-id* を前のステップで返された ID に置き換えます。

```
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 というファイルを作成し、次のコードを追加して、次のように置き換えます。

   - *asset-id* をターゲットアセットの ID に
   - *action-definition-id* を TrainingWithInference アクションの ID に
   - *StartTime* をトレーニングデータの開始に (エポック秒で指定)
   - *EndTime* をトレーニングデータの終了に (エポック秒で指定)
   - *TargetSamplingRate* を Lookout for Equipment による後処理後のデータのサンプリングレートに。許可される値は PT1S | PT5S | PT10S | PT15S | PT30S | PT1M | PT5M | PT10M | PT15M | PT30M | PT1H です。

```
{
  "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. 次のコマンドを実行して、トレーニングと推論を開始します。

```
aws iotsitewise execute-action --cli-input-json file://train-start-inference-
prediction.json
```

予測をトレーニングする (CLI)

予測定義が定義されたので、それに基づいてアセットをトレーニングできます。アセットの予測をトレーニングするには、ターゲットリソースの assetId が必要です。

予測をトレーニングするには

1. 以下のコマンドを実行して、assetModelCompositeModelSummaries で assetModelCompositeModelId を見つけます。*asset-model-id* を アセットまたはコンポーネントモデルを更新する (AWS CLI) で作成したアセットモデルの ID に置き換えます。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 次のコマンドを実行して、Training アクションの actionDefinitionId を検索します。*asset-model-id* を前のステップで使用した ID に置き換え、*asset-model-composite-model-id* を前のステップで返された ID に置き換えます。

```
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.json というファイルを作成し、次のコードを追加して、次のように置き換えます。

   - *asset-id* をターゲットアセットの ID に
   - *action-definition-id* をトレーニングアクションの ID に
   - *StartTime* をトレーニングデータの開始に (エポック秒で指定)
   - *EndTime* をトレーニングデータの終了に (エポック秒で指定)
   - (オプション) *BucketName* を、ラベルデータを保持する Amazon S3 バケットの名前に

- (オプション) *Prefix* を、Amazon S3 バケットに関連付けられたプレフィックスに。
- *TargetSamplingRate* を Lookout for Equipment による後処理後のデータのサンプリングレートに。許可される値は `PT1S | PT5S | PT10S | PT15S | PT30S | PT1M | PT5M | PT10M | PT15M | PT30M | PT1H` です。

> **Note**
> バケット名とプレフィックスの両方を含めるか、どちらも含めないでください。

```
{
  "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. 次のコマンドを実行して、トレーニングを開始します。

```
aws iotsitewise execute-action --cli-input-json file://train-prediction.json
```

推論を開始する前に、トレーニングを完了する必要があります。トレーニングのステータスを確認するには、次のいずれかを行います。

- コンソールから、予測対象のアセットに移動します。
- から AWS CLI、 `trainingStatus`プロパティ`propertyId`の`BatchGetAssetPropertyValue`を使用して を呼び出します。

予測の推論を開始または停止する (CLI)

予測がトレーニングされたら、推論を開始して、Lookout for Equipment にアセットのモニタリングを開始するように指示できます。推論を開始または停止するには、ターゲットリソースの `assetId` が必要です。

推論を開始するには

1. 以下のコマンドを実行して、`assetModelCompositeModelSummaries` で `assetModelCompositeModelId` を見つけます。*asset-model-id* を アセットまたはコンポーネントモデルを更新する (AWS CLI) で作成したアセットモデルの ID に置き換えます。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 次のコマンドを実行して、`Inference` アクションの `actionDefinitionId` を検索します。*asset-model-id* を前のステップで使用した ID に置き換え、*asset-model-composite-model-id* を前のステップで返された ID に置き換えます。

```
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` というファイルを作成し、次のコードを追加して、次のように置き換えます。

   - *asset-id* をターゲットアセットの ID に
   - *action-definition-id* を推論開始アクションの ID に
   - *Offset* を、使用するバッファの量に
   - *Frequency* を、データがアップロードされる頻度に

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4EInference\": {\"inferenceMode\":\"START
\",\"dataDelayOffsetInMinutes\": Offset, \"dataUploadFrequency\": \"Frequency\"}}"
```

```
}}
```

4. 次のコマンドを実行して、推論を実行します。

```
aws iotsitewise execute-action --cli-input-json file://start-inference.json
```

推論を停止するには

1. 以下のコマンドを実行して、assetModelCompositeModelSummaries で assetModelCompositeModelId を見つけます。*asset-model-id* を アセットまたはコンポーネントモデルを更新する (AWS CLI) で作成したアセットモデルの ID に置き換えます。

```
aws iotsitewise describe-asset-model \
  --asset-model-id asset-model-id \
```

2. 次のコマンドを実行して、Inference アクションの actionDefinitionId を検索します。*asset-model-id* を前のステップで使用した ID に置き換え、*asset-model-composite-model-id* を前のステップで返された ID に置き換えます。

```
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 というファイルを作成し、次のコードを追加して、次のように置き換えます。

   - *asset-id* をターゲットアセットの ID に
   - *action-definition-id* を推論開始アクションの ID に

```
{
  "targetResource": {
    "assetId": "asset-id"
  },
  "actionDefinitionId": "action-definition-Id",
  "actionPayload":{ "stringValue": "{\"l4EInference\":{\"inferenceMode\":\"STOP
\"}}"
}}
```

4. 次のコマンドを実行して、推論を停止します。

```
aws iotsitewise execute-action --cli-input-json file://stop-inference.json
```

ネイティブ異常検出

AWS IoT SiteWise ネイティブ異常検出は、異常な機器動作を検出し、潜在的な障害を特定する産業機器を監視するための機械学習 (ML) 機能です。ネイティブ異常検出を使用すると、予測メンテナンスプログラムを実装し、最適でない機器プロセスを特定できます。

AWS IoT SiteWise ネイティブ異常検出には、広範な ML の知識や経験は必要ありません。プロパティを選択して、潜在的な障害を検出するカスタム ML モデルをトレーニングするだけです。 AWS IoT SiteWise ネイティブ異常検出は、機器の通常の動作状態を学習するための最適なモデルを自動的に作成します。このモデルは、履歴データで発生した異常な機器動作を見つけるように最適化されています。 AWS IoT SiteWise コンソールまたは SDK を使用してモデルを実行し、希望するスケジュールに従って新しい時系列データを処理します。

AWS IoT SiteWise ネイティブ異常検出を使用するには、以下を実行します。

- トレーニングするプロパティと期間を選択します。
- データ (ラベルデータ) に存在する場合は、データ (ラベルデータ) に表示される過去の障害の期間を追加します。
- AWS IoT SiteWise ネイティブ異常検出を使用して ML モデルをトレーニングします。
- トレーニング済みモデルに対してライブデータストリームをテストするように推論スケジュールを設定します。

ネイティブ異常検出は、動作条件の変動が限られて動作する固定および固定の産業機器を監視します。サポートされている機器には、ポンプ、圧縮機、モーター、コンピュータ数値制御 (CNC) マシン、タービンなどの回転機械が含まれます。プロセス産業のアプリケーションには、ヒートエクスチェンジャー、ボイラー、およびコンポジットが含まれます。ネイティブ異常検出は、 に統合 AWS IoT SiteWise され、既存のメンテナンスシステムを補完するバックエンド分析サービスです。

トピック

- ネイティブ異常検出機能
- 前提条件
- アセットのセンサーで異常検出を有効にする
- アセット間でセンサーの異常検出を有効にする
- 高度なトレーニング設定
- 高度な推論設定

- 推論結果を確認する
- 異常な動作に対してカスタムアクションをトリガーする (AWS Management Console)
- ベストプラクティス

ネイティブ異常検出機能

AWS IoT SiteWise プラットフォームは、産業オペレーションを変革できるさまざまな強力な機能を提供します。

- 予測メンテナンス: 障害を早期に検出し、作業指示システムと統合してインサイトに対応します。
- との緊密な統合 AWS IoT SiteWise: 既存のインフラストラクチャへの緊密な統合を活用して、データを外部サービスに移動する必要がなくなります。
- 自動モデル選択とトレーニング: 機械学習の専門知識を必要とせずに AWS IoT SiteWise、このサポートを活用できます。
- 推論スケジューリングをカスタマイズする: 運用ニーズとシフトタイミングに合わせて推論ジョブをスケジュールします。
- ラベル付きデータ: モデルトレーニング中の既知の障害間隔の精度を向上させます。
- ポイント単位の診断によるモデル評価: イベントレベルでモデルのパフォーマンスを評価します。
- アセット間でスケーラブル: 個々のアセットまたは複数のアセット間で計算モデルを作成し、スケーラビリティを実現します。
- センサーレベルのインサイト: 異常の原因となっている特定のセンサー (複数可) を識別する詳細な診断を取得します。
- 価値実現までの時間の短縮: センサーモデリングから異常検出に数週間ではなく数時間で移行できます。

前提条件

これらのステップを完了するには、アセットモデルと少なくとも 1 つのアセットを作成する必要があります。詳細については、「アセットモデルの作成 (AWS CLI)」および「アセットの作成 (AWS CLI)」を参照してください。現時点では、外部 IDsはサポートされていません。

を初めて使用する場合 AWS IoT SiteWise （履歴データがない場合）、CreateBulkImportJob API を呼び出してアセットプロパティ値をインポートする必要があります AWS IoT SiteWise。これはモデルのトレーニングに使用されます。詳細については、AWS IoT SiteWise 「一括インポートジョブの作成 (AWS CLI)」を参照してください。

計算モデル APIs AWS CLI のセットアップ

AWS CLI 設定を更新し、計算モデル APIs にアクセスするには、次の手順に従います。

- 最新の awscli バージョン をインストールしますaws-cli。
- 新しい APIs を確認して、インストールを確認します。

```
aws iotsitewise help
```

コマンド出力には、新しく追加された計算モデルオペレーションを含む AWS IoT SiteWise APIs の完全なリストが表示されます。

プロパティ要件

異常検出を設定するには、次の要件と UpdateAssetModel (AWS CLI) が必要です。

- または DOUBLE INTEGER データ型のいずれかである少なくとも 1 つの入力プロパティ。これは測定プロパティまたは変換プロパティのいずれかであり、モデルのトレーニングに使用されます。
- STRING データ型の結果プロパティ。これは測定プロパティでなければならず、異常検出結果を保存します。

ラベリングの前提条件

- データラベルを Amazon S3 バケットにアップロードします。
- このバケットのバケットポリシーを更新して AWS IoT SiteWise 、 がラベルを読み取れるようにします。

  コンソールで、アクセス許可 -> バケットポリシーに移動します。次のポリシーを貼り付け、bucket-arn をバケットの ARN に置き換えます。

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Sid": "SiteWiseReadAccess",
             "Effect": "Allow",
             "Principal": {
                 "Service": "iotsitewise.amazonaws.com"
```

```
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "bucket-arn",
                "bucket-arn/*"
            ]
        }
    ]
}
```

モデル評価の前提条件

- モデル評価では、提供されている Amazon S3 バケットの場所にポイント単位のモデル診断が生成されます。
- ポイント単位の診断結果を Amazon S3 バケットに書き込むには、このバケットのバケットポリシーを更新して、 が AWS IoT SiteWise 結果を書き込めるようにします。

  コンソールで、 に移動します`Permissions -> Bucket policy`。次のポリシーを貼り付け、`bucket-arn`をバケットの ARN に置き換えます。

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Sid": "SiteWiseWriteAccess",
              "Effect": "Allow",
              "Principal": {
                  "Service": "iotsitewise.amazonaws.com"
              },
              "Action": [
                  "s3:GetObject",
                  "s3:ListBucket",
                  "s3:PutObject"
              ],
              "Resource": [
                  "bucket-arn",
                  "bucket-arn/*"
              ]
```

```
            }
        ]
    }
```

アセットのセンサーで異常検出を有効にする

計算モデルを作成する (AWS CLI)

計算モデルを作成するには、 コマンドラインインターフェイス () AWS を使用しますAWS CLI。計算モデルを定義したら、モデルをトレーニングし、アセットの異常検出を実行するように推論をスケジュールします AWS IoT SiteWise。

- 次の内容`anomaly-detection-computation-model-payload.json`のファイルを作成します。

```
{
    "computationModelName": "anomaly-detection-computation-model-name",
    "computationModelConfiguration": {
        "anomalyDetection": {
            "inputProperties": "${input_properties}",
            "resultProperty": "${result_property}"
        }
    },
    "computationModelDataBinding": {
        "input_properties": {
            "list": [{
                    "assetModelProperty": {
                        "assetModelId": "asset-model-id",
                        "propertyId": "input-property-id-1"
                    }
                },
                {
                    "assetModelProperty": {
                        "assetModelId": "asset-model-id",
                        "propertyId": "input-property-id-2"
                    }
                }
            ]
        },
        "result_property": {
            "assetModelProperty": {
```

```
                "assetModelId": "asset-model-id",
                "propertyId": "results-property-id"
            }
        }
    }
}
```

- 次のコマンドを実行して、計算モデルを作成します。

```
aws iotsitewise create-computation-model \
    --cli-input-json file://anomaly-detection-computation-model-payload.json
```

ExecuteAction API ペイロードの準備

トレーニングと推論を実行する次のステップは、ExecuteAction API を使用して実行されます。トレーニングと推論の両方が JSON アクションペイロード設定で設定されます。ExecuteAction API を呼び出すときは、アクションペイロードを`stringValue`ペイロードの値として指定する必要があります。

ペイロードは API 要件に厳密に従う必要があります。具体的には、値は制御文字 (改行、タブ、キャリッジリターンなど) を含まないフラット文字列である必要があります。

以下のオプションは、有効なアクションペイロードを提供する 2 つの信頼できる方法を提供します。

オプション 1: クリーンペイロードファイルを使用する

次の手順では、クリーンペイロードファイルのステップについて説明します。

1. ファイルをクリーンアップしてコントロール文字を削除します。

```
tr -d '\n\r\t' < original-action-payload.json > training-or-inference-action-
payload.json
```

2. ファイル を使用して アクションを実行します`@=file://...`。

```
aws iotsitewise execute-action \
    --target-resource computationModelId=<MODEL_ID> \
    --action-definition-id <ACTION_DEFINITION_ID> \
    --resolve-to assetId=<ASSET_ID> \
```

```
    --action-payload stringValue@=file://training-or-inference-action-payload.json
```

オプション 2: エスケープされた引用符を含むインライン文字列

次の手順では、ペイロードをインラインで提供し、中間ファイルを回避する手順について説明します。

- JSON 文字列内でエスケープされた二重引用符 (\") を使用します。
- `StringValue=..` 式全体を二重引用符で囲みます。

Example エスケープされたアクションペイロードの :

```
aws iotsitewise execute-action \
    --target-resource computationModelId=<MODEL_ID> \
    --action-definition-id <ACTION_DEFINITION_ID> \
    --resolve-to assetId=<ASSET_ID> \
    --action-payload "stringValue={\"exportDataStartTime\":1717225200,
\"exportDataEndTime\":1722789360,\"targetSamplingRate\":\"PT1M\"}"
```

をトレーニングする AWS CLI

計算モデルを作成すると、アセットに対してモデルをトレーニングできます。アセットのモデルをトレーニングするには、次の手順に従います。

1. 次のコマンドを実行して、`AWS/ANOMALY_DETECTION_TRAINING` アクションの `actionDefinitionId` を検索します。を前のステップで返された ID `computation-model-id`に置き換えます。

   ```
   aws iotsitewise describe-computation-model \
       --computation-model-id computation-model-id
   ```

2. という名前のファイルを作成し`anomaly-detection-training-payload.json`、次の値を追加します。

   > **Note**
   >
   > ペイロードは に準拠している必要があります[オプション 1: クリーンペイロードファイルを使用する](#)。

a. `StartTime` エポック秒で提供されるトレーニングデータの開始。

b. `EndTime` エポック秒で提供されるトレーニングデータの終了。

c. オプションで を設定して高度なトレーニング設定、モデルのパフォーマンスを向上させることができます。

   i. (オプション) `TargetSamplingRate`データのサンプリングレート。

   ii. (オプション) モデルトレーニングを改善`LabelInputConfiguration`するための異常な動作が発生した期間を指定します。

   iii. (オプション) `ModelEvaluationConfiguration` トレーニングの完了後に指定された時間範囲で推論を実行してモデルのパフォーマンスを評価します。

```
{
  "exportDataStartTime": StartTime,
  "exportDataEndTime": EndTime
}
```

Example トレーニングペイロードの例：

```
{
  "exportDataStartTime": 1717225200,
  "exportDataEndTime": 1722789360
}
```

3. 次のコマンドを実行してトレーニングを開始します。コマンド内の以下のパラメータを置き換えてください。

   a. `computation-model-id` ターゲット計算モデルの ID を指定します。

   b. `asset-id` モデルをトレーニングするアセットの ID を指定します。

   c. `training-action-definition-id` ステップ 1 の`AWS/ANOMALY_DETECTION_TRAINING`アクションの ID を持つ 。

```
aws iotsitewise execute-action \
    --target-resource computationModelId=computation-model-id \
    --resolve-to assetId=asset-id \
    --action-definition-id training-action-definition-id \
```

```
    --action-payload stringValue@=file://anomaly-detection-training-payload.json
```

Example 実行アクションの：

```
aws iotsitewise execute-action --target-resource computationModelId=27cb824c-
fd84-45b0-946b-0a5b0466d890 --resolve-to assetId=cefd4b68-481b-4735-
b466-6a4220cd19ee --action-definition-id e54cea94-5d1c-4230-a59e-4f54dcbc972d --
action-payload stringValue@=file://anomaly-detection-training-payload.json
```

4. 次のコマンドを実行して、モデルトレーニングプロセスのステータスを確認します。最新の実行概要には、実行ステータス (RUNNING/COMPLETED/) が表示されますFAILED。

```
aws iotsitewise list-executions \
    --target-resource-type COMPUTATION_MODEL \
    --target-resource-id computation-model-id\
    --resolve-to-resource-type ASSET \
    --resolve-to-resource-id asset-id
```

5. 次のコマンドを実行して、最新のトレーニング済みモデルの設定を確認します。このコマンドは、少なくとも 1 つのモデルが正常にトレーニングされた場合にのみ出力を生成します。

```
aws iotsitewise describe-computation-model-execution-summary \
    --computation-model-id computation-model-id \
    --resolve-to-resource-type ASSET \
    --resolve-to-resource-id asset-id
```

6. `ComputationModel` が AssetModelProperty を使用している場合は、ListComputationModelResolveToResources API を使用して、実行されたアクションを持つアセットを識別します。

```
aws iotsitewise list-computation-model-resolve-to-resources \
    --computation-model-id computation-model-id
```

推論の開始と停止 (AWS CLI)

モデルをトレーニングしたら、推論を開始します。これにより、 に異常がないか産業用アセットをアクティブにモニタリング AWS IoT SiteWise するように指示します。

推論を開始する

1. 次のコマンドを実行して、AWS/ANOMALY_DETECTION_INFERENCE アクションの actionDefinitionId を検索します。を、前に作成した計算モデルの実際の ID computation-model-idに置き換えます。

```
aws iotsitewise describe-computation-model \
    --computation-model-id computation-model-id
```

2. ファイルを作成しanomaly-detection-start-inference-payload.json、次の値を追加します。

> **Note**
> ペイロードは に準拠している必要があります[オプション 1: クリーンペイロードファイルを使用する](#)。

```
"inferenceMode": "START",
"dataUploadFrequency": "DataUploadFrequency"
```

   a. DataUploadFrequency: 異常検出を実行するために推論スケジュールを実行する頻度を設定します。許可される値は PT5M, PT10M, PT15M, PT30M, PT1H, PT2H..PT12H, PT1D です。
   b. (オプション) DataDelayOffsetInMinutes 遅延オフセットを分単位で指定します。この値を 0 ~ 60 分に設定します。
   c. (オプション) アクティブ化TargetModelVersionするモデルバージョン。
   d. (オプション) シフト設定weeklyOperatingWindowで を設定します。
   e. オプションで を設定できます[高度な推論設定](#)。
      i. [高頻度推論 (5 分 ~ 1 時間)](#).
      ii. [低頻度推論 (2 時間 – 1 日)](#).
      iii. [柔軟なスケジューリング](#).
3. 推論を開始するには、次のコマンドを実行します。ペイロードファイルの次のパラメータを置き換えます。
   a. computation-model-id ターゲット計算モデルの ID を指定します。

b. `asset-id` モデルがトレーニングされたアセットの ID。

c. `inference-action-definition-id` ステップ 1 のAWS/ANOMALY_DETECTION_INFERENCEアクションの ID を持つ 。

```
aws iotsitewise execute-action \
    --target-resource computationModelId=computation-model-id \
    --resolve-to assetId=asset-id \
    --action-definition-id inference-action-definition-id \
    --action-payload stringValue@=file://anomaly-detection-inference-payload.json
```

4. 次のコマンドを実行して、推論がまだ実行されているかどうかを確認します。推論がアクティブなTRUE場合、 `inferenceTimerActive`フィールドは に設定されます。

```
aws iotsitewise describe-computation-model-execution-summary \
    --computation-model-id computation-model-id \
    --resolve-to-resource-type ASSET \
    --resolve-to-resource-id asset-id
```

5. 次のコマンドは、すべての推論実行を一覧表示します。

```
aws iotsitewise list-executions \
   --target-resource-type COMPUTATION_MODEL \
   --target-resource-id computation-model-id \
   --resolve-to-resource-type ASSET \
   --resolve-to-resource-id asset-id
```

6. 次のコマンドを実行して、個々の実行を記述します。を前のステップ 5 の ID `execution-id`に置き換えます。

```
aws iotsitewise describe-execution \
    --execution-id execution-id
```

推論を停止する

1. 次のコマンドを実行して、AWS/ANOMALY_DETECTION_INFERENCE アクションの `actionDefinitionId` を検索します。を、前に作成した計算モデルの実際の ID `computation-model-id`に置き換えます。

```
aws iotsitewise describe-computation-model \
    --computation-model-id computation-model-id
```

2. ファイルを作成しanomaly-detection-stop-inference-payload.json、次のコードを追加します。

```
{
    "inferenceMode": "STOP"
}
```

> **Note**
>
> ペイロードは に準拠している必要があります[オプション 1: クリーンペイロードファイルを使用する](#)。

3. 推論を停止するには、次のコマンドを実行します。ペイロードファイルの次のパラメータを置き換えます。

   a. `computation-model-id` ターゲット計算モデルの ID を指定します。

   b. `asset-id` モデルがトレーニングされたアセットの ID。

   c. `inference-action-definition-id` ステップ 1 のAWS/ANOMALY_DETECTION_INFERENCEアクションの ID を持つ 。

   Example stop inference コマンドの：

```
aws iotsitewise execute-action \
    --target-resource computationModelId=computation-model-id \
    --resolve-to assetId=asset-id \
    --action-definition-id inference-action-definition-id \
    --action-payload stringValue@=file://anomaly-detection-stop-inference-
payload.json
```

データバインディングで特定のリソースを使用する計算モデルを検索する

特定のリソースにバインドされている計算モデルを一覧表示するには：

- アセットモデル (このアセットモデルのプロパティのいずれかがバインドされているすべての計算モデルを取得します) 。
- asset (このアセットのプロパティのいずれかがバインドされているすべての計算モデルを取得します)
- アセットモデルプロパティ (このプロパティがバインドされているすべての計算モデルを取得します)
- アセットプロパティ (このプロパティがバインドされているすべての計算モデルを取得します。 これは情報提供を目的としている場合もあれば、ユーザーがこのプロパティを別の計算モデルにバインドしようとしても、既に別の場所にバインドされている場合に必要になる場合もあります) 。

ListComputationModelDataBindingUsages API を使用して、アセット (プロパティ) またはアセットモデル (プロパティ) `ComputationModelId`をデータバインディングとして使用する のリストを取得します。

次の情報`request.json`を使用して を準備します。

```
{
  "dataBindingValueFilter": {
    "asset": {
      "assetId": "<string>"
    }
    // OR
    "assetModel": {
      "assetModelId": "<string>"
    }
    // OR
    "assetProperty": {
      "assetId": "<string>",
      "propertyId": "<string>"
    }
    // OR
    "assetModelProperty": {
      "assetModelId": "<string>",
      "propertyId": "<string>"
    }
  },
  "nextToken": "<string>",
  "maxResults": "<number>"
}
```

`list-computation-model-data-binding-usages` コマンドを使用して、アセットまたはアセットモデルをデータバインディングとしてモデルを取得します。

```
aws iotsitewise list-computation-model-data-binding-usages \
--cli-input-json file://request.json
```

アセット間でセンサーの異常検出を有効にする

計算モデルを作成する (AWS CLI)

計算モデルを作成するには、 コマンドラインインターフェイス () AWS を使用しますAWS CLI。計算モデルを定義したら、モデルをトレーニングし、推論をスケジュールして、 のアセット間で異常検出を実行します AWS IoT SiteWise。

次の手順では、このプロセスについて説明します。

1. 異常検出を設定するには、UpdateAssetModel (AWS CLI) を使用し、次の要件を満たします。
   a. または DOUBLE INTEGER データ型のいずれかである少なくとも 1 つの入力プロパティ。これは測定プロパティまたは変換プロパティのいずれかであり、モデルのトレーニングに使用されます。
   b. STRING データ型の結果プロパティ。これは測定プロパティでなければならず、異常検出結果を保存します。
2. 次の内容`anomaly-detection-computation-model-payload.json`のファイルを作成します。

   > **Note**
   > をデータソース`assetProperty`として直接指定して計算モデルを作成します。

```
{
    "computationModelName": "name of ComputationModel",
    "computationModelConfiguration": {
        "anomalyDetection": {
            "inputProperties": "${properties}",
            "resultProperty": "${p3}"
        }
    },
```

```
    "computationModelDataBinding": {
        "properties": {
            "list": [
                {
                    "assetProperty": {
                        "assetId": "asset-id",
                        "propertyId": "input-property-id-1"
                    }
                },
                {
                    "assetProperty": {
                        "assetId": "asset-id",
                        "propertyId": "input-property-id-2"
                    }
                }
            ]
        },
        "p3": {
            "assetProperty": {
                "assetId": "asset-id",
                "propertyId": "results-property-id"
            }
        }
    }
}
```

3. 次のコマンドを実行して、計算モデルを作成します。

```
aws iotsitewise create-computation-model \
    --cli-input-json file://anomaly-detection-computation-model-payload.json
```

ExecuteAction API ペイロードの準備

トレーニングと推論を実行する次のステップは、ExecuteAction API を使用して実行されます。トレーニングと推論の両方が JSON アクションペイロード設定で設定されます。ExecuteAction API を呼び出すときは、アクションペイロードを`stringValue`ペイロードの値として指定する必要があります。

ペイロードは API 要件に厳密に従う必要があります。具体的には、値は制御文字 (改行、タブ、キャリッジリターンなど) のないフラット文字列である必要があります。以下のオプションは、有効なアクションペイロードを提供する 2 つの信頼できる方法を提供します。

オプション 1: クリーンペイロードファイルを使用する

次の手順では、クリーンペイロードファイルのステップについて説明します。

1. ファイルをクリーンアップして、コントロール文字を削除します。

```
tr -d '\n\r\t' < original-action-payload.json > training-or-inference-action-
payload.json
```

2. ファイル を使用して アクションを実行します`@=file://...`。

```
aws iotsitewise execute-action \
    --target-resource computationModelId=<MODEL_ID> \
    --action-definition-id <ACTION_DEFINITION_ID> \
    --action-payload stringValue@=file://training-or-inference-action-payload.json
```

オプション 2: エスケープされた引用符を含むインライン文字列

次の手順では、ペイロードをインラインで提供し、中間ファイルを回避する手順について説明します。

- JSON 文字列内でエスケープされた二重引用符 (\") を使用します。
- `StringValue=..` 式全体を二重引用符で囲みます。

Example エスケープされたアクションペイロードの :

```
aws iotsitewise execute-action \
    --target-resource computationModelId=<MODEL_ID> \
    --action-definition-id <ACTION_DEFINITION_ID> \
    --action-payload "stringValue={\"exportDataStartTime\":1717225200,
\"exportDataEndTime\":1722789360,\"targetSamplingRate\":\"PT1M\"}"
```

をトレーニングする AWS CLI

1. 次のコマンドを実行して、`AWS/ANOMALY_DETECTION_TRAINING` アクションの `actionDefinitionId` を検索します。を前のステップで返された ID `computation-model-id`に置き換えます。

```
aws iotsitewise describe-computation-model \
```

```
    --computation-model-id computation-model-id
```

2. という名前のファイルを作成しanomaly-detection-training-payload.json、次の値を追加します。

> **Note**
>
> ペイロードは に準拠している必要があります[オプション 1: クリーンペイロードファイルを使用する](#)。

   a. StartTime エポック秒で提供されるトレーニングデータの開始。
   b. EndTime エポック秒で提供されるトレーニングデータの終了。
   c. オプションで を設定できます[高度な推論設定](#)。
      i. (オプション) TargetSamplingRateデータのサンプリングレート。
      ii. (オプション) モデルトレーニングを改善するために異常な動作が発生した期間LabelInputConfigurationを指定します。
      iii. (オプション) ModelEvaluationConfiguration トレーニングの完了後に指定された時間範囲で推論を実行してモデルのパフォーマンスを評価します。

```
{
  "exportDataStartTime": StartTime,
  "exportDataEndTime": EndTime
}
```

Example トレーニングペイロードの例：

```
{
  "exportDataStartTime": 1717225200,
  "exportDataEndTime": 1722789360
}
```

3. 次のコマンドを実行してトレーニングを開始します (アセットをターゲットリソースとして指定しません) 。コマンド内の以下のパラメータを置き換えてください。

```
aws iotsitewise execute-action \
    --target-resource computationModelId=computation-model-id \
```

```
    --action-definition-id training-action-definition-id \
    --action-payload stringValue@=file://anomaly-detection-training-payload.json
```

4. 次のコマンドを実行して、モデルトレーニングプロセスのステータスを確認します。最新の実行概要には、実行ステータス (RUNNING/COMPLETED/) が表示されますFAILED。

```
aws iotsitewise list-executions \
    --target-resource-type COMPUTATION_MODEL \
    --target-resource-id computation-model-id
```

5. 次のコマンドを実行して、最新のトレーニング済みモデルの設定を確認します。このコマンドは、少なくとも 1 つのモデルが正常にトレーニングを完了した場合にのみ出力を生成します。

```
aws iotsitewise describe-computation-model-execution-summary \
    --computation-model-id computation-model-id
```

推論の開始と停止 (AWS CLI)

モデルをトレーニングしたら、推論を開始します。推論は、 に産業用アセットの異常のモニタリングを開始するよう指示 AWS IoT SiteWise します。

推論を開始する

1. 次のコマンドを実行して、`AWS/ANOMALY_DETECTION_INFERENCE` アクションの `actionDefinitionId` を検索します。を、前に作成した計算モデルの実際の ID `computation-model-id`に置き換えます。

```
aws iotsitewise describe-computation-model \
    --computation-model-id computation-model-id
```

2. ファイルを作成し`anomaly-detection-start-inference-payload.json`、次のコードを追加します。説明に従って、次のパラメータを置き換えます。

> **Note**
>
> ペイロードは に準拠している必要があります[オプション 1: クリーンペイロードファイルを使用する](#)。

a. `DataUploadFrequency`: 異常検出を実行するために推論スケジュールを実行する頻度を設定します。許可される値は `PT5M`, `PT10M`, `PT15M`, `PT30M`, `PT1H`, `PT2H..PT12H`, `PT1D` です。

```
"inferenceMode": "START",
"dataUploadFrequency": "DataUploadFrequency"
```

b. (オプション) `DataDelayOffsetInMinutes` 遅延オフセットを分単位で指定します。この値を 0 ~ 60 分に設定します。

c. (オプション) アクティブ化`TargetModelVersion`するモデルバージョン。

d. (オプション) シフト設定`weeklyOperatingWindow`で を設定します。

e. オプションで を設定できます高度な推論設定。

   i. 高頻度推論 (5 分 ~ 1 時間) .

   ii. 低頻度推論 (2 時間 – 1 日) .

   iii. 柔軟なスケジューリング.

3. 推論を開始するには、次のコマンドを実行します。ペイロードファイルの次のパラメータを置き換えます。

   a. `computation-model-id` ターゲット計算モデルの ID を指定します。

   b. `inference-action-definition-id` ステップ 1 の`AWS/ANOMALY_DETECTION_INFERENCE`アクションの ID を持つ 。

```
aws iotsitewise execute-action \
    --target-resource computationModelId=computation-model-id \
    --action-definition-id inference-action-definition-id \
    --action-payload stringValue@=file://anomaly-detection-inference-payload.json
```

4. 次のコマンドを実行して、推論がまだ実行されているかどうかを確認します。推論がアクティブな`TRUE`場合、 `inferenceTimerActive`フィールドは に設定されます。

```
aws iotsitewise describe-computation-model-execution-summary \
    --computation-model-id computation-model-id
```

5. 次のコマンドは、すべての推論実行を一覧表示します。

```
aws iotsitewise list-executions \
    --target-resource-type COMPUTATION_MODEL \
    --target-resource-id computation-model-id
```

6. 次のコマンドを実行して、個々の実行を記述します。を前のステップ 5 の ID `execution-id`に置き換えます。

```
aws iotsitewise describe-execution \
    --execution-id execution-id
```

推論を停止する

1. 次のコマンドを実行して、`AWS/ANOMALY_DETECTION_INFERENCE` アクションの `actionDefinitionId` を検索します。を、前に作成した計算モデルの実際の ID `computation-model-id`に置き換えます。

```
aws iotsitewise describe-computation-model \
    --computation-model-id computation-model-id
```

2. ファイルを作成し`anomaly-detection-stop-inference-payload.json`、次のコードを追加します。

```
{
    "inferenceMode": "STOP"
}
```

> **Note**
>
> ペイロードは に準拠している必要があります[オプション 1: クリーンペイロードファイルを使用する](#)。

3. 推論を停止するには、次のコマンドを実行します。ペイロードファイルの次のパラメータを置き換えます。

   a. `computation-model-id` ターゲット計算モデルの ID を指定します。

   b. `inference-action-definition-id` ステップ 1 の`AWS/ANOMALY_DETECTION_INFERENCE`アクションの ID を持つ 。

Example stop inference コマンドの：

```
aws iotsitewise execute-action \
--target-resource computationModelId=computation-model-id \
--action-definition-id inference-action-definition-id \
--action-payload stringValue@=file://anomaly-detection-stop-inference-payload.json
```

高度なトレーニング設定

サンプルレート設定

サンプルレートは、センサーの読み取り値を記録する頻度 (1 秒に 1 回、1 分に 1 回など) を定義します。この設定は、トレーニングデータの粒度に直接影響し、センサーの動作の短期的な変動をキャプチャするモデルの能力に影響します。

ベストプラクティスについては高頻度データとトレーニングと推論間の整合性のサンプリング、「」を参照してください。

ターゲットサンプリングレートを設定する

必要に応じて、トレーニング設定`TargetSamplingRate`で を指定して、データがサンプリングされる頻度を制御できます。サポートされている値は以下のとおりです。

```
PT1S | PT5S | PT10S | PT15S | PT30S | PT1M | PT5M | PT10M | PT15M | PT30M | PT1H
```

これらは ISO 8601 期間形式であり、次の時間形式を表します。

- PT1S = 1 秒
- PT1M = 1 分
- PT1H = 1 時間

データ解決とトレーニング効率の適切なバランスを取るサンプリングレートを選択します。次の料金を利用できます。

- サンプリングレートが高いほど (PT1S) 詳細が細かくなりますが、データ量とトレーニング時間が長くなる可能性があります。

- サンプリングレート (、) を低くすると、データサイズとコストを削減できますが、存続期間の短い異常を見逃す可能性があります。PT10M PT1H

タイムスタンプのずれの処理

AWS IoT SiteWise は、トレーニング中に複数のデータストリーム間でタイムスタンプのずれを自動的に補正します。これにより、入力信号が時間内に完全に整列しなくても、一貫したモデル動作が保証されます。

ベストプラクティスについては高頻度データとトレーニングと推論間の整合性のサンプリング、「」を参照してください。

サンプリングを有効にする

次のコードを に追加しますanomaly-detection-training-payload.json。

データのサンプリングレートを使用して、トレーニングアクションペイロードTargetSamplingRateに を追加してサンプリングを設定します。使用できる値は ですPT1S | PT5S | PT10S | PT15S | PT30S | PT1M | PT5M | PT10M | PT15M | PT30M | PT1H。

```
{
    "exportDataStartTime": StartTime,
    "exportDataEndTime": EndTime,
    "targetSamplingRate": "TargetSamplingRate"
}
```

Example サンプルレート設定の：

```
{
    "exportDataStartTime": 1717225200,
    "exportDataEndTime": 1722789360,
    "targetSamplingRate": "PT1M"
}
```

データのラベル付け

データにラベルを付けるときは、異常な機器動作の期間を表す時間間隔を定義する必要があります。このラベル情報は CSV ファイルとして提供され、各行は機器が正しく動作しなかった時間範囲を指定します。

各行には 2 つのタイムスタンプが含まれます。

- 異常な動作が開始されたと考えられる時刻を示す開始時刻。
- 障害または問題が最初に検出された時刻を表す終了時刻。

この CSV ファイルは Amazon S3 バケットに保存され、モデルトレーニング中に使用され、システムが異常な動作の既知の例から学習するのに役立ちます。次の例は、ラベルデータを`.csv`ファイルとして表示する方法を示しています。ファイルには ヘッダーがありません。

Example CSV ファイルの：

```
2024-06-21T00:00:00.000000,2024-06-21T12:00:00.000000
2024-07-11T00:00:00.000000,2024-07-11T12:00:00.000000
2024-07-31T00:00:00.000000,2024-07-31T12:00:00.000000
```

行 1 は 2024 年 6 月 21 日のメンテナンスイベントを表し、 が異常な動作を探す AWS IoT SiteWise ための のウィンドウは 12 時間 (から `2024-06-21T00:00:00.000000Z` `2024-06-21T12:00:00.000000Z`) です。

行 2 は、2024 年 7 月 11 日のメンテナンスイベントを表し、 が異常な動作を探す AWS IoT SiteWise ための の 12 時間のウィンドウ (から `2024-07-11T00:00:00.000000Z` `2024-07-11T12:00:00.000000Z`) があります。

行 3 は、2024 年 7 月 31 日のメンテナンスイベントを表し、 が異常な動作を探す AWS IoT SiteWise ための のウィンドウは 12 時間 (から `2024-07-31T00:00:00.000000Z` `2024-07-31T12:00:00.000000Z`) です。

AWS IoT SiteWise は、これらのすべての時間枠を使用して、これらのイベントに関する異常な動作を特定できるモデルをトレーニングおよび評価します。すべてのイベントが検出可能であるわけではなく、結果は基盤となるデータの品質と特性に大きく依存することに注意してください。

サンプリングのベストプラクティスの詳細については、「」を参照してください[ベストプラクティス](#)。

データラベリング手順

- 「ラベル付け[データの前提条件](#)」のラベル付けの前提条件に従って Amazon S3 バケットを設定します。
- ファイルをラベル付けバケットにアップロードします。

- に以下を追加しますanomaly-detection-training-payload.json。
  - ファイルの labelInputConfigurationセクションで場所を指定します。labels-bucket をバケット名に置き換え、 files-prefix を file (s) パスまたはプレフィックスの一部に置き換えます。の場所にあるすべてのファイルが解析され、 (成功すると) ラベルファイルとして使用されます。

```
{
    "exportDataStartTime": StartTime,
    "exportDataEndTime": EndTime,
    "labelInputConfiguration":
      {
       "bucketName": "label-bucket",
       "prefix": "files-prefix"
      }
}
```

Example ラベル設定の：

```
{
    "exportDataStartTime": 1717225200,
    "exportDataEndTime": 1722789360,
    "labelInputConfiguration": {
      "bucketName": "anomaly-detection-customer-data-278129555252-iad",
      "prefix": "Labels/model=b2d8ab3e-73af-48d8-9b8f-a290bef931b4/
asset[d3347728-4796-4c5c-afdb-ea2f551ffe7a]/Lables.csv"
    }
}
```

モデルの評価

AWS IoT SiteWise トレーニングモデルのポイント単位のモデル診断は、個々のイベントでのモデルパフォーマンスの評価です。トレーニング中、 は入力データセットの各行の異常スコアとセンサー寄与度診断 AWS IoT SiteWise を生成します。異常スコアが高いほど、異常イベントが発生する可能性が高くなります。

ExecuteAction API とAWS/ANOMALY_DETECTION_TRAININGアクションタイプを使用してモデルをトレーニングする場合、ポイント単位の診断を使用できます。

モデル評価を設定するには、

- 「ラベル付けデータの前提条件」のラベル付けの前提条件に従って Amazon S3 バケットを設定します。
- に以下を追加しますanomaly-detection-training-payload.json。
  - モデルのパフォーマンスを評価するために使用されるウィンドウ内のデータの evaluationStartTimeと evaluationEndTime (両方とも エポック秒) を指定します。
  - 評価診断を書き込むには、Amazon S3 バケットの場所 (resultDestination) を指定します。

> **Note**
>
> モデル評価間隔 (dataStartTime から dataEndtime) は、トレーニング間隔と重複するか、連続している必要があります。ギャップは許可されません。

```
{
  "exportDataStartTime": StartTime,
  "exportDataEndTime": EndTime,
  "modelEvaluationConfiguration": {
    "dataStartTime": evaluationStartTime,
    "dataEndTime": evaluationEndTime
    "resultDestination": {
      "bucketName": "s3BucketName",
      "prefix": "bucketPrefix"
    }
  }
}
```

Example モデル評価設定の：

```
{
  "exportDataStartTime": 1717225200,
  "exportDataEndTime": 1722789360,
  "modelEvaluationConfiguration": {
    "dataStartTime": 1722789360,
    "dataEndTime": 1725174000,
    "resultDestination": {
      "bucketName": "anomaly-detection-customer-data-278129555252-iad",
      "prefix": "Evaluation/asset[d3347728-4796-4c5c-afdb-ea2f551ffe7a]/1747681026-
evaluation_results.jsonl"
```

```
    }
  }
}
```

高度な推論設定

AWS IoT SiteWise を使用すると、お客様は運用上のニーズに合わせてモデル推論スケジュールを設定できます。

推論スケジューリングは、3 つのモードに大別されます。

- 高頻度推論 (5 分 ~ 1 時間)
- 低頻度推論 (2 時間 – 1 日)
- 柔軟なスケジューリング

高頻度推論 (5 分 ~ 1 時間)

このモードは、継続的に動作するプロセスや、センサー値の変化率が高いプロセスに最適です。この設定では、推論は 5 分ごとに頻繁に実行されます。

ユースケース：

- これは、コンプレッサーやコンベアなどの急速に変化する機器のモニタリングに使用されます。
- これは、即時の対応を必要とする存続期間の短い異常をキャッチするのに役立ちます。
- これは、データが一貫して流れている常時オンのオペレーションです。

条件付きオフセットのサポート：

条件付きオフセット (0 ~ 60 分) を定義して、データ取り込み後に推論を遅らせることができます。これにより、遅延着信データが引き続き分析ウィンドウに含まれます。

高頻度推論を設定するには：

- 推論の開始`PT5M`, `PT10M`, `PT15M`, `PT30M`, `PT1H`時に、 の値`DataUploadFrequency`を使用して`AWS/ANOMALY_DETECTION_INFERENCE`アクションペイロード値を設定します。
- (オプション) 遅延オフセット`DataDelayOffsetInMinutes`を分単位で設定します。この値を 0 ~ 60 分に設定します。

```
{
    "inferenceMode": "START",
    "dataDelayOffsetInMinutes": "DataDelayOffsetInMinutes",
    "dataUploadFrequency": "DataUploadFrequency"
}
```

Example 高頻度推論設定：

```
{
    "inferenceMode": "START",
    "dataDelayOffsetInMinutes": "2",
    "dataUploadFrequency": "PT5M"
}
```

低頻度推論 (2 時間 – 1 日)

このモードは、動きの遅いプロセスや、毎日の評価で十分なユースケースに適しています。お客様は、推論を 1 時間ごとまたは 1 日に 1 回実行するように設定します。

1 日間隔の開始時刻のサポート：

毎日の推論では、オプションでタイムゾーン認識とともに **startTime** (毎日午前 8 時) を指定します。

タイムゾーンのサポート：

が指定されている場合、 `startTime`は、Internet Assigned Numbers Authority (IANA) によって管理されるタイムゾーンデータベース AWS IoT SiteWise を使用します。これにより、リージョン間でも推論がローカルの勤務時間と一致するようになります。

条件付きオフセットのサポート：

他のモードと同様に、条件オフセットは 0 ~ 60 分に設定されています。

ユースケース：

- バッチプロセスまたはシフトベースのオペレーションの毎日のヘルスチェック。
- メンテナンス中またはダウンタイム中の推論を回避します。
- これは、コンピューティング使用量を最小限に抑える必要があるリソースに制約のある環境に役立ちます。

低頻度推論を設定するには：

- 値 `DataUploadFrequency`を使用して`AWS/ANOMALY_DETECTION_INFERENCE`アクションペイロード値を設定します`PT2H..PT12H`。
  - 1 日の場合、 `DataUploadFrequency`は です`P1D`。
- （オプション) 遅延オフセット`DataDelayOffsetInMinutes`を分単位で設定します。この値を 0 ~ 60 分に設定します。

Example 低頻度推論設定：

```
{
    "inferenceMode": "START",
    "dataUploadFrequency": "P1D",
    "inferenceStartTime": "13:00",
    "inferenceTimeZone": "America/Chicago"
}
```

柔軟なスケジューリング

柔軟なスケジューリングにより、お客様は推論が実行される特定の日と時間範囲を定義できます。これにより、お客様は生産時間、シフトタイミング、計画されたダウンタイムに基づいてスケジューリングを完全に制御できます。

は、次の場合`weeklyOperatingWindow`に役立ちます。

- 機器は特定の時間 (午前 8 時 ~ 午後 4 時) にのみ実行されます。
- 週末は本番稼働しません。
- 毎日のメンテナンスは、既知のタイムブロック中にスケジュールされます。

タイムゾーンのサポート：

が指定されている場合、 `startTime`は、Internet Assigned Numbers Authority (IANA) によって管理されるタイムゾーンデータベース AWS IoT SiteWise を使用します。これにより、リージョン間でも推論がローカルの勤務時間と一致するようになります。

条件付きオフセットのサポート：

他のモードと同様に、0 ~ 60 分の条件付きオフセットを設定できます。

の利点weeklyOperatingWindow：

- アイドルまたはメンテナンス期間中の推論を回避し、誤検出を減らします。
- これは、異常検出を運用上の優先順位とシフトベースのワークフローに合わせます。

柔軟なスケジューリングを設定するには：

- でAWS/ANOMALY_DETECTION_INFERENCEアクションペイロード値を設定しますDataUploadFrequency。
- （オプション）DataDelayOffsetInMinutes 遅延オフセットを分単位で指定します。この値を 0～60 分に設定します。
- シフト設定weeklyOperatingWindowを使用して を設定します。
  - のキーweeklyOperatingWindowは曜日: ですmonday|tuesday|wednesday|thursday|friday|saturday|sunday。
  - 各時間範囲は "HH:MM-HH:MM" () として 24 時間形式である必要があります"08:00-16:00"。
  - 1 日あたり複数の範囲を指定できます。

Example 柔軟なスケジューリング設定：

```
{
    "inferenceMode": "START",
    "dataUploadFrequency": "PT5M",
    "weeklyOperatingWindow": {
      "tuesday": ["11:00-13:00"],
      "monday": ["10:00-11:00", "13:00-15:00"]
    }
}
```

モデルバージョンのアクティベーション

推論を開始するときは、オプションで特定のモデルバージョンをアクティブ化して異常検出に使用できます。この機能を使用すると、特定のトレーニング済みモデルバージョンを選択したり、以前のバージョンにロールバックしたり、モデルの自動昇格の決定を上書きしたりできます。

ユースケース:

- 本番ロールバック: 現在のバージョンでパフォーマンスの低下や予期しない動作が見られると、安定したモデルバージョンにすばやく戻します。
- A/B テスト: 制御された期間中にモデルバージョンを切り替えることで、異なるモデルバージョン間のパフォーマンスを比較します。
- 手動モデル選択: 自動昇格の決定を上書きし、ビジネス要件に基づいて希望するモデルバージョンを手動で選択します。
- ステージデプロイ: 本番稼働用のフル使用に昇格する前に、重要ではない時間枠で新しいモデルバージョンをテストします。
- パフォーマンスの最適化: 特定の運用条件や季節的なパターンにより適したパフォーマンスを発揮するモデルバージョンを選択します。
- メンテナンス中のロールバック: システムメンテナンス中、またはアップグレード中に、十分にテストされた古いモデルバージョンを使用して安定性を確保します。

モデルバージョンの選択動作

`targetModelVersion` が指定されている場合：

- システムは、推論のためにリクエストされたモデルバージョンをアクティブ化します。
- 指定されたモデルバージョンが存在することを検証します。
- 自動昇格設定を上書きします。

`targetModelVersion` が指定されていない場合：

- 推論が以前に開始されている場合、最後にアクティブなモデルバージョンをアクティブ化します。
- 推論がアクティブ化されなかった場合、 は最新のトレーニング済みモデルバージョンを使用します。

特定のモデルバージョンをアクティブ化するには：

- targetModelVersion を目的のモデルバージョン番号に設定して、推論アクションペイロードを設定します。
- 指定されたモデルバージョンが存在する場合は検証され、アクティブ化されます。

Example モデルバージョンのアクティベーション：

```
{
    "inferenceMode": "START",
    "dataUploadFrequency": "PT15M",
    "targetModelVersion": 2
}
```

モデルバージョンの確認

現在アクティブなモデルバージョンを確認するには：

- レスポンスにアクティブなモデルバージョンを含む DescribeComputationModelExecutionSummary API を使用します。

使用可能なすべてのモデルバージョンを表示するには：

- ListExecutions API を使用して、過去のモデルバージョンの完全なリストを取得します。
- DescribeExecution API を使用して、エクスポートデータの時間範囲、計算モデルバージョン、請求対象期間などのトレーニング済みモデル情報を分単位で取得します。

モデルバージョンの特性

- モデルバージョン番号は 1 から順番に割り当てられます。
- 以前にトレーニングしたモデルバージョンをアクティブ化できます。
- アクティブ化されたモデルバージョンは、明示的に変更されるまで保持されます。
- モデルバージョンのアクティベーションは、すべての推論スケジューリングモード (高頻度、低頻度、柔軟性) で機能します。
- 指定されたモデルバージョンが存在しない場合、推論アクションはエラーで失敗します。

推論結果を確認する

推論結果を取得する

最新の推論結果

次のコマンドを実行して、アセットプロパティの最新の推論結果を取得します。詳細については、AWS CLI 「 コマンドリファレンスガイド」のget-asset-property-value」を参照してください。

```
aws iotsitewise get-asset-property-value \
    –asset-id asset-id \
    –property-id result-property-id
```

推論結果の履歴

次のコマンドを実行して、指定した時間枠の推論結果の履歴を取得します。詳細については、AWS CLI 「 コマンドリファレンスガイド」のget-asset-property-value-history」を参照してください。

```
aws iotsitewise get-asset-property-value-history \
    –asset-id asset-id \
    –property-id result-property-id \
    –start-date start-time \
    –end-date end-time
```

レスポンスの例

Example 推論結果レスポンスの：

```
{
  "value": {
    "stringValue": "{\"timestamp\": \"2025-02-10T22:42:00.000000\", \"prediction\": 0,
 \"prediction_reason\": \"NO_ANOMALY_DETECTED\", \"diagnostics\": [{\"name\": \"asset-
id\\\\property-id\", \"value\": 0.53528}]}"
  },
  "timestamp": {
    "timeInSeconds": 1739227320,
    "offsetInNanos": 0
  },
  "quality": "GOOD"
}
```

レスポンスフィールド

- value.stringValue – 次のフィールドを含む推論結果を含む JSON 文字列。
  - timestamp – 推論が実行される TQV のタイムスタンプ。
  - prediction – 予測結果 (異常なしの場合は 0、検出された異常の場合は 1)。
  - prediction_reason – 予測の理由 (`NO_ANOMALY_DETECTED` または `ANOMALY_DETECTED`) 。
  - diagnostics – 寄与要因を示す診断情報の配列。
- timestamp – 結果が記録されたタイムスタンプ AWS IoT SiteWise。
- quality – データポイントの品質 (通常は `GOOD`) 。

推論結果を理解する

AWS IoT SiteWise 異常検出によって返される推論結果には、異常が検出されたかどうか、異常に寄与したセンサーなど、特定のタイムスタンプでのモデルの予測に関する重要な情報が含まれます。

Example 詳細な推論結果：

```
{
    "timestamp": "2021-03-11T22:25:00.000000",
    "prediction": 1,
    "prediction_reason": "ANOMALY_DETECTED",
    "anomaly_score": 0.72385,
    "diagnostics": [
      { "name": "asset_id_1\\\\property_id_1", "value": 0.02346 },
      { "name": "asset_id_2\\\\property_id_2", "value": 0.10011 },
      { "name": "asset_id_3\\\\property_id_3", "value": 0.11162 }
    ]
}
```

`diagnostics` フィールドは、モデルが特定の予測を行う理由を解釈するのに役立ちます。リストの各エントリには以下が含まれます。

- `name`: 予測に寄与したセンサー (形式: `asset_id\\\\property_id`) 。
- `value`: その時点のセンサーの相対的な重みまたは重要度を表す 0 から 1 までの浮動小数点数。

ユーザーのメリット：

- どのセンサーが異常に最も大きな影響を与えたかを理解します。

- 高ウェイトセンサーを物理的な機器の動作と関連付けます。
- 根本原因分析を通知します。

> **Note**
>
> `prediction = 0` (通常の動作) の場合でも、診断リストが返されます。これは、正常な状態であっても、どのセンサーが現在モデルの決定に影響を与えているかを評価するのに役立ちます。

異常な動作に対してカスタムアクションをトリガーする (AWS Management Console)

AWS IoT SiteWise MQTT 通知を と組み合わせて使用することで、異常な動作に応じてカスタムアクションを有効にできますAWS IoT Core。

で MQTT 通知を設定し AWS IoT SiteWise、推論結果 AWS IoT Core に基づいて でカスタムアクションをトリガーするには、次の手順に従います。

- 推論が実行される AWS IoT SiteWiseでアセットを見つけます。
- 計算モデルの作成`resultProperty`時に として使用したプロパティを特定します。このプロパティの MQTT 通知を有効にします。
- MQTT 通知を有効にしたら、 が AWS IoT SiteWise 生成する通知トピックをコピーします。
- に移動します AWS IoT Core。MQTT テストクライアントで、コピーした通知トピックをサブスクライブして受信メッセージをモニタリングします。
- 推論結果を処理する AWS IoT Core ルールを作成します。（これにより、システムが異常を検出した場合にのみアクションがトリガーされます）。を が AWS IoT SiteWise 生成する通知トピック`notification-topic`に置き換えます。

```
SELECT * FROM "notification-topic"
  WHERE indexof(get(get(payload.values, 0).value, 'stringValue'),
 "NO_ANOMALY_DETECTED") < 0
```

が AWS IoT Core サポートするアクションのいずれかをトリガーするようにルールを設定します。でサポートされているアクションについて説明します[AWS IoT Core](#)。

ベストプラクティス

最小日付範囲を理解する

トレーニングデータ期間には最低 14 日を使用します。ただし、多くの場合、より長い期間のデータを含めることをお勧めします。

トレーニングデータセットが、アセットが通常のすべての運用モードで運用されている期間にまたがっていることを確認します。このアプローチは、予想される動作と真の異常 AWS IoT SiteWise を正確に区別するのに役立ちます。

トレーニングデータがすべての一般的な動作モードを表していない場合は、不慣れだが通常のパターンに異常として誤ってフラグを付け、誤検出が増加する AWS IoT SiteWise 可能性があります。

高頻度データとトレーニングと推論間の整合性のサンプリング

センサーが 1 Hz を超える周波数 (1 秒あたり 1 回以上の読み取り) でデータを生成する場合は、トレーニング中にサンプリングを適用します。サンプリングは、重要な傾向を維持しながらデータ量を削減します。これにより、ノイズや一時的な変動の影響を最小限に抑えることで、効率的な処理が可能になり、モデルの一般化が向上します。

AWS IoT SiteWise ネイティブ異常検出は現在、1 Hz 未満のレートで取り込まれたデータをサポートしていません。異常検出を設定する前に、データがこの最小頻度要件を満たしていることを確認します。

さらに、 は、推論のトレーニング中に設定したサンプリングレートも AWS IoT SiteWise 使用します。一貫性を維持し、正確な異常検出結果を確保するには、運用上のニーズとセンサーデータの動作の両方に合ったサンプリングレートを選択します。

でサンプリングレートを設定する方法の詳細については、「」を参照してください[サンプルレート設定](#)。

ラベル付けに関する推奨事項

効果的なモデル評価と継続的な改善には、異常の正確で一貫したラベル付けが不可欠です。異常にラベルを付けるときは、次のベストプラクティスを考慮してください。

- 関連する異常を統合する： 同じ根本的な問題の一部である場合、厳密に発生する異常を個別のイベントとしてラベル付けしないでください。例えば、異常が互いに 1~2 日以内に発生し、同じ根

本原因によって引き起こされる場合は、単一の異常ウィンドウとして扱います。このアプローチは、モデルが異常な動作のパターンをよりよく学習し、評価データのノイズを減らすのに役立ちます。

- ポイントだけでなく異常ウィンドウにラベルを付ける： 個々のデータポイントを異常としてマークする代わりに、偏差の開始から回復までの異常な動作を反映するウィンドウ全体にラベルを付けます。このアプローチにより、境界がより明確になり、実際の運用上の問題とのモデルの整合性が向上します。
- 不確実なピリオドを除外する： ピリオドが異常かどうか不明な場合は、ラベルを付けないままにします。あいまいなラベルはモデルを混乱させ、時間の経過とともに精度を低下させる可能性があります。

ラベルを追加する方法の詳細については、「」を参照してくださいデータのラベル付け。

でデータストレージを管理する AWS IoT SiteWise

データを次のストレージ階層に保存 AWS IoT SiteWise するように を設定できます。

ホット階層

ホットストレージ階層はマネージド AWS IoT SiteWise 時系列ストレージです。ホット階層は、書き込みから読み取りまでのレイテンシーが低く、頻繁にアクセスされるデータに最も効果的です。ホット階層に保存されているデータは、機器内の測定値の最新値にすばやくアクセスする必要がある産業アプリケーションで使用されます。これには、インタラクティブなダッシュボードを備え、リアルタイムメトリクスを視覚化するアプリケーションや、オペレーションをモニタリングし、アラームを起動してパフォーマンスの問題を特定するアプリケーションが含まれます。

デフォルトでは、 に取り込まれたデータはホット階層に AWS IoT SiteWise 保存されます。ホット階層の保持期間を定義できます。その後、 は設定に基づいてホット階層内のデータをウォームまたはコールド階層ストレージ AWS IoT SiteWise に移動します。最高のパフォーマンスとコスト効率のためには、ホット階層の保持期間を、データの頻繁な取得にかかる時間よりも長く設定します。これは、リアルタイムメトリクス、アラーム、およびモニタリングシナリオで使用されます。保持期間が設定されていない場合、データはホット階層に無期限に保存されます。

ウォーム階層

ウォームストレージ階層は、履歴データのコスト効率の高いストレージに効果的な AWS IoT SiteWise マネージド階層です。これは、書き込みから読み取りまでのレイテンシー特性が中程度の大量のデータを取得するのに最適です。ウォーム階層を使用して、大規模なワークロードに必要な履歴データを保存します。例えば、分析、ビジネスインテリジェンスアプリケーション (BI)、レポートツール、機械学習 (ML) モデルのトレーニングのためのデータ取得に使用されます。コールドストレージ階層を有効にすると、ウォーム階層の保持期間を定義できます。保持期間が終了すると、 はウォーム階層からデータ AWS IoT SiteWise を削除します。

コールド階層

コールドストレージ階層は、Amazon S3 バケットを使用して、ほとんど使用されないデータを保存します。コールド階層を有効にすると、 AWS IoT SiteWise は測定値、メトリクス、変換と集計、アセットモデル定義などの時系列を 6 時間ごとにレプリケートします。コールド階層は、履歴レポートとバックアップの読み取りレイテンシーが高くてもよいデータを保存するために使用されます。

トピック

- でストレージ設定を構成する AWS IoT SiteWise
- のストレージ設定のトラブルシューティング AWS IoT SiteWise
- Cold 階層に保存されたデータのファイルパスとスキーマ

でストレージ設定を構成する AWS IoT SiteWise

サービスマネージドウォーム階層ストレージにオプトインするように、またコールド階層にデータをレプリケートするように、ストレージ設定を構成できます。ウォーム階層とホット階層の保持期間の詳細については、「データ保持の影響」を参照してください。ストレージ設定を構成するときは、次の操作を行います。

- ホット階層の保持 — データが削除される前にホット階層に保存され、ストレージ設定に基づいてサービスマネージドウォーム階層ストレージまたはコールド階層ストレージに移動される期間、保持期間を設定します。 AWS IoT SiteWise は、保持期間が終了する前に存在していたホット階層内のデータをすべて削除します。保持期間を設定しなかった場合、データはホット階層に無期限に保存されます。
- ウォーム階層の保持 — AWS IoT SiteWise ストレージから削除されてカスタマー管理のコールド階層ストレージに移動する前に、ウォーム階層にデータを保存する期間を設定します。 は、保持期間が終了する前に存在していたウォーム階層からすべてのデータ AWS IoT SiteWise を削除します。保持期間が設定されていない場合、データはウォーム階層に無期限に保存されます。

> **Note**
>
> クエリのパフォーマンスを向上させるには、ウォーム階層ストレージとともにホット階層の保持期間を設定します。

ホットおよびウォーム階層ストレージにおけるデータ保持の影響

- ホット階層ストレージの保持期間を短縮すると、データはホット階層からウォーム階層またはコールド階層に完全に移動されます。ウォーム階層の保持期間を短くすると、データはコールド階層に移動され、ウォーム階層から完全に削除されます。
- ホットまたはウォーム階層ストレージの保持期間を長くすると、その変更によって AWS IoT SiteWise 、 から に送信されるデータに影響します。 AWS IoT SiteWise は、ホット階層に入力するためにウォームまたはコールドストレージからデータを取得しません。例えば、ホット階層スト

レージの保持期間が最初に 30 日間に設定され、その後 60 日間に延長された場合、ホット階層ストレージに 60 日分のデータが含まれるまでに 30 日かかります。

トピック

- ウォーム階層のストレージ設定を構成する (コンソール)
- ウォーム階層のストレージ設定を構成する (AWS CLI)
- コールド階層のストレージ設定を構成する (コンソール)
- コールド階層のストレージ設定を構成する (AWS CLI)

ウォーム階層のストレージ設定を構成する (コンソール)

次の手順では、 AWS IoT SiteWise コンソールでウォーム階層にデータをレプリケートするようにストレージ設定を構成する方法を示します。

コンソールでストレージの設定を設定するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [設定] の [ストレージ] を選択します。
3. 右上隅の[編集] を選択します。
4. [ストレージを編集] ページで、次の作業を行います:
5. [ホット階層設定] では、以下を行います。

   - ホット階層のデータの保持期間 (この期間に達するとデータは削除される) を設定する場合は、[保持期間を有効化] を選択します。
   - 保持期間を設定するには、整数を入力して単位を選択します。保持期間は、30 日以上にする必要があります。

   AWS IoT SiteWise は、保持期間より古いホット階層内のデータをすべて削除します。保持期間を設定しなかった場合、データは無期限に保存されます。
6. (推奨) [ウォーム階層設定] では、以下を実行します。

   - ウォーム階層ストレージにオプトインするには、[ウォーム階層ストレージのオプトインを確認します] を選択して、ウォーム階層ストレージにオプトインします。
   - (オプション) 保持期間を設定するには、整数を入力し、単位を選択します。保持期間は、365 日以上にする必要があります。

AWS IoT SiteWise は、保持期間より前に存在していたウォーム階層内のデータを削除します。保持期間を設定しなかった場合、データは無期限に保存されます。

> **Note**
>
> - ウォーム階層にオプトインすると、設定は 1 回のみ表示されます。
> - ホット階層の保持を設定するには、ウォーム階層ストレージまたはコールド階層ストレージが必要です。コスト効率と履歴データの取得のために、 AWS IoT SiteWise では、長期データをウォーム階層に保存することをお勧めします。
> - ウォーム階層の保持を設定するには、コールド階層ストレージが必要です。

7. 設定を保存するには、[保存] を選択します。

[AWS IoT SiteWise ストレージ] セクションでは、[ウォーム階層ストレージ] は次のいずれかの状態です。

- [有効] – ホット階層の保持期間より前にデータが存在している場合、 AWS IoT SiteWise はデータをウォーム階層に移動します。
- [無効] – ウォーム階層ストレージは無効です。

ウォーム階層のストレージ設定を構成する (AWS CLI)

AWS CLI および次のコマンドを使用して、ウォーム階層にデータを移動するようにストレージ設定を構成することができます。

既存の設定が上書きされないように、次のコマンドを実行して、現在のストレージ設定情報を取得します。

```
aws iotsitewise describe-storage-configuration
```

Example 既存のコールド階層設定のないレスポンス

```
{
          "storageType": "SITEWISE_DEFAULT_STORAGE",
          "disassociatedDataStorage": "ENABLED",
```

```
          "configurationStatus": {
              "state": "ACTIVE"
          },
          "lastUpdateDate": "2021-10-14T15:53:35-07:00",
          "warmTier": "DISABLED"
}
```

Example 既存のコールド階層設定のあるレスポンス

```
{
      "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"
}
```

でウォーム階層のストレージ設定を構成する AWS CLI

ストレージ設定を構成するには、次のコマンドを実行します。を、 AWS IoT SiteWise ストレージ設定を含むファイルの名前`file-name`に置き換えます。

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise ホット階層とウォーム階層を使用した設定

```
{
             "storageType": "SITEWISE_DEFAULT_STORAGE",
             "disassociatedDataStorage": "ENABLED",
             "warmTier": "ENABLED",
```

```
              "retentionPeriod": {
                 "numberOfDays": hot-tier-retention-in-days
               }

}
```

`hot-tier-retention-in-days` は 30 日以上の整数である必要があります。

Example レスポンス

```
{
              "storageType": "SITEWISE_DEFAULT_STORAGE",
              "configurationStatus": {
              "state": "UPDATE_IN_PROGRESS"
              }
}
```

コールド階層ストレージが有効になっている場合は、「AWS CLI および既存のコールド階層でストレージ設定を構成する」を参照してください。

AWS CLI および既存のコールド階層でストレージ設定を構成する

既存のコールド階層ストレージ AWS CLI で を使用してストレージ設定を構成する

- ストレージ設定を構成するには、次のコマンドを実行します。*file-name* を、 AWS IoT SiteWise ストレージの設定を含むファイルの名前に置き換えます。

  ```
  aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
  ```

  Example AWS IoT SiteWise ストレージ設定

  - *amzn-s3-demo-bucket* をユーザーの Amazon S3 バケット名に置き換えます。
  - *[prefix]* (プレフィックス) は Amazon S3 のプレフィックスに置き換えてください。
  - *aws-account-id* を AWS アカウント ID に置き換えます。
  - *role-name* を、 が Amazon S3 にデータを送信 AWS IoT SiteWise することを許可する Amazon S3 アクセスロールの名前に置き換えます。
  - *hot-tier-retention-in-days* を、30 日以上の整数に置き換えます。
  - *warm-tier-retention-in-days* を、365 日以上の整数に置き換えます。

> **Note**
>
> AWS IoT SiteWise は、コールド階層の保持期間より古いウォーム階層のデータをすべて削除します。保持期間を設定しなかった場合、データは無期限に保存されます。

```
{
      "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 レスポンス

```
{
      "storageType": "MULTI_LAYER_STORAGE",
      "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
       }
}
```

コールド階層のストレージ設定を構成する (コンソール)

次の手順では、 AWS IoT SiteWise コンソールでコールド階層にデータをレプリケートするようにストレージ設定を構成する方法を示します。

コンソールでストレージの設定を設定するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで [設定] の [ストレージ] を選択します。
3. 右上隅の[編集] を選択します。
4. [ストレージを編集] ページで、次の作業を行います:

   a. [ストレージ設定] で [コールド階層ストレージを有効にする] を選択します。デフォルトでは、コールド階層ストレージは無効です。

   b. S3 バケットの場所には、既存の Amazon S3 バケット名とプレフィックスを入力します。

   > **Note**
   >
   > - Amazon S3 では、プレフィックスを Amazon S3 バケット内のフォルダ名として使用します。プレフィックスは1～255文字で、末尾はスラッシュ (/) とする。お客様の AWS IoT SiteWise データはこのフォルダーに保存されます。
   > - Amazon S3 バケットがない場合は、[表示] () を選択し、Amazon S3 コンソールでバケットを作成します。詳細については、Amazon S3 ユーザーガイド の 最初の S3 バケットを作成する を参照してください。

   c. S3 アクセスロールの場合、次のいずれかを実行します。

   - AWS マネージドテンプレートからロールを作成する を選択すると、 が Amazon S3 AWS IoT SiteWise にデータを送信できるようにする IAM ロール AWS が自動的に作成されます。 Amazon S3
   - [既存のロールの使用] を選択し、リストから作成したロールを選択します。

   > **Note**
   >
   > - [S3 バケットの場所] には、前のステップで使用したのと同じ Amazon S3 バケット名をIAM ポリシーで使用する必要があります。
   > - 次の例に示すようなアクセス許可をロールが持っていることを確認してください。

Example アクセス許可ポリシー:

JSON

```
{
      "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-bucket を、ユーザーの Amazon S3 バケットの名前に置き換えます。

- Amazon S3 バケットがカスタマーマネージド KMS キーを使用して暗号化されている場合、KMS キーには `kms:Decrypt`および `kms:GenerateDataKey`オペレーションの IAM ロールを持つアクセスポリシーが必要です。

d. ホット階層を設定するには、「ウォーム階層のストレージ設定を構成する (コンソール)」の「ステップ 5」を参照してください。

e. (オプション) [AWS IoT Analytics 統合] では、以下を行います。

> **Note**
>
> サポート終了通知: 2025 年 12 月 15 日に、 AWS は のサポートを終了します AWS IoT Analytics。2025 年 12 月 15 日以降、 AWS IoT Analytics コンソールまたは

AWS IoT Analytics リソースにアクセスできなくなります。詳細については、AWS IoT Analytics 「サポート終了」を参照してください。

i. AWS IoT Analytics を使用してデータをクエリする場合は、Enabled AWS IoT Analytics data store を選択します。

ii. AWS IoT SiteWise はデータストアの名前を生成するか、別の名前を入力できます。

AWS IoT SiteWise は、データを保存 AWS IoT Analytics するために にデータストアを自動的に作成します。データをクエリするには、 AWS IoT Analytics を使用してデータセットを作成できます。詳細については、「 ユーザーガイド」の AWS IoT SiteWise 「データの使用」を参照してください。 AWS IoT Analytics

f. [保存] を選択します。

[AWS IoT SiteWise ストレージ] セクションの[コールド階層ストレージ]は、次のいずれかの値に設定できます。

- 有効 – 指定した Amazon S3 バケットにデータを AWS IoT SiteWise レプリケートします。
- Enable – AWS IoT SiteWise コールド階層ストレージを有効にするリクエストを処理しています。このプロセスは完了までに数分かかることがあります。
- Enable_Failed – コールド階層ストレージを有効にするリクエストを処理 AWS IoT SiteWise できませんでした。 AWS IoT SiteWise が Amazon CloudWatch Logs にログを送信できるようにした場合、これらのログを使用して問題のトラブルシューティングを行うことができます。詳細については、「Amazon CloudWatch Logs によるモニタリング」を参照してください。
- [無効] – コールド階層ストレージは無効です。

コールド階層のストレージ設定を構成する (AWS CLI)

以下では、 AWS CLIを使用して、ストレージの設定でコールド階層にデータが複製されるようにする手順を示します。

を使用してストレージ設定を構成するには AWS CLI

1. アカウント内の Amazon S3 バケットにデータを書き出すには、次のコマンドを実行し、ストレージの設定を行ってください。*file-name* を、 AWS IoT SiteWise ストレージ設定を含むファイルの名前に置き換えます。

```
aws iotsitewise put-storage-configuration --cli-input-json file://file-name.json
```

Example AWS IoT SiteWise ストレージ設定

- *amzn-s3-demo-bucket* をユーザーの Amazon S3 バケット名に置き換えます。
- *[prefix]* (プレフィックス) は Amazon S3 のプレフィックスに置き換えてください。
- *aws-account-id* を AWS アカウント ID に置き換えます。
- *role-name* を、 が Amazon S3 にデータを送信 AWS IoT SiteWise することを許可する Amazon S3 アクセスロールの名前に置き換えます。
- *retention-in-days* は、30 日以上の整数に置き換えます。

```
{
      "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 ストレージ設定と IAM ポリシーでは、同じ Amazon S3 バケット名を使用する必要があります。
- 次の例に示すようなアクセス許可をロールが持っていることを確認してください。

  Example アクセス許可ポリシー:

  JSON

```
{
      "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-bucket を、ユーザーの Amazon S3 バケットの名前に置き換えます。

- Amazon S3 バケットがカスタマーマネージド KMS キーを使用して暗号化されている場合、KMS キーには `kms:Decrypt`および `kms:GenerateDataKey`オペレーションの IAM ロールを持つアクセスポリシーが必要です。

Example レスポンス

```
{
    "storageType": "MULTI_LAYER_STORAGE",
    "retentionPeriod": {
        "numberOfDays": 100,
        "unlimited": false
    },
    "configurationStatus": {
        "state": "UPDATE_IN_PROGRESS"
    }
}
```

> **Note**
>
> がストレージ設定を更新する AWS IoT SiteWise までに数分かかる場合があります。

2. ストレージの設定情報を取得するには、次のコマンドを実行します。

```
aws iotsitewise describe-storage-configuration
```

Example レスポンス

```
{
      "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 バケットへのデータ書き出しを停止するには、次のコマンドを実行してストレージの設定を行ってください。

```
aws iotsitewise put-storage-configuration --storage-type SITEWISE_DEFAULT_STORAGE
```

> **Note**
>
> デフォルトでは、データは のホット階層にのみ保存されます AWS IoT SiteWise。

Example レスポンス

```
{
      "storageType": "SITEWISE_DEFAULT_STORAGE",
      "configurationStatus": {
          "state": "UPDATE_IN_PROGRESS"
      }
```

```
  }
```

4. ストレージの設定情報を取得するには、次のコマンドを実行します。

```
aws iotsitewise describe-storage-configuration
```

Example レスポンス

```
{
      "storageType": "SITEWISE_DEFAULT_STORAGE",
      "configurationStatus": {
          "state": "ACTIVE"
      },
      "lastUpdateDate": "2021-03-30T15:57:14-07:00"
  }
```

(オプション) AWS IoT Analytics データストアを作成する (AWS CLI)

> **Note**
>
> サポート終了通知: 2025 年 12 月 15 日に、 AWS は のサポートを終了します AWS IoT Analytics。2025 年 12 月 15 日以降、 AWS IoT Analytics コンソールまたは AWS IoT Analytics リソースにアクセスできなくなります。詳細については、AWS IoT Analytics 「サポート終了」を参照してください。

AWS IoT Analytics データストアは、データを受信して保存するスケーラブルでクエリ可能なリポジトリです。 AWS IoT SiteWise コンソールまたは AWS IoT Analytics APIs を使用して、 AWS IoT Analytics データを保存するための AWS IoT SiteWise データストアを作成できます。データをクエリするには、 を使用してデータセットを作成します AWS IoT Analytics。詳細については、 AWS IoT Analytics ユーザーガイドの「AWS IoT SiteWise データの使用」を参照してください。

次の手順では AWS CLI 、 を使用して にデータストアを作成します AWS IoT Analytics。

データストアを作成するには、次のコマンドを実行します。*file-name* は、データストアの設定を含むファイルの名前に置き換えてください。

```
aws iotanalytics create-datastore --cli-input-json file://file-name.json
```

> **Note**
>
> - 既存の Amazon S3 バケット名を指定する必要があります。 Amazon S3 バケットを持っていない場合は、まずバケットを作成します。詳細については、Amazon S3 ユーザーガイドの 最初の S3 バケットを作成する を参照してください。
> - AWS IoT SiteWise ストレージ設定、IAM ポリシー、 AWS IoT Analytics データストア設定では、同じ Amazon S3 バケット名を使用する必要があります。

Example AWS IoT Analytics データストアの設定

data-store-name と *amzn-s3-demo-bucket* を AWS IoT Analytics データストア名と Amazon S3 バケット名に置き換えます。

```
{
      "datastoreName": "data-store-name",
      "datastoreStorage": {
          "iotSiteWiseMultiLayerStorage": {
              "customerManagedS3Storage": {
                  "bucket": "amzn-s3-demo-bucket"
              }
          }
      },
      "retentionPeriod": {
          "numberOfDays": 90
      }
  }
```

Example レスポンス

```
{
      "datastoreName": "datastore_IoTSiteWise_demo",
      "datastoreArn": "arn:aws:iotanalytics:us-west-2:123456789012:datastore/
datastore_IoTSiteWise_demo",
      "retentionPeriod": {
          "numberOfDays": 90,
          "unlimited": false
      }
  }
```

のストレージ設定のトラブルシューティング AWS IoT SiteWise

次の情報を使用して、ストレージ設定に関する問題のトラブルシューティングと解決を行います。

問題

- エラー:バケットは存在しません
- エラー: Amazon S3 パスへのアクセスが拒否されました
- エラー:ロール ARN を引き受けることができません
- エラー: クロスリージョン Amazon S3 バケットにアクセスできませんでした

エラー:バケットは存在しません

解決策： AWS IoT SiteWise Amazon S3 バケットが見つかりませんでした。現在のRegion にある既存の Amazon S3 バケット名を入力していることを確認してください。

エラー: Amazon S3 パスへのアクセスが拒否されました

解決策： AWS IoT SiteWise Amazon S3 バケットにアクセスできませんでした。以下の操作を実行します。

- IAM ポリシーで指定した Amazon S3 バケットと同じものを使用していることを確認してください。
- 次の例に示すようなアクセス許可をロールが持っていることを確認してください。

  Example アクセス許可ポリシー

  JSON

```
{
      "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-bucket を、ユーザーの Amazon S3 バケットの名前に置き換えます。

エラー:ロール ARN を引き受けることができません

解決策： AWS IoT SiteWise ユーザーに代わって IAM ロールを引き受けることができませんでした。ロールが次のサービスを信頼することを確認してください: `iotsitewise.amazonaws.com`。詳細については、IAM ユーザーガイドの「ロールを引き受けることができない」を参照してください。

エラー: クロスリージョン Amazon S3 バケットにアクセスできませんでした

解決策： 指定した Amazon S3 バケットは別の AWS リージョンにあります。Amazon S3 バケットと AWS IoT SiteWise アセットが同じリージョンにあることを確認します。

Cold 階層に保存されたデータのファイルパスとスキーマ

AWS IoT SiteWise は、測定値、メトリクス、変換と集計、アセットとアセットモデルの定義などの時系列をレプリケートすることで、データをコールド階層に保存します。次に、Cold 階層に送られるデータのファイルパスとスキーマについて説明します。

トピック

- 機器データ (測定)
- メトリクス、変換、集計
- アセットメタデータ
- アセット階層メタデータ
- ストレージデータインデックスファイル

機器データ (測定)

AWS IoT SiteWise は、機器データ (測定値) を 6 時間に 1 回コールド階層にエクスポートします。raw データはCold 階層にApache AVRO (`.avro`) 形式で保存されます。

ファイルパス

AWS IoT SiteWise は、次のテンプレートを使用してコールド階層に機器データ (測定値) を保存します。

```
{keyPrefix}/raw/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/raw_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Amazon S3 におけるraw データのファイルパスは、すべて次の構成要素を含んでいます。

パスコンポーネント。	説明
`keyPrefix`	AWS IoT SiteWise ストレージ設定で指定した Amazon S3 プレフィックス。Amazon S3 では、プレフィックスをバケット内のフォルダ名として使用します。
`raw`	機器からの時系列データ (測定値) を格納するフォルダ。`raw` フォルダーは、プレフィックスフォルダーに保存されます。
`seriesBucket`	00 から ff の間の 16 進数です。`timeSeriesId` から導き出される数値です。このパーティションは、コールド階層への AWS IoT SiteWise 書き込み時にスループットを向上させるために使用されます。Amazon Athena を使用してクエリを実行する場合、パーティションを使用して細かい粒度のパーティショニングを行い、クエリパフォーマンスを向上させることができます。 アセットメタデータの `seriesBucket` と `timeSeriesBucket` は同じ数字です。

パスコンポーネント。	説明
`startYear`	時系列データに関連する排他的開始時刻の年号。
`startMonth`	時系列データに関連付けられている排他的開始時刻の月。
`startDay`	時系列データに関連付けられている排他的開始時刻の月日。
`fileName`	ファイル名は、アンダースコア (_) をデリミターとして使用し、次のように区切ります。 • `raw` プレフィックス。 • `timeSeriesId` 値。 • 時系列データに関連付けられた排他的開始時刻のエポックタイムスタンプ。 • データの品質。有効な値: `GOOD`、`BAD`、`UNCERTAIN` 。詳細については、[AWS IoT SiteWise API Reference] (APIリファレンス) のAssetPropertyValue を参照してください。 [Snappy]圧縮を利用して、`.avro`形式で保存されます。

Example Cold 階層の raw データへのファイルパス

```
keyPrefix/raw/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
raw_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

フィールド

Cold 階層に書き出しされる raw データのスキーマは、次のフィールドを含んでいる。

AWS IoT SiteWise は、今後追加のフィールドが導入される可能性があるため、コールド階層から raw データを読み取るシステムにスキーマ進化のサポートを実装するようお客様にアドバイスします。

Null データは、すべての値フィールドが null として表されます。ただし、 AWS IoT SiteWise APIs でクエリを実行する場合でも、お客様は正しいデータ型を受け取ります。

フィールド名	サポートされている型	デフォルトの 型	説明
`seriesId`	`string`	該当なし	機器からの時系列データ (測定値) を識別するための ID。このフィールドを使用して、クエリで raw データとアセットメタデータを結合することができます。
`timeInSeconds`	`long`	該当なし	Unix エポック形式のタイムスタンプの日付 (秒単位)。分数ナノ秒のデータは、`offsetInNanos` で提供されます。
`offsetInNanos`	`long`	該当なし	`timeInSeconds` からのナノ秒のオフセット。
`quality`	`string`	該当なし	時系列値の品質。
`doubleValue`	`double`、または `null`	`null`	ダブル型 (浮動小数点数) の時系列データ。
`stringValue`	`string`、または `null`	`null`	文字列型 (文字の並び) の時系列データ。

フィールド名	サポートされている型	デフォルトの 型	説明
integerValue	int、または null	null	整数型 (整数) の時系列データ。
booleanValue	boolean、または null	null	ブール型 (true または false) の時系列データ。
jsonValue	string、または null	null	JSON 型の時系列データ (複雑なデータ型を文字列として保存されたもの)。
recordVersion	long、または null	null	レコードのバージョン番号。バージョン番号を使って、最新のレコードを選択することができます。新しいレコードほどバージョン番号が大きくなっています。

Example Cold 階層の raw データ


```
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675887,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.75},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675889,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.69},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675890,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.66},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675891,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.92},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
```

```
  {"seriesId":"e9687d2a-0dbe-4f65-9ed6-6f443cba41f7_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeInSeconds":1625675892,"offsetInNanos":0,"quality":"GOOD","doubleValue":
{"double":0.73},"stringValue":null,"integerValue":null,"booleanValue":null,"jsonValue":null,"re
```

メトリクス、変換、集計

AWS IoT SiteWise は、メトリクス、変換、集計を 6 時間に 1 回コールド階層にエクスポートします。メトリクス、変換、集計は、コールド階層にApache Avro (`.avro`) の形式で保存されます。

ファイルパス

AWS IoT SiteWise は、次のテンプレートを使用して、メトリクス、変換、集計をコールド階層に保存します。

```
{keyPrefix}/agg/startYear={startYear}/startMonth={startMonth}/startDay={startDay}/
seriesBucket={seriesBucket}/agg_{timeseriesId}_{startTimestamp}_{quality}.avro
```

Amazon S3 上のメトリクスへ、変換、集計のファイルパスには、次の要素が含まれます。

パスコンポーネント。	説明
`keyPrefix`	AWS IoT SiteWise ストレージ設定で指定した Amazon S3 プレフィックス。Amazon S3 では、プレフィックスをバケット内のフォルダ名として使用します。
`agg`	メトリクスの時系列データを格納するフォルダーです。agg フォルダーは、プレフィックスフォルダーに保存されます。
`seriesBucket`	00 から ff の間の 16 進数です。`timeSeriesId` から導き出される数値です。このパーティションは、コールド階層への AWS IoT SiteWise 書き込み時にスループットを向上させるために使用されます。Amazon Athena を使用してクエリを実行する場合、パーティションを使用して細かい粒度のパーティショニングを行い、クエリパフォーマンスを向上させることができます。

パスコンポーネント。	説明
	アセットメタデータの seriesBucket と timeSeriesBucket は同じ数字です。
startYear	時系列データに関連する排他的開始時刻の年号。
startMonth	時系列データに関連付けられている排他的開始時刻の月。
startDay	時系列データに関連付けられている排他的開始時刻の月日。
fileName	ファイル名は、アンダースコア (_) をデリミターとして使用し、次のように区切ります。 • raw プレフィックス。 • timeSeriesId 値。 • 時系列データに関連付けられた排他的開始時刻のエポックタイムスタンプ。 • データの品質。有効な値: GOOD、BAD、UNCERTAIN 。詳細については、[AWS IoT SiteWise API Reference] (API リファレンス) のAssetPropertyValue を参照してください。 Snappy圧縮を利用して、.avro形式で保存されます。

Example Cold 階層のメトリクスへのファイルパス

```
keyPrefix/agg/startYear=2021/startMonth=1/startDay=2/seriesBucket=a2/
agg_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1609577700_GOOD.avro
```

フィールド

コールド階層に書き出されるメトリクス、変換、集計のスキーマには、次のフィールドが含まれます。

フィールド名	サポートされている型	デフォルトの 型	説明
`seriesId`	`string`	該当なし	機器、メトリクス、変換から時系列データを識別するための ID です。このフィールドを使用して、クエリで raw データとアセットメタデータを結合することができます。
`timeInSeconds`	`long`	該当なし	Unix エポック形式のタイムスタンプの日付 (秒単位)。分数ナノ秒のデータは、`offsetInNanos` で提供されます。
`offsetInNanos`	`long`	該当なし	`timeInSeconds` からのナノ秒のオフセット。
`quality`	`string`	該当なし	アセットデータをフィルタリングするための品質です。
`resolution`	`string`	該当なし	データを集計する時間間隔。

フィールド名	サポートされている型	デフォルトの 型	説明
count	double、または null	null	現在の時間間隔における、指定された変数のデータポイントの総数。
average	double、または null	null	現在の時間間隔における指定された変数値の平均。
min	double、または null	null	現在の時間間隔における指定された変数の最小値。
max	boolean、または null	null	現在の時間間隔における指定された変数の最大値。
sum	string、または null	null	現在の時間間隔における指定された変数値の合計。
recordVersion	long、または null	null	レコードのバージョン番号。バージョン番号を使って、最新のレコードを選択することができます。新しいレコードほどバージョン番号が大きくなっています。

Example コールド階層のメトリックデータ

```
{"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334060,"offsetInNanos":0,"quality":"GOOD","resolution":"
```

```
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334120,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":46.0},"min":{"double":32.0},"max":{"double":60.0},"sum":
{"double":1334.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334540,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637334600,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":46.0},"min":{"double":32.0},"max":{"double":60.0},"sum":
{"double":1334.0},"recordVersion":null}
  {"seriesId":"f74c2828-5317-4df3-
ba16-6d41b5bcb531","timeInSeconds":1637335020,"offsetInNanos":0,"quality":"GOOD","resolution":"
{"double":16.0},"min":{"double":1.0},"max":{"double":31.0},"sum":
{"double":496.0},"recordVersion":null}
```

アセットメタデータ

が初めてコールド階層 AWS IoT SiteWise にデータをエクスポートできるようにすると、アセットメタデータはコールド階層にエクスポートされます。初期設定後、 はアセットモデル定義またはアセット定義を変更する場合にのみ、アセットメタデータを階層に AWS IoT SiteWise エクスポートします。アセットメタデータは、コールド階層に改行区切り JSON (`.ndjson`) 形式で保存されます。

ファイルパス

AWS IoT SiteWise は、次のテンプレートを使用してアセットメタデータをコールド階層に保存します。

```
{keyPrefix}/asset_metadata/asset_{assetId}.ndjson
```

Cold 階層におけるアセットメタデータのファイルパスを生成するために、Nは次のテンプレートを使用する。

パスコンポーネント。	説明
`keyPrefix`	のストレージ設定で指定した Amazon S3 AWS IoT SiteWiseプレフィックス。 Amazon S3 で

パスコンポーネント。	説明
	は、プレフィックスをバケット内のフォルダ名として使用します。
asset_metadata	アセットのメタデータを保存するフォルダです。asset_metadata フォルダーは、プレフィックスフォルダーに保存されます。
fileName	ファイル名は、アンダースコア (_) をデリミターとして使用し、次のように区切ります。 • asset プレフィックス。 • assetId 値。 ファイルは .ndjson 形式で保存されます。

Example Cold 階層のアセットメタデータへのファイルパス

```
keyPrefix/asset_metadata/asset_35901915-d476-4dca-8637-d9ed4df939ed.ndjson
```

フィールド

Cold 階層に書き出しされるアセットメタデータのスキーマは、次のフィールドを含んでいます。

フィールド名	説明
assetId	アセットの ID。
assetName	アセットの名前。
assetExternalId	アセットの外部 ID。
assetModelId	このアセットを作成するために使用されたアセットモデルの ID。
assetModelName	アセットモデルの名前。
assetModelExternalId	アセットモデルの外部 ID。

フィールド名	説明
assetPropertyId	アセットプロパティの ID。
assetPropertyName	アセットプロパティの名前。
assetPropertyExternalId	アセットプロパティの外部 ID。
assetPropertyDataType	アセットプロパティのデータ型。
assetPropertyUnit	アセットプロパティの単位 (例:NewtonsそしてRPM)。
assetPropertyAlias	OPC UA サーバーのデータストリームパス (例えば、/company/windfarm/3/turbine/7/temperature) など、アセットプロパティを識別するためのエイリアス。
timeSeriesId	機器、メトリクス、変換から時系列データを識別するための ID です。このフィールドを使用して、クエリで raw データとアセットメタデータを結合することができます。
timeSeriesBucket	00 から ff の間の 16 進数です。timeSeriesId から導き出される数値です。このパーティションは、コールド階層への AWS IoT SiteWise 書き込み時にスループットを向上させるために使用されます。Amazon Athena を使用してクエリを実行する場合、パーティションを使用して細かい粒度のパーティショニングを行い、クエリパフォーマンスを向上させることができます。 raw データのファイルパスの timeSeriesBucket と seriesBucket が同じ数字です。
assetCompositeModelId	複合モデルの ID。

フィールド名	説明
assetCompositeModelExternalId	複合モデルの外部 ID。
assetCompositeModelDescription	複合モデルの説明。
assetCompositeModelName	複合モデルの名前。
assetCompositeModelType	複合モデルのタイプ。アラーム複合モデルの場合、この型は AWS/ALARM　です。
assetCreationDate	アセットが作成された日付 (Unix エポックタイム)。
assetLastUpdateDate	アセットが最後に更新された日付を Unix エポックタイムで表したもの。
assetStatusErrorCode	エラーコードです。
assetStatusErrorMessage	エラーメッセージです。
assetStatusState	アセットの現在の状態。

Example Cold 階層のアセットメタデータ


```
  {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind
 Turbine Asset Model","assetPropertyId":"95e63da7-d34e-43e1-
bc6f-1b490154b07a","assetPropertyExternalId":null,"assetPropertyName":"Temperature","assetPrope
Washington/Seattle/WT2/temp","timeSeriesId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a","timeSeriesBucket":"f6","assetArn":null,"assetCompositeModelDescription":nul
  {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind Turbine Asset
 Model","assetPropertyId":"c706d54d-4c11-42dc-9a01-63662fc697b4","assetPropertyExternalId":null
```

```
Washington/Seattle/WT2/pressure","timeSeriesId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d_c706d54d-4c11-42dc-9a01-63662fc697b4","timeSeriesBucket":"1e","assetArn":null,"ass
  {"assetId":"7020c8e2-e6db-40fa-9845-
ed0dddd4c77d","assetExternalId":null,"assetName":"Wind Turbine Asset
 2","assetModelId":"ec1d924f-f07d-444f-b072-
e2994c165d35","assetModelExternalId":null,"assetModelName":"Wind
 Turbine Asset Model","assetPropertyId":"8cf1162f-dead-4fbe-b468-
c8e24cde9f50","assetPropertyExternalId":null,"assetPropertyName":"Max
 Temperature","assetPropertyDataType":"DOUBLE","assetPropertyUnit":null,"assetPropertyAlias":nu
e6db-40fa-9845-ed0dddd4c77d_8cf1162f-dead-4fbe-b468-
c8e24cde9f50","timeSeriesBucket":"d7","assetArn":null,"assetCompositeModelDescription":null,"as

 {"assetId":"3a5f2a22-3b37-4332-9c1c-404ea1d73fab","assetExternalId":null,"assetName":"BatchAss
ebc75e75e827","assetModelExternalId":null,"assetModelName":"FlashTestAssetModelDouble","assetPr
b410-
ab401a9176ed","assetPropertyExternalId":null,"assetPropertyName":"measurementProperty","assetPr
ae89-
ff316f5ff8aa","timeSeriesBucket":"af","assetArn":null,"assetCompositeModelDescription":null,"as
```


アセット階層メタデータ

が AWS IoT SiteWise をコールド階層に初めてデータを保存すると、アセット階層メタデータがコールド階層にエクスポートされます。初期設定後、 はアセットモデルまたはアセット定義を変更する場合にのみ、アセット階層メタデータをコールド階層に AWS IoT SiteWise エクスポートします。アセット階層メタデータは、コールド階層に改行区切り JSON (`.ndjson`) 形式で保存されます。

階層、ターゲットアセット、またはソースアセットの外部識別子は、DescribeAsset API を呼び出すことで取得されます。

ファイルパス

AWS IoT SiteWise は、次のテンプレートを使用してアセット階層メタデータをコールド階層に保存します。

```
{keyPrefix}/asset_hierarchy_metadata/{parentAssetId}_{hierarchyId}.ndjson
```

Cold 階層のアセット階層メタデータの各ファイルパスは、次のコンポーネントを含んでいます。

パスコンポーネント。	説明
keyPrefix	AWS IoT SiteWise ストレージ設定で指定した Amazon S3 プレフィックス。Amazon S3 では、プレフィックスをバケット内のフォルダ名として使用します。
asset_hierarchy_metadata	アセット階層のメタデータを保存するフォルダです。asset_hierarchy_metadata フォルダーは、プレフィックスフォルダーに保存されます。
fileName	ファイル名は、アンダースコア (_) をデリミターとして使用し、次のように区切ります。 • parentAssetId 値。 • hierarchyId 値。 ファイルは .ndjson 形式で保存されます。

Example Cold 階層におけるアセット階層メタデータへのファイルパス

```
keyPrefix/asset_hierarchy_metadata/35901915-d476-4dca-8637-
d9ed4df939ed_c5b3ced8-589a-48c7-9998-cdccfc9747a0.ndjson
```

フィールド

Cold 階層にに書き出しされるアセット階層メタデータのスキーマは、次のフィールドを含んでいます。

フィールド名	説明
sourceAssetId	このアセットリレーションシップのソースアセットの ID。
targetAssetId	このアセットリレーションシップのターゲットアセットの ID。

フィールド名	説明
hierarchyId	階層の ID。
associationType	このアセットリレーションシップの関連付け型。 値は CHILD でなければならない。ターゲットアセットは、ソースアセットのアセットです。

Example Cold 階層におけるアセット階層メタデータ

```
{"sourceAssetId":"80388e72-2284-44fb-9c89-
bfbaf0dfedd2","targetAssetId":"2b866c25-0c74-4750-bdf5-
b73683c8a2a2","hierarchyId":"bbed9f59-0412-4585-
a61d-6044db526aee","associationType":"CHILD"}
  {"sourceAssetId":"80388e72-2284-44fb-9c89-
bfbaf0dfedd2","targetAssetId":"6b51246e-984d-460d-
bc0b-470ea47d1e31","hierarchyId":"bbed9f59-0412-4585-
a61d-6044db526aee","associationType":"CHILD"}
```

Cold 階層でデータを表示するには

1. [Amazon S3 console] (Amazon S3 のコンソール) に移動します。
2. ナビゲーションペインで、[バケット] を選択し、Amazon S3 バケットを選択します。
3. raw データ、アセットメタデータ、またはアセット階層メタデータが含まれるフォルダに移動します。
4. ファイルを選択し、[アクション] から[ダウンロード] を選択します。

ストレージデータインデックスファイル

AWS IoT SiteWise は、これらのファイルを使用してデータクエリのパフォーマンスを最適化します。これらは Amazon S3 バケットに表示されますが、使用する必要はありません。

ファイルパス

AWS IoT SiteWise は、次のテンプレートを使用してデータインデックスファイルをコールド階層に保存します。

```
keyPrefix/index/series=timeseriesId/startYear=startYear/startMonth=startMonth/
startDay=startDay/index_timeseriesId_startTimestamp_quality
```

Example データストレージインデックスファイルへのファイルパス

```
keyPrefix/index/series=7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-
d34e-43e1-bc6f-1b490154b07a/startYear=2022/startMonth=02/startDay=03/
index_7020c8e2-e6db-40fa-9845-ed0dddd4c77d_95e63da7-d34e-43e1-
bc6f-1b490154b07a_1643846400_GOOD
```

AWS SDKs AWS IoT SiteWise を使用するためのコード例

次のコード例は、 AWS Software Development Kit (SDK) AWS IoT SiteWise で を使用する方法を示しています。

基本は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

開始方法

こんにち AWS IoT SiteWiseは

次のコード例は、 AWS IoT SiteWiseの使用を開始する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のListAssetModels」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のListAssetModels」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスのListAssetModels」を参照してください。

コードの例

- AWS SDKs AWS IoT SiteWise を使用するための基本的な例
  - こんにち AWS IoT SiteWiseは
  - AWS SDK AWS IoT SiteWise を使用した の基本について説明します。
  - AWS SDKs AWS IoT SiteWise を使用するためのアクション

- AWS SDK または CLI BatchPutAssetPropertyValueで を使用する
- AWS SDK または CLI CreateAssetで を使用する
- AWS SDK または CLI CreateAssetModelで を使用する
- AWS SDK または CLI CreateGatewayで を使用する
- AWS SDK または CLI CreatePortalで を使用する
- AWS SDK または CLI DeleteAssetで を使用する
- AWS SDK または CLI DeleteAssetModelで を使用する
- AWS SDK または CLI DeleteGatewayで を使用する
- AWS SDK または CLI DeletePortalで を使用する
- AWS SDK または CLI DescribeAssetModelで を使用する
- AWS SDK または CLI DescribeGatewayで を使用する
- AWS SDK または CLI DescribePortalで を使用する
- AWS SDK または CLI GetAssetPropertyValueで を使用する
- AWS SDK または CLI ListAssetModelsで を使用する

AWS SDKs AWS IoT SiteWise を使用するための基本的な例

次のコード例は、 SDKs AWS IoT SiteWise で AWS の基本を使用する方法を示しています。

例

- こんにち AWS IoT SiteWiseは
- AWS SDK AWS IoT SiteWise を使用した の基本について説明します。
- AWS SDKs AWS IoT SiteWise を使用するためのアクション
  - AWS SDK または CLI BatchPutAssetPropertyValueで を使用する
  - AWS SDK または CLI CreateAssetで を使用する
  - AWS SDK または CLI CreateAssetModelで を使用する
  - AWS SDK または CLI CreateGatewayで を使用する
  - AWS SDK または CLI CreatePortalで を使用する
  - AWS SDK または CLI DeleteAssetで を使用する
  - AWS SDK または CLI DeleteAssetModelで を使用する
  - AWS SDK または CLI DeleteGatewayで を使用する

- AWS SDK または CLI DeletePortalで を使用する
- AWS SDK または CLI DescribeAssetModelで を使用する
- AWS SDK または CLI DescribeGatewayで を使用する
- AWS SDK または CLI DescribePortalで を使用する
- AWS SDK または CLI GetAssetPropertyValueで を使用する
- AWS SDK または CLI ListAssetModelsで を使用する

こんにち AWS IoT SiteWiseは

次のコード例は、 AWS IoT SiteWiseの使用を開始する方法を示しています。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のListAssetModels」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のListAssetModels」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスのListAssetModels」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK AWS IoT SiteWise を使用した の基本について説明します。

次のコード例は、以下を実行する方法を示しています。

- AWS IoT SiteWise アセットモデルを作成します。
- AWS IoT SiteWise アセットを作成します。
- プロパティ ID 値を取得します。
- AWS IoT SiteWise アセットにデータを送信します。

- AWS IoT SiteWise Asset プロパティの値を取得します。
- AWS IoT SiteWise ポータルを作成します。
- AWS IoT SiteWise ゲートウェイを作成します。
- AWS IoT SiteWise ゲートウェイを記述します。
- AWS IoT SiteWise アセットを削除します。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

AWS IoT SiteWise 機能を示すインタラクティブなシナリオを実行します。

```
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 メソッドのラッパークラス。

```
public class SitewiseActions {
```

```
   private static final Logger logger =
LoggerFactory.getLogger(SitewiseActions.class);

   private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

   private static IoTSiteWiseAsyncClient getAsyncClient() {
       if (ioTSiteWiseAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryStrategy(RetryMode.STANDARD)
               .build();

           ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return ioTSiteWiseAsyncClient;
   }


   /**
    * Creates an asset model.
    *
    * @param name the name of the asset model to create.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
```

```
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse>
createAssetModelAsync(String name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
           .build();

       return getAsyncClient().createAssetModel(createAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
               }
           });
   }
```

```
   /**
    * Creates an asset with the specified name and asset model Id.
    *
    * @param assetName    the name of the asset to create.
    * @param assetModelId the Id of the asset model to associate with the asset.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<CreateAssetResponse> createAssetAsync(String
assetName, String assetModelId) {
       CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
           .assetModelId(assetModelId)
           .assetDescription("Created using the AWS SDK for Java")
           .assetName(assetName)
           .build();

       return getAsyncClient().createAsset(createAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
               }
           });
   }

   /**
    * Sends data to the SiteWise service.
    *
    * @param assetId        the ID of the asset to which the data will be sent.
    * @param tempPropertyId the ID of the temperature property.
    * @param humidityPropId the ID of the humidity property.
```

```
    * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
    *         calling code can attach callbacks, then handle the result or
exception by calling
    *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();

       TimeInNanos time = TimeInNanos.builder()
           .timeInSeconds(timestamp / 1000)
           .offsetInNanos((int) ((timestamp % 1000) * 1000000))
           .build();

       BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
           .entries(Arrays.asList(
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-3")
                   .assetId(assetId)
                   .propertyId(tempPropertyId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Temperature"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
                   .build(),
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-4")
                   .assetId(assetId)
```

```
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Fetches the value of an asset property.
     *
     * @param propId  the ID of the asset property to fetch.
     * @param assetId the ID of the asset to fetch the property value for.
     * @return a {@link CompletableFuture} that represents a {@link Double}
result. The calling code can attach
     *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
        GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
```

```
                .propertyId(propId)
                .assetId(assetId)
                .build();

        return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
                .handle((response, exception) -> {
                    if (exception != null) {
                        logger.error("Error occurred while fetching property
value: {}.", exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    return response.propertyValue().value().doubleValue();
                });
    }

    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result
that associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
```

```
                } else {
                    logger.error("Error occurred while fetching property IDs:
{}.", throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }

    /**
     * Deletes an asset.
     *
     * @param assetId the ID of the asset to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String
assetId) {
        DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
            .assetId(assetId)
            .build();

        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
                }
            });
    }

    /**
     * Deletes an Asset Model with the specified ID.
     *
     * @param assetModelId the ID of the Asset Model to delete.
```

```
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse>
deleteAssetModelAsync(String assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
   }

   /**
    * Creates a new IoT SiteWise portal.
    *
    * @param portalName   the name of the portal to create.
    * @param iamRole      the IAM role ARN to use for the portal.
    * @param contactEmail the email address of the portal contact.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the portal ID. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
```

```
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> createPortalAsync(String portalName, String
iamRole, String contactEmail) {
       CreatePortalRequest createPortalRequest = CreatePortalRequest.builder()
           .portalName(portalName)
           .portalDescription("This is my custom IoT SiteWise portal.")
           .portalContactEmail(contactEmail)
           .roleArn(iamRole)
           .build();

       return getAsyncClient().createPortal(createPortalRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create portal: {} ",
exception.getCause().getMessage());
                   throw (CompletionException) exception;
               }
               return response.portalId();
           });
   }

   /**
    * Deletes a portal.
    *
    * @param portalId the ID of the portal to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeletePortalResponse}. The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<DeletePortalResponse> deletePortalAsync(String
portalId) {
```

```
        DeletePortalRequest deletePortalRequest = DeletePortalRequest.builder()
            .portalId(portalId)
            .build();

        return getAsyncClient().deletePortal(deletePortalRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete portal with ID: {}. Error:
{}", portalId, exception.getCause().getMessage());
                }
            });
    }

    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
     * @return a {@link CompletableFuture} that represents a {@link String}
result of the asset model ID or null if the
     *         asset model cannot be found. The calling code can attach
callbacks, then handle the result or exception
     *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<String> getAssetModelIdAsync(String assetModelName)
{
        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
```

```
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
                    return null;
                });
    }

    /**
     * Retrieves a portal's description.
     *
     * @param portalId the ID of the portal to describe.
     * @return a {@link CompletableFuture} that represents a {@link String}
result of the portal's start URL
     *         (see: {@link DescribePortalResponse#portalStartUrl()}). The
calling code can attach callbacks, then handle the
     *         result or exception by calling {@link CompletableFuture#join()} or
{@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<String> describePortalAsync(String portalId) {
        DescribePortalRequest request = DescribePortalRequest.builder()
            .portalId(portalId)
            .build();

        return getAsyncClient().describePortal(request)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred retrieving the portal
description: {}", exception.getCause().getMessage());
                    throw (CompletionException) exception;
                }
                return response.portalStartUrl();
            });
    }


    /**
```

```
    * Creates a new IoT Sitewise gateway.
    *
    * @param gatewayName The name of the gateway to create.
    * @param myThing     The name of the core device thing to associate with the
gateway.
    * @return a {@link CompletableFuture} that represents a {@link String}
result of the gateways ID. The calling code
    *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}.
By calling
    *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
    */
   public CompletableFuture<String> createGatewayAsync(String gatewayName,
String myThing) {
       GreengrassV2 gg = GreengrassV2.builder()
           .coreDeviceThingName(myThing)
           .build();

       GatewayPlatform platform = GatewayPlatform.builder()
           .greengrassV2(gg)
           .build();

       Map<String, String> tag = new HashMap<>();
       tag.put("Environment", "Production");

       CreateGatewayRequest createGatewayRequest =
CreateGatewayRequest.builder()
           .gatewayName(gatewayName)
           .gatewayPlatform(platform)
           .tags(tag)
           .build();

       return getAsyncClient().createGateway(createGatewayRequest)
           .handle((response, exception) -> {
               if (exception != null) {
                   logger.error("Error creating the gateway.");
                   throw (CompletionException) exception;
               }
```

```
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }

    /**
     * Deletes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
     *         can attach callbacks, then handle the result or exception by
calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
        DeleteGatewayRequest deleteGatewayRequest =
DeleteGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().deleteGateway(deleteGatewayRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
```

```
     * @return a {@link CompletableFuture} that represents a {@link
 DescribeGatewayResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by
 calling {@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
 method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
 By calling
     *         {@link CompletionException#getCause()}, the calling code can
 access the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
 gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
 method: {}", exception.getCause().getMessage());
                }
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}
```

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
  //} from "@aws-doc-sdk-examples/lib/scenario/index.js";
} from "../../libs/scenario/index.js";
import {
  IoTSiteWiseClient,
  CreateAssetModelCommand,
  CreateAssetCommand,
  ListAssetModelPropertiesCommand,
  BatchPutAssetPropertyValueCommand,
  GetAssetPropertyValueCommand,
  CreatePortalCommand,
  DescribePortalCommand,
  CreateGatewayCommand,
  DescribeGatewayCommand,
  DeletePortalCommand,
  DeleteGatewayCommand,
  DeleteAssetCommand,
  DeleteAssetModelCommand,
  DescribeAssetModelCommand,
} from "@aws-sdk/client-iotsitewise";
import {
  CloudFormationClient,
  CreateStackCommand,
  DeleteStackCommand,
  DescribeStacksCommand,
  waitUntilStackExists,
  waitUntilStackCreateComplete,
  waitUntilStackDeleteComplete,
```

```
} from "@aws-sdk/client-cloudformation";
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { parseArgs } from "node:util";
import { readFileSync } from "node:fs";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

const __filename = fileURLToPath(import.meta.url);
const __dirname = dirname(__filename);
const stackName = "SiteWiseBasicsStack";

/**
 * @typedef {{
 *   iotSiteWiseClient: import('@aws-sdk/client-iotsitewise').IotSiteWiseClient,
 *   cloudFormationClient: import('@aws-sdk/client-
cloudformation').CloudFormationClient,
 *   stackName,
 *   stack,
 *   askToDeleteResources: true,
 *   asset: {assetName: "MyAsset1"},
 *   assetModel: {assetModelName: "MyAssetModel1"},
 *   portal: {portalName: "MyPortal1"},
 *   gateway: {gatewayName: "MyGateway1"},
 *   propertyIds: [],
 *   contactEmail: "user@mydomain.com",
 *   thing: "MyThing1",
 *   sampleData: { temperature: 23.5, humidity: 65.0}
 * }} State
 */

/**
 * Used repeatedly to have the user press enter.
 * @type {ScenarioInput}
 */
const pressEnter = new ScenarioInput("continue", "Press Enter to continue", {
  type: "confirm",
});

const greet = new ScenarioOutput(
  "greet",
  `AWS IoT SiteWise is a fully managed industrial software-as-a-service (SaaS)
 that makes it easy to collect, store, organize, and monitor data from industrial
 equipment and processes. It is designed to help industrial and manufacturing
```

```
 organizations collect data from their equipment and processes, and use that data
 to make informed decisions about their operations.
One of the key features of AWS IoT SiteWise is its ability to connect to a
 wide range of industrial equipment and systems, including programmable logic
 controllers (PLCs), sensors, and other industrial devices. It can collect data
 from these devices and organize it into a unified data model, making it easier
 to analyze and gain insights from the data. AWS IoT SiteWise also provides tools
 for visualizing the data, setting up alarms and alerts, and generating reports.
Another key feature of AWS IoT SiteWise is its ability to scale to handle large
 volumes of data. It can collect and store data from thousands of devices and
 process millions of data points per second, making it suitable for large-scale
 industrial operations. Additionally, AWS IoT SiteWise is designed to be secure
 and compliant, with features like role-based access controls, data encryption,
 and integration with other AWS services for additional security and compliance
 features.

Let's get started...`,
  { header: true },
);

const displayBuildCloudFormationStack = new ScenarioOutput(
  "displayBuildCloudFormationStack",
  "This scenario uses AWS CloudFormation to create an IAM role that is required
 for this scenario. The stack will now be deployed.",
);

const sdkBuildCloudFormationStack = new ScenarioAction(
  "sdkBuildCloudFormationStack",
  async (/** @type {State} */ state) => {
    try {
      const data = readFileSync(
        `${__dirname}/../../../../resources/cfn/iotsitewise_basics/SitewiseRoles-
template.yml`,
        "utf8",
      );
      await state.cloudFormationClient.send(
        new CreateStackCommand({
          StackName: stackName,
          TemplateBody: data,
          Capabilities: ["CAPABILITY_IAM"],
        }),
      );
      await waitUntilStackExists(
        { client: state.cloudFormationClient },
```

```
        { StackName: stackName },
      );
      await waitUntilStackCreateComplete(
        { client: state.cloudFormationClient },
        { StackName: stackName },
      );
      const stack = await state.cloudFormationClient.send(
        new DescribeStacksCommand({
          StackName: stackName,
        }),
      );
      state.stack = stack.Stacks[0].Outputs[0];
      console.log(`The ARN of the IAM role is ${state.stack.OutputValue}`);
    } catch (caught) {
      console.error(caught.message);
      throw caught;
    }
  },
);

const displayCreateAWSSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSSiteWiseAssetModel",
  `1. Create an AWS SiteWise Asset Model
An AWS IoT SiteWise Asset Model is a way to represent the physical assets, such
 as equipment, processes, and systems, that exist in an industrial environment.
 This model provides a structured and hierarchical representation of these
 assets, allowing users to define the relationships and properties of each asset.

This scenario creates two asset model properties: temperature and humidity.`,
);

const sdkCreateAWSSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    let assetModelResponse;
    try {
      assetModelResponse = await state.iotSiteWiseClient.send(
        new CreateAssetModelCommand({
          assetModelName: state.assetModel.assetModelName,
          assetModelProperties: [
            {
              name: "Temperature",
              dataType: "DOUBLE",
              type: {
```

```
                measurement: {},
              },
            },
            {
              name: "Humidity",
              dataType: "DOUBLE",
              type: {
                measurement: {},
              },
            },
          ],
        }),
      );
      state.assetModel.assetModelId = assetModelResponse.assetModelId;
      console.log(
        `Asset Model successfully created. Asset Model ID:
 ${state.assetModel.assetModelId}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceAlreadyExistsException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} already exists.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateAWSIoTSiteWiseAssetModel = new ScenarioOutput(
  "displayCreateAWSIoTSiteWiseAssetModel",
  `2. Create an AWS IoT SiteWise Asset
The IoT SiteWise model that we just created defines the structure and metadata
 for your physical assets. Now we create an asset from the asset model.

Let's wait 30 seconds for the asset to be ready.`,
);

const waitThirtySeconds = new ScenarioAction("waitThirtySeconds", async () => {
  await wait(30); // wait 30 seconds
  console.log("Time's up! Let's check the asset's status.");
});
```

```
const sdkCreateAWSIoTSiteWiseAssetModel = new ScenarioAction(
  "sdkCreateAWSIoTSiteWiseAssetModel",
  async (/** @type {State} */ state) => {
    try {
      const assetResponse = await state.iotSiteWiseClient.send(
        new CreateAssetCommand({
          assetModelId: state.assetModel.assetModelId,
          assetName: state.asset.assetName,
        }),
      );
      state.asset.assetId = assetResponse.assetId;
      console.log(`Asset created with ID: ${state.asset.assetId}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset ${state.assetModel.assetModelName} was not found.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrievePropertyId = new ScenarioOutput(
  "displayRetrievePropertyId",
  `3. Retrieve the property ID values

To send data to an asset, we need to get the property ID values. In this
 scenario, we access the temperature and humidity property ID values.`,
);

const sdkRetrievePropertyId = new ScenarioAction(
  "sdkRetrievePropertyId",
  async (state) => {
    try {
      const retrieveResponse = await state.iotSiteWiseClient.send(
        new ListAssetModelPropertiesCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
```

```
      for (const retrieveResponseKey in
 retrieveResponse.assetModelPropertySummaries) {
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Humidity"
        ) {
          state.propertyIds.Humidity =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
        if (
          retrieveResponse.assetModelPropertySummaries[retrieveResponseKey]
            .name === "Temperature"
        ) {
          state.propertyIds.Temperature =
            retrieveResponse.assetModelPropertySummaries[
              retrieveResponseKey
            ].id;
        }
      }
      console.log(`The Humidity propertyId is ${state.propertyIds.Humidity}`);
      console.log(
        `The Temperature propertyId is ${state.propertyIds.Temperature}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem retrieving the properties: ${caught.message}`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displaySendDataToIoTSiteWiseAsset = new ScenarioOutput(
  "displaySendDataToIoTSiteWiseAsset",
  `4. Send data to an AWS IoT SiteWise Asset

By sending data to an IoT SiteWise Asset, you can aggregate data from multiple
 sources, normalize the data into a standard format, and store it in a
```

```
 centralized location. This makes it easier to analyze and gain insights from the
 data.

In this example, we generate sample temperature and humidity data and send it to
 the AWS IoT SiteWise asset.`,
);

const sdkSendDataToIoTSiteWiseAsset = new ScenarioAction(
  "sdkSendDataToIoTSiteWiseAsset",
  async (state) => {
    try {
      const sendResponse = await state.iotSiteWiseClient.send(
        new BatchPutAssetPropertyValueCommand({
          entries: [
            {
              entryId: "entry-3",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Humidity,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.humidity,
                  },
                  timestamp: {
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
            {
              entryId: "entry-4",
              assetId: state.asset.assetId,
              propertyId: state.propertyIds.Temperature,
              propertyValues: [
                {
                  value: {
                    doubleValue: state.sampleData.temperature,
                  },
                  timestamp: {
                    timeInSeconds: Math.floor(Date.now() / 1000),
                  },
                },
              ],
            },
```

```
          ],
        }),
      );
      console.log("The data was sent successfully.");
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayRetrieveValueOfIoTSiteWiseAsset = new ScenarioOutput(
  "displayRetrieveValueOfIoTSiteWiseAsset",
  `5. Retrieve the value of the IoT SiteWise Asset property

IoT SiteWise is an AWS service that allows you to collect, process, and analyze
 industrial data from connected equipment and sensors. One of the key benefits of
 reading an IoT SiteWise property is the ability to gain valuable insights from
 your industrial data.`,
);

const sdkRetrieveValueOfIoTSiteWiseAsset = new ScenarioAction(
  "sdkRetrieveValueOfIoTSiteWiseAsset",
  async (/** @type {State} */ state) => {
    try {
      const temperatureResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Temperature,
        }),
      );
      const humidityResponse = await state.iotSiteWiseClient.send(
        new GetAssetPropertyValueCommand({
          assetId: state.asset.assetId,
          propertyId: state.propertyIds.Humidity,
        }),
      );
      console.log(
        `The property value for Temperature is
 ${temperatureResponse.propertyValue.value.doubleValue}`,
```

```
      );
      console.log(
        `The property value for Humidity is
 ${humidityResponse.propertyValue.value.doubleValue}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateIoTSiteWisePortal = new ScenarioOutput(
  "displayCreateIoTSiteWisePortal",
  `6. Create an IoT SiteWise Portal

An IoT SiteWise Portal allows you to aggregate data from multiple industrial
 sources, such as sensors, equipment, and control systems, into a centralized
 platform.`,
);

const sdkCreateIoTSiteWisePortal = new ScenarioAction(
  "sdkCreateIoTSiteWisePortal",
  async (/** @type {State} */ state) => {
    try {
      const createPortalResponse = await state.iotSiteWiseClient.send(
        new CreatePortalCommand({
          portalName: state.portal.portalName,
          portalContactEmail: state.contactEmail,
          roleArn: state.stack.OutputValue,
        }),
      );
      state.portal = { ...state.portal, ...createPortalResponse };
      await wait(5); // Allow the portal to properly propagate.
      console.log(
        `Portal created successfully. Portal ID
 ${createPortalResponse.portalId}`,
      );
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
```

```
        console.log(
          `There was a problem creating the Portal: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribePortal = new ScenarioOutput(
  "displayDescribePortal",
  `7. Describe the Portal

In this step, we get a description of the portal and display the portal URL.`,
);

const sdkDescribePortal = new ScenarioAction(
  "sdkDescribePortal",
  async (/** @type {State} */ state) => {
    try {
      const describePortalResponse = await state.iotSiteWiseClient.send(
        new DescribePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(`Portal URL: ${describePortalResponse.portalStartUrl}`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayCreateIoTSiteWiseGateway = new ScenarioOutput(
  "displayCreateIoTSiteWiseGateway",
  `8. Create an IoT SiteWise Gateway
```

```
IoT SiteWise Gateway serves as the bridge between industrial equipment, sensors,
 and the cloud-based IoT SiteWise service. It is responsible for securely
 collecting, processing, and transmitting data from various industrial assets
 to the IoT SiteWise platform, enabling real-time monitoring, analysis, and
 optimization of industrial operations.`,
);

const sdkCreateIoTSiteWiseGateway = new ScenarioAction(
  "sdkCreateIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const createGatewayResponse = await state.iotSiteWiseClient.send(
        new CreateGatewayCommand({
          gatewayName: state.gateway.gatewayName,
          gatewayPlatform: {
            greengrassV2: {
              coreDeviceThingName: state.thing,
            },
          },
        }),
      );
      console.log(
        `Gateway creation completed successfully. ID is
 ${createGatewayResponse.gatewayId}`,
      );
      state.gateway.gatewayId = createGatewayResponse.gatewayId;
    } catch (caught) {
      if (caught.name === "IoTSiteWiseException") {
        console.log(
          `There was a problem creating the gateway: ${caught.message}.`,
        );
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const displayDescribeIoTSiteWiseGateway = new ScenarioOutput(
  "displayDescribeIoTSiteWiseGateway",
  "9. Describe the IoT SiteWise Gateway",
);
```

```
const sdkDescribeIoTSiteWiseGateway = new ScenarioAction(
  "sdkDescribeIoTSiteWiseGateway",
  async (/** @type {State} */ state) => {
    try {
      const describeGatewayResponse = await state.iotSiteWiseClient.send(
        new DescribeGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
      console.log("Gateway creation completed successfully.");
      console.log(`Gateway Name: ${describeGatewayResponse.gatewayName}`);
      console.log(`Gateway ARN: ${describeGatewayResponse.gatewayArn}`);
      console.log(
        `Gateway Platform:
 ${Object.keys(describeGatewayResponse.gatewayPlatform)}`,
      );
      console.log(
        `Gateway Creation Date: ${describeGatewayResponse.creationDate}`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
        throw caught;
      }
      console.error(`${caught.message}`);
      throw caught;
    }
  },
);

const askToDeleteResources = new ScenarioInput(
  "askToDeleteResources",
  `10. Delete the AWS IoT SiteWise Assets

Before you can delete the Asset Model, you must delete the assets.`,
  { type: "confirm" },
);

const displayConfirmDeleteResources = new ScenarioAction(
  "displayConfirmDeleteResources",
  async (/** @type {State} */ state) => {
    if (state.askToDeleteResources) {
      return "You selected to delete the SiteWise assets.";
    }
```

```
    return "The resources will not be deleted. Please delete them manually to
 avoid charges.";
  },
);

const sdkDeleteResources = new ScenarioAction(
  "sdkDeleteResources",
  async (/** @type {State} */ state) => {
    await wait(10); // Give the portal status time to catch up.
    try {
      await state.iotSiteWiseClient.send(
        new DeletePortalCommand({
          portalId: state.portal.portalId,
        }),
      );
      console.log(
        `Portal ${state.portal.portalName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Portal ${state.portal.portalName} was not found.`);
      } else {
        console.log(`When trying to delete the portal: ${caught.message}`);
      }
    }

    try {
      await state.iotSiteWiseClient.send(
        new DeleteGatewayCommand({
          gatewayId: state.gateway.gatewayId,
        }),
      );
      console.log(
        `Gateway ${state.gateway.gatewayName} was deleted successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Gateway ${state.gateway.gatewayId} was not found.`);
      } else {
        console.log(`When trying to delete the gateway: ${caught.message}`);
      }
    }

    try {
```

```
      await state.iotSiteWiseClient.send(
        new DeleteAssetCommand({
          assetId: state.asset.assetId,
        }),
      );
      await wait(5); // Allow the delete to finish.
      console.log(`Asset ${state.asset.assetName} was deleted successfully.`);
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(`The Asset ${state.asset.assetName} was not found.`);
      } else {
        console.log(`When deleting the asset: ${caught.message}`);
      }
    }

    await wait(30); // Allow asset deletion to finish.
    try {
      await state.iotSiteWiseClient.send(
        new DeleteAssetModelCommand({
          assetModelId: state.assetModel.assetModelId,
        }),
      );
      console.log(
        `Asset Model ${state.assetModel.assetModelName} was deleted
successfully.`,
      );
    } catch (caught) {
      if (caught.name === "ResourceNotFoundException") {
        console.log(
          `The Asset Model ${state.assetModel.assetModelName} was not found.`,
        );
      } else {
        console.log(`When deleting the asset model: ${caught.message}`);
      }
    }

    try {
      await state.cloudFormationClient.send(
        new DeleteStackCommand({
          StackName: stackName,
        }),
      );
      await waitUntilStackDeleteComplete(
        { client: state.cloudFormationClient },
```

```
        { StackName: stackName },
      );
      console.log("The stack was deleted successfully.");
    } catch (caught) {
      console.log(
        `${caught.message}. The stack was NOT deleted. Please clean up the
 resources manually.`,
      );
    }
  },
  { skipWhen: (/** @type {{}} */ state) => !state.askToDeleteResources },
);

const goodbye = new ScenarioOutput(
  "goodbye",
  "This concludes the IoT Sitewise Basics scenario for the AWS Javascript SDK v3.
 Thank you!",
);

const myScenario = new Scenario(
  "IoTSiteWise Basics",
  [
    greet,
    pressEnter,
    displayBuildCloudFormationStack,
    sdkBuildCloudFormationStack,
    pressEnter,
    displayCreateAWSSiteWiseAssetModel,
    sdkCreateAWSSiteWiseAssetModel,
    displayCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    waitThirtySeconds,
    sdkCreateAWSIoTSiteWiseAssetModel,
    pressEnter,
    displayRetrievePropertyId,
    sdkRetrievePropertyId,
    pressEnter,
    displaySendDataToIoTSiteWiseAsset,
    sdkSendDataToIoTSiteWiseAsset,
    pressEnter,
    displayRetrieveValueOfIoTSiteWiseAsset,
    sdkRetrieveValueOfIoTSiteWiseAsset,
    pressEnter,
    displayCreateIoTSiteWisePortal,
```

```
    sdkCreateIoTSiteWisePortal,
    pressEnter,
    displayDescribePortal,
    sdkDescribePortal,
    pressEnter,
    displayCreateIoTSiteWiseGateway,
    sdkCreateIoTSiteWiseGateway,
    pressEnter,
    displayDescribeIoTSiteWiseGateway,
    sdkDescribeIoTSiteWiseGateway,
    pressEnter,
    askToDeleteResources,
    displayConfirmDeleteResources,
    sdkDeleteResources,
    goodbye,
  ],
  {
    iotSiteWiseClient: new IoTSiteWiseClient({}),
    cloudFormationClient: new CloudFormationClient({}),
    asset: { assetName: "MyAsset1" },
    assetModel: { assetModelName: "MyAssetModel1" },
    portal: { portalName: "MyPortal1" },
    gateway: { gatewayName: "MyGateway1" },
    propertyIds: [],
    contactEmail: "user@mydomain.com",
    thing: "MyThing1",
    sampleData: { temperature: 23.5, humidity: 65.0 },
  },
);

/** @type {{ stepHandlerOptions: StepHandlerOptions }} */
export const main = async (stepHandlerOptions) => {
  await myScenario.run(stepHandlerOptions);
};

// Invoke main function if this file was run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const { values } = parseArgs({
    options: {
      yes: {
        type: "boolean",
        short: "y",
      },
    },
```

```
  });
  main({ confirmAll: values.yes });
}
```

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

コマンドプロンプトからインタラクティブのシナリオを実行します。

```
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 アクションをラップする IoTSitewiseWrapper クラス。

```
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(gateway_id=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(gateway_id=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 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDKs AWS IoT SiteWise を使用するためのアクション

次のコード例は、 AWS SDKs で個々の AWS IoT SiteWise アクションを実行する方法を示しています。それぞれの例には、GitHub へのリンクがあり、そこにはコードの設定と実行に関する説明が記載されています。

以下の例には、最も一般的に使用されるアクションのみ含まれています。詳細な一覧については、「AWS IoT SiteWise API リファレンス」を参照してください。

例

- AWS SDK または CLI BatchPutAssetPropertyValueで を使用する
- AWS SDK または CLI CreateAssetで を使用する

- AWS SDK または CLI CreateAssetModelで を使用する
- AWS SDK または CLI CreateGatewayで を使用する
- AWS SDK または CLI CreatePortalで を使用する
- AWS SDK または CLI DeleteAssetで を使用する
- AWS SDK または CLI DeleteAssetModelで を使用する
- AWS SDK または CLI DeleteGatewayで を使用する
- AWS SDK または CLI DeletePortalで を使用する
- AWS SDK または CLI DescribeAssetModelで を使用する
- AWS SDK または CLI DescribeGatewayで を使用する
- AWS SDK または CLI DescribePortalで を使用する
- AWS SDK または CLI GetAssetPropertyValueで を使用する
- AWS SDK または CLI ListAssetModelsで を使用する

AWS SDK または CLI **BatchPutAssetPropertyValue**で を使用する

次のサンプルコードは、BatchPutAssetPropertyValue を使用する方法を説明しています。

CLI

AWS CLI

アセットプロパティにデータを送信するには

次の batch-put-asset-property-value の例では、プロパティエイリアスによって識別されるアセットプロパティに電力と温度のデータを送信します。

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

batch-put-asset-property-value.json の内容:

```
{
    "entries": [
        {
            "entryId": "1575691200-company-windfarm-3-turbine-7-power",
```

```
            "propertyAlias": "company-windfarm-3-turbine-7-power",
            "propertyValues": [
                {
                    "value": {
                        "doubleValue": 4.92
                    },
                    "timestamp": {
                        "timeInSeconds": 1575691200
                    },
                    "quality": "GOOD"
                }
            ]
        },
        {
            "entryId": "1575691200-company-windfarm-3-turbine-7-temperature",
            "propertyAlias": "company-windfarm-3-turbine-7-temperature",
            "propertyValues": [
                {
                    "value": {
                        "integerValue": 38
                    },
                    "timestamp": {
                        "timeInSeconds": 1575691200
                    }
                }
            ]
        }
    ]
}
```

出力:

```
{
    "errorEntries": []
}
```

詳細については、AWS 「 IoT SiteWise ユーザーガイド」の「IoT SiteWise API を使用したデータの取り込み」を参照してください。 AWS IoT SiteWise

- API の詳細については、「AWS CLI コマンドリファレンス」の「BatchPutAssetPropertyValue」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /**
    * 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());
                }
            });
    }
```

- API の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のBatchPutAssetPropertyValue」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のBatchPutAssetPropertyValue」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

値リストからエントリパラメータを生成するヘルパー関数。

```
    def properties_to_values(
        self, asset_id: str, values: list[dict[str, Any]]
    ) -> list[dict[str, Any]]:
        """
        Utility function to convert a values list to the entries parameter for
 batch_put_asset_property_value.
        :param asset_id : The asset ID.
        :param values : A list of dictionaries containing the values in the form
                        {propertyId : property_id,
```

```
                        valueType : [stringValue|integerValue|doubleValue|
booleanValue],
                        value : the_value}.
        :return: An entries list to pass as the 'entries' parameter to
 batch_put_asset_property_value.
        """
        entries = []
        for value in values:
            epoch_ns = time.time_ns()
            self.entry_id += 1
            if value["valueType"] == "stringValue":
                property_value = {"stringValue": value["value"]}
            elif value["valueType"] == "integerValue":
                property_value = {"integerValue": value["value"]}
            elif value["valueType"] == "booleanValue":
                property_value = {"booleanValue": value["value"]}
            elif value["valueType"] == "doubleValue":
                property_value = {"doubleValue": value["value"]}
            else:
                raise ValueError("Invalid valueType: %s", value["valueType"])
            entry = {
                "entryId": f"{self.entry_id}",
                "assetId": asset_id,
                "propertyId": value["propertyId"],
                "propertyValues": [
                    {
                        "value": property_value,
                        "timestamp": {
                            "timeInSeconds": int(epoch_ns / 1000000000),
                            "offsetInNanos": epoch_ns % 1000000000,
                        },
                    }
                ],
            }
            entries.append(entry)
        return entries
```

ヘルパー関数に渡す値リストの例を次に示します。

```
        values = [
            {
```

```
                "propertyId": humidity_property_id,
                "valueType": "doubleValue",
                "value": 65.0,
            },
            {
                "propertyId": temperature_property_id,
                "valueType": "doubleValue",
                "value": 23.5,
            },
        ]
```

- API の詳細については、 AWS SDK for Python (Boto3) API リファレンスのBatchPutAssetPropertyValue」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateAsset**で を使用する

次のサンプルコードは、`CreateAsset` を使用する方法を説明しています。

CLI

AWS CLI

アセットを作成するには

次の `create-asset` の例では、風力タービンアセットモデルから風力タービンアセットを作成します。

```
aws iotsitewise create-asset \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE \
    --asset-name "Wind Turbine 1"
```

出力:

```
{
    "assetId": "a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
    "assetArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset/
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
```

```
    "assetStatus": {
        "state": "CREATING"
    }
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Creating assets」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreateAsset」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」の「CreateAsset」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」の「CreateAsset」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスの「CreateAsset」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateAssetModel**で を使用する

次のサンプルコードは、`CreateAssetModel` を使用する方法を説明しています。

CLI

AWS CLI

アセットモデルを作成するには

次の `create-asset-model` の例では、次のプロパティを使用して風力タービンを定義するアセットモデルを作成します。

シリアル番号 - 風力タービンのシリアル番号 発電電力 - 風力タービンからの発電電力のデータストリーム 温度 C - 風力タービンからの温度のデータストリーム (摂氏) 温度 F - 摂氏から華氏にマップされた温度のデータポイント

```
aws iotsitewise create-asset-model \
    --cli-input-json file://create-wind-turbine-model.json
```

`create-wind-turbine-model.json` の内容:

```
{
    "assetModelName": "Wind Turbine Model",
    "assetModelDescription": "Represents a wind turbine",
    "assetModelProperties": [
```

```
        {
            "name": "Serial Number",
            "dataType": "STRING",
            "type": {
                "attribute": {}
            }
        },
        {
            "name": "Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "measurement": {}
            }
        },
        {
            "name": "Temperature C",
            "dataType": "DOUBLE",
            "unit": "Celsius",
            "type": {
                "measurement": {}
            }
        },
        {
            "name": "Temperature F",
            "dataType": "DOUBLE",
            "unit": "Fahrenheit",
            "type": {
                "transform": {
                    "expression": "temp_c * 9 / 5 + 32",
                    "variables": [
                        {
                            "name": "temp_c",
                            "value": {
                                "propertyId": "Temperature C"
                            }
                        }
                    ]
                }
            }
        },
        {
            "name": "Total Generated Power",
            "dataType": "DOUBLE",
```

```
            "unit": "kW",
            "type": {
                "metric": {
                    "expression": "sum(power)",
                    "variables": [
                        {
                            "name": "power",
                            "value": {
                                "propertyId": "Generated Power"
                            }
                        }
                    ],
                    "window": {
                        "tumbling": {
                            "interval": "1h"
                        }
                    }
                }
            }
        }
    ]
}
```

出力:

```
{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "assetModelStatus": {
        "state": "CREATING"
    }
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Defining asset models」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreateAssetModel」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /**
    * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」の「CreateAssetModel」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」の「CreateAssetModel」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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
```

関数に渡すプロパティリストの例を次に示します。

```
            properties = [
                {
                    "name": temperature_property_name,
                    "dataType": "DOUBLE",
                    "type": {
                        "measurement": {},
                    },
                },
                {
                    "name": humidity_property_name,
                    "dataType": "DOUBLE",
```

```
                    "type": {
                        "measurement": {},
                    },
                },
            ]
```

- API の詳細については、 AWS SDK for Python (Boto3) API リファレンスの「CreateAssetModel」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreateGateway**で を使用する

次のサンプルコードは、`CreateGateway` を使用する方法を説明しています。

CLI

AWS CLI

ゲートウェイを作成するには

次の`create-gateway`例では、 AWS IoT Greengrass で実行されるゲートウェイを作成します。

```
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}
```

出力:

```
{
    "gatewayId": "a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE",
    "gatewayArn": "arn:aws:iotsitewise:us-west-2:123456789012:gateway/
a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE"
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Configuring a gateway」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreateGateway」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /**
    * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」の「CreateGateway」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」の「CreateGateway」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスの「CreateGateway」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **CreatePortal**で を使用する

次のサンプルコードは、`CreatePortal` を使用する方法を説明しています。

CLI

AWS CLI

ポータルを作成するには

次の `create-portal` の例では、風力発電所会社のウェブポータルを作成します。ポータルは、 AWS シングルサインオンを有効にしたのと同じリージョンでのみ作成できます。

```
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
```

出力:

```
{
    "portalId": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
    "portalArn": "arn:aws:iotsitewise:us-west-2:123456789012:portal/
a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
    "portalStartUrl": "https://a1b2c3d4-5678-90ab-cdef-
aaaaaEXAMPLE.app.iotsitewise.aws",
    "portalStatus": {
        "state": "CREATING"
    },
    "ssoApplicationId": "ins-a1b2c3d4-EXAMPLE"
}
```

詳細については、AWS 「 IoT SiteWise ユーザーガイド」の「IoT SiteWise Monitor の開始方法」および「 IoT SiteWise AWS ユーザーガイド」の「SSO の有効化」を参照してください。 AWS IoT SiteWise AWS IoT SiteWise

- API の詳細については、「AWS CLI コマンドリファレンス」の「CreatePortal」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /**
    * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のCreatePortal」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API CreatePortal」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスの「CreatePortal」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteAsset**で を使用する

次のサンプルコードは、DeleteAsset を使用する方法を説明しています。

CLI

AWS CLI

アセットを削除するには

次の delete-asset の例では、風力タービンアセットを削除します。

```
aws iotsitewise delete-asset \
    --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE
```

出力:

```
{
    "assetStatus": {
        "state": "DELETING"
    }
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Deleting assets」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteAsset」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /**
    * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のDeleteAsset」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のDeleteAsset」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスのDeleteAsset」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteAssetModel**で を使用する

次のサンプルコードは、`DeleteAssetModel` を使用する方法を説明しています。

CLI

AWS CLI

アセットモデルを削除するには

次の `delete-asset-model` の例では、風力タービンアセットモデルを削除します。

```
aws iotsitewise delete-asset-model \
```

```
    --asset-model-id a1b2c3d4-5678-90ab-cdef-11111EXAMPLE
```

出力:

```
{
    "assetModelStatus": {
        "state": "DELETING"
    }
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Deleting asset models」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteAssetModel」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のDeleteAssetModel」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のDeleteAssetModel」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスのDeleteAssetModel」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeleteGateway**で を使用する

次のサンプルコードは、`DeleteGateway` を使用する方法を説明しています。

CLI

AWS CLI

ゲートウェイを削除するには

次の `delete-gateway` の例では、ゲートウェイを削除します。

```
aws iotsitewise delete-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

このコマンドでは何も出力されません。

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Ingesting data using a gateway」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeleteGateway」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のDeleteGateway」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のDeleteGateway」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスのDeleteGateway」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DeletePortal**で を使用する

次のサンプルコードは、`DeletePortal` を使用する方法を説明しています。

CLI

AWS CLI

ポータルを削除するには

次の `delete-portal` の例では、風力発電所会社のウェブポータルを削除します。

```
aws iotsitewise delete-portal \
    --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

出力:

```
{
    "portalStatus": {
        "state": "DELETING"
    }
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Deleting a portal」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DeletePortal」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のDeletePortal」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のDeletePortal」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスのDeletePortal」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeAssetModel**で を使用する

次のサンプルコードは、DescribeAssetModel を使用する方法を説明しています。

CLI

AWS CLI

アセットモデルの説明を表示するには

次の describe-asset-model の例では、風力発電所アセットモデルの説明を表示します。

```
aws iotsitewise describe-asset-model \
    --asset-model-id a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

出力:

```
{
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetModelArn": "arn:aws:iotsitewise:us-west-2:123456789012:asset-model/
a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetModelName": "Wind Farm Model",
    "assetModelDescription": "Represents a wind farm that comprises many wind
 turbines",
    "assetModelProperties": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-99999EXAMPLE",
            "name": "Total Generated Power",
            "dataType": "DOUBLE",
            "unit": "kW",
            "type": {
                "metric": {
                    "expression": "sum(power)",
                    "variables": [
                        {
```

```
                            "name": "power",
                            "value": {
                                "propertyId": "a1b2c3d4-5678-90ab-
cdef-66666EXAMPLE",
                                "hierarchyId": "a1b2c3d4-5678-90ab-
cdef-77777EXAMPLE"
                            }
                        }
                    ],
                    "window": {
                        "tumbling": {
                            "interval": "1h"
                        }
                    }
                }
            }
        },
        {
            "id": "a1b2c3d4-5678-90ab-cdef-88888EXAMPLE",
            "name": "Region",
            "dataType": "STRING",
            "type": {
                "attribute": {
                    "defaultValue": " "
                }
            }
        }
    ],
    "assetModelHierarchies": [
        {
            "id": "a1b2c3d4-5678-90ab-cdef-77777EXAMPLE",
            "name": "Wind Turbines",
            "childAssetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"
        }
    ],
    "assetModelCreationDate": 1575671284.0,
    "assetModelLastUpdateDate": 1575671988.0,
    "assetModelStatus": {
        "state": "ACTIVE"
    }
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Describing a specific asset model」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DescribeAssetModel」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result
that associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}.
By calling
     *         {@link CompletionException#getCause()}, the calling code can
access the original exception.
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
```

```
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
 AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs:
 {}.", throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
```

- API の詳細については、AWS SDK for Java 2.x 「 API リファレンス」の「DescribeAssetModel」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」の「DescribeAssetModel」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribeGateway**で を使用する

次のサンプルコードは、DescribeGateway を使用する方法を説明しています。

CLI

AWS CLI

ゲートウェイの説明を表示するには

次の describe-gateway の例では、ゲートウェイの説明を表示します。

```
aws iotsitewise describe-gateway \
    --gateway-id a1b2c3d4-5678-90ab-cdef-1a1a1EXAMPLE
```

出力:

```
{
```

```
    "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
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Ingesting data using a gateway」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DescribeGateway」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」の「DescribeGateway」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」の「DescribeGateway」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスの「DescribeGateway」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **DescribePortal**で を使用する

次のサンプルコードは、DescribePortal を使用する方法を説明しています。

CLI

AWS CLI

ポータルの説明を表示するには

次の describe-portal の例では、風力発電所会社のウェブポータルの説明を表示します。

```
aws iotsitewise describe-portal \
    --portal-id a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE
```

出力:

```
{
    "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"
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「Administering your portals」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「DescribePortal」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」の「DescribePortal」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のDescribePortal」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスの「DescribePortal」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **GetAssetPropertyValue**で を使用する

次のサンプルコードは、`GetAssetPropertyValue` を使用する方法を説明しています。

CLI

AWS CLI

アセットプロパティの現在の値を取得するには

次の `get-asset-property-value` の例では、風力タービンアセットの現在の総電力を取得します。

```
aws iotsitewise get-asset-property-value \
    --asset-id a1b2c3d4-5678-90ab-cdef-33333EXAMPLE \
    --property-id a1b2c3d4-5678-90ab-cdef-66666EXAMPLE
```

出力:

```
{
    "propertyValue": {
        "value": {
            "doubleValue": 6890.8677520453875
        },
        "timestamp": {
            "timeInSeconds": 1580853000,
            "offsetInNanos": 0
        },
        "quality": "GOOD"
    }
}
```

詳細については、「AWS IoT SiteWise ユーザーガイド」の「現在のアセットプロパティ値をクエリする」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「GetAssetPropertyValue」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
   /**
    * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のGetAssetPropertyValue」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のGetAssetPropertyValue」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスのGetAssetPropertyValue」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

AWS SDK または CLI **ListAssetModels**で を使用する

次のサンプルコードは、ListAssetModels を使用する方法を説明しています。

CLI

AWS CLI

すべてのアセットモデルを一覧表示するには

次の のlist-asset-models例では、現在のリージョンの AWS アカウントで定義されているすべてのアセットモデルを一覧表示します。

```
aws iotsitewise list-asset-models
```

出力:

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

詳細については、「AWS IoT SiteWise ユーザーガイド」の「すべてのアセットモデルのリスト表示」を参照してください。

- API の詳細については、「AWS CLI コマンドリファレンス」の「ListAssetModels」を参照してください。

Java

SDK for Java 2.x

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
    /**
     * 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 の詳細については、AWS SDK for Java 2.x 「 API リファレンス」のListAssetModels」を参照してください。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for JavaScript 「 API リファレンス」のListAssetModels」を参照してください。

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub には、その他のリソースもあります。AWS コード例リポジトリ で全く同じ例を見つけて、設定と実行の方法を確認してください。

```
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 の詳細については、 AWS SDK for Python (Boto3) API リファレンスのListAssetModels」を参照してください。

AWS SDK 開発者ガイドとコード例の完全なリストについては、「」を参照してくださいAWS SDK でのこのサービスの使用。このトピックには、使用開始方法に関する情報と、以前の SDK バージョンの詳細も含まれています。

のセキュリティ AWS IoT SiteWise

でのクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、最もセキュリティの影響を受けやすい組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。 AWS また、 では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証します。が適用されるコンプライアンスプログラムの詳細については AWS IoT SiteWise、「コンプライアンスAWS プログラムによる対象範囲内のサービスコンプライアンス」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、 を使用する際の責任共有モデルの適用方法を理解するのに役立ちます AWS IoT SiteWise。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成する AWS IoT SiteWise ように を設定する方法について説明します。また、 AWS IoT SiteWise リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- でのデータ保護 AWS IoT SiteWise
- でのデータの暗号化 AWS IoT SiteWise
- の ID とアクセスの管理 AWS IoT SiteWise
- のコンプライアンス検証 AWS IoT SiteWise
- の耐障害性 AWS IoT SiteWise
- のインフラストラクチャセキュリティ AWS IoT SiteWise
- での設定と脆弱性の分析 AWS IoT SiteWise
- の VPC エンドポイント AWS IoT SiteWise
- のセキュリティのベストプラクティス AWS IoT SiteWise

でのデータ保護 AWS IoT SiteWise

責任 AWS 共有モデル、 でのデータ保護に適用されます AWS IoT SiteWise。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール AWS IoT SiteWise 、API、または SDK を使用して AWS CLIまたは他の AWS のサービス を操作する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

トピック

- のインターネットトラフィックのプライバシー AWS IoT SiteWise
- AWS IoT SiteWise アシスタントビジネスサービスの改善

のインターネットトラフィックのプライバシー AWS IoT SiteWise

SiteWise Edge ゲートウェイなどの AWS IoT SiteWise とオンプレミスアプリケーション間の接続は、Transport Layer Security (TLS) 接続で保護されます。詳細については、「の転送中のデータ暗号化 AWS IoT SiteWise」を参照してください。

AWS IoT SiteWise は、 AWS リージョン内のアベイラビリティーゾーン間の接続や AWS アカウント間の接続をサポートしていません。

IAM Identity Center は、一度に 1 つのリージョンでしか設定できません。SiteWise Monitor は、IAM Identity Center に対して設定されたリージョンに接続します。つまり、IAM Identity Center へのアクセスには 1 つのリージョンを使用しますが、どのリージョンにもポータルを作成することができます。

AWS IoT SiteWise アシスタントビジネスサービスの改善

AWS IoT SiteWise Assistant は、サービスの改善や基盤となる LLMs の改善に顧客データを使用しません。

でのデータの暗号化 AWS IoT SiteWise

データ暗号化とは、転送中 (との間で送受信されるとき AWS IoT SiteWise、および SiteWise Edge ゲートウェイとサーバー間で送受信されるとき) および保管中 (ローカルデバイスまたは AWS サービスに保存されるとき) のデータを保護することを意味します。転送中のデータは、Transport Layer Security (TLS) を使用して保護することも、クライアント側の暗号化を使用して保存することもできます。

> **Note**
>
> AWS IoT SiteWise エッジ処理は、SiteWise Edge ゲートウェイ内でホストされ、ローカルネットワーク経由でアクセス可能な APIs を公開します。これらの APIsは、 AWS IoT SiteWise Edge コネクタが所有するサーバー証明書によってバックアップされた TLS 接続を介して公開されます。これらの API では、クライアント認証にアクセス制御パスワードが使用されます。サーバー証明書のプライベートキーとアクセスコントロールパスワードはどち

らもディスクに保存されます。 AWS IoT SiteWise エッジ処理は、保管中のこれらの認証情報のセキュリティのためにファイルシステムの暗号化に依存します。

サーバー側の暗号化とクライアント側の暗号化の詳細については、以下のトピックを参照してください。

トピック

- での保管時の暗号化 AWS IoT SiteWise
- の転送中のデータ暗号化 AWS IoT SiteWise
- でのキー管理 AWS IoT SiteWise

での保管時の暗号化 AWS IoT SiteWise

AWS IoT SiteWise は、データを AWS クラウドと AWS IoT SiteWise Edge ゲートウェイに保存します。

AWS クラウドに保管中のデータ

AWS IoT SiteWise は、デフォルトで保管中のデータを暗号化する他の AWS サービスにデータを保存します。保管時の暗号化は AWS Key Management Service （ AWS KMS) と統合され、アセットプロパティ値と集計値を暗号化するために使用される暗号化キーを管理します AWS IoT SiteWise。AWS IoT SiteWiseのアセットプロパティ値と集計値の暗号化には、カスタマーマネージドキーの使用を選択することができます。 AWS KMSを通じて、暗号化キーの作成、管理、閲覧が可能です。

AWS 所有のキー を選択してデータを暗号化するか、カスタマーマネージドキーを選択してアセットプロパティ値と集計値を暗号化できます。

仕組み

保管時の暗号化は と統合され、データの暗号化に使用される暗号化キー AWS KMS を管理します。

- AWS 所有のキー – デフォルトの暗号化キー。このキー AWS IoT SiteWise を所有します。 AWS アカウントでこのキーを表示することはできません。また、 AWS CloudTrail のログでキーに対する操作を確認することもできません。このキーは追加料金なしで使用することができます。
- カスタマーマネージドキー - キーは、お客様が作成、所有、管理するアカウントに保存されます。ユーザーは、KMS キーに関する完全なコントロール権を持ちます。 AWS KMS 追加料金が適用されます。

AWS 所有のキー

AWS 所有のキー はアカウントに保存されません。これらは、複数の AWS アカウントで使用するために が AWS 所有および管理する KMS キーのコレクションの一部です。 AWS サービスは を使用して AWS 所有のキー データを保護します。

それらの使用を表示、管理、使用 AWS 所有のキー、または監査することはできません。ただし、データを暗号化するキーを保護するための作業やプログラムを操作したり変更したりする必要はありません。

を使用する場合、月額料金や使用料金は請求されず AWS 所有のキー、アカウントの AWS KMS クォータにはカウントされません。

カスタマーマネージドキー

カスタマーマネージドキーは、お客様が作成、所有、管理するアカウント内の KMS キーです。これらの KMS キーはお客様が完全に制御でき、次のような操作が可能です。

- キーポリシー、IAM ポリシー、グラントの確立と維持
- 有効化と無効化
- 暗号化マテリアルのローテーション
- タグの追加
- それらを参照するエイリアスの作成
- 削除のスケジュール設定

CloudTrail と Amazon CloudWatch Logs を使用して、 が AWS KMS ユーザーに代わって AWS IoT SiteWise に送信するリクエストを追跡することもできます。

カスタマーマネージドキーを使用している場合は、 アカウントに保存されている KMS キー AWS IoT SiteWise へのアクセスを許可する必要があります。 AWS IoT SiteWise は、エンベロープ暗号化とキー階層を使用してデータを暗号化します。 AWS KMS 暗号化キーは、このキー階層のルートキーを暗号化するために使用されます。詳細については、「AWS Key Management Service デベロッパーガイド」の「エンベロープ暗号化」を参照してください。

次のポリシー例では、ユーザーに代わってカスタマーマネージドキーの作成にアクセス AWS IoT SiteWise 許可を付与します。キーを作成する際に、`kms:CreateGrant` および `kms:DescribeKey` アクションを許可する必要があります。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1603902045292",
      "Action": [
        "kms:CreateGrant",
        "kms:DescribeKey"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

作成したグラントの暗号化コンテキストは、`aws:iotsitewise:subscriberId` とアカウント ID を使用します。

SiteWise Edge ゲートウェイに保管中のデータ

AWS IoT SiteWise ゲートウェイは、ローカルファイルシステムに次のデータを保存します。

- OPC UA ソースの設定情報
- 接続された OPC UA ソースからの OPC UA データストリームパスのセット
- SiteWise Edge ゲートウェイがインターネットへの接続を失ったときにキャッシュされる産業用データ

SiteWise Edge ゲートウェイは AWS IoT Greengrass、Unix ファイルのアクセス許可とフルディスク暗号化 (有効になっている場合) AWS IoT Greengrass に基づいて で実行され、コア上の保管中のデータを保護します。ファイルシステムとデバイスを保護するのはお客様の責任となります。

ただし、 AWS IoT Greengrass Secrets Manager から取得した OPC UA サーバーシークレットのローカルコピーは暗号化されます。詳細については、[AWS IoT Greengrass Version 1 Developer Guide] (デベロッパーガイド) の[Secrets encryption] (秘密暗号化) を参照してください。

AWS IoT Greengrass コアの保管時の暗号化の詳細については、「 AWS IoT Greengrass Version 1 デベロッパーガイド」の「保管時の暗号化」を参照してください。

の転送中のデータ暗号化 AWS IoT SiteWise

AWS IoT SiteWise は転送中の暗号化を使用して、デバイス、ゲートウェイ、 AWS クラウド間で送信されるデータを保護します。との通信 AWS IoT SiteWise は HTTPS および TLS 1.2 を使用して暗号化されるため、データの機密性が維持され、不正アクセスや傍受から保護されます。

データの転送時には 3 つの通信モードがあります。

- インターネット経由 – ローカルデバイス (SiteWise Edge ゲートウェイを含む) と 間の通信 AWS IoT SiteWise は暗号化されます。
- [ローカルネットワーク経由] - OpsHub for SiteWise アプリケーションと SiteWise Edge ゲートウェイ間の通信は常に暗号化されます。お客様のブラウザ内で実行されている SiteWise モニターアプリケーションと SiteWise Edge ゲートウェイ間の通信は常に暗号化されます。SiteWise Edge ゲートウェイと OPC UA ソースの間の通信は暗号化できます。
- SiteWise Edge ゲートウェイ上のコンポーネント間 — SiteWise Edge ゲートウェイ上の AWS IoT Greengrass コンポーネント間の通信は暗号化されません。

トピック

- インターネット経由で転送されるデータ
- ローカルネットワーク経由で転送されるデータ
- SiteWise Edge 上のローカルコンポーネント間で転送中のデータ

インターネット経由で転送されるデータ

AWS IoT SiteWise は Transport Layer Security (TLS) を使用して、インターネット経由のすべての通信を暗号化します。 AWS クラウドに送信されるすべてのデータは、MQTT または HTTPS プロトコルを使用して TLS 接続を介して送信されるため、デフォルトでは安全です。SiteWise Edge ゲートウェイは で実行され AWS IoT Greengrass、プロパティ値通知は AWS IoT トランスポートセキュリティモデルを使用します。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Transport security] (トランスポートセキュリティ) を参照してください。

ローカルネットワーク経由で転送されるデータ

SiteWise Edge ゲートウェイは、ローカルの OPC UA ソースとの通信について、OPC UA 仕様に従います。転送されるデータを暗号化するメッセージセキュリティモードを使用するようにソースを設定するのは、お客様の責任となります。

[署名] メッセージセキュリティモードを選択した場合、SiteWise Edge ゲートウェイとソースの間で転送中のデータは署名されますが、暗号化はされません。[署名と暗号化] メッセージセキュリティモードを選択した場合、SiteWise Edge ゲートウェイとソースの間で転送中のデータは署名および暗号化されます。ソースの設定の詳細については、「AWS IoT SiteWise Edge ゲートウェイにデータソースを追加する」を参照してください。

エッジコンソールアプリケーションと SiteWise Edge ゲートウェイ間の通信は、常に TLS によって暗号化されます。SiteWise Edge ゲートウェイ上の SiteWise Edge コネクタは、 AWS IoT SiteWise アプリケーションのエッジコンソールとの TLS 接続を確立できるように、自己署名証明書を生成して保存します。 AWS IoT SiteWise アプリケーションを SiteWise Edge ゲートウェイに接続する前に、この証明書を SiteWise Edge ゲートウェイからアプリケーションのエッジコンソールにコピーする必要があります。これにより、 AWS IoT SiteWise アプリケーションのエッジコンソールが信頼できる SiteWise Edge ゲートウェイに接続していることを確認できます。

機密保持とサーバーの真正性のための TLS に加え、SiteWise Edge は、SigV4 プロトコルを使用して、エッジコンソールアプリケーションの真正性を確立します。SiteWise Edge ゲートウェイの SiteWise Edge コネクタは、エッジコンソールアプリケーション、ブラウザ内で実行されている SiteWise Monitor アプリケーション、および AWS IoT SiteWise SDK に基づく他のクライアントからの受信接続を検証するためのパスワードを受け入れて保存します。

パスワードとサーバー証明書の生成については、the section called “ゲートウェイを管理” を参照してください。

SiteWise Edge 上のローカルコンポーネント間で転送中のデータ

SiteWise Edge ゲートウェイは で実行され AWS IoT Greengrass、データはデバイスから離れないため、 AWS IoT Greengrass コア上でローカルに交換されたデータは暗号化されません。これには、AWS IoT SiteWise コネクタなどのコンポーネント間の AWS IoT Greengrass 通信が含まれます。詳細については、[AWS IoT Greengrass Version 1 Developer Guide] (デベロッパーガイド) の[Data on the core device] (コアデバイスのデータ) を参照してください。

でのキー管理 AWS IoT SiteWise

AWS IoT SiteWise クラウドキー管理

デフォルトでは、 AWS IoT SiteWise AWS マネージドキー を使用して AWS クラウド内のデータを保護します。 AWS IoT SiteWiseの一部のデータを暗号化するために、カスタマーマネージドキーを使用するように設定を更新することができます。 AWS Key Management Service (AWS KMS). を通じて、暗号化キーの作成、管理、閲覧が可能です。

AWS IoT SiteWise は、 に保存されているカスタマーマネージドキーによるサーバー側の暗号化 AWS KMS をサポートし、次のデータを暗号化します。

- アセットプロパティ値。
- 集計値。

> **Note**
>
> 他のデータとリソースは、 によって管理されるキーによるデフォルトの暗号化を使用して暗号化されます AWS IoT SiteWise。このキーは、 AWS IoT SiteWise アカウントに保存されます。

詳細については、「 AWS Key Management Service デベロッパーガイド」の「What is AWS Key Management Service？」を参照してください。

カスタマーマネージドキーを使用した暗号化の有効化。

でカスタマーマネージドキーを使用するには AWS IoT SiteWise、設定を更新 AWS IoT SiteWise する必要があります。

KMS キーを使用した暗号化を有効にするには。

1. AWS IoT SiteWise コンソールに移動します。
2. [Account Settings] (アカウント設定) を選び、[Edit] (編集) を選んで[Edit account settings] (アカウント設定の編集) 画面を表示します。
3. [Encryption key type] (暗号化キー型) で[Choose a different AWS KMS key] (別のキーを選択) を選択します。これにより、 AWS KMSに保存されたカスタマーマネージドキーによる暗号化が有効となります。

   > **Note**
   >
   > 現在、カスタマーマネージドキーの暗号化は、アセットのプロパティ値と集計値に対してのみ使用可能です。

4. KMS キーは、次のいずれかを選択してください。

   - 既存の KMS キーを使用するには - リストから KMS キーエイリアスを選択します。

- 新しい KMS キーを作成するには – AWS KMS キーの作成を選択します。

> **Note**
>
> これにより、 AWS KMS ダッシュボードが開きます。詳細については、[AWS Key Management Service Developer Guide] (デベロッパーガイド) の[Creating keys] (キーの作成) を参照してください。

5. [保存] を選択して、設定を更新します。

SiteWise Edge ゲートウェイキー管理

SiteWise Edge ゲートウェイは で実行され AWS IoT Greengrass、 AWS IoT Greengrass コアデバイスはパブリックキーとプライベートキーを使用して AWS クラウドで認証し、OPC UA 認証シークレットなどのローカルシークレットを暗号化します。詳細については、[AWS IoT Greengrass Version 1 Developer Guide] (デベロッパーガイド) の[Key management] (キー管理) を参照してください。

の ID とアクセスの管理 AWS IoT SiteWise

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS IoT SiteWise リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- AWS IoT SiteWise セキュリティ対象者
- で ID を使用して認証する AWS IoT SiteWise
- が IAM と AWS IoT SiteWise 連携する方法
- AWS の 管理ポリシー AWS IoT SiteWise
- のサービスにリンクされたロールを使用する AWS IoT SiteWise
- でイベントアラームのアクセス許可を設定する AWS IoT SiteWise
- でのサービス間の混乱した代理の防止 AWS IoT SiteWise
- AWS IoT SiteWise ID とアクセスのトラブルシューティング

AWS IoT SiteWise セキュリティ対象者

AWS Identity and Access Management (IAM) の使用方法は、作業内容によって異なります AWS IoT SiteWise。

サービスユーザー – AWS IoT SiteWise サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が提供されます。さらに多くの AWS IoT SiteWise 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。 AWS IoT SiteWise機能にアクセスできない場合は、「AWS IoT SiteWise ID とアクセスのトラブルシューティング」を参照してください。

サービス管理者 – 社内の AWS IoT SiteWise リソースを担当している場合は、通常、 へのフルアクセスがあります AWS IoT SiteWise。サービスユーザーがどの AWS IoT SiteWise 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で IAM を使用する方法の詳細については AWS IoT SiteWise、「」を参照してくださいが IAM と AWS IoT SiteWise 連携する方法。

IAM 管理者 - 管理者は、 AWS IoT SiteWiseへのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる AWS IoT SiteWise アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS IoT SiteWise アイデンティティベースのポリシーの例。

で ID を使用して認証する AWS IoT SiteWise

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けることによって、認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーション ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 AWS サインイン ユーザーガイド」の「 へのサインイン AWS アカウント方法」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストを暗号化して署名します。AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 4」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。たとえば、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「IAM のAWS 多要素認証」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションすることをお勧めします。詳細については、「IAM ユーザーガイド」の「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールについては、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスにおけるロールとリソースベースのポリシーの違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。
- クロスサービスアクセス — 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスで

は、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。

- 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用してアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
- サービスロール - サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

が IAM と AWS IoT SiteWise 連携する方法

AWS Identity and Access Management (IAM) を使用して へのアクセスを管理する前に AWS IoT SiteWise、使用できる IAM 機能を理解しておく必要があります AWS IoT SiteWise。

IAM の機能	でサポートされていますか AWS IoT SiteWis
リソースレベルのアクセス許可を持つアイデンティティベースポリシー	はい
ポリシーアクション	はい
ポリシーリソース	あり
ポリシー条件キー	はい
リソースベースのポリシー	いいえ
アクセスコントロールリスト (ACL)	いいえ
タグベースの認可 (ABAC)	はい

IAM の機能	でサポートされていますか AWS IoT SiteWis
一時的な認証情報	はい
転送アクセスセッション (FAS)	はい
サービスリンクロール	はい
サービスロール	はい

AWS IoT SiteWise およびその他の AWS のサービスが IAM と連携する方法の概要を把握するには、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

目次

- AWS IoT SiteWise IAM ロール
  - で一時的な認証情報を使用する AWS IoT SiteWise
  - の転送アクセスセッション (FAS) AWS IoT SiteWise
  - サービスにリンクされた役割
  - サービス役割

- AWS IoT SiteWiseで IAM ロールを選択する
- AWS IoT SiteWise タグに基づく認可
- AWS IoT SiteWise ID ベースのポリシー
  - ポリシーアクション
    - BatchPutAssetPropertyValue 認可
  - ポリシーリソース
  - ポリシー条件キー
  - 例
- AWS IoT SiteWise アイデンティティベースのポリシーの例
  - ポリシーに関するベストプラクティス
  - AWS IoT SiteWise コンソールを使用する
  - 自分の権限の表示をユーザーに許可する
  - ユーザーが 1 つの階層でアセットにデータを取り込むことを許可する
  - タグに基づいて AWS IoT SiteWise アセットを表示する
- のポリシーを使用してアクセスを管理する AWS IoT SiteWise
  - アイデンティティベースのポリシー
  - リソースベースのポリシー
  - アクセスコントロールリスト (ACL)
  - その他のポリシータイプ
  - 複数のポリシータイプ

AWS IoT SiteWise IAM ロール

IAM 役割 は特定のアクセス許可を持つ、 AWS アカウント内のエンティティです。

で一時的な認証情報を使用する AWS IoT SiteWise

一時的な認証情報を使用して、フェデレーションでサインインする、IAM ロールを引き受ける、またはクロスアカウントロールを引き受けることができます。一時的なセキュリティ認証情報を取得するには、AssumeRole または GetFederationToken などの AWS STS API オペレーションを呼び出します。

AWS IoT SiteWise では、一時的な認証情報の使用がサポートされています。

SiteWise Monitor は、ポータルにアクセスするためのフェデレーティッドユーザーをサポートします。ポータルのユーザーは、IAM Identity Center または IAM の認証情報を使って認証を受けます。

> **Important**
>
> ユーザーまたはロールがこのポータルにサインインするには `iotsitewise:DescribePortal` アクセス許可が必要です。

ユーザーがポータルにサインインすると、SiteWise Monitor が次のアクセス許可を提供するセッションポリシーを生成します。

- そのポータルのロールがアクセスを提供するアカウント AWS IoT SiteWise 内の のアセットとアセットデータへの読み取り専用アクセス。
- ユーザーが管理者 (プロジェクト所有者) または読み取り専用 (プロジェクトビューワー) のアクセス権を持つ、そのポータル内のプロジェクトへのアクセス。

フェデレーティッドユーザーのアクセス許可の詳細については、「のサービスロールを使用する AWS IoT SiteWise Monitor」を参照してください。

の転送アクセスセッション (FAS) AWS IoT SiteWise

転送アクセスセッション (FAS) のサポート: あり

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする と組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

サービスにリンクされた役割

サービスにリンクされたロールを使用すると、 AWS サービスは他の サービスのリソースにアクセスしてユーザーに代わってアクションを実行できます。 サービスにリンクされたロールは AWS アカウントに表示され、 サービスによって所有されます。IAM 管理者は、サービスリンクロールの許可を表示できますが、編集することはできません。

AWS IoT SiteWise は、サービスにリンクされたロールをサポートします。 AWS IoT SiteWise サービスにリンクされたロールの作成または管理の詳細については、「のサービスにリンクされたロールを使用する AWS IoT SiteWise」を参照してください。

サービス役割

この機能により、ユーザーに代わってサービスがサービス役割を引き受けることが許可されます。この役割により、サービスがお客様に代わって他のサービスのリソースにアクセスし、アクションを完了することが許可されます。サービスロールは AWS アカウントに表示され、アカウントによって所有されます。つまり、IAM 管理者はこの役割の権限を変更できます。ただし、それにより、サービスの機能が損なわれる場合があります。

AWS IoT SiteWise はサービスロールを使用して、SiteWise Monitor ポータルユーザーがユーザーに代わって一部のリソース AWS IoT SiteWise にアクセスできるようにします。詳細については、「のサービスロールを使用する AWS IoT SiteWise Monitor」を参照してください。

AWS IoT Events アラームモデルを作成する前に、必要なアクセス許可が必要です AWS IoT SiteWise。詳細については、「でイベントアラームのアクセス許可を設定する AWS IoT SiteWise」を参照してください。

AWS IoT SiteWiseで IAM ロールを選択する

で`portal`リソースを作成するときは AWS IoT SiteWise、SiteWise Monitor ポータルのフェデレーティッドユーザーが AWS IoT SiteWise ユーザーに代わって にアクセスすることを許可するロールを選択する必要があります。以前にサービスロールを作成している場合、 AWS IoT SiteWise は選択するロールのリストを提供します。それ以外の場合は、ポータルの作成時に必要なアクセス許可を持つロールを作成できます。アセットとアセットデータへのアクセスを許可するロールを選択することが重要です。詳細については、「のサービスロールを使用する AWS IoT SiteWise Monitor」を参照してください。

AWS IoT SiteWise タグに基づく認可

AWS IoT SiteWise リソースにタグをアタッチしたり、リクエストでタグを渡すことができます AWS IoT SiteWise。タグに基づいてアクセスを制御するには`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。 AWS IoT SiteWise リソースのタグ付けの詳細については、「AWS IoT SiteWise リソースにタグを付ける」を参照してください。

リソースのタグに基づいてリソースへのアクセスを制限するためのアイデンティティベースのポリシーの例を表示するには、「タグに基づいて AWS IoT SiteWise アセットを表示する」を参照してください。

AWS IoT SiteWise ID ベースのポリシー

IAM ポリシーを使用すると、 AWS IoT SiteWiseで誰が何をできるかを制御できます。許可されるアクションと許可されないアクションを決定し、これらのアクションに特定の条件を設定できます。たとえば、 で情報を表示または変更できるユーザーに関するルールを作成できます AWS IoT SiteWise。 は、特定のアクション、リソース、および条件キー AWS IoT SiteWise をサポートしています。JSON ポリシーで使用するすべての要素については「IAM ユーザーガイド」の「IAM JSON ポリシーエレメントのリファレンス」を参照してください。

ポリシーアクション

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは依存アクションと呼ばれます。

このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

のポリシーアクションは、アクションの前にプレフィックス AWS IoT SiteWise を使用します`iotsitewise:`。たとえば、 `BatchPutAssetPropertyValue` API オペレーション AWS IoT SiteWise を使用して にアセットプロパティデータをアップロードするアクセス許可を付与するには、ポリシーに `iotsitewise:BatchPutAssetPropertyValue`アクションを含めます。ポリシーステートメントには、 `Action`または `NotAction` element を含める必要があります。 は、このサービスで実行できるタスクを記述する独自のアクションのセット AWS IoT SiteWise を定義します。

単一のステートメントに複数の アクションを指定するには、次のようにコンマで区切ります。

```
"Action": [
```

```
  "iotsitewise:action1",
  "iotsitewise:action2"
]
```

ワイルドカード *を使用して複数のアクションを指定することができます。例えば、Describe という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "iotsitewise:Describe*"
```

AWS IoT SiteWise アクションのリストを確認するには、IAM ユーザーガイドの「 で定義されるアクション AWS IoT SiteWise」を参照してください。

BatchPutAssetPropertyValue 認可

AWS IoT SiteWise は、BatchPutAssetPropertyValue アクションへのアクセスを異常な方法で許可します。ほとんどのアクションでは、アクセスを許可または拒否するとき、アクセス許可が付与されていない場合、そのアクションはエラーを返します。BatchPutAssetPropertyValue を使用すると、1 つの API リクエストで複数のデータエントリを異なるアセットおよびアセットプロパティに送信できます。 AWS IoT SiteWise は各データエントリを個別に認可します。リクエストで認可に失敗した個々のエントリの場合、 はエラーの返されたリストAccessDeniedExceptionに AWS IoT SiteWise を含めます。 は、同じリクエスト内の別のエントリが失敗した場合でも、認可および成功したエントリのデータ AWS IoT SiteWise を受け取ります。

> **Important**
>
> データストリームにデータを取り込む前に、次のことを行ってください。
>
> - データストリームの識別にプロパティエイリアスを使用する場合は、time-series リソースを認可します。
> - 関連するアセットプロパティを含むアセットを識別するためにアセット ID を使用する場合、asset リソースを認可します。

ポリシーリソース

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ステートメントにはResource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

操作のリスト化など、リソースレベルの許可をサポートしないアクションの場合は、ワイルドカード (*) を使用して、ステートメントがすべてのリソースに適用されることを示します。

```
"Resource": "*"
```

各 IAM ポリシーステートメントはARN を使用して指定したリソースに適用されます。ARN には以下の一般的な構文があります。

```
arn:${Partition}:${Service}:${Region}:${Account}:${ResourceType}/${ResourcePath}
```

ARN の形式の詳細については、「Amazon AWS リソースネーム (ARNs) でリソースを識別する」を参照してください。

たとえば、ステートメントで ID a1b2c3d4-5678-90ab-cdef-22222EXAMPLE のアセットを指定するには、次の ARN を使用します。

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE"
```

特定のアカウントに属するすべてのデータストリームを指定する場合は、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:iotsitewise:region:123456789012:time-series/*"
```

特定のアカウントに属するすべてのアセットを指定するには、ワイルドカード (*) を使用します。

```
"Resource": "arn:aws:iotsitewise:region:123456789012:asset/*"
```

リソースを作成するためのアクションなど、一部の AWS IoT SiteWise アクションは、特定のリソースで実行できません。このような場合は、ワイルドカード (*) を使用する必要があります。

```
"Resource": "*"
```

複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": [
  "resource1",
  "resource2"
]
```

AWS IoT SiteWise リソースタイプとその ARNs」を参照してください。 AWS IoT SiteWise どのアクションで各リソースの ARN を指定できるかについては、「AWS IoT SiteWiseで定義されるアクション」を参照してください。

ポリシー条件キー

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素 (または `Condition` ブロック) を使用すると、ステートメントが有効な条件を指定できます。`Condition` 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の `Condition` 要素を指定する場合、または 1 つの `Condition` 要素に複数のキーを指定する場合、 AWS では `AND` 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理`OR`オペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

> **Important**
>
> 多くの条件キーはリソースに固有のものであり、一部の API アクションでは複数のリソースを使用します。条件キーを使用してポリシーステートメントを作成する場合は、ポリシーステートメントの `Resource` 要素で、条件キーが適用されるリソースを指定します。指定しない場合、そのポリシーはユーザーに対してすべてのアクションの実行を禁止する可能性があります。これは、条件キーが適用されないリソースに対して条件チェックが失敗するためで

す。リソースを指定しない場合や、ポリシーの `Action` 要素に複数の API アクションを含めている場合は、`...IfExists` 条件タイプを使用して、条件キーが適用されないリソースに対して無視されるようにする必要があります。詳細については、[IAM User Guide] (IAM ユーザーガイド) の[...IfExists conditions] (..IfExists 条件) を参照してください。

AWS IoT SiteWise は独自の条件キーのセットを定義し、いくつかのグローバル条件キーの使用もサポートしています。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

AWS IoT SiteWise 条件キー

条件キー	説明	型
`iotsitewise:isAssociatedWithAssetProperty`	データストリームがアセットプロパティと関連付けられているかどうか。この条件キーを使用して、データストリームの関連するアセットプロパティの存在に基づくアクセス許可を定義することができます。 値の例: `true`	String
`iotsitewise:assetHierarchyPath`	アセットの階層パス。各アセット ID をスラッシュで区切った文字列です。この条件キーを使用して、アカウント内のすべてのアセットの階層のサブセットに基づいたアクセス許可を定義します。 値の例: `/a1b2c3d4-5678-90ab-cdef-22222EXAMPLE/a1b2c3d4-5678-90ab-cdef-66666EXAMPLE`	String

条件キー	説明	型
`iotsitewise:propertyId`	アセットプロパティの ID。この条件キーを使用して、アセットモデルの指定されたプロパティに基づいたアクセス許可を定義します。この条件キーは、そのモデルのすべてのアセットに適用されます。 値の例: `a1b2c3d4-5678-90ab-cdef-33333EXAMPLE`	String
`iotsitewise:childAssetId`	別のアセットに子として関連付けられているアセットの ID。この条件キーを使用して、子アセットに基づいたアクセス許可を定義します。親アセットに基づいてアクセス許可を定義するには、ポリシーステートメントのリソースセクションを使用します。 値の例: `a1b2c3d4-5678-90ab-cdef-66666EXAMPLE`	String
`iotsitewise:iam`	アクセスポリシーを一覧表示する際の IAM アイデンティティ のARN です。この条件キーを使用して、IAM アイデンティティのアクセスポリシー許可を定義します。 値の例: `arn:aws:iam::123456789012:user/JohnDoe`	文字列、Null

条件キー	説明	型
`iotsitewise:propertyAlias`	アセットプロパティまたはデータストリームを識別するエイリアス。この条件キーを使用して、エイリアスに基づくアクセス許可を定義します。	String
`iotsitewise:user`	アクセスポリシーをリスト化するときの IAM Identity Center ユーザーの ID。この条件キーを使用して、IAM Identity Center ユーザーのアクセスポリシーアクセス許可を定義します。 値の例: `a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE`	文字列、Null
`iotsitewise:group`	アクセスポリシーをリスト化するときの IAM Identity Center グループの ID。この条件キーを使用して、IAM Identity Center グループのアクセスポリシーアクセス許可を定義します。 値の例: `a1b2c3d4e5-a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE`	文字列、Null

条件キー	説明	型
iotsitewise:portal	アクセスポリシー内のポータルの ID。この条件キーを使用して、ポータルに基づいたアクセスポリシー許可を定義します。 値の例: a1b2c3d4-5678-90ab-cdef-77777EXAMPLE	文字列、Null
iotsitewise:project	アクセスポリシー内のプロジェクトの ID、またはダッシュボードのプロジェクトの ID。この条件キーを使用して、ポータルに基づいたダッシュボードまたはアクセスポリシー許可を定義します。 値の例: a1b2c3d4-5678-90ab-cdef-88888EXAMPLE	文字列、Null

条件キーを使用できるアクションとリソースについては、「 で定義されるアクション AWS IoT SiteWise」を参照してください。

例

AWS IoT SiteWise アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAWS IoT SiteWise アイデンティティベースのポリシーの例。

AWS IoT SiteWise アイデンティティベースのポリシーの例

デフォルトでは、エンティティ (ユーザーやロール) には、 AWS IoT SiteWise リソースを作成または変更するアクセス許可は付与されていません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできませ

ん。アクセス許可を調整するには、 AWS Identity and Access Management (IAM) 管理者が以下を実行する必要があります。

1. 必要なリソースに対して特定の API 操作を実行するためにユーザーとロールにアクセス許可を付与する IAM ポリシーを作成します。
2. それらのアクセス許可が必要なユーザーまたはグループに、それらのポリシーをアタッチします。

これらの JSON ポリシードキュメント例を使用して IAM のアイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「JSON タブでのポリシーの作成」を参照してください。

トピック

- ポリシーに関するベストプラクティス
- AWS IoT SiteWise コンソールを使用する
- 自分の権限の表示をユーザーに許可する
- ユーザーが 1 つの階層でアセットにデータを取り込むことを許可する
- タグに基づいて AWS IoT SiteWise アセットを表示する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内の AWS IoT SiteWise リソースを誰かが作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行 – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」または「ジョブ機能の AWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する

方法の詳細については、「IAM ユーザーガイド」の「IAM でのポリシーとアクセス許可」を参照してください。

- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素:条件」を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、「IAM ユーザーガイド」の「MFA を使用した安全な API アクセス」を参照してください。

IAM でのベストプラクティスの詳細については、「IAM ユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

AWS IoT SiteWise コンソールを使用する

AWS IoT SiteWise コンソールにアクセスするには、基本的なアクセス許可のセットが必要です。これらのアクセス許可により、 AWS アカウント内の AWS IoT SiteWise リソースの詳細を表示および管理できます。

制限が厳しいポリシーを作成すると、そのポリシーを持つユーザーまたはロール (エンティティ) ではコンソールが期待どおりに動作しない可能性があります。これらのエンティティが AWS IoT SiteWise 引き続きコンソールを使用できるようにするには、AWSIoTSiteWiseConsoleFullAccess 管理ポリシーをアタッチするか、それらのエンティティに同等のアクセス許可を定義します。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

エンティティがコンソールではなく AWS Command Line Interface (CLI) または AWS IoT SiteWise API のみを使用している場合、これらの最小限のアクセス許可は必要ありません。その場合、API タスクに必要な特定のアクションへのアクセス権を付与します。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

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

ユーザーが 1 つの階層でアセットにデータを取り込むことを許可する

この例では、ルートアセット から、特定のアセット階層内のすべてのアセットプロパティにデータを書き込むためのアクセス権を AWS アカウントのユーザーに付与しますa1b2c3d4-5678-90ab-cdef-22222EXAMPLE。このポリシーでは、ユーザーに iotsitewise:BatchPutAssetPropertyValue アクセス許可を付与します。このポリシーは、iotsitewise:assetHierarchyPath 条件キーを使用して、階層パスがアセットまたはその子孫と一致するアセットへのアクセスを制限します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PutAssetPropertyValuesForHierarchy",
      "Effect": "Allow",
      "Action": "iotsitewise:BatchPutAssetPropertyValue",
      "Resource": "arn:aws:iotsitewise:*:*:asset/*",
      "Condition": {
        "StringLike": {
          "iotsitewise:assetHierarchyPath": [
            "/",
            "//*"
          ]
        }
      }
    }
  ]
}
```

タグに基づいて AWS IoT SiteWise アセットを表示する

アイデンティティベースのポリシーの条件を使用して、タグに基づいて AWS IoT SiteWise リソースへのアクセスを制御します。この例は、アセットの表示を許可するポリシーを作成する方法を示しています。ただし、アクセス許可は、アセットタグ Owner にそのユーザーのユーザー名の値がある場合のみ、付与されます。このポリシーでは、このアクションをコンソールで実行するために必要なアクセス許可も付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListAllAssets",
      "Effect": "Allow",
      "Action": [
        ":ListAssets",
        ":ListAssociatedAssets"
      ],
      "Resource": "*"
    },
    {
      "Sid": "DescribeAssetIfOwner",
      "Effect": "Allow",
      "Action": ":DescribeAsset",
      "Resource": "arn:aws::*:*:asset/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Owner": "${aws:username}"
        }
      }
    }
  ]
}
```

このポリシーをアカウントのユーザーにアタッチします。という名前のユーザーが AWS IoT SiteWise アセットを表示`richard-roe`しようとする場合、アセットには `Owner=richard-roe`または のタグを付ける必要があります`owner=richard-roe`。それ以外の場合、リチャードはアクセスを拒否されます。条件タグキーの名前では大文字と小文字が区別されません。そのため、`Owner` は `Owner` と `owner` の両方に一致します。詳細については、「IAM ユーザーガイド」の「IAM JSON ポリシー要素: 条件」を参照してください。

のポリシーを使用してアクセスを管理する AWS IoT SiteWise

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッ

ション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの

場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、および Amazon VPC は AWS WAF、ACLs。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、追加のあまり一般的ではないポリシータイプをサポートしています。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPsは、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所有する複数の をグループ化して一元管理するためのサービス AWS アカウント です。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。

- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS のサービス
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

AWS の 管理ポリシー AWS IoT SiteWise

自分でポリシーを記述するのではなく、 AWS 管理ポリシーを使用して、ユーザー、グループ、ロールにアクセス許可を簡単に追加できます。チームに必要なアクセス許可のみを提供する IAM カスタマー管理ポリシーを作成するには、時間と専門知識が必要です。セットアップを高速化するには、一般的なユースケースに AWS マネージドポリシーを使用することを検討してください。 AWS アカウントで AWS 管理ポリシーを検索します。 AWS マネージドポリシーの詳細については「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWS サービスは、 AWS 管理ポリシーの更新と保守を行います。つまり、これらのポリシーのアクセス許可を変更することはできません。場合によっては、新機能に対応するためのアクセス許可を追加し、ポリシーがアタッチされたすべての ID に影響を与える AWS IoT SiteWise ことがあります。このような更新は、新しいサービスや新機能の導入を伴うのが一般的です。ただし、アクセス許可は削除されないため、設定はそのまま維持されます。

さらに、 は、複数のサービスにまたがるジョブ関数の マネージドポリシー AWS をサポートしています。例えば、ReadOnlyAccess AWS 管理ポリシーは、すべての AWS サービスとリソースへの読

み取り専用アクセスを提供します。サービスが新機能を起動すると、 は新しいオペレーションとリソースの読み取り専用アクセス許可 AWS を追加します。ジョブ機能ポリシーのリスト付き説明については、[IAM User Guide] (IAM ユーザーガイド) の[AWS managed policies for job functions] (ジョブ関数のマネージドポリシー) を参照してください。

AWS マネージドポリシー: AWSIoTSiteWiseReadOnlyAccess

`AWSIoTSiteWiseReadOnlyAccess` AWS 管理ポリシーを使用して、 への読み取り専用アクセスを許可します AWS IoT SiteWise。

`AWSIoTSiteWiseReadOnlyAccess` ポリシーを IAM アイデンティティにアタッチできます。

サービスレベルのアクセス許可

このポリシーは、 への読み取り専用アクセスを提供します AWS IoT SiteWise。このポリシーには、他のサービス権限は含まれていません。

JSON

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

AWS マネージドポリシー: AWSServiceRoleForIoTSiteWise

`AWSServiceRoleForIoTSiteWise` ロールは、次のアクセス許可で `AWSServiceRoleForIoTSiteWise` ポリシーを使用します。このポリシー:

- AWS IoT SiteWise が SiteWise Edge ゲートウェイ (で実行される) をデプロイできるようにしますAWS IoT Greengrass。
- AWS IoT SiteWise ログ記録の実行を に許可します。
- AWS IoT SiteWise が AWS IoT TwinMaker データベースに対してメタデータ検索クエリを実行できるようにします。

単一のユーザーアカウント AWS IoT SiteWise で を使用している場合、AWSServiceRoleForIoTSiteWiseロールは IAM アカウントにAWSServiceRoleForIoTSiteWiseポリシーを作成し、 AWSServiceRoleForIoTSiteWiseのサービスにリンクされたロールにアタッチします。 AWS IoT SiteWise

JSON

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

JSON

```
{
 "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-us-gov:logs:*:*:log-group:/aws/iotsitewise*"
    },
    {
     "Sid": "AllowSiteWiseAccessLog",
     "Effect": "Allow",
     "Action": [
      "logs:CreateLogStream",
      "logs:DescribeLogStreams",
      "logs:PutLogEvents"
     ],
     "Resource": "arn:aws-us-gov:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
    },
    {
     "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
     "Effect": "Allow",
     "Action": [
      "iottwinmaker:GetWorkspace",
      "iottwinmaker:ExecuteQuery"
     ],
     "Resource": "arn:aws-us-gov:iottwinmaker:*:*:workspace/*",
     "Condition": {
      "ForAnyValue:StringEquals": {
       "iottwinmaker:linkedServices": [
        "IOTSITEWISE"
       ]
      }
     }
    }
   ]
  }
```

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowSiteWiseReadGreenGrass",
   "Effect": "Allow",
   "Action": [
    "greengrass:GetAssociatedRole",
    "greengrass:GetCoreDefinition",
    "greengrass:GetCoreDefinitionVersion",
    "greengrass:GetGroup",
    "greengrass:GetGroupVersion"
   ],
   "Resource": "*"
  },
  {
   "Sid": "AllowSiteWiseAccessLogGroup",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogGroup",
    "logs:DescribeLogGroups"
   ],
   "Resource": "arn:aws-cn:logs:*:*:log-group:/aws/iotsitewise*"
  },
  {
   "Sid": "AllowSiteWiseAccessLog",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:DescribeLogStreams",
    "logs:PutLogEvents"
   ],
   "Resource": "arn:aws-cn:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
  },
  {
   "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
   "Effect": "Allow",
   "Action": [
    "iottwinmaker:GetWorkspace",
    "iottwinmaker:ExecuteQuery"
```

```
    ],
    "Resource": "arn:aws-cn:iottwinmaker:*:*:workspace/*",
    "Condition": {
     "ForAnyValue:StringEquals": {
      "iottwinmaker:linkedServices": [
       "IOTSITEWISE"
      ]
     }
    }
   }
  ]
}
```

AWS IoT SiteWiseAWS 管理ポリシーの更新

このサービスが変更の追跡を開始した時点から AWS IoT SiteWise、 の AWS マネージドポリシーの更新に関する詳細を表示できます。このページの変更に関する自動アラートについては、 AWS IoT SiteWise ドキュメント履歴ページの RSS フィードにサブスクライブしてください。

変更	説明	日付
AWSServiceRoleForIoTSiteWise – 既存のポリシーの更新	AWS IoT SiteWise は、データベースに対して AWS IoT TwinMaker メタデータ検索クエリを実行できるようになりました。	2023 年 11 月 6 日
AWSIoTSiteWiseReadOnlyAccess – 既存のポリシーに対する更新	AWS IoT SiteWise に新しいポリシープレフィックス が追加されました。これにより`BatchGet*` 、バッチ読み取りオペレーションを実行できます。	2022 年 9 月 16 日
AWSIoTSiteWiseReadOnlyAccess – 新しいポリシー。	AWS IoT SiteWise は、 への読み取り専用アクセスを許可する新しいポリシーを追加しました AWS IoT SiteWise。	2021 年 11 月 24 日

変更	説明	日付
AWS IoT SiteWise が変更の追跡を開始しました	AWS IoT SiteWise は、 AWS 管理ポリシーの変更の追跡を開始しました。	2021 年 11 月 24 日

のサービスにリンクされたロールを使用する AWS IoT SiteWise

AWS IoT SiteWise は AWS Identity and Access Management (IAM) サービスにリンクされたロールを使用します。サービスにリンクされたロールは、 に直接リンクされた一意のタイプの IAM ロールです AWS IoT SiteWise。 サービスにリンクされたロールは、 によって事前定義 AWS IoT SiteWise されており、サービスがユーザーに代わって他の AWS サービスを呼び出すために必要なすべてのアクセス許可が含まれています。

サービスにリンクされたロールは、必要なすべてのアクセス許可を自動的に含め AWS IoT SiteWise ることで、 の設定を簡素化します。 は、サービスにリンクされたロールのアクセス許可 AWS IoT SiteWise を定義します。特に定義されている場合を除き、 のみがそのロールを引き受け AWS IoT SiteWise ることができます。定義された許可には、信頼ポリシーと許可ポリシーが含まれます。また、このアクセス許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスリンクロールを削除するには、最初に関連リソースを削除する必要があります。これにより、 AWS IoT SiteWise リソースへのアクセス許可が誤って削除されないため、リソースが保護されます。

サービスにリンクされたロールをサポートする他のサービスの詳細については、AWS 「IAM と連携するサービス」を参照し、「サービスにリンクされたロール」列で「はい」を持つサービスを探します。そのサービスに関するサービスにリンクされたロールのドキュメントを表示するには、リンクが設定されている Yes] (はい) を選択します。

トピック

- AWS IoT SiteWiseのサービスリンクロールのアクセス許可
- AWS IoT SiteWiseサービスにリンクされたロールを作成する
- AWS IoT SiteWiseのサービスリンクロールを更新する
- AWS IoT SiteWiseのサービスにリンクされたロールを削除する
- AWS IoT SiteWise サービスにリンクされたロールでサポートされているリージョン
- のサービスロールを使用する AWS IoT SiteWise Monitor

AWS IoT SiteWiseのサービスリンクロールのアクセス許可

AWS IoT SiteWise は、AWSServiceRoleForIoTSiteWise という名前のサービスにリンクされたロールを使用します。 はこのサービスにリンクされたロール AWS IoT SiteWise を使用して SiteWise Edge ゲートウェイ (で実行される AWS IoT Greengrass) をデプロイし、ログ記録を実行します。

`AWSServiceRoleForIoTSiteWise` サービスリンクロールは、次のアクセス許可を持つ `AWSServiceRoleForIoTSiteWise` ポリシーを使用します。このポリシー:

- AWS IoT SiteWise が SiteWise Edge ゲートウェイ (で実行される) をデプロイできるようにします`AWS IoT Greengrass`。
- AWS IoT SiteWise ログ記録の実行を に許可します。
- AWS IoT SiteWise が AWS IoT TwinMaker データベースに対してメタデータ検索クエリを実行できるようにします。

`AWSServiceRoleForIoTSiteWise` で許可されるアクションの詳細については、「AWS IoT SiteWiseのAWS マネージドポリシー」を参照してください。

JSON

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

JSON

```
{
 "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-us-gov:logs:*:*:log-group:/aws/iotsitewise*"
   },
   {
    "Sid": "AllowSiteWiseAccessLog",
    "Effect": "Allow",
    "Action": [
     "logs:CreateLogStream",
     "logs:DescribeLogStreams",
     "logs:PutLogEvents"
    ],
    "Resource": "arn:aws-us-gov:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
   },
   {
    "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
    "Effect": "Allow",
    "Action": [
     "iottwinmaker:GetWorkspace",
     "iottwinmaker:ExecuteQuery"
    ],
    "Resource": "arn:aws-us-gov:iottwinmaker:*:*:workspace/*",
    "Condition": {
     "ForAnyValue:StringEquals": {
      "iottwinmaker:linkedServices": [
       "IOTSITEWISE"
      ]
     }
    }
```

```
  }
 ]
}
```

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowSiteWiseReadGreenGrass",
   "Effect": "Allow",
   "Action": [
    "greengrass:GetAssociatedRole",
    "greengrass:GetCoreDefinition",
    "greengrass:GetCoreDefinitionVersion",
    "greengrass:GetGroup",
    "greengrass:GetGroupVersion"
   ],
   "Resource": "*"
  },
  {
   "Sid": "AllowSiteWiseAccessLogGroup",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogGroup",
    "logs:DescribeLogGroups"
   ],
   "Resource": "arn:aws-cn:logs:*:*:log-group:/aws/iotsitewise*"
  },
  {
   "Sid": "AllowSiteWiseAccessLog",
   "Effect": "Allow",
   "Action": [
    "logs:CreateLogStream",
    "logs:DescribeLogStreams",
    "logs:PutLogEvents"
   ],
   "Resource": "arn:aws-cn:logs:*:*:log-group:/aws/iotsitewise*:log-stream:*"
  },
```

```
    {
     "Sid": "AllowSiteWiseAccessSiteWiseManagedWorkspaceInTwinMaker",
     "Effect": "Allow",
     "Action": [
      "iottwinmaker:GetWorkspace",
      "iottwinmaker:ExecuteQuery"
     ],
     "Resource": "arn:aws-cn:iottwinmaker:*:*:workspace/*",
     "Condition": {
      "ForAnyValue:StringEquals": {
       "iottwinmaker:linkedServices": [
        "IOTSITEWISE"
       ]
      }
     }
    }
   ]
  }
```

ログを使用して、SiteWise Edge ゲートウェイのモニタリングとトラブルシューティングができます。詳細については、「SiteWise Edge ゲートウェイログをモニタリングする」を参照してください。

IAM エンティティ (ユーザ、グループ、ロールなど) にサービスリンクロールの作成、編集、削除を許可するには、まずアクセス許可を設定します。詳細についてはIAM ユーザーガイド の「サービスにリンクされた役割のアクセス許可」を参照してください。

AWS IoT SiteWiseサービスにリンクされたロールを作成する

AWS IoT SiteWise では、特定のアクションを実行し、ユーザーに代わってリソースにアクセスするために、サービスにリンクされたロールが必要です。サービスにリンクされたロールは、直接リンクされた一意のタイプの AWS Identity and Access Management (IAM) ロールです AWS IoT SiteWise。このロールを作成することで、データストレージや AWS IoT デバイス通信用の Amazon S3 など、そのオペレーションに必要な他の AWS サービスやリソースにアクセス AWS IoT SiteWise するために必要なアクセス許可を付与します。

サービスリンクロールを手動で作成する必要はありません。 AWS IoT SiteWise コンソールで次のオペレーションを実行すると、 によってサービスにリンクされたロールが自動的に AWS IoT SiteWise 作成されます。

- Greengrass V1 ゲートウェイを作成します。
- ログ記録オプションを設定します。
- クエリ実行バナーでオプトインボタンを選択します。

このサービスリンクロールを削除した後で再度作成する必要が生じた場合は同じ方法でアカウントにロールを再作成できます。 AWS IoT SiteWise コンソールでオペレーションを実行すると、 はサービスにリンクされたロールを再度 AWS IoT SiteWise 作成します。

IAM コンソールまたは API を使用して、 AWS IoT SiteWiseに対してサービスにリンクされたロールを作成することもできます。

- IAM コンソールでこれを行うには、AWSServiceRoleForIoTSiteWise ポリシーと`iotsitewise.amazonaws.com` との信頼関係でロールを作成します。
- AWS CLI または IAM API を使用してこれを行うには、 `arn:aws:iam::aws:policy/aws-service-role/AWSServiceRoleForIoTSiteWise`ポリシーと との信頼関係を持つロールを作成します`iotsitewise.amazonaws.com`。

詳細については、「IAM ユーザーガイド」の「サービスリンクロールの作成」を参照してください。

このサービスリンクロールを削除しても、同じ方法でロールを再作成できます。

AWS IoT SiteWiseのサービスリンクロールを更新する

AWS IoT SiteWise では、AWSServiceRoleForIoTSiteWise サービスにリンクされたロールを編集することはできません。サービスリンクロールを作成すると、多くのエンティティによってロールが参照される可能性があるため、ロール名を変更することはできません。ただし、IAM を使用したロール記述の編集はできます。詳細については、「IAM ユーザーガイド」の「サービスリンクロールの更新」を参照してください。

AWS IoT SiteWiseのサービスにリンクされたロールを削除する

サービスリンクロールを必要とする機能またはサービスが使用されなくなった場合は、関連付けられたロールを削除することをお勧めします。これは、モニタリングまたは維持されていない非アクティブなエンティティが発生しないようにするためです。ただし、手動で削除する前に、サービスリンクロールのリソースをクリーンアップする必要があります。

> **Note**
>
> リソースを削除しようとしたときに AWS IoT SiteWise サービスがロールを使用している場合は、削除が失敗する可能性があります。その場合は、数分待ってから再試行してください。

AWSServiceRoleForIoTSiteWise で使用される AWS IoT SiteWise リソースを削除するには

1. のログ記録を無効にします AWS IoT SiteWise。詳細については、ログ記録レベルを変更するを参照してください。
2. アクティブな SiteWise Edge ゲートウェイを削除します。

サービスリンクロールを IAM で手動削除するには

IAM コンソール、 AWS CLI、または AWS API を使用して、AWSServiceRoleForIoTSiteWise サービスにリンクされたロールを削除します。詳細については、「IAM ユーザーガイド」の「ロールまたはインスタンスプロファイルを削除する」を参照してください。

AWS IoT SiteWise サービスにリンクされたロールでサポートされているリージョン

AWS IoT SiteWise は、サービスが利用可能なすべてのリージョンでサービスにリンクされたロールの使用をサポートします。詳細については、「AWS IoT SiteWise エンドポイントとクォータ」を参照してください。

のサービスロールを使用する AWS IoT SiteWise Monitor

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」を参照してください。

SiteWise Monitor ポータルのフェデレーティッドユーザーに AWS IoT SiteWise および AWS IAM Identity Center リソースへのアクセスを許可するには、作成する各ポータルにサービスロールをアタッチする必要があります。サービスロールは、信頼されたエンティティとして SiteWise Monitor を指定し、AWSIoTSiteWiseMonitorPortalAccess 管理ポリシーを含めるか、[equivalent permissions] (同等のアクセス許可) を定義する必要があります。このポリシーは によって維持 AWS され、SiteWise Monitor が AWS IoT SiteWise および IAM Identity Center リソースにアクセスするために使用する一連のアクセス許可を定義します。

SiteWise Monitor ポータルの作成では、そのポータルのユーザーが AWS IoT SiteWise および IAM Identity Center のリソースにアクセスすることを許可するロールを選択する必要があります。 AWS IoT SiteWise コンソールでロールを作成して設定できます。ロールは IAM で後から編集できます。ロールから必要なアクセス許可を削除したり、ロールを削除したりすると、SiteWise Monitor ポータルのユーザーがポータルを使用できなくなります。

> **Note**
>
> 2020 年 4 月 29 日より前に作成されたポータルでは、サービスロールは必須ではありませんでした。この日付より前に作成されたポータルを引き続き使用する場合は、サービスロールをアタッチする必要があります。これを行うには、[AWS IoT SiteWise console] (コンソール) で[Portals] (ポータル) ページに移動し、[Migrate all portals to use IAM roles] (すべてのポータルでIAMロールを使用できるように移行する) を選択します。

次のセクションでは、 AWS Management Console または AWS Command Line Interfaceで SiteWise Monitor のサービスロールを作成および管理する方法について説明します。

目次

- SiteWise Monitor (Classic) のサービスロールのアクセス許可
- SiteWise Monitor のサービスロールのアクセス許可 (AI 対応)
- SiteWise Monitor サービスロールを管理する (コンソール)
  - ポータルのサービスロールを見つける (コンソール)
  - SiteWise Monitor サービスロールを作成する (AWS IoT SiteWise コンソール)
  - SiteWise Monitor サービスロールを作成する (IAM コンソール)
  - ポータルのサービスロールを変更する (コンソール)
- SiteWise Monitor サービスロールを管理する (CLI)
  - ポータルのサービスロールを見つける (CLI)
  - SiteWise Monitor サービスロールを作成する (CLI)
- SiteWise Monitor の AWSIoTSiteWiseMonitorServiceRole に対する更新

SiteWise Monitor (Classic) のサービスロールのアクセス許可

ポータルを作成すると、 は名前が AWSIoTSiteWiseMonitorServiceRole で始まるロール AWS IoT SiteWise を作成します。このロールにより、フェデレーティッド SiteWise Monitor ユーザーはポー

タルの設定、アセット、アセットデータ、IAM Identity Center 設定、にアクセスすることができます。

このロールは、その前提として以下のサービスを信頼します。

- `monitor.iotsitewise.amazonaws.com`

このロールは、AWSIoTSiteWiseMonitorServicePortalPolicy で始まる次のアクセス許可ポリシーを使用して、SiteWise Monitor ユーザーがアカウント内のリソースに対するアクションを完了できるようにします。AWSIoTSiteWiseMonitorPortalAccess 管理ポリシーは、同等のアクセス許可を定義しています。

JSON

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

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotsitewise:CreateProject",
        "iotsitewise:DescribeProject",
```

```
        "iotsitewise:UpdateProject",
        "iotsitewise:DeleteProject",
        "iotsitewise:ListProjects",
        "iotsitewise:BatchAssociateProjectAssets",
        "iotsitewise:BatchDisassociateProjectAssets",
        "iotsitewise:ListProjectAssets",
        "iotsitewise:CreateDashboard",
        "iotsitewise:DescribeDashboard",
        "iotsitewise:UpdateDashboard",
        "iotsitewise:DeleteDashboard",
        "iotsitewise:ListDashboards",
        "iotsitewise:CreateAccessPolicy",
        "iotsitewise:DescribeAccessPolicy",
        "iotsitewise:UpdateAccessPolicy",
        "iotsitewise:DeleteAccessPolicy",
        "iotsitewise:ListAccessPolicies",
        "iotsitewise:DescribeAsset",
        "iotsitewise:ListAssets",
        "iotsitewise:ListAssociatedAssets",
        "iotsitewise:DescribeAssetProperty",
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:GetAssetPropertyValueHistory",
        "iotsitewise:GetAssetPropertyAggregates"
      ],
      "Resource": "*"
    }
  ]
}
```

アラームに必要なアクセス許可の詳細については、でイベントアラームのアクセス許可を設定する AWS IoT SiteWise を参照してください。

ポータルユーザーがサインインすると、SiteWise Monitor は、サービスロールとそのユーザーのアクセスポリシーの交点に基づいて、[session policy] (セッションポリシー) を作成します。アクセスポリシーによって、ポータルおよびプロジェクトへの アイデンティティのアクセスレベルが定義されます。ポータルのアクセス許可とアクセスポリシーの詳細については、「SiteWise Monitor ポータルを管理する」および「CreateAccessPolicy」を参照してください。

SiteWise Monitor のサービスロールのアクセス許可 (AI 対応)

ポータルを作成すると、 は IoTSiteWisePortalRole で始まる名前のロール AWS IoT SiteWise を作成します。このロールにより、フェデレーティッド SiteWise Monitor ユーザーはポータルの設定、アセット、アセットデータ、IAM Identity Center 設定、にアクセスすることができます。

> **Warning**
>
> SiteWise Monitor (AI 対応) では、プロジェクト所有者ロールとプロジェクトビューワーロールはサポートされていません。

このロールは、その前提として以下のサービスを信頼します。

- `monitor.iotsitewise.amazonaws.com`

このロールは、 IoTSiteWiseAIPortalAccessPolicy で始まる次のアクセス許可ポリシーを使用して、SiteWise Monitor ユーザーがアカウントのリソースに対するアクションを完了できるようにします。

JSON

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

ポータルユーザーがサインインすると、SiteWise Monitor は、サービスロールとそのユーザーのアクセスポリシーの交点に基づいて、[session policy] (セッションポリシー) を作成します。

SiteWise Monitor サービスロールを管理する (コンソール)

は、ポータルの SiteWise Monitor サービスロールの管理 AWS IoT SiteWise コンソール を容易にします。ポータルを作成すると、コンソールはアタッチメントに適した既存のロールをチェックしま

す。何も使用できない場合、コンソールはサービスロールを自動的に作成および設定できます。詳細については、「SiteWise Monitor でポータルを作成する」を参照してください。

トピック

- ポータルのサービスロールを見つける (コンソール)
- SiteWise Monitor サービスロールを作成する (AWS IoT SiteWise コンソール)
- SiteWise Monitor サービスロールを作成する (IAM コンソール)
- ポータルのサービスロールを変更する (コンソール)

ポータルのサービスロールを見つける (コンソール)

SiteWise Monitor ポータルにアタッチされているサービスロールを確認するには、次のステップに従います。

ポータルのサービスロールを確認するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左のナビゲーションペインで、[ポータル] を選択します。
3. サービスロールを確認するポータルを選択します。

   ポータルにアタッチされているロールが [アクセス許可]、[サービスロール] で表示されます。

SiteWise Monitor サービスロールを作成する (AWS IoT SiteWise コンソール)

SiteWise Monitor ポータルを作成すると、ポータル用のサービスロールを作成することができます。詳細については、「SiteWise Monitor でポータルを作成する」を参照してください。

AWS IoT SiteWise コンソールで既存のポータルのサービスロールを作成することもできます。これは、ポータルの既存のサービスロールを置き換えるものです。

既存のポータルのサービスロールを作成するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Portals (ポータル)] を選択します。
3. 新しいサービスロールを作成するポータルを選択します。
4. [Portal details (ポータルの詳細)] で、[編集] を選択します。

5. [アクセス許可] で、リストから [Create and use a new service role (新しいサービスロールを作成および使用)] を選択します。
6. 新しいロールの名前を入力します。
7. [保存] を選択します。

SiteWise Monitor サービスロールを作成する (IAM コンソール)

IAM コンソールのサービスロールテンプレートから、サービスロールを作成することができます。このロールテンプレートは、AWSIoTSiteWiseMonitorPortalAccess 管理ポリシーを含み、信頼できるエンティティとして SiteWise モニターを指定します。

ポータルサービスロールテンプレートからサービスロールを作成するステップ。

1. [IAM console] (IAM コンソール) に入ります。
2. ナビゲーションペインで [Roles (ロール)] を選択します。
3. [ロールの作成] を選択してください。
4. [Choose a use case] (ユースケースを選択する) で、[IoT SiteWise]を選択します。
5. [Select your use case] (ユースケースの選択) で、IoT SiteWise Monitor - Portal を選択します。
6. [Next: Permissions (次へ: アクセス許可)] を選択します。
7. [次へ: タグ] を選択します。
8. [次へ: レビュー] を選択します。
9. 新しいサービスロールの[Role name] (ロール名) を入力します。
10. [ロールの作成] を選択してください。

ポータルのサービスロールを変更する (コンソール)

ポータルに SiteWise Monitor の別のサービスロールを選択するには、次のステップに従います。

ポータルのサービスロールを変更するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Portals (ポータル)] を選択します。
3. サービスロールを変更するポータルを選択します。
4. [Portal details (ポータルの詳細)] で、[編集] を選択します。
5. [アクセス許可] で、[Use an existing role (既存のロールを使用)] を選択します。

6. そのポータルにアタッチする既存のロールを選択します。
7. [保存] を選択します。

SiteWise Monitor サービスロールを管理する (CLI)

は、次のポータルサービスロール管理タスク AWS CLI に使用できます。

トピック

- ポータルのサービスロールを見つける (CLI)
- SiteWise Monitor サービスロールを作成する (CLI)

ポータルのサービスロールを見つける (CLI)

SiteWise Monitor ポータルにアタッチされているサービスロールを確認するには、次のコマンドを実行して現在のリージョンのすべてのポータルを一覧表示します。

```
aws iotsitewise list-portals
```

このオペレーションは、ポータルの概要を含むレスポンスを次の形式で返します。

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

ポータルの ID が分かっている場合は、DescribePortal オペレーションを使用してポータルのロールを確認することもできます。

SiteWise Monitor サービスロールを作成する (CLI)

SiteWise Monitor のサービスロールを新しく作成するには、次のステップに従います。

SiteWise Monitor のサービス・ロールを作成するには。

1. SiteWise Monitor にロールの引き受けを許可する信頼ポリシーを設定したロールを作成します。この例では、JSON 文字列に保存された信頼ポリシーから **MySiteWiseMonitorPortalRole** という名前のロールを作成します。

   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. 出力のロールメタデータからロールの ARN をコピーします。ポータルの作成時に、この ARN を使用してロールとポータルを関連付けます。ポータルの作成の詳細については、[AWS IoT SiteWise API Reference] (API リファレンス) の[CreatePortal] (ポータルの作成) を参照してください。

3. a. SiteWise Monitor (Classic) の場合 – `AWSIoTSiteWiseMonitorPortalAccess`ポリシーをロールにアタッチするか、同等のアクセス許可を定義するポリシーをアタッチします。

   ```
   aws iam attach-role-policy --role-name MySiteWiseMonitorPortalRole --policy-arn
    arn:aws:iam::aws:policy/service-role/AWSIoTSiteWiseMonitorPortalAccess
   ```

   b. SiteWise Monitor (AI 対応) の場合 – `IoTSiteWiseAIPortalAccessPolicy`ポリシーをロールにアタッチするか、同等のアクセス許可を定義するポリシーをアタッチします。例え

ば、ポータルアクセス許可を持つポリシーを作成します。次の例では、 という名前のポリシーを作成します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": "*"
        }
    ]
}'
```

既存のポータルにサービスロールをアタッチするには

1. ポータルの既存の詳細を取得するには、次のコマンドを実行します。*portal-id* をポータルの ID で置き換えてください。

```
aws iotsitewise describe-portal --portal-id portal-id
```

このオペレーションは、ポータルの詳細を含むレスポンスを次の形式で返します。

```
{
    "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. サービスロールをポータルにアタッチするには、次のコマンドを実行します。*role-arn* をサービスロールの ARN で置き換え、その他のパラメータをポータルの既存の値で置き換えてください。

```
aws iotsitewise update-portal \
  --portal-id portal-id \
  --role-arn role-arn \
  --portal-name portal-name \
  --portal-description portal-description \
  --portal-contact-email portal-contact-email
```

SiteWise Monitor の AWSIoTSiteWiseMonitorServiceRole に対する更新

本サービスが変更の追跡を開始した時点から、SiteWise Monitor の AWSIoTSiteWiseMonitorServiceRole に対する更新に関する詳細を表示することができます。このページの変更に関する自動アラートについては、 AWS IoT SiteWise ドキュメント履歴ページの RSS フィードにサブスクライブしてください。

変更	説明	日付
AWSIoTSiteWiseMonitorPortalAccess – ポリシーの更新	AWS IoT SiteWise は、アラーム機能の AWSIoTSiteWiseMonitorPortalAccess 管理ポリシーを更新しました。	2021 年 5 月 27 日
AWS IoT SiteWise が変更の追跡を開始しました	AWS IoT SiteWise は、サービスロールの変更の追跡を開始しました。	2020 年 12 月 15 日

でイベントアラームのアクセス許可を設定する AWS IoT SiteWise

AWS IoT Events アラームモデルを使用して AWS IoT SiteWise アセットプロパティをモニタリングする場合は、次の IAM アクセス許可が必要です。

- がデータを送信 AWS IoT Events できるようにする AWS IoT Events サービスロール AWS IoT SiteWise。詳細については、[AWS IoT Events Developer Guide] (デベロッパーガイド) の[Identity and access management for AWS IoT Events] (アイデンティティとアクセス管理) を参照してください。
- および の AWS IoT SiteWise アクション権限が必要です`iotsitewise:DescribeAssetModel``iotsitewise:UpdateAssetModelPropertyRouting`。これらのアクセス許可により、 AWS IoT SiteWise はアセットプロパティ値を AWS IoT Events アラームモデルに送信できます。

詳細については、[IAM User Guide] (IAM ユーザーガイド) の[Resource-based policies] (リソースベースポリシー) を参照してください。

必要なアクセス許可 (API アクション)。

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。

AWS IoT Events アラームモデルを定義する前に、 がアラームモデル AWS IoT SiteWise にアセットプロパティ値を送信できるようにする以下のアクセス許可を付与する必要があります。

- `iotsitewise:DescribeAssetModel`、 `iotsitewise:ListAssetModels` – アセットプロパティが存在するかどうかの確認 AWS IoT Events を に許可します。
- `iotsitewise:UpdateAssetModelPropertyRouting` – AWS IoT SiteWise がデータの送信先となるサブスクリプションを自動的に作成 AWS IoT SiteWise できるようにします AWS IoT Events。

AWS IoT SiteWise サポートされているアクションの詳細については、「サービス認可リファレンス」の「 で定義されるアクション AWS IoT SiteWise」を参照してください。

Example アクセス許可ポリシー 1 の例。

次のポリシーでは、 AWS IoT SiteWise はアセットプロパティ値を任意の AWS IoT Events アラームモデルに送信できます。

JSON

```
{
    "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 アクセス許可ポリシー 2 の例。

次のポリシーでは AWS IoT SiteWise 、 は指定されたアセットプロパティの値を指定された AWS IoT Events アラームモデルに送信できます。

JSON

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

(オプション) ListInputRoutingsの許可。

アセットモデルを更新または削除すると、 AWS IoT SiteWise のアラームモデル AWS IoT Events がこのアセットモデルに関連付けられたアセットプロパティをモニタリングしているかどうかを確認できます。これにより、 AWS IoT Events アラームが現在使用しているアセットプロパティを削除できなくなります。でこの機能を有効にするには AWS IoT SiteWise、 アクセス`iotevents:ListInputRoutings`許可が必要です。このアクセス許可により AWS IoT

SiteWise 、 は でサポートされている ListInputRoutings API オペレーションを呼び出すことができます AWS IoT Events。

> **Note**
>
> `ListInputRoutings` 許可の追加を強く推奨します。

Example アクセス許可ポリシーの例。

次のポリシーでは、アセットモデルを更新および削除し、 で `ListInputRoutings` API を使用できます AWS IoT SiteWise。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:UpdateAssetModel",
                "iotsitewise:DeleteAssetModel",
                "iotevents:ListInputRoutings"
            ],
            "Resource": "arn:aws:iotsitewise:us-east-1:123456789012:asset-model/
*"
        }
    ]
}
```

SiteWise Monitor に必要なアクセス許可。

SiteWise Monitor ポータルでアラーム機能を使用する場合、次のポリシーで[SiteWise Monitor service role] (SiteWise Monitorサービスロール) を更新する必要があります。

JSON

```
{
```

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

でのサービス間の混乱した代理の防止 AWS IoT SiteWise

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、あるサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスが操作され、それ自身のアクセス許可を使用して、本来アクセス許可が付与されるべきではない方法で別の顧客のリソースに対して働きかけることがあります。これを防ぐため、 AWS では、アカウント内のリソースへのアクセス許可が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

リソースポリシーで `aws:SourceArn`および `aws:SourceAccount` グローバル条件コンテキストキーを使用して、 がリソースに別のサービス AWS IoT SiteWise に付与するアクセス許可を制限することをお勧めします。`aws:SourceArn` 値が、Amazon S3 バケットの Amazon リソースネーム (ARN) などのアカウント ID が含まれていない場合、アクセス許可を制限するため、両方のグローバル条件コンテキストキーを使用する必要があります。同じポリシーステートメントでこれらのグローバル条件コンテキストキーの両方を使用し、アカウント ID に`aws:SourceArn` の値が含まれていない場合、`aws:SourceAccount` 値と `aws:SourceArn` 値の中のアカウントには、同じアカウント ID を使用する必要があります。

- クロスサービスのアクセスにリソースを 1 つだけ関連付けたい場合は、`aws:SourceArn` を使用します。
- そのアカウント内のリソースをクロスサービスの使用に関連付けることを許可する場合は、`aws:SourceAccount` を使用します。

の値は、`sts:AssumeRole`リクエストに関連付けられている AWS IoT SiteWise カスタマーリソース`aws:SourceArn`である必要があります。

混乱した代理問題から保護するための最も効果的な方法は、リソースの完全な ARN を指定して `aws:SourceArn` グローバル条件コンテキストキーを使用することです。リソースの ARN 全体が不明または複数のリソースを指定する場合、ARN の未知部分にワイルドカード `*`が付いた `aws:SourceArn` グローバルコンテキスト条件キー を使用します。例えば、`arn:aws:servicename:*:123456789012:*`。

Example - 混乱した代理の防止

次の例は、 で `aws:SourceArn`および `aws:SourceAccount` グローバル条件コンテキストキーを使用して、混乱した代理問題 AWS IoT SiteWise を防ぐ方法を示しています。

JSON

```
{
  "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 ID とアクセスのトラブルシューティング

次の情報は、 AWS IoT SiteWise および AWS Identity and Access Management (IAM) の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちます。

トピック

- でアクションを実行する権限がありません AWS IoT SiteWise
- iam:PassRole を実行する権限がない
- 自分の AWS アカウント以外のユーザーに自分の AWS IoT SiteWise リソースへのアクセスを許可したい

でアクションを実行する権限がありません AWS IoT SiteWise

にアクションを実行する権限がないと AWS Management Console 通知された場合は、管理者に連絡してサポートを依頼する必要があります。担当の管理者はお客様のユーザー名とパスワードを発行した人です。

次のエラー例は、`mateojackson` IAM ユーザーがコンソールを使用してアセットの詳細を表示する際に、`iotsitewise:DescribeAsset` アクセス許可を持っていない場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 iotsitewise:DescribeAsset on resource: a1b2c3d4-5678-90ab-cdef-22222EXAMPLE
```

この場合、Mateo は管理者に依頼し、`iotsitewise:DescribeAsset` アクションを使用して ID が `a1b2c3d4-5678-90ab-cdef-22222EXAMPLE` であるアセットリソースにアクセスできるようにポリシーを更新してもらいます。

`iam:PassRole` を実行する権限がない

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWS IoT SiteWiseにロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して AWS IoT SiteWise でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するに

は、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の AWS アカウント以外のユーザーに自分の AWS IoT SiteWise リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- がこれらの機能 AWS IoT SiteWise をサポートしているかどうかを確認するには、「」を参照してくださいが IAM と AWS IoT SiteWise 連携する方法。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「IAM ユーザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

のコンプライアンス検証 AWS IoT SiteWise

AWS IoT SiteWise は AWS コンプライアンスプログラムの対象ではありません。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスAWS プログラムによる対象範囲内のサービスコンプライアンス」を参照してください。一般的な情報については、AWS 「 Compliance ProgramsAssurance」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS IoT SiteWise は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイする手順について説明します AWS。
- 「Architecting for HIPAA Security and Compliance」ホワイトペーパー – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 「 デベロッパーガイド」のルールを使用してリソースを評価する – この AWS Config サービスは、リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。
- [Ten security golden rules for Industrial IoT solutions] (産業用 IoT ソリューションにおける10のセキュリティ黄金律) - このブログ記事では、産業用制御システム (ICS) 、産業用 IoT (IIoT) 、およびクラウド環境のセキュリティに役立つ 10 の黄金律を紹介します。
- 製造 OT のセキュリティのベストプラクティス – このホワイトペーパーでは、 AWS クラウド用のこれらのオンプレミスハイブリッド製造ワークロードを設計、デプロイ、設計するためのセキュリティのベストプラクティスについて説明します。

の耐障害性 AWS IoT SiteWise

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された複数の物理的に分離されたアベイラビリティーゾーンと分離されたアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS IoT SiteWise はフルマネージド型で、Amazon S3 や Amazon EC2 などの可用性と耐久性の高い AWS サービスを使用しています。アベイラビリティーゾーンが中断した場合に可用性を確保するために、 は複数のアベイラビリティーゾーンで AWS IoT SiteWise 動作します。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

グローバル AWS インフラストラクチャに加えて、 AWS IoT SiteWise には、データの耐障害性とバックアップのニーズをサポートするのに役立ついくつかの機能が用意されています。

- MQTT メッセージ AWS IoT Core を通じて にプロパティ値の更新を発行し、そのデータに基づいて動作するようにルールを設定できます。この機能を使用すると、Amazon S3 や Amazon DynamoDB などの他の AWS サービスでデータをバックアップできます。詳細については、「他の AWS サービスとやり取りする」および「アセットプロパティ通知とともにデータを Amazon S3 にエクスポートする」を参照してください。
- APIs AWS IoT SiteWise `Get*`、アセットプロパティの履歴データを取得およびバックアップできます。詳細については、「の履歴アセットプロパティ値をクエリする AWS IoT SiteWise」を参照してください。
- APIs を使用して AWS IoT SiteWise `Describe*`、アセットやモデルなどのリソースの定義を取得できます。これらの定義をバックアップし、後でそれらを使用してリソースを再作成できます。詳細については、「AWS IoT SiteWise APIリファレンス」を参照してください。

のインフラストラクチャセキュリティ AWS IoT SiteWise

マネージドサービスである AWS IoT SiteWise は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュ

リティのベストプラクティスを使用して AWS 環境を設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS が公開した API コールを使用して、ネットワーク AWS IoT SiteWise 経由で にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードはJava 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。またはAWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

SiteWise Edge ゲートウェイは、X.509 証明書と暗号化キー AWS IoT Greengrassを使用して AWS クラウドに接続し、認証します。詳細については、「AWS IoT Greengrass Version 1 デベロッパーガイド」の「AWS IoT Greengrassのデバイス認証と認可」を参照してください。

での設定と脆弱性の分析 AWS IoT SiteWise

IoT フリートは、多様な機能を持ち、存続期間が長く、地理的に分散される多数のデバイスで構成されることがあります。このような特性によってフリートのセットアップが複雑になり、エラーを起こしやすくなります。通常、デバイスの処理能力、メモリ、およびストレージは限られているため、暗号化やその他のセキュリティ対策を常にサポートできるとは限りません。さらに、デバイスは多く場合、既知の脆弱性を持つソフトウェアを使用しています。このような要素が原因で、IoT フリートはハッカーの魅力的なターゲットとなり、デバイスフリートを継続的に保護することが困難になっています。

AWS IoT Device Defender セキュリティの問題やベストプラクティスからの逸脱を特定するためのツールを提供することで、これらの課題に対処します。 AWS IoT Device Defender を使用して、接続されたデバイスを分析、監査、モニタリングして異常な動作を検出し、セキュリティリスクを軽減します。 AWS IoT Device Defender は、デバイスフリートを監査して、セキュリティのベストプラクティスに準拠し、デバイスの異常な動作を検出できます。これにより、 AWS IoT デバイスフリート全体に一貫したセキュリティポリシーを適用し、デバイスが侵害されたときに迅速に対応できます。詳細については、「AWS IoT Device Defender デベロッパーガイド」の「AWS IoT Device Defenderとは」を参照してください。

SiteWise Edge ゲートウェイを使用してサービスにデータを取り込む場合は、SiteWise Edge ゲートウェイの環境を設定および保守する責任があります。この責任には、SiteWise Edge ゲートウェイのシステムソフトウェア、 AWS IoT Greengrass ソフトウェア、コネクタの最新バージョンへのアップグレードが含まれます AWS IoT SiteWise 。詳細については、「 AWS IoT Greengrass Version 1 デベロッパーガイド」の AWS IoT Greengrass 「 コアを設定する」および「」を参照してくださいSiteWise Edge ゲートウェイを管理。

の VPC エンドポイント AWS IoT SiteWise

インターフェイス VPC エンドポイントは、仮想プライベートクラウド (VPC) と の間にプライベート接続を確立します AWS IoT SiteWise。 はインターフェイスエンドポイントAWS PrivateLinkを強化し、API オペレーションへの AWS IoT SiteWise プライベートアクセスを有効にします。 は、インターフェイスエンドポイントを介して IPv4 と IPv6 (デュアルスタック) の両方 AWS IoT SiteWise をサポートします。インターネットゲートウェイ、NAT デバイス、VPN 接続、または AWS Direct Connectの必要性を迂回できます。VPC 内のインスタンスは、パブリック IP アドレスがなくても AWS IoT SiteWise API と通信できます。VPC と の間のトラフィック AWS IoT SiteWise はネットワークを AWS 離れません。

各インターフェイスエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。

のインターフェイス VPC エンドポイントを設定する前に AWS IoT SiteWise、「 AWS PrivateLink ガイド」の「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」を参照してください。

AWS IoT SiteWiseでの VPC エンドポイントの API オペレーション

AWS IoT SiteWise では、VPC から次の AWS IoT SiteWise API オペレーションを呼び出すことができます。

- すべてのデータプレーン API オペレーションでは、以下のエンドポイントを使用します。*region* をユーザーの AWS リージョンに置き換えてください。

  ```
  data.iotsitewise.region.amazonaws.com
  ```

  データプレーン API には、以下のオペレーションが含まれます。

  - BatchGetAssetPropertyValue
  - BatchGetAssetPropertyValueHistory

- BatchPutAssetPropertyValue
- GetAssetPropertyAggregates
- GetAssetPropertyValue
- GetAssetPropertyValueHistory
- GetInterpolatedAssetPropertyValues

- アセットモデル、アセット、SiteWise Edge ゲートウェイ、タグ、およびアカウント設定の管理に使用するコントロールプレーン API オペレーションについては、以下のエンドポイントを使用します。*region* の部分はお客様の AWS リージョンに置き換えてください。

```
api.iotsitewise.region.amazonaws.com
```

サポートされているコントロールプレーン API オペレーションには以下が含まれます。

- 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**
>
> コントロールプレーン API オペレーションのインターフェイス VPC エンドポイントは現在、以下の SiteWise Monitor API オペレーションの呼び出しをサポートしていません。
>
> - BatchAssociateProjectAssets
> - BatchDisassociateProjectAssets
> - CreateAccessPolicy
> - CreateDashboard
> - CreatePortal
> - CreateProject
> - DeleteAccessPolicy
> - DeletePortal
> - DeleteProject
> - DescribeAccessPolicy
> - DescribePortal

- DescribeProject
- ListAccessPolicies
- ListDashboards
- ListPortals
- ListProjects
- ListProjectAssets
- UpdateAccessPolicy
- UpdateDashboard
- UpdatePortal
- UpdateProject

AWS IoT SiteWiseのインターフェイス VPC エンドポイントの作成

AWS IoT SiteWise サービスの VPC エンドポイントを作成するには、Amazon VPC コンソールまたは AWS Command Line Interface () を使用しますAWS CLI。詳細については、「 AWS PrivateLink ガイド」の「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」を参照してください。

次のいずれかのサービス名を使用して AWS IoT SiteWise 、 の VPC エンドポイントを作成します。

- データプレーン API オペレーションの場合、次のサービス名を使用します。

```
com.amazonaws.region.iotsitewise.data
```

- コントロールプレーン API オペレーションの場合、次のサービス名を使用します。

```
com.amazonaws.region.iotsitewise.api
```

インターフェイス VPC エンドポイント AWS IoT SiteWise を介したアクセス

インターフェイスエンドポイントを作成すると、通信に使用できるエンドポイント固有の DNS ホスト名が生成されます AWS IoT SiteWise。プライベート DNS のオプションはデフォルトで有効に

なっています。詳細については、「Amazon VPC ユーザーガイド」の「プライベートホストゾーンの使用」を参照してください。

エンドポイントのプライベート DNS を有効にすると、次のいずれかの VPC エンドポイント AWS IoT SiteWise を介して に API リクエストを行うことができます。

- データプレーン API オペレーションの場合、以下のエンドポイントを使用します。*region* は実際の AWS リージョンに置き換えてください。

```
data.iotsitewise.region.amazonaws.com
```

- コントロールプレーン API オペレーションの場合、以下のエンドポイントを使用します。*region* は実際の AWS リージョンに置き換えてください。

```
api.iotsitewise.region.amazonaws.com
```

エンドポイントのプライベート DNS を無効にする場合は、エンドポイント AWS IoT SiteWise 経由で にアクセスするには、以下を実行する必要があります。

1. API リクエストで VPC エンドポイント URL を指定します。
   - データプレーン API オペレーションには、次のエンドポイント URL を使用します。*#vpc-endpoint-id#* と *#######* は、VPC エンドポイント ID とリージョンに置き換えてください。

     ```
     vpc-endpoint-id.data.iotsitewise.region.vpce.amazonaws.com
     ```

   - コントロールプレーン API オペレーションには、次のエンドポイント URL を使用します。*#vpc-endpoint-id#* と *#######* は、VPC エンドポイント ID とリージョンに置き換えてください。

     ```
     vpc-endpoint-id.api.iotsitewise.region.vpce.amazonaws.com
     ```

2. ホストプレフィックスインジェクションを無効にする。 AWS CLI および AWS SDKs は、各 API オペレーションを呼び出すときに、サービスエンドポイントにさまざまなホストプレフィックスを付加します。この機能を使用すると、 AWS CLI VPC エンドポイントを指定 AWS IoT SiteWise したときに に対して有効でない URLs が および AWS SDKs によって生成されます。

> **Important**
>
> AWS CLI または でホストプレフィックスインジェクションを無効にすることはできません AWS Tools for PowerShell。つまり、プライベート DNS を無効にすると、これらのツールを使用して VPC エンドポイント AWS IoT SiteWise 経由で にアクセスできなくなります。プライベート DNS を有効にして、 AWS CLI または AWS Tools for PowerShell を使用してエンドポイント AWS IoT SiteWise 経由で にアクセスします。

AWS SDKs でホストプレフィックスインジェクションを無効にする方法の詳細については、各 SDK の以下のドキュメントセクションを参照してください。

- 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

詳細については、「 AWS PrivateLink ガイド」の「インターフェイス VPC エンドポイントを使用して AWS サービスにアクセスする」を参照してください。

の VPC エンドポイントポリシーを作成する AWS IoT SiteWise

VPC エンドポイントには、 AWS IoT SiteWiseへのアクセスを制御するエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- オペレーションを実行できるプリンシパル。
- 実行できる操作。
- この操作を実行できるリソース。

詳細については、「Amazon VPC ユーザーガイド」の「エンドポイントポリシーを使用して VPC エンドポイントへのアクセスを制御する」を参照してください。

例: AWS IoT SiteWise アクションの VPC エンドポイントポリシー

以下は、 のエンドポイントポリシーの例です AWS IoT SiteWise。エンドポイントにアタッチすると、このポリシーは、指定されたアセットの AWS アカウント *123456789012* *iotsitewiseadmin*のユーザーに、リストされた AWS IoT SiteWise アクションへのアクセスを許可します。

```
{
    "Statement": [
        {
            "Action": [
                "iotsitewise:CreateAsset",
                "iotsitewise:ListGateways",
                "iotsitewise:ListTagsForResource"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iotsitewise:us-west-2:123456789012:asset/
a1b2c3d4-5678-90ab-cdef-33333EXAMPLE",
            "Principal": {
                "AWS": [
                    "123456789012:user/iotsitewiseadmin"
                ]
            }
        }
    ]
}
```

のセキュリティのベストプラクティス AWS IoT SiteWise

このトピックには、 のセキュリティのベストプラクティスが含まれています AWS IoT SiteWise。

OPC UA サーバーで認証情報を使用する

OPC UA サーバーへの接続には認証情報を要求します。これを行うには、サーバーのドキュメントを参照してください。次に、SiteWise Edge ゲートウェイが OPC UA サーバーに接続できるようにするには、SiteWise Edge ゲートウェイにサーバー認証シークレットを追加します。詳細については、「SiteWise Edge のデータソース認証を設定する」を参照してください。

OPC UA サーバーに暗号化通信モードを使用

SiteWise Edge ゲートウェイの OPC UA ソースを設定する際に、非推奨の暗号化メッセージセキュリティモードを選択します。これにより、OPC UA サーバーから SiteWise Edge ゲートウェイに移動する際の産業用データのセキュリティを確保することができます。詳細については、「ローカルネットワーク経由で転送されるデータ」および「SiteWise Edge で OPC UA ソースを設定する」を参照してください。

コンポーネントを最新の状態に保つ。

SiteWise Edge ゲートウェイを使用してサービスにデータを取り込む場合は、SiteWise Edge ゲートウェイの環境を設定および保守する責任があります。これには、ゲートウェイのシステムソフトウェア、 AWS IoT Greengrass ソフトウェアとコネクタの最新バージョンへのアップグレードが含まれます。

> **Note**
>
> AWS IoT SiteWise Edge コネクタは、シークレットをファイルシステムに保存します。これらのシークレットは、SiteWise Edge ゲートウェイ内にキャッシュされたデータを表示できるユーザーを制御します。SiteWise Edge ゲートウェイを実行しているシステムのディスクまたはファイルシステムの暗号化を有効にすることを強くお勧めします。

AWS IoT SiteWise コンソールでコンポーネントをアップグレードする方法については、「」を参照してくださいSiteWise Edge ゲートウェイコンポーネントパックのバージョンを変更する。

SiteWise Edge ゲートウェイのファイルシステムの暗号化

SiteWise Edge ゲートウェイを暗号化して保護することにより、産業用データが SiteWise Edge ゲートウェイを通過するときに保護されます。SiteWise Edge ゲートウェイにハードウェアセキュリティモジュールがある場合は、SiteWise Edge ゲートウェイを保護する AWS IoT Greengrass ように を設定できます。詳細については、[AWS IoT Greengrass Version 1 Developer Guide] (デベロッパーガイド) の[Hardware security integration] (ハードウェアセキュリティの統合) を参照してください。ハードウェアセキュリティモジュールがない場合は、オペレーティングシステムのドキュメントを参照して、ファイルシステムを暗号化して保護する方法を確認してください。

エッジ設定へのセキュアなアクセス。

エッジコンソールアプリケーションのパスワードや SiteWise Monitor アプリケーションのパスワードは共有しないでください。このパスワードは、誰でも見られる場所に置かないでください。パスワードに適切な有効期限を設定することで、健全なパスワードローテーションポリシーを実施する。

Siemens Industrial Edge Management のデータの保護

AWS IoT SiteWise Edge と共有することを選択するデバイスデータは、Siemens IEM Databus設定トピックで決定されます。SiteWise Edge と共有するトピックを選択することにより、トピックレベルのデータを AWS IoT SiteWiseと共有します。Siemens Industrial Edge Marketplace は独立したマーケットプレイスであり、 とは別のものです AWS。共有データを保護するために、ユーザーが Siemens Secured Storage を使用しない限り、SiteWise Edge アプリケーションは実行されません。詳細については、Siemens ドキュメントの「セキュアストレージ」を参照してください。

SiteWise Monitor ユーザーへの最小限のアクセス許可の付与

最小アクセス許可の原則に従って、ポータルのユーザーのアクセスポリシーのアクセス許可には最小限のものを使用します。

- ポータルを作成する場合は、そのポータルに必要な最小限のアセットを許可するロールを定義します。詳細については、「のサービスロールを使用する AWS IoT SiteWise Monitor」を参照してください。
- ポータル管理者がプロジェクトを作成して共有する場合は、そのプロジェクトに必要な最小限のアセットを使用します。
- ポータルやプロジェクトにアクセスする必要がなくなった ID は、そのリソースから削除する。その ID が組織に適用されなくなった場合は、ID ストアからその ID を削除してください。

最小アクセス許可の原則のベストプラクティスは、IAM ロールにも該当します。詳細については、「ポリシーに関するベストプラクティス」を参照してください。

機密情報を公開しない

認証情報やその他の機密情報 (個人を特定できる情報 (PII) など) のログへの記録を防止する必要があります。SiteWise Edge ゲートウェイのローカルログへのアクセスを経由する場合でも、ルート権限が必要であり、CloudWatch Logs へのアクセスには IAM アクセス許可が必要ですが、次の防止策を実施することをお勧めします。

- アセットやモデルの名前、説明、プロパティに機密情報を使用しない。
- SiteWise Edge ゲートウェイやソース名に機密情報を使用しないでください。
- ポータル、プロジェクト、ダッシュボードの名前や説明に機密情報を使用しない。

AWS IoT Greengrass セキュリティのベストプラクティスに従う

SiteWise Edge ゲートウェイ AWS IoT Greengrass のセキュリティのベストプラクティスに従います。詳細については、[AWS IoT Greengrass Version 1 Developer Guide] (デベロッパーガイド) の [Security best practices] (セキュリティベストプラクティス) を参照してください。

関連情報

- [AWS IoT Developer Guide] (デベロッパーガイド) の [Security best practices] (セキュリティベストプラクティス)。
- Ten security golden rules for IoT solutions (ブログ記事)

でのログ記録とモニタリング AWS IoT SiteWise

モニタリングは、 AWS IoT SiteWise およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 は、サービスを監視し、問題が発生したときに報告し、必要に応じて自動アクションを実行するために、以下のモニタリングツール AWS IoT SiteWise をサポートしています。

- Amazon CloudWatch は、 AWS リソースと で実行しているアプリケーションを AWS リアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定されたメトリクスが特定のしきい値に達したときに通知したりアクションを実行したりするアラームの設定を行います。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon CloudWatch Logs は、SiteWise Edge ゲートウェイ、CloudTrail、およびその他のソースのログファイルをモニタリング、保存、アクセスします。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、「Amazon CloudWatch Logs ユーザーガイド」を参照してください。
- AWS CloudTrail は、 AWS アカウントによって、またはアカウントに代わって行われた API コールおよび関連イベントをキャプチャします。次に、CloudTrail は、ユーザーが指定した Amazon S3 バケットにログファイルを配送します。が呼び出したユーザーとアカウント AWS、呼び出し元のソース IP アドレス、呼び出しの発生日時を特定できます。詳細については、AWS CloudTrail ユーザーガイドをご参照ください。

トピック

- Amazon CloudWatch Logs によるモニタリング
- SiteWise Edge ゲートウェイログをモニタリングする
- Amazon CloudWatch メトリクス AWS IoT SiteWise によるモニタリング
- を使用した AWS IoT SiteWise API コールのログ記録 AWS CloudTrail

Amazon CloudWatch Logs によるモニタリング

CloudWatch Logs に情報を AWS IoT SiteWise ログ記録して、サービスをモニタリングおよびトラブルシューティングするように を設定します。

AWS IoT SiteWise コンソールを使用すると、 はサービスにリンクされたロール AWS IoT SiteWise を作成し、サービスがユーザーに代わって情報をログ記録できるようにします。 AWS IoT SiteWise コンソールを使用しない場合は、サービスにリンクされたロールを手動で作成してログを受信する必要があります。詳細については、「AWS IoT SiteWiseサービスにリンクされたロールを作成する」を参照してください。

が CloudWatch ストリームにログイベント AWS IoT SiteWise を配置できるようにするリソースポリシーが必要です。CloudWatch Logs のリソースポリシーを作成、更新する場合は、次のコマンドを実行します。*logging-policy-name* は、作成するポリシーの名前に置き換えてください。

```
aws logs put-resource-policy --policy-name logging-policy-name --policy-
document "{ \"Version\": \"2012-10-17\", \"Statement\": [ { \"Sid\":
 \"IoTSiteWiseToCloudWatchLogs\", \"Effect\": \"Allow\", \"Principal\": { \"Service\":
 [ \"iotsitewise.amazonaws.com\" ] }, \"Action\":\"logs:PutLogEvents\", \"Resource\":
 \"*\" } ] }"
```

CloudWatch Logs は、aws:SourceArn と aws:SourceAccount の条件コンテキストキーもサポートしています。これらの条件コンテクストキーはオプションです。

が指定されたリソースに関連付けられたログ AWS IoT SiteWise のみを CloudWatch ストリームに配置することを許可する AWS IoT SiteWise リソースポリシーを作成または更新するには、 コマンドを実行し、以下を実行します。

- *logging-policy-name* は、作成するポリシーの名前に置き換えてください。
- *source-ARN* をアセットモデルやアセットなどの AWS IoT SiteWise リソースの ARN に置き換えます。各 AWS IoT SiteWise リソースタイプの ARN を確認するには、「サービス認可リファレンス」の「 で定義されるリソースタイプ AWS IoT SiteWise」を参照してください。
- *account-ID* を、指定された AWS IoT SiteWise リソースに関連付けられた AWS アカウント ID に置き換えます。

```
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\"]}}}]}"
```

デフォルトでは、 AWS IoT SiteWise は CloudWatch Logs に情報をログに記録しません。ログ記録を有効にするには、Disabled () 以外のログ記録レベルを選択しますOFF。 は、次のログ記録レベル AWS IoT SiteWise をサポートします。

- OFF – ログ記録はオフになります。
- ERROR - エラーがログ記録されます。
- INFO - エラーおよび情報メッセージがログ記録されます。

AWS IoT Greengrassを介して CloudWatch Logs に情報をログ記録するように SiteWise Edge ゲートウェイを設定できます。詳細については、「SiteWise Edge ゲートウェイログをモニタリングする」を参照してください。

AWS IoT SiteWise ルールアクションのトラブルシューティングを行う場合は、CloudWatch Logs に情報をログ AWS IoT Core 記録するように を設定することもできます。詳細については、「AWS IoT SiteWise ルールアクションのトラブルシューティング」を参照してください。

目次

- ログインの管理 AWS IoT SiteWise
  - ログ記録レベルを検索する
  - ログ記録レベルを変更する
- 例: AWS IoT SiteWise ログファイルエントリ

ログインの管理 AWS IoT SiteWise

AWS IoT SiteWise コンソールまたは を使用して、次のログ記録設定タスク AWS CLI を実行します。

ログ記録レベルを検索する

Console

AWS IoT SiteWise コンソールで現在のログ記録レベルを確認するには、次の手順に従います。

現在の AWS IoT SiteWise ログ記録レベルを確認するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションペインで、[ログ記録オプション] を選択します。

[ロギングステータス] に、現在のログ記録ステータスが表示されます。ログ記録が有効な場合は、現在のログ記録レベルが [詳細レベル] の下に表示されます。

AWS CLI

次のコマンドを実行して、 で現在の AWS IoT SiteWise ログ記録レベルを見つけます AWS CLI。

```
aws iotsitewise describe-logging-options
```

このオペレーションは、ログ記録レベルを次の形式で返します。

```
{
  "loggingOptions": {
    "level": "String"
  }
}
```

ログ記録レベルを変更する

AWS IoT SiteWise コンソールまたは を使用してログ記録レベルを変更するには、次の手順に従います AWS CLI。

Console

AWS IoT SiteWise ログ記録レベルを変更するには

1. AWS IoT SiteWise コンソールに移動します。
2. 左側のナビゲーションペインで、[Logging options (ログ記録オプション)] を選択します。
3. [編集] を選択します。
4. 有効にする [詳細レベル] を選択します。
5. [Save] を選択します。

AWS CLI

次の AWS CLI コマンドを実行して、 AWS IoT SiteWise ログ記録レベルを変更します。*logging-level* は、必要なログ記録レベルに置き換えます。

```
aws iotsitewise put-logging-options --logging-options level=logging-level
```

例: AWS IoT SiteWise ログファイルエントリ

各 AWS IoT SiteWise ログエントリには、イベント情報とそのイベントの関連リソースが含まれているため、ログデータを理解して分析できます。

次の例は、アセットモデルを正常に作成したときに AWS IoT SiteWise ログに記録する CloudWatch Logs エントリを示しています。

```
{
  "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 ゲートウェイログをモニタリングする

Amazon CloudWatch Logs またはローカルファイルシステムに情報をログ記録するように AWS IoT SiteWise Edge ゲートウェイを設定できます。

トピック

- Amazon CloudWatch Logs を使用
- でサービスログを使用する AWS IoT SiteWise

Amazon CloudWatch Logs を使用

CloudWatch Logs にログを送信するように SiteWise Edge ゲートウェイを設定できます。詳細については、「AWS IoT Greengrass Version 2 デベロッパーガイド」の「CloudWatch Logs のログ記録を有効化」を参照してください。

CloudWatch Logs を設定してアクセスするには (コンソール)

1. [CloudWatch console] (CloudWatch のコンソール) に移動する。

2. ナビゲーションペインで、[ロググループ] を選択します。
3. AWS IoT SiteWise コンポーネントログは、次のロググループにあります。

   - `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgeCollectorOpcua` – SiteWise Edge ゲートウェイの OPC UA ソースからデータを収集する SiteWise Edge ゲートウェイのコンポーネントのログ。
   - `/aws/greengrass/UserComponent/region/aws.iot.SiteWiseEdgePublisher` – OPC UA データストリームを発行する SiteWise Edge ゲートウェイのコンポーネントのログ AWS IoT SiteWise。

   デバッグする関数のロググループを選択します。

4. AWS IoT Greengrass グループの名前で終わる名前のログストリームを選択します。CloudWatch では、デフォルトで最新のログストリームが最初に表示されます。

5. 過去 5 分間のログを表示するには、以下を実行します。

   a. 右上隅の [カスタム] を選択します。
   b. [相対値] を選択します。
   c. [5 分] を選択します。
   d. [適用] を選択します。

6. (オプション) 表示するログの数を減らすには、右上隅で [1 分] を選択します。
7. ログエントリの一番下までスクロールして、最新のログを表示します。

でサービスログを使用する AWS IoT SiteWise

SiteWise Edge ゲートウェイデバイスには、問題のデバッグに役立つサービスログファイルが含まれています。以下のセクションは、OPC UA Collector および AWS IoT SiteWise Publisher AWS IoT SiteWise コンポーネントのサービスログファイルを検索して使用するのに役立ちます。

AWS IoT SiteWise OPC UA Collector サービスログファイル

OPC UA Collector AWS IoT SiteWise コンポーネントは、次のログファイルを使用します。

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log
```

このコンポーネントのログを確認するには

- コアデバイスに次のコマンドを実行して、このコンポーネントのログファイルをリアルタイムに確認します。*/greengrass/v2* または *C:\greengrass\v2* を AWS IoT Greengrass ルートフォルダへのパスに置き換えます。

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log -Tail
   10 -Wait
  ```

AWS IoT SiteWise パブリッシャーサービスログファイル

AWS IoT SiteWise Publisher コンポーネントは、次のログファイルを使用します。

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log
```

このコンポーネントのログを確認するには

- コアデバイスに次のコマンドを実行して、このコンポーネントのログファイルをリアルタイムに確認します。*/greengrass/v2* または *C:\greengrass\v2* を AWS IoT Greengrass ルートフォルダへのパスに置き換えます。

  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 メトリクス AWS IoT SiteWise によるモニタリング

CloudWatch AWS IoT SiteWise を使用してCloudWatch は raw データを収集し、読み取り可能なほぼリアルタイムのメトリクスに加工します。これらの統計は 15 か月間保持されるため、履歴情報にアクセスし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

AWS IoT SiteWise は、以下のセクションにリストされているメトリクスとディメンションを `AWS/IoTSiteWise`名前空間に発行します。

> **Tip**
> AWS IoT SiteWise は 1 分間隔でメトリクスを発行します。CloudWatch コンソールでこれらのメトリクスをグラフで表示する場合は、[Period] (期間) を [1 minute] (1 分) にすることが推奨されます。これにより、メトリクスデータが最も高い解像度で表示されます。

トピック

- AWS IoT Greengrass Version 2 ゲートウェイメトリクス

AWS IoT Greengrass Version 2 ゲートウェイメトリクス

AWS IoT SiteWise は、Classic ストリーム、V2 ゲートウェイ、および MQTT 対応 V3 ゲートウェイのゲートウェイメトリクスを発行します。特に明記されていない限り、各メトリクスは両方のセルフホストゲートウェイバージョンに適用されます。すべての SiteWise Edge ゲートウェイメトリクスは、1 分間隔で発行されます。

SiteWise Edge ゲートウェイメトリクス

メトリクス	説明
`Gateway.AvailableMemory`	SiteWise Edge ゲートウェイの使用可能メモリ。 単位: バイト ディメンション: なし
`Gateway.AvailableDiskSpace`	SiteWise Edge ゲートウェイの使用可能ディスク容量。 単位: バイト ディメンション: なし
`Gateway.CloudConnectivity`	SiteWise Edge ゲートウェイのクラウド接続ステータス。 単位: なし ディメンション: GatewayId
`Gateway.CpuUsage`	SiteWise Edge ゲートウェイの CPU 使用率。 単位: パーセント ディメンション: なし
`Gateway.TotalDiskSpace`	SiteWise Edge ゲートウェイの合計ディスク容量。 単位: バイト ディメンション: なし
`Gateway.TotalMemory`	SiteWise Edge ゲートウェイの合計メモリ。 単位: バイト

メトリクス	説明
	ディメンション: なし
`Gateway.UsedDiskSpace`	SiteWise Edge ゲートウェイの使用済みディスク容量。 単位: バイト ディメンション: なし
`Gateway.UsedMemory`	SiteWise Edge ゲートウェイの使用済みメモリ。 単位: バイト ディメンション: なし
`Gateway.UsedPercentageDiskSpace`	SiteWise Edge ゲートウェイのディスク容量の使用率。 単位: バイト ディメンション: なし
`Gateway.UsedPercentageMemory`	SiteWise Edge ゲートウェイのメモリ使用率。 単位: バイト ディメンション: なし

AWS IoT SiteWise パブリッシャーメトリクス

メトリクス	説明
`IoTSiteWisePublisher.ComponentBuildVersion`	このメトリクスは、ゲートウェイで実行されている IoT SiteWise パブリッシャーコンポーネントのビルドバージョンを示します。値 1 は、ゲートウェイが`ComponentBuildVersion` ディメンションに対応するパブリッシャ

メトリクス	説明
	ーのバージョンを実行していることを示します。 単位: 1 ディメンション: GatewayId、ComponentBuildVersion
`IoTSiteWisePublisher.DroppedCount`	SiteWise Edge ゲートウェイ (`GatewayId`) によってドロップされ、クラウドに発行されなかったデータポイントの数。1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId
`IoTSiteWisePublisher.Heartbeat`	SiteWise Edge ゲートウェイでパブリッシャーによって 1 分ごとに生成されます。 単位: 1 (1 は、パブリッシャーが実行中であることを表し、データポイントが欠落しているのは、パブリッシャーが実行していないことを表します。) ディメンション: GatewayId
`IoTSiteWisePublisher.IsConnectedToMqttBroker`	SiteWise Edge ゲートウェイでパブリッシャーによって 1 分ごとに生成されます。 単位: 1 (1 は、パブリッシャーが MQTT ブローカーに接続されていることを表します。) ディメンション: GatewayId

メトリクス	説明
IoTSiteWisePublisher.MessageCheckpointPersistenceErrorCount	メトリクスは、ゲートウェイがパブリッシャーによって処理されたデータの追跡に使用されるチェックポイントファイルに関する問題を検出したことを示します。1 の値は、障害が発生したことを示します。 単位: なし ディメンション: AccountId、GatewayId
IoTSiteWisePublisher.MqttMessageReceivedSuccessCount	パブリッシャーが MQTT ブローカーから正常に受信したメッセージの数。1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId
IoTSiteWisePublisher.MqttReceivedSuccessBytes	パブリッシャーが MQTT ブローカーから正常に受信したメッセージデータのバイト数。1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId
IoTSiteWisePublisher.NumberOfSubscriptionsToMqttBroker	パブリッシャーによって MQTT ブローカーにサブスクライブされたトピックの数。1 分ごとに生成されます。マルチレベルワイルドカードトピックは 1 としてカウントされます。 単位: 数 ディメンション: GatewayId

メトリクス	説明
`IoTSiteWisePublisher.NumberOfUniqueMqttTopicsReceived`	パブリッシャーが MQTT ブローカーから受信した一意のトピックの数。1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId
`IoTSiteWisePublisher.PublishFailureCount`	SiteWise Edge ゲートウェイ (`GatewayId`) が発行に失敗したデータポイントの数。1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId
`IoTSiteWisePublisher.PublishRejectedCount`	SiteWise Edge ゲートウェイ (`GatewayId`) がクラウド側から拒否されたデータポイントの数。1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId
`IoTSiteWisePublisher.PublishSuccessCount`	SiteWise Edge ゲートウェイ (`GatewayId`) がクラウドに正常に発行したデータポイントの数。1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId
`IoTSiteWisePublisher.PublishToS3FailureCount`	ゲートウェイ (`GatewayId`) が Amazon S3 バケットへの発行に失敗したデータポイントの数。 単位: 数 ディメンション: GatewayId

メトリクス	説明
`IoTSiteWisePublisher.PublishToS3SuccessCount`	ゲートウェイ (`GatewayId`) が Amazon S3 バケットに正常に発行したデータポイントの数。 単位: 数 ディメンション: GatewayId

OPC UA コレクターメトリクス

メトリクス	説明
`OpcUaCollector.ActiveDataStreamCount`	SiteWise Edge ゲートウェイ (`gatewayId`) が OPC UA ソース (`sourceName`) に対してサブスクライブしたデータストリームの数。 単位: 数 ディメンション: GatewayId、SourceName、PropertyGroup
`OpcUaCollector.ComponentBuildVersion` (Classic ストリーム、V2 ゲートウェイでは利用できません)	このメトリクスは、ゲートウェイで実行されている IoT SiteWise OPC UA コレクターコンポーネントのビルドバージョンを示します。値 1 は、ゲートウェイが`ComponentBuildVersion` ディメンションに対応するコレクターのバージョンを実行していることを示します。 単位: 1 ディメンション: GatewayId、ComponentBuildVersion
`OpcUaCollector.ConversionErrors`	SiteWise Edge ゲートウェイ (`gatewayId`) が OPC UA ソース (`sourceName`) に対して受信し、データの送信中に変換エラーが発生したデータポイントの数 AWS IoT SiteWise。これ

メトリクス	説明
	らのデータポイントは OPC-UA コレクターによって取り込まれません。 単位: 数 ディメンション: GatewayId、SourceName
`OpcUaCollector.Heartbeat`	SiteWise Edge ゲートウェイ (`gatewayId`) に接続された各 OPC UA ソース (`sourceName`) について 1 分ごとに生成されます。 ユニット: カウント (1 はソースが接続されていることを表し、0 はソースが切断されていることを表します。) ディメンション: GatewayId、SourceName
`OpcUaCollector.IncomingValuesCount`	SiteWise Edge ゲートウェイ (`gatewayId`) が OPC UA ソース (`sourceName`) について受信したデータポイントの数。1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId、SourceName、PropertyGroup
`OpcUaCollector.IncomingValueErrors`	SiteWise Edge ゲートウェイ (`gatewayId`) が OPC UA ソース (`sourceName`) から受信し、値が無効であったデータポイントの数。これらのデータポイントは、OPC UA コレクターによって取り込まれず、1 分ごとに生成されます。 単位: 数 ディメンション: GatewayId、SourceName、PropertyGroup

メトリクス	説明
`OpcUaCollector.IsConnectedToMqttBroker` (Classic ストリーム、V2 ゲートウェイでは利用できません)	SiteWise Edge ゲートウェイの IoT SiteWise OPC UA SiteWise コレクターコンポーネントによって毎分生成されます。 SiteWise 単位: 1 (IoT SiteWise OPC UA コレクターコンポーネントが MQTT ブローカーに接続されていることを示す 1) ディメンション: GatewayId
`OpcUaCollector.MqttMessagesDroppedCount` (Classic ストリーム、V2 ゲートウェイでは利用できません)	IoT SiteWise OPC UA コレクターコンポーネントによってドロップされた MQTT メッセージの数。 単位: 数 ディメンション: GatewayId、SourceName
`OpcUaCollector.MqttMessagesPublishedBytes` (Classic ストリーム、V2 ゲートウェイでは利用できません)	IoT SiteWise OPC UA コレクターコンポーネントが MQTT ブローカーに正常に発行した MQTT メッセージデータのバイト数。 単位: 数 ディメンション: GatewayId、SourceName
`OpcUaCollector.MqttMessagesPublishedCount` (Classic ストリーム、V2 ゲートウェイでは利用できません)	IoT SiteWise OPC UA コレクターコンポーネントが MQTT ブローカーに正常に発行した MQTT メッセージの数。 単位: 数 ディメンション: GatewayId、SourceName

メトリクス	説明
`OpcUaCollector.NullValueCount` (Classic ストリーム、V2 ゲートウェイでは使用できません)	IoT SiteWise OPC UA コレクターコンポーネントが OPC UA サーバーから受信した null 値の数。 単位: 数 ディメンション: GatewayId、SourceName、PropertyGroup
`OpcUaCollector.NumberOfUniqueMqttTopicsPublished` (Classic ストリーム、V2 ゲートウェイでは使用できません)	IoT SiteWise OPC UA コレクターが MQTT ブローカーに発行する一意の MQTT トピックの数。 単位: 数 ディメンション: GatewayId、SourceName

AWS IoT SiteWise プロセッサメトリクス

メトリクス	説明
`Gateway.DataProcessor.IngestionThrottled` (MQTT 対応、V3 ゲートウェイでは利用できません)	スロットリングされたデータポイントの数。1 分ごとに生成されます。 単位: 数 ディメンション: ThrottledAt
`Gateway.DataProcessor.MeasurementRejected` (MQTT 対応、V3 ゲートウェイでは利用できません)	拒否された測定値の数。1 分ごとに生成されます。 単位: 数 ディメンション: Reason
`Gateway.DataProcessor.MessagesRemaining` (MQTT 対応、V3 ゲートウェイでは利用できません)	ストリームに残っているメッセージの数。1 分ごとに生成されます。

メトリクス	説明
	単位: 数 ディメンション: StreamName
`Gateway.DataProcessor.ProcessingError` (MQTT 対応、V3 ゲートウェイでは利用できません)	処理エラーの数。1 分ごとに生成されます。 単位: 数 ディメンション: Reason

を使用した AWS IoT SiteWise API コールのログ記録 AWS CloudTrail

AWS IoT SiteWise は、ユーザー AWS CloudTrail、ロール、または のサービスによって実行されたアクションを記録する AWS サービスである と統合されています AWS IoT SiteWise。CloudTrail は、 AWS IoT SiteWise の API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、 AWS IoT SiteWise コンソールからの呼び出しと AWS IoT SiteWise API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合、 イベントを含む Amazon S3 バケットへの CloudTrail イベントの継続的な配信をアクティブ化できます AWS IoT SiteWise。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail によって収集された情報を使用して、リクエストの実行元の IP アドレス AWS IoT SiteWise、リクエストの実行者、リクエストの実行日時などの詳細を確認できます。

CloudTrail の詳細については、「AWS CloudTrail ユーザーガイド」を参照してください。

AWS IoT SiteWise CloudTrail の情報

CloudTrail は、 AWS アカウントの作成時にアカウントでアクティブ化されます。でサポートされているイベントアクティビティが発生すると AWS IoT SiteWise、そのアクティビティは CloudTrail イベントとイベント履歴の他の AWS サービスイベントに記録されます。 AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「Viewing events with CloudTrail event history」(CloudTrail イベント履歴でのイベントの表示) を参照してください。

のイベントなど、 AWS アカウントのイベントの継続的な記録については AWS IoT SiteWise、証跡を作成します。証跡により、ログファイルを CloudTrail で Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成すると、すべての AWS リージョンに証跡が適用されま

す。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づいて対応するため、他の AWS サービスを構成できます。詳細については、次を参照してください:

- 追跡を作成するための概要
- 「CloudTrail がサポートされているサービスと統合」
- 「CloudTrail の Amazon SNS 通知の設定」
- CloudTrail ログファイルを複数のリージョンから受け取ると複数のアカウントから CloudTrail ログファイルを受け取る

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストが root または AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

AWS IoT SiteWise CloudTrail のデータイベント

データイベントでは、リソース上またはリソース内で実行されるリソースオペレーション (Amazon S3 オブジェクトの読み取りまたは書き込みなど) についての情報が得られます。これらのイベントは、データプレーンオペレーションとも呼ばれます。データイベントは、多くの場合、高ボリュームのアクティビティです。デフォルトでは、CloudTrail はデータイベントをログ記録しません。CloudTrail [イベント履歴] にはデータイベントは記録されません。

追加の変更がイベントデータに適用されます。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

CloudTrail コンソール、または CloudTrail CloudTrail API オペレーションを使用して AWS CLI、AWS IoT SiteWise リソースタイプのデータイベントをログに記録できます。このセクションの表は、使用可能なリソースタイプを示しています AWS IoT SiteWise。

- CloudTrail コンソールを使用してデータイベントを記録するには、データイベントをログに記録する証跡またはイベントデータストアを作成するか、既存の証跡またはイベントデータストアを更新してデータイベントをログに記録します。

  1. データイベントをログに記録するには、[データイベント] を選択します。
  2. [データイベントタイプ] リストから、データイベントをログ記録するリソースのタイプを選択します。
  3. 使用するログセレクタテンプレートを選択します。リソースタイプのすべてのデータイベントをログに記録したり、すべての `readOnly` イベントをログに記録したり、すべての `writeOnly` イベントをログに記録したり、カスタムログセレクタテンプレートを作成して `readOnly`、`eventName`、`resources.ARN` フィールドでフィルタリングしたりできます。

- を使用してデータイベントをログに記録するには AWS CLI、 `--advanced-event-selectors` パラメータを設定して、 `eventCategory`フィールドを に、 `Data resources.type`フィールドをリソースタイプ値に等しくします (表を参照) 。条件を追加して、`readOnly`、`eventName` および `resources.ARN` フィールドの値でフィルタリングできます。
  - データイベントを記録する証跡を設定するには、AWS CloudTrail put-event-selectors コマンドを実行します。詳細については、「AWS CLIを使用した証跡へのデータイベントのログ記録」を参照してください。
  - データイベントを記録するようにイベントデータストアを設定するには、AWS CloudTrail create-event-data-store コマンドを実行して、データイベントを記録する新しいイベントデータストアを作成するか、AWS CloudTrail update-event-data-store コマンドを実行して、既存のイベントデータストアを更新します。詳細については、「AWS CLIによるイベントデータストアのデータイベントのログ記録」を参照してください。

次の表に、 AWS IoT SiteWise リソースタイプを示します。データイベントタイプ (コンソール) 列には、CloudTrail コンソールの [データイベントタイプ] リストから選択する値が表示されます。resources.type 値列には、 AWS CLI または CloudTrail APIs を使用して高度なイベントセレクタを設定するときに指定する `resources.type`値が表示されます。CloudTrail に記録されたデータ API 列には、リソース タイプの CloudTrail にログ記録された API コールが表示されます。

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API*
AWS IoT SiteWise アセット	`AWS::IoTSiteWise::Asset`	• BatchPutAssetPropertyValue

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API*
		• GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory • BatchGetAssetPropertyAggregates
AWS IoT SiteWise 時系列	`AWS::IoTSiteWise::TimeSeries`	• BatchPutAssetPropertyValue • GetAssetPropertyValue • GetAssetPropertyValueHistory • GetAssetPropertyAggregates • GetInterpolatedAssetPropertyValues • BatchGetAssetPropertyValue • BatchGetAssetPropertyValueHistory • BatchGetAssetPropertyAggregates

データイベントタイプ (コンソール)	resources.type 値	CloudTrail にログ記録されたデータ API*
AWS IoT SiteWise Assistant	`AWS::SitewiseAssistant::Conversation`	• InvokeAssistant

> **Note**
>
> Cloudtrail イベントに記録された resources.type は、API リクエストで使用される識別子によって異なります。リクエストでアセット ID が指定された場合、アセット resources.type がログに記録され、それ以外の場合は TimeSeries resources.type がログに記録されます。

*`eventName`、`readOnly`、および `resources.ARN` フィールドでフィルタリングして、自分にとって重要なイベントのみをログに記録するように高度なイベントセレクタを設定できます。フィールドの詳細については、「AdvancedFieldSelector」を参照してください。

AWS IoT SiteWise CloudTrail の管理イベント

管理イベントのログ記録は、 AWS アカウントのリソースで実行される管理オペレーションに関する情報を提供します。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。CloudTrail は、デフォルトで管理イベントをログ記録します。

AWS IoT SiteWise は、すべての AWS IoT SiteWise コントロールプレーンオペレーションを管理イベントとしてログに記録します。CloudTrail に AWS IoT SiteWise ログ記録する AWS IoT SiteWise コントロールプレーンオペレーションのリストについては、 AWS IoT SiteWise API リファレンスを参照してください。

例: AWS IoT SiteWise ログファイルエントリ

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルには、単一または複数のログエントリがあります。各イベントは任意の送信元からの単一のリクエストを表し、リクエストされたオペレーション、オペレーションの日時、リクエストパラメータなどに関する情報を含みます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。

`CreateAsset` オペレーションを示す CloudTrail ログエントリの例は、次のとおりです。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/Administrator",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "Administrator",
    "sessionContext": {
      "sessionIssuer": {},
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-03-11T17:26:40Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
  "eventTime": "2020-03-11T18:01:22Z",
  "eventSource": "iotsitewise.amazonaws.com",
  "eventName": "CreateAsset",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "assetName": "Wind Turbine 1",
    "assetModelId": "a1b2c3d4-5678-90ab-cdef-11111EXAMPLE",
    "clientToken": "a1b2c3d4-5678-90ab-cdef-00000EXAMPLE"
  },
  "responseElements": {
    "assetId": "a1b2c3d4-5678-90ab-cdef-22222EXAMPLE",
    "assetArn": "arn:aws:iotsitewise:us-east-1:123456789012:asset/a1b2c3d4-5678-90ab-
cdef-22222EXAMPLE",
    "assetStatus": {
      "state": "CREATING"
    }
  },
  "requestID": "a1b2c3d4-5678-90ab-cdef-aaaaaEXAMPLE",
  "eventID": "a1b2c3d4-5678-90ab-cdef-bbbbbEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```

AWS IoT SiteWise リソースにタグを付ける

AWS IoT SiteWise リソースにタグを付けると、組織アセットを効率的に分類、管理、取得する強力な方法が得られます。キーと値のペアで構成されるタグを割り当てることで、説明メタデータをリソースにアタッチできます。タグからのメタデータを使用して、オペレーションを合理化できます。例えば、風力発電所のシナリオでは、タグを使用すると、場所、容量、運用ステータスなど、特定の属性を持つタービンにラベルを付けることができ、 AWS IoT SiteWise内で迅速な識別と管理が可能になります。

タグを AWS Identity and Access Management (IAM) ポリシーと統合すると、条件付きアクセスルールを定義することで、セキュリティと運用の制御を強化できます。つまり、特定のタグを持つユーザーのみを指定することができます。例えば、特定のロールまたは部門でタグ付けされたユーザーのみが、特定のリソースにアクセスまたは変更することができます。

でタグを使用する AWS IoT SiteWise

タグを使用して、ユースケースの目的、所有者、環境、またはその他の分類で AWS IoT SiteWise リソースを分類します。同じ型のリソースが多い場合に、そのタグに基づいて特定のリソースをすばやく識別できます。

各タグは、ユーザーが定義したキーとオプション値で構成されます。例えば、アセットモデルの一連のタグを確立して、それらがサポートする産業プロセスに従って追跡できます。管理するリソースのタイプごとにカスタマイズされたタグキーのセットを開発することをお勧めします。一貫性のある一連のタグキーを使用することで、リソースの管理が容易になります。

でタグ付けする AWS Management Console

のタグエディタ AWS Management Console は、すべての AWS サービスからリソースのタグを作成および管理するための一元的で統一された方法を提供します。詳細については、「 AWS リソースのタグ付け」および「タグエディタユーザーガイド」の「タグエディタの開始方法」を参照してください。

AWS IoT SiteWise API でタグ付けする

AWS IoT SiteWise API はタグも使用します。タグを作成する前に、タグの制限に注意してください。詳細については、「AWS 全般のリファレンス」の「タグの命名規則と使用規則」を参照してください。

- リソースの作成時にタグを追加するには、リソースの `tags` プロパティでタグを定義します。
- 既存のリソースにタグを追加したり、タグ値を更新するには、TagResource オペレーションを使用します。
- リソースからタグを削除するには、UntagResource オペレーションを使用します。
- リソースに関連付けられたタグを取得するには、ListTagsForResource オペレーションを使用するか、リソースを取得してその `tags` プロパティを調べます。

次の表に、 AWS IoT SiteWise API を使用してタグ付けできるリソースと、対応する `Create` および `Describe`オペレーションを示します。

タグ付け可能な AWS IoT SiteWise リソース

リソース	作成オペレーション	オペレーションの説明
アセットモデルまたはコンポーネントモデル	CreateAssetModel	DescribeAssetModel
アセット	CreateAsset	DescribeAsset
SiteWise Edge ゲートウェイ	CreateGateway	DescribeGateway
Portal	CreatePortal	DescribePortal
プロジェクト	CreateProject	DescribeProject
ダッシュボード	CreateDashboard	DescribeDashboard
アクセスポリシー	CreateAccessPolicy	DescribeAccessPolicy
時系列	BatchPutAssetPropertyValue	DescribeTimeSeries

では`BatchPutAssetPropertyValue`、アセットモデルとアセットを作成する AWS IoT SiteWise 前に、 に産業データを送信するようにデータソースを設定できます。 は、機器から raw データのストリームを受信するデータストリーム AWS IoT SiteWise を自動的に作成します。詳細については、「データインジェストの管理」を参照してください。

ビューで次のオペレーションを使用して、タグ付けをサポートしているリソースのタグを管理します。

- TagResource - リソースにタグを追加するか、既存のタグの値を更新します。
- ListTagsForResource - リソースのタグをリストします。
- UntagResource - リソースからタグを削除します。

任意の時点でリソースにタグを追加または削除します。既存のタグキーの値を更新するには、同じキーと望ましい新しい値を持つ新しいタグをリソースに追加します。このアクションにより、古い値が新しい値に置き換えられます。空の文字列をタグ値として割り当てることはできますが、null 値を割り当てることはできません。

リソースを削除すると、リソースにリンクされたタグも削除されます。

IAM ポリシーでタグを使用する

IAM ポリシーでリソースタグを使用して、ユーザーのアクセスとアクセス許可を制御します。例えば、ポリシーにより、ユーザーは特定のタグがアタッチされたリソースのみを作成できます。ポリシーにより、特定のタグを持つリソースをユーザーが作成または変更できないよう制限することもできます。

> **Note**
> タグを使用してリソースへのユーザーのアクセスを許可または拒否する場合は、ユーザーが同じリソースに対してそれらのタグを追加または削除する機能を拒否する必要があります。そうしないと、ユーザーはそのタグを変更することによって制限を迂回し、リソースにアクセスできてしまいます。

ポリシーステートメントの `Condition` 要素 (`Condition` ブロックとも呼ばれます) の次の条件コンテキストキーと値を使用できます。

`aws:ResourceTag/`*`tag-key`*`: `*`tag-value`*

特定のタグを持つリソースに対してアクションを許可または拒否します。

`aws:RequestTag/`*`tag-key`*`: `*`tag-value`*

タグ付け可能なリソースを作成または変更するときに、特定のタグを使用する (または使用しない) よう要求します。

`aws:TagKeys: [tag-key, ...]`

タグ付け可能なリソースを作成または変更するときに、特定のタグキーセットを使用する (または使用しない) よう要求します。

> **Note**
>
> IAM ポリシーの条件コンテキストキーと値は、必須パラメータとしてタグ付け可能なリソースを持つアクションにのみ適用されます。たとえば、ListAssetsに対してタグベースの条件付きアクセスを設定できます。リクエストでタグ付け可能なリソースが参照されないため、PutLoggingOptions でタグベースの条件付きアクセスを設定できません。

詳細については、「IAM ユーザーガイド」の AWS 「リソースタグを使用したリソースへのアクセスの制御」および「IAM JSON ポリシーリファレンス」を参照してください。 https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies.html

タグを使用した IAM ポリシーの例

- タグに基づいて AWS IoT SiteWise アセットを表示する

トラブルシューティング AWS IoT SiteWise

次の情報を使用して、 に関する問題をトラブルシューティングします AWS IoT SiteWise。

トピック

- SiteWise Edge ゲートウェイのトラブルシューティング
- AWS IoT SiteWise ポータルのトラブルシューティング
- AWS IoT SiteWise ルールアクションのトラブルシューティング
- 一括インポートおよびエクスポートオペレーションのトラブルシューティング

SiteWise Edge ゲートウェイのトラブルシューティング

関連するトピックを調べて、一般的な AWS IoT SiteWise Edge ゲートウェイの問題をトラブルシューティングします。

SiteWise Edge ゲートウェイによって報告された CloudWatch メトリクスを表示して、接続またはデータストリームに関する問題のトラブルシューティングを行うこともできます。詳細については、「Amazon CloudWatch メトリクス AWS IoT SiteWise によるモニタリング」を参照してください。

トピック

- SiteWise Edge ゲートウェイログを設定してアクセスする
- SiteWise Edge ゲートウェイの問題のトラブルシューティング
- での AWS IoT SiteWise Edge アプリケーションのトラブルシューティング Siemens Industrial Edge
- Edge でのオープンソース統合のトラブルシューティング
- AWS IoT Greengrass 問題のトラブルシューティング

SiteWise Edge ゲートウェイログを設定してアクセスする

SiteWise Edge ゲートウェイログを表示する前に、Amazon CloudWatch Logs にログを送信するか、ローカルファイルシステムにログを保存するように、SiteWise Edge ゲートウェイを設定する必要があります。

- を使用して SiteWise Edge ゲートウェイのログファイルを表示する場合は AWS Management Console 、CloudWatch Logs を使用します。詳細については、「Amazon CloudWatch Logs を使用」を参照してください。
- コマンドラインまたはローカルソフトウェアを使用して SiteWise Edge ゲートウェイのログファイルを表示する場合は、ローカルファイルシステムログを使用します。詳細については、「でサービスログを使用する AWS IoT SiteWise」を参照してください。

SiteWise Edge ゲートウェイの問題のトラブルシューティング

SiteWise Edge ゲートウェイの問題をトラブルシューティングするには、次の情報を使用します。

問題

- SiteWise Edge ゲートウェイにパックをデプロイできない
- AWS IoT SiteWise は OPC UA サーバーからデータを受信しません
- ダッシュボードにデータが表示されない
- /greengrass/v2/logs にある aws.iot.SiteWiseEdgePublisher のログに「メインクラスが見つかりません、または読み込めませんでした」というエラーが表示される
- 「SESSION_TAKEN_OVER」または「com.aws.greengrass.mqttclient.MqttClient: Spooler 経由でメッセージを発行できませんでした。再試行します。」 とログに表示されます。
- 「com.aws.greengrass.deployment.IotJobsHelper: デプロイジョブが見つかりません。」 または「デプロイ結果は既に報告されています。」 とログに表示されます。
- OPC UA データソースのプロパティグループでタイムスタンプ設定を設定しようとすると、「SYNC_FAILED」ステータスが表示される
- 変換されたデータ型が含まれません
- トラストストアの問題
- プロキシが有効なインストールの問題

SiteWise Edge ゲートウェイにパックをデプロイできない

nucleus コンポーネント (`aws.greengrass.Nucleus`) AWS IoT Greengrass が古い場合、SiteWise Edge ゲートウェイにパックをデプロイできない可能性があります。 AWS IoT Greengrass V2 コンソールを使用して nucleus AWS IoT Greengrass コンポーネントをアップグレードできます。

nucleus AWS IoT Greengrass コンポーネントをアップグレードするには (コンソール)

1. AWS IoT Greengrass コンソールに移動します。
2. ナビゲーションペインの AWS IoT Greengrass で、[デプロイ] を選択します。
3. デプロイのリストで、修正するデプロイを選択します。
4. [修正] を選択します。
5. [Specify target] (ターゲットの指定) ページで [Next] (次へ) を選択します。
6. [コンポーネントの選択] ページの [パブリックコンポーネント] の検索ボックスに **`aws.greengrass.Nucleus`** と入力し、aws.greengrass.Nucleus を選択します。
7. [次へ] を選択します。
8. [コンポーネントの設定] ページで、[次へ] をクリックします。
9. [詳細設定の設定] ページで、[次へ] を選択します。
10. [Review] ページで、[デプロイ] を選択します。

AWS IoT SiteWise は OPC UA サーバーからデータを受信しません

AWS IoT SiteWise アセットが OPC UA サーバーから送信されたデータを受信していない場合は、SiteWise Edge ゲートウェイのログを検索して問題をトラブルシューティングできます。次のメッセージを含む情報レベルの `swPublisher` ログを探します。

```
Emitting diagnostic name=PublishError.SomeException
```

ログの *`SomeException`* のタイプに基づいて、次の例外タイプとそれに対応する問題を使用して、SiteWise Edge ゲートウェイのトラブルシューティングを行います。

- ResourceNotFoundException - OPC UA サーバーは、どのアセットのプロパティエイリアスにも一致しないデータを送信しています。この例外は、次の 2 つの場合に発生します。
  - プロパティエイリアスが、定義したソースプレフィックスを含め、OPC UA 変数と正確に一致しません。プロパティエイリアスとソースプレフィックスが正しいことを確認します。
  - OPC UA 変数をアセットプロパティにマッピングしていません。詳細については、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。

    必要なすべての OPC UA 変数を既にマッピングしている場合は AWS IoT SiteWise、SiteWise Edge ゲートウェイが送信する OPC UA 変数をフィルタリングできます。詳細については、「SiteWise Edge で OPC UA ノードフィルターを使用する」を参照してください。

- InvalidRequestException - OPC UA 変数のデータ型がアセットプロパティのデータ型に一致しません。例えば、OPC UA 変数が整数データ型の場合、対応するアセットプロパティは整数データ型である必要があります。倍精度型のアセットプロパティが OPC UA 整数値を受け取ることはできません。この問題を解決するには、正しいデータ型で新しいプロパティを定義します。
- TimestampOutOfRangeException – SiteWise Edge ゲートウェイは、 が AWS IoT SiteWise 受け入れる範囲外のデータを送信しています。 は、過去 7 日より前、または今後 5 分より新しいタイムスタンプを持つデータポイント AWS IoT SiteWise を拒否します。SiteWise Edge ゲートウェイの電源または AWS クラウドへの接続が失われた場合は、SiteWise Edge ゲートウェイのキャッシュをクリアする必要がある場合があります。
- ThrottlingException または LimitExceededException – リクエストが、取り込まれたデータポイントのレートやアセットプロパティデータ API オペレーションのリクエストレートなどの AWS IoT SiteWise サービスクォータを超えました。設定が AWS IoT SiteWise クォータ を超えないことを確認します。

ダッシュボードにデータが表示されない

ダッシュボードにデータが表示されない場合、SiteWise Edge ゲートウェイの [パブリッシャー設定] と データソース] が同期していない可能性があります。同期していない場合、データソース名を変更すると、クラウドからエッジへの同期が速まり、非同期エラーが解決される場合があります。

データソースの名前を更新するには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. ダッシュボードに接続された SiteWise Edge ゲートウェイを選択します。
4. [データソース] で [編集] を選択します。
5. 新しいソースの [名前] を選択し、[保存] を選択して変更を確定します。
6. データソーステーブルのデータソース名が更新されて、変更が反映されていることを確認します。

/greengrass/v2/logs にある aws.iot.SiteWiseEdgePublisher のログに「メインクラスが見つかりません、または読み込めませんでした」というエラーが表示される

このエラーが表示される場合は、SiteWise Edge ゲートウェイの Java バージョンを更新する必要がある場合があります。

- ターミナルから、次のコマンドを実行します。

```
java -version
```

SiteWise Edge ゲートウェイが実行している Java のバージョンは、`OpenJDK Runtime Environment` の下に表示されます。以下のようなレスポンスが表示されます。

```
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)
```

Java バージョン 11.0.20.8.1 を実行している場合は、IoT SiteWise パブリッシャーパックをバージョン 2.4.1 またはそれ以降に更新する必要があります。Java バージョン 11.0.20.8.1 のみが影響を受けます。他の Java バージョンが使用されている環境では、そのまま古いバージョンの IoT SiteWise Publisher コンポーネントを使用できます。コンポーネントパックの更新の詳細については、「SiteWise Edge ゲートウェイコンポーネントパックのバージョンを変更する」を参照してください。

「SESSION_TAKEN_OVER」または「com.aws.greengrass.mqttclient.MqttClient: Spooler 経由でメッセージを発行できませんでした。再試行します。」 とログに表示されます。

`/greengrass/v2/logs/greengrass.log` のログに、`SESSION_TAKEN_OVER` を含む警告または `com.aws.greengrass.mqttclient.MqttClient: Failed to publish the message via Spooler and will retry.` を含むエラーが表示された場合は、複数のデバイス上の複数の SiteWise Edge ゲートウェイで同じ設定ファイルを使用しようとしている可能性があります。各 SiteWise Edge ゲートウェイには、 AWS アカウントに接続するための一意の設定ファイルが必要です。

「com.aws.greengrass.deployment.IotJobsHelper: デプロイジョブが見つかりません。」 または「デプロイ結果は既に報告されています。」 とログに表示されます。

`/greengrass/v2/logs/greengrass.log` のログに `com.aws.greengrass.deployment.IotJobsHelper: No deployment job found.` または `Deployment result already reported.` が表示される場合は、同じ設定ファイルを再利用しようとしている可能性があります。

以下のように複数の解決策があります。

- 設定ファイルを再利用する場合は、次のようにします。

  1. AWS IoT SiteWise コンソールに移動します。
  2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
  3. 再利用する SiteWise Edge ゲートウェイを選択します。
  4. [更新] タブを選択します。
  5. 別のパブリッシャーバージョンを選択し、[デプロイ] を選択します。

Siemens Industrial Edge 用のゲートウェイを作成する のステップに従って、新しい設定ファイルを作成します。

OPC UA データソースのプロパティグループでタイムスタンプ設定を設定しようとすると、「SYNC_FAILED」ステータスが表示される

バージョン 2.5.0 AWS IoT Greengrass で の OPC UA コレクターコンポーネント AWS IoT SiteWise を更新したときに、新しいタイムスタンプ設定オプションが導入されました。デバイスからのタイムスタンプ、またはサーバーからのタイムスタンプを使用できます。OPC UA コレクターコンポーネントの古いバージョンはこのオプションをサポートしておらず、同期に失敗します。

データソースの同期ステータスの失敗を解決するには、2 つの方法があります。推奨される方法は、IoT SiteWise OPC UA コレクターコンポーネントをバージョン 2.5.0 以降にアップグレードすることです。または、タイムスタンプを に設定した場合、古い OPC UA コレクターコンポーネントバージョンを引き続き使用できます`Source`。IoT SiteWise OPC UA コレクターコンポーネントをアップグレードする方法については、「」を参照してください コンポーネントのバージョン AWS IoT SiteWise を更新する。すべてのコンポーネントの最新バージョンを使用することをお勧めします。

> **Note**
>
> データソースの同期ステータスが失敗しても、データの中断はありません。ソースデータは引き続き流れます AWS IoT SiteWise。設定は、単に AWS IoT Greengrass V2 デプロイの IoT SiteWise OPC UA コレクターコンポーネントと同期していません。

プロパティグループのタイムスタンプ設定を変更するには

1. AWS IoT SiteWise コンソールに移動します。

2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. 編集するゲートウェイを選択します。
4. データソースセクションで、失敗した同期ステータスのデータソースを選択し、編集を選択します。
5. 詳細設定を展開し、グループ設定を展開します。
6. タイムスタンプで、ソースを選択します。Source を選択すると、設定から `timestampToReturn`プロパティが削除されます。この設定により、デフォルトでデバイスからのデータソースタイムスタンプの収集が可能になり、データソースは IoT SiteWise OPC UA コレクターコンポーネントと同期できます。
7. [保存] を選択します。

変換されたデータ型が含まれません

サポートされていない OPC UA データ型を の文字列に変換するときにエラーが表示される場合は AWS IoT SiteWise、いくつかの理由が考えられます。

- 変換しようとしているデータ型は複雑なデータ型です。複雑なデータ型はサポートされていません。
- [送信先] を [Amazon S3 を使用してAWS IoT SiteWise バッファリング]として使用する場合、完全な文字列値は Amazon S3 バケットにプッシュされたファイルに保持されます。後で にデータを取り込むと AWS IoT SiteWise、1024 バイトを超える文字列値はすべて拒否されます。

トラストストアの問題

SiteWise Edge の信頼ストアに関連する問題が発生した場合は、次のトラブルシューティングステップを検討してください。

- AWS IoT Greengrass ルート CA 証明書が存在し、適切な信頼ストアに正しくフォーマットされていることを確認します。
- Java KeyStore パスワードが正しく設定され、SiteWise Edge コンポーネントにアクセスできることを確認します。
- カスタム証明書 (HTTPS プロキシの場合など) が正しい形式 (通常は PEM) で、信頼ストアに正しくインポートされていることを確認します。
- 信頼ストアに正しいファイルアクセス許可があり、SiteWise Edge プロセスにアクセスできることを確認します。

- SiteWise Edge ログで SSL/TLS 関連のエラーを確認します。これはトラストストアの問題を示している可能性があります。
- などのツールを使用して SSL/TLS 接続`openssl`を個別にテストし、トラストストアの機能を検証する

プロキシが有効なインストールの問題

プロキシ設定プロセス中に問題が発生した場合は、以下のトラブルシューティング手順を検討してください。

- プロキシ URL が正しくフォーマットされ、適切なスキーム (`http://` または `https://`) が含まれていることを確認します。
- プロキシ認証情報に特殊文字が含まれている場合は、必ず URL エンコードしてください。
- プロキシなしリストに必要なすべてのローカルアドレスと AWS サービスエンドポイントが含まれていることを確認します。
- HTTPS プロキシの場合は、提供された CA 証明書が PEM 形式であることを確認します。
- インストールログで、問題の原因を示す可能性のある特定のエラーメッセージを確認します。
- プロキシ接続が正しく機能していることを個別にテストする

での AWS IoT SiteWise Edge アプリケーションのトラブルシューティング Siemens Industrial Edge

Siemens Industrial Edge デバイス上の AWS IoT SiteWise Edge アプリケーションのトラブルシューティングを行うには、 Siemens Industrial Edge Managementまたは Siemens Industrial Edge Device (IED) ポータルからアプリケーションのログにアクセスできます。詳細については、Siemens ドキュメントの「ログのダウンロード」を参照してください。

データが に表示されない AWS IoT SiteWise

- Databus ユーザーに問題がなく、Databus_Configuration のチェックマークアイコンが灰色ではなく緑色であることを確認します。
- Secure Storage を含むバージョンで Siemens Industrial Edge Management を実行していない可能性があります。Siemens OS のバージョンをアップグレードします。詳細については、「Siemens Secure Storage と AWS IoT SiteWise Edge アプリケーション」を参照してください。

ログに「Config ファイル欠落 AWS_REGION」と表示される

Siemens ログに `Config file missing AWS_REGION` と表示される場合、設定ファイルの JSON が破損しています。新しい設定ファイルを作成する必要があります。Siemens Industrial Edge 用のゲートウェイを作成する のステップに従って、新しい設定ファイルを作成します。

Edge ゲートウェイ設定に「Out of sync」というエラーメッセージが表示される

デプロイの完了後にSiemens Industrial Edgeゲートウェイに`Out of sync`エラーメッセージが表示された場合は、IoT SiteWise パブリッシャーコンポーネントがゲートウェイと同期していないことを意味します。IoT SiteWise パブリッシャーコンポーネントはSiemens Industrial Edge、ゲートウェイでバックグラウンドで動作し、MQTT トピック機能を提供します。`iotsitewise:publisher:3` ではなく 機能名前空間を使用するようにSiemens Industrial Edgeゲートウェイをアップグレードしました`iotsitewise:publisher:2`。パブリッシャーの最新バージョンに更新して、この問題を解決できます。

IoT SiteWise パブリッシャーの最新バージョンにアップグレードするには

1. AWS IoT SiteWise コンソールに移動します。
2. ナビゲーションペインで、[Edge ゲートウェイ] を選択します。
3. 編集するSiemens Industrial Edgeゲートウェイを選択します。
4. Edge 機能セクションで、ソフトウェアバージョンの表示を選択します。
5. パブリッシャードロップダウンメニューで IoT SiteWise パブリッシャーの最新バージョンを選択します。
6. [完了] をクリックします。

Edge でのオープンソース統合のトラブルシューティング

このセクションでは、オープンソースツールを SiteWise Edge と統合するときに発生する可能性がある一般的な問題の解決策を示します。

> **Note**
>
> Node-RED®、InfluxDB®、Grafana® は SiteWise Edge のベンダーまたはサプライヤーではありません。

接続の問題

Node-RED が MQTT ブローカーに接続できない

指定されたポートで MQTT ブローカーが実行され、アクセス可能であることを確認します。ネットワーク設定を確認し、ブローカーアドレスが正しいことを確認します。

MQTT ブローカーのステータスを確認するには、以下を実行します。

```
docker ps | grep emqx
```

InfluxDB 接続エラー

認証トークンが有効であり、正しい組織名とバケット名を指定していることを確認します。InfluxDB が実行されており、アクセス可能であることを確認します。

InfluxDB のステータスを確認するには、以下を実行します。

```
curl -I http://localhost:8086
```

Grafana が InfluxDB に接続できない

URL、認証トークン、組織、バケットなど、Grafana の InfluxDB データソース設定が正しいことを確認します。

データフローの問題

にデータが表示されない AWS IoT SiteWise

Node-RED フローのプロパティエイリアスが予想される形式と一致していることを確認します。MQTT トピック構造が正しく、SiteWise Edge ゲートウェイが MQTT ブローカーからデータを受信するように適切に設定されていることを確認します。

InfluxDB に SiteWise Edge データが保存されていない

Node-RED 保持フローが正しく設定され、InfluxDB ライターノードに適切なバケットと測定設定があることを確認します。Node-RED デバッグ出力にエラーがないか確認します。

データフォーマットエラー

データ変換関数が形式間でデータを正しく変換していることを確認します。Node-RED デバッグノードを使用して、フローの各段階でデータを検査します。

パフォーマンスの問題

CPU またはメモリの使用率が高い

リソースの使用状況を監視し、必要に応じてコンポーネントの設定を調整します。データ収集頻度を減らすか、データフィルタリングを実装して処理負荷を減らすことを検討してください。

リソースの使用状況をモニタリングするには、以下を実行します。

```
docker stats
```

Grafana ダッシュボードの読み込みが遅い

InfluxDB クエリを最適化し、ダッシュボードパネルに時間範囲の制限を追加することを検討してください。適切な集計関数を使用して、表示されるデータポイントの数を減らします。

ログ記録と診断

問題をトラブルシューティングするには、各コンポーネントのログを確認します。

Node-RED ログ

Node-RED コンソールでログを表示するか、以下を実行します。

```
docker logs node-red
```

InfluxDB ログ

以下を実行してログにアクセスします。

```
docker logs influxdb
```

Grafana ログ

以下を実行してログを表示します。

```
docker logs grafana
```

SiteWise Edge ログ

SiteWise Edge ゲートウェイログで MQTT 接続とデータ処理の問題を確認します。詳細については、「SiteWise Edge ゲートウェイのトラブルシューティング」を参照してください。

AWS IoT Greengrass 問題のトラブルシューティング

SiteWise Edge ゲートウェイの設定またはデプロイに関する多くの問題の解決策については AWS IoT Greengrass、「 AWS IoT Greengrass デベロッパーガイド」の「トラブルシューティング AWS IoT Greengrass」を参照してください。

AWS IoT SiteWise ポータルのトラブルシューティング

AWS IoT SiteWise ポータルの一般的な問題をトラブルシューティングします。

ユーザー、管理者が AWS IoT SiteWise ポータルにアクセスできない

ユーザーまたは管理者が AWS IoT SiteWise ポータルにアクセスできない場合は、アタッチされた AWS Identity and Access Management (IAM) ポリシーで、ログインの成功を妨げるアクセス許可が制限されている可能性があります。

ログインが失敗する IAM ポリシーの例を以下に示します。

> **Note**
>
> アタッチされている IAM ポリシーに `"Condition"` 要素を含むポリシーがあると、ログイン失敗の原因となります。

例 1: IP アドレスが制限されているため、ログインに失敗する。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribePortal"
            ],
            "Resource": "*",
            "Condition": {
                "IpAddress": {
```

```
                    "aws:SourceIp": [
                        "REPLACESAMPLEIP"
                    ]
                }
            }
        }
    ]
}
```

例 2: タグが含まれているため、ログインに失敗する。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iotsitewise:DescribePortal"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "aws:ResourceTag/project": "*"
                }
            }
        }
    ]
}
```

ポータルへのユーザーまたは管理者の追加では、IP の制限などのユーザー許可を制限する IAM ポリシーを作成しないようにしてください。アクセス許可が制限されたアタッチされたポリシーは、AWS IoT SiteWise ポータルに接続できません。

AWS IoT SiteWise ルールアクションのトラブルシューティング

で AWS IoT SiteWise ルールアクションをトラブルシューティングするには AWS IoT Core、次のいずれかの手順を実行します。

- Amazon CloudWatch Logs の設定をする
- ルールの再パブリッシュエラーアクションを設定する

次に、エラーメッセージとこのトピックのエラーを比較して、問題のトラブルシューティングを行います。

トピック

- AWS IoT Core ログの設定
- 再発行エラーアクションを設定
- ルール問題をトラブルシューティング
- ルールをトラブルシューティング (AWS IoT SiteWise)
- ルールをトラブルシューティング (DynamoDB)

AWS IoT Core ログの設定

さまざまなレベルの情報を CloudWatch Logs に記録する AWS IoT ように を設定できます。

CloudWatch Logs を設定してアクセスするには。

1. のログ記録を設定するには AWS IoT Core、「 AWS IoT デベロッパーガイド」のCloudWatch Logs によるモニタリング」を参照してください。
2. [CloudWatch console] (CloudWatch のコンソール) に移動する。
3. ナビゲーションペインで、[Log groups] (ロググループ) を選択します。
4. [AWSIotLogs] グループを選択します。
5. 最近のログストリームを選択します。CloudWatch では、デフォルトで最新のログストリームが最初に表示されます。
6. ログエントリを選択して、ログメッセージを展開します。ログエントリは、次のスクリーンショットのようになります。

7. エラーメッセージとこのトピックのエラーを比較して、問題のトラブルシューティングを行います。

再発行エラーアクションを設定

エラーメッセージを処理するために、ルールにエラーアクションを設定できます。この手順では、MQTT テストクライアントでエラーメッセージを表示するために、エラーアクションとして再パブリッシュルールアクションを設定します。

> **Note**
>
> 再パブリッシュエラーアクションでは、ERROR レベルログと同等のログのみが出力されます。より詳細なログが必要な場合は、[CloudWatch Logs] を設定する必要があります。

再パブリッシュエラーアクションをルールに追加するには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで [Act (アクト)] を選択し、[ルール] を選択します。
3. ルールを選択します。
4. [エラーアクション] で、[アクションの追加] を選択します。
5. AWS IoT トピックにメッセージを再発行を選択します。

6. ページの下部にある [アクションの設定] を選択します。
7. トピックに一意のトピック (など) を入力します**sitewise/windfarm/rule/error**。 AWS IoT Core はこのトピックにエラーメッセージを再発行します。
8. 選択 を選択して、エラーアクションを実行するための AWS IoT Core アクセスを許可します。
9. ルールに作成したロールの横にある [選択] を選択します。
10. [ロールの更新] を選択して、追加のアクセス許可をロールに追加します。
11. [アクションを追加] を選択します。

    ルールのエラーアクションは、次のスクリーンショットのようになります。

12. コンソールの左上にある戻る矢印を選択して、 AWS IoT コンソールのホームに戻ります。

再パブリッシュエラーアクションを設定したら、 AWS IoT Coreの MQTT テストクライアントでエラーメッセージを見ることができます。

次の手順では、MQTT テストクライアントでエラートピックにサブスクライブします。MQTT テストクライアントでは、問題のトラブルシューティングを行うためにルールのエラーメッセージを受信することができます。

エラーアクショントピックをサブスクライブするには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションページで、[テスト] を選択して MQTT テストクライアントを開きます。

3. [トピックのサブスクリプション] フィールドに、前に設定したエラートピック (**sitewise/windfarm/rule/error** など) を入力し、[トピックへのサブスクライブ] を選択します。

4. エラーメッセージが表示されるまで監視し、エラーメッセージで `failures` 配列を展開します。

   次に、エラーメッセージとこのトピックのエラーを比較して、問題のトラブルシューティングを行います。

ルール問題をトラブルシューティング

ルールの問題をトラブルシューティングするには、次の情報を使用します。

問題

- Error: Member must be within 604800 seconds before and 300 seconds after the current timestamp (エラー: メンバーは、現在のタイムスタンプ前の 604800 秒以内で、現在のタイムスタンプ後の 300 秒以内でなければなりません)
- Error: Property value does not match data type <type> (エラー: プロパティ値がデータ型 <type> と一致しません)
- Error: User: <role-arn> is not authorized to perform: iotsitewise:BatchPutAssetPropertyValue on resource (エラー: ユーザー: <role-arn> には、リソースで iotsitewise:BatchPutAssetPropertyValue を実行するアクセス許可がありません)
- Error: iot.amazonaws.com is unable to perform: sts:AssumeRole on resource: <role-arn> (エラー: iot.amazonaws.com がリソースで sts:AssumeRole を実行できません (<role-arn>))
- Info: No requests were sent. (情報: リクエストは送信されませんでした。) 置換テンプレートの実行後に、PutAssetPropertyValueEntries が空でした。

Error: Member must be within 604800 seconds before and 300 seconds after the current timestamp (エラー: メンバーは、現在のタイムスタンプ前の 604800 秒以内で、現在のタイムスタンプ後の 300 秒以内でなければなりません)

タイムスタンプが、現在の Unix エポック時間と比較して、7 日より前か、5 分より後です。次の操作を試してください：

- タイムスタンプが Unix エポック (UTC) 時間であることを確認します。別のタイムゾーンでタイムスタンプを指定すると、このエラーが表示されます。
- タイムスタンプが秒単位であることを確認します。 AWS IoT SiteWise は、タイムスタンプを秒単位 (Unix エポック時間) に分割し、ナノ秒単位でオフセットします。
- 過去に 7 日以内のタイムスタンプが設定されたデータをアップロードしていることを確認します。

Error: Property value does not match data type <type> (エラー: プロパティ値がデータ型 <type> と一致しません)

ルールアクションのエントリのデータ型が、ターゲットアセットプロパティとは異なります。たとえば、ターゲットアセットプロパティが `DOUBLE` で、選択したデータ型が [整数] であるか、値を `integerValue` に渡したとします。次の操作を試してください：

- AWS IoT コンソールからルールを設定する場合は、エントリごとに正しいデータ型が選択されていることを確認します。
- API または AWS Command Line Interface （ AWS CLI) からルールを設定する場合は、`value`オブジェクトが正しい型フィールド (`doubleValue``DOUBLE`プロパティなど) を使用していることを確認します。

Error: User: <role-arn> is not authorized to perform: iotsitewise:BatchPutAssetPropertyValue on resource (エラー: ユーザー: <role-arn> には、リソースで iotsitewise:BatchPutAssetPropertyValue を実行するアクセス許可がありません)

ルールにターゲットアセットプロパティにアクセスする権限がないか、ターゲットアセットプロパティが存在しません。次の操作を試してください：

- プロパティエイリアスが正しいことと、指定されたプロパティエイリアスを持つアセットプロパティがあることを確認します。詳細については、「のデータストリームを管理する AWS IoT SiteWise」を参照してください。
- ルールにロールがあることと、ターゲットのアセットプロパティ (ターゲットアセットの階層など) への `iotsitewise:BatchPutAssetPropertyValue` アクセス許可がロールによって許可されていることを確認します。詳細については、「必要なアクセス権を付与 AWS IoT する」を参照してください。

Error: iot.amazonaws.com is unable to perform: sts:AssumeRole on resource: <role-arn> (エラー: iot.amazonaws.com がリソースで sts:AssumeRole を実行できません (<role-arn>))

ユーザーには、 AWS Identity and Access Management (IAM) のルールでロールを引き受ける権限がありません。

作成されたルールで、そのロールに対する `iam:PassRole` へのアクセス許可がユーザーに許可されていることを確認します。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Pass role permissions] (ロールアクセス許可を渡す) を参照してください。

Info: No requests were sent. (情報: リクエストは送信されませんでした。) 置換テンプレートの実行後に、PutAssetPropertyValueEntries が空でした。

> **Note**
>
> このメッセージは `INFO` レベルのログです。

リクエストに、必須パラメータをすべて含むエントリが少なくとも 1 つ含まれている必要があります。

置換テンプレートなどのルールのパラメータが空でない値になることを確認します。置換テンプレートは、ルールクエリステートメントの `AS` 句に定義されている値にアクセスできません。詳細については、[AWS IoT Developer Guide] (デベロッパーガイド) の[Substitution templates] (置換テンプレート) を参照してください。

ルールをトラブルシューティング (AWS IoT SiteWise)

CPU とメモリの使用状況データが想定 AWS IoT SiteWise どおりに に表示されない場合は、この手順のステップに従ってルールをトラブルシューティングします。この手順では、MQTT テストクライアントでエラーメッセージを表示するために、エラーアクションとして再パブリッシュルールアクションを設定します。また、トラブルシューティングするために CloudWatch Logs へのロギングを設定することもできます。詳細については、「AWS IoT SiteWise ルールアクションのトラブルシューティング」を参照してください。

再パブリッシュエラーアクションをルールに追加するには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで [メッセージルーティング] を選択し、[ルール] を選択します。
3. 先ほど作成したルールを選択し、[編集] を選択します。
4. [エラーアクション - オプション] で、[エラーアクションの追加] を選択します。
5. AWS IoT トピックにメッセージを再発行を選択します。
6. トピックで、エラーへのパス (など) を入力します**`sitewise/rule/tutorial/error`**。 AWS IoT Core はこのトピックにエラーメッセージを再発行します。
7. 先に作成したロール (SiteWiseTutorialDeviceRuleRole など) を選択します。
8. [Update] (更新) を選択します。

再パブリッシュエラーアクションを設定したら、 AWS IoT Coreの MQTT テストクライアントでエラーメッセージを見ることができます。

次の手順では、MQTT テストクライアントでエラートピックにサブスクライブします。

エラーアクショントピックをサブスクライブするには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションページで、[MQTT テストクライアント] を選択して MQTT テストクライアントを開きます。
3. [トピックのフィルター] で、**`sitewise/rule/tutorial/error`** と入力し、[サブスクリプション] を選択します。

エラーメッセージが表示された場合は、エラーメッセージで `failures` 配列を表示して問題を診断します。考えられる問題とその解決方法の詳細については、「AWS IoT SiteWise ルールアクションのトラブルシューティング」を参照してください。

エラーが表示されない場合は、ルールが有効になっていて、再パブリッシュエラーアクションで設定したトピックと同じトピックにサブスクライブしていることを確認します。それでもエラーが表示されない場合は、デバイススクリプトが実行され、デバイスのシャドウが正常に更新されていることを確認します。

> **Note**
>
> デバイスのシャドウ更新トピックをサブスクライブして、 AWS IoT SiteWise アクションが解析するペイロードを表示することもできます。これを行うには、次のトピックをサブスクライブしてください。
>
> ```
> $aws/things/+/shadow/update/accepted
> ```

ルールをトラブルシューティング (DynamoDB)

デモのアセットデータが想定通りに DynamoDB テーブルで表示されない場合は、ここに記載されているステップに従って、ルールをトラブルシューティングします。この手順では、MQTT テストクライアントでエラーメッセージを表示するために、エラーアクションとして再パブリッシュルールアクションを設定します。また、トラブルシューティングするために CloudWatch Logs へのロギングを設定することもできます。詳細については、AWS IoT [Developer Guide] (デベロッパーガイド) の [Monitoring with CloudWatch Logs] (CloudWatch Logsでモニタリングする) を参照してください。

再パブリッシュエラーアクションをルールに追加するには

1. AWS IoT コンソールに移動します。
2. 左側のナビゲーションペインで [アクト] を選択し、[ルール] を選択します。
3. 先ほど作成したルールを選択します。

4. [エラーアクション] で、[アクションの追加] を選択します。
5. AWS IoT トピックにメッセージを再発行を選択します。

6. ページの下部にある [アクションの設定] を選択します。
7. [Topic] (トピック) で、**`windspeed/error`** と入力します。 AWS IoT Core はこのトピックにエラーメッセージを再発行します。

8. Select を選択して、前に作成したロールを使用してエラーアクションを実行するためのアクセスを AWS IoT Core に付与します。
9. ロールの横にある [選択] を選択します。

10. [ロールの更新] を選択して、追加のアクセス許可をロールに追加します。

11. [アクションの追加] を選択して、エラーアクションの追加を終了します。
12. コンソールの左上にある戻る矢印を選択して、 AWS IoT Core コンソールホームに戻ります。

再発行エラーアクションを設定したら、 AWS IoT Core の MQTT テストクライアントでエラーメッセージを表示できます。

次の手順では、MQTT テストクライアントでエラートピックにサブスクライブします。

エラーアクショントピックをサブスクライブするには

1. AWS IoT Core コンソールの左側のナビゲーションページで、テストを選択します。
2. [トピックのサブスクリプション] フィールドで、**`windspeed/error`** と入力し、[トピックへのサブスクリプション] を選びます。

3. エラーメッセージが表示されるのを監視し、エラーメッセージで `failures` 配列を調べて、次の一般的な問題を診断します。
   - ルールクエリステートメントのタイプミス
   - ロールのアクセス許可の不足

エラーが表示されない場合は、ルールが有効になっていて、再パブリッシュエラーアクションで設定したトピックと同じトピックにサブスクライブしていることを確認します。それでもエラーが表示されない場合は、デモの風力発電所アセットが存在し、風速プロパティで通知が有効になっているかを確認します。デモアセットの有効期限が切れて消えた場合は AWS IoT SiteWise、新しいデモを作成し、更新されたアセットモデルとプロパティ IDs を反映するようにルールクエリステートメントを更新できます。

一括インポートおよびエクスポートオペレーションのトラブルシューティング

転送ジョブ中に生成されたエラーを処理および診断するには、 AWS IoT TwinMaker GetMetadataTransferJob API を参照してください。

1. 転送ジョブを作成して実行した後、GetMetadataTransferJob API を呼び出します。

```
aws iottwinmaker get-metadata-transfer-job \
--metadata-transfer-job-id your_metadata_transfer_job_id \
--region us-east-1
```

2. ジョブの状態が次のいずれかの状態に変更されます。
   - COMPLETED
   - CANCELLED
   - ERROR
3. GetMetadataTransferJob API は MetadataTransferJobProgress オブジェクトを返します。
4. MetadataTransferJobProgress オブジェクトには、以下のパラメータが含まれています。
   - failedCount: 転送プロセス中に失敗したアセットの数を示します。
   - skippedCount: 転送プロセス中にスキップされたアセットの数を示します。
   - succeededCount: 転送プロセス中に成功したアセットの数を示します。
   - totalCount: 転送プロセスに関連するアセットの合計数を示します。
5. さらに、reportUrl 要素が API コールによって返され、事前署名付き URL を含みます。転送ジョブに調査が必要なエラーがあった場合、この URL で完全なエラーレポートをダウンロードできます。

AWS IoT SiteWise エンドポイントとクォータ

以下のセクションでは、 のエンドポイントとクォータについて説明します AWS IoT SiteWise。

トピック

- AWS IoT SiteWise エンドポイント
- AWS IoT SiteWise クォータ

AWS IoT SiteWise エンドポイント

AWS 全般のリファレンス ガイドには、 の AWS IoT SiteWise エンドポイントが一覧表示されています AWS アカウント。詳細については、「AWS 全般のリファレンス ガイド」の「AWS IoT SiteWise エンドポイントとクォータ」を参照してください。

AWS IoT SiteWise クォータ

次の表では、 のクォータについて説明します AWS IoT SiteWise。クォータおよびクォータ増加のリクエスト方法の詳細については、『AWS 全般のリファレンス』の 「AWS Service Quotas」を参照してください。 AWS IoT SiteWise クォータの詳細については、のAWS IoT SiteWise 「サービスクォータ」を参照してくださいAWS 全般のリファレンス。

AWS IoT SiteWise アセットとアセットモデルのクォータ

リソース	説明	Quota	調整可能
各 のアセットモデルの数 AWS リージョン AWS アカウント	AWS リージョン ので作成できるアセットモデルの最大数 AWS アカウント。	1,000	はい
各アセットモデル内のアセットの数	各アセットモデルに作成できるアセットの最大数。	10,000	はい
各親アセット内の子アセットの数	親アセットに関連付けることができる子アセットの最大数。	2000	はい

リソース	説明	Quota	調整可能
アセットモデル階層ツリーの深さ	アセットモデルのアセット階層ツリーの最大深度。	30	はい
各アセットモデルの階層定義の数	アセットモデルに含めることができる階層定義の最大数。	30	はい
各アセットモデルのルートレベルのプロパティの数	`assetModelProperties` 各アセットモデルの の最大数。このカウントには `compositeModelProperties` は含まれません。このクォータは、このアセットモデルから作成された一意のアセットにも適用されます。	500	はい

リソース	説明	Quota	調整可能
アセットモデルのプロパティの数	タイプ ASSET_MODEL または COMPONENT_MODEL のアセットモデルのプロパティの最大数。この数は、ルートアセットモデルのプロパティと、含まれているコンポーネントモデルベースまたはインライン複合モデルの組み合わせによって決まります。このクォータは、このアセットモデルから作成された一意のアセットにも適用されます。	5000	はい
各複合モデルのプロパティの数	複合モデルで許可されるプロパティの最大数。また、タイプ COMPONENT_MODEL のアセットモデルで許可されるプロパティの最大数。	100	はい
アセットモデルのプロパティツリーの深さ	たとえば、測定プロパティ A を使用する変換プロパティ B を使用する変換プロパティ C を持つモデルの深さは、3 です。	10	いいえ

リソース	説明	Quota	調整可能
各階層ツリーのアセットモデルの数	1 つの階層ツリーに含めることができるアセットモデルの最大数。	100	はい
アセットモデルの直接依存プロパティの数	このクォータは、プロパティ式の定義に従って、1 つのプロパティに直接依存できるプロパティの数を制限します。依存プロパティの数は、アセットモデルの直接依存プロパティの数よりも大きくする必要があります。アセットモデルの依存プロパティよりも直接依存するプロパティがある場合は、両方のクォータの引き上げをリクエストします。	20	はい
アセットモデル内の依存プロパティの数	このクォータは、プロパティ式の定義に従って、1 つのプロパティに直接または間接的に依存できるプロパティの数を制限します。	30	いいえ
アセットモデル内の複合モデルの数	1 つのアセットモデルに保持できる複合モデルの最大数。	50	はい

リソース	説明	Quota	調整可能
複合モデルの深さ	インラインモデルとcomponent-model-based複合モデルツリーの最大深度。	2	はい
同じコンポーネントモデルを使用する一意のアセットモデルの数	COMPONENT_MODEL タイプの特定のアセットモデルを直接参照するコンポーネントモデルベースの複合モデルが少なくとも 1 つある一意のアセットモデルの最大数。	20	はい
プロパティ式内のプロパティ変数の数	たとえば、式 `avg(power) + max(temp)` には `temp`および `power`の 2 つのプロパティ変数があります。これは変換の計算結果にも当てはまります。	10	いいえ
プロパティ式内の関数の数	たとえば、式 `avg(power) + max(temp)` には、`avg`および`max` という 2 つの関数があります。	10	いいえ

AWS IoT SiteWise アセットプロパティデータのクォータ

リソース	説明	Quota	調整可能
アセットプロパティデータ API オペレーションのリクエストレート	各 で実行できる 1 秒あたりのアセットプロパティデータ API リクエストの最大数 AWS リージョン AWS アカウント。このクォータは、`GetAssetPropertyValue` や `BatchPutAssetPropertyValue` などの API オペレーションに適用されます。	1,000	はい
各アセットプロパティのデータ品質に対する 1 秒あたりのデータポイントの数	このクォータは、各アセットプロパティのデータ品質について、秒単位で同じタイムスタンプを持つ timestamp-quality-value (TQV) データポイントの最大数に適用されます。各アセットプロパティについて、任意の秒に対して、品質適合、品質不明、および品質不良のデータポイントをこの数まで保存できます。	10	いいえ
の属性以外のアセットプロパティごと	SiteWise Edge ゲートウェイ、 AWS IoT	10	いいえ

リソース	説明	Quota	調整可能
に 1 秒ごとに取り込まれた`BatchPutAssetPropertyValue` エントリの数 AWS リージョン AWS アカウント。	Core ルール、API コールなど、すべてのソース`BatchPutAssetPropertyValue` からの の各属性以外のアセットプロパティのエントリの最大数。		
AWS リージョン の各属性アセットプロパティに対して 1 秒ごとに取り込まれた`BatchPutAssetPropertyValue` エントリの数 AWS アカウント。	このクォータは、SiteWise Edge ゲートウェイ、 AWS IoT Core ルール、API コールなど、すべてのソース`BatchPutAssetPropertyValue` からの の各属性アセットプロパティのエントリに適用されます。	1	いいえ
取り込まれたデータポイントの割合	AWS リージョン ごとに 1 秒あたりに取り込まれるtimestamp-quality-value (TQV) データポイントの最大数 AWS アカウント。	5000	はい

リソース	説明	Quota	調整可能
BatchGetAssetPropertyAggregates のリクエストレート	各 で実行できる 1 秒 AWS リージョン あたりのBatchGetAssetPropertyAggregates リクエストの最大数 AWS アカウント。	200	はい
BatchGetAssetPropertyValue のリクエストレート	各 で実行できる 1 秒 AWS リージョン あたりのBatchGetAssetPropertyValue リクエストの最大数 AWS アカウント。	500	はい
BatchGetAssetPropertyValueHistory のリクエストレート	を実行できる 1 秒あたりのBatchGetAssetPropertyValueHistory リクエストの最大数。	200	はい
内の各 のアセットプロパティごとに 1 秒あたり AWS リージョン に取り込まれたBatchPutAssetPropertyValue エントリの数 AWS アカウント。	このクォータは、SiteWise Edge ゲートウェイ、 AWS IoT Core ルール、API コールなど、すべてのソースBatchPutAssetPropertyValue からの の各アセットプロパティのエントリに適用されます。	10	いいえ

リソース	説明	Quota	調整可能
各アセットプロパティのGetAssetPropertyAggregates リクエストとBatchGetAssetPropertyAggregates エントリクエリのレート	各アセットプロパティの 1 秒あたりのGetAssetPropertyAggregates リクエストとBatchGetAssetPropertyAggregates エントリの最大数 AWS リージョン AWS アカウント。	50	いいえ
各アセットプロパティのGetAssetPropertyValue リクエストとBatchGetAssetPropertyValue エントリクエリのレート	各アセットプロパティの 1 秒あたりのGetAssetPropertyValue リクエストとBatchGetAssetPropertyValue エントリの合計の最大数 AWS リージョン AWS アカウント。	500	いいえ

リソース	説明	Quota	調整可能
各アセットプロパティのGetAssetPropertyValueHistory リクエストとBatchGetAssetPropertyValueHistory エントリクエリのレート	各アセットプロパティの 1 秒あたりのGetAssetPropertyValueHistory リクエストとBatchGetAssetPropertyValueHistory エントリの合計の最大数 AWS リージョン AWS アカウント。	30	いいえ
GetInterpolatedAssetPropertyValues リクエストのレート	各 で実行できる 1 秒 AWS リージョン あたりのGetInterpolatedAssetPropertyValues リクエストの最大数 AWS アカウント。	500	はい
各GetInterpolatedAssetPropertyValues リクエストの結果の数	ページ分割されたGetInterpolatedAssetPropertyValues リクエストに対して返される結果の最大数。	10	[Yes (はい)]

リソース	説明	Quota	調整可能
`GetAssetPropertyValueHistory` および から取得したデータポイントのレート `BatchGetAssetPropertyValueHistory`	`GetAssetPropertyValueHistory` および AWS アカウント 全体で の各 について 1 秒 AWS リージョン ごとに取得されるデータポイントの最大バイトレート (MB/秒) `BatchGetAssetPropertyValueHistory` 。このクォータで評価されるレスポンスペイロードでは、各データポイントの Timestamp-Quality-Value (TQV) フィールドを使用しており、各 API リクエストのバイトサイズは次の 4 KB 単位で四捨五入されます。 1 秒ごとに取得されるTimestamp-quality-value (TQV) データポイントは、データ型によって異なります。 • 整数 – 1 秒あたり最大 500 万 TQV	100	はい

リソース	説明	Quota	調整可能
	• ダブル - 1 秒あたり最大 400 万 TQV • ブール値 - 1 秒あたり最大 600 万 TQV • 文字列 – 各文字列の値のサイズによって異なる。		

SiteWise Edge ゲートウェイのクォータ

リソース	説明	Quota	調整可能
AWS リージョン の各 の SiteWise Edge ゲートウェイの数 AWS アカウント	AWS リージョン の で作成できる SiteWise Edge ゲートウェイの最大数 AWS アカウント。	100	はい
SiteWise Edge ゲートウェイの OPC UA ソースの数	SiteWise Edge ゲートウェイで設定できる OPC UA ソースの最大数。	100	いいえ
SiteWise Edge ゲートウェイ内の送信先の合計数	SiteWise Edge ゲートウェイで設定できる送信先の最大数。	100	いいえ

のクォータ AWS IoT SiteWise Monitor

リソース	説明	Quota	調整可能
AWS リージョン の各のポータルの数 AWS アカウント	AWS リージョン の で作成できる SiteWise Monitor ポー	100	はい

リソース	説明	Quota	調整可能
	タルの最大数 AWS アカウント。		
ポータル内のプロジェクトの数	SiteWise Monitor ポータル内で作成できるプロジェクトの最大数。	100	はい
プロジェクト内のダッシュボードの数	SiteWise Monitor のプロジェクト内で作成できるダッシュボードの最大数。	100	はい
プロジェクト内のルートアセットの数	SiteWise Monitor でプロジェクトに追加できる最上位アセットの最大数。	1	いいえ
ダッシュボード内の視覚化の数	SiteWise Monitor でダッシュボードに追加できるビジュアル要素 (グラフ、グラフ、テーブルなど) の最大数。	10	[Yes (はい)]
各ダッシュボードの視覚化におけるメトリクスの数	SiteWise Monitor のダッシュボードの 1 つの視覚化で表示できるメトリクスまたはデータポイントの最大数。	5	はい

リソース	説明	Quota	調整可能
ダッシュボードの視覚化ごとのしきい値の数	SiteWise Monitor のダッシュボードの視覚化ごとに設定できるしきい値の最大数。	12	いいえ

メタデータの AWS IoT SiteWise 一括インポートとエクスポートのクォータ

リソース	説明	Quota	調整可能
キュー内のメタデータ転送ジョブの数	キュー内の `PENDING` メタデータ転送ジョブの最大数。	10	[Yes (はい)]
メタデータ転送ジョブのインポートファイルのサイズ	インポートされたファイルの最大サイズ (MB)。	100	はい
ジョブ内の AWS IoT SiteWise インポートリソースの数	1 つのジョブ内の AWS IoT SiteWise インポートリソースの最大数。リソースには、アセットとアセットモデルが含まれます。	5000	はい
ジョブ内の AWS IoT SiteWise エクスポートリソースの数	1 つのジョブ内の AWS IoT SiteWise エクスポートリソースの最大数。リソースには、アセットとアセットモデルが含まれます。	5000	はい

データの AWS IoT SiteWise 一括インポートのクォータ

リソース	説明	Quota	調整可能
実行中の一括インポートジョブの数。	が同時に実行できる一括インポートジョブの最大数。	100	はい
CSV ファイルのサイズ	一括インポートジョブの最大 CSV ファイルサイズ (GB 単位) 。	10	いいえ
非圧縮 parquet ファイルのサイズ	一括インポートジョブの非圧縮 parquet ファイルの最大ファイルサイズ (MB)。	256 MB	いいえ
バッファインジェスト用の CSV ファイルのサイズ	一括インポートジョブでバッファインジェストを使用する場合の最大 CSV ファイルサイズ (MB)。	256 MB	いいえ
非圧縮 parquet 行グループのサイズ	非圧縮の Parquet 行グループの最大サイズ。	64 MB	いいえ
Parquet ファイル内の一意の測定値の数	Parquet ファイル内の一意の測定値の最大数。	10000	いいえ
バッファ取り込みの過去と今日のタイムスタンプの間の日数	CreateBulkImportJob API でバッファされた取り込みを使用する場合の、過去と今日のタイムスタンプの間の最大日数。	30	いいえ

リソース	説明	Quota	調整可能
各 `CreateBulkImportJobs` の各 AWS リージョンのリクエストレート AWS アカウント		10	[Yes (はい)]
各 AWS リージョン `ListBulkImportJobs` の各のリクエストレート AWS アカウント		50	はい
各 AWS リージョン `DescribeBulkImportJobs` の各 のリクエストレート AWS アカウント		50	はい

AWS IoT SiteWise Assistant API スロットリングのクォータ

AWS IoT SiteWise Assistant API スロットリング制限のクォータ

リソース	説明	Quota	調整可能
`InvokeAssistant` オペレーションのリクエストレート	で AWS IoT SiteWise `InvokeAssistant` API に対して実行できる 1 分あたりのトランザクション (TPM) の最大数 AWS アカウント。TPM の制限は、サポートされているすべてのリージョンに適用され、	10	いいえ

リソース	説明	Quota	調整可能
	一部のリージョンで調整可能です。		

異常検出のクォータ

AWS IoT SiteWise ネイティブ異常検出のクォータ

リソース	説明	Quota	調整可能
計算モデルの最大数	計算モデルの最大数。	1,000	はい
各計算モデルの入力プロパティの最大数	計算モデルが保持できる入力プロパティの最大数。	80	いいえ
トレーニングデータ内の TQV の最大数 (リサンプリング後)	再サンプリング後のトレーニングデータ内の TQV の最大数。	1,500,000	いいえ
評価データ内の TQV の最大数 (リサンプリング後)	再サンプリング後の評価データ内の TQV の最大数。	1,500,000	いいえ
トレーニングデータの最小タイムスパン。	トレーニングデータの最小時間範囲。詳細については、「最小日付範囲を理解する」を参照してください。	14 日間	いいえ
同時ACTIVEトレーニングジョブの最大数	任意の時点での同時ACTIVEトレーニングジョブの最大数。	25	はい

リソース	説明	Quota	調整可能
ラベルデータファイルの最大サイズ	ラベルデータファイルの最大サイズ。	2 MB	いいえ
ACTIVE 高頻度推論の最大数	任意の時点でACTIVEの高頻度推論の最大数。	250	はい
ACTIVE 低頻度推論の最大数	任意の時点でのACTIVE低頻度推論の最大数。	500	はい
ACTIVE トレーニング期間の最大時間	ACTIVE トレーニングを実行できる最大時間。	24 時間	はい

AWS IoT SiteWise ユーザーガイドのドキュメント履歴

次の表に、このリリースのドキュメントを示します AWS IoT SiteWise。

- API バージョン: 2019-12-02

変更	説明	日付
にネイティブ異常サポートを追加 AWS IoT SiteWise	AWS IoT SiteWise ネイティブ異常検出は、異常な機器動作を検出し、潜在的な障害を特定する産業機器を監視するための機械学習 (ML) サービスです。ネイティブ異常検出を使用すると、予測メンテナンスプログラムを実装し、最適でない機器プロセスを特定できます。	2025 年 7 月 28 日
既存の SQL に Rich SQL 関数と新しい SQL データ型、句、演算子を追加	新しい dataypes、句、演算子のサポートが追加されました。	2025 年 7 月 22 日
ユーザー名とパスワードを使用して外部アプリケーションを EMQX ブローカーに接続する	ユーザー名とパスワードを使用して外部アプリケーションを、新しい MQTT 対応 V3 AWS IoT Greengrass ゲートウェイの作成時にデフォルトでデプロイされる EMQX ブローカーに接続するためのサポートが追加されました。	2025 年 5 月 1 日
SiteWise Edge での MQTT 対応 V3 ゲートウェイのサポート	新機能の追加と非推奨コンテンツの削除 • MQTT 対応 V3 ゲートウェイのサポートが追加されま	2025 年 2 月 26 日

	した。パスフィルターを使用して MQTT トピックをサブスクライブする送信先設定を強化しました。これには、Amazon S3 へのリアルタイムのデータ取り込み AWS IoT SiteWise やバッファされたデータ取り込みが含まれます。 • IoT SiteWise OPC UA コレクターのバージョン 3.0.0 と IoT SiteWise パブリッシャーコンポーネントのバージョン 4.0.0 をリリースしました AWS IoT Greengrass V2。 • 以前のバージョンのセルフホスト SiteWise Edge ゲートウェイの名前を Classic ストリーム、V2 ゲートウェイに変更しました。 • SiteWise Edge ドキュメント AWS IoT Greengrass V1 の への参照を削除しました。これは、 での使用がサポートされなくなったためです AWS IoT SiteWise。	
AWS IoT SiteWise Assistant のサポート	生成 AI を活用した AWS IoT SiteWise アシスタントである Assistant のサポートが追加されました。	2024 年 11 月 18 日

SiteWise Edge APIs の設定可能なセッションタイムアウトを追加	および SiteWise Edge API の AWS OpsHub 非アクティブ期間を管理するための設定可能なセッションタイムアウト設定を追加しました。 APIs	2024 年 10 月 31 日
SiteWise Edge APIs の設定可能なプロキシ設定を追加	SiteWise Edge ゲートウェイの HTTPS プロキシサポートを有効にするための信頼ストア情報の管理を追加しました。	2024 年 10 月 31 日
SiteWise Edge APIs の で CORS を有効にする	SiteWise Edge APIs、安全なクロスドメインウェブアプリケーションアクセスが可能になりました。	2024 年 9 月 30 日
CloudRail および Litmus Edge パートナーデータソースのサポート	パートナーデータソースとしての CloudRail と Litmus Edge の両方のサポートを追加しました。	2024 年 9 月 5 日
Siemens Industrial Edge で SiteWise Edge を実行する一般的な可用性	AWS IoT SiteWise は、Siemens Industrial Edge デバイスで SiteWise Edge を実行する一般提供をサポートするようになりました。	2024 年 7 月 24 日
OPC UA データソースでのタイムスタンプ設定のサポートを追加	AWS IoT SiteWise は、OPC UA データソースのタイムスタンプ設定をサポートするようになりました。	2024 年 7 月 24 日
OPC UA データソースでのデータ型変換のサポートを追加	AWS IoT SiteWise サポートされていない OPC UA データ型のデータ型変換をサポートするようになりました。	2024 年 7 月 24 日

Siemens Industrial Edge での SiteWise Edge のプレビューを実行するためのサポートを追加	AWS IoT SiteWise は、Siemens Industrial Edge デバイスで SiteWise Edge のプレビューの実行をサポートするようになりました。	2023 年 11 月 26 日
ウォーム階層ストレージのサポートを追加	AWS IoT SiteWise は、フルマネージド型のストレージ層であるウォームストレージをサポートするようになりました。これにより、お客様は産業データを安全に保存してアクセスできます。	2023 年 11 月 15 日
ユーザー定義の一意の識別子のサポートを追加	AWS IoT SiteWise は、アセット、アセットモデル、プロパティ、階層のユーザー定義の一意の識別子の使用をサポートするようになりました。	2023 年 11 月 15 日
産業用アセットの多変量異常検出のサポートを追加	AWS IoT SiteWise は、履歴およびリアルタイムの機器データを Amazon Lookout for Equipment と統合することで、産業アセットの多変量異常検出をサポートするようになりました。	2023 年 11 月 15 日
AWS IoT SiteWiseでの時系列データのコスト効率とスケーラブルな取り込みのサポートを追加	AWS IoT SiteWise は、分析ユースケースに必要な時系列データのコスト効率が高くスケーラブルな取り込みをサポートするようになりました。	2023 年 11 月 15 日

一括インポート、エクスポート、更新のサポートを追加	AWS IoT SiteWise は、産業機器メタデータの一括インポート、エクスポート、更新をサポートするようになりました。	2023 年 11 月 15 日
アセットモデルコンポーネントのサポートを追加	AWS IoT SiteWise は、産業のお客様が再利用可能なコンポーネントを作成するのに役立つアセットモデルコンポーネントをサポートするようになりました。	2023 年 11 月 15 日
IoT ダッシュボードアプリケーションのサポートを追加	AWS IoT SiteWise は、運用データを視覚化して操作できるオープンソースのダッシュボードアプリケーションをサポートするようになりました。	2023 年 11 月 15 日
のサービスにリンクされたロールを更新しました AWS IoT SiteWise	AWS IoT SiteWise には新しいサービスにリンクされたロールがあり、 AWS IoT TwinMaker データベースに対してメタデータ検索クエリを実行できます。	2023 年 11 月 6 日
AWS IoT SiteWise データストリームリソースのタグ付けを更新	データストリームリソースのタグ付けに対するサポートを更新。	2022 年 8 月 18 日
SiteWise Edge ゲートウェイの更新	パブリッシャーの設定で、エッジからクラウドに送信されるデータと、データをクラウドに送信する順序を制御できるようになりました。	2022 年 1 月 12 日

AWS IoT SiteWise デモを更新しました	デモを使用して SiteWise Monitor ポータルを作成できるようになりました。	2022 年 1 月 10 日
ストレージ管理を更新しました	保存期間を定義することで、ホット階層にデータを保存する期間を制御できるようになりました。	2021 年 11 月 29 日
データストリーム管理のサポートが追加されました	アセットモデルとアセットを作成する AWS IoT SiteWise 前に、 にデータを取り込むことができるようになりました。	2021 年 11 月 24 日
アセットモデル階層を更新しました	子アセットモデルを複数の親アセットモデルに関連付けることができるようになりました。	2021 年 10 月 28 日
リージョンへの参入	AWS GovCloud (米国西部) AWS IoT SiteWise で起動しました。	2021 年 9 月 29 日
関数を更新しました	次の機能が追加されました。 • メトリクスでは、[aggregation functions] (集計関数) や[temporal functions] (一時関数) でネスト表現を使用することができます。 • 変換では、[pretrigger () function] (pretrigger () function 関数) を使用して、現在の変換コンピューティングのトリガーとなったプロパティ更新前の変数値を取得することができます。	2021 年 8 月 10 日

カスタムメトリクスの時間間隔	メトリクスの時間間隔とオフセットのカスタムに対応しました。	2021 年 8 月 3 日
エッジ AWS IoT SiteWise での の使用	エッジ処理機能の一般提供を開始しました。	2021 年 7 月 29 日
Amazon S3 へのデータエクスポート	AWS IoT SiteWise で Amazon S3 にデータをエクスポートできるようになりました。	2021 年 7 月 27 日
VPC エンドポイント (AWS PrivateLink)	コントロールプレーン API 操作のためのインターフェース VPC エンドポイントの一般提供を開始しました。	2021 年 7 月 15 日
変換	変換で、複数のアセットプロパティ変数を入力できるようになりました。	2021 年 7 月 8 日
timestamp () 関数を更新しました	変換では、`timestamp()` 関数の引数として変数を与えることができるようになりました。	2021年6月16日
アラームの一般提供	アラーム機能は一般公開されています。	2021 年 5 月 27 日
Modbus-TCP プロトコル・アダプター バージョン 2 をリリース	[Modbus-TCP Protocol Adapter connector] (Modbus-TCP プロトコルアダプターコネクター) のバージョン 2 が利用可能です。このリリースでは、ASCII、UTF8、および ISO8859 エンコードされたソース文字列のサポートのサポートが追加されました。	2021 年 5 月 24 日

サービスクォータを更新	GetInterpolatedAssetPropertyValues API に、`GetInterpolatedAssetPropertyValues` リクエストの割合、`GetInterpolatedAssetPropertyValues` リクエストごとの結果数、`GetInterpolatedAssetPropertyValues` の過去開始日から本日までの日数というクォータを追加しました。	2021 年 4 月 29 日
数式を更新しました	次の演算子、関数を追加しました。 • 次の [operators] (演算子) を追加しました: `<`、`>`、`<=`、`>=`、`==`、`!=`、`!`、`and`、`or`、および `not`。 • 次の [comparison function] (比較機能) を追加しました: `neq(x, y)`。 • 次の [string functions] (文字列関数) を追加しました: `join()`、`format()`、および `f''`。	2021 年 4 月 22 日
VPC エンドポイント (AWS PrivateLink)	インターフェイス VPC エンドポイントを作成して、仮想プライベートクラウド (VPC) と AWS IoT SiteWise コントロールプレーン APIs の間にプライベート接続を確立する方法に関する情報を追加しました。	2021 年 3 月 16 日

IAM フェデレーション	SiteWise Monitor のポータル管理者とユーザーは、IAM 認証情報を使って割り当てられたポータルにログインできるようになりました。	2021 年 3 月 16 日
リージョンへの参入	中国 (北京) AWS IoT SiteWise で開始されました。	2021 年 2 月 3 日
IoT SiteWise コネクタのバージョン 10 をリリースしました	IoT SiteWise コネクタのバージョン 10 が使用可能です。本リリースでは、送信元接続が失われ、再確立された場合の処理を改善するよう `StreamManager` を設定しました。このバージョンは、使用可能な `SourceTimestamp` がないときに、`ServerTimestamp` で OPC UA 値も受け付けます。	2021 年 1 月 22 日
日付および時刻関数	AWS IoT SiteWise で日付と時刻の関数がサポートされるようになりました。	2021 年 1 月 21 日
関数構文	AWS IoT SiteWise 関数に Uniform Function Call Syntax (UFCS) を使用できるようになりました。	2021 年 1 月 11 日
Grafana との統合	Grafana ダッシュボードで AWS IoT SiteWise データを視覚化する方法に関する情報を追加しました。	2020 年 12 月 15 日

AWS IoT SiteWise 機能リリース	アラームによるデータのモニタリング、エッジでの産業用データの処理、SiteWise Edge ゲートウェイへの Modbus TCP や Ethernet/IP ソースの使用、デッドバンドによる受信データのフィルタリングなど、さまざまな機能を利用できるようになりました。 • AWS IoT SiteWiseでのアラームの定義、設定、対応に使用できる [Monitoring data with alarms] (アラーム付きモニタリングデータ) セクションを追加しました。 • エッジデバイスで産業用データの処理を設定するために使用できる、[Edge processing] (エッジ処理) セクションを追加しました。 • SiteWise Edge ゲートウェイのソースドキュメントに、「Modbus TCP と Ethernet/IP」セクションを追加しました。 • 受信した産業用データの [source destination] (送信先) をカスタマイズするために使用できる送信先セクションを追加しました。 • 産業用ローカルサーバーから SiteWise Edge ゲートウェイに送信されるデータの頻度や種類を制御するために使用できる「OPC UA	2020 年 12 月 15 日

	フィルタリング」セクションを追加しました。	
AWS IoT SiteWise がカスタマー管理CMKs をサポートするようになりました。	AWS IoT SiteWise は、カスタマー管理CMKs による暗号化をサポートするようになりました。	2020 年 11 月 24 日
IoT SiteWise コネクタのバージョン 8 をリリースしました	IoT SiteWise コネクタのバージョン 8 が使用可能です。このリリースでは、コネクターのネットワーク接続が断続的に発生する場合の安定性を改善しました。	2020 年 11 月 19 日
数式で文字列と条件式を使用する	変換とメトリクスの数式式に文字列と条件関数を使用する方法に関する情報を追加しました。	2020 年 11 月 16 日
AWS IoT Greengrass ストリームマネージャーを使用したデータの取り込み	AWS IoT Greengrass エッジデバイスを使用してローカルデータソースから大量の IoT データを取り込む方法に関する情報を追加しました。	2020 年 9 月 16 日
VPC エンドポイント (AWS PrivateLink)	インターフェイス VPC エンドポイントを作成して、仮想プライベートクラウド (VPC) と AWS IoT SiteWise データ APIs の間にプライベート接続を確立する方法に関する情報を追加しました。	2020 年 9 月 4 日

IoT SiteWiseコネクタのバージョン 7 をリリースしました	IoT SiteWise コネクタのバージョン 7 が使用可能です。このリリースでは、SiteWise Edge ゲートウェイメトリクスに関する問題が修正されています。	2020 年 8 月 14 日
AWS IoT SiteWise コンソールからの IAM Identity Center ユーザーの作成	AWS IoT SiteWise コンソールで IAM Identity Center ユーザーを作成する方法に関する情報を追加しました。新規または既存のポータルにユーザーを割り当てる際に、IAM Identity Center ユーザーを作成することができるようになりました。[Visualizing and sharing wind farm data] (風力発電施設データの視覚化と共有) のチュートリアルを更新しました。この変更により、チュートリアルのステップ数が減少しました。	2020 年 8 月 4 日
SiteWise Edge ゲートウェイのトラブルシューティングを改善	SiteWise Edge ゲートウェイのトラブルシューティング方法と、ソースの OPC UA クライアント証明書をエクスポートする方法に関する追加情報を追加しました。	2020 年 6 月 18 日

コンソールタスクのドキュメント	産業用アセットのモデリング、アセットプロパティデータのクエリ、その他のサービスの操作に関するコンソールタスクのドキュメントが追加されました。次の手順に従って、 AWS IoT SiteWise コンソールでタスクを完了できます。	2020 年 6 月 11 日
エクスポートされたデータチュートリアルの分析	Amazon Athena を使用して、エクスポート機能 AWS CloudFormation テンプレートを使用して Amazon S3 にエクスポートしたアセットデータを分析する方法を学ぶためのチュートリアルを追加しました。	2020 年 5 月 27 日
数式の使用の改善	AWS IoT SiteWise 数式プロパティの動作に関する詳細情報を追加し、フィルタリングされたデータポイントをカウントする方法の例を追加しました。	2020 年 5 月 18 日

IoT SiteWise コネクタのバージョン 6 をリリースしました	IoT SiteWise コネクタのバージョン 6 が使用可能です。このリリースでは、CloudWatch メトリクスのサポートと、新しい OPC UA タグの自動検出が追加されました。これにより、OPC UA ソースのタグが変更されたときに、SiteWise Edge ゲートウェイを再起動する必要がなくなります。このバージョンのコネクタには、ストリームマネージャーと AWS IoT Greengrass Core ソフトウェア v1.10.0 以降が必要です。	2020 年 4 月 29 日

AWS IoT SiteWise 機能リリース	AWS IoT SiteWise 機能リリース。API を使用した SiteWise Edge ゲートウェイの管理、ポータルへのロゴの追加、SiteWise Edge ゲートウェイメトリクスの表示などができるようになりました。 • 新しいデータ値を Amazon S3 バケットにエクスポートするために使用できる テンプレートを含む Amazon S3 へのデータのエクスポートセクションを追加しました。 AWS CloudFormation Amazon S3 • SiteWise Edge ゲートウェイのソースドキュメントを改善し、新しい SiteWise Edge ゲートウェイ API を含む「データソースの設定」セクションを追加しました。 • SiteWise Edge ゲートウェイが公開する CloudWatch のメトリクスを説明する「SiteWise Edge ゲートウェイメトリクス」セクションを追加しました。 • Amazon EC2 で SiteWise Edge ゲートウェイを設定するセクションに、Amazon EC2 インスタンスで SiteWise Edge ゲートウェイの依存関係をすばやく設定するために使用で	2020 年 4 月 29 日

	きる AWS CloudFormation テンプレートを追加しました。 • SiteWise Monitor のポータルの新しい許可機能を説明する [portal service roles] (ポータルサービスロール) のセクションを追加しました。 • ポータルサービスロールとポータルロゴについて「ポータルドキュメント」を更新しました。 • AWS IoT SiteWise リソースのタグ付けセクションを追加しました。 • 新しいダッシュボード定義構造について「ダッシュボードの作成 (CLI) 」セクションを更新しました。 • 「セキュリティ」セクションを追加しました。	
からのデータの取り込み AWS IoT Events	イベントが発生した AWS IoT Events ときに からデータを取り込む方法に関する情報を追加しました。	2020 年 4 月 20 日
SiteWise Monitor チュートリアルでの風力発電施設データの視覚化および共有する	を使用してアセットデータを視覚化および共有する方法を学ぶ AWS IoT SiteWise Monitor ためのチュートリアルを追加しました。	2020 年 3 月 12 日

AWS IoT SiteWise の概念	サービスとその一般的な用語について学習するために使用できる AWS IoT SiteWise 概念の用語集を追加しました。	2020 年 3 月 5 日
AWS IoT Greengrass インストール手順を削除しました	AWS IoT SiteWise ユーザーガイドから AWS IoT Greengrass Core ソフトウェアのインストール手順を削除しました。AWS IoT Greengrass デベロッパーガイドには、Amazon EC2 や Docker などの他のプラットフォーム AWS IoT Greengrass でセットアップするためのデバイスセットアップスクリプトと手順が記載されています。	2020 年 2 月 14 日
AWS IoT Core ルールを使用したデータの取り込みを改善	の使用方法と、MQTT メッセージからデータを取り込むために使用できる AWS IoT SiteWise ルールアクションのトラブルシューティング方法に関する詳細情報を追加しました AWS IoT Core。	2020 年 2 月 14 日
IoT SiteWise コネクタのバージョン 5 をリリースしました	IoT SiteWise コネクタのバージョン 5 が使用可能です。このリリースでは、 AWS IoT Greengrass Core ソフトウェア v1.9.4 との互換性の問題を修正しました。	2020 年 2 月 12 日

IoT SiteWise コネクタのバージョン 4 をリリースしました	IoT SiteWise コネクタのバージョン 4 が使用可能です。このリリースでは、OPC UA サーバーの再接続に関する問題を修正します。	2020 年 2 月 7 日
産業用アセットのモデリングを再編成	「アセットとモデルの更新」セクションを、産業用アセットのモデリング内の複数のトピックに再編成しました。 • アセットおよびモデルの状態 • のデータストリームを管理する AWS IoT SiteWise • 属性値を更新する • アセットの関連付けと関連付け解除 • アセットとモデルを更新する • でアセットとモデルを削除する AWS IoT SiteWise	2020 年 2 月 4 日
AWS IoT モノのチュートリアルからのデータの取り込み	新規または既存の AWS IoT モノのフリートからデータを取り込むルール AWS IoT SiteWise アクションを設定する方法を学ぶためのチュートリアルを追加しました。	2020 年 2 月 4 日
からデータを取得する再構成 AWS IoT SiteWise	「データの取得」セクションを 2 つの最上位セクションに再構成しました。アセットプロパティ値と集計のクエリと、他の AWS サービスとのやり取りです。	2020 年 1 月 21 日

プロパティ値の更新を Amazon DynamoDB チュートリアルにパブリッシュする	プロパティ値通知を使用してアセットデータを DynamoDB に格納する方法を学習するためのチュートリアルが追加されました。	2020 年 1 月 8 日
数式の使用	変換プロパティとメトリクスプロパティで使用できる定数と関数を整理するための数式参照を追加しました。「アセットプロパティ」をプロパティタイプごとの個別のトピックに再編成しました。	2020 年 1 月 7 日
OPC UA ノードフィルターの使用	OPC UA ノードフィルターを使用して、SiteWise Edge ゲートウェイソースを追加する際の SiteWise Edge ゲートウェイのパフォーマンスを高める方法についての情報を追加しました。	2020 年 1 月 3 日
コネクタのアップグレード	新しいコネクタバージョンがリリースされたときに SiteWise Edge ゲートウェイをアップグレードする方法についての情報を追加しました。	2019 年 12 月 30 日
IoT SiteWise コネクタのバージョン 3 をリリースしました	IoT SiteWise コネクタのバージョン 3 が使用可能です。このリリースでは、iot: * アクセス許可の要件が削除されました。	2019 年 12 月 17 日

IoT SiteWise コネクタのバージョン 2 をリリースしました	IoT SiteWise コネクタのバージョン 2 が使用可能です。このリリースでは、複数の OPC UA シークレットリソースのサポートを追加しました。	2019 年 12 月 10 日
ダッシュボードの作成 (AWS CLI)	AWS IoT SiteWise Monitor を使用して でダッシュボードを作成する方法に関する情報を追加しました AWS CLI。	2019 年 12 月 6 日

AWS IoT SiteWise バージョン 2 のリリース	バージョン 2 のプレビューをリリースしました AWS IoT SiteWise。OPC UA、MQTT、HTTP 経由でデータを取り込み、アセット階層でデータをモデリングし、SiteWise Monitor でデータを視覚化できるようになりました。 • アセット、アセットモデル、およびアセット階層の変更について、「アセットモデリング」セクションを書き直しました。 • データインジェストセクションを更新し、 AWS IoT Greengrass コネクタステップと非ゲートウェイデータインジェストセクションを追加しました。 • AWS IoT SiteWise Monitor セクションと、SiteWise Monitor Web アプリケーションの使用方法を示す [separate application guide] (別のアプリケーションガイド) を追加しました。 • 「からのデータのクエリ AWS IoT SiteWise」セクションと「他の AWS サービスとやり取りする」セクションを追加しました。	2019 年 12 月 2 日

	• 更新されたデモ体験に合わせて、「開始方法」セクションを書き直しました。	
AWS IoT SiteWise バージョン 1 のリリース	バージョン 1 の初期プレビューをリリースしました AWS IoT SiteWise。	2019 年 2 月 25 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。